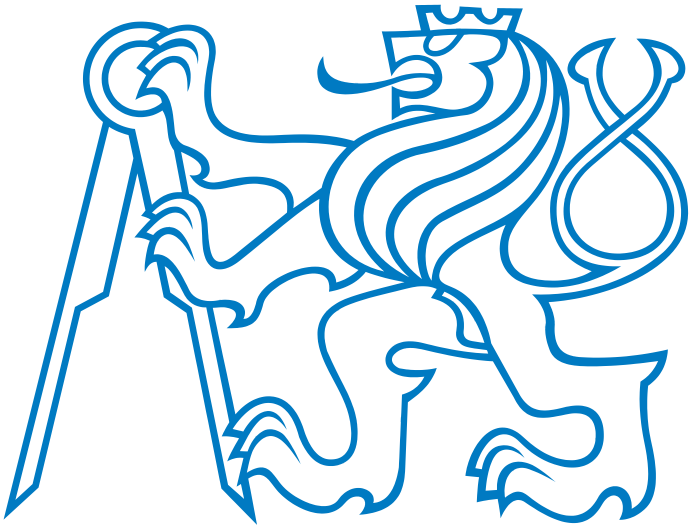
Czech Technical University in Prague
Faculty of Electrical Engineering



Habilitation Thesis

May, 2013

Jan Faigl

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering

MULTI-GOAL PATH PLANNING IN MOBILE ROBOTIC TASKS

Habilitation Thesis

Jan Faigl

Prague, May, 2013

Copyright

Abstract

This habilitation thesis presents advancements in multi-goal path planning for mobile robotic problems arising from inspection, surveillance, exploration or data collecting missions. The main discussed approach is based on self-organizing map (an unsupervised neural network) technique in which the path being found is represented by a sequence of neurons' weights. These weights are adapted to the goals requested to be visited by an unsupervised learning procedure during which the neural network evolves in the problem domain. The simplicity of the adaptation rules provides a great flexibility to address various problems of inspection planning in 2D environments, where the goals can be represented by points or areas of interest. In addition, the same technique is extended to planning in 3D environments with considering kinematic constraints. This approach allows to combine the task planning with a lower level motion planning in a unified way.

Moreover, a problem of path planning considering localization uncertainty of a mobile robot during autonomous navigation is addressed by the same principle. The adaptation procedure utilizes a mathematical model of evolution of the localization uncertainty in a map-and-replay navigation. Based on the found principle of the uncertainty decreasing, the self-organizing neural network finds a path to visit the given set of goals that increases the precision of the goal visits and thus it increases reliability and robustness of the autonomous navigation.

Finally, the multi-goal path planning problem is considered in exploration missions, where it is employed to select the next goals for single or a group of mobile robots operating in an unknown environment. Although, the found plan is followed only for a moment due to re-planning when new information about the environment being explored is collected, the proposed method improves performance of exploration missions in comparison with standard approaches that just consider only an immediate reward by consideration of the visitation of a single goal.

Anotace

Tato habilitační práce představuje výsledky v řešení problémů plánování cest přes více cílů, které vycházejí z úloh inspekce, dohledu, průzkumu a sběru dat mobilním robotem. Hlavní diskutovaný přístup je založen na samo-organizující neuronové síti, ve které je požadovaná cesta reprezentována sekvencí vah neuronů. Během fáze učení sítě jsou tyto váhy adaptovány k požadovaným cílům navštívení podle jednoduchých pravidel samo-organizace. Právě jednoduchost modifikace těchto pravidel poskytuje flexibilitu pro řešení rozličných problémů plánování inspekce pro 2D prostředí, ve kterém mohou být cíle reprezentovány body nebo oblastmi zájmu. Navíc, identická technika adaptace umožňuje rozšíření úlohy na plánování ve 3D prostředí a také zohlednění kinematických omezení mobilního robotu. Tento přístup tak umožňuje kombinovat plánování úloh s plánováním pohybu.

Kromě toho lze stejný princip plánování cesty přes více cílů použít také pro řešení plánovacího problému zohledňujícího zdroje lokalizační nejistoty autonomní navigace mobilního robotu. Při plánování je využito matematického modelu vývoje lokalizační nejistoty v tzv. „map-and-replay" navigaci, ve které je robot nejdříve proveden daným

prostředím a následně může opakovat projetí naučené trasy plně autonomně. Tento přístup využívá identifikovaného principu snižování lokalizační nejistoty v průběhu navigace a neuronová síť poskytuje cestu, která zvyšuje přesnost navštívení cílů. Nalezené cesty tak vedou k vyšší spolehlivosti a robustnosti autonomní navigace v úlohách dohledu, ve kterých je úkolem periodicky navštěvovat zadané cílové oblasti.

V závěru práce je pak problém plánování cesty přes více cílů uvažován v úloze průzkumu, kde je klíčovým problémem zvolení dalšího cíle, ke kterému se robot (nebo skupina robotů) naviguje tak, aby bylo prostředí prozkoumáno co možná nejrychleji. V průběhu průzkumu jsou získávány nové informace o prostředí, a proto je výhodné na základně nových informací opakovaně stanovovat nové cíle. Přestože je v důsledku tohoto přeplánování nalezená cesta přes více cílů sledována pouze směrem k první cíly, je výsledná doba průzkumu výrazně kratší, než v případě přístupů, které uvažují pouze okamžitý očekávaný benefit pouze jediného cíle, ke kterému robot směřuje.

"The human brain is incapable of creating anything which is really complex."
Kolmogorov

To my wife Lenka . . .

Acknowledgments

I would like to express my thanks to my colleagues for a great time during working on the topics presented in this thesis; namely to Tomáš Krajník, Miroslav Kulich, Vojtěch Vonásek and my student Petr Janoušek. I would also like to acknowledge, with all the respect, Libor Přeučil who pushed me to start thinking about establishing own research stream and without his intervention I would probably not write the thesis. In addition, I would also like to acknowledgment Prof. Vladimír Mařík who help me to understand and to see relations from a different perspective. Finally, here, I would like to acknowledgment Prof. Pavel Ripka for his moral support during days full of changes and Prof. Zbyněk Škvor for his encouragement towards initiating my habilitation procedure.

My special thanks also go to Jiří Vokřínek, Petr Benda and Prof. Michal Pěchouček for encouragement and support to submit the habilitation.

Lastly, but not least, I would like to thank to my wife Lenka for her patience, toleration and all the support she gives me at every moment of our joint life.

Contents

List of Appendices

Abbreviations

AGP	Art Gallery Problem
BRIEF	Binary Robust Independent Elementary Features
GLONASS	Global Navigation Satellite System
GPS	Global Positioning System
MTP	Multi-Goal Path Planning Problem
MTSP	Multiple Traveling Salesmen Problem
RRG	Rapidly-exploring Random Graph
PRM	Probabilistic Roadmap Method
RRT	Rapidly-Exploring Random Tree
SLAM	Simultaneous Localization and Mapping
SND	Smooth Nearness Diagram (Navigation)
SOM	Self-Organizing Map
SURF	Speeded Up Robust Features
SURFNav	SURF based Navigation
TSP	Traveling Salesmen Problem
WRP	Watchman Route Problem
UAVs	Unmanned Aerial Vehicles

1 Introduction

Multi-goal path planning for mobile robots is a general problem arising from practical robotic scenarios like inspection, patrolling, surveillance or data collecting tasks where mobile robots are requested to (semi)autonomously visit a set of locations, e.g., to take a sensor measurement. This class of problems can be characterized by a situation where a mobile robot is requested to collect information about some object in a hazardous operational environment as quickly as possible. A basic variant of the *multi-goal path planning problem* (MTP) is a problem to find the shortest path within a model of the robot's working environment such that the robot navigated along the path will visit all the given goals starting from some initial location and returning to the same location.

The complexity of the MTP can be easily seen for a case where the goals can be represented by points with precise positions known in advance. In this case, the MTP can be directly formulated as the well-known *traveling salesman problem* (TSP), which is known to be NP-hard and for which optimal and heuristic approaches have been developed in operational research [45, 5]. The computational complexity and more precisely the real computational requirements are the main issues for a practical deployment of the methods within a robotic mission. Although we can assume the plan can be prepared in advanced prior the deployment, it is not necessarily the case for search and rescue missions where a prompt response is desired. Moreover, it is also worth to remind that prior knowledge about the environment can be more or less reflecting its current state and the planning is typically based on several assumptions that can respect the complexity of the real world only partially. Thus, an ability to reconsider new information about the current situation and to quickly provide a new plan according to the most updated information is a desirable feature of the mission planning system. Therefore, fast approximate approaches are rather preferred for practical deployment than computationally demanding optimal solvers because of the need for quick replanning.

Regarding the context of robotic missions, the basic variant of the MTP represents a fundamental approach to solve the aforementioned robotic tasks. In this problem formulation, we assume the goals are known a priori and the main problem is to determine a sequence of the goals visits; hence, this problem is also called *sequencing part* of more complex tasks, where the problem is also to determine the goals themselves [42]. Such a complex mission can be a problem of collecting information about environment. A feasible approach for this type of inspection tasks is to decouple the problem into two subproblems: 1) a problem of determining the most suitable sensing locations [43, 49, 31]; 2) and the sequencing part formulated as the MTP, which can be eventually solved by a TSP solver. For example, this approach can be considered for robotic arms or vehicles that are requested to perform a set of operations at the desired locations [83, 74]. In particular, problems where all paths between the goals can be pre-computed. Then, the problem can be solved as the TSP approximately using the Chained Lin-Kernighan heuristic [4] or exactly by a branch and bound method [57].

On the other hand, the current progress in robotic technology allows to consider a larger set of particular application scenarios and wider practical deployments in areas that are beyond limits of previous technologies, e.g., enabled by small unmanned aerial vehicles. These novel scenarios are more challenging, as it is usually not sufficient to consider the basic variant of the problem formulation, because the assumed constraints are

not longer realistic and valid. Besides, it is also desirable to provide a better performance (in standard problems) and thus it is necessary to include additional constraints to find a solution that will be feasible in real deployment and provide expected benefits.

An example of such scenarios are problems where the goals (or their most suitable locations) are not known a priori, as they depend on operational constraints and the connecting path itself. For example in the autonomous ship inspection [22, 24], the problem is to cover a complex shape using an unmanned vehicle with a limited sensoric system. The problem can be addressed by sampling-based planning of inspection paths in which a set of candidate goals (from which a part of the ship is covered) are found first. Having the set of candidate sensing locations, the path can be found using all candidates by a solution of the TSP [17] or appropriate locations can be selected by an optimization method combining sensing and travel costs [90], which can provide a better solution if sensing locations are determined independently from the path planning [42]. However, the problem of determining the best sensing locations is a challenging problem itself [54]. The problem can be considered as a variant of the *art gallery problem* or *set cover problem*, which are known to be NP-hard.

In the decoupled approach, it is obvious that for increasing number of locations the planning (the sequencing part of the problem) will be more computationally demanding due to a larger search space. Thus, considering the problem as an integer programming optimization or using traditional branch and bound algorithms will become quickly computationally intractable [82]. In addition, a part of the problem is also determination of goal-to-goal paths that can be demanding as well, especially for a high dimensional configuration space, which is difficult to explicitly represent and a feasible approach is to consider sampling based methods [58].

Moreover, additional constraints can be considered in order to find feasible and realistic plans. Such constraints include sources of uncertainties in determination of goal locations as well as in robot motion and navigation, limitations of sensory systems [75], limited power sources, restricted communication, or changes of the environment, i.e., respecting dynamics of the environment like changes of obstacles, locations of the goals, wind for aerial vehicles [85] or ocean currents for underwater vehicles [79].

The NP-hardness of the sequencing part of the problem (arising from the underlying TSP-like formulation) together with determining (or selecting) appropriate goal locations and finding goal-to-goal paths make the multi-goal path planning problem difficult to address by standard approaches, especially regarding the required computational time. This is also the reason why approximate solutions are considered to be a more practical than demanding optimal solutions, which can quickly become computationally infeasible. Practical needs to solve the discussed problems and limitations of classical operational research approaches steer the robotic research in multi-goal path planning to consider novel approaches developed in *artificial intelligence* domain. Relatively recently developed *soft-computing techniques* provide a great flexibility to consider various operational constraints while still provide effective meta-heuristics to address complex large scale problems. The techniques include genetic algorithms, e.g., considered in the multi-goal planning for blasthole drill in [21], *ant colony optimization* or *neural networks* already applied for the MTP in mobile robot inspection tasks [23, 25].

The planning approach based on the *self-organizing map* (SOM), which can be also considered as an unsupervised neural network, has demonstrated its flexibility to address

inspection planning problems including variants with discrete [32] and continuous sensing [26] models. According to these results, this planning approach represents a promising technique and its flexibility provides a ground for considering additional constraints and more general problem variants. A contribution towards discovering the advantages in the multi-goal path planning represents the main approach discussed in this thesis.

1.1 Contribution of the Habilitation Thesis

The aim of this habilitation thesis is to provide an overview and insights to the author's work on the topic of multi-goal path planning in mobile robotic tasks, which includes inspection, surveillance or patrolling planning and exploration missions. The main results achieved in this field can be summarized in two frameworks:

1. A unifying multi-goal path planning framework for mobile robot inspection planning in 2D environments.
2. Multi-goal path planning framework considering sources of localization uncertainties providing a more reliable autonomous navigation.

Besides, contributions to the state-of-the-art in the related fields have been achieved during the effort towards these two main results. Moreover, additional developments have been made towards further extensions and generalization for a more general problem formulations also including consideration of additional constraints. However, these achieved results can be considered as initial or preliminary, and therefore, they are not denoted as a new framework yet. A significance of the contributions is supported by already published articles and papers and thus the thesis has a form of the collection of the selected articles and papers accompanied by a commentary describing the main ideas and results.

The commentary part is organized as follows. The core of the main results is based on a self-organizing map technique for the TSP, which has been extended to provide a better performance in multi-goal path planning problems. A brief overview of the proposed approach and the related state-of-the-art methods are presented in Section 2; however, the progress beyond the previous work supported by the publications [27, 28, 29] is presented in Section 2.1. In Section 3, a unifying framework for inspection planning for point and polygonal goals in 2D environments [36] is presented. Section 4 is dedicated to the planning of inspection tasks in 3D environment, which is enabled by supporting structures for speeding up visibility queries in 3D [46]. A problem of multi-goal path planning for autonomous surveillance or patrolling missions is introduced in Section 5. Here, the problem is augmented by consideration of precision of the goals' visits during an autonomous navigation [52]; thus, the problem is to plan a path such that the mobile robot will periodically visit the given set of goals [30, 34]. Finally, the multi-goal path planning in mobile robot exploration of unknown environment is presented in Section 6. The results indicate that considering the problem of selection of new goals as the multi-goal path planning problem can improve the performance of the exploration significantly [56, 35].

2 Self-Organizing Maps for Multi-Goal Path Planning

Self-organizing maps (SOM), also known as Kohonen's unsupervised neural networks, can be considered as a mechanism providing a conversion of nonlinear statistical relationships between high-dimensional data into a simpler (usually two dimensional) lattice. SOM has been widely adopted by many researchers in various fields [50], and it has also been applied to combinatorial problems, in particular to the *traveling salesman problem* (TSP) by Angéniol [2] and Fort [37] in 1988. Since that, many SOM based approaches for the TSP have been proposed and extensive overviews can be found in [15, 16, 25]. Although a description of the adaptation procedure can be found in literature, it is presented here to improve readability of this text and to provide a reader the main principle used for the developed planning approaches.

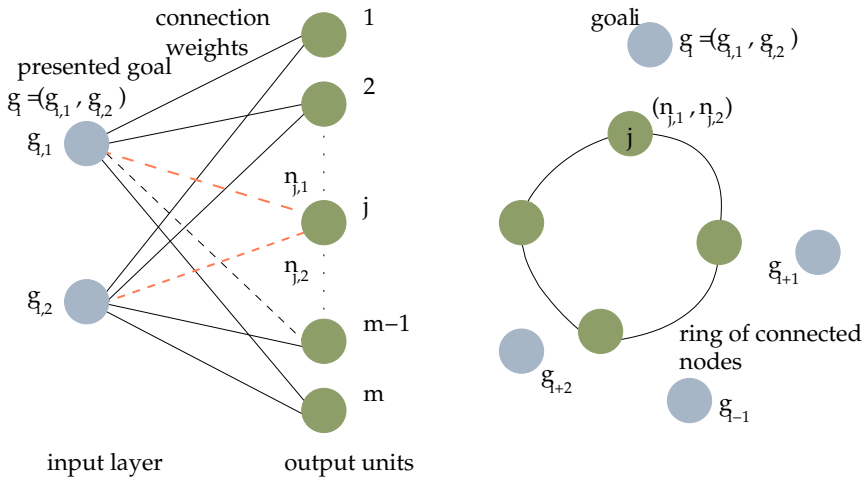


Figure 1: A schema of the two-layered neural network and associated geometric representation.

SOM for the TSP in a plane is a two-layered neural network. The network contains two dimensional input vector and an array of output units that are organized into a uni-dimensional structure. An input vector represents coordinates of a point goal and connections' weights (between the input and output units) represent coordinates of the output units. Connections' weights can be considered as nodes representing a path, which provides direct geometric interpretation of neurons' weights. So, the nodes form a ring in the plane because of the uni-dimensional structure of the output layer, see Fig. 1.

The network learning process is an iterative stochastic procedure in which goals are presented to the network in a random order. The procedure basically consists of two phases: (1) selection of winner node to the presented goal; (2) adaptation of the winner and its neighbouring nodes towards the goal. Although many SOM based approaches for the TSP have been proposed, the approach proposed by authors of [80] represents a straightforward algorithm, which also provides good results for instances of the TSP from the TSPLIB [72] (used for benchmarking TSP solvers). That is why a slightly rephrased (to improve clarity and readability) learning procedure [80] is depicted in Algorithm 1 to show the main steps of the self-organizing adaptation.

The winner node is selected according to $\nu^\star = \mathrm{argmin}_\nu |(g, \nu|)$, where $|(.,.)|$ denotes the

Algorithm 1: SOM Procedure for the TSP

Input: $\boldsymbol{G} = \{g_1, \ldots, g_n\}$ - set of goals
Input: (d, σ, μ, α) - parameters of SOM
Input: δ – maximal allowable error
Input: i_{max} – maximal number of adaptation steps
Output: $(\nu_1, \ldots, \nu_m)$ - sequence of node weights representing the city tour

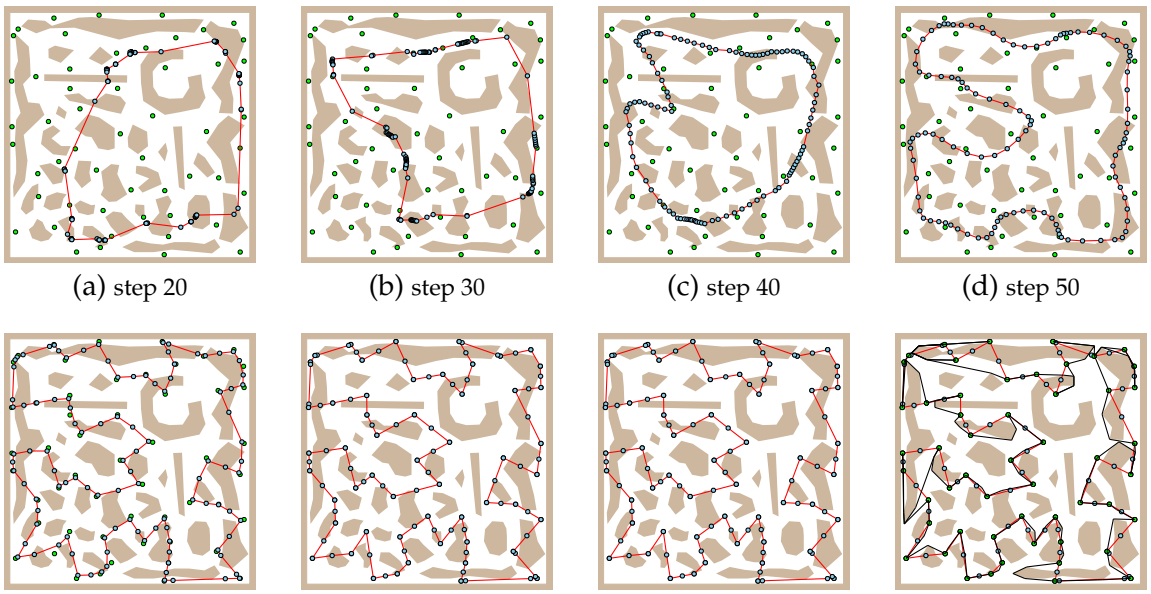
```
init(ν_1, ..., ν_m)                                  // set of neurons weights
i ← 0                                                // the adaptation step counter
repeat
    error ← 0
    I ← ∅                                            // a set of inhibited nodes
    Π(G) ← a random permutation of the goals
    foreach g ∈ Π(G) do
        ν* ← argmin_{ν∈N, ν∉I}(ν, g)|,               // select winner node to g
        error ← max{error, |ν*, g|}
        adapt(ν*, g)
        I ← I ∪ {ν*}                                 // inhibit winner node
    σ ← (1 − α)σ                                     // decrease the gain
    i ← i + 1                                        // increment the step counter
until error ≤ δ or i ≥ i_max
```

Euclidean distance between the goal g and the node ν for the Euclidean TSP. The adaptation function (`adapt`) moves the winner node and its neighbouring nodes towards the presented goal g according to the rule $\nu'_j = \nu_j + \mu f(\sigma, d)(g - \nu_j)$, where μ is the fractional learning rate. The movement of the nodes is defined by the neighbouring function $f(\sigma, d) = exp(-d^2/\sigma^2)$ for $d < 0.2m$ and $f(\sigma, d) = 0$ otherwise, where σ is the learning gain parameter, d is the cardinal distance measured along the ring (in the number of nodes) and m is the number of nodes in the ring. Authors of [80] recommend the initial values of learning and decreasing rates and the learning gain: $\mu = 0.6$, $\alpha = 0.1$ and $\sigma_0 = 0.06 + 12.41n$, respectively.

The algorithm is terminated after a finite number of adaptation steps (i_{max} steps at maximum); however, a stability of the learning rule has been shown in [88] using the joint spectral radius. Moreover, the inhibition of the winners guarantees that each goal has associated a distinct winner; thus, a sequence of all goals visits can be obtained by traversing the ring at the end of each learning epoch.

Regarding the multi-goal path planning problem (MTP), the main difference of regular SOM based approaches for the TSP is that the MTP stands in finding a cost-effective path (the shortest one) visiting a given set of goals in an environment with obstacles. Thus, it is necessary to deal with obstacles in the MTP while SOM for the TSP is usually considered in a plane, where distances between neurons' weights and presented goals are simply computed as Euclidean distances. Although a naïve approach based on using the Euclidean distance provides a solution of the MTP, the final path is very poor. An example of such a learning process is visualized in Fig. 2.

The visualization of the learning process motivates to consider the length of the shortest path from a node ν towards the currently presented goal g as the distance metric for



(a) step 20 (b) step 30 (c) step 40 (d) step 50

(e) step 60 (f) step 70 (g) step 80 (end of the adaptation) (h) final path (in black)

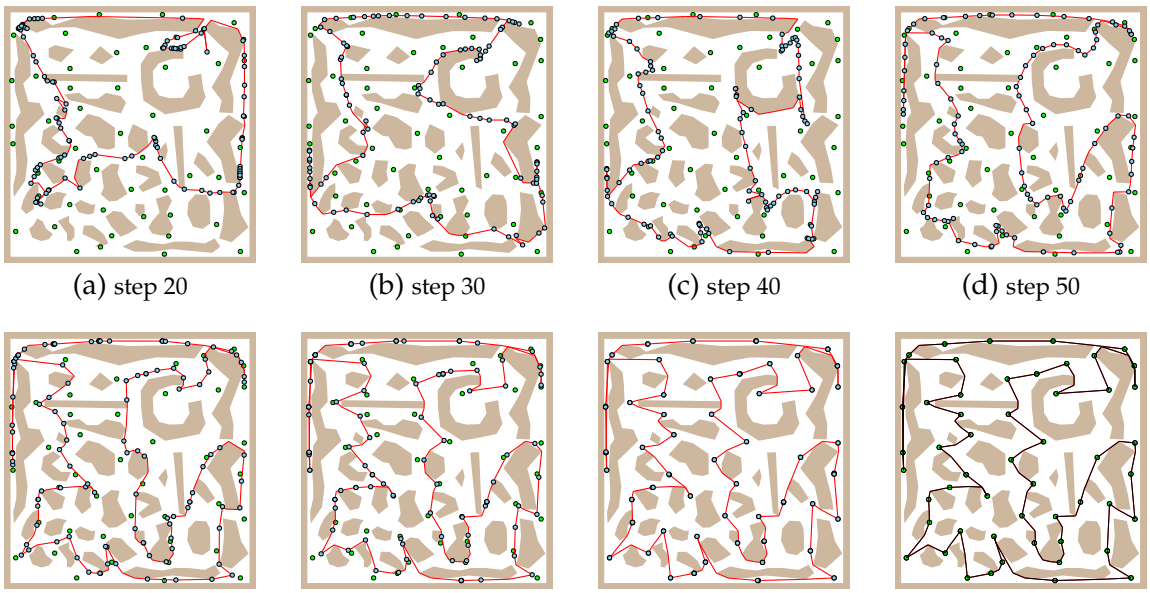
Figure 2: An evolution of SOM for the TSP using Euclidean distance. The nodes are represented as blue disks and the goals (cities) as the green disks. For a better visualization of the evolution, the nodes are connected by a red straight line segments, which forms the ring of nodes evolving in $\mathbf{R}^2$. The final path is determined from the sequence obtained by traversing the final ring, where the winners (goals) are connected by the shortest path among obstacles.

the winner selection instead of pure Euclidean distance. Then, the winner node can be moved along the path towards the particular goal instead of a simple update of the neuron weights. Even though this extension seems to be a straightforward and obvious, a particular implementation depends on the determination of the shortest path among obstacles, which is much more time consuming, e.g., using a visibility graph [25], than the computation of the pure Euclidean distance $|(\nu, g)|$.

It was the computational cost, which prevented to use SOM for the multi-goal path planning, as it has been noted by several authors in literature. However, recently, it has been shown that a simple and computationally effective approximation of the shortest path seems to be sufficient for convergence of the adaptation schema [32]. The approximation is based on a ray-shooting technique combined with the walking in triangulation that is considered in the supporting convex partitioning of the polygonal representation of the robot working environment. Using this technique, a shortest path query is typically answered in units of micro seconds using a standard single core cpu (e.g., running at 2 GHz) and thus the approximation significantly reduces the required computational time and allows to use the SOM adaptation technique for the multi-goal path planning.

The main idea of the approximation employed in the adaptation is that the approximation becomes a more precise as the nodes are closer to the goals. Thus, at the first steps, the nodes are relatively far from the presented goals, and therefore, a rough estimation of the node–goal distance is sufficient to select an appropriate winner to the particular goal.

Later, the winners are closer to the particular goals; hence, in most of the cases the shortest path is just a straight line segment connecting the goal and its winner node.



(a) step 20 (b) step 30 (c) step 40 (d) step 50

(e) step 60 (f) step 70 (g) step 80 (end of the adaptation) (h) final path (in black)

Figure 3: A visualization of the SOM evolution using approximate shortest path. The nodes are represented as blue disks and the goals (cities) as the green disks. For a better visualization of the evolution, the nodes are connected by a red straight line segments, which forms the ring of nodes evolving in $\mathcal{W}$.

The principle of the node–goal shortest path approximation can also be used for approximation of the shortest path between two nodes (details can be found in [25]), which allows to visualize the path represented by the ring during the evolution of the self-organizing map. An example of the visualization is shown in Fig. 3. Notice, it is not necessary to determine the shortest path between two nodes during the learning process; however, such a visualization provides an insight to the SOM evolution in the polygonal domain $\mathcal{W}$, which in fact is the main source of the inspiration for a further extension of the approach to more general problems.

2.1 Progress beyond the previous work

The above described approximation of the shortest path has been introduced in [25, 32]. Therein, it is combined with relatively simple and straightforward SOM adaptation rules proposed by Samerkae et al. in [80] and later used in multiple traveling salesmen [81] and vehicle routing problems [64]. Although the achieved results in the multi-goal path planning problems with point goals are competitive to the heuristic approach GENIUS [32], according to [15] the co-adaptive neural network provides better results in the Euclidean TSP from the TSPLIB. Moreover, the so-called *Co-adaptive net* is also less computationally demanding in these Euclidean problems, which is mainly caused by a less number of the node–goal distance queries needed in the adaptation. Hence, it can be expected that the

adaptation schema of the *Co-adaptive net* can significantly improve performance of the SOM based solution of the MTP because the most computationally demanding parts are just the distance queries.

Beside the *Co-adaptive net* algorithm proposed in 2003, additional modifications of the SOM adaptation rules have been proposed by several authors. For example, a convex hull property has been studied in [13, 60, 93], geometrical properties of the ring in [6, 7], initialization of neurons' weights in [8] or even neighborhood functions for decreasing topological defects have been discussed in [66]. The aim of this effort is to improve solution quality, which was partially achieved by novel modifications and extension of the main adaptation principle proposed by Kohonen. On the other hand, a different direction of the research focuses on simplifying the adaptation rules and also reduction of the required parameters that have to be hand tuned. In [89], the original Kohonen's exponential rules are studied, and the authors proposed simplified rules, which were later used and extended in [94], where the authors proposed a set of particular parameters providing fast convergence without affecting the quality of solutions.

The main disadvantage of the aforementioned variants is that they provide a particular improvement considering different aspect of the SOM evolution. A particular reduction of the computational cost leads to a bit worse solution, and therefore, additional modifications are proposed to "compensate" this loss. In addition, it also happened that a particular approach promising performance improvements is hard to replicate due to missing details and specific values of all parameters in the original paper. In this sense, the most elaborating paper about the performance of various SOM for the TSP is the work of Cochrane and Beasley [15]. The authors consider the most relevant approaches (to the date of that publication) and compare them. They also provide a study of particular parameters of the SOM adaptation to the quality of solution and required computational time. Finally, a comparison of their Co-adaptive net algorithm, which can be considered as the most complex SOM based approach for the TSP, with heuristic approaches from the operational research is presented. In particular, Christofides heuristic algorithm (providing approximation ratio 3/2 of the optimum) and Helsgaun's efficient implementation [45] of the Lin-Kernighan heuristic [61] are considered. Based on the comparison, the authors note that SOM provides relatively poor performance regarding the heuristics. Their Co-adaptive net provides competitive results to the Christofides heuristic in several instances of the TSP, but the Lin-Kernighan heuristic provides results in one or two orders of magnitude better while the computational cost is competitive to the SOM approach.

Regarding the performance of SOM, it is worth to mention that researchers studying SOM advocate that the heuristics for the TSP have more than 20 years longer history than SOM and for example one of the most powerful LK-heuristic was proposed in 1973 [61] while the efficient implementation has been proposed relatively recently in 2000 [45]. On the other hand, in [25, 26], it has been shown that SOM provides additional benefits in routing problems where geometric or conceptual properties are involved, e.g., problems that Shermer called *hybrid visibility problems* [77], which have direct relation to inspection and exploration tasks studied in mobile robotics [42]. Despite to these comments, it is left to the reader to consider the presented methods are suitable techniques for the discussed class of problems.

2.1.1 Performance improvement of SOM for the multi-goal path planning

The recent advancements of SOM for multi-goal path planning is a source of encouragement for additional investigation of SOM performance using a more complex adaptation rules and improved adaptation schemata. The aforementioned variants and simplified adaption algorithms have been studied and compared in:

> [27] – Faigl, J.: *On the Performance of Self-organizing Maps for the Non-Euclidean Traveling Salesman Problem in the Polygonal Domain*. Information Sciences. 2011, vol. 181, no. 19, p. 4214-4229. ISSN 0020-0255. IF=2.833. Authorship 100%. The paper is attached on page 41.

In this work, the previous approaches are combined and discussed. Based on the performance evaluation of the approaches, variants of the winner selection procedure, adaptation rules, parameters and initialization are proposed. The variants are considered in two adaptation schemata, the Somhom's approach [80] and the complex Co-adaptive net [15]. Contrary to previous comparisons of SOM approaches for the TSP, where only Euclidean problems are considered, here, the performance of SOM is studied in multi-goal path planning problems.

The studied problems are organized into three sets according to the number of goals n: small problems n <50, middle problems n <200, and large problems n ≥200. The achieved results can be considered as more than encouraging. The quality of found solutions by the proposed modified SOM adaptation schemata is improved and also the computational cost is significantly decreased. In average, the found solutions are about units of percents worse than the optimum while the computational requirements are competitive to the Chained Lin-Kernighan algorithm [4] available in the CONCORDE framework [3]. Moreover, the best found solutions (considering several trials) are close to the average solutions, which provides optimistic expectations about the methods' performance in other problems.

Regarding the initial results of SOM in the multi-goal path planning presented in [25], the studied modifications of the adaptation rules lead to the adaptation procedure that is up to two orders of magnitude faster, while the quality of solutions is almost preserved (it is about tenths of the percentage points worse). The results also provide a ground for reconsideration of SOM performance in comparison to other approaches. SOM in the MTP provides better results (according to the optimal solutions of the TSP) than SOM in the Euclidean TSP; thus, it seems the gab between heuristics from operational research and SOM approaches is smaller for the TSP, where distances between the goals are not pure Euclidean and paths among obstacles have to be taken into account.

The optimistic statement about SOM performance is supported by results for the large problems [27]. For these problems, the computational cost of the SOM adaptation is in several cases similar or lower than for the LK heuristic. In both approaches, there is an initial phase where all goal–goal paths are determined, which needs similar computational time to find the solution itself. However, in the SOM approach, it is not necessary to determine all the goal–goal paths as approximate shortest path can be used. Only paths between all vertices of the polygonal map are necessary to support the approximation. Hence, for problem instances with many goals representing a dense sensing locations, the SOM approach can be even a less computationally demanding than the preparation phase for a combinatorial heuristic. This observation provides interesting insight that combin-

ing SOM with a simple approximation a relatively high quality sequence of goals' visits can be found while it is not necessary to determine the actual shortest paths between the goals, which are required in the pure combinatorial approaches.

2.1.2 Improvement of SOM based inspection planning

The encouraging results of the improved SOM adaption schemata [27] for the TSP with the point goals provide a ground for a further improvement of the inspection planning also considering continuous sensing, a.k.a the watchman route problem, introduced in [25]. Using the new adaption rules and novel initialization of the neurons' weights lead to a faster convergence of the network and reduced computational burden without significant influence to the solution quality. These new results are presented in:

> [28] – Faigl, J. - Přeučil, L.: *Inspection Planning in the Polygonal Domain by Self-Organizing Map*. Applied Soft Computing. 2011, vol. 11, no. 8, p. 5028-5041. ISSN 1568-4946. IF=2.612. Authorship 90%. The paper is attached on page 57.

2.1.3 A Parameter less SOM adaptation schema

The further work on SOM for the TSP has been not only in improving the performance, but also in the simplification of the adaptation procedure itself. Inspired by recent adaptation rules [94] and results achieved in [27, 28] the most suitable values of the initial learning gain and rate have been identified. The rules make the adaptation procedure almost independent to the selection of these parameters in the considered problems; however, the quality of solution and also the required computational time depends on the number of neurons, which depends on the number of goals.

In literature, it is recommended to set the number of neurons to 2-3 times more than the number of goals, which also corresponds with the formula experimentally established by Samerkae et al. in [80]. A high number of neurons does not provide significant improvements of the solution quality and it only increases the computational burden. On the other hand, a lower number of neurons provides a faster selection of the winner as less number of node–goal distance queries have to be resolved. A less number of neurons than $2n$, where n is the number of goals, is not recommended because of convergence of the network. The neurons have to spread among the problem domain and it is also desirable to select a unique winner for each goal during the learning epoch, even though the inhibition mechanism is not necessary.

In [70], the author proposes a mechanism for dynamic determination of neurons during the learning. The approach is based on two heuristics. The first heuristic is a deletion of neurons, which are not winners so often. On the other hand, the second heuristic is basically responsible for creating new neurons. It is based on consideration of a segment of the ring during the selection of the winner. So, instead of node–goal distance, a shortest segment (defined by the two neighbouring nodes of the ring) to the particular goal presented to the network is determined. If the projection of the goal to the segment is a point inside the segment (excluding the endpoints) a new node is created and it becomes the winner that is then attracted towards the presented goals. The idea is visualized in Fig. 4a.

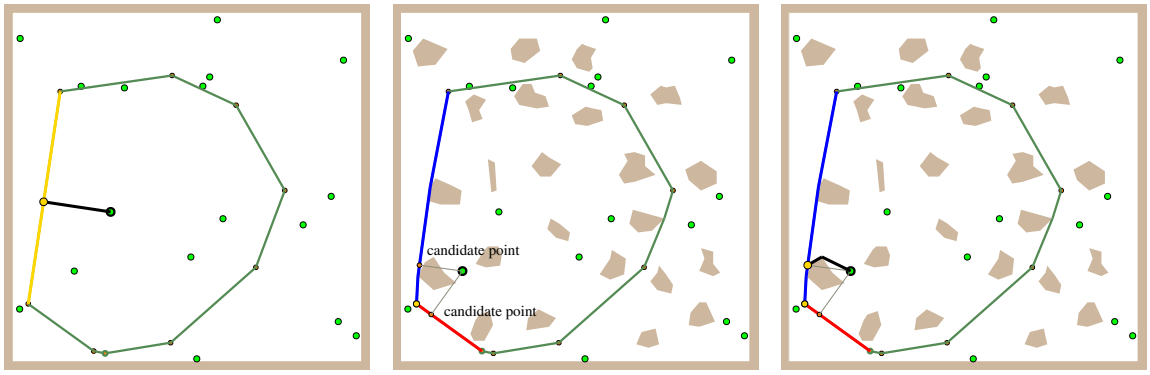
Regarding the adaptation of SOM in the polygonal domain, there is another reason to consider more neurons, it is the approximation of the shortest path. For the case of

solving the watchman route problem, the path represented by the ring is used to determine the actual coverage from the ring. A path represented by a ring consisting of more nodes (e.g., that are equidistantly placed along the path) is a more precise than a path where the nodes are relatively far from each other. The algorithm for the approximation of the shortest path provides exact shortest path if the endpoints of the path are placed at the polygon's vertices, because these paths are precomputed in advance. Besides, such a query is answered instantaneously. Therefore, it is desirable to have neurons (beside the winners) close to map vertices to support the approximation.

Based on this background, the approach [70], and adaptation rules [94] a new adaption schema is proposed in:

> [29] – Faigl, J. - Přeučil, L.: *Self-Organizing Map for the Multi-Goal Path Planning with Polygonal Goals*. In Proceedings of Artificial Neural Networks and Machine Learning. Heidelberg: Springer, 2011, p. 85-92. ISBN 978-3-642-21734-0. Authorship 90%. The paper is attached on page 71.

Determination of the shortest path from a point to a segment in the polygonal domain is computationally demanding, and therefore, an approximate algorithm is rather used.

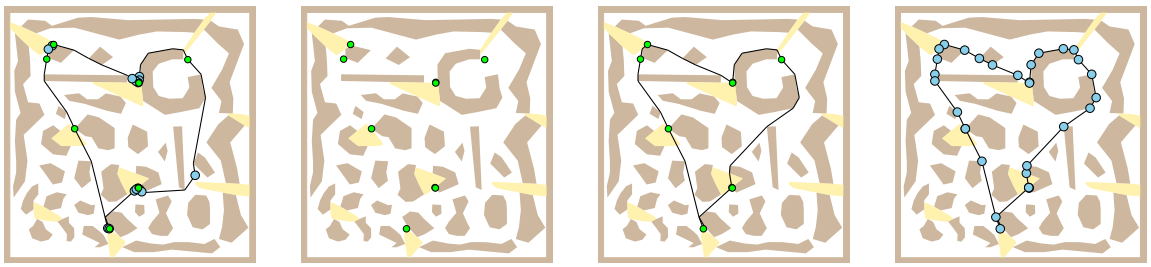


(a) point on the closest segment to the goal (b) candidate points on the ring's segment (c) a new winner node on the ring's segment

Figure 4: A visualization of the approximation of the shortest point–segment path.

First, the closest segment to the particular goal is found considering Euclidean distance, which may provide a point on the segment as a result. Then, approximate point–goal path is determined and if such a path is shorter than a path from an endpoint of the segment to the goal, the path is considered as the shortest path of the goal to the segment. Otherwise, a shorter path from the endpoints is used. The concept of the approximation is visualized in Fig. 4.

Having this approximation, a parameter less adaption schema is proposed as follows. The initial number of neurons is set to $2n$, where n is the number of goals. Then, new neurons are created as a result of the shortest path to the segment. Besides, a new neuron is also created in the case the closest neuron to the goal has been already selected as a winner in the current epoch. In this case, the new neuron has the identical weights (i.e., the 2D coordinates of the node) and it is adapted towards the goal as a regular winner together with its neighbouring nodes.



(a) a path represented by the old ring (b) preserved winners of the old ring (c) connected winners (approx. shortest path) (d) a path represented by the new ring

Figure 5: Ring regeneration preserving winners of the current learning epoch and introduction of new neurons at the positions of the map vertices as vertices of the path that is represented by the ring.

A ring regeneration mechanism is introduced to reduce the number of neurons and to decrease the computational burden. At the end of each learning epoch, a new ring is created as a path in $\mathcal{W}$ using only the winner nodes of the current epoch, i.e., all other nodes are removed. The path is a sequence of straight line segments $(s_1, \ldots, s_r)$, which avoid the obstacles. Additional nodes are created from the endpoints of each segment s_i that do not correspond to the winners, i.e., nodes corresponding to the path's vertices. The ring regeneration is visualized in Fig. 5. After the ring is regenerated, the adaptation parameters are updated and if a stable solution has not been found the adaptation continues in a new learning epoch.

This novel parameter less adaptation schema has been introduced in [29]. In fact, it is based on a bit more complex segment–segment approximation, which provides a generalization for polygonal goals. However, the main contribution presented in the paper is the ring regeneration and the parameter less adaptation schema. The generalization of the SOM based multi-goal path planning for point, segment, as well as polygonal goals has been presented in [33], where several adaptation strategies are proposed. A more elaborating description of this approach is in [36] and the particular commentary is presented in the next section.

3 Unifying Multi-Goal Path Planning Approach

The multi-goal path planning problems with point goals represent a class of problems, where it is required to visit particular locations to perform some action, e.g., precise operation by means of a robotic arm. On the other hand, a mobile robot operating in a large environment will unlikely visit the particular point location precisely using a global navigation technique due to limited precision (i.e., based on satellite navigation systems like, GPS, GLONASS, or Galileo). The robot will more likely reach the location within some distance according to the precision of the navigation. Then, another (a more precise) local navigation method can be used, e.g., based on visual servoing technique.

Moreover, there is a large class of problems in which it is sufficient to just reach a vicinity of the goal location. For example, in autonomous data collecting scenarios, data can be read from the sensor within the communication range providing a reliable connection

with the sensors; thus, it is not necessary to reach the exact position of the sensor [86]. This is also the case, for visual sensing scenarios, where it is sufficient to see the particular object of interest from the distance providing an adequate level of details. Therefore, a more general variant of the multi-goal path planning problem with polygonal goals can be rather considered to find a cost effective solution[1].

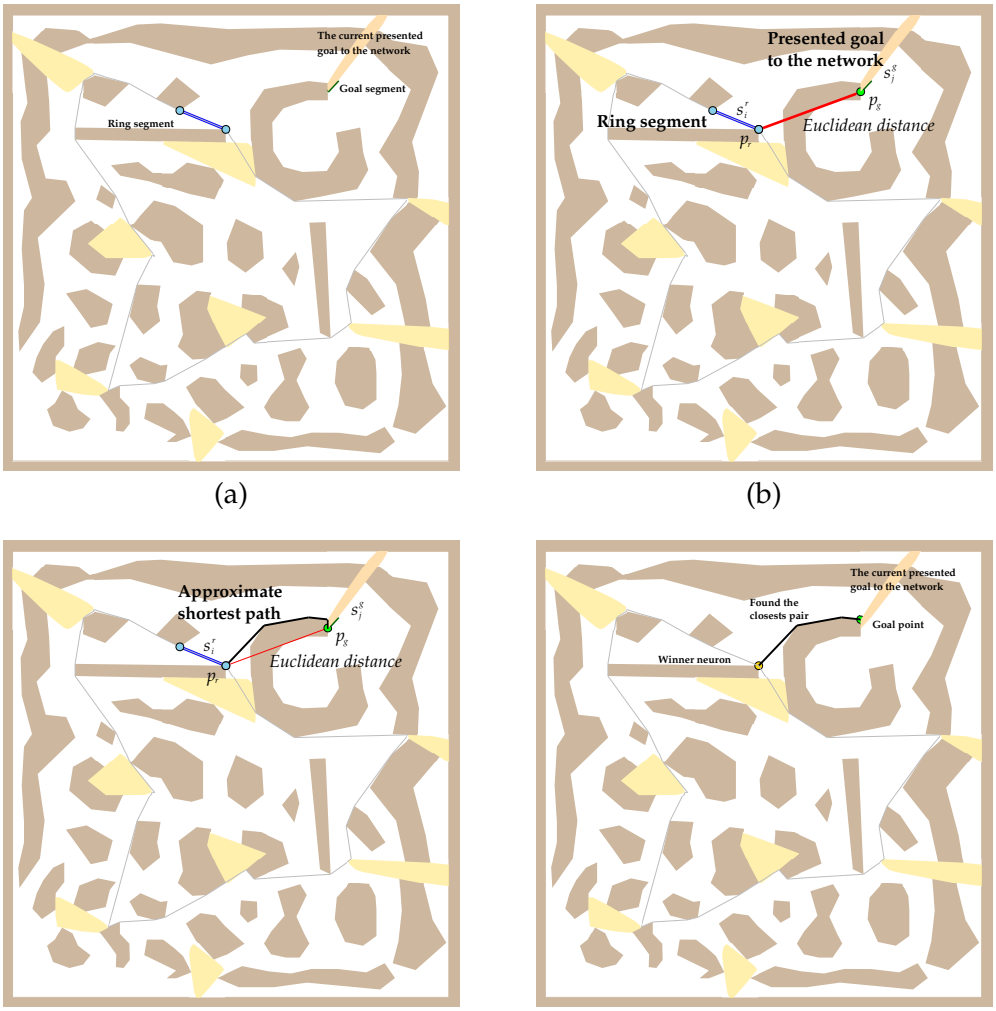


Figure 6: Principle of approximate shortest path between two segments: (**a**) a segment of the path represented by a ring and a segment forming a boundary of the polygonal goal; (**b**) the shortest connection of the segments without considering obstacles; (**c**) the approximate shortest path between two-points determined from the shortest connection of the segments; (**d**) the final approximation of the shortest path between two segments with a point on the particular segment.

In literature, there can be found several problem formulations and particular methods to address them (even optimal algorithm); however, it is worth to mention that these

[1]This does not mean the problem formulation with the point goals should be abandoned. A solution of such problems can be considered as a global plan providing a solution of complex missions, which, in fact, is one of the motivations for this problem formulation.

algorithms are only for restricted problem variants. For example goals form a disjoint set of convex polygons attached to a simple polygon in the *safari route problem* [68], which can be solved in $O(n^3)$ [84]. If the route enter to the convex goal is not allowed, the problem is called the *zoo-keeper problem*, which can be solved in $O(n \log n)$ for a given starting point and the full shortest path map [11]. However, both problems are NP-hard in general.

Here, it should also be noted that for problems with disjoint convex goals that are relatively far from each other, a sequence of the goals' visits can be found as a solution of the TSP for the centroids of the goals. Then, having the sequence of polygonal goals the problem of finding the path connecting them can be formulated as the *touring polygons problem* if the start and end points are given. In [18], a polynomial algorithm is proposed for this type of problems where the convex goals are inside a simple polygon. However, again, the *touring polygons problem* is NP-hard in general.

The point–point shortest path approximation can be considered as an enabling technique to use SOM for the TSP as a multi-goal path planner. On a similar base, approximation of the shortest path between two segments can be considered as an additional technique allowing to solve more general problems (e.g., inspection planning to visit a given set of regions) using the identical adaptation procedure for the TSP with point goals. The main principle of the approximation is depicted in Fig. 6.

The approximation is based on the point–point approximation. First, the shortest path between two segments is determined without considering obstacles; so, just a shortest straight line segment connecting the two segments is determined. A projection point on each segment can be determined as a result of this query. These points are then used for point–point shortest paths among obstacles connecting the points themselves or a combination of the point and an endpoint of the particular segment. The shortest variant is selected as the final approximation of the segment–segment shortest path.

Although the approximation is quite simple, its combination with the SOM based evolution of the ring of nodes in a polygonal map is powerful enough to enable a solution of a more general multi-goal path planning problems with polygonal goals, which is described in:

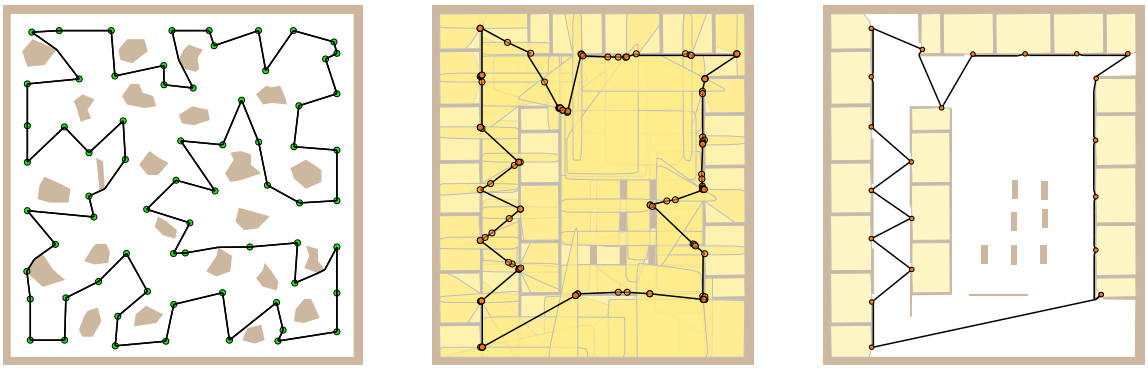
> [36] – Faigl J. - Vonásek V. - Přeučil L.: *Visiting Convex Regions in a Polygonal Map*. Robotics and Autonomous Systems, 2013, `http://dx.doi.org/10.1016/j.robot.2012.08.013`, (in press), IF=1.056., Authorship 80%. The paper is attached on page 79.

In this paper, the multi-goal path planning problem is formulated as follows: find a shortest path visiting a given set of polygonal goals (possibly overlapping each other) in a polygonal map $\mathcal{W}$. In particular, the main presented results are for convex polygonal goals; however, the algorithm is general to also address the problems with non-convex goals. The convexity of the polygonal goals is mainly considered because of related covering problems, where visual sensing is used. A convex polygonal goal with the size restricted to the sensor range can be covered from any point of its border; hence, the convexity of the polygonal goal is mainly assumed for this kind of problems.

The paper presents several modifications of the adaptation rules to deal with polygonal goals. It starts from considering centroids of the goals as a point goals and simple avoidance of the winner node adaptation towards such a point goal once the winner is

already inside the region associated to the goal. The next extension considers an intersection point of the shortest path from a node to the point goal with the boundary of the polygonal area. The intersection point is then used as a point towards which the winner node is attracted instead of the centroid of the goal.

Finally, the last extension generalizes the adaptation using the shortest segment–segment paths. The ring of nodes is considered as a path consisting of a sequence of straight line segments. Besides, each goal is also considered as a set of straight line segments forming the border of the polygonal goal; however, only the segments do not intersecting with the obstacles are considered. During the selection of the winner for the currently presented goal to the network, the aforementioned approximation of the shortest segment-segment path is employed. Once the closest segment of the ring to the goal border is determined, a new node can be created according to the rules proposed in [29]. Then, the winner node is adapted towards the point on the particular segment of the goal region.



(a) point goals - *traveling salesman problem* (TSP); n=68, T=0.35 s

(b) convex cover set - *watchman route problem* (WRP); n=106, T=2.00 s

(c) convex goals (*covering rooms*); n=21, T=0.10 s

(d) polygonal goals; n=9, T=0.09 s

(e) polygonal goals; n=35, T=0.84 s

(f) polygonal (non-convex) goals; n=4, T=2 ms

Figure 7: Examples of found solutions of different variants of the multi-goal path planning problem. The required computational time T is for the c++ implementation of the learning algorithm running on a single core of the iCore7@3.4 GHz cpu.

A detailed evaluation of the self-organizing map based algorithms and a comparison with a reference algorithm is presented in [36], and therefore, only selected solutions with updated required computational time are presented in Fig. 7. The considered problem for-

mulation allows overlapping polygonal goals and thus the algorithm is also able to solve variants of the *watchman route problem*, which is shown in Fig. 7b. In Fig. 7c, particular rooms are selected as goals and the algorithm finds the sequence of their visits. Notice that this sequence can be found exactly as a solution of the TSP using the centroids of the goal areas; however, the main point of this problem is a selection of the particular points of each goal, from which a room is inspected. A more interesting problems are situated in the bottom row of Fig. 7, where an additional problem with overlapping goals is in the middle column.

The approximation of the shortest segment–segment path is the enabling underlying technique that allows to address the generalized problems with polygonal goals. This approximation inherits the approximation of the shortest point–segment as well as point–point paths; hence, a SOM adaption using this approximation can be simply used to solve multi-goal path planning problems with point or segment goals. The simplicity of the extension based on this approximation demonstrates flexibility of the SOM based multi-goal path planning and provides the foundations to a unifying multi-goal path planning framework for various planning problems in 2D. The main feature of the SOM approach is that both sub-problems of the inspection planning are solved simultaneously, i.e., the particular sensing locations are determined during the solution of the sequencing part.

4 Inspection Planning in 3D

The unifying framework for various 2D routing problems, described in the previous section, provides a ground for further extensions. Several extending directions can be identified; however, regarding the current challenges investigated by the research community, three directions seem to be the most important ones.

The first is a consideration of a motion planning for connecting the goals instead of a simple path planning that provides a path consisting of straight line segments. From a mission planning context, such a path can be sufficient for application scenarios where mobile robots have differential drive and satisfy the holonomic constraints or when a path is used to guide a human or robotic entity in a telematic system [55]. On the other hand, it is not the case for car-like robots or more complex robotic systems with many degrees of freedom. It is also not the case if it is necessary to plan in an unstructured environment (e.g., like in the search and rescue mission), where it is needed to consider 3D model of the environment.

The second direction is a natural extension of the 2D planning framework for planning in 3D environment. This is tightly coupled with the motion planning because planning a path for a non-point robot is non-trivial [71, 62]; hence, motion planning techniques that are able to provide a feasible path in high dimensional configuration spaces have to be considered.

The third important extension is planning for a multi-robot team. This problem has been partially addressed in [25], where the SOM approach for the multiple traveling salesmen problem with minmax criterion [81] has been extended to the multi-robot multi-goal path planning problem by using the aforementioned approximation of the shortest path. The results provided by the SOM are competitive to the combinatorial heuristic [38].

Moreover, due to the nature of the SOM evolution in $\mathcal{W}$, found solutions tend to be non-crossing [25], which is a desirable feature for mutually collision free paths. However, it is necessary to consider the time domain to guarantee the robots' plans are collision free, and therefore, a motion planning technique providing trajectories (i.e., paths parametrized by time) have to be used instead of a simple path planning method.

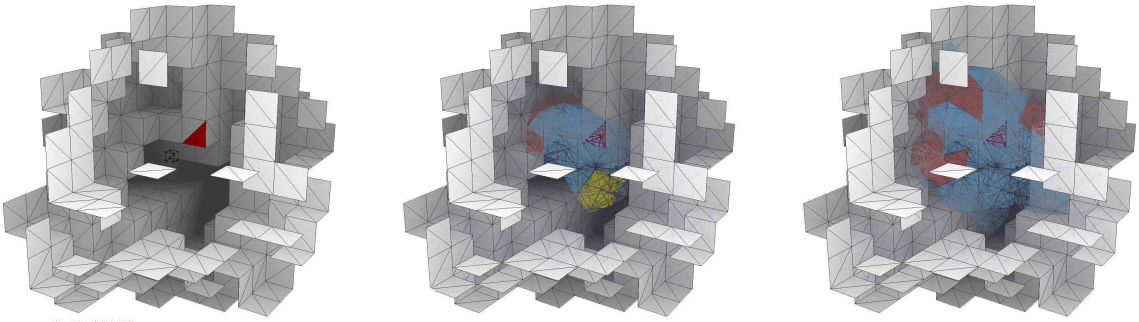
Based on the aforementioned comments, it is desirable to firstly extend the unifying inspection planning framework for 3D environment with considering kinematic constraints and a motion planning technique. However, the fundamental issue of the inspection planning is a determination of visibility (or coverage). In 2D case, a visibility graph or visibility region from a particular point in the polygonal domain $\mathcal{W}$ can be determined in a polynomial time [69] but visibility queries in 3D are more computationally demanding [19, 67].

A feasible way to determine a coverage from a point in 3D is to use techniques from computer graphics for 3D modeling. Regarding the context of the inspection planning, where the particular sensing locations are not prescribed in advance, such a ray-casting method is too computationally demanding to be considered during the SOM evolution in 3D environment. Hence, inspired by the SOM based algorithm for the watchman route problem, another approach has been proposed to deal with the 3D coverage queries. The idea is based on discretization of the 3D free space of the robot's working environment to a tetrahedral mesh similarly as a triangular mesh has been used for 2D problems in [26]. On top of this tetrahedral mesh, a set of tetrahedra from which a particular object of interest can be covered is determined in advance. Such a set can be then used during the SOM evolution in 3D to answer if a configuration in a particular tetrahedra can cover the given goal. This idea has been elaborated in the master thesis of Petr Janoušek under supervision of the author. Later, the results of the thesis have been consolidated and published in:

[46] – Janoušek P. - Faigl J.: *Speeding up coverage queries in 3D multi-goal path planning*. In Int. Conf. on Robotics and Automation (ICRA), 2013, p. 5067–5072. Authorship 50%. The paper is attached on page 93.

The proposed approach is relatively simple and straightforward. It is studied in the inspection planning problem that can be formulated as follows: having a set of objects of interest $\mathcal{M}$, find a shortest path such that all objects will be seen from the path with the sensing range ρ. The problem is considered for an operational environment $\mathcal{W} \subset \boldsymbol{R}^3$ that is defined by a set of triangles representing obstacles and the objects to be covered. Without lost of generality, it is assumed that each object of interest $m \in \mathcal{M}$ is sufficiently small, i.e., surfaces of large objects are tessellated into smaller triangles.

For each $m \in \mathcal{M}$ the subspace of $\mathcal{W}$ from which m can be covered using the omnidirectional sensor with sensing range ρ is found as an approximation consisting of tetrahedra of the tetrahedral mesh. The subspace is called *covering space* and for the object of interest m it is denoted as C_m. The idea of C_m construction is based on incremental adding of tetrahedra $\boldsymbol{e}$ for which a segment $(p, p_m), p \in \boldsymbol{e}, p_m \in m$ do not intersect an obstacle. A notion of *transitive dependency* of a tetrahedron being added to C_m on other tetrahedra is introduced together with building an auxiliary graph with information about the transitive dependencies. Using this supporting graph, the searching for the possible tetrahedra of C_m is effective and the complexity of the procedure to construct C_m is $\Theta(n)$, where n is the number of tetrahedra of the tetrahedral mesh. A visualization of the construction



(a) (b) (c)

Figure 8: A construction of the covering space: (**a**) 3D environment represented by a set of triangles, the object of interest m being covered is the red triangle in the middle; (**b**) processed tetrahedra, the currently evaluated tetrahedron is in yellow, from the red tetrahedra coverage of m cannot be guaranteed; (**c**) created covering set C_m.

process is shown in Fig. 8 and detailed information about the procedure can be found in [46].

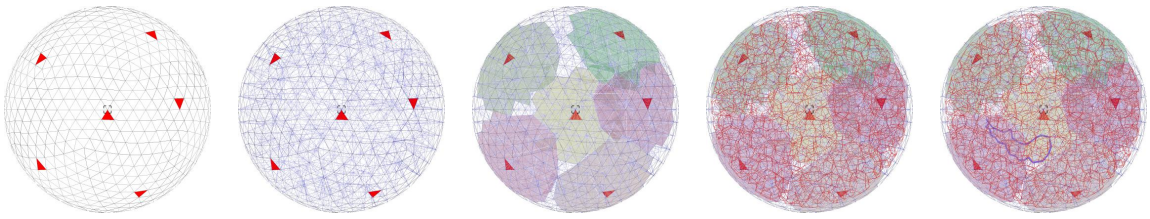
Having the supporting structure of C_m for each $m \in \mathcal{M}$, an application of the SOM based approach for multi-goal path planning is straightforward. During the learning, the goals are presented to the network in a random order and the winner node ν^* is selected for each goal. For coverage of m it is not necessary to visit m as it is sufficient to visit any point of the particular C_m; hence, during the winner selection, configurations of C_m are considered and the possibly best configuration p is found together with the winner node according to

$$(\nu^*, p) \quad = \quad \text{argmin}_{\nu \in \mathcal{N}, \boldsymbol{e} \in C_m} \, \text{distance}(\nu, \text{centroid}(\boldsymbol{e})). \tag{1}$$

In general, any configuration of C_m can be used; however, a centroid of each tetrahedron $\boldsymbol{e} \in C_m$ is considered for simplicity in this initial feasibility study of the SOM based planning in 3D environment.

Moreover, centroids are also used to address the motion planning issue by planning on a motion planning roadmap. In particular, the probabilistic roadmap method (PRM) [48] is used; however, instead of random configurations, the centroids of the tetrahedra are considered as the vertices of the roadmap for simplicity. The roadmap is a graph G_{prm}, where the vertices represent particular configurations of the robot and they are connected by an edge only if a feasible path between the two incident configurations is found by a local path planner. Then, the SOM adaptation on a graph proposed by Yamakawa et al. [91] is used to adapt the network. Using this adaptation, position of each node (the neuron's weights) are restricted to the edges and vertices of G_{prm}; hence, the feasibility of the path represented by the ring of the nodes is guaranteed according to the kinematic constraints considered during the construction of G_{prm}.

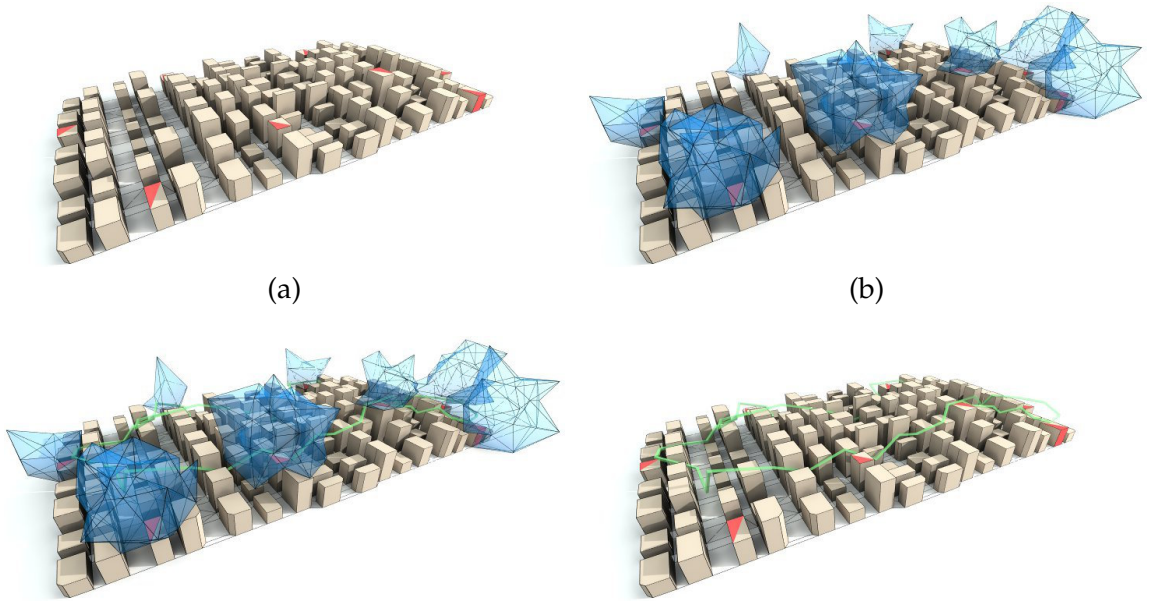
Considering the aforementioned building blocks, the proposed planning can be summarized in five steps. First, a triangular mesh defining the surfaces of obstacles and objects of interest is created. Then, a tetrahedral mesh of the free space of the environment is created and on top of this mesh covering spaces are determined. After that, the motion planning roadmap is constructed from the centroids of the mesh's tetrahedra. Finally, SOM



(a) Triangular Mesh (b) Tetrahedral mesh (c) Covering spaces (d) Motion Planning Roadmap (e) SOM adaptation

Figure 9: Visualization of the particular steps of the SOM based multi-goal inspection path planning in 3D environment.

adaptation is performed using the roadmap graph and covering spaces. A visualization of the particular steps is shown in Fig. 9.



(a) (b) (c) (d)

Figure 10: An example of 3D inspection planning problem, covering spaces and found solution.

Regarding the visibility (or coverage) queries, the query 'if a goal m can be covered from a particular configuration represented by the neuron' is answered instantaneously, because a node is always associated to some tetrahedron as its movement is restricted to the edges of G_{prm}. Hence, using the proposed simple approximation of the covering spaces, the visibility queries are not a computational issue. An example of the visualized problem, covering spaces and found solution is presented in Fig. 10.

The ring evolution on the graph G_{prm} provides a straightforward way how to deal with kinematic constraints. Instead of the PRM, other motion planning techniques can be considered, which can even include kinodynamic constraints like the Rapidly-Exploring Random Tree (RRT) [59]. Besides, novel probabilistically optimal algorithms proposed by

Sertac Karaman and Emilio Frazzoli [47] can be considered as well, e.g., PRM* or RRG. On the other hand, huge and dense underlying graphs can lead to computationally demanding determination of the shortest path in the graph. Due to a large number of vertices, pre-computation of the distance matrix can be inefficient due to memory requirements. The feasibility study [46] provides initial insights regarding this issue. The presented results for graphs with more than one hundred thousands of nodes seem to be promising as the required relevant part of the distance matrix needs only 10 GB memory including the storage of the particular shortest paths. However, the future work relies on an efficient handling of the graph size, computation of the trajectories on demand and development of a suitable approximation, which can speedup the winner selection as well as decrease the required memory. According to the results achieved in 2D planning and this early work [46], it is assumed it will be computationally feasibility using the nowadays computers.

5 Surveillance Mission Planning with Considering Sources of Uncertainty in Navigation

A robot that is autonomously navigated has to be localized within the operational environment with a sufficient precision according to the task it is being to fulfill. It is clear the precision is always limited due to imprecise sensors and noise. One of the popular approaches to localize the robot is the so-called SLAM method, from the *simultaneous localization and mapping*, which stands in simultaneous building an environment model while the model is also used to localize the robot within the created map [87]. SLAM approaches are able to provide estimation of the robot position in cases where no prior information about the environment is available in advance. Even though a significant progress has been made in this approach and plethora methods have been proposed, a more reliable navigation with a higher precision can be achieved using a prior map of the environment.

Imagine a situation where it is desired to monitor given areas of interests by a single mobile robot. The mobile robot is requested to autonomously visit the given set of areas, where sensor measurements can be taken. Therefore, one of the key problem is to ensure that the robot will be in a sufficient distance from the requested location whenever the robot's localization system announces the robot reaches the location. Regarding a surveillance scenario, the problem is to provide a periodical coverage of the areas of interest; hence, the problem is to visit the areas as frequently as possible, which means to navigate the robot along the shortest path connecting the areas while the robot position estimation is sufficiently small. In such a scenario, the environment is known in advance, and therefore, a map-and-replay technique can be a more appropriate because it can provide a more reliable and precise information about the robot's position.

The idea of such a navigation approach (e.g., [73, 76, 39]), is based on recognition of visual features that are matched with the previously stored features forming a prior map of the environment. The map is created during the initial mapping phase, in which the robot can be manually driven along the desired path. Then, during the autonomous navigation, the robot is requested to follow the previously learned path while the current seen features are compared with the expected features stored in the map and the result of the comparison is used to steer the robot control to follow the previously learned path.

Figure 11: Examples of the detected landmarks, matching of the current and previously learned landmarks and outdoor environments where the navigation method provides reliable autonomous navigation.

Based on this idea, a similar autonomous navigation method has been proposed in:

> [52] – Krajník, T. - Faigl, J. - Vonásek, V. - Košnar, K. - Kulich, M. - et al.: *Simple, Yet Stable Bearing-Only Navigation*. Journal of Field Robotics. 2010, vol. 27, no. 5, p. 511-533. ISSN 1556-4959. IF=2.244. Authorship 20%. The paper is attached on page 99.

The method is based on detection of salient objects that are utilized for heading corrections of the robot during its autonomous navigation along straight line segments. In particular, SURF (Speeded Up Robust Features) [10] descriptor is used; hence, the method is called SURFNav.

The main idea of SURFNav is that the localization problem is decomposed into an independent estimation of the robot position along a line segment of the learned path and an estimation of the robot's heading towards the end of the current segment along which the robot is navigated. The traveled distance is estimated using the robot's odometry system while the heading is corrected by the visual feedback of the currently seen landmarks and previously learned landmarks along the path.

The learned path consists of a sequence of straight line segments and for each segment a set of visual landmarks is remembered and the local length of the segment is measured by the odometry. This local consideration of the odometry is advantageous as it avoids the long-term accumulation of the error. During the replay phase, the robot is placed at the starting segment and it is requested to travel the learned path. The histogram voting method is used to implement the so-called visual compass and to steer the robot heading during the navigation along a particular segment. Once the traveled distance reaches the segment length, the robot is turned into the direction of the next segment by the compass, and the next segment is traversed in the same manner.

Although the method seems to be simple, it provides robust and reliable navigation under different lighting conditions for indoor as well as outdoor environments [51]. Examples of environments where the method has been successfully used are shown in Fig. 11. Moreover, contrary to other methods, a formal proof of its stability is provided in [52]. The proof is based on evaluation of the localization error of the robot navigated along a closed path and the stability condition is formulated as Lyapunov discrete equation for the covariance matrix of the robot's position. Then, the bound of the robot's position error at the end of the closed path is determined for the case of the infinity number of the travelled loops and convergence conditions are established. A detailed description and the formal proof itself can be found in [52]. Here, it is worth to remind that the main principle of the

bounded localization error is the independent estimation of the robot position along the path and its heading corrections according to the detected and matched visual features. In addition, it should also be mentioned, from a practical point of view, it is assumed that the estimation of the heading is a more precise than odometry measurements.

Regarding the surveillance or inspection mission planning, the SURFNav method provides additional advantage over other methods. The formal proof provides mathematical description of the position uncertainty evolution along the path consisting of straight line segments. The uncertainty is in the term of the covariance matrix of the robot position. Even though the model equations can be found in [52] they are presented here to improve readability of this text. The covariance matrix $\boldsymbol{A}_{i+1}$ at the end of the segment i can be computed using the formula:

$$\boldsymbol{A}_{i+1} = \boldsymbol{R}_i^T \boldsymbol{M}_i \boldsymbol{R}_i \boldsymbol{A}_i \boldsymbol{R}_i^T \boldsymbol{M}_i^T \boldsymbol{R}_i + \boldsymbol{R}_i^T \boldsymbol{S}_i \boldsymbol{R}_i, \tag{2}$$

where

$$\boldsymbol{M}_i = \begin{bmatrix} 1 & 0 \\ 0 & e^{-\frac{s_i}{\rho}} \end{bmatrix}, \boldsymbol{S}_i = \begin{bmatrix} s_i\eta^2 & 0 \\ 0 & \tau^2 \end{bmatrix}, \tag{3}$$

s_i is the length of the ith segment, $\boldsymbol{R}$ is the rotation matrix, ρ denotes an "average" distance of the visible landmarks (a parameter of the operational environment) and η, τ represent precision of the odometry and the heading sensor, respectively.

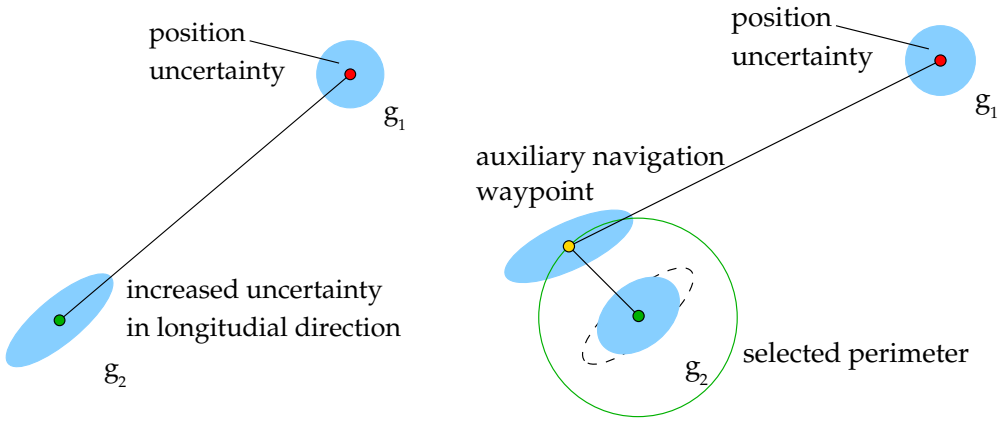
The model of the localization uncertainty described by (2) and (3) can be considered in the path planning in order to find a path with an eventually lower localization uncertainty than planning just a shortest path. Contrary to approaches like planning in the joint space of pose$\times$uncertainties [14] or based on simulation of the localization withing the map that are computationally demanding, the model of SURFNav is fast to compute. Moreover, it also provides informative estimation of the real localization error and a mechanism of the uncertainty reduction, which in fact is the source of the bounded error. These are the main advantages of the method over other approaches. For example a simple model of the increasing odometry error just leads to minimize the planned path length, but it does not provide a way to "correct" the robot pose estimation. Therefore such model cannot be efficiently used in a long-term planning.

The idea of the multi-goal path planning with consideration of the localization uncertainty of the SURFNav method has been introduced in:

> [30] – Faigl, J. - Krajník, T. - Vonásek, V. - Přeučil, L.: *Surveillance Planning with Localization Uncertainty for UAVs*. In 3rd Israeli Conference on Robotics,. Ariel: Ariel University Center, 2010. Authorship 60%. The paper is attached on page 123.

The main principle of the decreasing the localization uncertainty is schematically visualized in Fig. 12, where the corresponding covariance matrices are visualized as ellipses [78].

Consider a navigation of the robot from the location g_1 to the location g_2. The localization uncertainty is increased in the longitudinal direction due to the odometry error while it is decreased in the lateral direction because of heading corrections during the navigation towards g_2. Now, imagine that an auxiliary navigation waypoint is visited prior g_2. Then, the localization uncertainty in the previous longitudinal direction is decreased because of corrections during navigation from the waypoint to the goal, see Fig. 12b.



(a) direct connection of goals (b) auxiliary navigation waypoint

Figure 12: A principle of the localization uncertainty decreasing.

This principle of the localization uncertainty decreasing is employed in the self-organizing adaptation and the multi-goal path planning problem is formulated as the problem of finding a path that visits all the given goals while the length of the path as well as localization uncertainty at the goals is minimized. This multi-modal optimization follows the requirement of the precise visitation of the goals as an underlying problem of visiting the goals as frequently as possible, which is the main requirement of the surveillance task.

Although for a straight line segment an optimal position of such an auxiliary navigation waypoint can be determined, i.e., a point on the Thales' circle defined by the segment, it is not the case of multi-goal path planning, where the particular path between two goals consists of several straight line segments. In addition, the expected localization error also depends on the sequence of the goals' visits. Therefore, the problem is transformed to a variant of the traveling salesman problem with neighborhoods, where prior visitation of each goal an auxiliary navigation waypoint within a specific perimeter is selected in order to minimize the localization uncertainty at the goal.

Having a goal g and its neighborhood of the possible auxiliary navigation waypoints P_g the waypoint is determined according to minimization of the expected error at the goal g:

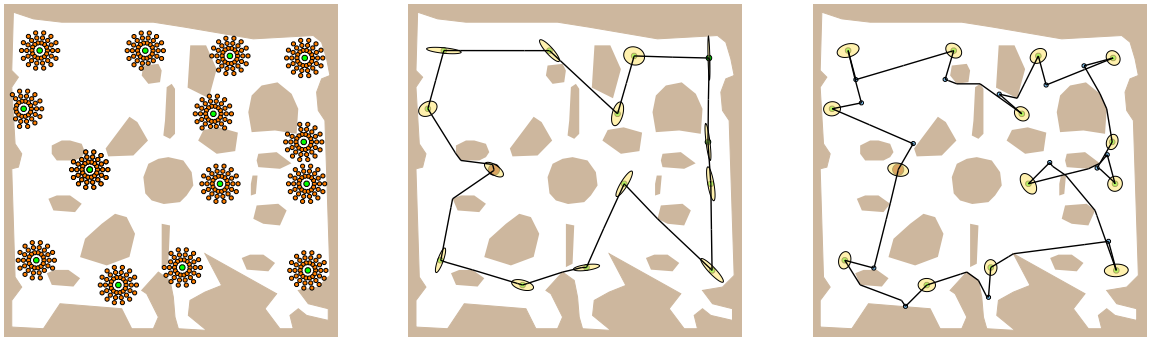
$$p_{g,\star} = \text{argmin}_{p \in P_g}(||\boldsymbol{A}_g||^2), \tag{4}$$

where $||\boldsymbol{A}_g||^2$ denotes the norm of the covariance matrix, i.e., the maximal eigenvalue of $\boldsymbol{A}_g \boldsymbol{A}_g^T$. The adaptation schema of SOM is modified as follows. A regular winner selection is used to select the winner; however, the backward neighbouring nodes of the winner are adapted to the selected p_g while the winner and its forward neighbouring nodes are adapted towards the goal g.

Notice, that in this case the ring is oriented because determination of the covariance matrices depends on the path traveled from the robot starting position. The flexibility of SOM allows to include the orientation of the ring straightforwardly by "disconnecting" the ring. It means that only forward neighbouring nodes are adapted for the first nodes of the ring, while for the last nodes of the ring only the backward neighbouring nodes are adapted. Besides, at the beginning (or end) of each learning epoch the first node and

the last node of the output layer (ring) are adapted towards the robot's initial locations without selection of the winner, which provides a closed final path.

The proposed adaptation rule is called *dadapt* and it has been proposed in [30], where the navigation method is employed in the autonomous navigation of micro aerial vehicles (quad-rotor helicopter) in the surveillance task. Regarding the achieved results the proposed SOM based multi-goal path planning provides a path that leads to a more reliable autonomous surveillance, where the task is to capture an image of each goal area and to identify a marker from the image. In the terms of absolute numbers, the reliability has been increased from 82.2% to 95%, for a further details see [30] attached on page 123.



(a) possible auxiliary navigation waypoints around the goal locations

(b) simple multi-goal path planning, L=416 m, E_{max}=1.23 m

(c) planning with auxiliary navigation waypoints, L=425 m, E_{max}=0.7 m

Figure 13: An example of auxiliary navigation waypoints and found solution without and with considering the evolution of the localization uncertainty. The path length is denoted as L and the maximal expected error of the robot position at the goal locations is denoted as E_{max}.

Later, the planning method has been considered for ground robotic vehicles in:

> [34] – Faigl, J. - Krajník, T. - Vonásek, V. - Přeučil, L.: *On Localization Uncertainty in an Autonomous Inspection*. In Proceedings of 2012 IEEE International Conference on Robotics and Automation. Piscataway: IEEE, 2012, p. 1119-1124. ISBN 978-1-4673-1405-3. Authorship 40%. The paper is attached on page 129.

In addition, the model of the localization has been extended to consider not only an "average" distance of the visible landmarks but also to use a local model of visible landmarks based on a map of the landmarks, which allows to include non homogeneous spatial distribution of the features among the environment. For example, explicit consideration of visible objects that can be used for the vision based navigation. In comparison to the planning method introduced in [30], where the auxiliary navigation waypoints have been selected from a set at a specific perimeter, the approach [34] is extended to environment with obstacles and it does not rely on a single perimeter. Particular waypoints are selected from a larger set of the possible waypoints including several perimeters and thus the approach can be considered as a more general. An example of auxiliary navigation waypoints and found solutions is depicted in Fig. 13.

Based on the results presented in [30, 34], it seems that the trade-off between the precision of the goal visits and the total length of the path is proportional, i.e., a path with

auxiliary navigation waypoints is about 30% longer than a path connecting just the goals while the visitation of the waypoints improve precision of the localization at the goals also about 30%.

Although the expected improvement corresponds with the real measured precision. The absolute values of the error are different, which is mainly due to the estimated parameters of the navigation method. However, it is expected that a local model of visible landmarks introduced in [34] should provide a more precise estimation of the expected localization error. This is a subject of future work on consolidation of the achievements

Here, it should be mentioned that regarding the detection of the landmarks, it seems that BRIEF features and descriptor are more suitable for navigation than SURF, according to the recent results presented in [53].

6 Multi-Goal Path Planning in Mobile Robot Exploration

The multi-goal path planning can be considered as a planning regarding a longer horizon of the mission execution than just a pure path planning from the initial location to the goal location. In the inspection planning, it is assumed the model of the environment is known in advance and the whole mission is prepared prior the robot deployment. Even though a replanning can frequently occur during the mission execution, in inspection task, it makes sense to consider the whole plan in order to find a cost effective solution and the eventual benefit of the multi-goal path planning is evident.

On the other hand, planning in mobile robot exploration seems to be a more reactive, as no information about the environment is known in advance. Just to remind, the exploration task is a problem of building a map of an unknown environment by a single robot or a group of mobile robots and first approaches to address this problem have been proposed in eighties. In mobile robot exploration, the main problem is to determine the next goals towards which the robots are navigated to collect new information about the environment.

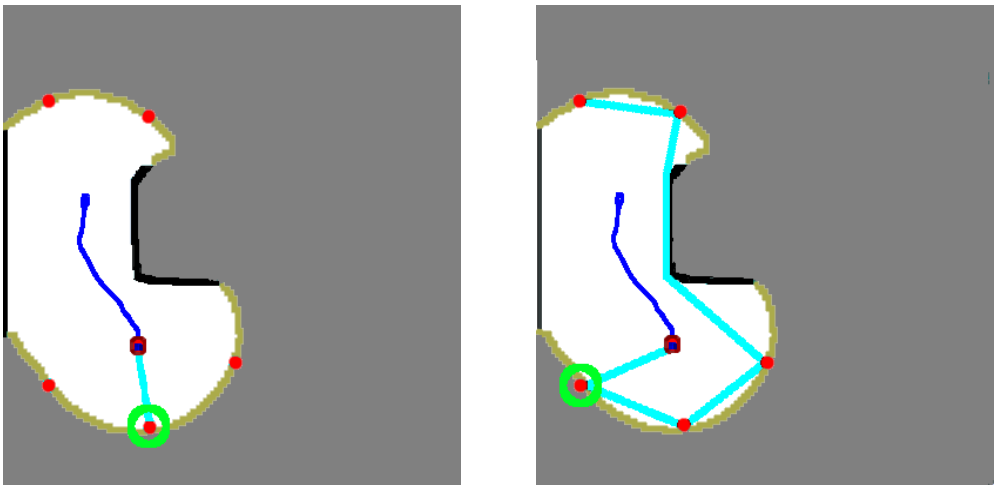
The fundamental method how to generate goal candidates is the frontier-based approach [92]. A frontier is an area between unknown and already explored space; hence, it is a good candidate to be the next goal because the robot will likely explore the unknown space during navigation towards such a goal. Frontiers can be easily determined in a grid based map, and therefore, the frontier-based exploration is usually combined with the occupancy grid for a straightforward integration of new sensor measurements [65]. Thus, the frontier-grid-based exploration is one of the most popular exploration approaches because of its simplicity and several extensions have been proposed since the original Yamauchi work, e.g., see an overview of the approaches in the recent work [9].

During the exploration, robots are navigated towards goals assigned in the next-best-view manner [44]. The goals are iteratively selected from the actual goal candidates (e.g., frontier cells) according to the selected optimization criterion. A unifying concept of how to evaluate candidate positions is based on the goal utility. Although various utility functions have been proposed, all of them basically combine information gain (or expected benefit [12]) together with the required travelling distance to the goal [1]. Then, the robot's next goal is repeatably selected from the next goal candidates.

The next-best-view approach can be considered as a greedy approach aimed to the immediate reward, i.e., the selection of the next robot goal without considering further robot movements. This makes sense regarding the current available information about the working environment, which is a natural subject of change during the exploration, and therefore, the replanning can use the most new available information. However, the problem of the exploration can be stated differently. Instead of determining just the next robot goal, the exploration can be formulated as a problem of visiting all the current frontiers; hence, a variant of the multi-goal path planning problem.

Even though this formulation of visiting the given set of all goal candidates has been discussed in [95, 12], the formulation took relatively little attention by research community, probably due to complexity of the problem. Regarding this, the most elaborating study of this TSP like formulation has been presented recently in:

[56] – Kulich, M. - Faigl, J. - Přeučil, L.: *On Distance Utility in the Exploration Task.* In Proceedings of 2011 IEEE International Conference on Robotics and Automation. Madison: Omnipress, 2011, p. 4455-4460. ISBN 978-1-61284-386-5. Authorship 20%. The paper is attached on page 135.



(a) greedy goal selection (b) TSP distance cost selection

Figure 14: An example of the goal selection using pure greedy approach (left) and the TSP distance cost (right). The selected next goal is emphasized by the green ring.

The idea of the TSP based evaluation of the robot's goal candidates is shown in Fig. 14. The evaluation is based on the length of the shortest path connecting the current set of goal candidates. So, a longer planning horizon is considered rather than the immediate next goal. Notice, the determined path is open, while the TSP is formulated for a closed path. However, a transformation of the problem to the TSP is straightforward, just a "virtual" vertex is added to the graph representation of the problem. The vertex has to be visited as the last vertex prior the route return to the vertex representing the robot's current position, which is the natural starting and ending vertex of the TSP route. This visitation of the vertex can be supported by the cost of the edges connecting the virtual vertex with other vertices; thus, the vertex is connected with the starting vertex by an edge with zero cost, while the cost of the edges to other vertices is a large value, see [56] for a further

description.

The computational complexity of the associated TSP is addressed by reducing the number of goal candidates and instead of all frontier cells only selected representatives are considered. The selection is performed as follows. First, all frontier cells are organized into a set of single connected components, i.e., the so-called free edges [44]. Then, the idea is to use only few goal candidates representing the free edges and from which all particular frontier cells would be covered (if a robot would visit the representatives). The representatives of the free edge are means of n_r clusters that are found for each free edge using the K-means clustering algorithm. The number of clusters n_r is determined considering the range of the sensor ρ_g (in the number of grid cells), see [56]. Beside this significant reduction of the possible goal candidates, the TSP is solved approximately using the Chained Lin-Kernighan heuristic [4], and therefore, it is computationally feasible even for real-time planning.

The main results presented in [56] is that using the proposed TSP distance cost, the time needed to explore the whole environment is reduced up to about 30% in comparison with the greedy selection of the robot's next goal from the current goal candidates. These results are promising especially with regards to the fact that the TSP distance cost does not include any kind of expectations about possible coverage of unexplored areas and which is solely based on the distances between the candidates. Based on these findings, the developed TSP distance cost has been considered in the multi-robot exploration in:

> [35] – Faigl, J. - Kulich, M. - Přeučil, L.: *Goal Assignment using Distance Cost in Multi-Robot Exploration*. In Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscataway: IEEE, 2012, vol. 1, p. 3741-3746. ISBN 978-1-4673-1735-1. Authorship 60%. The paper is attached on page 141.

According to the taxonomy [40], the multi-robot exploration is formulated as the optimal assignment problem studied in operational research. The problem is to find the best assignment of n goals to m robots maximizing the overall utility, i.e., to find one goal for each robot. In the case of linear optimization criterion, the problem can be solved in polynomial time using the Hungarian algorithm, which has been applied in multi-robot exploration by authors of [63], where Voronoi Graphs of the current known environment are used to explore a single room by one robot.

In [35], the assignment problem is formulated as the multiple traveling salesman problem (MTSP) that is solved approximately using the ⟨*cluster first, route second*⟩ heuristic. The clusters are found by the K-means algorithm and the next goal is selected from the particular cluster according to the TSP distance cost [56]. The presented results indicate that this formulation provides a shorter exploration time than using the Hungarian algorithm. An example of travelled trajectories during the exploration is shown in Fig. 15. These results are obtained by a developed multi-robot exploration framework, which support study and evaluation of the performance of exploration strategies. The framework follows recommendations of benchmarking exploration strategies [1] and it provides a ground for focused study of the decision mechanisms without influence of other parts of the real navigation system, which may affect the performance significantly. Besides, it also allows to evaluate the results statistically using thousands of simulations, which are unlikely possible with real robots.

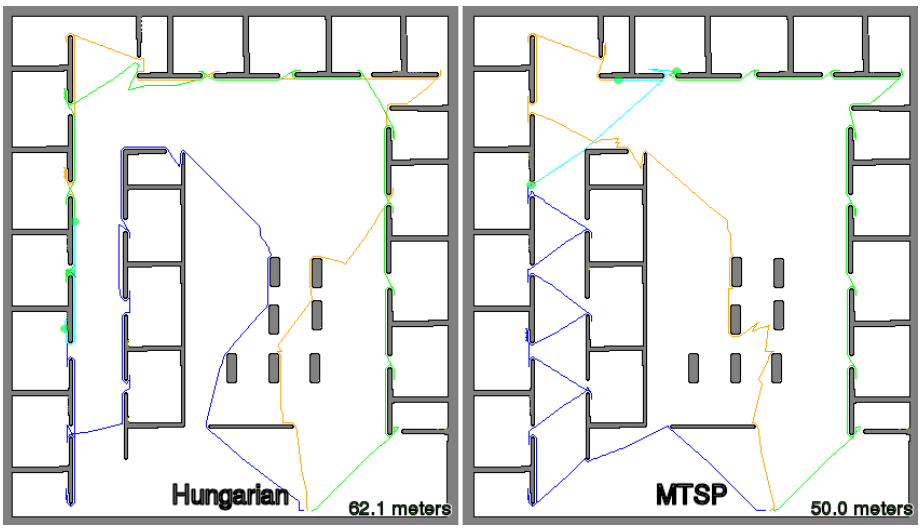


Figure 15: An example of multi-robot exploration strategies. The maximal travelled distance is 62.1 meters using the Hungarian assignment (left) while for the proposed MTSP based strategy, the maximal travelled distance by a robot is 50 meters.

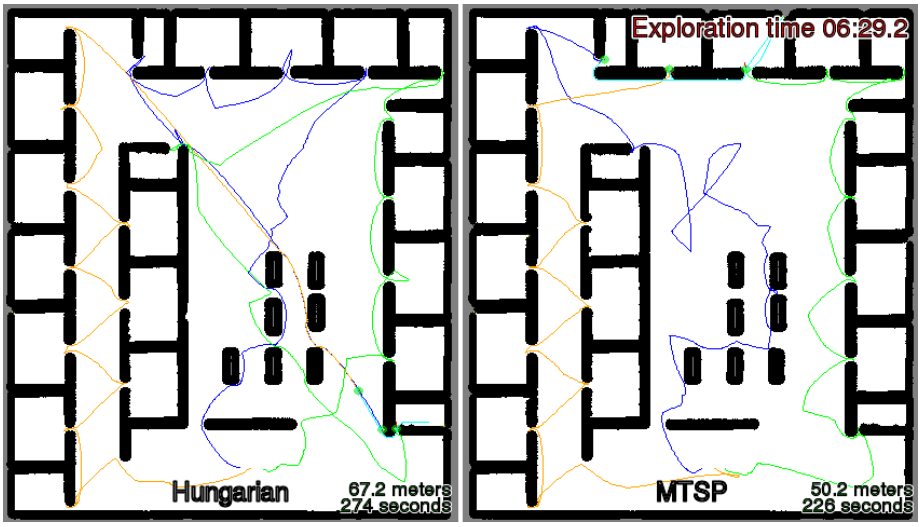


Figure 16: Example of traveled paths during the multi-robot exploration strategies. The total mission time needed to explore the whole environment is 274 seconds for the Hungarian assignment (left) and 226 seconds for the MTSP based exploration strategy.

An example of exploration trajectories in a more realistic deployment using the Player/Stage framework [41] is shown in Fig. 16. In this type of evaluation, the whole control stack for autonomous navigation is employed and the reactive based SND navigation [20] is used for a low-level control of the robot motion. Contrary to the previous evaluation using the developed framework, which allows to set the replanning period precisely, in this deployment, the robots are continuously navigated towards their actual goals, while the re-planning is performed as fast as possible. Therefore, a more computationally demanding method, e.g., the MTSP strategy, has a longer replanning period, while simpler techniques can take an advantage of faster replanning considering new information about

the environment being explored. However, also in this deployment, the proposed MTSP exploration provides better results and thus the results support the idea of the multi-goal path planning in exploration missions.

Regarding the SOM based approach for the multi-goal path planning, which demonstrates competitive results in the inspection tasks, the current approach for the MTP in the exploration relies on combinatorial heuristic for the TSP. For the current formulation of the exploration strategy, SOM does not provide suitable solutions. Having a fast re-planning loop, the found paths can be significantly different, which may lead to oscillation of the robot between two goals. In fact, this issue can also occur in utility based strategy combining distance cost with the expected information gain as it is commented in [44]. However, the future intention is to reformulate the exploration strategy as a coverage problem considering a continuous sensing, i.e., the problem of determining paths from which all the frontiers cells would be covered instead of selection of frontier cells towards which the robots are navigated.

7 Concluding Remarks and Future Work

The presented introduction to multi-goal path planning for mobile robots and the commentary to the author's work in this field is mostly based on the self-organizing maps. Prior the presented work, this technique took a little attention for these types of problems by the community, especially in robotic research. From the pragmatical point of view, it is apprehensible as the SOM technique for the TSP provides generally worse results and it was even more computationally demanding than heuristics from operational research.

On the other hand, with a relatively little effort, the SOM based multi-goal path planner provides solution of huge class of problems and represents a general approach for inspection planning including sensor models with discrete and continuous sensing. During the last years of promoting the developed approaches, the author receive positive feedbacks from the research community in this sense. The proposed algorithms are "incredibly" simple; however, the provided solutions have competitive quality to the specialized and in several cases also significantly more complex approaches.

The main principle proposed can be summarized in a combination of simple supporting structures (like the approximation of the shortest path) with a self-organizing principle, which can also be considered as simple. The achieved results provide an insight that for solving the addressed problems, which are motivated by inspection planning, the self-organizing map discovers the approximation of the problem topology that represents a solution of the problem. Hence, the natural evolution process considering only local properties is able to provide a solution according to the global optimization criterion. This insight provides a ground for a future work on distributed solution of large problems based on self-organizing principles.

References

[1] F. Amigoni and V. Caglioti. An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems*, 58(5):684–699, 2010.

[2] B. Angéniol, G. de la C. Vaubois, and J.-Y. L. Texier. Self-organizing feature maps and the travelling salesman problem. *Neural Networks*, 1:289–293, 1988.

[3] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. CONCORDE TSP Solver, 2003. URL `http://www.tsp.gatech.edu/concorde.html`. [cited 2 May 2013].

[4] D. Applegate, W. Cook, and A. Rohe. Chained Lin-Kernighan for large traveling salesman problems. *Informs J. on Computing*, 15(1):82–92, 2003.

[5] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.

[6] N. Aras, B. J. Oommen, and I. K. Altinel. The Kohonen network incorporating explicit statistics and its application to the travelling salesman problem. *Neural Networks*, 12 (9):1273–1284, 1999.

[7] N. Aras, I. Altinel, and J. Oommen. A Kohonen-like decomposition method for the Euclidean traveling salesman problem-KNIES_DECOMPOSE. *IEEE Transactions on Neural Networks*, 14(4):869–890, July 2003.

[8] Y. Bai, W. Zhang, and Z. Jin. An new self-organizing maps strategy for solving the traveling salesman problem. *Chaos, Solitons & Fractals*, 28(4):1082–1089, 2006.

[9] N. Basilico and F. Amigoni. Exploration strategies based on multi-criteria decision making for searching environments in rescue operations. *Autonomous Robots*, 31(4): 401–417, 2011.

[10] H. Bay, A. Ess, T. Tuytelaars, and L. J. V. Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[11] S. Bespamyatnikh. An $o(n \log n)$ algorithm for the Zoo-keeper's problem. *Computational Geometry: Theory and Applications*, 24(2):63–74, 2002.

[12] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[13] L. Burke. "Conscientious" neural nets for tour construction in the traveling salesman problem: the vigilant net. *Computers and Operations Research*, 23(2):121–129, 1996.

[14] A. Censi, D. Calisi, A. D. Luca, and G. Oriolo. A Bayesian framework for optimal motion planning with uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, May 2008.

[15] E. M. Cochrane and J. E. Beasley. The co-adaptive neural network approach to the Euclidean travelling salesman problem. *Neural Networks*, 16(10):1499–1525, 2003.

[16] J.-C. Créput and A. Koukam. A memetic neural network for the Euclidean traveling salesman problem. *Neurocomputing*, 72(4-6):1250–1264, 2009.

[17] T. Danner and L. E. Kavraki. Randomized Planning for Short Inspection Paths. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 971–976, San Fransisco, CA, April 2000. IEEE Press.

[18] M. Dror, A. Efrat, A. Lubiw, and J. S. B. Mitchell. Touring a sequence of polygons. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 473–482, San Diego, CA, USA, 2003. ACM Press.

[19] F. Durand, G. Drettakis, and C. Puech. The 3D Visibility Complex. *CM Transactions on Graphics*, 21(2):176–206, 2002.

[20] J. W. Durham and F. Bullo. Smooth nearness-diagram navigation. In *IROS*, pages 690–695, 2008.

[21] P. Elinas. Multi-goal planning for an autonomous blasthole drill. In A. Gerevini, A. E. Howe, A. Cesta, and I. Refanidis, editors, *ICAPS*. AAAI, 2009.

[22] B. Englot and F. Hover. Inspection planning for sensor coverage of 3d marine structures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4412–4417, 2010.

[23] B. Englot and F. Hover. Multi-goal feasible path planning using ant colony optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2255–2260, 2011.

[24] B. Englot and F. Hover. Planning complex inspection tasks using redundant roadmaps. In *15th International Symposium of Robotics Research (ISRR)*, Flagstaff, AZ, August 2011.

[25] J. Faigl. *Multi-Goal Path Planning for Cooperative Sensing*. PhD thesis, Czech Technical University in Prague, 2010.

[26] J. Faigl. Approximate Solution of the Multiple Watchman Routes Problem with Restricted Visibility Range. *IEEE Transactions on Neural Networks*, 21(10):1668–1679, 2010.

[27] J. Faigl. On the performance of self-organizing maps for the non-euclidean traveling salesman problem in the polygonal domain. *Information Sciences*, 181:4214–4229, October 2011.

[28] J. Faigl and L. Preucil. Inspection planning in the polygonal domain by self-organizing map. *Applied Soft Computing*, 11(8):5028–5041, 2011.

[29] J. Faigl and L. Přeučil. Self-Organizing Map for the Multi-Goal Path Planning with Polygonal Goals. In *International Conference on Artificial Neural Networks (ICANN)*, pages 85–92, Heidelberg, 2011. Springer.

[30] J. Faigl, T. Krajník, V. Vonásek, and L. Přeučil. Surveillance Planning with Localization Uncertainty for UAVs. In *3rd Israeli Conference on Robotics,*, Ariel, 2010. Ariel University Center.

[31] J. Faigl, M. Kulich, and L. Přeučil. A sensor placement algorithm for a mobile robot inspection planning. *Journal of Intelligent & Robotic Systems*, 62(3-4):329–353, 2011.

[32] J. Faigl, M. Kulich, V. Vonásek, and L. Přeučil. An Application of Self-Organizing Map in the non-Euclidean Traveling Salesman Problem. *Neurocomputing*, 74(5):671–679, 2011.

[33] J. Faigl, V. Vonásek, and L. Přeučil. A Multi-Goal Path Planning for Goal Regions in the Polygonal Domain. In *Proceedings of the 5th European Conference on Mobile Robots*, pages 171–176, Örebro, 2011. AASS Research Centre.

[34] J. Faigl, T. Krajník, V. Vonásek, and L. Přeučil. On Localization Uncertainty in an Autonomous Inspection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1119–1124, 2012.

[35] J. Faigl, M. Kulich, and L. Přeučil. Goal assignment using distance cost in multi-robot exploration. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2012. (submitted).

[36] J. Faigl, V. Vonásek, and L. Přeučil. Visiting convex regions in a polygonal map. *Robotics and Autonomous Systems*, 2012. (*accepted, to appear*).

[37] J. C. Fort. Solving a combinatorial problem via self-organizing process: An application of the Kohonen algorithm to the traveling salesman problem. *Biological Cybernetics*, 59(1):33–40, 1988.

[38] P. M. França, M. Gendreau, G. Laporte, and F. M. Müller. The m-Traveling Salesman Problem with Minmax Objective. *Transportation Science*, 29(3):267–275, 1995.

[39] P. T. Furgale and T. D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics,* special issue on "Visual mapping and navigation outdoors", 27(5):534–560, 2010.

[40] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954, 2004.

[41] B. P. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proc. of the 11th Int. Conf. on Advanced Robotics*, pages 317–323, 2003.

[42] H. González-Baños, D. Hsu, and J.-C. Latombe. Motion planning: Recent developments. In S. Ge and F. Lewis, editors, *Autonomous Mobile Robots: Sensing, Control, Decision-Making and Applications*, chapter 10. CRC Press, 2006.

[43] H. González-Baños and J.-C. Latombe. Planning Robot Motions for Range-Image Acquisition and Automatic 3D Model Construction. In *AAAI Fall Symposium*, 1998.

[44] H. H. Gonzalez-Banos and J.-C. Latombe. Navigation Strategies for Exploring Indoor Environments. *International Journal of Robotics Research*, 21(10-11):829–848, 2002.

[45] K. Helsgaun. An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. *European Journal of Operational Research*, 126(1), 2000.

[46] P. Janoušek and J. Faigl. Speeding up coverage queries in 3d multi-goal path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5067–5072, 2013.

[47] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *I. J. Robotic Res.*, 30(7):846–894, 2011.

[48] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, aug 1996.

[49] G. Kazazakis and A. Argyros. Fast Positioning of Limited Visibility Guards for the Inspection of 2D Workspaces. In *Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS 2002)*, Lausanne, Switzerland, September 2002.

[50] T. Kohonen, M. R. Schroeder, and T. S. Huang, editors. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 2001.

[51] T. Krajník, J. Faigl, and L. Přeučil. A Simple Yet Reliable Visual Navigation System. In *Mobile Robotics for Environment/Agriculture*, volume 1, Zaragoza, 2010. University of Zaragoza.

[52] T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, and L. Přeučil. Simple yet stable bearing-only navigation. *Journal of Field Robotics*, 27(5):511–533, 2010.

[53] T. Krajník, P. C. P., J. Faigl, H. Szücsová, M. Nitsche, L. Přeučil, and M. Mejail. Image features for long-term mobile robot autonomy. In *In Workshop Proceedings on Long-Term Autonomy, held in conjunction with ICRA*, 2013.

[54] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *The Fifth International Conference on Information Processing in Sensor Networks (IPSN)*, pages 2–10, 2006.

[55] M. Kulich, J. Kout, L. Přeučil, and J. P. et al. PeLoTe - a Heterogeneous Telematic System for Cooperative Search and Rescue Missions. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2004.

[56] M. Kulich, J. Faigl, and L. Přeučil. On Distance Utility in the Exploration Task. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4455–4460, 2011.

[57] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, June 1992.

[58] S. M. Lavalle. *Planning Algorithms*. Cambridge University Press, May 2006.

[59] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotic Research*, 20(5):378–400, 2001.

[60] K.-S. Leung, H.-D. Jin, and Z.-B. Xu. An expanding self-organizing neural network for the traveling salesman problem. *Neurocomputing*, 62:267–292, 2004.

[61] S. Lin and B. W. Kernighan. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21(2):498–516, 1973.

[62] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, 1979.

[63] K. M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1160–1165, 2008.

[64] A. Modares, S. Somhom, and T. Enkawa. A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. *International Transactions in Operations Research*, 6:591–606, 1999.

[65] H. Moravec and A. E. Elfes. High resolution maps from wide angle sonar. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 116–121, March 1985.

[66] K. Murakoshi and Y. Sato. Reducing topological defects in self-organizing maps using multiple scale neighborhood functions. *Biosystems*, 90(1):101–104, 2007.

[67] M. Nouri Bygi and F. Ghodsi. 3d visibility and partial visibility complex. In *International Conference on Computational Science and its Applications, (ICCSA)*, pages 208–207, aug. 2007.

[68] S. Ntafos. Watchman routes under limited visibility. *Computational Geometry: Theory and Applications*, 1(3):149–170, 1992.

[69] M. H. Overmars and E. Welzl. New methods for computing visibility graphs. In *SCG '88: Proceedings of the fourth annual symposium on Computational geometry*, pages 164–171, New York, NY, USA, 1988. ACM.

[70] A. Plebe. An Effective Traveling Salesman Problem Solver Based on Self-Organizing Map. In *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*, pages 908–913, London, UK, 2002. Springer-Verlag.

[71] J. H. Reif. Complexity of the mover's problem and generalizations. In *SFCS '79: Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 421–427, Washington, DC, USA, 1979. IEEE Computer Society.

[72] G. Reinelt. TSPLIB - A Traveling Salesman Problem Library. *Journal on Computing*, 3 (4):376–384, 1991.

[73] A. Remazeilles and F. Chaumette. Image-based robot navigation from an image memory. *Robot. Auton. Syst.*, 55(4):345–356, 2007. ISSN 0921-8890.

[74] M. Saha, T. Roughgarden, J.-C. Latombe, and G. Sánchez-Ante. Planning Tours of Robotic Arms among Partitioned Goals. *International Journal of Robotics Research*, 25 (3):207–223, 2006.

[75] W. R. Scott, G. Roth, and J.-F. Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Comput. Surv.*, 35(1):64–96, 2003.

[76] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette. A mapping and localization framework for scalable appearance-based navigation. *CVIU*, 113(2):172—187, 2009.

[77] T. C. Shermer. Recent results in art galleries. In *Proc. IEEE*, volume 80, pages 1384–1399, September 1992.

[78] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, 1986.

[79] R. Smith, A. Pereira, Y. Chao, P. Li, D. Caron, B. Jones, and G. Sukhatme. Autonomous underwater vehicle trajectory design coupled with predictive ocean models: A case study. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4770–4777, may 2010.

[80] S. Somhom, A. Modares, and T. Enkawa. A self-organising model for the travelling salesman problem. *Journal of the Operational Research Society*, pages 919–928, 1997.

[81] S. Somhom, A. Modares, and T. Enkawa. Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Computers and Operations Research*, 26(4):395–407, 1999.

[82] P. Somol, P. Pudil, and J. Kittler. Fast branch amp; bound algorithms for optimal feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7): 900–912, july 2004.

[83] S. N. Spitz and A. A. G. Requicha. Multiple-Goals Path Planning for Coordinate Measuring Machines. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2322–2327, 2000.

[84] X. Tan and T. Hirata. Finding shortest safari routes in simple polygons. *Information Processing Letters*, 87(4):179–186, 2003.

[85] L. Techy, C. Woolsey, and K. Morgansen. Planar path planning for flight vehicles in wind with turn rate and acceleration bounds. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3240–3245, 2010.

[86] O. Tekdas, D. Bhadauria, and V. Isler. Efficient data collection from wireless nodes under the two-ring communication model. *International Journal of Robotics Research*, 31(6):774–784, 2012.

[87] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[88] M. Tucci and M. Raugi. Stability analysis of self-organizing maps and vector quantization algorithms. In *IJCNN*, pages 1–5, 2010.

[89] F. C. Vieira, A. ao Duarte Dória Neto, and J. A. F. Costa. An Efficient Approach to the Travelling Salesman Problem Using Self-Organizing Maps. *International Journal of Neural Systems (IJNS)*, 13(2):59–66, 2003.

[90] P. Wang. *View planning with combined view and travel cost*. PhD thesis, Simon Fraser University, 2007.

[91] T. Yamakawa, K. Horio, and M. Hoshino. Self-Organizing Map with Input Data Represented as Graph. In *Neural Information Processing*, pages 907–914. Springer Berlin / Heidelberg, 2006.

[92] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, AGENTS '98, pages 47–53, NY, USA, 1998. ACM.

[93] H. Yang and H. Yang. An Self-organizing Neural Network with Convex-hull Expanding Property for TSP. In *Neural Networks and Brain, 2005. ICNN&B '05. International Conference on*, volume 1, pages 379–383, Oct. 2005.

[94] W. Zhang, Y. Bai, and H. P. Hu. The incorporation of an efficient initialization method and parameter adaptation using self-organizing maps to solve the TSP. *Applied Mathematics and Computation*, 172(1):603–623, 2006.

[95] R. Zlot, A. Stentz, M. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 3016 –3023, 2002.

Appendices

p. 41 – Faigl, J.: *On the Performance of Self-organizing Maps for the Non-Euclidean Traveling Salesman Problem in the Polygonal Domain*. Information Sciences. 2011, vol. 181, no. 19, p. 4214-4229. ISSN 0020-0255. IF=2.833. Authorship 100%.

p. 57 – Faigl, J. - Přeučil, L.: *Inspection Planning in the Polygonal Domain by Self-Organizing Map*. Applied Soft Computing. 2011, vol. 11, no. 8, p. 5028-5041. ISSN 1568-4946. IF=2.612. Authorship 90%.

p. 71 – Faigl, J. - Přeučil, L.: *Self-Organizing Map for the Multi-Goal Path Planning with Polygonal Goals*. In Proceedings of Artificial Neural Networks and Machine Learning. Heidelberg: Springer, 2011, p. 85-92. ISBN 978-3-642-21734-0. Authorship 90%.

p. 79 – Faigl J. - Vonásek V. - Přeučil L., *Visiting Convex Regions in a Polygonal Map*. Robotics and Autonomous Systems, 2013, `http://dx.doi.org/10.1016/j.robot.2012.08.013`, (in press), IF=1.056., Authorship 80%.

p. 93 – Janoušek P. - Faigl J.: *Speeding up coverage queries in 3D multi-goal path planning*. In Int. Conf. on Robotics and Automation (ICRA), 2013, p. 5067–5072. Authorship 50%.

p. 99 – Krajník, T. - Faigl, J. - Vonásek, V. - Košnar, K. - Kulich, M. - et al.: *Simple, Yet Stable Bearing-Only Navigation*. Journal of Field Robotics. 2010, vol. 27, no. 5, p. 511-533. ISSN 1556-4959. IF=2.244. Authorship 20%.

p. 123 – Faigl, J. - Krajník, T. - Vonásek, V. - Přeučil, L.: *Surveillance Planning with Localization Uncertainty for UAVs*. In 3rd Israeli Conference on Robotics,. Ariel: Ariel University Center, 2010. Authorship 60%.

p. 129 – Faigl, J. - Krajník, T. - Vonásek, V. - Přeučil, L.: *On Localization Uncertainty in an Autonomous Inspection*. In Proceedings of 2012 IEEE International Conference on Robotics and Automation. Piscataway: IEEE, 2012, p. 1119-1124. ISBN 978-1-4673-1405-3. Authorship 40%.

p. 135 Kulich, M. - Faigl, J. - Přeučil, L.: *On Distance Utility in the Exploration Task*. In Proceedings of 2011 IEEE International Conference on Robotics and Automation. Madison: Omnipress, 2011, p. 4455-4460. ISBN 978-1-61284-386-5. Authorship 20%.

p. 141 Faigl, J. - Kulich, M. - Přeučil, L.: *Goal Assignment using Distance Cost in Multi-Robot Exploration*. In Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscataway: IEEE, 2012, vol. 1, p. 3741-3746. ISBN 978-1-4673-1735-1. Authorship 60%.

Appendix 1 - Faigl, J.: *On the Performance of Self-organizing Maps for the Non-Euclidean Traveling Salesman Problem in the Polygonal Domain* [27], referenced on page 9.

Information Sciences 181 (2011) 4214–4229

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

On the performance of self-organizing maps for the non-Euclidean Traveling Salesman Problem in the polygonal domain

Jan Faigl *

Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27 Prague 6, Czech Republic

ARTICLE INFO

Article history:
Received 9 July 2010
Received in revised form 4 April 2011
Accepted 21 May 2011
Available online 27 May 2011

Keywords:
Self-organizing map (SOM)
Traveling Salesman Problem (TSP)
Multi-goal path planning

ABSTRACT

In this paper, two state-of-the-art algorithms for the Traveling Salesman Problem (TSP) are examined in the multi-goal path planning problem motivated by inspection planning in the polygonal domain $\mathcal{W}$. Both algorithms are based on the self-organizing map (SOM) for which an application in $\mathcal{W}$ is not typical. The first is Somhom's algorithm, and the second is the Co-adaptive net. These algorithms are augmented by a simple approximation of the shortest path among obstacles in $\mathcal{W}$. Moreover, the competitive and cooperative rules are modified by recent adaptation rules for the Euclidean TSP, and by proposed enhancements to improve the algorithms' performance in the non-Euclidean TSP. Based on the modifications, two new variants of the algorithms are proposed that reduce the required computational time of their predecessors by an order of magnitude, therefore making SOM more competitive with combinatorial heuristics. The results show how SOM approaches can be used in the polygonal domain so they can provide additional features over the classical combinatorial approaches based on the complete visibility graph.

1. Introduction

The self-organizing map (SOM) also known as Kohonen's unsupervised neural network, was first applied to the Traveling Salesman Problem (TSP) by Angéniol et al. [1] and Fort [19] in 1988. The TSP is probably the most famous combinatorial problem studied by the operational research community for more than five decades [10]. The problem is to find a route for visiting a given set of n cities (goals) so that the length of the route is minimized. In SOM, the output neurons are organized into a unidimensional structure (cycle), and a solution is represented by synaptic weights that are adapted to the cities during the self-adaptation process. After the adaptation, the neurons are associated to the cities, and because of the unidimensional structure, the final city tour can be retrieved by traversing the cycle.

The SOM adaptation schema for the TSP consists of two phases. A city is presented to the network, and a winner neuron is selected in the *competitive* phase. For a planar TSP where cities represent points in $\mathcal{R}^2$, the neurons' weights can be considered as points in the plane that are called nodes in this paper. So, the winner neuron is the node with the smallest distance to the city. Then, the adaptation can be described as a movement of the winner node together with its neighboring nodes toward the city. The adaptation is called a *cooperative* phase, as neighboring nodes also move, although by a shorter distance. After the complete presentation of all cities (one adaptation step), the procedure is repeated until the termination condition is not met, e.g., when the winner nodes are sufficiently close to the cities.

Several SOM approaches have been proposed [9,8,6,7,36,34,3,35,4,11] in the history of the SOM application to the TSP. In these approaches, the adaptation rules have been modified [37,39], heuristics have been considered [30], and combinations

* Tel.: +420 224357224.
E-mail address: xfaigl@labe.felk.cvut.cz

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.05.019

with genetic algorithms [26], memetic [12] or immune system [27] approaches have been proposed. Even though these new approaches improve the performance of SOM for the TSP, SOM is still not competitive with the combinatorial approaches to the TSP in both aspects: the solution quality and required computational time [12]. It should also be noted that all of the above mentioned SOM approaches consider only the Euclidean variant of the TSP, i.e., distances between cities are Euclidean, while combinatorial approaches generally work with graphs.

Herein, the TSP is considered in the context of inspection planning, where cities represent sensing locations in the polygonal domain $\mathcal{W}$ [17,21]. The problem is to find a path for a mobile robot so that the robot will "see" the whole working space. The practical motivation of the problem is a search and rescue mission in which possible victims need to be found quickly [25]. The problem can be formulated as the TSP, i.e., a problem to find a path that visits the given set of sensing locations, where measurements of the robot's vicinity are taken. The robot's working space is represented by a polygonal map that may contain obstacles, therefore collision-free paths among obstacles (geodesic paths) have to be considered [32]. It is also the case of the SOM adaptation procedure where the geodesic paths (distances) between nodes and cities have to be considered rather than Euclidean distances, otherwise a poor solution would be found. The node–city distances are used in the competitive phase, in which a winner node is selected. Shortest paths are then used in the cooperative phase where nodes are adapted toward a city along a particular path, i.e., the node is placed at a new position on the path closer to the city.

It is clear that a determination of the shortest path among obstacles is more computationally intensive than a direct usage of the Euclidean distance. From this perspective, the complexity of SOM algorithms increase in $\mathcal{W}$ because the adaptation rule has to be augmented by an algorithm to find geodesic paths. However, combinatorial approaches based on a graph problem representation can be directly used in $\mathcal{W}$ without any modifications. The costs of edges between cities are lengths of the shortest paths between cities, and the graph can be constructed from the visibility graph by Dijkstra's algorithm. Therefore, the gap between SOM and combinatorial heuristics seems to be wider for the TSP in the polygonal domain.

In [18], a simple, yet sufficient approximation of the shortest path in $\mathcal{W}$ has been used in SOM adaptation rules to decrease the computational burden. Although this approximation enables the application of SOM principles in $\mathcal{W}$, the required computational time of self-organization is still significantly higher (hundreds of times) than for the Euclidean-TSP. The main issue of the conventional SOM is the high number of node–city distance queries in the competitive phase, and also the relatively high number of node–city path queries in the cooperative phase. In this performance study, the computational requirements of these adaptation phases are examined in two different ways. First, technical aspects of the queries are considered, i.e., the required computational time is reduced by informing the competitive selection procedure and assuming practical approximations in the cooperative phase. The second way aims to decrease the number of queries considering recent adaptation rules; these reduce the required number of adaptation steps, and effectively decrease the size of the winner node neighborhood. The rules are closely related to the initialization of adaptation parameters; therefore, various initializations are considered as well.

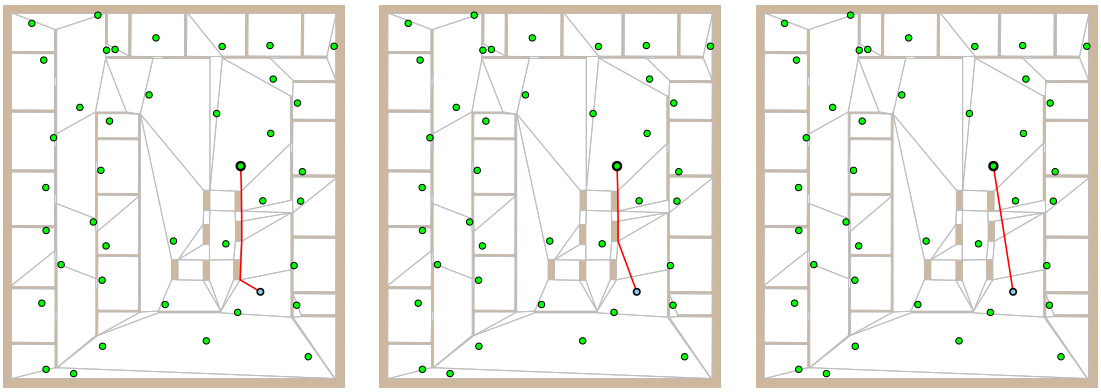
The rest of the paper is organized as follows. Section 2 describes the notation and terminology used related to the geometrical structures supporting the shortest path queries and adaptation procedure. Related work is acknowledged in Section 3. In Section 4, a brief description of the SOM adaptation procedures used herein and an approximation of the shortest path in the polygonal domain $\mathcal{W}$ is presented. Section 5 presents proposed modifications and combinations of published competitive and cooperative rules to decrease the required computational time possibly without affecting the solution's quality. Experimental results are presented in Section 6. Finally, the concluding remarks are presented in Section 7.

2. Terms used and notation

The SOM adaptation is considered in the polygonal domain, i.e., a polygonal map; therefore, a few terminology notes are presented in this section to clarify the terms used and symbols for supporting geometrical structures.

A world is represented by a polygonal map $\mathcal{W}$ consisting of N_V vertices; thus, $\mathcal{W}$ is a closed, multiply connected region, whose boundary is a union of N_V line segments, forming $h + 1$ closed polygonal cycles, where h is the number of holes (obstacles). A distance between two points inside $\mathcal{W}$ is a length of a path among obstacles that can be a straight line segment or consisting of vertices. Thus, a path between two points s and t consists of a finite number of line segments joining the points and vertices.

$\mathcal{W}$ can be divided into a set of non-overlapping convex polygons that are formed from vertices. Such convex polygons are called cells and represent a *convex polygon partition* of $\mathcal{W}$, i.e., each cell C forms a closed polygonal cycle of line segments joining vertices. A line segment is called *diagonal* if it connects two non-adjacent vertices and if it is contained in $\mathcal{W}$. A point inside $\mathcal{W}$ is always inside a cell, and a collision-free path between two points $s \in C_s$ and $t \in C_t$ can be constructed from the shortest path between vertices of C_s and C_t. Weights of the i*th* neuron represent a point v_i (called node) that lies in $\mathcal{W}$; therefore, v_i is always inside a cell. Such a cell of the node v is denoted as C_v. An example of a polygonal map, its convex partition, and a path from a node to a city is shown in Fig. 1.



(a) path over v_0 (b) path over v_1 (c) full refinement

Fig. 1. An example of path refinement, the gray segments represent diagonals of the convex partition, small disks are cities, and a node is connected with the city by the approximation of the shortest path (red segments). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The symbols used are as follows.

$\mathcal{W}$	the polygonal domain representing the working space to be inspected, $\mathcal{W} \subset \mathbb{R}^2$
N_V	the number of vertices of $\mathcal{W}$
n	the number of cities
m	the number of nodes (neurons)
$\|s,t\|$	the Euclidean distance between points s and t
$\|S(v_i,v_j)\|$	the length of the shortest path (among obstacles) between two vertices v_i and v_j in
$\boldsymbol{P}$	a set of convex polygons
v_i	a vertex of the polygonal domain $\mathcal{W}$
v_i	a node representing the weights of the ith neuron
C_v	the cell of $\boldsymbol{P}$ in which node v lies
G	the learning gain (also called the neighborhood function variance)
μ	the learning rate
α	the gain-decreasing rate
d	the number of neighboring nodes of the winner node

3. Related works

The first application of SOM principles to the TSP [1] follows constructive heuristics and starts with one node. In that approach, a node is duplicated if it is the winner for two different cities, and it is deleted if it is not selected as the winner for three complete presentations of cities to the network. Growing ring structure has also been used in FLEXMAP proposed in 1991 [20]; however, the deletion mechanism was omitted. The maximal number of required nodes has been close to $2.5n$, where n is the number of cities, up to a problem with 2392 cities. In [6], Budinich used the same number of nodes as cities, and the inhibition was replaced by a real value derived from the winner node and its neighboring nodes. The tour is constructed from an ordered sequence of cities according to the value.

An inhibition of frequently selected winner nodes has been used in the Guilty net algorithm [9]. The inhibition mechanism was substituted by the vigilance parameters in the Vigilant Net presented in [7] where the initialization of weights is discussed. Superior results are reported for starting positions of nodes as the convex hull approximation of the cities. Aras used the geometrical properties of the connected nodes forming a ring and the topology of cities in his KNIES algorithm [3]. This algorithm uses a regular adaptation of the winner node, which moves toward the city. In addition, nodes that are not in the activation bubble (set of neighboring nodes), do not move closer to the city, but move in a way that allows the preservation of the global statistical properties of the data points. The proposed algorithm has been used to solve large TSP instances by the decomposition of the problem into several clusters [4].

Probably the most complex and powerful SOM algorithm for the TSP is the Co-adaptive net introduced in [11]. This algorithm uses a higher number of nodes than the number of cities, and it utilizes an adaptive neighborhood of the winner node that is updated after each adaptation step. The learning process is divided into competition and

co-operation phases. The co-operation phase is based on winner nodes or their neighbors not moving more than once. The algorithm also uses the near-tour to tour construction, which creates a complete tour if the current winner nodes are not distinct. The best tour is kept during the adaptation, and it is used if the final solution is worse. The authors presented a huge set of results and comparisons with other approaches, and reported that their approach outperforms other variants and together with [36] provides better results than Aras's KNIES [3], which uses the statistical properties of the data points.

Even though the statistical approach (KNIES) has been outperformed, the convex hull property has been studied in the expanding SOM variant called ESOM [26]. To follow the convex hull property and to design an appropriate learning rule that considers the global parameters of the problem, Intergrated SOM (ISOM) uses an evolutionary principle and combines SOM with a genetic algorithm [22]. The convex hull property is also studied in [38], where the authors considered a more conservative learning rule than ESOM: the movement of the nodes, which follows the expansion to preserve the convex hull property is restricted. This algorithm provides almost identical results as those of ESOM, but the learning rule is much simpler.

Another research direction studied is the initialization of neuron weights and the setting of adaptation parameters: the learning rate μ, the learning gain G, and the gain-decreasing rate α. An initialization of weights was studied in [5], where authors examined four initialization methods: random, small circle around centroid of the cities, a tour found by the nearest neighborhood algorithm, and a random initialization of nodes on a rhombic frame located to the right of the cities' centroids. The fourth initialization method is reported as the most suitable technique. Kohonen's exponential evolution of the adaptation rules is studied in [37]. To reduce the number of parameters, the authors proposed simplified adaptation rules based only on the number of performed adaptation steps k. The learning rate is defined as $\mu = 1/\sqrt[3]{k}$ and the learning gain as $G = G(1 - 0.01k)$ with initial value $G_0 = m/32$, where m is the number of neurons. They used initial weights representing nodes on a rectangular frame around the cities, and the authors reported superior results in the selected TSPLIB [31] instances in comparison with the SOM approaches [19,1,9]. These simplified rules have also been applied in [39], where the authors proposed to use $\mu = 1/\sqrt[4]{k}$ and the initial value of the gain $G_0 = 10$. For small values of G, the value of the neighborhood function is very small; thus, the neighboring nodes are negligibly moved. Considering this fact, the authors recommended to gradually decrease the neighborhood of the winner node after each adaptation step. It decreases the computation burden while not affecting the quality of solution. The recommended initial size of the neighborhood is $d = 0.4$ m, which is decreased by $d = 0.98d$ at the end of each adaptation step.

In [28], Murakoshi and Sato applied multiple scale neighborhood functions to decrease topological defects that may occur during the self-adaptation. The functions have a form

$$f(G, l) = \beta_j \mu e^{-\frac{l^2}{(\gamma_j G)^2}}, \tag{1}$$

where β_j and γ_j are the gain and width factors of the j*th* neighborhood function. The authors used six functions $\beta_j = 2^{-|3-j|}$ and $\gamma_j = 2^{-(3-j)}$ for $j \in \{1,2,\ldots,6\}$. These functions have been incorporated into the SOM adaptation procedure [1], where a function has been randomly chosen during the adaptation. The authors reported up to 42.65 % less kinks than in the original version of the procedure [1].

4. Self-organizing maps for the Traveling Salesman Problem

Two state-of-the-art SOM adaptation procedures are considered in this performance study as the primal algorithms being modified. The SOM algorithms have been selected regarding the results presented in [11], where these approaches provide the best performance. Although more recent approaches increase the quality of solution, the improvement is not significant, and such approaches are also more computationally demanding. That is why Somhom's algorithm introduced in [36] and the Co-adaptive net algorithm [11] have been selected for evaluation of their performance for problems in the polygonal domain $\mathcal{W}$. The performance of these algorithms is improved by the proposed modifications and by the combination of selected approaches (briefly described in the previous section), and therefore, the original algorithms are described in more detail in the next subsections. Moreover, an overview of the used shortest path approximation in $\mathcal{W}$ is presented in Section 4.3.

4.1. Somhom's algorithm

The algorithm presented in [36] by Somhom et al. uses an inhibition mechanism, i.e., a neuron can be a winner only for one city during a single adaptation step. In the rest of this paper, Somhom's algorithm is denoted as SME. The basic schema of the algorithm is similar for both SOM algorithms considered. The schema is shown in Algorithm 1.

Appendix 1 - Faigl, J.: *On the Performance of Self-organizing Maps for the Non-Euclidean Traveling Salesman Problem in the Polygonal Domain* [27], referenced on page 9.

Algorithm Self-organizing map for the TSP

Input: $\boldsymbol{C} = \{c_1, \ldots, c_n\}$ – a set of cities
Input: (d, G, μ, α) – the parameters of SOM
Input: δ – the maximal allowable error
Input: i_{max} – the maximum number of adaptation steps
Output: $(v_1, \ldots, v_M)$ – a sequence of node weights representing city tour

```
init(v_1, ..., v_M)                          // an initial set of neurons weights
i ← 0                                        // the adaptation step counter
repeat
    error ← 0
    I ← ∅                                    // the set of inhibited nodes
    Π(C) ← a random permutation of cities
    foreach c ∈ Π(C) do
        v* ← select winner(node to c), v* ∉ I)   // call the select winner procedure
        error ← max{error, |v*, c|}
        adapt(v*, c)                         // call the adapt procedure
        I ← I ∪ {v*}                         // inhibit winner node
    G ← (1 − α)G                             // decrease the gain
    i ← i + 1                                // increment the adaptation step counter
until error < δ or i ⩾ i_max
```

The algorithm works as follows. The ring of nodes is initialized as a small ring around one of the cities. The adaptation procedure consists of a sequence of adaptation steps in which all cities are randomly presented to the neural network. For each presented city, the winner node is selected according to $v^\star = \text{argmin}_v |c, v|$, where $|\cdot,\cdot|$ denotes the Euclidean distance between the city c and the node v for the Euclidean TSP. The adaptation procedure (`adapt`) moves the winner node and its neighboring nodes toward the presenting city c according to the rule $v'_j = v_j + \mu f(G, l)(c - v_j)$, where μ is the learning rate. The neighboring function is $f(G, l) = exp(-l^2/G^2)$ for $l < d$ and $f(G, l) = 0$ otherwise, where G is the gain parameter, l is the distance in the number of nodes measured along the ring, and d is the size of the winner node neighborhood that is set to $d = 0.2m$, where m is the number of nodes. The initial value of G is set proportionally to the problem size $G_0 = 0.06 + 12.41n$. The values of learning and decreasing rates are $\mu = 0.6$ and $\alpha = 0.1$, respectively.

The original termination condition is based only on the maximal distance of a winner node to the city that is less than a given δ. However, in the case of poor convergence, e.g., due to the used approximation, the adaptation procedure is terminated after a given number of adaptation steps i_{max}. The city tour can be reconstructed from the ring of nodes because each city has a distinct winner. The used value of acceptable error is $\delta = 0.001$, and the used maximal number of steps is $i_{max} = 180$.

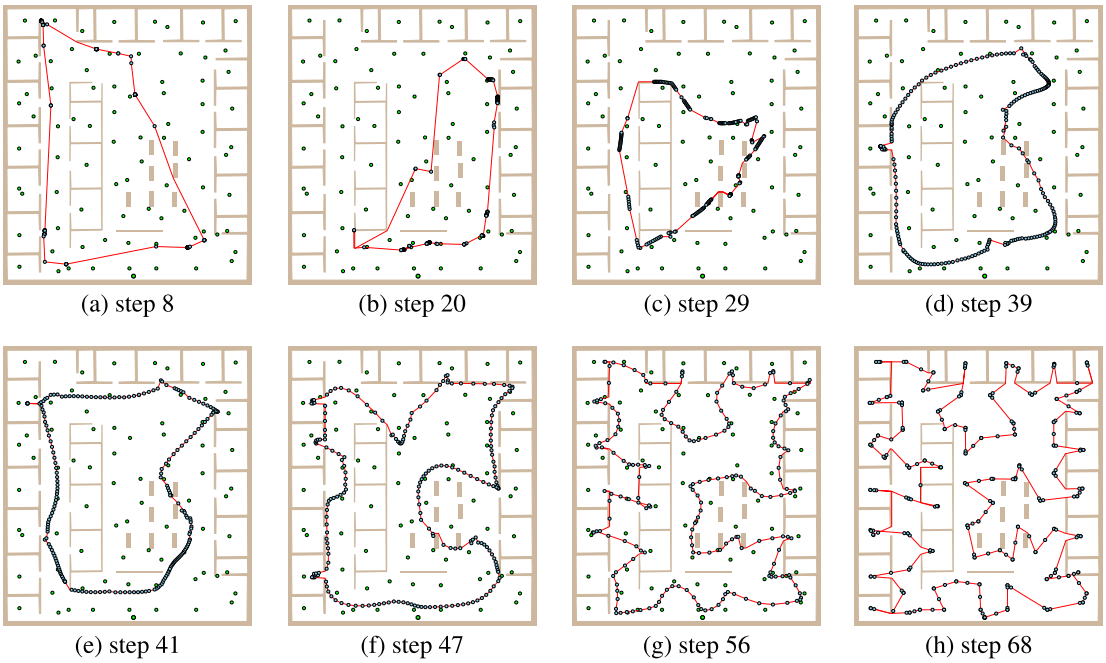
4.2. Co-adaptive net

The Co-adaptive net algorithm [11] also uses a randomization of the presented cities, but it does not use the inhibition mechanism. Instead, the winning number w_i is maintained for each neuron during the adaptation step. The required computational time of the select winner procedure is decreased by considering the restricted set of neighboring nodes of the previous winners. To avoid degenerate solutions, after every K adaptation steps, the winner is selected from the whole set of nodes.

One of the two adaptation procedures is selected according to the value of the gain G. In the case of $G < G_{cross}$, the winner node $v^\star$ and its neighboring nodes are moved toward the city if and only if $w_{\star} = 0$. For $G \geqslant G_{cross}$, the winner and the neighbors (for $w_{\star} = 0$), or only the neighbors (for $w_{\star} = 1$) are moved; otherwise, none of nodes is adapted. The neighboring function is similar to the one used in the SME algorithm, but a node-specific gain is used $g_i = G(1 - |v_i, c|/\sqrt{2})$. The gain G is changed after each adaptation step by $G \leftarrow (1 - \alpha)G$ for $G \leqslant G_{cross}/2$, and $G \leftarrow (1 - 2\alpha)G$ otherwise.

Another important part of the Co-adaptive net is a construction of the city tour because the inhibition is not used. If a tour constructed from the winner nodes with $w_i = 1$ contains at least $\min\{n - 100, 0.98n\}$ cities after an adaptation step, winner nodes are found for the cities not in the tour considering the inhibition. The city tour may be constructed after each adaptation step, and the best-found tour (the shortest one) is returned as the solution of the TSP.

The adaptation is terminated if the winner nodes are sufficiently close to the cities (similar to the SME approach), if the neurons are in the same positions as they were at the end of the previous adaptation step, or if the current gain is small ($G \leqslant 0.01$), which is equivalent to the maximal number of adaptation steps.



(a) step 8 (b) step 20 (c) step 29 (d) step 39

(e) step 41 (f) step 47 (g) step 56 (h) step 68

Fig. 2. An example of ring evolution in the environment *jh*; the small green disks represent cities, and the blue disks are nodes.

Based on the results presented in [11] the following parameters are used as default settings of the Co-adaptive net in this paper: $m = 2.5n$, $G_0 = n/3$, $G_{cross} = 10$, $\alpha = 0.02$, $\mu = 0.626$, the size of the restricted set of nodes in the `select winner` procedure is $C^{\star} = 250$, the full search is performed after every 10 adaptation steps, and the size of the neighborhood in the `adapt` procedure is set to $\min\{2G + 1, 200, m/2\}$.

The authors of the Co-adaptive net use the center of the cities as the point around which a ring is initialized. Such a point can lie in an obstacle for the TSP in $\mathcal{W}$; therefore, alternative initializations are considered.

4.3. Approximation of the shortest path in the polygonal domain

A simple approximation of the shortest path based on a convex partition of the polygonal domain $\mathcal{W}$ has been used in [18]. The approximation is based on a refinement of a primary path found in a convex partition of $\mathcal{W}$. A convex partition $\boldsymbol{P}$ is a set of convex cells C_i, $\boldsymbol{P} = \{C_1, C_2, \ldots, C_k\}$ such that the union of the cells is $\mathcal{W}$, $\bigcup_{i=1}^{k} C_i = \mathcal{W}$. Cells are induced by the diagonals of $\mathcal{W}$, and each cell is formed from a sequence of $\mathcal{W}$ vertices. A node v is inside $\mathcal{W}$ during the adaptation; thus, it is always inside some cell C_v. The initial approximate path from v to the city c is found as the shortest path $S(w, c)$ over vertex w of C_v to c such that $w = \text{argmin}_{w_i \in C_v} |v, w_i| + |S(w_i, c)|$, where $|\cdot,\cdot|$ denotes the Euclidean distance between two points, and $|S(.,.)|$ is the length of the shortest path between two vertices, or vertex and city, see Fig. 1(a).

The problem of finding the cell C_v is a point-location problem, which can be solved in $O(\log v)$ or in the average complexity $O(1)$ by the "bucketing" technique [14]. Alternatively, the cell can be determined during the node movement toward the city by the walking technique similar to [13]. The complexity of such cell determination is bounded by $O(\log n_d)$, where n_d is the number of passed diagonals of the used convex polygon partition.

The initial path can be improved by the following refinement procedure. Assume a node v inside the cell C_v and the approximation of the path from v to the vertex v_k as a sequence of vertices $(v_0, v_1, \ldots, v_k)$, $v_0 \in C_v$. A refinement is an examination of a direct visibility test between v and v_i. The visibility test is similar to [23], a convex partition is used instead of a triangulation. If a straight line from v to the vertex v_k crosses only diagonals or lies entirely in the same cell, then the vertex v_k is visible, and all vertices v_i for $i < k$ can be removed from the sequence. Examples of a refined path are shown in Fig. 1.

The shortest paths from vertices to cities can be pre-computed by Dijkstra's algorithm in $O(n + e \log(N_V + n))$, where n is the number of cities, N_V is the number of vertices, and e is the number of visible pairs (city-city, city-vertex, and vertex-vertex) of the complete visibility graph. The number of edges is bounded by $e \leqslant N_V + N_V n$. The graph can be found in $O((N_V + n)^2)$ by the algorithm [29].

The adaptation process using the approximate path is visualized in Fig. 2. The nodes are connected by the approximate shortest paths between two nodes, which uses the same principle as the node–city paths, the vertices of the nodes' cells are considered. The path between nodes is not needed in the adaptation process; it is used only for the visualization.

5. Modifications used and proposed

5.1. Approximation of the shortest path

Three variants of the refinement procedure of the approximate shortest path described in Section 4.3 have been considered in the experimental evaluation of the modified SOM algorithms. The refinement using only one vertex of the primary path over the vertex of the node cell is denoted as the *va*-1 variant. Two additional variants are *va*-0, which does not use the refinement procedure, and *pa*, which represents the complete refinement of all vertices on the primary path.

Based on the results presented in [18], the *va*-1 variant provides the best trade-off between the quality of the solutions and the required computational time. The *va*-0 variant is faster, but the network does not converge in some cases due to imprecise approximations.

5.2. Select winner procedure

A path among obstacles in $\mathcal{W}$ has to be found to determine the winner node of the current presented city to the network, which means m node–city distance queries have to be performed for each presented city. However, the required computational time can be reduced if the Euclidean distance of the node to the city is considered before the node–city distance is queried. If the Euclidean node–city distance is longer than the Euclidean distance of the current winner candidate, it is not necessary to determine the path among obstacles. This Euclidean pre-selection is denoted as the *euclid-pre* select winner method in the experimental part of this paper.

Moreover, after several adaptation steps, the winners are preserved over the steps. Thus, the previous winner to the city can be used as the initial winning candidate. Such an initial selection of the winner candidate can avoid unnecessary computations of the shortest path. In the final adaptation steps, winners are very close to cities, and a city and its winner node are typically in the same cell; in other cases, the shortest path can be just a straight line segment. Therefore, the determination of node–city distance can be very fast, and the Euclidean distance is sufficient to confirm that the previous winner is really the closest node to the city. This winner selection method with the Euclidean distance pre-selection is denoted as *informed*.

These improvements can be considered technical, because they do not affect the quality of the solution found and only decrease the required computational time at the cost of a more complex algorithm.

5.3. Adaptation rule

The `adapt` procedure is more complex than the `select winner` procedure because a path has to be retrieved in the node–city path query and the adapted node is moved towards the city, i.e., a particular straight line segment of the path has to be determined. The node v is moved closer to the city c proportionally to the node–city distance $D(v,c)$, learning rate, and neighboring function. The distance of v to c is decreased about $\beta D(v,c)$, where β has the form $\beta = \mu exp(-l^2/G^2)$. The value of β decreases with the increasing distance of the neighboring node. It also decreases with each adaptation step, as the learning gain G decreases. If β is very small, the movement can be negligible; therefore, once the β is under a given threshold, the adaptation of neighboring nodes can be omitted. This modification of the adaptation rule is called *β-condition* in this paper, and it can be used for rules without decreasing the neighborhood size. The influence of this modification has been experimentally examined for the SME and Co-adaptive net algorithms.

An additional speed improvement of the `adapt` procedure can be based on the usage of the winner path to the city c for the neighboring nodes. If nodes are close to each other, and if a path contains a map vertex (avoiding an obstacle), a path from the neighboring nodes will likely pass the same vertex. Thus, the neighboring node v can be moved along the same path as the winner node $v^{\star}$, while the distance is decreased by the Euclidean distance between $v^{\star}$ and v, i.e., v is placed at the position of $v^{\star}$ before its movement and adaptation toward c. The situation is schematically shown in Fig. 3. This modification of the adaptation rule is called *approx. adapt*, and it is combined with the *β-condition* modification.

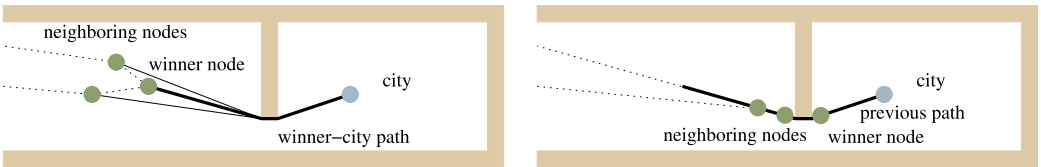
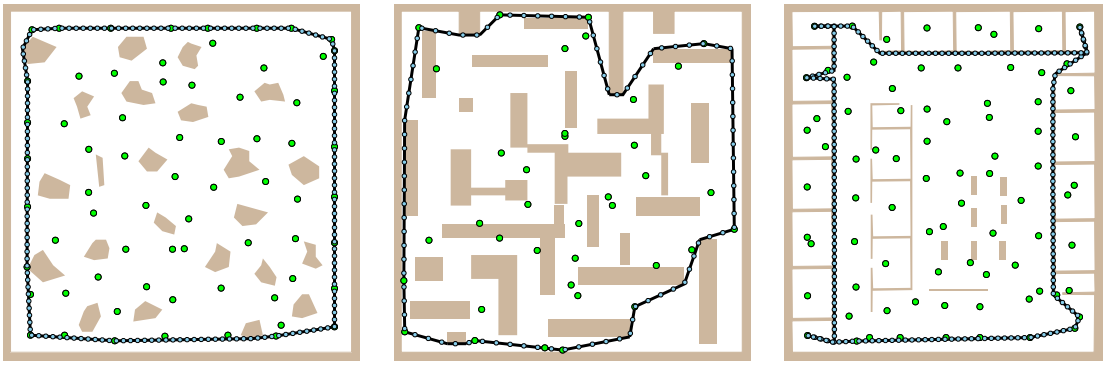


Fig. 3. Utilization of the winner–city path for the neighboring nodes.



(a) *potholes* (b) *warehouse* (c) *jh*

Fig. 4. Examples of the *hull* initialization, the green disks are cities, small blue disks are nodes, and the bold black line segments represent a connected ring of nodes.

5.4. Adaptation parameters

Beside the original adaptation parameters of the SME and Co-adaptive net algorithms, the following modifications are considered as well. The rules proposed in [39] and denoted as *Zhang–Bai–Hu* rules are also used in SME. Furthermore, the original SME adaptation rule is complemented by the decreasing size of the winner node neighborhood, i.e., the size d is updated to $d = 0.98d$ at the end of each adaptation step.

The multiple scale neighborhood functions used by Murakoshi and Sato in [28] are utilized in the Co-adaptive net; this modification is denoted by the abbreviation MSNF for short.

5.5. Initialization

Due to obstacles, the initialization of the nodes used for the Euclidean TSP described in [5] cannot be directly used in the polygonal domain $\mathcal{W}$. That is why the following initializations are considered in the experimental examination of the modified algorithms.

The first initialization method is called *first* because the first city is used to initialize the ring of nodes as a circle with a small radius (5 mm) around the city. The small radius ensures that the nodes are placed in $\mathcal{W}$, as cities are always placed at a greater distance from the obstacles.

The second method uses the closest city to the centroid of cities, and a ring is also created as a small circle with the same radius like in the first method. The method is called *center* in this paper.

The third initialization is similar to the *center* method, but the center of the circle is selected as the city with the smallest standard deviation of the distances to other cities. The method is called *dev*.

Inspired by approach [7], the last examined initialization method is called *hull* because it is based on the convex hull of the cities. The cities at the border of the convex hull are connected by the shortest paths. The connected cycle is then used to initialize nodes equidistantly along the cycle. Examples of the *hull* initialization are shown in Fig. 4.

6. Performance evaluation

The performance of the SOM algorithms is evaluated for a set of inspection planning problems.[1] The environments are represented by polygonal maps. The name of the environment with a subscript denoting the visibility range ρ in meters represents the particular TSP. It means that a robot performing measurement at the city position senses its surrounding environment in the distance ρ [17]. Parameters of the environments are shown in Table 1, where N_V is the number of vertices, N_H is the number of holes, and N_C is the number of convex cells of the supporting convex partition. Environments *jh*, *pb*, *ta*, and *h*2 represent maps of real buildings; thus, they provide a representative problem in size. In particular, maps *jh*, *pb*, and *ta* have been used as experimental sites for search and rescue scenarios in the PeLoTe project [24].

The algorithms have been implemented in C++ and compiled by the G++ 4.2 with the −O2 optimization flag. All results have been obtained within the same computational environment using single core of the Athlon X2 CPU running at 2 GHz, 1 GB RAM, and FreeBSD 8.1. Thus, all required computational times presented can be directly compared.

The cities are applied to the network in a random order in all examined algorithms, and therefore, each particular algorithm variant is executed twenty times for each problem, and average values are determined. The quality of solutions is eval-

[1] The problems with necessary supporting structures are available at http://purl.org/faigl/tsp/.

Table 1
Testing environments with obstacles.

Name	Dimensions [m × m]	Area [m^2]	N_V	N_H	N_C
jari	4.5 × 4.9	20	48	1	14
complex2	20.0 × 20.0	322	40	3	21
m1	4.8 × 4.8	20	51	4	26
m2	4.8 × 4.8	15	51	6	20
map	4.8 × 4.8	14	68	8	36
potholes	20.0 × 20.0	367	153	23	75
rooms	20.0 × 20.0	351	80	0	33
a	8.9 × 14.1	71	99	6	22
dense	21.0 × 21.5	299	288	32	150
m3	4.8 × 4.8	17	308	50	120
warehouse	40.0 × 40.0	1192	142	24	83
jh	20.6 × 23.2	455	196	9	77
pb	133.3 × 104.8	1453	89	3	41
ta	39.6 × 46.8	731	74	2	30
h2	84.9 × 49.7	2816	2062	34	476

uated as the percentage deviation to the optimum tour length of the mean solution value, $PDM = (\bar{L} - L_{opt})/L_{opt} \cdot 100\%$, and as the percentage deviation from the optimum of the best solution value (PDB), where L_{opt} is the length of the optimal solution found by the Concorde solver [2]. The PDM and PDB have variances due to randomization, therefore a tolerance between a half and one percent is considered in the quality evaluation of solutions found by the particular modified algorithm.

The speed improvement of a particular algorithm variant is measured as the ratio of the average required computational times of the original algorithm and its modified variant. The required computational time consists of the preparation time T_{init} and the time needed to adapt the network T_{adapt}. The preparation phase is a creation of supporting structures: the convex polygon partition, visibility graph, and shortest paths between cities and map vertices. The convex partition is found in tens of millisecond using Seidel's algorithm [33], and the construction of the complete visibility graph takes 41 ms for the largest problem $h2_2$ with 575 cities and 2062 map vertices. These times are negligible according to the total required computational time, and they are not included in the presented time values. The most time consuming preparation part is determination of the shortest path between cities and vertices. This time is included in the presented total required computational time denoted as T. Regarding the preparation time the speed improvement of a particular algorithm variant is computed from T_{adapt}.

The adaptation procedure itself is composed of the selection of winners and adaptation toward cities. The particular required computational times in these parts are useful for determining the most computationally intensive part of the algorithm. Therefore, $\%T_s$ and $\%T_a$ denote computational times spent in the particular part of the adaptation procedure (`select winner` and `adapt` respectively) in percentages of the total adaptation time T_{adapt}.

To avoid presentation of many detailed results, the examined problems are organized into three sets according to the number of cities, see Table 2. In the overall comparison of the examined algorithms' modifications, T_{adapt} for the original algorithm is the reference computational time, i.e., an average computational time for each problem of the reference algorithm is divided by the average T_{adapt} for the algorithm variant. The speed improvement, denoted as *Sp.*, is computed as an average value of improvements over all problems in the set.

6.1. The SME algorithm

The original Somhom's adaptation procedure has been augmented by the algorithm to find the approximate shortest path in $\mathcal{W}$. The *pa* refinement variant and the pure *geodesic* winner selection are used. Besides, the tour length at each adaptation step is computed, and the best tour found during the adaptation is used as the found solution. This algorithm variant is used

Table 2
Problems sets.

Small set		Middle set		Large set	
problem name	n	problem name	n	problem name	n
jari	6	$dense_4$	53	$potholes_1$	282
complex2	8	$potholes_2$	68	jh_1	356
m1	13	$m3_1$	71	$pb_{1.5}$	415
m2	14	$warehouse_4$	79	$h2_2$	568
map	17	jh_2	80	ta_1	574
potholes	17	pb_4	104		
rooms	22	ta_2	141		
a	22	$h2_5$	168		

Table 3
The SME algorithm – improvements of the `select winner` procedure with the *va*-1 refinement.

Problems	select winner – geodesic				select winner – euclid-pre				select winner – informed			
	PDM	PDB	Sp.	Steps	PDM	PDB	Sp.	Steps	PDM	PDB	Sp.	Steps
small	1.71	0.00	1.2	64	1.10	0.01	2.0	64	1.36	0.01	2.0	64
middle	4.94	2.18	1.3	84	4.77	2.00	2.1	84	4.61	2.38	2.1	84
large	3.94	2.99	1.3	100	4.16	3.04	2.2	100	4.00	3.05	2.1	100

Table 4
The SME algorithm – adaptation rule modifications.

Problems	original adaptation rule				β-condition modification				approx. adapt.+β-condition			
	PDM	PDB	Sp.	Steps	PDM	PDB	Sp.	Steps	PDM	PDB	Sp.	Steps
small	1.36	0.01	2.0	64	1.29	0.01	2.4	64	1.62	0.01	2.4	64
middle	4.61	2.38	2.1	84	4.58	2.46	3.3	84	5.23	2.86	3.2	84
large	4.00	3.05	2.1	100	4.03	2.87	4.0	100	4.65	3.63	3.9	100

as the reference algorithm in the presented experimental results of SME algorithm modifications. This variant is used as the base algorithm for other examined modifications as follows.

First, the select winner methods described in Section 5.2 have been considered with the *va*-1 refinement variant, the results are presented in Table 3. The *Sp.* column shows how many times the performance of the algorithm has been improved in comparison to the reference algorithm with the *pa* refinement and the pure *geodesic* winner selection. In the case of the *geodesic* winner selection, the algorithm spent about forty five percentage points in the select winner procedure, and about fifty five percentage points in the `adapt` procedure. After applying the Euclidean pre-selection of winner node candidates, the dominant algorithm part is the `adapt` procedure. Consideration of the previous winner does not significantly reduce the required computational time, and the results are pretty much similar to the *euclid-pre* variant. The PDM variances of the select winner methods are below 0.5 % threshold; thus, the overall quality of solutions is considered to be same.

The most time consuming part of the SME algorithm with the *informed* `select winner` procedure is the `adapt` procedure, therefore, modifications of the procedure have been examined. The experimental results with modified adaptation rules described in Section 5.3 are presented in Table 4. The *informed* select winner method and the *va*-1 refinement are used, and the *β-condition* is set to $\beta = 10^{-5}$. The *β-condition* effectively decreases the active neighborhood of the winner node, which decreases the required computational time without noticeable degradation of the solution quality. The *approx. adapt* modification does not provide any improvements, and the solution quality is also worse. In all cases, solutions are found in the same average number of the adaptation steps, see the column *Steps*.

Additional speed improvement is achieved using modifications of adaptation parameters described in Section 5.4. The experimental results are shown in Table 5. For the *small* and *middle* sized problems decreasing the size of the winner node neighborhood leads to an algorithm three times faster, but the solution quality is worse and a higher number of adaptation steps is needed. The results indicate that for these problems the restriction of the neighborhood is too strong, mainly at the beginning of the adaptation. However, for *large* problems the initial number of nodes seems to be sufficiently high (possibly unnecessarily high), as the solution quality is preserved. The Zhang–Bai–Hu rules dramatically reduces the required computational time for problems with a higher number of cities, although the solution quality is more than two times worse for larger problems.

The poor solution quality of the Zhang–Bai–Hu rules is improved by the *hull* initialization, see results for the path refinement variants presented in Table 6. The most significant improvement in the solution quality and also in the required computational time is for large problems. Other initialization methods do not increase the solution quality, which is also the case for other examined modifications of the SME algorithm. The significant reduction of the number of the shortest path queries allows consideration of the full path refinement variant, although the benefit is not evident from the presented results. The algorithm with the modifications used has the `select winner` and `adapt` parts almost equally computationally intensive regarding $\%T_s = 46$ % and $\%T_a = 49$ % for large problems.

Table 5
Influence of the adaptation parameters to SME with the *va-1*, *informed*, and *β-condition* modifications.

Problems	original adapt. param.			orig. with decreasing d				Zhang-Bai-Hu rules			
	PDM	Sp.	Steps	PDM	PDB	Sp.	Steps	PDM	PDB	Sp.	Steps
small	1.29	2.4	64	4.53	0.36	8.1	163	5.71	0.13	9.1	160
middle	4.58	3.3	84	8.16	4.36	7.6	168	6.15	2.52	21.3	71
large	4.03	4.0	100	3.99	3.14	6.5	100	9.21	4.17	86.4	49

Table 6
SME with the *va*-1 refinement, *informed*, *β-condition* modifications, Zhang-Bai-Hu rules, and *hull* initialization.

Problems	va-0 refinement			va-1 refinement				pa refinement			
	PDM	Sp.	Steps	PDM	PDB	Sp.	Steps	PDM	PDB	Sp.	Steps
small	6.03	9.8	177	5.62	0.08	8.3	166	5.96	0.16	8.3	160
middle	6.93	30.8	89	5.00	2.60	22.2	63	4.80	2.86	20.7	65
large	12.55	116.5	64	4.44	3.51	105.7	43	4.33	3.02	99.4	43

Table 7
Influence of adaptation rule modifications to the Co-adaptive net with the *va*-1 refinement and *informed* select winner procedure.

Problems	Original adaptation rule				*β*-condition				*β*-condition + MSNF			
	PDM	PDB	Sp.	Steps	PDM	PDB	Sp.	Steps	PDM	PDB	Sp.	Steps
small	1.49	0.46	1.1	174	1.33	0.11	1.1	175	1.22	0.07	1.1	196
middle	6.28	3.05	1.2	290	6.59	3.20	1.2	290	4.56	2.13	1.0	311
large	6.60	4.89	1.2	388	6.64	4.87	1.2	389	5.39	4.10	1.0	390

The additional speed improvements can be achieved by a more restricted size of the winner node neighborhood. However, using initial size $m/8$ only decreased the solution quality without significant speed improvement. The multiple scale neighborhood functions do not improve the solution quality; thus, these results are not presented.

6.2. The co-adaptive net algorithm

The *informed* `select winner` procedure with the *pa* refinement is used in the Co-adaptive net algorithm. Even though authors of the Co-adaptive net algorithm initialized the ring as a small circle around cities' centroid, the four initialization methods described in Section 5.5 do not provide significant differences in the solution quality nor the computational requirements, and therefore, the *first* initialization method is used as default. This algorithm variant is the reference algorithm (of the required computational time) in the overall comparisons of its examined modifications.

Similarly to the SME algorithm, the *va*-1 refinement does not provide noticeable changes in the solution quality in comparison with the full path refinement; however, the performance is improved by about more than ten percentage points. The original adaptation rule is modified to consider the *β-condition*, then the rule is combined with the Multi Scale Neighborhood Functions (MSNF). The results for the three problems sets are presented in Table 7. Notice, the Co-adaptive net requires a higher number of adaptation steps than the SME algorithm. However, the total required computational time is lower because less nodes are involved in the `select winner` and `adapt` procedures. Considering MSNF increases the solution quality and the number of required adaptation steps. Here, it should be noted that the network adaptation has been terminated by the $G < 0.01$ condition in all algorithm variants. The minimal distance of the winner node to the city is significantly higher than in the SME algorithm, i.e., by units or tens in comparison to Somhom's $\delta = 0.001$.

The used gain-decreasing rate $\alpha = 0.02$ is relatively small, and the computational burden can be decreased by a higher value. The experimental results for various α are presented in Table 8. The original Co-adaptive net algorithm is very sensitive to changes of α while MSNF provides significantly better results. The value $\alpha = 0.1$ provides almost the same solution quality (about one or two percent worse) as the original algorithm, and it is more than four times faster. Also in this case, another initializations of the ring do not provide any significant improvements.

6.3. Algorithms comparison

Based on the examination of described modifications two new variants of the SME and the Co-adaptive net algorithms are selected as successors of their originals. The applied modifications are selected as the best trade-off between the solution quality and required computational time, mainly concerning the $h2_2$ problem. Particular parts of the original algorithms and the proposed modifications are as follows.

Table 8
An influence of the gain-decreasing rate α to the Co-adaptive net with the *va*-1 refinement and *informed* select winner procedure.

Problems	original adaptation rule						*β*-condition + MSNF					
	$\alpha = 0.05$		$\alpha = 0.1$		$\alpha = 0.2$		$\alpha = 0.05$		$\alpha = 0.1$		$\alpha = 0.2$	
	PDM	Sp.	PDM	Sp.	PDM	Sp.	PDM	Sp.	PDM	Sp.	PDM	Sp.
small	2.59	2.6	3.49	4.8	4.31	8.8	2.03	2.6	2.59	4.6	4.47	8.6
middle	7.93	2.9	8.36	5.6	10.10	11.4	5.30	2.6	6.58	5.1	7.87	10.0
large	7.43	3.0	10.27	5.8	23.05	12.4	6.08	2.5	6.94	5.1	8.87	10.4

Appendix 1 - Faigl, J.: *On the Performance of Self-organizing Maps for the Non-Euclidean Traveling Salesman Problem in the Polygonal Domain* [27], referenced on page 9.

Table 9
Proposed modifications of the original algorithms.

A part of the adaptation procedure	Modified SME	Modified co-adaptive net
Initialization	*hull*	*center*
Select winner	*informed*	*informed*
Adaptation rule	*β-condition*	*β-condition*
Neighbourhood function(s)	–	MSNF
Adaptation parameters/rule	Zhang–Bai–Hu	$\alpha = 0.1$

Table 10
The algorithms performance.

Name	n	L_{opt}[m]	The SME algorithm						The Co-adaptive Net					
			original			*modified*			*original*			*modified*		
			PDM	PDB	T [s]	PDM	PDB	T[s]	PDM	PDB	T[s]	PDM	PDB	T[s]
jari	6	13.6	0.00	0.00	0.02	1.06	0.00	0.01	0.00	0.00	0.03	0.10	0.00	0.01
complex2	8	58.5	0.00	0.00	0.04	8.77	0.00	0.01	0.00	0.00	0.05	1.30	0.00	0.01
m1	13	17.1	0.00	0.00	0.09	4.64	0.00	0.02	0.05	0.00	0.07	0.63	0.00	0.02
m2	14	19.4	8.64	5.32	0.10	13.12	0.00	0.02	2.25	0.00	0.08	6.97	0.00	0.03
map	17	26.5	1.98	0.00	0.17	10.31	0.00	0.04	3.22	0.00	0.14	3.81	0.00	0.05
potholes	17	88.5	1.11	0.00	0.31	3.65	0.00	0.10	0.94	0.00	0.24	1.99	0.00	0.11
a	22	52.7	0.01	0.00	0.38	1.95	0.00	0.06	0.30	0.00	0.22	0.79	0.00	0.08
rooms	22	165.9	0.60	0.08	0.36	4.18	1.27	0.06	3.47	1.02	0.23	4.14	2.26	0.06
$dense_4$	53	179.1	15.40	9.12	2.95	12.14	7.28	0.53	10.22	4.76	1.29	12.48	8.08	0.51
$potholes_2$	68	154.5	5.21	3.37	4.51	5.55	3.54	0.42	4.97	1.54	1.39	7.08	4.19	0.42
$m3_1$	71	39.0	6.23	4.25	5.05	6.88	4.72	0.69	8.89	3.63	1.85	8.86	5.54	0.69
$warehouse_4$	79	369.2	5.26	2.19	5.54	5.69	3.27	0.49	7.16	2.76	1.66	7.23	2.60	0.48
jh_2	80	201.9	1.63	0.28	6.26	1.82	0.43	0.51	5.50	3.58	1.80	3.68	2.26	0.54
pb_4	104	654.6	1.00	0.02	7.24	0.70	0.04	0.37	0.84	0.15	2.26	1.58	0.10	0.54
ta_2	141	328.0	3.22	2.33	13.44	3.33	2.43	0.46	5.60	3.73	3.69	4.31	2.44	0.86
$h2_5$	168	943.0	1.70	0.91	46.58	2.26	1.17	5.73	7.10	4.54	15.29	4.08	2.55	7.14
$potholes_1$	282	277.3	5.97	3.93	89.42	6.55	4.00	1.80	7.22	4.89	17.44	8.82	7.25	4.66
jh_1	356	363.7	3.97	3.03	140.75	4.50	3.06	2.39	6.59	3.59	27.01	6.51	4.49	7.26
$pb_{1.5}$	415	839.6	2.10	1.32	133.99	1.84	1.38	2.07	2.96	1.68	24.97	3.31	2.34	6.07
$h2_2$	568	1316.2	2.29	1.20	516.90	2.79	1.74	12.35	9.11	7.30	89.89	6.41	4.32	28.29
ta_1	574	541.1	5.48	4.24	259.38	5.99	4.93	3.68	6.85	4.72	39.14	8.65	7.21	9.70

The full path refinement *pa* and the pure *geodesic* variant of the `select winner` procedure are used in the original SME algorithm. The nodes are initialized around the first city. The *pa* refinement is also used in the proposed successor of the SME algorithm, as the computational burden is only slightly increased in comparison with the *va*-1 variant. The *informed* `select winner` procedure, the *hull* initialization method, *β-condition*, and the Zhang–Bai–Hu adaptation rules are utilized. In both Co-adaptive net algorithms, the *informed* modification of the `select winner` procedure with the *pa* path refinement are utilized. The initial size of the winner neighborhood is set to $m/2$. In the case of the original Co-adaptive net, the parameters described in Section 4.2 are used, and nodes are initialized by the method *dev*, which provides the highest solution quality for large problems. Nodes are initialized by the *center* initialization method in the modified Co-adaptive net algorithm that uses the *β-condition* with MSNF. The only changed parameter is the gain-decreasing rate $\alpha = 0.1$, which decreases the computational burden without significant solution quality changes. The proposed modifications of the particular part of the adaptation procedures are depicted in Table 9, where '-' denotes the original part the algorithm.

Detailed performance results of the original and the modified algorithms are presented in Table 10. To provide an overview of the algorithms' performance, average values of the required computational time and the solution quality measured by the PDM are shown in Fig. 5 as histograms of the problem size. Selected solutions found by the modified SME algorithm are presented in Fig. 6.

Regarding the presented results the modified SME algorithm provides superior results for *middle* and *large* problem sets. For problems with less than fifty cities the modified Co-adaptive net algorithm provides better results. This can be caused by a fewer number of neighboring nodes used in the SME algorithm in comparison to the Co-adaptive net algorithm. Besides, the Co-adaptive net uses the winning number, and the algorithm avoids adaptation of the winners and neighboring nodes, which means the nodes are moved with less frequency than in the SME algorithm. The performance of the Co-adaptive net algorithm in the examined large non-Euclidean TSP is quite surprising. Even though several modifications and parameter settings have been used, the algorithm does not provide competitive results to the modified SME algorithm. The original SME algorithm provides solutions with significantly higher quality, which is not the case of the TSPLIB problems presented in [11]. The applied modifications to Somhom's algorithm significantly decrease the required computational time, and make *penalization* by the shortest path determination less important. From a certain point of view, the modified algorithm in the non-

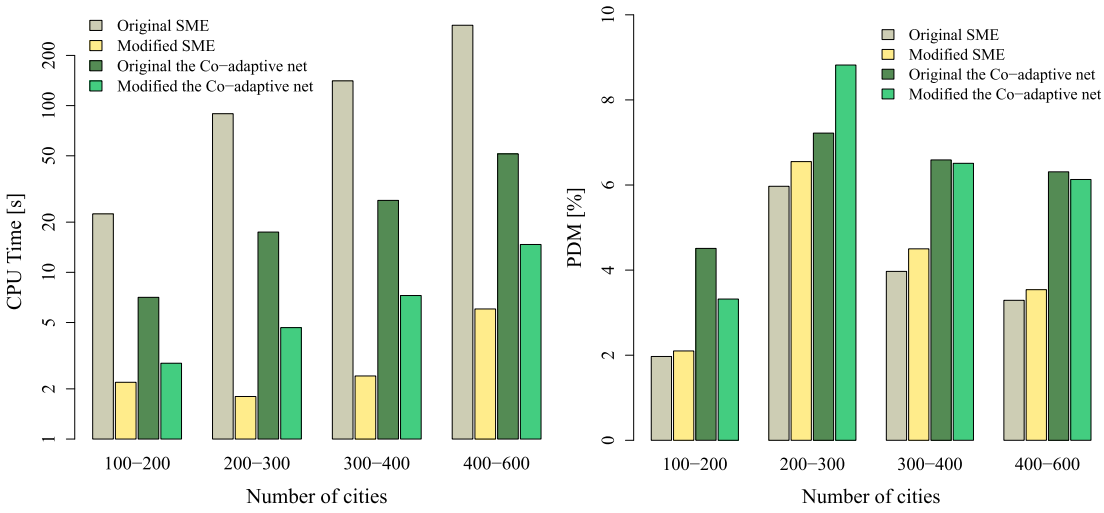


Fig. 5. Average values of the required computational time and the solution quality.

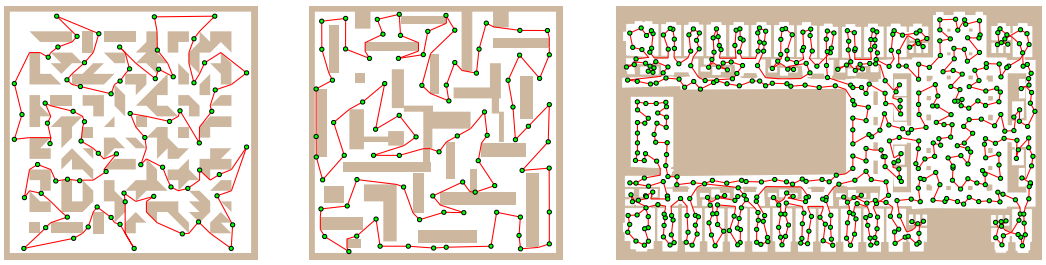
Euclidean problem starts to be competitive to the original algorithm in the Euclidean problem, e.g., according to results presented in [15].

6.4. Modified SME algorithm discussion

A detailed insight into the performance results of the modified SME algorithm gives several interesting observations, as is shown in Table 11. The worst performance of the algorithm in the *small* problems is related to the poor convergence as can be seen from the number of required adaptation steps. The presented results are average values over twenty runs; thus, the problems with 180 steps do not converge at all. The reason for this may be an excessively small value of the learning gain G together with the decreasing size of the neighborhood. However, the final found route (the length is denoted as L_{best}) is found very early, in S_{best} steps. So, worse solutions are found in the consecutive steps. The tour found in the last step is about units of percentage points worse than the best found tour, which is indicated in the column PDM_{last}. Although this is not the expected behaviour, the computational requirements are lower than for the original algorithm. These results indicate further potential improvements of the algorithm.

The minimal distance of winners to the cities is also affected by the poor convergence. Notice the *Error* values are in centimeters, due to default units of the maps used. Even though this error is not too important in the combinatorial TSP, as the solution can be considered as a sequence of cities, the error is crucial in other routing problems in the polygonal domain, e.g., the watchman [16] or safari route problem [15]. The difference is that in these problems, the ring may be the route itself, and not a representation of a route over cities. Here, it is worth mentioning that for the Co-adaptive net algorithm the error is negligible for *small* problems, and equals to tens of centimeters for larger problems.

The columns T, T_{init}, and T_{adapt} show the total required computational time, and particular times spent in the initialization and adaptation procedures. All shortest paths between cities and also from all map vertices to the cities are determined in



(a) $m3_1$, L=40.9 m (b) $warehouse_4$, L=381.3 m (c) $h2_2$, L=1339.1 m

Fig. 6. Selected solutions found by the modified SME algorithm, the small green disks represent cities that are connected by the shortest path among obstacles using the complete visibility graph.

Table 11
Detail performance results of the modified SME algorithm.

Name	n	L_{opt}	L_{best}	Steps	S_{best}	PDM	PDM_{last}	Error	T	T_{init}	T_{adapt}	T_{LK}
		[m]	[m]					[cm]	[s]	[s]	[s]	[s]
jari	6	13.6	13.70	97	1	1.06	2.87	5.7860	0.01	0.006	0.004	0.001
complex2	8	58.5	63.59	164	1	8.77	11.09	46.8591	0.01	0.004	0.008	0.001
m1	13	17.1	17.85	124	4	4.64	6.06	3.3118	0.02	0.007	0.009	0.001
m2	14	19.4	21.99	172	5	13.12	14.75	21.1784	0.02	0.007	0.014	0.001
map	17	26.5	29.26	180	6	10.31	11.79	18.0338	0.04	0.014	0.019	0.001
potholes	17	88.5	91.78	180	7	3.65	4.01	61.3482	0.10	0.074	0.019	0.001
a	22	52.7	53.77	180	9	1.95	2.67	39.7323	0.06	0.024	0.031	0.001
rooms	22	165.9	172.83	180	9	4.18	4.81	59.9355	0.06	0.021	0.032	0.091
$dense_4$	53	179.1	200.86	44	16	12.14	12.27	0.0008	0.53	0.286	0.235	0.018
$potholes_2$	68	154.5	163.13	42	16	5.55	5.58	0.0007	0.42	0.119	0.295	0.028
$m3_1$	71	39.0	41.74	47	16	6.88	7.05	0.0102	0.69	0.332	0.358	0.029
$warehouse_4$	79	369.2	390.18	139	16	5.69	5.76	1.4492	0.49	0.099	0.383	0.040
jh_2	80	201.9	205.61	42	16	1.82	1.87	0.0008	0.51	0.174	0.328	0.045
pb_4	104	654.6	659.18	45	18	0.70	0.73	0.0008	0.37	0.068	0.297	0.239
ta_2	141	328.0	338.94	43	17	3.33	3.35	0.0008	0.46	0.085	0.368	0.681
$h2_5$	168	943.0	964.24	119	19	2.26	2.31	0.9978	5.73	4.417	1.278	0.947
$potholes_1$	282	277.3	295.51	42	17	6.55	6.56	0.0008	1.80	0.633	1.144	0.233
jh_1	356	363.7	380.07	42	18	4.50	4.50	0.0008	2.39	0.837	1.526	0.867
$pb_{1.5}$	415	839.6	855.03	43	19	1.84	1.85	0.0008	2.07	0.585	1.458	1.978
$h2_2$	568	1316.2	1352.89	44	20	2.79	2.80	0.0008	12.35	8.004	4.280	4.396
ta_1	574	541.1	573.52	43	20	5.99	6.00	0.0008	3.68	1.434	2.201	0.935

the initialization. In several cases, T_{init} is similar to T_{adapt}. The initialization time is even greater than the adaptation time for the problem $h2_2$. The last column T_{LK} shows average values of the required computational time to find a solution by the `linkern` algorithm from the Concorde package [2], which uses the Chained Lin–Kernighan heuristic. The algorithm utilized a distance matrix that is found in time T_{init}; thus, the last two columns can be used to compare the computational burden of SOM and the combinatorial heuristic. In three cases, the SOM adaptation procedure is less computationally expensive than the heuristic approach. These results are particularly interesting because the used path approximation is a relatively complex algorithm in comparison with the usage of the distance matrix in the combinatorial heuristic.

Regarding T_{init} and T_{adapt} for the $h2_2$ problem, the most intensive part of the algorithm is the preparation of all the shortest paths. These paths are not involved in the adaptation process, and therefore, an additional speed improvement can be based on omitting the paths pre-computation, and consideration of approximate paths determined during the adaptation. The solution quality can decrease; so, the idea would need further investigation.

6.5. Discussion

Two state-of-the-art SOM algorithms have been examined for the non-Euclidean TSP in the polygonal domain $\mathcal{W}$. The algorithms have been improved in several ways by already published modifications of the adaptation rule, and also by new proposed improvements. One of the main issues of SOM application in $\mathcal{W}$ is determination of node–city path, which is computed many times. The presented modifications significantly reduce the number of node–city path queries, and reduce the required computational time up to one hundred times.

The improvements are mostly visible for problems with a high number of cities. However, SOM approaches for the Euclidean TSP are able to solve problems with thousands of cities. In the presented results, the largest examined problem has "only" about five hundreds cities. From the practical point of view, the largest examined problems are the inspection planning problems within real environments and quite small visibility range.[2] The considered problems represent a realistic upper bound of the problem size because for a higher number of cities the visibility range has to be unrealistically small, or the environments have to be significantly larger. The visibility range and the size of the environment relate with a structure of cities (sensing locations in the inspection planning) in an environment, which can affect the solution quality. Cities that are relatively close to each other make the local search for a shorter route more important, which is a quite difficult task for a conventional SOM; mainly because of the decreasing learning rate during the adaptation. In these cases, the proposed *hull* initialization improves the quality because the adaptation starts with a spread ring. The modified algorithms have been examined only in $\mathcal{W}$ and it is expected that the benefit of the presented modifications will not be significant in the Euclidean problems due to relatively inexpensive computation of node–city distances.

An additional performance improvements can be based on consideration of smaller or dynamic number of nodes. In literature, 2.5 times more nodes than cities has been reported as the most suitable. Also for the examined problems and algorithms, different numbers of nodes decrease the solution quality. The idea of the *approx. adapt* modification does not provide expected improvement. However, in early adaptation steps, nodes are often moved over map vertices that mean

[2] The visibility range in meters is denoted as the subscript of the problem name.

the neighboring nodes of the winner node are placed at the same shortest path from the particular map vertex to the city. In these cases, new nodes can be created and adaptation can start with only a very small number of nodes, which is an idea for future work.

One remark about the Co-adaptive net algorithm and the proposed improvements has to be mentioned. The algorithm is quite complex, which can be considered as a weak point of an eventual massive parallelization. This is also a weak point of the determination of the geodesic path, and the applied improvements to the `select winner` procedure, which can increase the complexity of a parallel implementation. Thus, it seems that one of the SOM features is lost in $\mathcal{W}$.

Another point of the Co-adaptive net algorithm is its relatively strict orientation to the routing optimization, which, in fact, is not an issue for the TSP. The modified SME algorithm provides much better performance in this aspect, i.e., the maximal distance of the winner nodes to cities. From this perspective, the modified Somhom's adaptation schema with the Zhang–Bai–Hu adaptation rules is more suitable for other routing problems in $\mathcal{W}$. Consideration of the ring evolution in $\mathcal{W}$ provides opportunity to find a solution of the watchman route problem [16] or other inspection problems where cities are not explicitly prescribed, which is one of the main SOM benefits over the classical combinatorial approaches [15].

7. Conclusion

Improved self-organizing map-based algorithms for the TSP in the polygonal domain have been proposed. The required computational time of the algorithms has been decreased by the proposed β-*condition* adaptation rule and the *informed* select winner procedure in combination with the approximate shortest path in $\mathcal{W}$. In addition, the performance of the SME algorithm has been improved using a combination of the Zhang–Bai–Hu adaptation rules with the new *hull* initialization technique. The successor of the Co-adaptive net algorithm utilizes the MSNF adaptive rule with the *center* initialization to improve the quality of solutions.

The algorithms have been examined in several instances of the inspection planning task in the polygonal domain $\mathcal{W}$. The proposed algorithms move the performance of the SOM algorithms in $\mathcal{W}$ to the next level, and allow their further application in other routing problems in the polygonal domain. The complexity of non-Euclidean distance determination is indicated in the SOM literature as a drawback. The encouraging results presented in this paper, and the significant performance improvements can be motivation for a further investigation of SOM applications in other variants of routing problems, not only in the polygonal domain but also in high-dimensional spaces with obstacles, where approximate paths between nodes and goals (cities) are necessary, e.g., route planning in 3D environments or in high-dimensional configuration spaces.

Acknowledgement

The work has been supported by the Ministry of Education of the Czech Republic under Project No. 7E08006 and by EU project No. 216240.

References

[1] B. Angéniol, G. de la, C. Vaubois, J-Y L. Texier, Self-organizing feature maps and the travelling salesman problem, Neural Netw. 1 (1988) 289–293.
[2] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, CONCORDE TSP Solver. (2003) (<http://www.tsp.gatech.edu/concorde.html>) (cited 8 July, 2010).
[3] N. Aras, B.J. Oommen, I.K. Altinel, The Kohonen network incorporating explicit statistics and its application to the travelling salesman problem, Neural Netw. 12 (9) (1999) 1273–1284.
[4] N. Aras, I.K. Altinel, J. Oommen, A Kohonen-like decomposition method for the Euclidean traveling salesman problem-KNIES_DECOMPOSE, IEEE Trans. Neural Netw. 14 (4) (2003) 869–890.
[5] Yanping Bai, Wendong Zhang, Zhen Jin, An new self-organizing maps strategy for solving the traveling salesman problem, Chaos Solitons Fract. 28 (4) (2006) 1082–1089.
[6] Marco Budinich, A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing, Neural Computing 8 (2) (1996) 416–424.
[7] Laura Burke, Conscientious neural nets for tour construction in the traveling salesman problem: the vigilant net, Comput. Oper. Res. 23 (2) (1996) 121–129.
[8] Laura I. Burke, Neural methods for the traveling salesman problem: insights from operations research, Neural Netw. 7 (4) (1994) 681–690.
[9] Laura I. Burke, Poulomi Damany, The guilty net for the traveling salesman problem, Comput. Oper. Res. 19 (3–4) (1992) 255–265.
[10] Vašek Chvátal, William Cook, George B. Dantzig, Delbert R. Fulkerson, Selmer M. Johnson, Solution of a Large-Scale Traveling-Salesman Problem, Springer, Berlin Heidelberg, 2010 (Chapter 1, pp. 7–28).
[11] E.M. Cochrane, J.E. Beasley, The co-adaptive neural network approach to the Euclidean travelling salesman problem, Neural Netw. 16 (10) (2003) 1499–1525.
[12] Jean-Charles Créput, Abderrafiaa Koukam, A memetic neural network for the Euclidean traveling salesman problem, Neurocomputing 72 (4–6) (2009) 1250–1264.
[13] Olivier Devillers, Sylvain Pion, Monique Teillaud, Projets Prisme, Walking in a triangulation, Int. J. Found. Comput. Sci. 13 (2001) 106–114.
[14] Masato Edahiro, Iwao Kokubo, Takao Asano, A new point-location algorithm and its practical efficiency: comparison with existing algorithms, ACM Trans. Graph 3 (2) (1984) 86–109.
[15] Jan Faigl, Multi-Goal Path Planning for Cooperative Sensing, Ph.D. Thesis, Czech Technical University in Prague, 2010a.
[16] Jan Faigl, Approximate solution of the multiple watchman routes problem with restricted visibility range, IEEE Trans. Neural Netw. 21 (10) (2010) 1668–1679.
[17] Jan Faigl, Miroslav Kulich, Libor Přeučil, A sensor placement algorithm for a mobile robot inspection planning, J. Intell. Robotic Syst. 62 (3) (2011) 329–353.
[18] Jan Faigl, Miroslav Kulich, Vojtěch Vonásek, Libor Přeučil, An application of self-organizing map in the non-euclidean traveling salesman problem, Neurochaptercomputing 74 (5) (2011) 671–679.

Appendix 1 - Faigl, J.: *On the Performance of Self-organizing Maps for the Non-Euclidean Traveling Salesman Problem in the Polygonal Domain* [27], referenced on page 9.

[19] J.C. Fort, Solving a combinatorial problem via self-organizing process: an application of the Kohonen algorithm to the traveling salesman problem, Biol. Cybern. 59 (1) (1988) 33–40.
[20] Bernd Fritzke, Peter Wilke, FLEXMAP – A Neural Network For The Traveling Salesman Problem With Linear Time And Space Complexity, in: Proceedings of IJCNN, Singapore, 1991, pp. 929–934.
[21] H.H. González-Baños, D. Hsu, J.-C. Latombe, Motion planning: recent developments, in: S.S. Ge, F.L. Lewis (Eds.), Autonomous Mobile Robots: Sensing Control, Decision-Making and Applications, CRC Press, 2006 (chapter 10).
[22] Hui-Dong Jin, Kwong-Sak Leung, Man-Leung Wong, Z.-B. Xu, An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem, IEEE Trans. Syst. Man Cybern. Part B: cybernetics 33 (6) (2003) 877–888.
[23] Marcelo Kallmann, Path Planning in Triangulations, Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games, vol. 31, Edinburgh, Scotland, 2005.
[24] Miroslav Kulich, Jan Kout, Libor Přeučil, Jiří Pavlíček, and Roman Mázl et al. PeLoTe - a Heterogeneous Telematic System for Cooperative Search and Rescue Missions, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 2004, volume 1, Sendai, 2004.
[25] Miroslav Kulich, Jan Faigl, Libor Přeučil, Cooperative planning for heterogeneous teams in rescue operations, in IEEE International Workshop on Safety, Security and Rescue Robotics, pp. 230–235, 2005.
[26] Kwong-Sak Leung, Hui-Dong Jin, Zong-Ben Xu, An expanding self-organizing neural network for the traveling salesman problem, Neurocomputing 62 (2004) 267–292.
[27] Thiago A.S. Masutti, Leandro N. de Castro, A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, Inf. Sci. 179 (10) (2009) 1454–1468.
[28] Kazushi Murakoshi, Yuichi Sato, Reducing topological defects in self-organizing maps using multiple scale neighborhood functions, Biosystems 90 (1) (2007) 101–104.
[29] M.H. Overmars, E. Welzl, New methods for computing visibility graphs, in: SCG '88: Proceedings of the fourth annual symposium on Computational geometry, ACM, New York, NY, USA, 1988, pp. 164–171.
[30] Alessio Plebe, Angelo Marcello Anile, A neural-network-based approach to the double traveling salesman problem, Neural Computing 14 (2) (2002) 437–471.
[31] Gerhard Reinelt, TSPLIB – a traveling salesman problem library, J. Comput. 3 (4) (1991) 376–384.
[32] Mitul Saha, Tim Roughgarden, Jean–Claude Latombe, and Gildardo Sánchez-Ante. Planning Tours of Robotic Arms among Partitioned Goals, Int. J. Rob. Res 25 (3) (2006) 207–223.
[33] Raimund Seidel, A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons, Comput. Geom. Theory Appl 1 (1) (1991) 51–64.
[34] K. Smith, M. Palaniswami, M. Krishnamoorthy, Neural techniques for combinatorial optimization with applications, IEEE Trans. on Neural Netw. 9 (6) (1998) 1301–1318.
[35] Kate A. Smith, Neural networks for combinatorial optimization: a review of more than a decade of research, INFORMS J. Comput. 11 (1) (1999) 15–34.
[36] Samerkae Somhom, Abdolhamid Modares, Takao Enkawa, A self-organising model for the travelling salesman problem, J. Oper. Res. Soc. (1997) 919–928.
[37] Frederico Carvalho Vieira, Adrião Duarte Dória Neto, José Alfredo Ferreira Costa, An efficient approach to the travelling salesman problem using self-organizing maps, Int. J. Neural Syst. 13 (2) (2003) 59–66.
[38] Haiqing Yang and Haihong Yang, An Self-organizing Neural Network with Convex-hull Expanding Property for TSP, in: Neural Networks and Brain, 2005. ICNN& B '05. International Conference on, vol. 1, pp. 379–383, October 2005.
[39] Wendong Zhang, Yanping Bai, Hong Ping Hu, The incorporation of an efficient initialization method and parameter adaptation using self-organizing maps to solve the TSP, Appl. Math. Comput. 172 (1) (2006) 603–623.

Appendix 2 - Faigl, J. - Přeučil, L.: *Inspection Planning in the Polygonal Domain by Self-Organizing Map* [28], referenced on page 10.

Applied Soft Computing 11 (2011) 5028–5041

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Inspection planning in the polygonal domain by Self-Organizing Map

Jan Faigl [a,*], Libor Přeučil [b]

[a] Center for Applied Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27, Prague 6, Czech Republic
[b] Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27, Prague 6, Czech Republic

ARTICLE INFO

Article history:
Received 13 August 2010
Received in revised form 24 February 2011
Accepted 23 May 2011
Available online 12 June 2011

Keywords:
Inspection planning
Multi-goal path planning
Self-Organizing Map (SOM)
Traveling Salesman Problem (TSP)
Watchman Route Problem (WRP)
Polygonal domain

ABSTRACT

Inspection planning is a problem of finding a (closed) shortest path from which a robot "sees" the whole workspace. The problem is closely related to the Traveling Salesman Problem (TSP) if the discrete sensing is performed only at the finite number of sensing locations. For the continuous sensing, the problem can be formulated as the Watchman Route Problem (WRP), which is known to be NP-hard for the polygonal representation of the robot workspace. Although several Self-Organizing Map (SOM) approaches have been proposed for the TSP, they are strictly focused to the Euclidean TSP, which is not the case of the inspection path planning in the polygonal domain. In this paper, a novel SOM adaptation schema is proposed to address both variants of the inspection planning with discrete and continuous sensing in the polygonal domain. The schema is compared with the state of the art SOM schema for the TSP in a set of multi-goal path planning problems and WRPs. The proposed algorithms are less computationally intensive (in order of tens) and provide better or competitive solutions.

1. Introduction

Inspection planning deals with finding a shortest inspection path such that all points of the workspace W are "seen" from the path. The problem is studied in mobile robotics in which a path for a mobile robot performing the inspection is planned in a priori known map of the environment. The map can be represented by the polygonal domain, which makes the problem close to computational geometry. A practical consideration of mobile robot sensing capabilities leads to two types of sensing: discrete and continuous. These sensing models are motivated by the cost of sensing and the cost of motions. The continuous sensing is suitable for problems where the cost of the sensing is relatively cheap in comparison to the cost of the motion. Contrary to the discrete sensing model, where the cost of the sensing is dominant, and the cost of the motion can be ignored. The combination of both costs is a difficult problem and it remains largely unexplored [1].

The inspection planning problem formulations can be found in computational geometry. The problem with continuous sensing can be formulated as the *Watchman Route Problem* (WRP) [2]. The WRP is a problem of finding a closed shortest path in the polygonal domain W such that all points of W are visible from at least one point at the path. Even though optimal algorithms for restricted class of polygons have been proposed, the WRP is NP-hard for W, and probably the first heuristic approach has been introduced in [3].

A decoupled approach can be used to address the inspection planning with discrete sensing. The problem is decomposed into the set cover problem and the consecutive multi-goal path planning problem. For the polygonal domain W, the set cover problem can be formulated as the *Art Gallery Problem* (AGP). The AGP stands to find a minimal number of guards to cover W. The guards represent sensing locations, and each guard covers a part of the environment by its star-shaped visibility polygon. The AGP is NP-hard even for a polygon without holes [4]. The multi-goal path planning problem can be formulated as the well-known *Traveling Salesman Problem* (TSP) if all paths between sensing locations are known [5,6].

The AGP and WRP are studied in the computational geometry domain for an unrestricted model of visibility. However, sensing (visibility) of real sensors (cameras or range finders) is limited, e.g., in sensing range and frequency. To distinguish the restricted visibility, authors of [1] call the problem of finding sensing locations *sensor placement* rather than the AGP. Similarly, the WRP with restricted visibility range to a distance d is called *d-Watchman Route Problem* (d-WRP) [7]. These variants of the problems with the restricted visibility range also belongs to the NP class; thus, approximate solutions are more suitable for real application to get "good" solutions with "reasonable" computational requirements.

The decoupled approach provides a feasible solution of the inspection planning, and has been used in robotic tasks [8,9]. Sensing locations for restricted visibility range can be found by different techniques [10]. The TSP can be solved by various approaches from the operational research [11], or by soft computing techniques like ant colony system [12], Self-Organizing Map (SOM) approaches [13], or immune system [14]. On the other hand, the WRP in W has been addressed (to the best of our knowledge) only by the

* Fax: +420 224357224.
E-mail address: xfaigl@labe.felk.cvut.cz (J. Faigl).

1568-4946/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2011.05.055

Appendix 2 - Faigl, J. - Přeučil, L.: *Inspection Planning in the Polygonal Domain by Self-Organizing Map* [28], referenced on page 10.

Nomenclature

W	the polygonal domain representing the robot workspace, $W \subset \mathbb{R}^2$
v	the number of vertices of W
h	the number of holes of W
p	the number of convex polygons of the convex partition of W
n	the number of goals
d	the visibility range
N_V	the number of triangular mesh vertices
N_T	the number of triangles
N_C	the number of convex polygons of the cover set of W
a_i	the visible area from a triangle
g	a goal, $g \in W$
ν	a node (neuron weights), $\nu \in W$
$S(\nu, g)$	an approximate path from ν to g
m	the number of nodes (neurons)
f	the neighbouring factor
δ	the size of the winner node neighbourhood
σ	the learning gain (neighbouring function variance)
$f(\sigma, l)$	the neighbouring function
α	the gain-decreasing rate
μ	learning rate
s	the node moving activity threshold
ε	the minimal allowable error

heuristic approach presented in [3]. The algorithm is based on a set of static guards that are used to determine the minimum spanning tree from the pairwise shortest paths between guards. The tree is split to construct a route that is shortened by vertex substitutions and removing of redundant vertices. Even though the approach is based on guards, solutions have been presented only for an unrestricted visibility range. In [15], a SOM based approach for the d-WRP has been presented; thus, SOM provides solutions for both inspection planning variants. However, the main difficulty of SOM application in the polygonal domain is determination of the shortest paths among obstacles, which is more computationally demanding than a pure computation of the Euclidean distances between neurons' weights and an input vector.

In this paper, SOM is applied to the inspection planning problem with discrete and continuous sensing in the polygonal domain. A new adaptation schema is proposed and compared with an already available schema for the TSP [16] in a set of problems created from a map of real environments and several visibility ranges. The main contribution of this paper is new adaptation schema for the multi-goal path planning problem in the polygonal domain that can be used to addressed the non-Euclidean TSP and d-WRP, i.e., inspection planning with continuous sensing.

The rest of this paper is organized as follows. The next section provides overview of the addressed problem. The related work is presented in Section 3. The proposed adaptation schema for the multi-goal path planning problem is presented in Section 4. Then, the schema is applied to the inspection planning with continuous sensing in Section 5. Experimental results of the proposed algorithms are presented in Section 6. Concluding discussion and remarks of the future work are presented in Section 7. The list of the used symbols is presented in Nomenclature.

2. Problem statement

An environment to be inspected by a mobile robot is a priori known, and a polygonal map of the environment is available. The robot is equipped with an omnidirectional sensor with a sensing range restricted to a distance d. The notion of d-visibility is assumed as follows. Two points p and q in a polygon $\boldsymbol{P}$ are called *d-visible*, if the line segment joining them is contained in $\boldsymbol{P}$, and if the segment length is less or equal to d. The sensor coverage is modeled by a disk with the radius d. A point robot is assumed, and a path between two points in the polygonal domain W consists of sequence of straight line segments joining the points and vertices of W, and all segments are entirely inside W. The addressed variants of the inspection planning are following.

Discrete sensing: The whole environment is covered by performing a finite number of measurements with the range d at sensing locations. A set of such sensing locations $\boldsymbol{G}$ is given, and all locations are reachable by the mobile robot. The problem is to find a closed path (possibly a shortest one) connecting all sensing locations. The problem is the multi-goal path planning problem that is considered as the non-Euclidean TSP in the polygonal domain.

Continuous sensing: Measurements can be taken along a path in the continuous sensing problem variant, therefore, sensing locations are not explicitly prescribed. The problem is to find a closed (possibly a shortest one) path such that each point of the environment is d-visible from some point of the path. The problem is formulated as the d-WRP in the polygonal domain.

Even though the cost of the sensing and the cost of the motion can be considered in the inspection planning, only the length of the inspection path is used as the quality metric in this paper. Mainly because a SOM approach for the multi-goal path planning provides an approximate solution of the related TSP, and a shorter path is a plus. Besides, the decoupled approach can also be used for the d-WRP. Having a prescribed set of sensing locations, eventually the smallest set, only the length of the path can be minimized. Therefore, the length of the inspection path as the only metric makes sense for both inspection variants.

3. Related work

3.1. Reference algorithm

To compare the solution quality of the examined algorithms the following decoupled approach is used to find a reference solution. A deterministic sensor placement algorithm [17] is used to find a set of sensing locations. The algorithm is based on a decomposition of W into a set of convex polygons. First, Seidel's algorithm [18] is used to find the primal convex partition. Convex polygons of the partition are eventually divided into convex sub-polygons if a convex polygon cannot be covered from one point with the d-visibility. The required computational time is proportional to the number of found sensing locations [17].

The inspection path is found as the solution of the TSP on a graph $G(V, E)$, where V stands for sensing locations, and E is the set of edges with costs computed as the length of the shortest path between the sensing locations. The paths are found by Dijkstra's algorithm in $O(nn_e \log(n+v))$ on the visibility graph, which is found in $O((n+v)^2)$ [19], where v denotes the number of polygon vertices, n is the number of sensing locations, and n_e is the number of edges of the visibility graph. Without loss of generality $G(V, E)$ is assumed to be complete. An optimal solution of the TSP is found by the concorde solver [20].

3.2. SOM procedures for the TSP

The basic idea of SOM for the TSP is based on Kohonen's two-layered unsupervised neural network in which the first layer represents coordinates of the presented goals to the network. The second layer consists of neurons organized in a cycle (ring), and

Appendix 2 - Faigl, J. - Přeučil, L.: *Inspection Planning in the Polygonal Domain by Self-Organizing Map* [28], referenced on page 10.

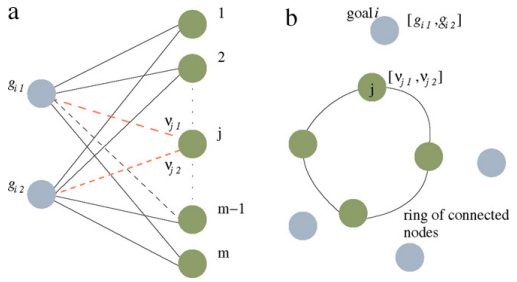


Fig. 1. Schema of the two-layered neural network and associated geometric representation.

each neuron is connected with the first layer. The weights of the connections represent coordinates of the node, see Fig. 1. The adaptation schema is an iterative two phases procedure. At each iteration, goals are presented to the network in a random order, and a winner node is found for each goal in the competitive phase. The winner selection uses the Euclidean distance of a node to the goal. The winner node and its neighbouring nodes are adapted towards the goal in the cooperative phase. The adaptation process is typically terminated if winner nodes are sufficiently close to the goals.

The TSP has been addressed by several SOM approaches during the last two decades. The pioneering work of Angéniol et al. [21] and Fort [22] in 1988 has been followed by particular improvements in the quality of found solutions and required computational time. An inhibition mechanism, which prevents nodes to win too often, has been used in [16]. In [23], authors consider creation/deletion of nodes, and selection of winners based on the shortest path to the segment joining two neighbouring nodes. Several nodes initializations have been examined in [24,25].

Probably the most complex algorithm is the Co-adaptive net that is extensively evaluated in [13]. The authors of the Co-adaptive net reported that their approach together with [16] provide superior results in various instances of the TSP from the TSPLIB [26]. However, the high number of the Co-adaptive net parameters can be considered as a drawback.

The algorithm proposed by Somhom et al. [16], denoted as SME in this paper, is particularly interesting. It provides competitive quality of solutions to the Co-adaptive net algorithm, but it is less complex, and it depends on a less number of parameters. In this paper, the SME algorithm is considered as the reference adaptation schema for the initial application of SOM to the inspection planning in W. The algorithm works as follows. Nodes are initialized as a small ring around the center of the goals. The winner node is selected according to $\nu^* = \text{argmin}_\nu |g, \nu|$, where $|.,.|$ denotes the Euclidean distance between the goal g and the node ν for the Euclidean TSP. Each node can be a winner only once in each adaptation step; thus, a winner is inhibited after its selection. The inhibition is cleaned at the begin of the next iteration, i.e., new presentation of goals to the network. The adaptation rule moves the winner node and its neighbouring nodes towards the presenting goal g according to $\nu'_j = \nu_j + \mu f(\sigma, l)(g - \nu_j)$, where μ is the fractional learning rate. The neighbouring function is $f(\sigma, l) = \exp(-l^2/\sigma^2)$ for $l < \delta$, and $f(\sigma, l) = 0$ otherwise, where σ is the gain parameter, l is the distance in the number of nodes measured along the ring, δ is the size of the winner node neighbourhood that is set to $\delta = 0.2m$, where m is the number of nodes set to $m = 2n$ for n goals. The initial value of σ is set proportionally to the problem size $\sigma_0 = 0.06 + 12.41n$, and it is decreased at the end of each adaptation step according to $\sigma = \sigma(1 - \alpha)$, where α is the gain-decreasing rate. The values of learning and decreasing rates are $\mu = 0.6$ and $\alpha = 0.1$ [27], respectively. The adaptation is terminated if all winners are in a distance less than $\varepsilon = 0.001$.

Regarding the number of parameters authors of [28] proposed alternative adaptation rules to Kohonen's exponential evolution. To avoid initial values of the learning and gain-decreasing rates, the authors proposed simplified adaptation rules based only on the number of performed adaptation steps k. The learning rate is defined as $\mu = 1/\sqrt[4]{k}$, and the learning gain as $\sigma = \sigma(1 - 0.01k)$. The initial value of the gain is $\sigma_0 = 10$. For small values of σ, the value of the neighbouring function is very small; thus, the neighbouring nodes are negligibly moved. To decrease the computational burden, the authors recommended to gradually decrease the neighbourhood of the winner node after each adaptation step. The recommended initial value of the neighbourhood is $\delta = 0.4m$ that is decreased according to $\delta = 0.98\delta$ at the end of each adaptation step.

3.3. Approximation of the shortest path in W

The main difficulty of SOM application to problems in the polygonal domain W is a determination of the shortest path among obstacles, which can be computationally intensive. In [29], a simple, yet sufficient approximation has been applied to the self-organizing adaptation procedure. It is based on a convex partition of W. The partition $\boldsymbol{P}$ is a set of disjoint convex cells $\boldsymbol{P} = \{C_1, C_2, \ldots, C_k\}$ such that the union of the cells is W. The cells are induced by the diagonals of W, and each cell is formed from a sequence of W vertices. During the adaptation, nodes are inside W, and therefore, they are always inside some cell. A collision free path for two points p_1 and p_2 that are inside cells $p_1 \in C_1$ and $p_2 \in C_2$ can be found as a path over the cells' vertices $v_1 \in C_1$ and $v_2 \in C_2$. The vertices are selected to minimize the length of the path $|p_1, v_1| + |S(v_1, v_2)| + |v_2, p_2|$, where $|.,.|$ denotes the Euclidean distance of two points, and $|S(.,.)|$ is the length of the shortest path between two vertices. Such a path can be further refined by consideration of direct visibility from the particular point to a vertex of the path. The used direct visibility test is similar to the method [30], a convex partition is used rather than a triangulation. An additional improvement of the approximate path can be achieved if vertices of obstacle edges that intersect the direct line segment from p_1 to p_2 are considered in the construction of the primal path. An example of the primal path and its refined variants is shown in Fig. 2.

The used supporting structures are a convex partition of W, and all shortest path between ν vertices of W. A convex partition can be found in $O(\nu \log \nu)$ [18]. The shortest path can be pre-computed by Dijkstra's algorithm using the visibility graph. The problem of finding the cell C_ν is the point-location problem, which can be solved in $O(\log \nu)$. Besides, the cell can be determined during the node movement towards the goal by the walking technique similar to [31]. The complexity of such cell determination is bounded by $O(\log n_d)$, where n_d is the number of passed diagonals of the used convex polygon partition.

In the TSP, goals are fixed, and therefore, all shortest path from the map vertices to the goals can be pre-computed, which reduces the required computational time for the adaptation at the cost of higher memory requirements. For large sets of goals, the approach visualized in Fig. 2 can be more appropriate, as it provides approximate path for two arbitrarily placed points in W, and its space requirements depends only on the number of W's vertices.

3.4. SOM procedure for the TSP in W

An application of the SME adaptation schema to the TSP in W has been presented in [32]. The main difference to the algorithm for the Euclidean TSP is in consideration of the approximate path

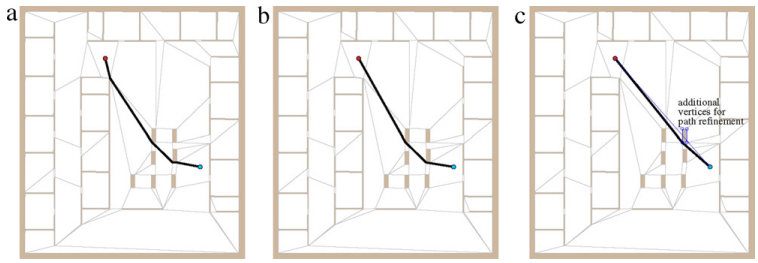


Fig. 2. Approximate path between two points; (a) rough approximate path over cells' vertices, (b) refined path, and (c) refined path with consideration of detected obstacle's vertices.

found by the above described procedure, in particular using the pre-computed shortest paths from the vertices to the goals. The path is used in the select winner part to determine distance of the node to the presented goal, and during the adaptation when nodes are moved towards the goal. Besides, the following modifications have been applied.

The termination condition also considers a maximal number of adaptation steps. An error of the path approximation can cause that the winner–goal distance is not effectively decreased during the adaptation, which can lead to a distance higher than the required ε. Even though such a convergence issue has been observed only for the rough path approximation, the maximal number of the iterations is advantageous as it guarantees termination of the algorithm.

A practical implementation of the select winner procedure can utilize the Euclidean distance to inform the winner searching process, and to decrease the computational burden. During the searching, all non inhibited nodes are examined, and the closest node is selected as the winner. Let v_g be an actual winner candidate to the presented goal g. A distance of a node v to g as a length of the path among obstacles is determined only if the Euclidean distance $|v, g|$ is shorter than the distance of $|v_g, g|$. This technical improvement does not affect the quality of solution. However, it has been observed that it provides solution up to two times faster for the SME adaptation schema.

In the original SME algorithm [16], the initial values of nodes are placed around a center of goals, which cannot be used in the W because such a center can be in an obstacle. Based on experimental results with various initialization it has been observed that the SME schema is insensitive to the initial point around which the ring is created. Thus, to ensure that nodes are placed in W they are placed around the first goal as a small ring with the radius 0.5 cm. The sufficient free space around the first goal is assumed. In the case of inspection planning the space around the goal is ensured by the sensor placement algorithm.

3.5. SOM procedure for the WRP in W

The SOM procedure for the WRP has been presented in [15]. The main idea of the procedure is that the ring of nodes represents the watchman route itself, and the nodes are adapted towards uncovered parts of W. Determination of the ring coverage is based on approximation of the continuous sensing along a straight line segment using a convex cover set. The cover set consists of a set (possibly overlapping) convex polygons, which dimensions are restricted to respect the limited visibility range d. A triangular mesh of W is used to support fast determination of incident convex polygons with a segment. A convex cover set is found on top of the mesh, i.e., a convex polygon of the set consists of mesh vertices, see Fig. 3a.

It is not required to have a minimal number of the convex polygons, because the cover set is used as follows. Each triangle is associated with at least one convex polygon, and each convex polygon has associated a set of triangles that are entirely inside the polygon. For a straight line segment lying in W, all incident triangles are found by the walking in triangulation technique [31]. From these triangles, all associated incident convex polygons are found, and the coverage along the segment is determined as a union of all triangles associated to the incident convex polygons, see Fig. 3b. The coverage of the ring is determined from the sequence of straight line segments joining each neighbouring nodes found by the approximation of the shortest path between two points in W.

The adaptation procedure follows the SME schema for the TSP in W. Centroids of the mesh triangles are used as goals presented to the network. However, a winner node is found only for triangles that are not covered. A coverage from the ring is determined at the beginning of each adaptation step. During the adaptation, the coverage is updated by adding all triangles associated to the convex polygons that are incident with the presented triangle (centroid) after the winner node adaptation towards the centroid. The additional modification of the adaptation rule relates to the visibility nature of the WRP. To see the presented triangle from the ring, it is sufficient if the winner reaches some of the incident convex polygons associated to the triangle. Thus, an alternate goal is found from the intersection of the path from the node to the triangle centroid with the incident convex polygons, see Fig. 3c.

The adaptation is terminated if the ring covers all triangles or after 180 adaptation steps, which can lead to an incomplete coverage.

4. Proposed adaptation schema for the multi-goal path planning

In this section, a new adaptation schema for the multi-goal path planning problem is proposed. The schema follows the SME adaptation schema, particularly the algorithm described in Section 3.4, but the main difference is in the winner selection rule. The selection utilizes a creation/deletion mechanism, which is similar to the one used in [21], nevertheless it is also inspired by the approach [23]. Beside the selection rule, particular parts of the algorithm have been improved considering modifications of the aforementioned approaches proposed by several authors, and experimental evaluation of adaptation parameters settings. The proposed selection rule together with the improvements lead to a new adaptation schema for the inspection planning with discrete sensing that provides better solutions, and has lower computational requirements than the former schema. To provide an overview of the proposed algorithm the adaptation schema is depicted in Algorithm 1. The selection rule and the particular improvements are described in the following sections.

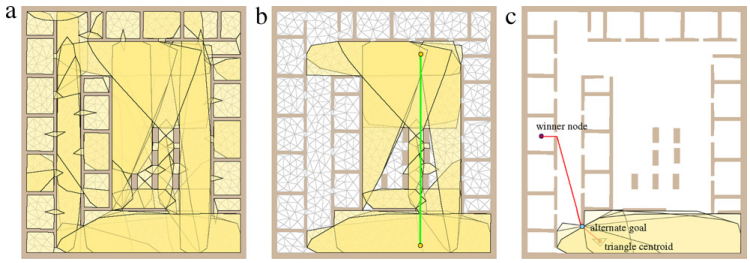


Fig. 3. Supporting structures for the WRP; (a) a convex cover set and an underlying triangular mesh, (b) an incident convex polygons with a straight line segment, and (c) an alternate goal.

Algorithm 1. SOM adaptation schema for the TSP.

Input: $\boldsymbol{G} = \{g_1, \ldots, g_n\}$ - a set of goals
Input: $(m, \sigma, \mu, \alpha, \delta, s)$ - the adaptation parameters
Input: ϵ – the maximal allowable error
Input: σ_{min} – the minimal allowable σ

```
init(ν_1, ..., ν_M)   // initial set of neurons weights
i ← 0                 // set the adaptation step
repeat
    error ← 0
    I ← ∅             // set of inhibited nodes
    Π(G) ← a random permutation of goals
    foreach g ∈ Π(G) do
        ν* ← select winner(node to g), ν* ∉ I)
        error ← max{error, |ν*, g|}
        adapt(ν*, g)  // call the adapt procedure
        I ← I ∪ {ν*}  // inhibit winner node
    k ← k + 1         // increment the adaptation step
    update_adaptation_parameters(σ, δ)
until error < ε or σ < σ_min
```

4.1. Initialization

Several initialization of the neurons weights have been proposed by various authors, e.g., a small ring around centroid of the goals [16], a tour found by the nearest neighbourhood [25], or convex hull of the goals [33]. In the polygonal domain, these initialization methods cannot be directly used, as the initial weights must be inside W. Based on experimental results, superior solutions have been achieved by the following modifications of the convex hull initialization. First, a convex hull of the goals is found without consideration of obstacles. After that, goals forming the hull, i.e., goals that are at the hull border, are connected by the shortest paths found using the visibility graph. So, a tour over the forming goals is constructed. Finally, nodes are equidistantly placed at the tour, starting at a random point of the tour. Examples of connected initial rings of nodes are shown in Fig. 4.

4.2. Winner selection

An idea behind the proposed winner selection method is based on consideration of a path between two nodes. The principle is shown in Fig. 5a for a problem without obstacles. The closest segment connecting two nodes is found instead of the closest winner. Then, the closest point at the segment is determined. If the point is different from the segment endpoints a new node is created with the point coordinates and added to the ring. Otherwise the closest node is a candidate to be the winner. If the winner candidate is inhibited a new node is created with the identical values of the winner weights. The newly created node becomes the winner of the current selection. The winner node is then adapted towards the presented goal.

The above described procedure can be effectively used for problems without obstacles, but determination of the closest segment to a point in W is more complex. That is why the following approximation is used. A regular winner node candidate is found using the approximation of the shortest path from a node to the goal. Then, two paths connecting the winner with its neighbouring nodes are determined as approximate paths between two points in W. For each of the paths, the closest segment point to the goal is determined using the Euclidean distance. These two points become candidates to be a winner of the current selection, see Fig. 5b. Due to obstacles, the candidate points can be farther than the winner candidate. Therefore a path from each candidate point to the goal is determined. If the length of the path for one of the points is shorter than the winner candidate distance to the goal, the corresponding point is used to create a new node. If it is not the case a new node is created if the winner node candidate is inhibited. The newly created node is the winner, otherwise the winner node candidate becomes the winner.

New nodes are created during the selection of winners, which can increase the computational burden. To remove unnecessary nodes a deletion mechanism is based on moving activity of nodes. The nodes that are not moved (adapted) in the last s adaptation steps are removed from the ring.

4.3. Adaptation

A winner node and its neighbouring nodes are moved towards the goal in the adapt procedure. A node ν is moved along the approximation of the shortest path $S(\nu, g)$ to the goal g by a distance $\beta|S(\nu, g)|$, where $\beta = \mu f(\sigma, l)$. The value of $f(\sigma, l)$ decreases with increasing distance of the node from the winner (in the number of nodes) and the number of adaptation steps, as σ is decreased. In final adaptation steps, the value of β is very small, and the movement can be negligible. Considering this observation the neighbouring nodes of the winner node are moved towards the goal only if $\beta > 10^{-5}$. This adaptation rule is denoted as β-*condition* in this paper.

Performed experiments show that this modification does not decrease the solution quality and increases speed of the algorithm two times for problems with about 500 goals. Even though a distance from a node to the goal is determined in the select winner procedure, the movement of the node is more computationally demanding. A path as a sequence of W vertices is not needed in the distance query, but it is required for the node movement, where a new node position at the path is determined. Therefore, the path is found in the adapt procedure before the node adaptation.

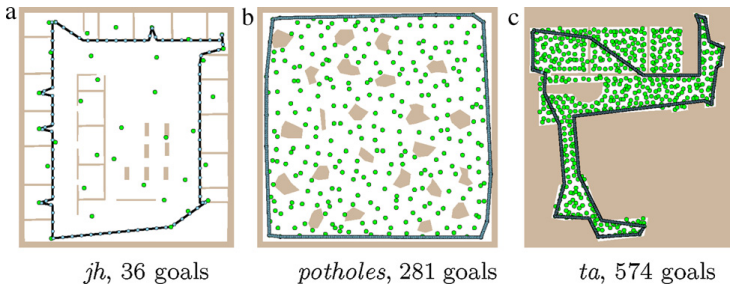


Fig. 4. Examples of initial rings of nodes in environments *jh*, *potholes* and *ta*; green disks represent goals and blue disks are nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

4.4. Adaptation parameters

Zhang et al. [28] proposed adaptation rules that are derived from the number of performed adaptation steps, which dramatically decrease the required number of adaptation steps to find a stable solution. However, for problems in W and with combination of the select winner procedure described in Section 4.2 better solutions are achieved with a slower decreasing σ. Also the solution quality is increased for a fixed value of the learning rate μ. The algorithm performance is also affected by the size of the winner node neighbourhood denoted as δ in this paper. Due to the used node creation/deletion mechanism, the number of nodes varies in each adaptation step. Rules that derive δ from the number of nodes can lead to a large winner's neighbourhood, and the proposed decreasing δ in [28] is not effective. So, the maximal value of δ is restricted to $\delta_m = 2n/8$, which corresponds to $m = 2n$ initial nodes and the neighbouring factor $f = 8$.

A summary of the used adaptation parameters is as follows. The initial values of the parameters are: $\sigma = 10$, $\mu = 0.6$, $m = 2n$, $f = 8$, $\delta = m/f$, $\alpha = 0.1$, $s = 8$. Values of δ and σ are changed (in the procedure `update_adaptation_parameters`) after each complete presentation of goals as follows:

- $\sigma \leftarrow \sigma(1 - 0.001k)$ – decrease the learning gain,
- $d \leftarrow 0.99^k \min\left\{\frac{m}{f}, \frac{2n}{f}\right\}$,

where m is the actual number of the nodes and k the actual number of the performed adaptation steps.

4.5. Termination condition

The initial value of the learning gain σ is independent to the problem size. Therefore, instead of a maximal number of adaptation steps the adaptation procedure can be terminated if σ is below given threshold $\sigma_{\min}$. The selected value is $\sigma_{\min} = 10^{-4}$, for which the value of the neighbouring function is small, and the neighbouring nodes are practically not moved. This termination condition is more intuitive and problem size independent contrary to the used maximal number of steps for the SME schema. Even though the adaptation is terminated before *error* is below the selected ε, the inhibition mechanism guarantees that all goals have associated distinct nodes. So, the final inspection tour over the goals is retrieved by traversing the ring.

4.6. Discussion

A collection of the presented adaptation rules provides new adaptation schema in which the number of nodes is not explicitly restricted, which is one of the benefit over the SME adaptation schema. Here, it should be mentioned that the particular rule (modification) can be used in other SOM approaches, e.g., the proposed *β-condition*. The rules can decrease the computational burden; however, they do not necessary improve the solution quality. During the evaluation of the rules, it has been observed that the proposed *hull* initialization does not improve solutions if the SME adaptation parameters are used. Moreover, the SME schema seems to be insensitive to the initial values of neurons' weights. The evaluation has been performed for a set of 21 inspection problems that represent instances of the non-Euclidean TSP. After this evaluation, the parameters presented in Section 4.4 have been selected.

5. Adaptation schema for the inspection planning with continuous sensing

The adaptation schema proposed in Section 4 has been applied to the algorithm for the d-WRP [15] briefly described in Section 3.5. The proposed schema has to be modified, because the solution of the d-WRP is represented by the ring of nodes itself, i.e.

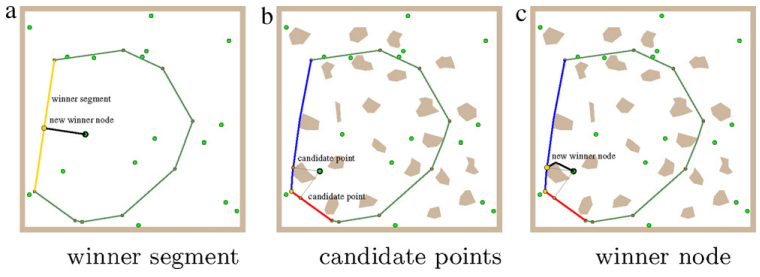


Fig. 5. A principle of the proposed winner selection rule.

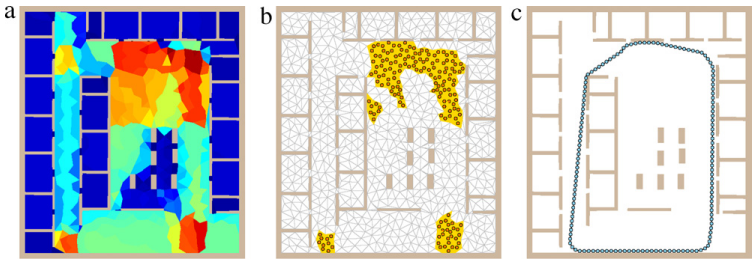


Fig. 6. An example of nodes initialization; (a) visualization of the triangle coverage, the highest coverage is in red (light), while the triangles that are incident with the smallest number of convex polygons are in blue (dark); (b) selected triangles (centroids) with highest coverage; (c) initial ring created from the convex hull of the selected triangles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

a sequence of straight line segments connecting the nodes. This aspect is considered in the following schema adjustments.

5.1. Initialization

Triangles, or more concretely their centroids, of the supporting triangular mesh are used as goals presented to the network. The nodes can be very close to the border polygon of W if a convex hull of the all centroids is used for the initial construction of the ring. It is because small triangles are typically located at corners. In addition, once a part of W is covered by some nodes, a winner is not selected to the particular goal, and the nodes lying in the part are moved only as neighbourhoods of another winner node. Such initially placed nodes can lead to an unnecessary long inspection path. To avoid such initialization only selected triangles are considered for the convex hull construction. The selection is based on an idea that if a ring starts from parts that are visible from large portion of W, then nodes will be attracted to other locations, and the parts will be covered by the segments connecting two neighbouring nodes.

First, for each triangle an area visible from the triangle is determined from the associated convex polygons of the cover set. The area is the sum of the areas of all triangles associated to the polygons. A visualization of the triangles visible areas is shown in Fig. 6a. Centroids of triangles with the largest visible area are selected for the convex hull construction, see Fig. 6b. Let the visible area of the ith triangle be a_i, and $a_{\max}$ be the largest visible area. All triangles with $a_i \geq a_{\max} - a_t$ are selected. After the selection, a convex hull of the triangles' centroids is created, and the centroids at the hull border are connected by the approximate shortest paths. Then, nodes are placed at the paths like in Section 4.1, see Fig. 6c.

The threshold value a_t can be set individually for a particular problem, as the triangular mesh provides only approximation of the coverage. However, the median of the visible areas provided the best results in the experimental evaluation. The initial number of nodes m is set to $m = 0.1n$, where n is the number of centroids (triangles), because n is typically higher than the number of sensing locations in the discrete inspection planning.

5.2. Three phases adaptation

The WRP algorithm tries to cover all triangles, and the adaptation is terminated if all triangles are covered. The adaptation procedure avoids selection of winner nodes for triangles that are already covered by nodes, or are covered from a path connecting

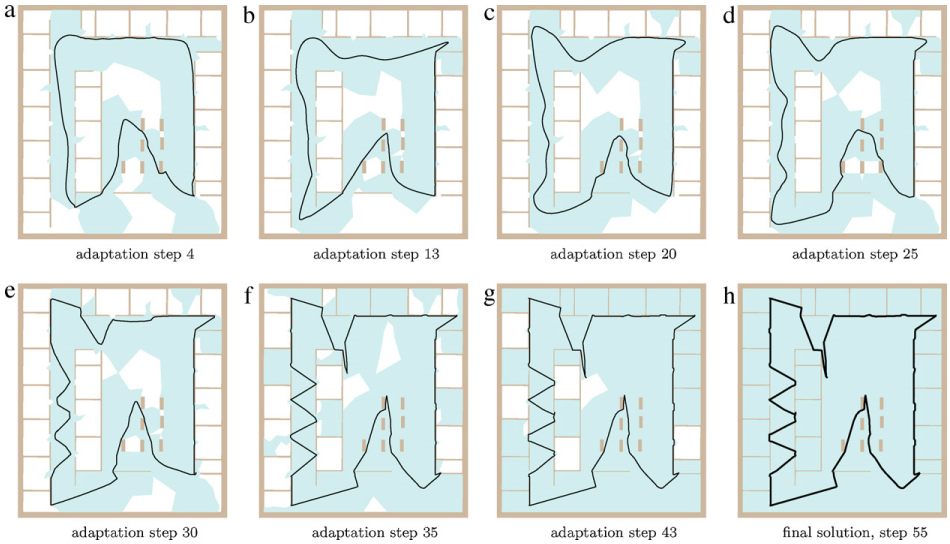


Fig. 7. An example of ring evolution and its coverage in the WRP, environment *jh* and visibility range 5 m.

Appendix 2 - Faigl, J. - Přeučil, L.: *Inspection Planning in the Polygonal Domain by Self-Organizing Map* [28], referenced on page 10.

Table 1
Map properties.

Name	Dimensions (m × m)	Area (m²)	v	h	p
jari	4.5 × 4.9	20	48	1	14
complex2	20.0 × 20.0	322	40	3	21
m1	4.8 × 4.8	20	51	4	26
m2	4.8 × 4.8	15	51	6	20
map	4.8 × 4.8	14	68	8	36
potholes	20.0 × 20.0	367	153	23	75
rooms	20.0 × 20.0	351	80	0	33
a	8.9 × 14.1	71	99	6	22
dense	21.0 × 21.5	299	288	32	150
m3	4.8 × 4.8	17	308	50	120
warehouse	40.0 × 40.0	1192	142	24	83
jh	20.6 × 23.2	455	196	9	77
pb	133.3 × 104.8	1453	89	3	41
ta	39.6 × 46.8	731	74	2	30
h2	84.9 × 49.7	2816	2062	34	476

the nodes. This is an important distinction in comparison to the multi-goal path planning. Also nodes deletion can decrease the ring coverage, and in a consequence it can lead to a convergence issue. In final adaptation steps, σ becomes low, and if some nodes are deleted the network does not effectively adapt to cover all triangles, because the adaptation is terminated due to $\sigma_{\min}$. Therefore the deletion rule cannot be simply used during the whole adaptation.

The following three phases adaptation is proposed to increase coverage of the final solution:

1. The creation/deletion rule described in Section 4.2 is used until 85% of triangles are covered.
2. The creation rule is used without the deletion until 95% coverage is reached.
3. The only winner node is selected without node creation in the final adaptation steps.

In the first phase, the ring is spread around the environment to cover most of the space while only the active nodes are preserved. The second phase is typically active only for several adaptation steps in which the number of nodes is increased. To decrease the computational burden, the number of neighbouring nodes δ is not increased in the second phase. The value of δ is computed as $\delta_I = m_I/f$, where m_I is the number of nodes in the last step of the first phase after the deletion. Nevertheless, δ is regularly decreased after each adaptation step by the rule $\delta = 0.99^k\delta_I$. The newly created nodes in the second phase support local searching that is finalized in the third stage.

An example of the ring evolution and the ring coverage is depicted in Fig. 7. Notice that the space is almost covered in the step 43; however, additional 12 steps are needed to achieve the full coverage. The advantage of the proposed adaptation rules is that these steps are performed very quickly, because the network adapts only to the uncovered triangles.

6. Experiments

The proposed SOM adaptation schema has been evaluated in a set of inspection planning problems for two types of sensing. The discrete sensing is considered as the multi-goal path planning problem formulated as the TSP, and the d-WRP formulation is used for the continuous sensing. The proposed algorithms are compared with the SME adaptation schema, in particular the schema is used in the TSP algorithm [32] and the d-WRP algorithm [15]. However, the Euclidean pre-selection (Section 3.4) and the β-*condition* (Section 4.3) are considered in the algorithms to decrease the computational

Table 2
Experimental results for the multi-goal path planning.

Problem	n	L_{opt} (m)	T_{init} (s)	SME					Proposed				
				m	PDM	PDB	s_L%	T_a (s)	m	PDM	PDB	s_L%	T_a (s)
jari	6	13.6	0.001	12	0.00	0.00	0.00	0.007	6	0.81	0.00	1.38	0.002
complex2	8	58.5	0.003	16	0.00	0.00	0.00	0.014	19	0.00	0.00	0.00	0.005
m1	13	17.1	0.006	26	0.03	0.00	0.09	0.029	17	0.07	0.00	0.14	0.016
m2	14	19.4	0.005	28	7.29	0.00	3.25	0.035	19	9.16	6.02	1.86	0.018
map	17	26.5	0.010	34	2.13	0.00	2.61	0.062	29	3.08	0.00	2.15	0.029
potholes	17	88.5	0.046	34	0.83	0.00	0.85	0.074	31	0.75	0.00	1.66	0.034
a	22	52.7	0.022	44	0.11	0.00	0.25	0.118	36	0.04	0.00	0.15	0.047
rooms	22	165.9	0.016	44	0.83	0.141	0.170	0.141	25	1.36	0.17	0.79	0.058
$dense_4$	53	179.1	0.198	106	13.76	5.85	3.77	1.092	188	8.85	4.85	2.02	0.299
$potholes_2$	68	154.5	0.092	136	5.61	2.75	1.20	1.461	154	4.43	2.37	1.13	0.424
m31	71	39.0	0.245	142	7.06	4.34	1.21	3.327	396	5.82	4.51	1.11	0.456
$warehouse_4$	79	369.2	0.074	158	6.27	2.20	2.39	2.091	233	4.57	2.28	1.27	0.534
jh_2	80	201.9	0.116	160	1.63	0.35	0.67	2.122	228	1.48	0.43	0.71	0.534
pb_4	104	654.6	0.043	208	0.64	0.05	0.28	2.846	443	0.10	0.00	0.11	0.578
ta2	141	328.0	0.070	282	3.04	2.11	0.49	5.154	421	3.21	2.14	0.65	0.918
$h2_5$	168	943.0	3.413	336	2.06	1.19	0.52	28.972	263	1.95	1.03	0.63	2.999
$potholes_1$	282	277.3	0.498	564	6.07	4.75	0.65	28.475	814	5.25	3.82	0.73	2.563
jh_1	356	363.7	0.644	712	3.87	2.84	0.36	49.823	1066	3.90	2.76	0.69	3.410
pb1.5	415	839.6	0.446	830	2.21	1.21	1.50	50.553	1436	1.43	0.91	0.24	3.360
$h2_2$	568	1316.2	6.107	1136	2.69	2.00	0.43	337.014	1352	2.26	1.61	0.43	9.096
ta_1	574	541.1	1.065	1148	5.59	4.55	0.56	100.442	1367	5.08	4.35	0.54	4.195

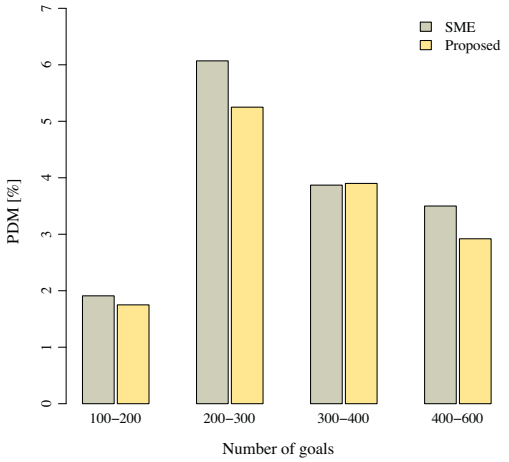


Fig. 8. Average values of the solution quality for the multi-goal path planning problems.

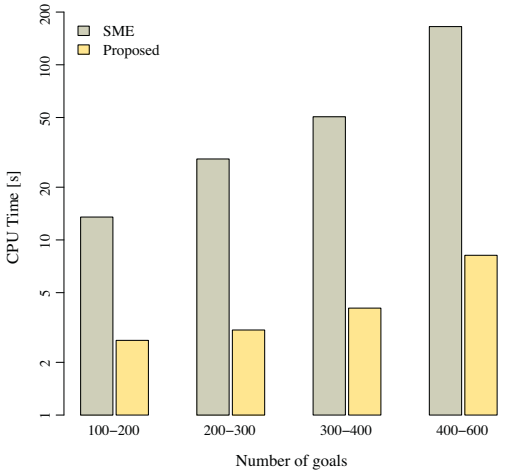


Fig. 9. Average values of the required computational time for the multi-goal path planning problems.

burden without noticeble changes to the solution quality. Besides, the reference solutions of the examined problems are found as solutions of the decoupled approach using the concorde solver, see Section 3.1. The full path refinement of the approximate shortest path in W is used in all algorithms; the node-goal paths are used for the TSP while approximate shortest paths between two points are used for the d-WRP.

The SOM algorithms are randomized, and therefore, 20 solutions are found for each problem and particular algorithm. The quality of solution is measured as the percent deviation to the reference path length of the mean solution value, $PDM = (\bar{L} - L_{ref})/L_{ref} \cdot 100\%$, and as the percent deviation from the reference of the best solution value (PDB), where L_{ref} is the length of the reference path. Besides, $s_L\%$ is denoted to the percent sample variance of the path length to the mean solution value.

All algorithms have been implemented in C++ and compiled by the G++ 4.2 with the −O2 optimization flag. All results have been obtained within the same computational environment using single core of the Athlon X2 5050e CPU at 2.6 GHz and 2 GB RAM running FreeBSD 8.1. Thus, all presented required computational times can be directly compared.

The proposed adaptation schema has been studied in a set of multi-goal path planning problems, the experimental results for the final found parameters are presented in the next subsection. Then, the parameters found in the evaluation of the multi-goal path planning have been used in the experimental evaluation of the inspection planning problems for various visibility ranges. The results are presented in Section 6.2.

6.1. Multi-goal path planning – the non-Euclidean TSP

The examined multi-goal path planning problems consist of polygonal maps and a set of goals (sensing locations). The maps represent real and artificial environments used for examination of path and motion planning approaches. The used approximation of the shortest path depends on the numbers of vertices and convex polygons, therefore, to provide an overview of maps' relation to the algorithm performance, the basic maps properties are depicted in Table 1. Moreover, all paths from vertices to goals are pre-computed for the node–goal path approximation.

Detailed experimental results of the SME schema and the proposed adaptation schema are shown in Table 2. The most time consuming preparation step is computation of all shortest paths from vertices to all goals, the time is denoted as T_{init} in the table. Construction of the supporting convex partition, and the visibility graph is negligible in comparison to the required computation time of the adaptation T_a. For the largest problem $h2_2$ the convex partition is found in 220 ms, and the visibility graph is found in 150 ms.

The proposed adaptation schema provides solutions with a higher quality than the SME schema. The required computational times are not significantly different for small problems, but for larger problems the proposed algorithm provides better solution in less computational time. An overview of the algorithms performance as average values of the solution quality measured by the PDM, and average values of the required computational time including T_{init} are shown in Figs. 8 and 9 as histograms for the number of goals. Examples of found solutions are depicted in Fig. 10.

Here, it should be noted that for SME schema various initialization have been considered. Also the number of nodes has been increased up to $m = 3n$. However, the changes of the solution quality are below s_L, and only the computational requirements are increased for higher values of m. Based on these observations, $2n$ nodes are initialized as a small ring around the first goal for the SME algorithm in the all presented experimental results.

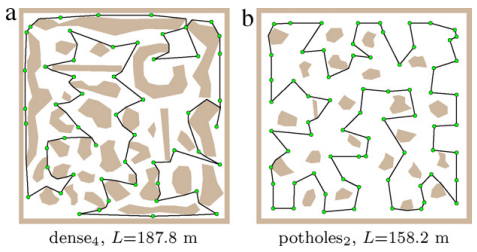


Fig. 10. Examples of the best solutions of the multi-goal path planning problem found by the proposed adaptation schema.

Appendix 2 - Faigl, J. - Přeučil, L.: *Inspection Planning in the Polygonal Domain by Self-Organizing Map* [28], referenced on page 10.

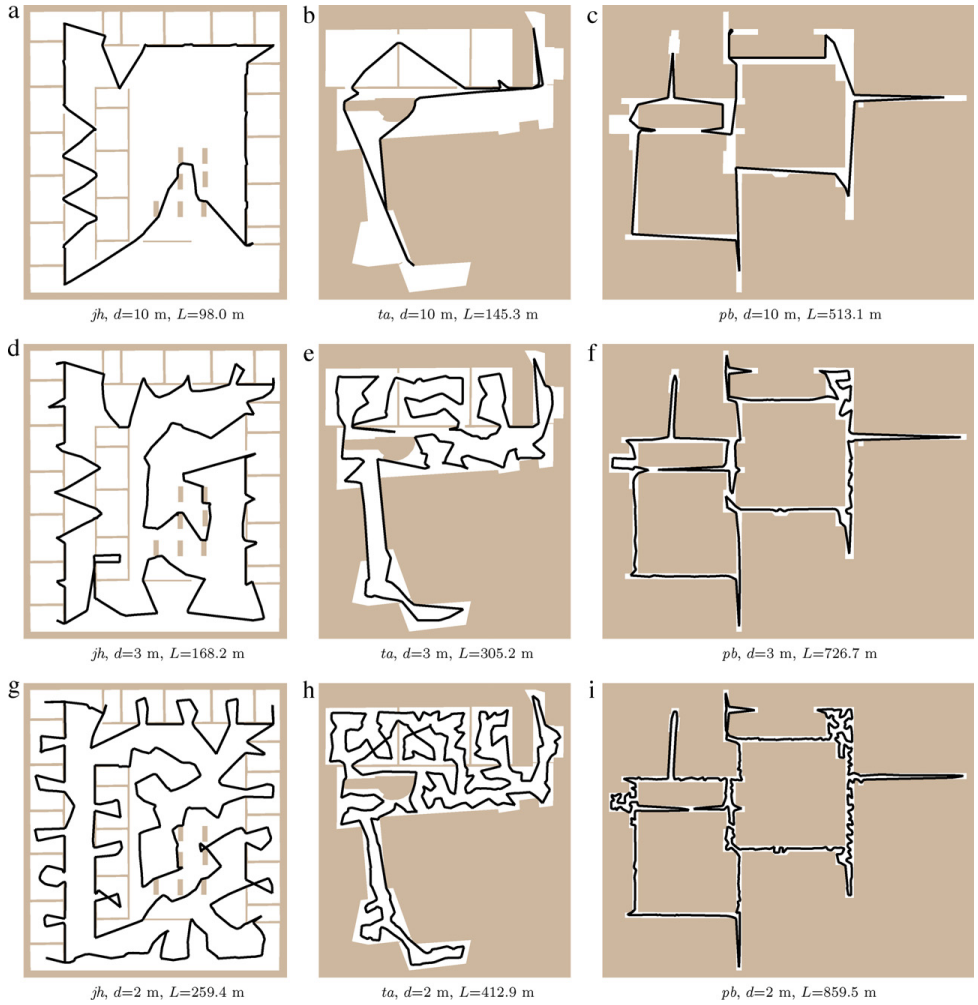


Fig. 11. Best found *d*-WRP solutions by the proposed adaptation schema.

6.2. Inspection planning with restricted visibility range

Three maps of real environments[1] denoted as *jh*, *ta* and *pb* are used for discrete and continuous sensing evaluation. The examined visibility ranges are from the set { *inf*, 10.0, 5.0, 4.0, 3.0, 2.0, 1.5, 1.0} meters, where *inf* denotes the unrestricted visibility range. The experimental results of the proposed algorithms are presented in the following sections.

6.2.1. Inspection planning with discrete sensing

A set of sensing locations is found by the sensor placement algorithm [17] for each polygonal map and the visibility range *d*. The algorithm has been selected mainly due to its similarity to the used supporting triangular mesh in the *d*-WRP algorithm. The sensing locations are goals in the multi-goal path planning problem that is solved as the TSP like in the previous experiments.

Detail experimental results are presented in Table 3. Notice the higher number of nodes *m* for the proposed schema. However, the required computational time is approximately 30 times lower for the largest problem. Moreover, the solution quality is better or competitive to the SME schema.

The results show that the proposed adaptation schema provides better results in less computational time. Even though the solution quality improvements are only in units of percents, the speedup improvements are in tens. The proposed schema typically finished the adaptation with a higher number of nodes (about more than 20%) than the SME algorithm. The higher number of nodes together with lower computational requirements indicate that a lower number of nodes is actually adapted. The proposed adaptation rules

[1] The maps represent real testing environments of the search and rescue mission experiments of the IST-2001-FET project number 38873 – PeLoTe – Building Presence through Localization for Hybrid Telematic Systems [34].

Table 3
Experimental results for the inspection planning with discrete sensing.

Map	d (m)	n	L_{ref} (m)	T_{init} (s)	SME					Proposed				
					m	PDM	PDB	s_L%	T_a (s)	m	PDM	PDB	s_L%	T_a (s)
jh	*inf*	77	193.3	0.12	154	1.55	0.51	0.57	2.0	213	1.04	0.59	0.47	0.6
jh	10.0	78	194.6	0.12	156	2.07	0.94	0.91	2.1	216	1.67	0.51	0.80	0.6
jh	5.0	85	204.3	0.12	170	1.98	0.81	0.67	2.6	254	1.64	0.84	0.61	0.7
jh	4.0	89	207.9	0.14	178	2.32	1.02	0.84	2.8	248	1.55	0.59	0.60	0.7
jh	3.0	100	215.5	0.14	200	2.09	0.96	0.75	3.6	297	2.04	0.42	1.40	0.8
jh	2.0	180	295.5	0.23	360	3.12	2.12	0.48	11.7	445	2.84	1.62	0.70	1.8
jh	1.5	282	359.0	0.44	564	3.47	2.63	0.48	29.1	673	2.78	1.88	0.55	2.7
jh	1.0	563	485.0	1.48	1126	4.10	3.54	0.34	125.5	1678	4.19	3.54	0.46	5.7
ta	*inf*	46	215.6	0.03	92	1.74	0.34	1.61	0.6	137	1.17	0.00	1.37	0.2
ta	10.0	47	216.9	0.03	94	2.74	0.66	2.20	0.6	131	1.05	0.11	0.78	0.2
ta	5.0	70	256.8	0.4	140	1.52	0.44	0.83	1.5	210	0.57	0.19	0.34	0.4
ta	4.0	93	291.3	0.06	186	2.19	1.56	0.53	2.6	238	2.79	1.46	1.06	0.6
ta	3.0	138	335.6	0.08	276	1.42	1.00	0.32	6.1	397	1.86	0.98	0.73	1.2
ta	2.0	255	427.0	0.21	510	3.95	3.36	0.37	22.0	707	4.20	2.67	0.52	2.0
ta	1.5	432	538.5	0.64	864	5.56	4.65	0.54	62.0	1068	5.40	4.65	0.45	3.2
ta	1.0	934	774.3	3.48	1868	6.24	5.48	0.36	327.1	2207	5.77	4.61	0.45	8.9
pb	*inf*	50	554.9	0.05	100	3.81	0.10	4.19	0.7	150	2.34	0.00	3.48	0.2
pb	10.0	76	615.9	0.07	152	1.06	0.29	2.11	1.6	235	0.37	0.18	0.09	0.4
pb	5.0	134	687.2	0.11	268	0.68	0.37	0.19	5.0	650	0.15	0.00	0.10	0.9
pb	4.0	165	721.9	0.13	330	2.15	0.78	2.29	7.8	526	0.73	0.42	0.31	1.2
pb	3.0	244	781.6	0.21	488	1.80	0.52	2.12	17.6	732	0.76	0.53	0.15	1.7
pb	2.0	473	919.0	0.68	946	2.54	1.42	2.09	68.3	1418	1.62	1.33	0.17	3.7
pb	1.5	870	1158.2	2.30	1740	2.79	2.41	0.63	241.5	2581	2.45	1.97	0.26	8.2
pb	1.0	1845	1606.6	13.33	3690	4.15	3.71	0.22	1186.4	4368	3.78	3.17	0.33	41.4

with decreasing σ and δ based on the number of performed adaptation steps decrease the computational burden. However, the rules also decrease the solution quality that is "compensated" by the proposed winner selection method.

The utilized approximate shortest path uses pre-computed paths from map vertices to the all goals. The required memory footprint of the algorithm is about 330 MB for the largest problem *pb* with $d = 1$ m. A typical value of the required memory by the program without the pre-computed paths is about 20 MB, which provides an estimation of the real space requirements of the supporting structures.

6.2.2. d-WRP – inspection planning with continuous sensing

The proposed d-WRP algorithm utilizes a triangular mesh and a convex cover set build on top of the mesh triangles. The number of triangles and convex polygons of the cover set has influence to the algorithm performance. For each map and a particular visibility range a triangular mesh has been created individually by the quality mesh generator `triangle` [35] for the required minimal angle 32.5°, and 25.0° for the map *jh*, and a selected maximum triangle area. The area is experimentally set according to the circumscribed circle of the triangle, which radius is derived from the restricted visibility range d. Particular properties of the used meshes are depicted in Table A1.

In the d-WRP algorithm, path queries are resolved by the approximation of the shortest path between two points in W using the convex partition of W. Because of the relatively small number of vertices in the examined maps, the initialization of the shortest path between map vertices is not computationally demanding like in the TSP algorithm. Nevertheless, the time is included in the presented results like in the discrete sensing. Construction of the triangular mesh, the convex polygon partition, and the visibility graphs is done in a fraction of second. Also a convex cover set for the largest problem, regarding the number of triangles, is found in hundreds of milliseconds. In comparison to the required computational time of the adaptation procedure the required times to create the supporting structures are negligible. Moreover, in comparison to the decoupled approach of the inspection planning the problem of determining a set of sensing locations can be more computationally demanding [10].

In the algorithm based on the SME schema, the number of nodes is set individually according to the number of triangles of used triangular mesh, see Table A1 and m in Table 4. For the proposed adaptation schema, the initial number of nodes is set to the tenth of the number of triangles, $m = 0.1N_T$.

Detail experimental results are presented in Table 4. The proposed adaptation schema provides solution in a less computational time. However, in several cases, the found solutions have worse quality than solutions provided by the SME schema. Regarding the PDB the proposed algorithm provides shorter inspection paths than the reference solutions in all cases. Also for small visibility ranges the proposed adaptation schema provides better results than the SME schema. In all cases, the found solutions provide full coverage of W, and the convergence issue has not been observed.

Examples of the best found solutions for the selected visibility ranges d are presented in Fig. 11. Notice the self-crossing route in Fig. 11h that is caused due to avoidance of the winner node selection to the already covered area. Once the corner is covered, the network does not adapt to that part. Also, such a crossing can be caused by the deletion of inactive nodes, because the shape of the ring can be significantly changed after removing the nodes, and self-crossing can suddenly occur.

The memory footprint of the d-WRP algorithm is smaller than for the multi-goal path planning, because only the vertex–vertex paths are pre-computed. The required memory is about 24 MB for the problem *pb* with $d = 1$ m.

During the experimental verification of the algorithm, a sensitivity to the initialization of the nodes has been observed for the proposed d-WRP algorithm. An initialization as a small ring around a point gives similar results, however in several cases the found solutions were worse than the reference solutions in units of percents. Although the proposed hull initialization provides overall best results, it can also stick the found route in a local solution. The reason for that is similar to self-crossings. Once triangles are covered, the restricted set of the neighbouring nodes does not spread nodes to other parts; thus, the nodes remain close to their previous positions. In the presented results, this can be observed for

Appendix 2 - Faigl, J. - Přeučil, L.: *Inspection Planning in the Polygonal Domain by Self-Organizing Map* [28], referenced on page 10.

Table 4
Experimental results for the inspection planning with continuous sensing – d-WRP.

Map	d (m)	L_{ref} (m)	SME					Proposed				
			m	PDM	PDB	s_L%	T (s)	m	PDM	PDB	s_L%	T (s)
jh	*inf*	193.3	87	−48.99	−49.79	1.88	2.33	129	−45.29	−49.58	9.05	0.31
jh	10.0	194.6	87	−48.96	−50.03	3.04	2.36	128	−46.97	−49.65	4.40	0.31
jh	5.0	204.3	87	−46.95	−49.32	3.22	2.59	154	−43.23	−48.33	5.45	0.34
jh	4.0	207.9	174	−40.62	−44.40	2.83	6.25	206	−33.73	−38.65	5.57	0.40
jh	3.0	215.5	186	−24.43	−25.94	1.21	11.02	321	−17.75	−21.95	4.19	1.29
jh	2.0	295.5	381	−13.30	−14.91	1.08	45.27	831	−7.54	−12.20	3.64	6.44
jh	1.5	359.0	682	−6.20	−7.57	0.85	146.43	1585	−1.75	−5.40	2.90	17.95
jh	1.0	485.0	1701	0.96	−0.29	0.79	948.87	3684	2.33	−0.74	2.80	114.40
ta	*inf*	215.6	101	−34.77	−35.23	0.45	0.53	112	−33.27	−34.59	3.45	0.08
ta	10.0	216.9	101	−32.84	−33.11	0.20	0.79	103	−32.03	−33.01	0.66	0.08
ta	5.0	256.8	101	−16.65	−18.64	1.84	2.40	120	−14.86	−17.96	2.73	0.24
ta	4.0	291.3	203	−13.23	−16.96	1.55	7.70	227	−9.81	−13.82	2.61	0.40
ta	3.0	335.6	376	−11.56	−14.14	1.65	26.21	427	−4.78	−9.08	3.05	1.38
ta	2.0	427.0	778	−3.25	−4.63	0.96	132.46	1263	0.29	−3.31	1.78	9.90
ta	1.5	538.5	1247	−2.05	−3.98	1.06	408.80	2419	−1.64	−4.40	1.09	48.29
ta	1.0	774.3	3522	1.07	1.0	0.74	2993.48	5570	−0.13	−1.47	0.68	347.47
pb	*inf*	554.9	240	−22.25	−23.73	3.82	2.87	254	−20.51	−22.49	4.66	0.59
pb	10.0	615.9	240	−13.08	−15.11	2.75	5.26	244	−14.63	−16.69	1.17	0.48
pb	5.0	687.2	240	−7.92	−9.33	1.63	11.55	267	−7.66	−9.31	1.46	1.23
pb	4.0	721.9	481	−5.92	−8.31	3.19	35.91	318	−6.82	−7.87	0.72	2.51
pb	3.0	781.6	616	−6.38	−7.34	0.61	86.75	811	−6.27	−7.02	0.56	10.22
pb	2.0	919.0	1408	−4.73	−5.37	0.43	503.42	2145	−5.40	−6.47	0.56	56.10
pb	1.5	1158.2	2858	−2.86	−4.24	0.61	2092.47	4614	−3.25	−4.06	0.51	282.68
pb	1.0	1606.6	5785	−0.08	−0.75	0.40	11357.65	11316	−1.40	−2.05	0.44	2239.71

small visibility ranges in the maps *jh* and *ta*, which contain several rooms, and does not occur in the map *pb*. Despite this issue, the found solutions are competitive with the SME schema.

6.3. Comparison of discrete and continuous sensing

An overall comparison of the solution quality for the discrete and continuous sensing approaches is presented in Fig. 12 as a histogram of average values for the visibility distances. Due to shorter paths of the d-WRP solutions than the reference solutions, the PDM is increased about 100%, i.e., 100% is the length of the reference path. The discrete sensing inspection solved as the multi-goal path planning is denoted as the TSP in the figure. Even though the histogram bins represent average values over all maps and particular selected visibility ranges, the histogram shows increasing quality of the d-WRP solution over the TSP for higher visibility ranges. With regard to the required computational time, see Fig. 13, the proposed d-WRP algorithm provides the best results (according to the PDM) in hundreds of milliseconds for high values of d. For small visibility ranges, a density of the sensing locations in the map is high, and d-WRP solutions are only about units of percents shorter.

6.4. Discussion

Regarding the experimental results the found inspection paths for the d-WRP are shorter than for the decoupled approach with the same visibility range d. However, it cannot be clearly stated that the continuous sensing is better than the discrete approach, because the sensing and motion costs have to be taken into account. Considering the available d-WRP algorithms and the presented

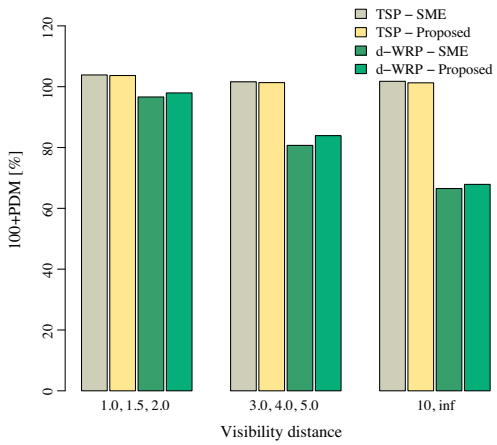


Fig. 12. Average values of the solution quality for the multi-goal path planning problems.

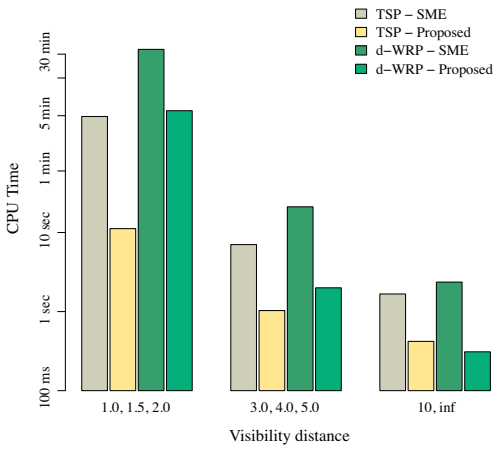


Fig. 13. Average values of the required computational time for the multi-goal path planning problems.

Table A1
Parameters of supporting structures for the WRP.

d (m)	Map *jh*			Map *ta*			Map *pb*		
	N_N	N_T	N_C	N_N	N_T	N_C	N_N	N_T	N_C
inf	576	872	100	638	1014	46	1630	2403	52
10.0	576	872	108	638	1014	70	1630	2403	111
5.0	576	872	130	638	1014	152	1630	2403	262
4.0	576	872	169	638	1014	209	1630	2403	373
3.0	608	931	258	776	1252	357	2018	3078	714
2.0	1183	1904	480	1151	1944	757	2955	4692	1564
1.5	1392	2272	852	1788	3117	1320	4319	7144	2787
1.0	1988	3401	1800	3849	7044	2955	8250	14462	6188

experimental results, it seems that for small visibility ranges the decoupled approach is appropriate. Moreover, a more sophisticated sensor placement algorithm can provide a lower number of sensing locations leading to shorter inspection paths [10]. For higher visibility ranges, the proposed d-WRP algorithm may be used for finding a solution of the sensor placement. Because the found path of the d-WRP is shorter than for the TSP, it is expected that such a solution provides overall better solution for both costs (sensing and motion). The final position of the winner nodes can be used as primal sensing locations, and to achieve complete coverage additional points at the final ring can be selected.

The problem of selection of the smallest set of points at the watchman route is called Vision Points problem in computational geometry. A combination of the proposed d-WRP algorithm and selection of the sensing locations can provide a suitable mechanism to combine the cost of motion with the cost of sensing based on SOM. This problem is tightly related with the problem of nodes deletion in the proposed adaptation schema, because nodes that may not be deleted are eventual candidates to be sensing locations. Even though the proposed three phases adaptation provides "good" solutions in less computational time, this problem needs future investigations that can open new application areas of SOM in the field of visibility problems studied in computational geometry.

7. Conclusion

New adaptation schema for the inspection planning has been presented in this paper. The schema uses new winner selection rule that considers a path between two nodes utilizing a node creation/deletion mechanism. Besides, the schema comprises particular SOM improvements proposed by several authors. The schema has been applied to the discrete and continuous sensing variants of the inspection planning. Both sensing variants are considered with the restricted visibility range. The discrete variant is the multi-goal path planning formulated as the non-Euclidean TSP, and the continuous sensing variant is formulated as the d-WRP.

The schema has been experimentally verified in a set of problems representing the non-Euclidean TSP and the d-WRP. The presented experimental results show that the proposed adaptation schema is faster than the SME schema, it provides better solutions for discrete sensing, and competitive solutions for the d-WRP.

For high visibility ranges, the proposed d-WRP algorithm provides significantly shorter inspection paths in comparison with the solutions for the discrete sensing. During the solution of the d-WRP the winner nodes can be considered as the sensing locations that makes the SOM algorithm applicable to the inspection planning with combination of the sensing and motion costs. Such a combination opens future applications of SOM principles to the similar visibility based routing problems.

Acknowledgements

The work has been supported by the Ministry of Education of the Czech Republic under Project No. 1M0567 and partially under Project No. 7E08006 and by EU Project No. 216342.

Appendix A. Parameters of the support triangular meshs for the WRP

See Table A1.

References

[1] H.H. González-Baños, D. Hsu, J.-C. Latombe, Motion planning: recent developments, in: S.S. Ge, F.L. Lewis (Eds.), Autonomous Mobile Robots: Sensing, Control, Decision-Making and Applications, CRC Press, 2006 (Chapter 10).
[2] W.-P. Chin, S. Ntafos, Optimum watchman routes, in: SCG'86: Proceedings of the Second Annual Symposium on Computational Geometry, ACM Press, Yorktown Heights, New York, United States, 1986, pp. 24–33.
[3] E. Packer, Robust Geometric Computing and Optimal Visibility Coverage, PhD Thesis, Stony Brook University, New York, 2008.
[4] J.C. Culberson, R.A. Reckhow, Covering polygons is hard, Journal of Algorithms 17 (1) (1994) 2–44.
[5] S.N. Spitz, A.A.G. Requicha, Multiple-goals path planning for coordinate measuring machines, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2000, pp. 2322–2327.
[6] M. Saha, G. Sánchez-Ante, J.-C. Latombe, Planning multigoal tours for robot arms, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol. 3, 2003, pp. 3797–3803.
[7] X. Tan, T. Hirata, Finding shortest safari routes in simple polygons, Information Processing Letters 87 (4) (2003) 179–186.
[8] T. Danner, L.E. Kavraki, Randomized planning for short inspection paths, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE Press, San Fransisco, CA, 2000, pp. 971–976.
[9] J. Faigl, M. Kulich, Sensing locations positioning for multi-robot inspection planning, in: DIS'06: Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and its Applications, IEEE Computer Society, Washington, DC, USA, 2006, pp. 79–84.
[10] J. Faigl, M. Kulich, L. Přeučil, A sensor placement algorithm for a mobile robot inspection planning, J. Intell. Robotic Syst. 62 (3) (2011) 329–353.
[11] V. Chvátal, W. Cook, G.B. Dantzig, D.R. Fulkerson, S.M. Johnson, Solution of a Large-Scale Traveling-Salesman Problem, Springer, Berlin Heidelberg, 2010, pp. 7–28 (Chapter 1).
[12] S. Ghafurian, N. Javadian, An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems, Applied Soft Computing 11 (January (1)) (2011) 1256–1262.
[13] E.M. Cochrane, J.E. Beasley, The co-adaptive neural network approach to the Euclidean travelling salesman problem, Neural Networks 16 (10) (2003) 1499–1525.
[14] T.A.S. Masutti, L.N. de Castro, A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, Information Sciences 179 (10) (2009) 1454–1468.
[15] J. Faigl, Approximate solution of the multiple watchman routes problem with restricted visibility range, IEEE Transactions on Neural Networks 21 (10) (2010) 1668–1679.
[16] S. Somhom, A. Modares, T. Enkawa, A self-organising model for the travelling salesman problem, Journal of the Operational Research Society 48 (1997) 919–928.
[17] G.D. Kazazakis, A.A. Argyros, Fast positioning of limited visibility guards for the inspection of 2d workspaces, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Lausanne, Switzerland, 2002 September, 2002.
[18] R. Seidel, A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons, Computational Geometry 1 (1) (1991) 51–64.

Appendix 2 - Faigl, J. - Přeučil, L.: *Inspection Planning in the Polygonal Domain by Self-Organizing Map* [28], referenced on page 10.

[19] M.H. Overmars, E. Welzl, New methods for computing visibility graphs, in: SCG'88: Proceedings of the Fourth Annual Symposium on Computational Geometry, ACM, New York, NY, USA, 1988, pp. 164–171.
[20] D. Applegate, R. Bixby, V. Chvátal, W. Cook. CONCORDE TSP Solver. http://www.tsp.gatech.edu/concorde.html, 2003 (cited 8 July 2010).
[21] B. Angéniol, G. de la, C. Vaubois, J-Y.L. Texier, Self-organizing feature maps and the travelling salesman problem, Neural Networks 1 (1988) 289–293.
[22] J.C. Fort., Solving a combinatorial problem via self-organizing process: an application of the Kohonen algorithm to the traveling salesman problem, Biological Cybernetics 59 (1) (1988) 33–40.
[23] A. Plebe, A.M. Anile, A neural-network-based approach to the double traveling salesman problem, Neural Computing 14 (2) (2002) 437–471.
[24] F.C. Vieira, A.D.D. Neto, J.A. Costa, An efficient approach to the travelling salesman problem using self-organizing maps, International Journal of Neural Systems 13 (2) (2003) 59–66.
[25] Y. Bai, W. Zhang, Z. Jin, An new self-organizing maps strategy for solving the traveling salesman problem, Chaos, Solitons & Fractals 28 (4) (2006) 1082–1089.
[26] R. Gerhard, TSPLIB – a traveling salesman problem library, Journal on Computing 3 (4) (1991) 376–384.
[27] S. Somhom, A. Modares, T. Enkawa, Competition-based neural network for the multiple travelling salesmen problem with minmax objective, Computers and Operations Research 26 (4) (1999) 395–407.
[28] W. Zhang, Y. Bai, H.P. Hu, The incorporation of an efficient initialization method and parameter adaptation using self-organizing maps to solve the TSP, Applied Mathematics and Computation 172 (1) (2006) 603–623.
[29] J. Faigl, Multi-Goal Path Planning for Cooperative Sensing, PhD Thesis, Czech Technical University in Prague, 2010.
[30] M. Kallmann, Path planning in triangulations, in: Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games, Edinburgh, Scotland, July 31, 2005.
[31] O. Devillers, S. Pion, M. Teillaud, P. Prisme, Walking in a triangulation, International Journal of Foundations of Computer Science 13 (2001) 106–114.
[32] J. Faigl, M. Kulich, V. Vonásek, L. Přeučil, An application of self-organizing map in the non-Euclidean traveling salesman problem, Neurocomputing 74 (5) (2011) 671–679.
[33] L Burke, "Conscientious" neural nets for tour construction in the traveling salesman problem: the vigilant net, Computers and Operations Research 23 (2) (1996) 121–129.
[34] L. Přeučil, J. Pavlček, R. Mázl, F. Driewer, K. Schilling, Next generation human–robot telematic teams, in: Multidisciplinary Collaboration for Socially Assistive Robotics, AAAI Press, Menlo Park, CA, 2007, pp. 65–69.
[35] J.R. Shewchuk, Delaunay refinement algorithms for triangular mesh generation, Computational Geometry: Theory and Applications 22 (2001) 1–3.

Appendix 3 - Faigl, J. - Přeučil, L.: *Self-Organizing Map for the Multi-Goal Path Planning with Polygonal Goals* [29], referenced on page 11.

Self-Organizing Map for the Multi-Goal Path Planning with Polygonal Goals

Jan Faigl and Libor Přeučil

Czech Technical University in Prague - Department of Cybernetics
{xfaigl,preucil}@labe.felk.cvut.cz

Abstract. This paper presents a self-organizing map approach for the multi-goal path planning problem with polygonal goals. The problem is to find a shortest closed collision free path for a mobile robot operating in a planar environment represented by a polygonal map $\mathcal{W}$. The requested path has to visit a given set of areas where the robot takes measurements in order to find an object of interest. Neurons' weights are considered as points in $\mathcal{W}$ and the solution is found as approximate shortest paths connecting the points (weights). The proposed self-organizing map has less number of parameters than a previous approach based on the self-organizing map for the traveling salesman problem. Moreover, the proposed algorithm provides better solutions within less computational time for problems with high number of polygonal goals.

1 Introduction

A problem of finding a collision-free path for a mobile robot such that the robot visits a given set of goals is called the multi-goal path planning problem (MTP). The problem arises in various robotic tasks and one of them is an inspection task in which model of the robot work space is a priori known. A model can be a building plan that can be represented as the polygonal domain, i.e., a polygonal map with obstacles. In such a map, a goal can be a single point or a polygonal region. Goals represent places in the environment where a mobile robot takes measurements. A practical motivation for this type of problems are searching missions where a mobile robot has to inspect the environment to find an object of interest, e.g., victims in search&rescue missions [7].

The planning problem for point goals can be formulated as the well-known traveling salesman problem (TSP), and for which many self-organizing map (SOM) approaches have been proposed since the first work of Angéniol and Fort. In the case of polygonal goals, the problem formulation can be found as the safari route problem [8], or the zookeeper problem [2]. These problems can be solved in a polynomial time for particular restricted problem formulations, e.g., problems without obstacles, with a given starting point, and polygonal goals attached to the boundary. However, these problem variants can be formulated as the traveling salesman problem with neighborhoods (TSPN) [6]. Although approximation algorithms for restricted variants of the TSPN exist [3,1], in general, the TSPN is APX-hard and cannot be approximated with a factor $2 - \epsilon$, where $\epsilon > 0$, unless P=NP [9].

T. Honkela et al. (Eds.): ICANN 2011, Part I, LNCS 6791, pp. 85–92, 2011.

Here, it is worth to mention that SOM approaches for the TSP are focused on its Euclidean variant, i.e., distances between nodes and goals are determined as the Euclidean distances between two points. The main difference of the MTP is that a path between two goals (or node–goal path) has to be collision free; thus, geodesic paths (distances) avoiding the collision with obstacles have to be considered in the self-organizing procedure, which increases the complexity of the adaptation process.

In this paper, new SOM adaptation procedure for the MTP with polygonal goals is proposed. The approach follows standard SOM adaptation schema for the TSP that has been extended to the polygonal domain using approximate shortest path in [5]. The adaptation uses new winner selection procedure that finds and creates new neurons using a distance to a segment of the goal. Moreover, practical aspects of the adaptation process in the polygonal map are considered to decrease the computation burden of the adaptation. In addition, simplified adaptation rules based on [11] are used and together with the novel winner selection procedure they lead to less number of adaptation parameters. The proposed procedure is also able to deal with point goals. As such, it provides a unified way to solve various modifications of the MTP, which includes safari route problem and also the watchman route problem as a variant of the MTP where goals are polygons of a convex cover set of $\mathcal{W}$ [4].

2 Self-Organizing Map for Multi-Goal Path Planning with Polygonal Goals

The problem addressed in this paper can be defined as follows. Having a polygonal map $\mathcal{W}$ and a set of goals $\mathbf{G} = \{g_1, \ldots, g_n\}$, the problem is to find a closed shortest path such that the path visits at least one point of each goal $g_i \in \mathbf{G}$. A goal can be a single point, or a polygonal region, and all goals entirely lie in $\mathcal{W}$. A polygonal goal g is represented as a sequence of points $g = (p_1^g, \ldots, p_k^g)$, which forms a border of g represented as a set of straight line segments $\delta g = \{s_1^g, s_2^g, \ldots, s_k^g\}$, where s_i^g is a straight line segment inside $\mathcal{W}$, $s_i^g = (p_i^g, p_{i+1}^g)$ for $0 \leq i < k$, and $s_k^g = (p_k^g, p_1^g)$.

The proposed adaptation procedure is based on two-layered competitive neural network. The input layer consists of two dimensional input vector. An array of output units is the second layer, and it forms a uni-dimensional ordered structure. The neuron's weights represent coordinates of a point in $\mathcal{W}$, which is called node, and denoted as ν in this paper. Connected nodes form a ring that represents the requested path. In SOM for the TSP (for example [10]), goals are presented to the network in a random order and neurons compete to be the winner using the Euclidean distances between them and the goal. Then, the winner node is adapted towards the presented goal. However, in the MTP, a collision free path has to be determined because of obstacles in $\mathcal{W}$. The adaptation process may be considered as a node movement along the node–goal path towards the goal, i.e., the node (neuron's weights) is placed on the path closer to the goal while it travels distance according to the neighbouring function f.

An approximation of the shortest path may be used for the node–goal path determination [5].

Novel winner selection procedure is proposed to address polygonal goals. The procedure is based on consideration of the ring as a sequence of straight line segments in $\mathcal{W}$. Again, due to obstacles in $\mathcal{W}$, such a sequence is found using an approximate shortest path between two points (point–point path) in $\mathcal{W}$ [4].

Let the ring r be a sequence of line segments $r = (s_1^r, s_2^r, \ldots, s_l^r)$. The winner node is found as a "closest" point of the ring to the set of segments representing the goal g. The exact shortest path between two segments in $\mathcal{W}$ is substituted by the following approximation. First, the Euclidean distance between the segments s_i^r and s_j^g is determined; thus, two points on the segments are found, $p_r \in s_i^r$ and $p_g \in s_j^g$. The point–point path for these points is found to approximate the shortest path between two segments in $\mathcal{W}$. So, a pair (p_r, p_g) with the minimal length of the approximate shortest path between p_r and p_g is the result of the winner selection procedure. The point p_r is used for creating new node if a node with the same coordinates is not already in the ring. The found point p_g at the goal segment is used as a point goal towards which nodes are adapted using the point–point path. In the case of a point goal g, a similar procedure is used for approximating shortest segment–point path and p_g is the point goal itself.

The adaptation is an iterative stochastic procedure starting with an initial creation of m nodes, where $m = 2n$ and n is the number of goals. The neurons' weights are set to form a small circle around the first goal g_1, or around the centroid of g_1 for the polygonal goal. The used neighbouring function is $f(\sigma, d) = exp(-d^2/\sigma^2)$ for $d < 0.2m$, and $f(\sigma, d) = 0$ otherwise, where σ is the learning gain (the neighbouring function variance) and d is the distance of the adapted node from the winner node measured in the number of nodes (the cardinal distance). The adaptation process performs as follows.

1. *Initialization:* For a set of n goals $\mathbf{G}$ and a polygonal map $\mathcal{W}$, create $2n$ nodes around the centroid of the first goal. Let the initial value of the leaning gain be $\sigma = 10$, and adaptation parameters be $\mu = 1$, $\beta = 10^{-5}$, and $i = 1$.
2. *Randomizing:* Create a random permutation of goals $\Pi(\mathbf{G})$.
3. *Winner Selection:* For a goal $g \in \Pi(\mathbf{G})$ and the current ring r as a path in $\mathcal{W}$ find the pair (p_r, p_g) using the proposed winner selection procedure. Create a new node ν with coordinates p_r if such a node does not already exist. A node at the coordinates p_r is the winner node $\nu^\star$.
4. *Adapt:* If g is a point goal or $\nu^\star$ is not inside the polygonal goal g:
   - Let the current number of nodes be m, and N be a set of $\nu^\star$'s neighborhoods in the cardinal distance less than or equal to $0.2m$.
   - Move $\nu^\star$ along approximate shortest path $S(\nu^\star, p_r)$ towards p_r by the distance $|S(\nu^\star, p_r)|\mu$, where $|S(.,.)|$ is the length of the approximate path.
   - Move nodes $\nu \in N$ for which $\mu f(\sigma, d) < \beta$ towards p_r along $S(\nu, p_r)$ by the distance $|S(\nu, p_r)|\mu f(\sigma, d)$, where f is the neighbouring function and d is the cardinal distance of ν to $\nu^\star$.

   Remove g from the permutation, $\Pi(\mathbf{G}) = \Pi(\mathbf{G}) \setminus \{g\}$, and if $|\Pi(\mathbf{G})| > 0$ go to Step 3.

5. *Ring regeneration:* Create a new ring as a path in $\mathcal{W}$ using only the winner nodes of the current adaptation step, i.e., remove all other nodes. Make nodes from the endpoints of $s^r \in r$ that do not correspond to the winners, i.e., nodes correspond to the sequence of path's vertices.
6. *Update adaptation parameters:* Set $i = i{+}1$, $\sigma = (1{-}0.001i)\sigma$, and $\mu = 1/\sqrt[4]{i}$.
7. *Termination condition:* If all polygonal goals have particular winner inside the polygonal goal, and if all point goals have the winner in a sufficient distance, e.g., less than 10^{-3}, or $\sigma < 10^{-4}$ Stop the adaptation. Otherwise go to Step 2.
8. *Final path construction:* Use the last winners to determine the final path using point–point approximate path in $\mathcal{W}$.

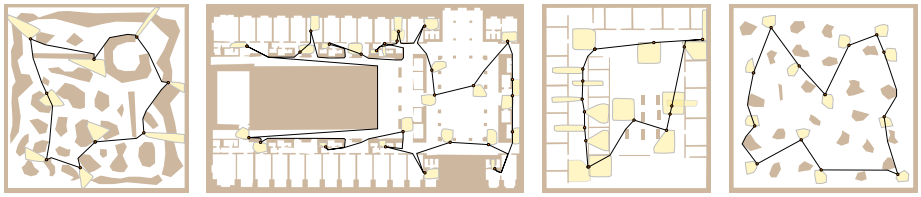
It is clear that the proposed adaptation procedure considering ring as a collision free path in $\mathcal{W}$ with the closest ring–goal segments selection is more computationally demanding than a consideration of node–goal points, which does not require determination of shortest path between two nodes. The adaptation performed only if $\mu f(\sigma, d) < \beta$ (called $\beta - condition$ rule) decreases the computational burden without significant influence to the solution quality. Also the used evolution of σ, μ [11] provides fast convergence. However, it decreases the solution quality in few cases in comparison to Somhom's parameters [10] used in [5,4]. An experimental comparison of these algorithms is presented in Section 3.

Regarding the necessary parameters settings the main advantage of the proposed procedure is that it does not require specific parameters tuning. Based on several experiments the procedure seems to be insensitive to changes of the initial values of σ and μ. Also the used size of the winner neighborhood $(0.2m)$ provides the best trade-off between the solution quality and computational time.

It is worth to mention that the used approximation of the shortest path between two points (described in [4]) is more computationally demanding, and it is less precise than the node–goal path approximation. However, it requires less memory. It is because precomputed shortest paths from all map vertices to the goals are used in the node–goal path queries. Thus, lower memory requirements and a faster initialization are additional advantages of the proposed method.

3 Experiments

The proposed adaptation procedure has been experimentally verified in two sets of problems with polygonal goals, and compared with the SOM approach for the watchman route problem (WRP) [4]. The first set represents a "generalized" safari route problem, where convex polygonal goals, possibly overlapping each other, are placed in $\mathcal{W}$. The second set represents the WRP with restricted visibility range presented in [4]. Moreover, the proposed procedure has been compared with the SOM approach for the TSP in $\mathcal{W}$ [5] where goals are points.



(a) $dense_5$-A (b) $h2_5$-A (c) jh_4-A (d) $potholes_2$-A

Fig. 1. Selected solutions of the safari route problems, light polygons are goals, small disk at convex goal are the last winner nodes, black lines are found paths

The WRP algorithm adapts nodes towards centroids of the convex polygonal goals[1]. An alternate point is determined at the polygon border using node–centroid path to avoid placement of nodes too close to the polygon centroid, i.e., the node movement towards the centroid is stopped at the border. For the safari route problem, the WRP algorithm has been modified to do not consider the ring coverage, and to adapt nodes towards the determined alternate points. Besides, the WRP and the TSP algorithms has been modified to use the $\beta - condition$ rule and the Euclidean distance for pre-selection of winner nodes candidates, i.e., approximate node–goal path is determined only if the Euclidean node–goal distance is less than the distance of the current winner node candidate to the goal. These two modifications are technical, as they do not affect the solution quality; however, they decrease the computational burden several times.

The examined algorithms have been implemented in C++, compiled by the G++ 4.2.1 with the -O2 optimization, and executed within the same computational environment using single core of the i7-970 CPU at 3.2 GHz, and 64-bit version of the FreeBSD 8.2. Thus, the presented average values of the required computational times T can be directly compared.

The SOM algorithms are randomized, and therefore, each problem has been solved 50 times, and the average length of the path L, the minimal found path length L_{min}, and the standard deviation in percents of L denoted as $s_L\%$ are used as the quality metrics. Reference solutions from [4,5] are used for the WRPs and the TSPs, and the solution quality is measured as the percent deviation to the reference path length of the average path length, $PDM = (L - L_{ref})/L_{ref} \cdot 100\%$, and as the percent deviation from the reference of the best solution, $PDB = (L_{min} - L_{ref})/L_{ref} \cdot 100\%$. All presented length values are in meters. The number of goals is denoted as n in the presented tables.

The experimental results for the safari route problems are presented in Table 1 and selected best solutions found by the proposed algorithm are depicted in Figure 1. The proposed procedure provides better solutions for most of the problems. The procedure is more computationally demanding for complex environments like the problem $h2_5$-A because shortest paths have many segments. This is also the case of the jh_{10}-coverage problem, which is an instance of the WRP with many overlapping convex goals.

[1] In [4], triangles of a triangular mesh are used to support determination of ring coverage, which is not necessary for safari route problems.

Table 1. Experimental results for the safari route problems

Problem	n	SOM for WRP [4]				Proposed			
		L	s_L%	L_{min}	T [s]	L	s_L%	L_{min}	T [s]
dense-small	35	114.2	3.45	105.63	0.34	113.7	3.99	102.80	0.98
dense$_5$-A	9	62.6	1.96	60.66	0.14	59.0	2.77	58.05	0.23
h2$_5$-A	26	407.2	0.98	399.34	1.22	405.2	0.88	396.07	2.12
jh-rooms	21	88.3	0.76	87.84	0.13	88.1	0.10	87.83	0.15
jh$_{10}$-doors	21	67.6	1.34	66.11	0.16	63.7	1.43	61.99	0.15
jh$_{10}$-coverage	106	106.9	1.34	103.89	1.49	97.9	6.20	92.99	2.66
jh$_4$-A	16	61.1	1.86	58.71	0.33	57.3	1.32	56.59	0.32
jh$_5$-corridors	11	65.8	1.87	62.77	0.14	59.7	0.35	59.53	0.20
pb$_5$-A	7	275.8	4.47	265.29	0.31	271.7	4.36	264.70	0.31
potholes$_2$-A	13	71.9	1.91	70.37	0.04	71.6	2.08	70.09	0.08

Table 2. Experimental results for the WRP

Map	d [m]	n	L_{ref} [m]	SOM for the WRP [4]				Proposed				
				PDM	PDB	s_L%	T [s]	PDM	PDB	L_{min}	s_L%	T [s]
jh	inf	100	207.8	-52.67	-53.39	1.53	1.45	-53.71	-54.17	95.27	2.78	2.40
jh	10.0	108	207.3	-51.84	-53.02	3.27	1.95	-50.65	-54.02	95.30	6.89	2.64
jh	5.0	130	216.4	-48.67	-51.75	3.39	1.27	-51.54	-53.06	101.56	4.18	5.75
jh	4.0	169	219.9	-43.48	-46.34	3.22	2.97	-48.38	-49.42	111.22	2.71	9.21
jh	3.0	258	225.5	-27.92	-30.60	1.61	5.18	-35.04	-37.04	142.01	2.27	13.12
jh	2.0	480	281.9	-8.99	-11.09	1.06	20.68	-17.25	-19.85	225.91	1.64	23.16
jh	1.5	852	350.3	-3.81	-5.51	1.02	109.81	-14.56	-15.68	295.39	0.74	100.40
jh	1.0	1800	470.8	3.96	2.36	0.59	430.88	-9.06	-10.25	422.50	0.55	452.03
pb	inf	52	533.3	-18.11	-22.26	4.98	1.44	-16.18	-23.18	409.69	5.70	1.28
pb	10.0	111	612.7	-12.48	-14.86	3.92	2.57	-15.46	-17.94	502.78	4.73	3.46
pb	5.0	262	682.9	-7.35	-9.34	2.45	5.56	-7.01	-10.62	610.38	4.45	15.23
pb	4.0	373	720.1	-6.17	-8.78	3.25	16.80	-7.46	-10.09	647.41	3.16	20.37
pb	3.0	714	774.8	-5.62	-6.72	0.55	42.52	-3.04	-9.54	700.81	6.95	114.08
pb	2.0	1564	901.9	-2.88	-4.41	1.02	244.72	-0.30	-9.40	817.12	4.53	373.74
pb	1.5	2787	1115.9	1.03	0.07	0.54	997.68	-9.12	-12.12	980.59	2.27	1078.42
pb	1.0	6188	1564.2	2.55	1.90	0.41	5651.06	-12.52	-13.89	1346.87	0.78	3276.43
ta	inf	46	203.6	-30.99	-31.48	0.52	0.28	-33.67	-33.94	134.52	1.69	0.76
ta	10.0	70	202.6	-28.11	-28.80	0.28	0.41	-28.36	-28.89	144.08	1.45	1.63
ta	5.0	152	254.1	-15.68	-17.97	1.81	1.26	-19.61	-20.35	202.39	0.83	6.39
ta	4.0	209	272.2	-7.39	-9.91	1.36	3.69	-15.70	-16.65	226.90	0.85	11.64
ta	3.0	357	315.0	-6.28	-8.75	1.61	12.61	-13.46	-14.42	269.57	1.30	15.58
ta	2.0	757	408.3	1.09	-1.20	0.87	66.48	-10.97	-12.52	357.18	1.00	59.00
ta	1.5	1320	522.1	1.06	-1.18	0.97	194.25	-12.81	-13.63	450.92	0.59	251.08
ta	1.0	2955	743.6	5.21	3.80	0.57	1398.71	-12.45	-13.54	642.89	0.57	987.77

The results for the WRP are presented in Table 2, where d denotes the restricted visibility range. Also in this type of problems, the proposed procedure provides better solutions. Although the procedure is more computationally demanding for small problems, it provides significantly better results with less required computational time for problems with d=1 m, which have many convex polygons. The results indicate that the proposed procedure scales better with

Table 3. Experimental results for the TSP

Problem	n	L_{ref} [m]	SOM for the TSP [5] PDM	PDB	s_L%	T [s]	Proposed PDM	PDB	s_L%	T [s]
jari	6	13.6	0.36	0.00	0.55	0.01	0.23	0.00	0.15	0.01
complex2	8	58.5	-0.00	-0.00	0.00	0.01	0.47	-0.00	1.60	0.02
m1	13	17.1	0.31	0.00	1.15	0.02	0.17	0.00	0.20	0.03
m2	14	19.4	9.52	0.00	3.50	0.03	10.76	5.32	3.16	0.04
map	17	26.5	5.92	0.73	4.39	0.05	6.87	0.73	4.37	0.07
potholes	17	88.5	4.58	2.37	2.17	0.06	5.56	2.37	2.48	0.06
a	22	52.7	0.89	0.31	1.00	0.09	1.58	0.31	2.37	0.11
rooms	22	165.9	1.02	0.00	0.86	0.11	0.12	0.00	0.11	0.15
dense_4	53	179.1	15.04	8.33	3.16	0.68	18.17	9.00	2.38	0.68
potholes_2	68	154.5	6.12	2.50	2.01	0.65	7.54	3.11	2.23	0.35
m3_1	71	39.0	6.71	2.29	1.53	1.41	8.72	4.80	1.64	1.00
warehouse_4	79	369.2	5.97	2.42	2.13	1.92	8.47	2.87	2.68	0.81
jh_2	80	201.9	1.94	0.48	0.64	0.95	2.04	0.67	0.66	0.71
pb_4	104	654.6	1.06	0.01	1.34	1.53	1.95	0.51	3.05	0.84
ta_2	141	328.0	2.97	1.69	0.69	2.27	3.69	2.19	0.75	1.11
h2_5	168	943.0	2.85	2.00	0.60	8.75	2.42	1.65	0.53	6.70
potholes_1	282	277.3	6.84	4.91	1.02	10.47	6.97	4.19	0.91	2.71
jh_1	356	363.7	4.02	2.74	0.56	22.29	4.32	3.23	0.46	7.05
$\text{pb}_{1.5}$	415	839.6	2.60	1.12	2.25	24.13	10.40	1.47	5.21	6.62
h2_2	568	1 316.2	2.81	1.87	0.51	87.61	3.00	1.97	0.46	32.19
ta_1	574	541.1	5.51	4.63	0.41	38.11	6.39	4.88	0.73	10.86

increasing number of goals. The reason for this is in the number of involved neurons. While the algorithm [4] derives the number from the number of goals, the proposed procedure dynamically adapts the number of neurons using shortest path in $\mathcal{W}$. Thus, for very large problems in the same map, additional neurons do not provide any benefit, and only increase the computational burden. The worse average results for the map *pb*, d=3 and d=2 are caused by the used point–point shortest path approximation, which provides unnecessary long paths in several cases. Nevertheless, the proposed procedure is able to find significantly better solutions, regarding the PDB, than the WRP algorithm [4].

The results for the TSP are presented in Table 3. The proposed procedure provides competitive results to the algorithm [5]. Worse average solutions are found for several problems. In these cases, the point–point path approximation provides longer paths than the point–goal path used in the TSP algorithm. The used schema of parameters evolution [11] leads to faster convergence, which "compensates" the more complex winner selection. However, the schema is the main reason for the worse performance of the proposed procedure than the TSP algorithm [5] with parameters' evolution [10].

4 Conclusion

Novel winner selection procedure for self-organizing maps has been proposed in this paper. The proposed adaptation procedure is able to deal with variants of

the multi-goal path planning problem including the TSP, the WRP and the safari route problem. Moreover, the procedure can be considered as parameterless, as the number of neurons is determined during the adaptation process. It provides a unified approach to solve various routing problems in the polygonal domain $\mathcal{W}$.

Although the proposed algorithm provides outstanding results in many cases, both the required computational time and the solution quality may be improved as the former algorithms for the WRP and the TSP provide better results in particular problems. Both these aspects are related to the evolution of the adaptation parameters, e.g., σ, μ, or size of the winner node neighborhood. Besides, the utilized approximation may be improved. Shortest path approximation and investigation of adaptation schemata with different evolution of parameters are subjects of the further work.

Acknowledgments. The work presented in this paper has been supported by the Ministry of Education of the Czech Republic under the program "National research program II" by Project No. 2C06005.

References

1. de Berg, M., Gudmundsson, J., Katz, M.J., Levcopoulos, C., Overmars, M.H., van der Stappen, A.F.: Tsp with neighborhoods of varying size. Journal of Algorithms 57(1), 22–36 (2005)
2. Chin, W.-P., Ntafos, S.: The zookeeper route problem. Information Sciences: an International Journal 63(3), 245–259 (1992)
3. Dumitrescu, A., Mitchell, J.S.B.: Approximation algorithms for tsp with neighborhoods in the plane. Journal of Algorithms 48(1), 135–159 (2003)
4. Faigl, J.: Approximate Solution of the Multiple Watchman Routes Problem with Restricted Visibility Range. IEEE Transactions on Neural Networks 21(10), 1668–1679 (2010)
5. Faigl, J., Kulich, M., Vonásek, V., Přeučil, L.: An Application of Self-Organizing Map in the non-Euclidean Traveling Salesman Problem. Neurocomputing 74, 671–679 (2011)
6. Goodman, J.E., O'Rourke, J. (eds.): Handbook of Discrete and Computational Geometry. CRC Press LLC, Boca Raton (2004)
7. Kulich, M., Faigl, J., Přeučil, L.: Cooperative planning for heterogeneous teams in rescue operations. In: IEEE International Workshop on Safety, Security and Rescue Robotics (2005)
8. Ntafos, S.: Watchman routes under limited visibility. Computational Geometry: Theory and Applications 1(3), 149–170 (1992)
9. Safra, S., Schwartz, O.: On the complexity of approximating tsp with neighborhoods and related problems. Computational Complexity 14(4), 281–307 (2006)
10. Somhom, S., Modares, A., Enkawa, T.: A self-organising model for the travelling salesman problem. Journal of the Operational Research Society, 919–928 (1997)
11. Zhang, W., Bai, Y., Hu, H.P.: The incorporation of an efficient initialization method and parameter adaptation using self-organizing maps to solve the TSP. Applied Mathematics and Computation 172(1), 603–623 (2006)

ARTICLE IN PRESS

Robotics and Autonomous Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Visiting convex regions in a polygonal map

Jan Faigl *, Vojtěch Vonásek, Libor Přeučil

Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27, Prague 6, Czech Republic

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Inspection planning
Multi-goal path planning
Self-organizing map
Safari route problem
Traveling salesman problem
Watchman route problem
Touring polygons problem
Polygonal domain

ABSTRACT

This paper is concerned with a variant of the multi-goal path planning in which goals are represented as convex polygons. The problem is to find a closed shortest path in a polygonal map such that all goals are visited. The proposed solution is based on a self-organizing map (SOM) algorithm for the traveling salesman problem. Neurons' weights are considered as nodes inside the polygonal domain and connected nodes represent a path that evolves according to the proposed adaptation rules. In addition, a reference algorithm based on the solution of the traveling salesman problem and the consecutive touring polygons problem is provided to find high quality solutions of the created set of problems. The problems are designed to represent various inspection and patrolling tasks and can form a kind of benchmark set for multi-goal path planning algorithms. The performance of the algorithms is examined in this problem set, which includes an instance of the watchman route problem with restricted visibility range. The proposed SOM based algorithms provide a unified approach to solve various visibility based routing problems in polygonal maps while they provide a competitive quality of solutions to the reference algorithm with significantly lower computational requirements.

1. Introduction

A problem to find a path visiting a set of given goals by a robot is called the *multi-goal path planning problem* (MTP). In particular, the MTP stands for finding a shortest path connecting a given set of goals located in a robot working environment. Approaches for the MTP are motivated by practical problems that include planning for a robotic arm [1,2], where the found path leads to minimization of the execution time providing a better utilization of the tools, or, in the case of a mobile robot, inspection planning [3], e.g., motivated by a search and rescue mission [4], where the time to find possible victims is critical.

A robot working environment can be represented by the polygonal domain $\mathcal{W}$ and goals may be represented by points. In such a case, the MTP can be formulated as the *traveling salesman problem* (TSP) [5]. Thus, the MTP becomes a combinatorial optimization problem to find a sequence of goals' visits, e.g., using all shortest paths between goals found in a visibility graph by Dijkstra's algorithm.

A more general variant of the MTP can be more appropriate if objects of interest may be located in certain regions of $\mathcal{W}$, e.g., when it is sufficient to reach a particular part of the environment to "see" or measure the requested object. A practical example of such a problem is collecting samples from particular areas, e.g., taking snapshots of objects or measuring concentration levels of substances' in regions or ponds, which are accessible from various directions.

In such a problem formulation, a goal is a polygonal region rather than a single point. Several algorithms addressing this problem can be found in the literature; however, only for its particular restricted variant. For example goals form a disjoint set of convex polygons attached to a simple polygon in the *safari route problem* [6], which can be solved in $O(n^3)$ [7]. If the route entry to the convex goal is not allowed, the problem is called the *zoo-keeper problem*, which can be solved in $O(n \log n)$ for a given starting point and the full shortest path map [8]. However, both problems are NP-hard in general.

A combinatorial approach [2] can be used for the MTP with partitioned goals, where each goal is represented by a finite (small) set of point goals. However, combinatorial approaches are unsuitable for continuous sets because of too many possibilities how to connect the goals.

In this paper, we present a self-organizing map (SOM) based algorithm for the general variant of the MTP with polygonal goals. The algorithm is based on SOM for the TSP in $\mathcal{W}$ [9]. Contrary to combinatorial approaches or other soft-computing techniques [10], a geometrical interpretation of SOM evolution in $\mathcal{W}$ allows easy and straightforward extensions to deal with polygonal goals. To show the flexibility of the SOM approach, several modifications of the adaptation rules are proposed and evaluated in a set of problems, which also demonstrate a geometric relation between the learning network and polygonal goals.

* Corresponding author. Tel.: +420 224 357 384; fax: +420 224 357 224.
E-mail address: faiglj@fel.cvut.cz (J. Faigl).

0921-8890/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2012.08.013

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013

The main advantage of the approach proposed is the ability to address general multi-goal path planning problems in $\mathcal{W}$ (not only in a simple polygon) and with goals not necessarily attached to $\mathcal{W}$; thus, the approach provides a unifying framework to solve various MTP variants.

Beside the SOM based algorithms, we present an alternative approach to address the MTP with polygonal goals that provides a reference solution of the problems solved. It is based on the solution of the TSP with point goals and a consecutive solution of the *touring polygons problem* (TPP). Although a polynomial algorithm for the TPP in a simple polygon has been proposed in [11], our reference algorithm is able to solve problems in $\mathcal{W}$ (not only in a simple polygon) and it also does not require disjoint convex goals. In addition, it is also probably easier to implement; thus, it represents a suitable reference algorithm for a comparison.

The rest of this paper is organized as follows. The next section provides an overview of the related work and similar problem formulations. Besides, it also contains a brief description of the SOM adaptation schema for the TSP in $\mathcal{W}$, because the proposed algorithms for the MTP with polygonal goals are its extensions. The addressed problem formulation and evaluation methodology of the solution quality and algorithms' comparison is presented in Section 3. A reference algorithm based on the solution of the related TSP and the consecutive TPP is presented in Section 4. In Section 5, the proposed modifications of the SOM's adaptation rules to deal with the polygonal goals are presented. The results and comparisons of the proposed algorithms and discussion of the results achieved are presented in Section 6. Concluding remarks are presented in Section 7.

2. Related work

In this section, we present an overview of approaches to address the visibility based routing problems, i.e., various formulations of the MTP. Mainly because once a point robot is assumed, $\mathcal{W}$ directly represents the robot configuration space and many visibility based approaches can be applied to solve such a variant of the MTP. As the proposed approach is based on the SOM algorithm for the TSP, its brief description is presented in Section 2.1.

The multi-goal path planning can be considered as a type of path planning under visibility constraints [12]. The goals in the MTP can represent sensing locations, where a robot takes measurements. Such locations can be found by a sensor placement algorithm that aims to find a minimal set of locations from which the whole $\mathcal{W}$ is covered ("*seen*") by a robot sensing device.

The sensor placement problem is related to the *art gallery problem* (AGP), which is a classical problem studied in computational geometry. The AGP was posed by Klee in 1973 and its most basic form is [13]: "*What is the smallest number of guards needed to guard an art gallery*?". The guards are static in the AGP and several problem variants for segment guards representing patrolling guards have been proposed [14,15]. Besides, a patrolling route for a single robot can be found as a solution of the *traveling salesman problem* (TSP) where guards' locations become cities that have to be visited [5]; hence, the problem becomes the MTP. Even though optimal algorithms for the AGP have been proposed for a restricted class of polygons [16,17], the AGP is known to be NP-hard for a polygon with holes [18], and therefore approximate algorithms are prefered to find guards [19,20]. Moreover, additional visibility constraints can be considered, which is the reason why the authors of [12] call the problem the *sensor placement problem* rather than the AGP. The constraints can restrict the visibility to a distance d, or an incident angle that regards a situation where a guard prefers to watch a scene directly rather than under an unsuitable angle [21]. Several approximate algorithms have been proposed to address the sensor placement problem, e.g., based on deterministic convex partitioning [22] or randomized approaches [23,3]. Once a set of guards covering the environment is determined, the problem to be solved is (again) the TSP.

The aforementioned approaches (consisting of finding the guards and the consecutive solution of the TSP) represent the so-called decoupled approach of the inspection planning to cover the whole $\mathcal{W}$. In the decoupled approach, the sensing of the environment is performed at discrete places, i.e., at the guards' positions, and therefore, the problem of guarding/searching $\mathcal{W}$ by one robot leads to minimize the visiting period of the sensing places. Regarding the cost of sensing and the cost of motion the number of places is minimized in the AGP part while the length of the path is minimized in the TSP. Due to independent solutions of the AGP and TSP, the decoupled approach is suitable for cases where the sensing cost is dominant over the motion cost [12].

A continuous sensing can be assumed if the motion cost is dominant and the sensing cost is relatively cheap. For such a case the problem can be formulated as the *watchman route problem* (WRP) that is a problem to find a closed shortest path such that all points of $\mathcal{W}$ are visible from at least one point of the path [24]. The WRP is NP-hard for the polygonal domain and similarly to the AGP, polynomial algorithms have been proposed for a restricted class of polygons [25]. The main difficulty of the problem is that the sensing locations are not explicitly prescribed, therefore approaches based on the TSP cannot be directly used as they will lead to the decoupled approach. Here, it is worth mentioning that a problem of finding a minimal set of guards lying on the shortest watchman route is called the *vision points problem* and is NP-hard [26].

A multi-robot variant of the WRP is the *m-watchman routes problem* (MWRP) that aims to find a route for each of m watchmen such that each point of the polygon $\mathcal{W}$ is visible from at least one route. For $m = 1$ the problem is the WRP and if m is so large that the total length of the routes is zero, the problem is the stationary AGP. Nilsson proved that the MWRP is NP-hard even in simple polygons [27].

Probably the first heuristic approach for the MWRP in a polygon with holes has been proposed by Packer in [28]. The approach is based on a set of static guards S found by the heuristic A_1 of [20] and constructing the minimum spanning tree of S. Distances between two guards are found as the length of the shortest path from the visibility graph. The tree is split into m sub-trees (for m watchmen) and Hamiltonian routes on each sub-tree are independently constructed. Vertices along a route are substituted by others that shorten the length of the route and maintain the full coverage. Finally, redundant vertices of the route are removed. Although this approach is based on a solution of the AGP, only unrestricted visibility range has been considered by the author.

If a visibility range is restricted to a distance d, two variants of the WRP can be found in the literature [7]. The *d-watchman route problem* is a variant to see only the boundary of the polygon, while the *d-sweeper route problem* aims to sweep a polygonal floor using a circular broom of radius d, so that the total travel of the broom is minimized [6]. An approximate algorithm for the MWRP with the d-visibility has been presented in [29].

The aforementioned *safari route* [6] and *zoo-keeper route* [30] problems introduced in Section 1 are also motivated by the WRP with restricted visibility range. These problems are variants of the MTP with polygonal goals, as in both of them the problem is to find a route inside a polygon P that visits a given collection of sub-polygons of P. Also in both original problem formulations the sub-polygons are convex and are entirely inside the polygon P. Although these problems are very close to the problem addressed in this paper, the main difference is that their original formulations are only for a simple polygon, for which the polynomial algorithms have been proposed, but the problems are NP-hard for the polygonal domain.

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013

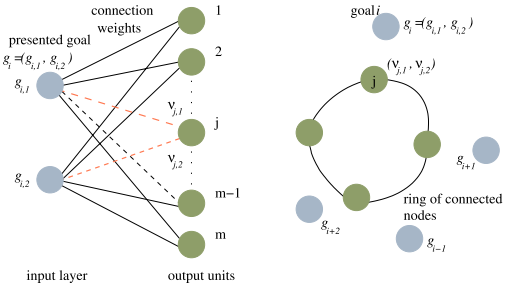


Fig. 1. A schema of the two-layered neural network and the associated geometric representation.

Routing problems with polygonal goals can be considered as variants of the TSP with neighborhoods (TSPN) [31]. The TSPN is studied for graphs or as a geometric variant in a plane but typically without obstacles. Approximate algorithms for restricted variants of the TSPN have been proposed, e.g., the TSPN with arbitrary connected neighborhoods with comparable diameters and for disjoint unit disk neighborhoods [32], or disjoint convex fat neighborhoods of arbitrary size [33]. However, the TSPN is APX-hard and cannot be approximated to within a factor $2-\epsilon$, where $\epsilon > 0$, unless $P = \text{NP}$ [34].

Having a sequence of polygonal goals $(P_1, P_2, \ldots, P_k)$, one can ask for a shortest path visiting in order at least one point of each polygon in the sequence. This problem is called the *touring polygons problem* [31], and it is a strict generalization of the safari, zoo-keeper, and watchman route problems in a simple polygon [11]. In a case of convex polygons in a plane, and given start and target points, an $O(kn\log(n/k))$ algorithm for disjoint polygons has been proposed by the authors of [11], where n is the number of vertices specifying the polygons. Besides, the authors also proposed an $O(nk^2\log n)$ algorithm for arbitrarily intersecting polygons lying in a simple polygon. If polygons are non-convex, the TPP is NP-hard [11].

In [35], an approximate algorithm for the TPP in a plane is proposed. The algorithm is based on an iterative procedure refining the path until the selected accuracy ϵ is achieved. In each iteration, a new point at a polygon p_i is eventually computed to shorten the path connecting three consecutive polygons p_{i-1}, p_i, and p_{i+1}. Once the length of the new path is shorter than the previous path's length (about less than ϵ), the refinement is terminated. A proof that the algorithm finds a global solution of the TPP is based on an approximate algorithm for solving the Euclidean shortest path problem in a three dimensional polyhedral space presented in [36].

2.1. SOM for routing problems in $\mathcal{W}$

A SOM algorithm for routing problems, in particular the SOM for the TSP in $\mathcal{W}$ [9], is Kohonen's type of unsupervised two-layered learning neural network. The network contains a two dimensional input vector and an array of output units that are organized into a uni-dimensional structure. An input vector represents coordinates of a point goal, and connections' weights (between the input and output units) represent coordinates of the output units. Connections' weights can be considered as nodes representing a path, which provides direct geometric interpretation of the neurons' weights. So, the nodes form a ring in $\mathcal{W}$ because of the uni-dimensional structure of the output layer, see Fig. 1.

The network learning process is an iterative stochastic procedure in which goals are presented to the network in a random order. The procedure basically consists of two phases: (1) selection of the winner node to the presented goal; and (2) adaptation of the winner and its neighboring nodes toward the goal. The learning procedure works as follows.

1. *Initialization*. For a set of n goals $\boldsymbol{G}$ and a polygonal map $\mathcal{W}$, create $2n$ nodes $\mathcal{N}$ around the first goal. Let the initial value of the learning gain be $\sigma = 12.41n + 0.06$, and adaptation parameters be $\mu = 0.6$, $\alpha = 0.1$.
2. *Randomizing*. Create a random permutation of goals $\Pi(\boldsymbol{G})$.
3. *Clear inhibition*. $\boldsymbol{I} \leftarrow \emptyset$.
4. *Winner selection*. Select the closest node $\nu^\star$ to the goal $g \in \Pi(\boldsymbol{G})$ according to:

   $$\nu^\star \leftarrow \operatorname*{argmin}_{\nu\in\mathcal{N}, \nu\notin\boldsymbol{I}} |S(\nu, g)|,$$

   where $|S(\nu, g)|$ is the length of the shortest path among obstacles $S(\nu, g)$ from ν to g.
5. *Adapt*. Move $\nu^\star$ and its neighboring nodes along a particular path toward g:
   - Let the current number of nodes be m, and N ($N \subseteq \mathcal{N}$) be a set of $\nu^\star$'s neighborhoods in the cardinal distance less than or equal to 0.2 m.
   - Move $\nu^\star$ along the shortest path $S(\nu^\star, g)$ toward g by the distance $|S(\nu^\star, g)|\mu$.
   - Move nodes $\nu \in N$ toward g along the path $S(\nu, g)$ by the distance $|S(\nu, g)|\mu f(\sigma, l)$, where f is the neighboring function $f = \exp(-l^2/\sigma^2)$ and l is the cardinal distance of ν to $\nu^\star$.
   - Update the permutation: $\Pi(\boldsymbol{G}) \leftarrow \Pi(\boldsymbol{G}) \setminus \{g\}$.
   - Inhibit the winner: $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \{\nu^\star\}$.

   If $|\Pi(\boldsymbol{G})| > 0$ go to Step 4.
6. *Decrease the learning gain*. $\sigma \leftarrow (1-\alpha)\sigma$.
7. *Termination condition*. If all goals have the winner in a sufficient distance, e.g., less than 10^{-3}, or $\sigma < 10^{-4}$ Stop the adaptation. Otherwise go to Step 2.
8. *Final path construction*. Use the last winners to determine a sequence of goals' visits.

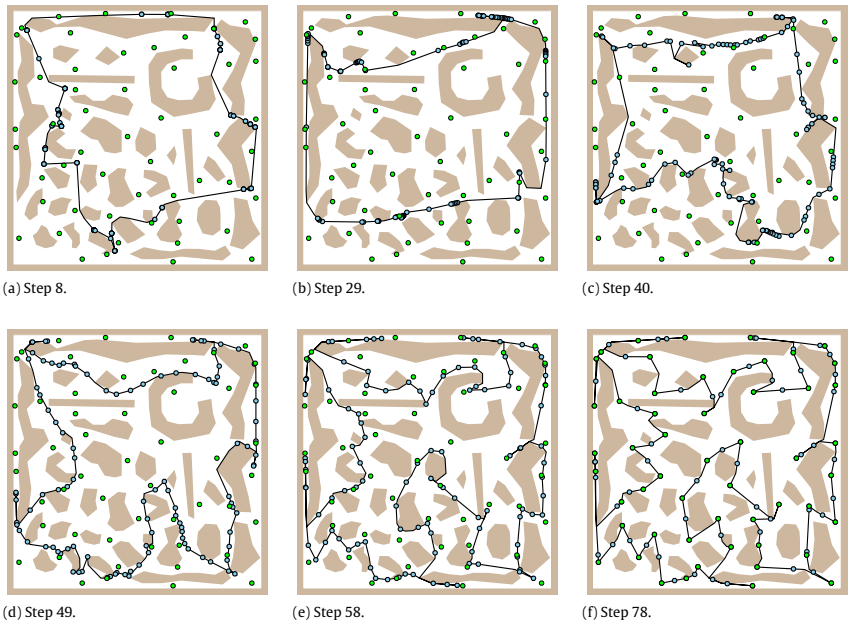
The algorithm is terminated after a finite number of adaptation steps as σ is decreased after presentation of all goals to the network. Moreover, the inhibition of the winners guarantees that each goal has associated a distinct winner; thus, a sequence of all goals' visits can be obtained by traversing the ring at the end of each adaptation step.

The computational burden of the adaptation procedure depends on determination of the shortest path in $\mathcal{W}$, because $2n^2$ node–goal distance queries (Step 4) and $(0.8n+1)n$ node–goal path queries (Step 5) have to be resolved in each adaptation step. Therefore, an approximate shortest path is considered using a supporting division of $\mathcal{W}$ into convex cells (convex partition of $\mathcal{W}$) and pre-computed all shortest paths between map vertices to the point goals. The approximate node–goal path is found as a path over vertices of the cells in which the points (node and goal) are located. Then, such a rough approximation is refined using a test of direct visibility from the node to the vertices of the path. Details and evaluation of refinement variants can be found in [9].

Beside the approximation, the computational burden can be decreased using the Euclidean pre-selection [37], because only the node with a shorter Euclidean distance to the goal than the distance (length of the approximate shortest path) of the current winner node candidate can become the winner.

In Fig. 2, a ring of nodes connected by an approximate shortest path between two points is shown to provide an overview of the ring evolution in $\mathcal{W}$.

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013



(a) Step 8. (b) Step 29. (c) Step 40.

(d) Step 49. (e) Step 58. (f) Step 78.

Fig. 2. An example of the ring evolution in a polygonal map for the MTP with point goals, small green disks represent goals and blue disks are nodes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3. Problem statement

The problem addressed in this paper can be formulated as follows. *Find a closed shortest path visiting a given set of goals represented as convex polygons (possibly overlapping each other) in a polygonal map* $\mathcal{W}$. The problem formulation is based on the safari route problem [6]; however, it is more general in three aspects. First, polygons can be placed inside a polygon with holes. Also, it is not required that convex polygons are attached to the boundary of $\mathcal{W}$ like in the original safari route problem formulation. Finally, polygons can overlap each other, and therefore, such polygons can represent a polygonal goal of an arbitrary shape.

The proposed problem formulation comprises the WRP with restricted visibility range d. The set of goals can be found as a convex cover set of $\mathcal{W}$, i.e., a set of convex polygons whose union is $\mathcal{W}$. The advantage of an algorithm solving the formulated problem is that it is not required to have a minimal cover set. The restricted convex polygons to the size d can be found by a simple algorithm based on a triangular mesh of $\mathcal{W}$ [29].

3.1. The quality of solution

The studied SOM based algorithms in this paper are randomized algorithms; therefore to examine the quality of found solutions 100 trials of each particular problem is considered. The solution quality can be then measured as the percent deviation from the reference path's length of the mean solution value (PDM)

$$\mathrm{PDM} = \frac{L - L_{\mathrm{ref}}}{L_{\mathrm{ref}}} \cdot 100\%, \tag{1}$$

and as the percent deviation from the reference of the best solution value (PDB)

$$\mathrm{PDB} = \frac{L_s - L_{\mathrm{ref}}}{L_{\mathrm{ref}}} \cdot 100\%, \tag{2}$$

where L_{ref} is the length of the reference path, and L and L_s are the average and the shortest path lengths from all the trials of the algorithm for the particular problem, respectively. The PDM and PDB provide an overview of the algorithm quality. The PDM can be interpreted as an expected solution quality and the PDB as what can be achieved by the algorithm.

It is expected that the reference value would always be lower than L and L_s; however, it may happen that it would not be the case, because the reference solution is also only an approximation as an optimal algorithm for the general MTP is not available. Therefore, negative values of the PDB and PDM indicate that the evaluated algorithm provides better solutions than the selected reference algorithm. An algorithm providing the reference solution is described in the next section.

3.2. Algorithm comparison

Due to randomized nature of the SOM based algorithm, it is desired to compare the performance and results of the proposed modifications of SOM for the TSP statistically. Therefore, for each modified variant of the SOM algorithm and each particular problem 100 trials are performed in order to obtain representative samples of the two evaluated distributions. The distributions are the required computational time of the SOM adaptation procedure, and the length of the path found. The algorithm comparison is based on statistical tests using a null hypothesis H_0, i.e., H_0 represents that the algorithms provide statistically identical results (regarding the required computational time, or the path length), and the alternative hypothesis is that the results are different.

The required computational time is evaluated using the *Wilcoxon* test of the null hypothesis, because the distributions of the time are not Gaussian. The algorithms are considered different, i.e., one is faster than the other, if the p-values obtained by the

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013

Wilcoxon test are less than 0.001. In such a case, the difference between the required computational times of the compared algorithms is statistically significant.

The comparison according to reference paths is evaluated in a different way. It is because the reference path is found using a deterministic algorithm, which always provides the same solution, while paths found by the SOM-based algorithm differ due to randomization. Therefore, the *one-sample Wilcoxon* test is used as it is suggested by the authors of [38]. Similarly to the above comparison, once the p-values of the test statistics are less than 0.001, the null hypothesis is rejected; thus, the paths provided by the algorithms differ.

It is expected the SOM algorithm would provide a bit worse solution (in terms of path length) than the reference algorithm, and therefore an interesting question is how much are the solutions found by SOM worse than the reference solution. The one-sample Wilcoxon test is used to find such a qualitative measure. It is performed as follows. The reference path's length is iteratively increased by a given percentage level p_L, and such a length is compared with the distribution of the paths found by the SOM algorithms. Once the null hypothesis is accepted, the iterative procedure is terminated, and the current value of p_L denotes the desired qualitative measure. In a case that a basic quality of solution indicator (e.g., PDM) is negative, the reference path's length is decreased in a similar manner, and the qualitative measure p_L indicates how much the SOM based solution is better than the reference one.

4. Reference algorithm

The multi-goal path planning problem is *de facto* the TSP once the paths between goals are known. Therefore, for a point goals, approximate algorithms guaranteeing the solution quality of the TSP can be used for a particular restricted problem variant of the MTP. However, the authors of [39] note that approximate factors characterize algorithms in the worst case, which are often several times worse than the optimal solution, and such loose bounds are not valuable in real-world situations.

Similarly, the main drawback of the approaches addressing the safari or zoo-keeper route problems is their focus only on the particular restricted problem variant. Also a more general MTP formulation as the TSPN does not really help due to the complexity of the general TSPN. For restricted variants of the TSPN, the situation is similar to the safari route problems, i.e., the approaches are considered only in a plane.

The aforementioned reasons lead us to propose a practical approach to find a reference solution of the MTP, which will provide a solution "good" enough for comparing it with the SOM based approaches, and which will also be easy to implement. The main idea of the reference algorithm proposed is based on a transformation of the MTP to the TPP using the optimal solution of the underlying TSP. The algorithm is as follows.

4.1. Transformation of the MTP with polygonal goals to the TPP

The main difference between the MTP and the TPP is that in the TPP, the sequence of goals visits is known; however, the difficulty is how to connect the consecutive goals, i.e., which point of each goal has to be visited in order to minimize the total distance traveled. Therefore, the TPP is obtained by solving the MTP as the TSP with point goals. Thus, for each polygonal goal a single point representative is determined.

Convex goals are assumed in the problem addressed, therefore a centroid of each polygonal goal can be used as a point goal in the TSP. More formally, let $\boldsymbol{G} = \{g_1, g_2, \ldots, g_n\}$ be a set of n convex goals in the polygonal map $\mathcal{W}$ with v vertices, and $c(g)$ denotes the centroid of the goal g. Then, all shortest paths between the centroids are found using Dijkstra's algorithm and the complete visibility graph that is found in $O((n+v)^2)$ [40]. Such an instance of the TSP with n point goals is solved exactly by the `concorde` solver [41]. The found solution of the TSP is then used to retrieve a sequence of the goals' visits for the consecutive TPP.

4.2. Approximate solution of the TPP

Even though approaches for optimal [11] or approximate solution of the TPP [35] have been proposed, their main drawback is that they consider goals only in a plane or in a simple polygon. Therefore, a simple approximate algorithm to deal with goals in the polygonal domain is proposed here. The algorithm is inspired by the iterative procedure proposed in [35] while the obstacles are addressed by sampling the boundary of each polygonal goal into a finite set of points. For simplicity, the sequence of goals' visits obtained from the solution of the TSP is $(g_1, g_2, \ldots, g_n)$ in the rest of this section.

Having a given sampling distance ρ and each polygonal goal g represented by a set of straight line segments $\boldsymbol{S}_g = \{s_1, s_2, \ldots, s_k\}$, the sampling is performed as follows. First, for each goal only segments entirely lying inside $\mathcal{W}$ are considered, as the path never goes through an obstacle. Then, each such segment s is sampled using its end points. If the length of the segment $|s|$ is less than 2ρ then a middle point is an additional representative point of s, otherwise additional points are sampled equidistantly using ρ. At the end of the sampling, each goal g_i has associated a set of the representative points $\boldsymbol{P}_i$.

The reference solution of the MTP with polygonal goals is found as a path over the goals using the sequence of representative points. The path is found by the following refinement procedure.

1. *Initialization*. Construct an initial touring polygons path using the first representative points of each polygonal goals. Let the path be defined by $Path = (p_1, p_2, \ldots, p_n)$, where $p_i \in \boldsymbol{P}_i$ is the selected representative point of the ith goal, and let $L = |Path|$ be the length of the shortest path induced by $Path$, e.g., found using the visibility graph of the points $\boldsymbol{P}_i$ in $\mathcal{W}$.
2. *Refinement*.
   - For $i = 1, 2, \ldots, n$
     - Find $p_i^* \in \boldsymbol{P}_i$ minimizing the length of the path $d(p_{i-1}, p_i^*) + d(p_i^*, p_{i+1})$, where $d(p_k, p_l)$ is the length of the shortest path (among obstacles) from p_k to p_l, $p_0 = p_n$, and $p_{n+1} = p_1$.
     - If the total length of the current path over point p_i^* is shorter than over p_i, replace the point p_i by p_i^*.
   - Compute new path length L_{new} using eventually refined representative points.
3. *Termination condition*. If $L_{\text{new}} - L < \epsilon$ Stop the refinement. Otherwise $L \leftarrow L_{\text{new}}$ and go to Step 2.
4. *Final path construction*. Use the last sequence of the representative points of the goals and construct the path using the shortest paths among obstacles between two consecutive points.

The refinement procedure is repeated until the change of the path length is not significant (smaller than the value ϵ). The value of ϵ can be set arbitrarily, but it is clear a smaller value improves the solution quality. Similarly a smaller value of the sampling distance ρ can provide a better path; however, it also increases the number of representatives, thus increases the computational burden. The algorithm has been used to find a reference solution of the problem set presented in Section 6.1. During the computation, $\epsilon = 0$ has been used and it has also been observed that the quality of found solutions is almost independent of the value of ρ, e.g., for less than 0.1 m (except the jh_{10}-*coverage* problem). It is caused mainly because the convex goals are formed from segments that typically contains several segments with length about 0.1 m, because the goals are created on top of a triangular mesh of the polygonal maps.

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013

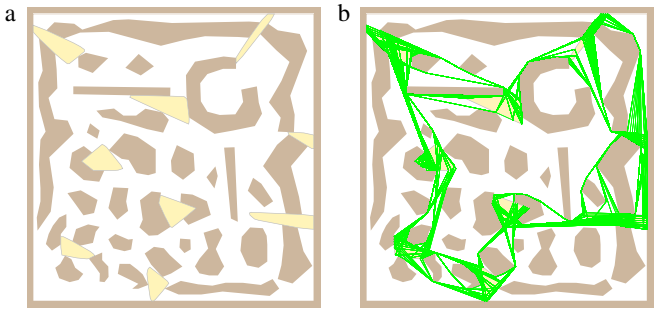


Fig. 3. An example of the shortest between goals, (a) problem $dense_5$-A, and (b) shortest paths between consecutive goals.

4.3. Comments

An eventual issue of the proposed reference algorithm could be the high computational requirements for a high number of the representative points. This is mainly related to the computation of the visibility graph and determination of the shortest paths among obstacles. Moreover, if the distance matrix (or paths) are pre-computed and stored in memory, the algorithm can be computationally infeasible due to memory requirements. In particular, this is the case of the jh_{10}-*coverage* problem, which represents an instance of the WRP.

These issues can be resolved using the approximate shortest path between two points in $\mathcal{W}$ [29], which is principally similar to the node–goal path described in Section 2.1. For the problems examined in this paper the approximation provides the same paths, while it is up to two orders of magnitude faster than the pre-computation of the required shortest path and construction of the complete visibility graph using the approach [40], because of the saved initialization phase.

Here, it should be noted that the number of required paths is relatively small, as only paths between two consecutive goals need to be determined. In Fig. 3, polygonal goals of the $dense_5$-A problem and particular paths between each consecutive goal are presented.

The optimal solution of the TSP can be computationally demanding, therefore a heuristic algorithm like the Chained Lin–Kernighan approach [42] can be a more practical approach. However, the optimal solution together with the proposed solution of the TPP provide a strictly deterministic approach, which does not require statistical evaluation of the reference solutions; thus, it simplifies the algorithms' comparison a bit.

Convexity of the goals and randomization

Although the convexity of the goals is used for determining the representative points for the TSP as the centroids of the polygonal goals, the convexity is not mandatory. Alternatively any point can be used as a representative of the polygonal goal, because the sequence of polygons' visits is retrieved from the TSP; thus, each point is associated to the selected polygon.

The reference algorithm proposed is strictly deterministic; however, it can be straightforwardly randomized. First, the initial path can be created from a randomly selected sampled point of each polygon. Besides, each loop of the refinement can be started from a random goal. Such a randomization has been extensively evaluated and its significant benefit has not been observed as it mainly affects the number of required refinements to find the same final path. It is because the refinement itself is very fast, while the initialization phase using the visibility graph and pre-computed shortest path is computationally demanding.

5. Adaptation rules for polygonal goals

Although it is obvious that a polygonal goal can be sampled into a finite set of points and the problem can be solved as the MTP with partitioned goals, the aforementioned SOM procedure can be straightforwardly extended to sample the goals during the self-adaptation. Thus, instead of explicit sampling of the goals three simple strategies of how to deal with adaptation toward polygonal goals are presented in this section. The proposed algorithms are based on SOM for the TSP using centroids of the polygonal goals as point goals, see Section 2.1. However, the *select winner* and *adapt* phases are modified to find a more appropriate point of the polygonal goal and to avoid unnecessary movement into the goal. Therefore, a new point representing a polygonal goal is determined during the adaptation and used as a point goal, which leads to computation of a shortest path between two arbitrary points in $\mathcal{W}$. Similarly to the node–goal queries an approximate node–point path is considered to decrease the computational burden. The approximation is also based on a convex partition of $\mathcal{W}$ and the shortest path over cells' vertices (a detailed description can be found in [29]).

5.1. Interior of the goal

Probably the simplest approach (called *goal interior* here) can be based on the regular adaptation to the centroids of the polygonal goals. However, the adaptation, i.e., the node movement toward the centroid, is performed only if the node is not inside the polygonal goal. Determination if a node is inside the polygonal goal with n vertices can be done in $O(n)$ computing the winding number or in $O(\log n)$ in the case of a convex goal. So, in this strategy, the centroids are more like attraction points toward which nodes are attracted because the adaptation process is terminated if all winner nodes are inside the particular polygonal goals. Then, the final path is constructed from a sequence of winner nodes using the approximate shortest node–node path. An example of solutions using the new termination condition and with the avoiding adaptation of winners inside goals is shown in Fig. 4.

This adaptation strategy clearly demonstrates one of the SOM's advantages that is its ability to reflect local properties of the environment during the adaptation, which, in this case, is the test if a node is inside the polygonal goal.

5.2. Attraction point

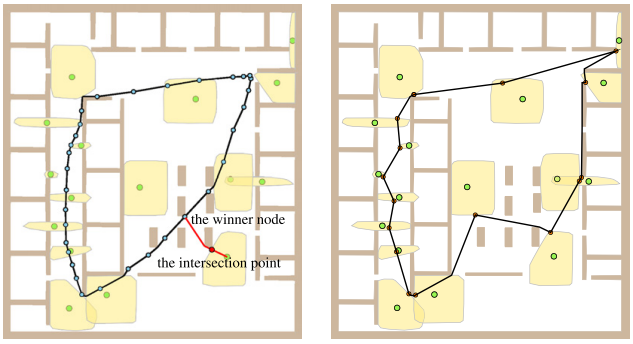
The strategy described above can be extended by determination of a new attraction point at the border of the particular polygonal goal toward which is being adapted. First, a winner node $\nu^\star$ is found regarding its distance to the centroid $c(g)$ of the goal g. Then, an

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013

(a) A path found using termination of the adaptation if all winners are inside goals, $L = 78.4$ m.

(b) A path found with avoiding adaptation of winners inside goals, $L = 65.0$ m.

Fig. 4. Examples of found paths without and with consideration of winners inside the goals. Goals are represented by yellow regions with small green disks representing the centroids of the regions. Winner nodes are represented by small orange disks. The length of the found path is denoted as L. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) An intersection point.

(b) A path found, $L = 59.5$ m.

Fig. 5. Examples of an intersection point and a path found using the *attraction* algorithm variant, blue disks are nodes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

intersection point p of g with the path $S(\nu, c(g))$ is determined. The point p is used as a point goal to adapt the winner and its neighboring nodes. This modification is denoted as *attraction* in the rest of this paper.

An example of determined intersection point p and the final path found is shown in Fig. 5. The path is about five meters shorter than a path found by avoiding adaptation of winner nodes inside the goals. Determination of the intersection point increases the computational burden, and therefore an experimental evaluation of the proposed algorithm variants is presented in Section 6.

5.3. Selection of alternate goal point

A polygonal goal can be visited using any point of its border. The closest point at the goal border to a node can be determined in the winner selection phase. To find such a point, straight line segments forming the goal are considered instead of the goal centroid. Moreover, a goal can be attached to the map, and therefore only segments lying inside the free space of $\mathcal{W}$ are used. Let $\boldsymbol{S}_g = \{s_1, s_2, \ldots, s_k\}$ be the border segments of the polygonal goal g that are entirely inside $\mathcal{W}$. Then, the winner node ν^* is selected from a set of non-inhibited nodes regarding the shortest path $S(\nu, s)$ from a point ν to the segment s, $s \in \boldsymbol{S}_g$. Beside the winner node, a point p at the border of g is found in the winner selection procedure as a result of determination of $S(\nu, s)$. The border point p is then used as an alternate point goal for the adaptation, therefore this modification is denoted as *alternate goal*.

Determination of the exact shortest point–segment path can be too computationally demanding, therefore the following approximation is considered. First, the Euclidean distance between the node ν and the segment s is determined. If the distance is smaller than the distance of the current winner node candidate, then the resulting point p of s is used to determine an approximate path among obstacles between p and ν. If $|S(p, \nu)|$ is shorter than the path length of the current winner node candidate to its border point, ν becomes the new winner candidate and p is the current alternate goal (border) point.

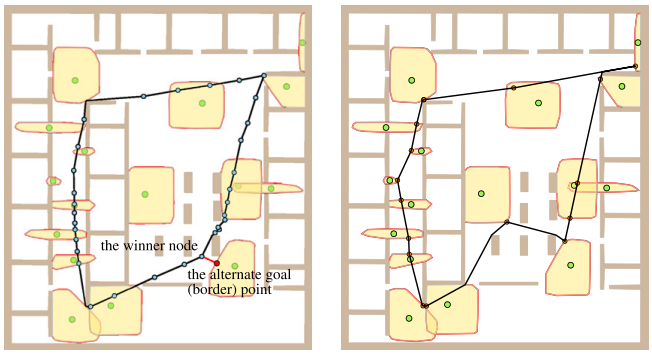
Even though this modification is similar to the modification described in Section 5.2, it provides sampling of the goal boundary with a shorter distance of the goal point to the winner node; thus, a shorter final path can be found. An example of found alternate goal point and the path found is shown in Fig. 6.

6. Results

6.1. Problems description

The proposed adaptation rules in Section 5 have been experimentally verified in a set of problems. Due to a lack of

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013



(a) An alternate goal point. (b) A path found, $L = 57.4$ m.

Fig. 6. An example of the alternate goal point and the final path found. Red straight line segments around the goal regions denote parts of the goal border inside the free space of $\mathcal{W}$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Properties of environments and their polygonal representation.

Map	Dimensions (m × m)	No. vertices	No. holes
jh	20.6 × 23.2	196	9
pb	133.3 × 104.8	89	3
h2	84.9 × 49.7	1061	34
dense	21.0 × 21.5	288	32
potholes	20.0 × 20.0	153	23

commonly available multi-goal path planning problems with polygonal goals several problems have been created within a map of real and artificial environments.[1] An overview of the basic properties of the environments is shown in Table 1. Maps *jh*, *pb*, and *h2* represent real environments (building plans), and maps *dense* and *potholes* are artificial environments with many obstacles.

Sets of polygonal obstacles have been placed within the maps in order to create representative multi-goal path planning problems. The name of the problem is derived from the name of the map, considered visibility range d in meters written as a subscript, and particular problem variant, e.g., the problem name is in the form map_d-*variant*. The value of d restricts the size of the convex polygonal goal, i.e., all vertices of each goal are closer than d. An unrestricted visibility range is considered in problems without the subscript. The convex polygonal goals are found on top of the triangular mesh, details about the procedure used can be found in [29].

The problems have been designed in order to create representative problems particularly focused on specific characteristics. Here, a short description of the motivation behind their design is provided to present the main aim of the problems. The problems are visualized in Fig. 7, where the centroids of the convex goals and reference paths found are showed as well. The map *jh* represents an office-like environment with many rooms. Therefore, several problems within this map are created with a motivation of patrolling or inspection tasks. The jh_4-*A* problem variant is a general problem with goals in a few rooms and partially trespassing to corridors. In the jh_5-*corridors* and *jh-rooms* problems, the expected path should not enter rooms. Thus, the aim of these problems is to demonstrate the ability of the evaluated algorithms to take an advantage of the polygonal goals, as the centroids are located relatively far from the border, and visitations of the centroids unnecessary increase the path.

[1] All problems and supporting materials are available at http://purl.org/faigl/safari

The jh_{10}-*coverage* problem represents an instance of the WRP with restricted visibility range, and therefore the algorithm's performance in this problem can indicate a flexibility of the approach tested.

The problem $h2_5$-*A* is within a large map, and it is included in the problem set mainly because of the map's complexity; thus, it serves as a load and study of the algorithm's performance in maps with many vertices, e.g., to study the influence of the supporting algorithms like the approximate shortest path. The pb_5-*A* problem is a very simple problem; however, a solution can stuck in a local optima, due to the goal in the middle of the map. The *dense* map is a complicated environment, and therefore several alternative paths connecting the goals exist, e.g., see Fig. 3. In addition, the *dense-small* problem contains several goals that are inside another goal. These goals can show the ability to avoid focus of the algorithm on the larger goals, as the visit of the inside goals is mandatory. Finally, the problem $potholes_2$-*A* contains many small obstacles, which are relatively sparse. The "right" sequence of goals visit is relatively easy to find; however, the final path length depends on a proper selection of the points at the border of the polygonal goals.

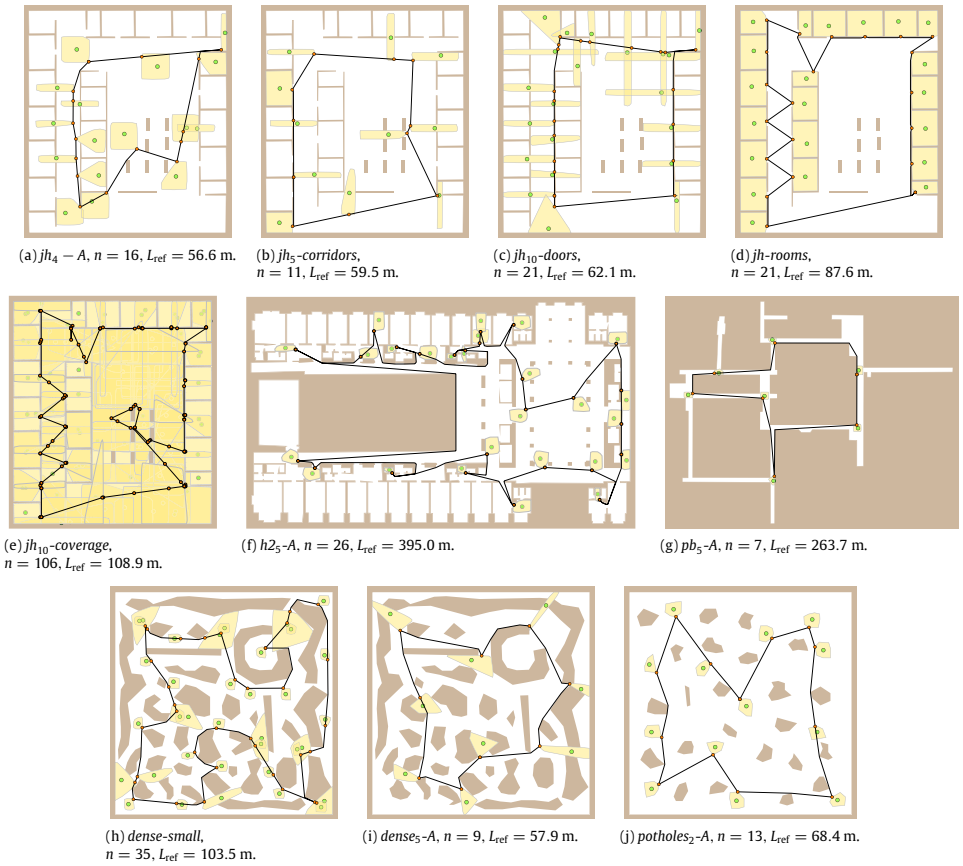
6.2. Results

Each problem of the aforementioned problem set has been solved using the reference algorithm described in Section 4 and three SOM based algorithms proposed in Section 5. Due to randomization of the SOM based algorithms, 100 trials have been performed for each algorithm and problem; thus, the total number of found solutions by the SOM's approaches is 3000 for all problems of the problem set.

All the results have been obtained using the same computational environment consisting of a C++ implementation of the algorithms compiled by G++ version 4.6 with -O2 optimization, a single core of the i7-970 CPU at 3.2 GHz, 12 GB RAM, and 64-bit version of FreeBSD 8.2. Therefore, all the required computational times presented can be directly compared.

The basic quality indicators (described in Section 3.1) are presented in Table 2. Regarding the PDM values presented, it is clear that a more sophisticated adaptation rule provides better results. However, it seems it is not the case of the computational complexity indicated by the column T, where the average of the required computational time of the adaptation is presented. A statistical evaluation of the results is presented in Table 3, where two algorithms are compared using the null hypothesis approach described in Section 3.2. Once the null hypothesis is rejected

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013



(a) $jh_4 - A$, $n = 16$, $L_{\text{ref}} = 56.6$ m. (b) jh_5-*corridors*, $n = 11$, $L_{\text{ref}} = 59.5$ m. (c) jh_{10}-*doors*, $n = 21$, $L_{\text{ref}} = 62.1$ m. (d) *jh-rooms*, $n = 21$, $L_{\text{ref}} = 87.6$ m.

(e) jh_{10}-*coverage*, $n = 106$, $L_{\text{ref}} = 108.9$ m. (f) $h2_5$-*A*, $n = 26$, $L_{\text{ref}} = 395.0$ m. (g) pb_5-*A*, $n = 7$, $L_{\text{ref}} = 263.7$ m.

(h) *dense-small*, $n = 35$, $L_{\text{ref}} = 103.5$ m. (i) $dense_5$-*A*, $n = 9$, $L_{\text{ref}} = 57.9$ m. (j) $potholes_2$-*A*, $n = 13$, $L_{\text{ref}} = 68.4$ m.

Fig. 7. The problems examined and the reference solutions found, n is the number of goals, L_{ref} is the length of the reference path, yellow regions are polygonal goals, and small green disks are the goals' centroids. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Results of the proposed SOM adaptation rules.

Problem	n	L_{ref} (m)	Goal interior			Attraction			Alternate goal		
			PDM	PDB	T_a (ms)	PDM	PDB	T_a (ms)	PDM	PDB	T_a (ms)
$dense_5$-A	9	57.9	17.30	10.87	12	7.11	3.62	16	1.81	0.24	23
dense-small	35	103.5	14.69	7.99	171	10.54	3.90	205	8.58	−0.13	274
$h2_5$-A	26	395.0	6.89	3.95	130	2.98	1.06	160	1.89	0.22	210
jh_{10}-coverage	106	108.9	22.91	15.40	872	−2.63	−7.62	1040	−13.75	−14.53	1578
jh_{10}-doors	21	62.1	14.86	8.51	35	8.79	5.53	38	0.38	−0.04	63
jh_4-A	16	56.6	16.95	11.34	24	7.43	3.18	28	0.69	0.18	45
jh_5-corridors	11	59.5	17.06	12.02	13	10.38	7.01	16	0.84	0.12	21
jh-rooms	21	87.6	17.75	13.82	53	0.78	0.18	68	0.61	0.21	73
pb_5-A	7	263.7	4.18	1.23	5	2.87	0.17	7	3.42	0.07	11
$potholes_2$-A	13	68.4	7.58	4.57	18	3.12	0.91	23	3.01	0.53	29

(the statistics are not the same), an additional null hypothesis is evaluated to determine if one algorithm is statistically better than the other one, i.e., using the average required computational time T_a and the average length of the path found L. The adaptation rules proposed are incremental, and therefore the *attraction* variant is compared against the *goal interior* variant, and similarly the *alternate goal* is compared against *attraction*. Because all the p-values are very small, characters '−', '+', and ' = ' are used to denote that an algorithm is slower, a solution is better, or the performance indicators are statistically identical.

The results show that the reference algorithm provides better solutions (except for the problem jh_{10}-*coverage*). Table 4 presents a deeper insight into the performance characteristics of the reference and SOM based algorithms. First, an estimation of the approximate factor of the SOM based algorithm is shown in the column p_L%, viz Section 3.2. For the *alternate goal* algorithm, this

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013

Table 3
Comparison of SOM based algorithms.

Problem	a_1: Attraction goal interior		a_2: Alternate goal attraction	
	Length	Time	Length	Time
dense$_5$-A	+	−	+	−
dense-small	+	−	+	=
h2$_5$-A	+	−	+	−
jh$_{10}$-coverage	+	−	+	−
jh$_{10}$-doors	+	−	+	−
jh$_4$-A	+	−	+	=
jh$_5$-corridors	+	−	+	−
jh-rooms	+	−	=	−
pb$_5$-A	+	−	=	−
potholes$_2$-A	+	−	=	=

+− the algorithm a_1 provides better paths than a_2.

factor is mostly less than one percent; however, the required computational time is significantly smaller (more than three orders of magnitude) than for the reference algorithm. Note the time T of the reference algorithm does not include the time needed to find the optimal solution of the TSP. The computational burden of the reference algorithm is caused by the computation of the visibility graph and all shortest paths between points of two consecutive goals. The number of the points used is denoted by n_p in the table, and it is significantly higher than the number of goals due to sampling of the goals' borders. The refinement itself is very fast as all required distances are pre-computed, therefore only the total time to solve the TPP is presented in the column T. On the other hand, the required computational time T of the SOM based algorithms consists of the time to initialize the supporting structures (for the approximate shortest path) T_{init}, which is shown as a number of percentage points of T in the column T_{init}%, and the adaptation time T_a. The initialization itself consists of construction of the convex partitioning and visibility graphs, and computation of all shortest paths between the map vertices (and centroids of the convex goals).[2] The required computational times of the constructions are presented in Table 5 and are negligible regarding the time to compute the shortest path, which is indicated by T_{init}. The partition is found by Seidel's algorithm [43] and the number of convex polygons utilized in the approximation of the shortest path is presented in the second column. The visibility graph is found using [40].

The computational requirements of the reference algorithm using the pre-computed shortest paths are very high. This is especially significant for the problem *jh$_{10}$-coverage*. Beside the required computational time, the required memory footprint to store the pre-computed paths and distances is about 6.5 GB[3] for

[2] Note these shortest paths are also required for the optimal solution of the TSP.
[3] Using a regular implementation of the distance matrix.

Table 5
Required computational times for preparing supporting structures.

Map	No. convex polygons	$T_{\text{partition}}$ (ms)	$T_{\text{visibility}}$ (ms)
jh	77	12	4.0
pb	41	10	0.7
h2	476	65	24.0
dense	150	12	1.8
potholes	75	8	0.7

$\rho = 0.05$ and it rapidly increases for a lower ρ, which makes this approach unsuitable for a high number of representative points. On the other hand, the approximation of the shortest paths used in the SOM based algorithms can also be used for the approximate TPP algorithm. In Table 6, basic performance indicators are presented for variants based on the exact shortest paths using the visibility graph and approximate shortest paths. In both variants, the sampling distance ρ is set to 0.1 m. Notice the refinement itself is very fast, and it is done in a fraction of the initialization time, e.g., in units of microseconds. Although the approximation reduces the initialization time (indicated in the column T_{init}%), the initialization is still a significant part of the total required computational time as the initialization is a pre-computation of all shortest paths between map vertices and only several refinement steps are needed to find a final solution. The approximation provides the same results as the exact shortest path except for the problem *jh$_{10}$-coverage*, where the final path found is about three percentage points worse due to limited precision of the approximation. However, the total required computational time and also required memory are significantly smaller. Thus, the approximation seems to be sufficient for these problems.

Based on the results, the best solutions found for each problem and over all approaches have been selected for a further comparison. The lengths of the best paths are depicted in Table 7.

6.3. Discussion

The presented results provide a performance overview of the proposed adaptation rules. The principle of the *attraction* and *alternate goal* algorithm variants are very similar; however, the *alternate goal* variant provides better results. The advantage of *alternate goal* is sampling of the goals' borders. Even though a simple approximation of the shortest path between a node (point) and a goal's segment is used, the precision of the approximation increases with node movements toward the goal, and therefore, a better point of the goal is sampled. This is an import benefit of the SOM adaptation, which allows usage of a relatively rough approximation of the shortest path.

On the other hand, the *attraction* algorithm variant is more straightforward, as the path to the centroid is utilized as a path to the fixed point goal. The fixed point goals allow us to use pre-computed all shortest paths from map vertices to the goals, which improves the precision of the approximate node–goal path.

Table 4
Comparison of the reference algorithms with the SOM based algorithms.

Problem	Reference (TPP part)			Goal interior			Attraction			Alternate goal		
	n_p	L (m)	T (s)	p_L%	T (s)	T_{init}%	p_L%	T (s)	T_{init}%	p_L%	T (s)	T_{init}%
dense$_5$-A	2 470	57.9	4.2	16.5	0.06	80	6.1	0.06	74	0.7	0.06	63
dense-small	7 857	103.5	91.7	13.9	0.22	22	9.3	0.25	19	8.0	0.33	17
h2$_5$-A	10 873	395.0	216.7	6.2	0.89	85	2.9	0.93	83	0.9	0.92	77
jh$_{10}$-coverage	23 720	108.9	3033.4	22.1	0.90	3	−2.0	1.06	2	−13.8	1.60	1
jh$_{10}$-doors	12 733	62.1	552.0	13.5	0.06	39	7.9	0.06	38	0.3	0.08	19
jh$_4$-A	6 773	56.6	97.9	15.8	0.03	25	6.8	0.04	33	0.6	0.06	24
jh$_5$-corridors	4 757	59.5	42.4	16.3	0.03	54	9.5	0.04	59	0.7	0.04	51
jh-rooms	989	87.6	0.7	17.0	0.07	22	0.5	0.08	18	0.5	0.10	23
pb$_5$-A	3 664	263.7	11.1	2.3	0.01	46	0.8	0.01	37	0.5	0.01	27
potholes$_2$-A	3 050	68.4	16.3	6.5	0.03	43	2.0	0.04	38	1.9	0.04	33

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013

Table 6
Comparison of the reference algorithms for the TPP.

Problem	n	n_p	Visibility graph			Approx. shortest path		
			L (m)	T (s)	T_{init}%	L (m)	T (s)	T_{init}%
dense$_5$-A	9	546	57.9	0.35	100	57.9	0.08	76
dense-small	35	1 733	103.5	2.71	100	103.5	0.07	54
h2$_5$-A	26	2 285	395.0	7.34	100	395.0	0.86	92
jh$_{10}$-coverage	106	12 274	110.0	499.93	100	113.5	0.54	4
jh$_{10}$-doors	21	2 684	62.1	8.60	100	62.1	0.10	21
jh$_4$-A	16	1 458	56.6	2.14	100	56.6	0.05	45
jh$_5$-corridors	11	1 015	59.5	1.05	100	59.5	0.02	37
jh-rooms	21	227	87.7	0.08	99	87.7	0.03	84
pb$_5$-A	7	784	263.7	0.32	100	263.7	0.01	29
potholes$_2$-A	13	644	68.4	0.51	100	68.4	0.02	63

Table 7
Best solutions found.

Problem	n	L_{best} (m)
dense$_5$-A	9	57.9
dense-small	35	103.4[a]
h2$_5$-A	26	395.0
jh$_{10}$-coverage	106	93.1[a]
jh$_{10}$-doors	21	62.0[a]
jh$_4$-A	16	56.6
jh$_5$-corridors	11	59.5
jh-rooms	21	87.6
pb$_5$-A	7	263.7
potholes$_2$-A	13	68.4

[a] The solution is found by the *alternate goal* algorithm.

Besides, such an approximation is less computationally intensive in the cost of higher memory requirements. However, this benefit is not evident from the results, because the *alternate goal* variant provides a faster convergence of the network.

The statistical comparison of the SOM based algorithms provides a strong statistical evidence (as the p-values obtained by the Wilcoxon test are almost always less than 0.001) that the variants proposed are different. In particular, a more sophisticated rule provides better solutions. Even though the required computational times also increase, the differences between the *attraction* and *alternate goal* variants are small and in a few cases statistically identical.

The reference algorithm provides better results than average solutions of the SOM based algorithms, except for the problem *jh*$_{10}$*-coverage*, which is an instance of the WRP with a restricted visibility range. In this particular problem, the path based on the optimal solution of the related TSP does not provide a competitive solution to the paths found by the SOM approach (regarding the PDM as well as p_L%), see Fig. 8. This indicates unsuitability of the pure combinatorial approaches for the WRP.

On the other hand, a worse average performance of the *alternate goal* algorithm is in the *dense-small* problem. In this problem, the SOM based solver was stuck at a local optima due to many obstacles; however, the best solution found over 100 trials is better than the reference solution. The best solutions are pretty much similar as can be seen in Fig. 9. In other problems, the differences in the final path length are very small and they are caused by sampling of the convex goals' boundary, i.e., a small change of the final visiting point can decrease the path a bit. Besides, the final path of SOM based solutions are determined using the approximate shortest path; thus, the approximation can also affect the solution quality.

The above two examples of solutions demonstrate that better solutions are obtained for a different sequence of goals' visits than the sequence prescribed by the optimal solution of the TSP for the centroids. In the *jh*$_{10}$*-coverage* problem, many goals overlap each other, and therefore, centroids are not a suitable representation; thus, it is not surprising the solution based on the TSP is worse. However, the *dense-small* problem also indicates that a bit better solution can be achieved for a different sequence. Notice, the solutions in Fig. 9 are almost identical, except the part approximately in the middle of the map. These examples demonstrate an advantage of the SOM based approach that includes the solution of the TSP and selection of the appropriate points of visits in a single unified way.

It should also be noted that an optimal solution of the TSP can be computationally demanding due to NP-hardness of the TSP. Therefore, regarding the results, the overall comparison of the solution quality, and the required computational time the *alternate goal* approach provides an acceptable trade-off between these two performance indicators. Besides, it also provides a greater flexibility than the reference algorithm based on a solution of the TPP, as it scales better for more complex problems with many goals, and it also includes an approximate solution of the TSP.

6.3.1. Non-convex goals

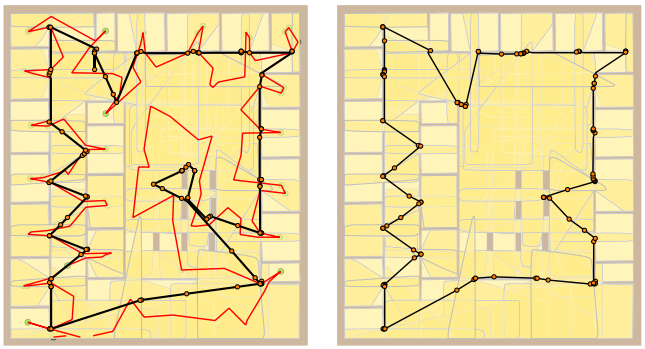
Although convex goals are assumed in the problem formulation, the presented adaptation rules do not depend on the goal convexity. The convex goals are advantageous in visual inspection tasks (covering tasks), because the whole goal region is inspected by visiting the goal at any point of the goal. Also a point representative of the convex goal can be simply computed as the centroid. If a goal is not convex a point that is inside the goal has to be determined for the *goal interior* and *attraction* algorithms. Basically any point inside the goal can be used, but a bias toward the point can be expected. The *alternate goal* algorithm variant uses a set of segments representing the goal, and therefore this algorithm can be directly used for problems with non-convex goals (see Fig. 10), which is an additional advantage of the SOM based approach for the MTP.

7. Conclusion

A self-organizing map based algorithm for the multi-goal path planning problem in the polygonal domain has been presented. Three variants of the algorithm addressing polygonal goals have been proposed and experimentally evaluated for a set of problems including an instance of the WRP with restricted visibility range (*jh*$_{10}$*-coverage*). Even though the solution quality is not guaranteed because of SOM, regarding the experimental results the algorithms provide high quality solutions. The advantage of the proposed *alternate goal* algorithm is that it provides a flexible approach to solve various routing problems including the TSP, WRP, safari route problems, and their variants in the polygonal domain, and eventually with non-convex goals.

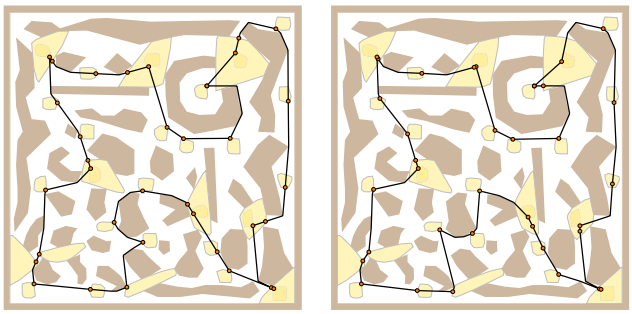
From a practical point of view, the SOM algorithms proposed are based on relatively simple algorithms and supporting structures, which is an additional benefit. The SOM adaptation schema is not a typical technique used for routing problems motivated by robotics

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013



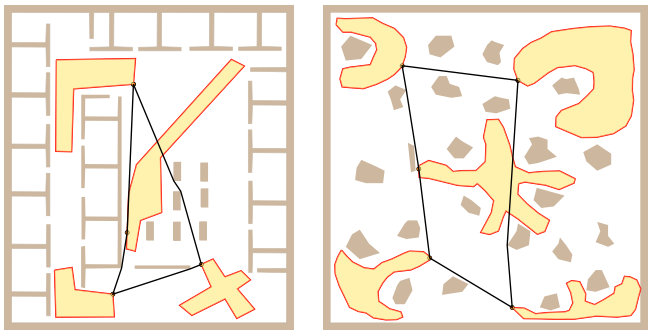
(a) The reference algorithm, $L = 108.9$ m. (b) The *alternate goal* algorithm, $L_{\text{best}} = 93.1$ m.

Fig. 8. The best solutions of jh_{10}*-coverage* found by the reference and *alternate goal* algorithms, the optimal solution of the related TSP is in red. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) The reference algorithm, $L_{\text{ref}} = 103.5$ m. (b) The *alternate goal* algorithm, $L_{\text{best}} = 103.4$ m.

Fig. 9. The best solutions found for the problem *dense-small* by the reference and *alternate goal* algorithms.



(a) The *jh* environment. (b) The *potholes* environment.

Fig. 10. Solutions found by the *alternate goal* algorithm for problems with non-convex goals.

applications. The results presented demonstrate flexibility of SOM based algorithm; thus, they may encourage roboticists to consider SOM as a suitable planning technique for other multi-goal path planning problems.

Beside the SOM approaches, a simple and straightforward reference algorithm has been presented. It provides an easily reproducible reference solution of the examined problems with polygonal goals. Therefore additional problems can be proposed to create a set of problems for benchmarking further multi-goal path planning algorithms. An initial set of such problems is provide together with the reference solutions found by the proposed approaches.

Although the proposed algorithms are able to deal with non-convex goals, the adaptation rules needs a further development and an additional evaluation in such problems, which is a subject of our future work.

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013

Acknowledgments

The presented work was supported by the Technology Agency of the Czech Republic under Project No. TE01020197 and by the Ministry of Education of the Czech Republic under Project No. LH11053 and partially by Project No. 7E08006, and by the EU Project No. 216342.

Appendix A. Nomenclature

$\mathcal{W}$	A polygonal map representing the robot workspace, $\mathcal{W} \subset \mathbb{R}^2$
$\boldsymbol{G}$	A set of (polygonal) goals to be visited
g	A goal, $g \subset \mathcal{W}$
n	A number of goals, $n = \lvert\boldsymbol{G}\rvert$
$\boldsymbol{S}_g$	Segments forming the goal g
n_p	A number of points representing all goals (for the reference algorithm)
$c(g)$	A centroid of the goal g
$S(p, g)$	An approximate path from p to g
$\lvert S(p, g)\rvert$	A length of the approximate path $S(p, g)$
L	A length of the path found
L_{ref}	A length of the reference path
L_{best}	A length of the best path found
$\mathcal{N}$	A set of nodes
ν	A node (neuron weights), $\nu \in \mathcal{W}$, $\nu \in \mathcal{N}$
σ	A learning gain (neighboring function variance)
$f(\sigma, l)$	A neighboring function
α	A gain decreasing rate
μ	A learning rate
PDM	The percent deviation from the reference path's length of the mean solution value
PDB	The percent deviation from the reference of the best solution value
$p_L\%$	An estimation of the approximate factor of SOM based solutions to the reference solution
T	The required computational time (or its average value in the case of SOM algorithms)
T_a	The required computational of the adaptation phase
$T_{\text{init}}\%$	The part (in percent) of T spent in the initialization

Appendix B. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.robot.2012.08.013.

References

[1] S.N. Spitz, A.A.G. Requicha, Multiple-goals path planning for coordinate measuring machines, in: ICRA, 2000, pp. 2322–2327.
[2] M. Saha, T. Roughgarden, J.-C. Latombe, G. Sánchez-Ante, Planning tours of robotic arms among partitioned goals, International Journal of Robotics Research 25 (3) (2006) 207–223.
[3] J. Faigl, M. Kulich, L. Přeučil, A sensor placement algorithm for a mobile robot inspection planning, Journal of Intelligent & Robotic Systems 62 (3) (2011) 329–353.
[4] M. Kulich, J. Faigl, L. Přeučil, Cooperative planning for heterogeneous teams in rescue operations, in: IEEE International Workshop on Safety, Security and Rescue Robotics, 2005, pp. 230–235.
[5] T. Danner, L. Kavraki, Randomized planning for short inspection paths, in: Proceedings of The IEEE International Conference on Robotics and Automation, ICRA, IEEE Press, San Fransisco, CA, 2000, pp. 971–976.
[6] S. Ntafos, Watchman routes under limited visibility, Computational Geometry: Theory and Applications 1 (3) (1992) 149–170.
[7] X. Tan, T. Hirata, Finding shortest safari routes in simple polygons, Information Processing Letters 87 (4) (2003) 179–186.
[8] S. Bespamyatnikh, An $o(n \log n)$ algorithm for the zoo-keeper's problem, Computational Geometry: Theory and Applications 24 (2) (2002) 63–74.
[9] J. Faigl, M. Kulich, V. Vonásek, L. Přeučil, An application of self-organizing map in the non-Euclidean traveling salesman problem, Neurocomputing 74 (2011) 671–679.
[10] J. Kubalík, J. Kléma, M. Kulich, Application of soft computing techniques to rescue operation planning, in: Artificial Intelligence and Soft Computing, ICAISC, Springer, 2004, pp. 897–902.
[11] M. Dror, A. Efrat, A. Lubiw, J.S.B. Mitchell, Touring a sequence of polygons, in: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, STOC'03, ACM Press, San Diego, CA, USA, 2003, pp. 473–482.
[12] H. González-Baños, D. Hsu, J.-C. Latombe, Motion planning: recent developments, in: S. Ge, F. Lewis (Eds.), Autonomous Mobile Robots: Sensing, Control, Decision-Making and Applications, CRC Press, 2006 (Chapter 10).
[13] R. Honsberger, Mathematical Gems, Vol. II, Mathematical Association of America, 1979.
[14] J. O'Rourke, Galleries need fewer mobile guards: a variation on Chvátal's theorem, Geometriae Dedicata 14 (1983) 273–283.
[15] T. Shermer, On recognizing unions of two convex polygons and related problems, Pattern Recognition Letters 14 (9) (1993) 737–745.
[16] S. Fisk, A short proof of Chvatal's watchman theorem, Journal of Combinatorial Theory, Series B 24 (1978) 374.
[17] M.C. Couto, C.C.S. de Souza, P.J. de Rezende, An exact and efficient algorithm for the orthogonal art gallery problem, in: Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing, SIBGRAPI'07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 87–94.
[18] J.C. Culberson, R.A. Reckhow, Covering polygons is hard, Journal of Algorithms 17 (1) (1994) 2–44.
[19] M.C. Couto, P.J. de Rezende, C.C. de Souza, An IP solution to the art gallery problem, in: Proceedings of the 25th Annual Symposium on Computational Geometry, SCG'09, ACM, New York, NY, USA, 2009, pp. 88–89.
[20] Y. Amit, J.S.B. Mitchell, E. Packer, Locating guards for visibility coverage of polygons, International Journal of Computational Geometry & Applications 20 (5) (2010) 601–630.
[21] A. Tabatabaei, A. Mohades, Clarity watchman route, in: The Kyoto International Conference on Computational Geometry and Graph Theory, Kyoto, Japan, 2007.
[22] G. Kazazakis, A. Argyros, Fast positioning of limited-visibility guards for the inspection of 2D workspaces, in: Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems, IROS 2002, vol. 3, Lausanne, Switzerland, 2002, pp. 2843–2848.
[23] H. González-Baños, A randomized art-gallery algorithm for sensor placement, in: Proceedings of the Seventeenth Annual Symposium on Computational Geometry, SCG'01, ACM, New York, NY, USA, 2001, pp. 232–240.
[24] F. Li, R. Klette, An approximate algorithm for solving the watchman route problem, in: Proceedings of the 2nd International Conference on Robot Vision, RobVis'08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 189–206.
[25] W.-P. Chin, S. Ntafos, Optimum watchman routes, in: Proceedings of the Second Annual Symposium on Computational Geometry, SCG'86, ACM Press, Yorktown Heights, New York, United States, 1986, pp. 24–33.
[26] S. Carlsson, B.J. Nilsson, S.C. Ntafos, Optimum guard covers and m-watchmen routes for restricted polygons, International Journal of Computational Geometry and Applications 3 (1) (1993) 85–105.
[27] B.J. Nilsson, Guarding art galleries: methods for mobile guards, Ph.D. Thesis, Lund University, 1995.
[28] E. Packer, Robust geometric computing and optimal visibility coverage, Ph.D. Thesis, Stony Brook University, New York, 2008.
[29] J. Faigl, Approximate solution of the multiple watchman routes problem with restricted visibility range, IEEE Transactions on Neural Networks 21 (10) (2010) 1668–1679.
[30] W.-P. Chin, S. Ntafos, The zookeeper route problem, Information Sciences: An International Journal 63 (3) (1992) 245–259.
[31] J.E. Goodman, J. O'Rourke (Eds.), Handbook of Discrete and Computational Geometry, CRC Press LLC, Boca Raton, FL, 2004.
[32] A. Dumitrescu, J.S.B. Mitchell, Approximation algorithms for tsp with neighborhoods in the plane, Journal of Algorithms 48 (1) (2003) 135–159.
[33] M. de Berg, J. Gudmundsson, M.J. Katz, C. Levcopoulos, M.H. Overmars, A.F. van der Stappen, Tsp with neighborhoods of varying size, Journal of Algorithms 57 (1) (2005) 22–36.
[34] S. Safra, O. Schwartz, On the complexity of approximating tsp with neighborhoods and related problems, Computational Complexity 14 (4) (2006) 281–307.
[35] F. Li, R. Klette, Approximate algorithms for touring a sequence of polygons, MI-Tech TR-24, The University of Auckland, Auckland, 2008. http://www.mi.auckland.ac.nz/tech-reports/MItech-TR-24.pdf.
[36] F. Li, R. Klette, Approximate shortest paths in simple polyhedra, in: I. Debled-Rennesson, E. Domenjoud, B. Kerautret, P. Even (Eds.), Discrete Geometry for Computer Imagery, in: Lecture Notes in Computer Science, vol. 6607, Springer, Berlin, Heidelberg, 2011, pp. 513–524.
[37] J. Faigl, On the performance of self-organizing maps for the non-euclidean traveling salesman problem in the polygonal domain, Information Sciences 181 (19) (2011) 4214–4229.
[38] A. Arcuri, L. Briand, A practical guide for using statistical tests to assess randomized algorithms in software engineering, in: Proceeding of the 33rd International Conference on Software Engineering, ICSE'11, ACM, New York, NY, USA, 2011, pp. 1–10.
[39] B. Yuan, M. Orlowska, S. Sadiq, Finding the optimal path in 3D spaces using edas—the wireless sensor networks scenario, in: Proceedings of the 8th International Conference on Adaptive and Natural Computing Algorithms, Part I, ICANNGA'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 536–545.

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013

[40] M.H. Overmars, E. Welzl, New methods for computing visibility graphs, in: Proceedings of the Fourth Annual Symposium on Computational Geometry, SCG'88, ACM, New York, NY, USA, 1988, pp. 164–171.
[41] D. Applegate, R. Bixby, V. Chvátal, W. Cook, CONCORDE TSP solver [Cited 8.07.10], 2003. http://www.tsp.gatech.edu/concorde.html.
[42] D. Applegate, W. Cook, A. Rohe, Chained lin-kernighan for large traveling salesman problems, INFORMS Journal on Computing 15 (1) (2003) 82–92.
[43] R. Seidel, A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons, Computational Geometry: Theory and Applications 1 (1) (1991) 51–64.

Jan Faigl received the M.Sc. degree in technical cybernetics and the Ph.D. degree in artificial intelligence and biocybernetics from the Czech Technical University in Prague (CTU), in 2003 and 2010, respectively.

Since 2006 he is a research fellow at Gerstner Laboratory, CTU. His research interests include computational geometry, planning, reasoning, and navigational methods for autonomous intelligent mobile robots.

Vojtěch Vonásek received his M.Sc. degree in cybernetics and artificial intelligence from the Czech Technical University in Prague, in 2006.

Currently he is a Ph.D. student at the Department of Cybernetics at the same university. His research interests include mobile robotics especially motion planning.

Libor Přeučil received the M.Sc. degree in technical cybernetics-robotics and the Ph.D. degree in computer vision and image processing from the Czech Technical University in Prague, in 1986 and 1993, respectively.

Since 1996 he is heading the Intelligent and Mobile Robotics group at Gerstner Laboratory, CTU. His research professional interests cover the area of intelligent robotics: self-guided vehicles control, human–robot and cooperative robotic systems, namely from the aspect of sensor fusion and uncertain information processing, robot localization, navigation and planning.

Please cite this article in press as: J. Faigl, et al., Visiting convex regions in a polygonal map, Robotics and Autonomous Systems (2012), doi:10.1016/j.robot.2012.08.013

2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Speeding Up Coverage Queries in 3D Multi-Goal Path Planning

Petr Janoušek, Jan Faigl

***Abstract*— In this paper, we present a supporting structure for speeding up visibility queries needed for a 3D multi-goal path planning arising from a robotic coverage problem where goals are sensing locations from which an object of interest can be covered. Although such coverage problems can be addressed by a decomposed approach where sensing locations are determined prior finding the sequence of their visits, the proposed approach is motivated by a solution of the problem in which sensing locations are simultaneously determined together with evaluation of the path connecting them in order to provide a cost effective inspection path. The proposed structure divides the space into elements that support determination of suitable sensing locations to cover the objects during solution of the multi-goal path planning.**

I. Introduction

A wide range of practical robotic applications can be formulated as the multi-goal path planning problem, which stands to determine the cost effective path visiting a set of goal locations. Then, the robot is requested to travel along the found path and perform its operation at the goals [1]. However, planning problems originated from inspection or surveillance missions also include a problem of determining the goal locations. Thus, the problem is to determine the most suitable locations according to the mission objective while the path connecting them will be feasible and cost effective.

In inspection or surveillance missions, the goals are locations from which an object of interest is measured using the sensoric system, which has usually limited range, and therefore, the mission task can be formulated as a variant of the robotic coverage problem. An applicable approach to address the coverage problem is to decompose the problem into an independent determination of the sensing locations and the consecutive multi-goal path planning [2], [3] that connects the locations providing the required coverage. For view planning applications where it is necessary to consider both the motion and sensing costs [4], the sensing locations should be selected simultaneously with the path planning, otherwise a poor solution would be found [5].

However, even an independent determination of the minimal number of sensing locations is computationally demanding [6], as the problem can be formulated as a variant of the art gallery problem that is NP-hard for polygonal maps. The sequencing part of the problem is a variant of the traveling salesman problem (TSP), which is also NP-hard. Thus, considering all possible sensing locations in the sequencing part of the planning is computationally intractable and regular branch-and-bound or state space search methods are not applicable because of very large search space. Therefore, approximate algorithms are preferred and sampling based methods are utilized to determine possible sensing locations considering particular visibility constraints [7], [8] for which a connecting path can be found using efficient heuristics for the TSP, e.g., [9]. Here, it is worth to mention that planning a path connecting two locations in a high-dimensional configuration space is a challenging problem itself [10], and therefore, also in this part of the problem, sampling based approaches are usually considered for high-dimensional configuration spaces together with lazy path evaluation techniques [11].

In this paper, we aim to address the coverage planning problem of determining a feasible and cost effective path from which a given set of objects will be covered using a camera with a limited visibility range, i.e., we do not consider explicitly prescribed sensing locations. The problem is a variant of the covering salesman problem [12] that can be decomposed to the set cover problem and consecutive the TSP [13], [14]. Contrary to such approaches, we rather aim to simultaneously determine the suitable sensing locations together with optimization of the path.

The proposed approach is motivated by the recent approximate algorithm for the watchman route problem (WRP) based on the self-organizing neural network [15]. The main idea of the algorithm is that a path is represented by a neural network that evolves in the problem domain, where it is adapted towards not covered parts of the environment while the coverage from the path is maintained during the evolution. This approach requires a lot of visibility queries that can be computational expensive and to avoid this issue a supporting structure based on a cover set built on top of a triangular mesh is utilized in [15], which speeded up the process significantly and make it computationally feasible.

The main contribution of this paper is to provide a step further to extend the principles used in the algorithm developed for the WRP in a plane to a more general 3D environment, where visibility queries can be computationally demanding [16], [17]. Therefore, in this paper, we propose a simple algorithm to built a supporting structure providing information what can be covered from a single point and estimation of the space from which a particular object can be covered. The approach is based on dividing the free space into n elements and associating information about possible coverage of the objects. In particular, the query about possible coverage of objects from a particular element has complexity $O(1)$ and having an object to be covered, the query about possible covering regions has complexity $O(k)$, where k is the size of the query output. It is worth

Jan Faigl is with dept. of Computer Science and Engineering, Czech Technical University in Prague, Technická 2, 166 27 Prague, Czech Republic faiglj@fel.cvut.cz

978-1-4673-5642-8/13/$31.00 ©2013 IEEE

to mention that the aim of the proposed structure is not to provide precise information about visibility according to all visibility constraints, e.g., a limited field of view. It should be rather considered as a supporting structure to restrict the search space where suitable sensing locations can be found.

Although visibility queries are part of various 3D frameworks and it is a fundamental problem in computer graphics, there is not a similar structure (to the best of our knowledge) and the frameworks are not directly applicable unless a kind of ray-tracing technique is used, which seems to be too slow for the proposed approach.

The proposed planning method is mostly similar to [14]. We also consider a general representation of the environment, where 3D objects are tessellated into a triangular mesh, where particular triangles have to be covered from the inspection path being found. Our approach mainly differs in the way how the sensing locations are determined. In [14], the sensing is considered at discrete locations, which are determined prior finding a path connecting the needed locations. In our approach, we consider a larger space (but still restricted) where sensing locations are determined during solution of the sequencing part of the planning. Thus, the proposed planning procedure can be considered as a determination of the path guaranteeing coverage of all goals. Then, the path can be further processed to determine particular discrete sensing locations, a.k.a. solution of the vision points problem. However, the determination of the locations is inherently included in the utilized process of neural network evolution. Hence, the sensing locations are found simultaneously with the path.

The paper is organized as follows. The problem definition providing an application context of the proposed speeding up structure is presented in Section II together with an overview of the proposed planning method. An algorithm to construct the supporting structure is presented in Section III. In Section IV, the proposed multi-goal path planning approach with the supporting structure is described with a use case of its application. Finally, concluding remarks are presented in Section V.

II. Problem Definition

Our motivational problem is planning a surveillance mission for a micro aerial vehicle (MAV) operating in a combined indoor/outdoor environment, where it is requested to periodically take a snapshot of Objects of Interest (OoIs); hence, the robot needs to avoid obstacles. It is worth to remind that in this paper we are concern the structure supporting the visibility (or coverage) queries. The aim of the structure is to restrict the search space for determining sensing locations that provide the requested coverage. Therefore, we consider several assumptions and an abstract formulation of the problem to make the description of the proposed structure and planning approach more straightforward.

The considered problem is planning an inspection path for a mobile robot equipped with a camera with the limited visibility range ρ that is requested to see a given set of OoIs. Here, as the first step of the proposed planning approach, we assume omnidirectional vision, e.g., realized by camera heads attached to the robot. The operational environment $\mathcal{W} \subset \mathbb{R}^3$ is represented by a set of vertices $\boldsymbol{V}_{\mathcal{W}}$ connected to triangles $\boldsymbol{T}_{\mathcal{W}}$ representing obstacles. Without lost of generality we assume the triangular mesh $\mathcal{T} = (\boldsymbol{V}_{\mathcal{W}}, \boldsymbol{T}_{\mathcal{W}})$ of the obstacle surfaces is sufficiently dense. Let $\mathcal{W}_f$ be the free space of $\mathcal{W}$, $\mathcal{W}_f \subset \mathcal{W}$. Each triangle considered in this paper is defined by a sequence of three vertices defining the triangle (surface) normal that is oriented towards $\mathcal{W}_f$. All objects to be covered form a set $\boldsymbol{M} \subseteq \boldsymbol{T}_{\mathcal{W}}$; so, the objects are represented by a set of triangles.

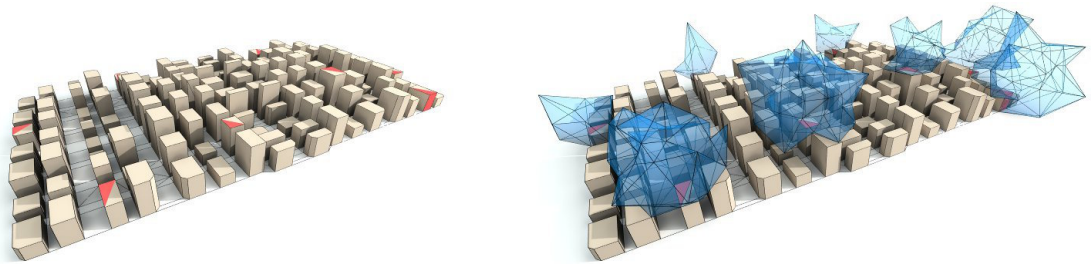
TABLE I
Used symbols

Symbol	Description
$\mathcal{W}$	the robot working space $\mathcal{W} \subset \mathbb{R}^3$
$\mathcal{C}$	the configuration space of the robot
ρ	the visibility distance range of the sensor (camera)
$\mathcal{W}_f \subseteq \mathcal{W}$	free space of $\mathcal{W}$
$\mathcal{T}$	a triangular mesh of obstacles of $\mathcal{W}$, $\mathcal{T} = (\boldsymbol{V}_{\mathcal{W}}, \boldsymbol{T}_{\mathcal{W}})$
$\boldsymbol{M}$	all parts of the objects' surfaces to be covered $\boldsymbol{M} \subseteq \boldsymbol{T}_{\mathcal{W}}$
o	the number of objects $o = \lvert\boldsymbol{M}\rvert$
m	an object to be covered $m \in \boldsymbol{M}$
G_{prm}	a graph representing roadmap $G_{prm} = (\boldsymbol{V}_{prm}, \boldsymbol{E}_{prm})$
$\boldsymbol{E}$	a tetrahedral mesh of $\mathcal{W}_f$
n	the number of tetrahedra in the mesh $n = \lvert\boldsymbol{E}\rvert$
$\boldsymbol{e}_i$	a tetrahedron of $\boldsymbol{E}$, $e_i \in \boldsymbol{E}$
$s_{\boldsymbol{e}}$	a surface (triangle) of the tetrahedron $\boldsymbol{e}$
C_m	a covering space of m with respect to ρ, $C_m \subseteq \mathcal{W}_f, C_m \subseteq \boldsymbol{E}$
$\boldsymbol{C}$	a union of all covering spaces $\boldsymbol{C} = \bigcup_{m \in \boldsymbol{M}} C_m$
$(\boldsymbol{e}, m, \boldsymbol{T})$	visibility dependency of $\boldsymbol{e}$ with respect to m, where $\boldsymbol{T}$ are surfaces of $\boldsymbol{e} \in \boldsymbol{E}$ with respect to $m \in \boldsymbol{M}$ supporting coverage of m from a point $p \in \boldsymbol{e}$
$\mathcal{V}_d$	a set of all visibility dependencies for each $\boldsymbol{e_i} \in \boldsymbol{E}$ with respect to m, $\mathcal{V}_d(m) = \{(\boldsymbol{e}_1, m, \boldsymbol{T}), \ldots, (\boldsymbol{e}_n, m, \boldsymbol{T})\}$

We consider a mobile robot capable of moving in 3D environment (e.g., MAV) and for its point-to-point motion planning we consider the notion of the configuration space $\mathcal{C}$ and the Probabilistic Road Map (PRM) planner [18].

For each object $m \in \boldsymbol{M}$ we define a covering space $C_m \subseteq \mathcal{W}_f$ from which m can be seen using the sensor with the distance range ρ. C_m is found by the algorithm proposed in Section III. Although $\mathcal{C}$ can be a high dimensional configuration space, we consider the visibility in 3D, and therefore, we divide $\mathcal{W}_f$ into a set of tetrahedra considering the triangular mesh $\mathcal{T}$, e.g., using [19]. We assume that for each tetrahedron the ratios of the lengths of tetrahedron's edges is as small as possible, which pragmatically means the tetrahedral mesh is sufficiently dense and the centroid of each tetrahedron is inside the tetrahedron. We consider the tetrahedral mesh $\boldsymbol{E}$ as a set of tetrahedra $\boldsymbol{e}$, where each $\boldsymbol{e}$ consists of four triangles, without a formal introduction of tetrahedral mesh vertices and triangles to make the text more readable. The particular point of our interest are tetrahedra incident with $m \in \boldsymbol{M}$ as such a tetrahedron is the initial step for building C_m. An example of visualization of the environment $\mathcal{W}$ and objects $\boldsymbol{M}$ is shown in Fig. 1.

$\mathcal{C}$ is linked with $\mathcal{W}$ using the centroids of the tetrahedra $\boldsymbol{E}$ and vertices of the roadmap find by the PRM method. For simplicity, we consider centroids as the initial configurations for building the roadmap, instead of a random sampling $\mathcal{C}$.



(a) environment and objects of interest – $\mathcal{W}$, $\boldsymbol{M}$ (b) covering spaces – $\boldsymbol{E}$

Fig. 1. Example of the environment representation, tetrahedralization of the freespace, objects of interest (red triangles) and their covering spaces. The red triangles denote the particular objects (part of them) of interest.

Then, we discard all tetrahedra without connected centroids with the roadmap; hence, for each tetrahedron, we have at least one feasible configuration. In the rest of the paper, we assume that all $\boldsymbol{e} \in \boldsymbol{E}$ have at least one feasible configuration and there exists a feasible path to other tetrahedra (their associated configurations). The built roadmap forms a graph $G_{prm} = (\boldsymbol{V}_{prm}, \boldsymbol{E}_{prm})$, where each vertex $v \in \boldsymbol{V}_{prm}$ is associated to a configuration $c \in \mathcal{C}_{free}$, which projection to $\mathbb{R}^3$ is the centroid of the particular $\boldsymbol{e} \in \boldsymbol{E}$.

The used notation is depicted in Table I. Having the above defined preliminaries, the planning problem can be defined as follows. *Find the shortest closed inspection path $\boldsymbol{I}$ in the graph G_{prm}, $\boldsymbol{I} = (v_1, v_2, \dots v_k), v_1 = v_k, v_i \in G_{prm}$ such that all objects $\boldsymbol{M}$ will be seen from $\boldsymbol{I}$ by the sensor with the visibility range ρ, i.e., for each $m \in \boldsymbol{M}$ there exists $v_i \in \boldsymbol{I}$ such that $v_i \in C_m$.*

A. Planning Method Overview

The planning problem is basically a selection of configurations from $\boldsymbol{V}_{prm}$ that are in the covering spaces $\boldsymbol{C}$ such that the path connecting them is the shortest one, which is the problem addressed by the self-organizing map (SOM) for the WRP [15] and later used for finding the path connecting a set of convex goals in 2D [20]. Considering such a selection technique is available, the planning approach can be summarized in the following steps:

1) Tetrahedralization of the working environment (e.g., using [19]);
2) Generation of the motion planning roadmap (e.g., using the PRM method [18]);
3) Construction of covering spaces C_m for each $m \in \boldsymbol{M}$;
4) Determination of the inspection path using SOM;

The first two steps are straightforward. The construction of the covering spaces is presented in the next section and a brief description how the SOM technique can be utilized is presented in Section IV.

III. Construction of Covering Spaces

The covering space C_m of the object m consists of a set of tetrahedra and represents a part of $\mathcal{W}_f$ from which the whole object m can be seen with respect to the limited visibility range ρ. Thus, for each m we are looking for tetrahedra C_m such that distance of vertices of each tetrahedron $\boldsymbol{e} \in C_m$ from the vertices of m is less or equal to ρ. However, we have to deal with obstacles, as for each point of C_m the visibility of the whole m must be guaranteed. The proposed construction algorithm for determining C_m is based on an iterative procedure that inserts a new tetrahedron $\boldsymbol{e}$ to C_m while the fundamental constraint of the visibility of m from $p \in \boldsymbol{e}$ is preserved.

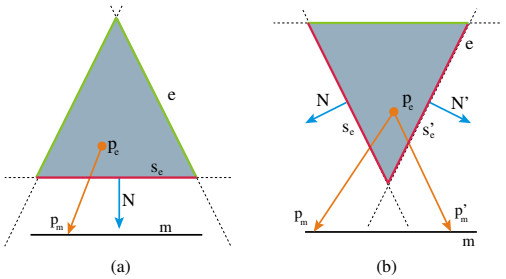


Fig. 2. Examples of the visibility dependency of the tetrahedron $\boldsymbol{e}$ with respect to m; (a) $(\boldsymbol{e}, m, \boldsymbol{T}) = \{s_e\}$; (b) $(\boldsymbol{e}, m, \boldsymbol{T}) = \{s_e, s'_e\}$.

The visibility means that for any point $p \in \boldsymbol{e}$ the ray connecting p with any point of m does not intersect an obstacle. Hence, the key point for keeping the visibility of m from $\boldsymbol{e}$ are surfaces of $\boldsymbol{e}$ that can be intersected by such a ray. We denote such surfaces as $\boldsymbol{T} \subset \boldsymbol{e}$ with respect to m and define a notion of visibility dependency $(\boldsymbol{e}, m, \boldsymbol{T})$, which assign $\boldsymbol{T}$ to $\boldsymbol{e}$ regarding m. For each $m \in \boldsymbol{M}$ and tetrahedron $\boldsymbol{e} \in \boldsymbol{E}$ we find the set of surfaces $\boldsymbol{T}$ by testing if a vertex of m is on the opposite side of the plane defined by $s_{\boldsymbol{e}} \in \boldsymbol{e}$ than a point inside $\boldsymbol{e}$. The test is performed as a scalar product of the $s_{\boldsymbol{e}}$ normal and ray (p_e, p_m). The principle is schematically depicted in Fig. 2. The complexity of determination of all visibility dependencies is $\Theta(no)$, where n is the number of tetrahedra ($n = |\boldsymbol{E}|$) and o is the number of objects $o = |\boldsymbol{M}|$.

Having the visibility dependencies $(\boldsymbol{e}, m, \boldsymbol{T})$ for each $\boldsymbol{e} \in \boldsymbol{E}$ and $m \in \boldsymbol{M}$, the building of C_m is relatively straightforward as we basically consider each tetrahedron $\boldsymbol{e}$ and test if the visibility constraint will be satisfied after insertion of $\boldsymbol{e}$ into C_m. However, the tetrahedra can be inserted in an arbitrary order, and therefore, we define a notion of transitive dependency of $\boldsymbol{e}$ on other tetrahedra that must be inserted to

Algorithm 1: Construction of covering space C_m

Input: m – an object to be covered
Input: $\boldsymbol{E}, \mathcal{V}_d(m)$ – tetrahedral mesh and all visiblity dependencies with respect to m
Output: C_m – the covering space for the object m

$\boldsymbol{e}_{obj} \leftarrow$ get $\boldsymbol{e}$ such that $\boldsymbol{e} \in \boldsymbol{E} \wedge s \in \boldsymbol{e} \wedge \text{incident}(s, m)$;
$C_m \leftarrow \{\boldsymbol{e}_{obj}\}$;
$E_{free} \leftarrow \boldsymbol{E} \setminus C_m$;
$E_{close} \leftarrow \emptyset$;
while $\exists \boldsymbol{e} \in \boldsymbol{E}_{free} \wedge \boldsymbol{e}_{nbr} \in C_m \wedge \text{incident}(\boldsymbol{e}, \boldsymbol{e}_{nbr})$ **do**
  $G(\boldsymbol{E}_G, \boldsymbol{H}) \leftarrow \text{get\_dependency}(m, C_m, (\boldsymbol{e}, m, \boldsymbol{T}))$;
  $C_m \leftarrow \text{add\_tetrahedra}(C_m, G(\boldsymbol{E}_G, \boldsymbol{H}))$;

C_m in order to satisfy the visibility of m from $\boldsymbol{e}$. So, during the iterative insertion we construct an auxiliary graph G with information about the transitive dependency. In addition, we maintain two sets $\boldsymbol{E}_{close}$ containing tetrahedra transitively dependent on obstacle or closed tetrahedron and $\boldsymbol{E}_{free}$ with all not yet processed tetrahedra. For the incremental construction of C_m we need a relation of neighbouring tetrahedra (or incident triangles), and therefore, we define the Boolean operator $\text{incident}(\boldsymbol{e}_1, \boldsymbol{e}_2)$ that is true if $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ share the same triangle (except its orientation) and false otherwise. The construction procedure is depicted in Algorithm 1 and the sub-procedures get_dependency and add_tetrahedra in Algorithm 2 and Algorithm 3, respectively.

Algorithm 2: Construction of the dependency graph G

Input: m – an object to be covered
Input: $\boldsymbol{e}_0$ – tetrahedron being added to C_m
Input: $\boldsymbol{E}, \mathcal{V}_d(m)$ – tetrahedral mesh and all visiblity dependencies for $\boldsymbol{e}_i \in \boldsymbol{E}$ with respect to m
Output: $G(\boldsymbol{E}_G, \boldsymbol{H})$ – the dependency graph for $\boldsymbol{e}_0$

$\boldsymbol{E}_{tmp} \leftarrow \{\boldsymbol{e} | \boldsymbol{e} \in \boldsymbol{E} \wedge \text{incident}(\boldsymbol{e}_0, \boldsymbol{e})\} \cup \{\boldsymbol{e}_0\}$;
$\boldsymbol{H} \leftarrow \emptyset$;
while $\boldsymbol{E}_{tmp} \cap \boldsymbol{E}_{free} \neq \emptyset$ **do**
  $\boldsymbol{e}_i \leftarrow$ get $\boldsymbol{e}$ such that $\boldsymbol{e} \in \boldsymbol{E}_{tmp} \wedge \boldsymbol{e} \in \boldsymbol{E}_{free}$;
  $\boldsymbol{E}_{free} \leftarrow \boldsymbol{E}_{free} \setminus \{\boldsymbol{e}_i\}$;
  $\boldsymbol{T}_{neigh} \leftarrow \{t | t \in (\boldsymbol{e}_i, m, \boldsymbol{T}) \wedge \text{incident}(\boldsymbol{e}_i, \text{t})\}$;
  if $\boldsymbol{T}_{neigh} \cap \boldsymbol{T}_{\mathcal{W}} \neq \emptyset$ **then**
    $\boldsymbol{E}_G \leftarrow \boldsymbol{E}_G \cup \{\boldsymbol{e}_i\}$ // *$\boldsymbol{e}_i$ is incident with obstacle*;
    $\boldsymbol{E}_{close} \leftarrow \boldsymbol{E}_{close} \cup \{\boldsymbol{e}_i\}$;
  else
    for $\boldsymbol{e}_{neigh} \in \{\boldsymbol{e} | \boldsymbol{e} \in \boldsymbol{E} \wedge t \in \boldsymbol{T}_{neigh} \wedge t \in \boldsymbol{e}\}$ **do**
      $\boldsymbol{E}_{tmp} \leftarrow \boldsymbol{E}_{tmp} \cup (\boldsymbol{E}_{free} \cap \{\boldsymbol{e}_{neigh}\})$;
      $\boldsymbol{E}_G \leftarrow \boldsymbol{E}_G \cup \{\boldsymbol{e}_{neigh}\}$;
      $\boldsymbol{H} \leftarrow \boldsymbol{H} \cup \{(\boldsymbol{e}_i, \boldsymbol{e}_{neigh})\}$ // *add the relation*;
$G \leftarrow G(\boldsymbol{E}_G, \boldsymbol{H})$;

In Algorithm 2, the auxiliary graph $G = (\boldsymbol{E}_G, \boldsymbol{H})$ is determined. G represents all tetrahedra on which $\boldsymbol{e}$ being added to C_m is transitively dependent, i.e., each vertex of G represents a tetrahedron and an oriented edge $h \in \boldsymbol{H}$ between $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ indicates the transitive dependency of $\boldsymbol{e}_1$ on $\boldsymbol{e}_2$. The tetrahedron $\boldsymbol{e}_0$ can be added to C_m only if it is not dependent on a tetrahedron that is in $\boldsymbol{E}_{close}$ or it is not incident with an obstacle. The procedure is repeated until all transitively dependent tetrahedra are processed (Line 3). Then, G is used for adding all tetrahedra that are not incident with an obstacle or closed tetrahedron into C_m using the procedure add_tetrahedra, see Algorithm 3.

Algorithm 3: Adding tetrahedra to C_m

Input: m – an object to be covered
Input: $G(\boldsymbol{E}_G, \boldsymbol{H})$ – the current dependency graph
Output: C_m – the covering space for the object m

$\boldsymbol{E}_{act} \leftarrow \boldsymbol{E}_{close}$ // *test all close tetrahedra*;
while $|\boldsymbol{E}_{act}| > 0$ **do**
  $\boldsymbol{e}_{act} \leftarrow$ get a tetrahedron from $\boldsymbol{E}_{act}$;
  $\boldsymbol{E}_{close} \leftarrow \boldsymbol{E}_{close} \cup \{\boldsymbol{e}_{act}\}$;
  $\boldsymbol{E}_{act} \leftarrow \boldsymbol{E}_{act} \setminus \{\boldsymbol{e}_{act}\}$;
  $\boldsymbol{E}_{dep} \leftarrow \{\boldsymbol{e} | \boldsymbol{e} \notin \boldsymbol{E}_{close} \wedge (\boldsymbol{e}, \boldsymbol{e}_{act}) \in \boldsymbol{H}\}$;
  $\boldsymbol{E}_{act} \leftarrow \boldsymbol{E}_{act} \cup \boldsymbol{E}_{dep}$;
$C_m \leftarrow C_m \cup (\boldsymbol{E}_G \setminus \boldsymbol{E}_{close})$ // *add non closed dep. tet.*;

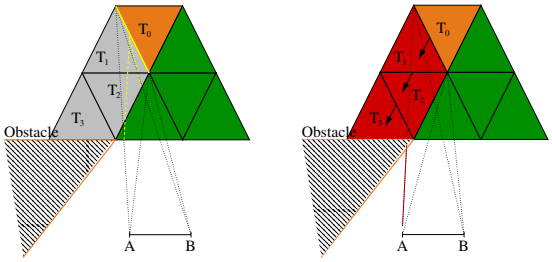


Fig. 3. An example of the transitive dependency, triangles represent surfaces' of tetrahedra. The orange tetrahedra are tested for being added to C_m, the green triangles are tetrahedra already in C_m, and gray triangles are not yet processed tetrahedra. The red triangles are tetrahedra from, which visibility to segment (AB) of m cannot be guaranteed. In the left figure, the tetrahedra can be added to C_m while for the right case, T_0 is transitively dependent on T_3 with respect to m and visibility of m from T_0 cannot be guaranteed because T_3 is incident with an obstacle.

An example of the transitive dependency is shown in Fig. 3 using triangles (2D slice of tetrahedron) for a clarity. Although the proposed construction of the covering space C_m is only approximation, the crucial point of C_m is that it guarantees coverage of m within the limited sensor range ρ.

A. Computational Complexity and Queries

The computational complexity of Algorithm 2 is $O(n)$ as in the worst case all tetrahedra can be processed. It is also the case of Algorithm 3 where up to $O(n)$ elements can be examined. Thus, the total complexity of the construction algorithm is $\Theta(no)$.

Regarding a usage of the structure, it provides $O(1)$ queries for test if a point in a particular tetrahedron can cover an object. For a general point, the complexity depends on the finding the particular tetrahedron for the point, which can be based on the kd-tree using centroids of the tetrahedra. Such a query can be answered in the average complexity $O(\log n)$.

TABLE II
CONSTRUCTING TIMES FOR TETRAHEDRAL MESH COVERING SPACES

Parameter	Scene 1	Scene 2	Scene 3	Scene 4
No. of triangles	1280	1280	2444	2444
No. of tetrahedra	2 538	10 852	27 071	113 505
Tetrahedral mesh [ms]	90	351	809	3 705
Covering spaces [ms]	33	46	271	746

However, the closest centroid does not guarantee the point is inside the particular tetrahedron for a weakly constructed tetrahedral mesh. Then, for such a case a local search method can be applied. On the other hand, the proposed planning approach considers configurations with associated information about its tetrahedron; hence, it is not necessary to perform the finding query and the information about covering is available instantly.

We consider four scenes with different number of tetrahedra to provide an overview of the real computational requirements using a computer with iCore7 3.4 GHz CPU. Scene parameters and construction times are depicted in Table II. In all cases, 10 covering spaces are constructed.

In addition, we consider 100 000 random queries in the Scene 2, where the average query time to find a tetrahedron for a random point is about 94 μs using an exhaustive search.

IV. MULTI-GOAL INSPECTION PLANNING

The planning algorithm for determining the inspection path $\boldsymbol{I}$ is inspired by the self-organizing map (SOM) approach for the WRP [15], which is based on SOM adaptation schema for the TSP [21]. However, rather than SOM for the polygonal domain, we utilize a SOM variant for a graph input [22] and use the roadmap G_{prm} as the graph.

The planning algorithm is basically an unsupervised learning procedure for two-layered competitive neural network. An input vector is a vertex of G_{prm} representing an object being covered. The output layer consists of k units representing neuron weights. The output units are organized into a unidimensional structure $\mathcal{N} = (\nu_1, \ldots, \nu_k)$, which is called a ring in the rest of this paper. Hence, the ring represents a path evolving in the graph G_{prm}. The weights are coordinates in $\mathcal{C}$; however, restricted to the roadmap G_{prm}.

The learning process is an iterative procedure in which objects $m \in \boldsymbol{M}$ are presented to the network in a random order. For simplicity, we can assume that each object m has associated a particular vertex $v_m \in \boldsymbol{V}_{prm}$. First, a winning neuron ν^* with the shortest distance to v_m is determined in the competitive phase. Then, ν^* together with its neighbouring nodes are adapted towards v_m. The adaptation is terminated if each object has its winner neuron in a sufficiently small distance. Using the graph input, the learning process can be described as an evolution of a path (ring) in the problem domain defined by the G_{prm}. The adaptation of nodes can be interpreted as their movement towards the presented goal along paths found in the G_{prm}, e.g., using Dijkstra's algorithm. The neurons' weights represent the inspection path $\boldsymbol{I}$ and the order of object visits can be retrieved by traversing the output layer (ring).

The procedure is relatively simple but its simplicity provides a great flexibility to address variant of inspection planning. First, we introduce usage of the covering space C_m. The modification of the standard TSP algorithm is straightforward. In the competitive phase, we consider C_m to select not only the winning neuron ν^* but also a new goal towards which the winner will be updated; thus, the competitive rule is as follows

$$(\nu^*, p) \leftarrow \mathrm{argmin}_{\nu \in \mathcal{N}, \boldsymbol{e} \in C_m}(\mathrm{distance}(\nu, \mathrm{centroid}(\boldsymbol{e})),$$

where p is a point of the particular $\boldsymbol{e}$, e.g., the centroid of $\boldsymbol{e}$. Then, ν^* and its neighbouring neurons are adapted towards p instead of v_m. The idea is that instead of direct planning to visit the object, we rather prefer to find some point from which the object will be covered. Here, we employ the constructed covering space C_m, which allows the proposed straightforward extension without any performance lost related to compute the coverage.

Notice, it is not necessary to consider all tetrahedra in C_m for evaluation of the best possible goal p. We can sample only few of them or we can use other techniques. The advantage of C_m is that it provides explicit representation of possible goal candidates from which the object can be covered.

A. Use Case

The feasibility of the proposed planning approach based on the covering spaces C_m has been verified in a city like environment represented by 2 444 triangles, see Fig. 1. We consider restricted visibility range ρ=2. The objects to be covered are 10 triangles representing 7 physical objects of interest. One object is represented by 3 incident triangles that is located at the bottom right part of the environment. Two underlying graphs with 27 071 and 113 505 nodes (for two tetrahedral meshes) are used to provide an overview of real computational requirements.

TABLE III
COMPUTATIONAL TIMES OF PARTICULAR PLANNING PARTS

Tet. mesh	PRM	Covering spaces	SOM epoch=20	SOM epoch=40	SOM epoch=60
0.8 s	0.7 s	210 ms	0.6 min	1.1 min	1.4 min
3.7 s	2.8 s	352 ms	3.3 min	6.5 min	8.4 min

Particular required computationally times of each part of the proposed planning method are depicted in Table III for a single core CPU running at 3.4 GHz. The final solution has been found after 60 learning epoch of the SOM learning procedure. However, SOM provides a solution at the end of each epoch and for example after 40 epochs the path length was 38.6 while the final solution after 60 epoch has length 27.3. In the case of a finer tetrahedral mesh (with 113 505 tetrahedra) the lengths are 35.2 and 26.3, respectively. An example of found solution is visualized in Fig. 4.

During the learning, particular distances and paths between nodes are computed on demand and saved for a latter usage, therefore, the complete distance matrix is not

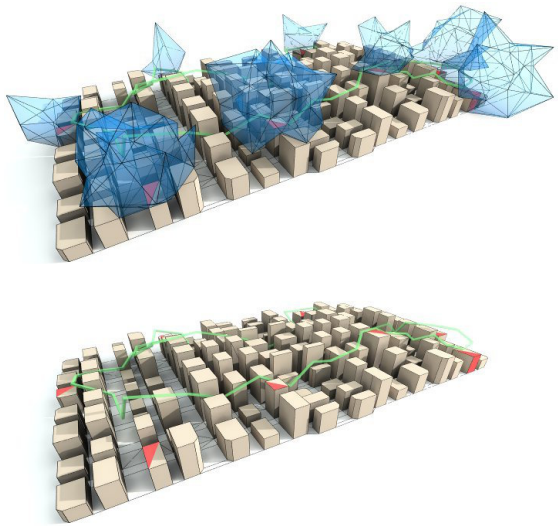


Fig. 4. An example of found solution in the city-like scenario.

computed. The memory footprint is about 1 GB and 10 GB, respectively, which is significantly lower than a required storage for the complete distance matrix. It should be noted that herein presented results are part of the feasibility study of the proposed covering space structure and the planning method, and we consider the distance matrix for simplicity.

V. Conclusion

In this paper, we present an algorithm for construction supporting structures to avoid computing covering queries in the proposed 3D inspection planning and demonstrate how it can be utilized in multi-goal inspection planning. The structure can be considered as an enabling technique for applying self-organizing map (SOM) based planning principles in 3D environments. SOM provides interesting results for inspection planning in planar environments where they are able to solve multi-goal path planning problems with point or polygonal goals [20] and even problems without explicitly prescribed goals [15].

The proposed structure of covering spaces is our early results towards applying self-organizing principles in a high-dimensional space. Although the current problem formulation assume an omnidirectional vision, the proposed principles provide a different mechanism of searching the state space than regular branch and bound algorithms or decomposed approaches. It allows a simultaneous determination of suitable goal locations during solving the sequencing part of the multi-goal path planning. Therefore, additional constraints can be considered during the self adaptation of the used neural network, e.g., a coverage from the ring can be computed according to the robot's orientation. Hence, we expect the proposed framework will allow to consider not only sensor with a limited field of view, but additional motion constraints of the robot. In addition, it is not necessary to precompute roadmap using centroids of all tetrahedra. Lazy motion planning such as [14] can be combined with the covering spaces. Such further developments are subject of our current work.

Acknowledgments

The work of J. Faigl was supported by the Czech Science Foundation (GAČR) under research project No. 13-18316P.

References

[1] S. N. Spitz and A. A. G. Requicha, "Multiple-Goals Path Planning for Coordinate Measuring Machines," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2000, pp. 2322–2327.

[2] F. Zhao, X. Xu, and S. Xie, "Computer-aided inspection planning-the state of the art," *Computers in Industry*, vol. 60, no. 7, pp. 453–466, 2009.

[3] T. Danner and L. E. Kavraki, "Randomized Planning for Short Inspection Paths," in *Proceedings of The IEEE International Conference on Robotics and Automation (ICRA)*. San Fransisco, CA: IEEE Press, April 2000, pp. 971–976.

[4] W. R. Scott, G. Roth, and J.-F. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Comput. Surv.*, vol. 35, no. 1, pp. 64–96, 2003.

[5] P. Wang, "View planning with combined view and travel cost," Ph.D. dissertation, Simon Fraser University, 2007.

[6] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Near-optimal sensor placements: maximizing information while minimizing communication cost," in *The Fifth International Conference on Information Processing in Sensor Networks (IPSN)*, 2006, pp. 2–10.

[7] H. H. Gonzalez-Banos and J.-C. Latombe, "Navigation Strategies for Exploring Indoor Environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.

[8] J. Faigl, M. Kulich, and L. Přeučil, "A sensor placement algorithm for a mobile robot inspection planning," *Journal of Intelligent & Robotic Systems*, vol. 62, no. 3-4, pp. 329–353, 2011.

[9] K. Helsgaun, "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic," *European Journal of Operational Research*, vol. 126, no. 1, 2000.

[10] S. M. Lavalle, *Planning Algorithms*. Cambridge University Press, May 2006.

[11] M. Saha, T. Roughgarden, J.-C. Latombe, and G. Sánchez-Ante, "Planning Tours of Robotic Arms among Partitioned Goals," *Int. J. Rob. Res.*, vol. 25, no. 3, pp. 207–223, 2006.

[12] J. R. Current and D. A. Shilling, "The covering salesman problem," *Transportation Science*, vol. 23, no. 3, 1989.

[13] P. S. Blaer and P. K. Allen, "View planning and automated data acquisition for three-dimensional modeling of complex sites," *J. Field Robot.*, vol. 26, no. 1112, pp. 865–891, Nov. 2009.

[14] B. Englot and F. Hover, "Planning complex inspection tasks using redundant roadmaps," in *15th International Symposium of Robotics Research (ISRR)*, Flagstaff, AZ, August 2011.

[15] J. Faigl, "Approximate Solution of the Multiple Watchman Routes Problem with Restricted Visibility Range," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1668–1679, 2010.

[16] F. Durand, G. Drettakis, and C. Puech, "The 3D Visibility Complex," *ACM Transactions on Graphics*, vol. 21, no. 2, pp. 176–206, 2002.

[17] M. Nouri Bygi and F. Ghodsi, "3d visibility and partial visibility complex," in *International Conference on Computational Science and its Applications, (ICCSA)*, aug. 2007, pp. 208–207.

[18] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[19] H. Si, "Tetgen - a quality tetrahedral mesh generator and a 3d delaunay triangulator," [cit. 09-05-2012]. [Online]. Available: http://tetgen.berlios.de

[20] J. Faigl, V. Vonásek, and L. Přeučil, "Visiting convex regions in a polygonal map," *Robotics and Autonomous Systems*, 2012, (*accepted, paper in press*), http://dx.doi.org/10.1016/j.robot.2012.08.013.

[21] S. Somhom, A. Modares, and T. Enkawa, "A self-organising model for the travelling salesman problem," *Journal of the Operational Research Society*, pp. 919–928, 1997.

[22] T. Yamakawa, K. Horio, and M. Hoshino, "Self-Organizing Map with Input Data Represented as Graph," in *Neural Information Processing*. Springer Berlin / Heidelberg, 2006, pp. 907–914.

Simple Yet Stable Bearing-Only Navigation

Tomáš Krajník, Jan Faigl, Vojtěch Vonásek, Karel Košnar, Miroslav Kulich, and Libor Přeučil
Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, 121 35 Praha 2, Karlovo náměstí 13, Czech Republic
e-mail: tkrajnik@labe.felk.cvut.cz, xfaigl@labe.felk.cvut.cz, vonasek@labe.felk.cvut.cz, kosnar@labe.felk.cvut.cz, kulich@labe.felk.cvut.cz, preucil@labe.felk.cvut.cz

Received 8 January 2010; accepted 7 June 2010

This article describes a simple monocular navigation system for a mobile robot based on the map-and-replay technique. The presented method is robust and easy to implement and does not require sensor calibration or structured environment, and its computational complexity is independent of the environment size. The method can navigate a robot while sensing only one landmark at a time, making it more robust than other monocular approaches. The aforementioned properties of the method allow even low-cost robots to effectively act in large outdoor and indoor environments with natural landmarks only. The basic idea is to utilize a monocular vision to correct only the robot's heading, leaving distance measurements to the odometry. The heading correction itself can suppress the odometric error and prevent the overall position error from diverging. The influence of a map-based heading estimation and odometric errors on the overall position uncertainty is examined. A claim is stated that for closed polygonal trajectories, the position error of this type of navigation does not diverge. The claim is defended mathematically and experimentally. The method has been experimentally tested in a set of indoor and outdoor experiments, during which the average position errors have been lower than 0.3 m for paths more than 1 km long.

1. INTRODUCTION

The fundamental problem of mobile robotics is to autonomously navigate a mobile robot along a given path. To fulfill this task efficiently, a robot should maintain some knowledge about its surrounding environment, especially its position relative to the path or desired destination. Such knowledge may be represented in the form of a map, which can be used to estimate the robot position as well as for motion planning. The map is either known a priori and the robot performs localization or is created online and the mobile robot performs so-called simultaneous localization and mapping (SLAM).

The solid mathematical background of the Kalman filter (Kalman, 1960) allowed the research community to establish a sufficient theoretical framework for extended Kalman filter (EKF)-based SLAM. Proof of EKF convergence (Dissanayake, Newman, Clark, Durrant-Whyte, & Csorba, 2001) and lower bounds (Gibbens, Dissanayake, & Durrant-Whyte, 2000) on robot position uncertainty have been formulated. Upper bounds are discussed in the paper by Mourikis and Roumeliotis (2004), where the authors emphasize the importance of robot heading precision during the mapping process. To our knowledge, there is no other paper concerning upper bounds of EKF position estimation. Unfortunately, optimality of the Kalman filter is proven only for linear systems and therefore the main weakness of EKF methods lies in the linearization. The papers by Julier and Uhlmann (2001) and Martinelli, Tomatis, and Siegwart (2005) indicate that due to errors introduced in linearization, EKF methods might provide inconsistent results. Although the linearization process poses a significant threat to the consistency of the position estimation, it can be elegantly avoided using the inverse depth representation (Civera, Davison, & Montiel, 2008; Montiel, Civera, & Davison, 2006).

A multimedia file may be found in the online version of this article.

1.1. Vision-Based Navigation

The theoretical solutions of bearing-only SLAM have gained importance as the computational power of today's computers allows real-time image processing. The nature of visual information allows us to build sparse maps from well-distinguishable landmarks (Lowe, 1999, 2004), which are relatively easy to register. However, the range information is not provided directly by standard cameras. Some bearing-only methods use stereovision in order to obtain immediate range information (Kidono, Miura, & Shirai, 2000). Other methods substitute stereovision by motion and use a single monocular camera (Davison, Reid, Molton, & Stasse, 2007; Holmes, Klein, & Murray, 2008; Montiel et al., 2006).

Most monocular approaches are computationally complex and achieve low operational speeds when mapping large-scale environments. This problem can be solved by dividing a large global map into smaller maps with mutual position information (Bosse, Newman, Leonard, &

Journal of Field Robotics 27(5), 511–533 (2010)
View this article online at wileyonlinelibrary.com • DOI: 10.1002/rob.20354

Teller, 2004; Clemente, Davison, Reid, Neira, & Tardós, 2007; Estrada, Neira, & Tardós, 2005; Williams, Cummins, Neira, Newman, Reid, et al., 2009).

A different approach is to build an environment map in advance and then use the map for localization (Blanc, Mezouar, & Martinet, 2005; Chen & Birchfield, 2009; Matsumoto, Inaba, & Inoue, 1996; Royer, Lhuillier, Dhome, & Lavest, 2007; Segvic, Remazeilles, Diosi, & Chaumette, 2007). In Royer et al. (2007), a monocular camera is carried through an environment and a video is recorded. The recorded video is then processed (in a matter of several minutes) and subsequently used to guide a mobile robot along the same trajectory. Chen and Birchfield (2006, 2009) present an even simpler form of navigation in a learned map. Their method utilizes a map consisting of salient image features remembered during a teleoperated drive. The map is divided into several conjoined segments, each associated with a set of visual features detected along it and a milestone image indicating the segment end. When a robot navigates a segment, its steering commands are calculated from positions of currently recognized and remembered features. The robot using this method moves forward with a constant speed and steers right or left with a constant velocity or does not steer at all. In Chen and Birchfield (2006), the segment end was detected by means of comparing the milestone image with the current view. The improved version (Chen & Birchfield, 2009) of the qualitative navigation uses a more sophisticated method to determine the segment end. The method takes into account the odometry, the current heading, and the similarity of the current and the milestone images. Still, the authors mention some problems with detection of the segment end. We claim that the segment end can be detected solely by the odometry and the comparison with the milestone image is not always necessary. Comparison with the milestone image increases robustness of the navigation method in cases of wheel slippage and other odometry errors.

The map-and-replay approach is closely related to visual servoing, in which the control of a robot is based on visual measurements (Chaumette & Hutchinson, 2006, 2007). Control inputs are either computed directly by a comparison of the current and reference images (Remazeilles & Chaumette, 2007; Segvic, Remazeilles, Diosi, & Chaumette, 2009) or by a computation of camera coordinates in the world reference frame (DeMenthon & Davis, 1992; Wilson, Hulls, & Bell, 1996). Normally, the visual servoing relies on a geometrical approach to calculate relations of map landmarks to the current image salient points (Segvic et al., 2009). These relations are used to calculate an interaction matrix (Chaumette & Hutchinson, 2006), which links observations to control inputs of the robot. The robot's control input can be computed from the Jacobian (Burschka & Hager, 2001), which relates world and image points, or from homography or fundamental matrices (Guerrero, Martinez-Cantin, & Sagüés, 2005; Remazeilles & Chaumette, 2007), which relate coordinates between actual and reference images. A strong reliance on the geometrical representation requires either camera calibration (Burschka & Hager, 2001; Remazeilles & Chaumette, 2007; Segvic et al., 2009) or structured environment (Guerrero et al., 2005; Remazeilles & Chaumette, 2007). A more detailed overview of visual servoing approaches related to the field of mobile robotics is presented in Chen and Birchfield (2009) and Segvic et al. (2009). Contrary to the visual servoing approach, our method does not require a calibrated camera and does not rely on the environment structure.

1.2. Motivation

The target of our efforts is to create a system that would be able to reliably navigate a mobile robot in an unstructured environment of any size. To achieve this challenging goal, we have decided that the navigation system should have the following properties:

- Scalability: Its computational complexity should be independent of the environment size.
- Simplicity: The method should be as simple as possible, because complex systems are more likely to contain errors.
- Swiftness: It has to satisfy real-time constraints.
- Standardness: It should use off-the-shelf equipment.
- Stability: The position uncertainty should not diverge with time.

The basic idea of the map-and-replay technique is similar to that of the industrial practice of programming stationary robots. One of the basic methods to program a stationary robot is by means of (tele)operation. A skilled operator guides the tip of the robot arm in order to perform a certain task (e.g., painting, welding). The robot records signals from its built-in receptors—typically incremental rotation sensors at its joints. During the robot operation, the recorded sequences serve as inputs for the robot's controllers. Though well established and efficient, this method is not applicable to mobile robots in unstructured environments due to the uncertainty in the robot–environment interaction. A typical example would be the use of odometry in a mobile robot localization—the uncertainty caused by wheel slippages tends to accumulate, which does not make odometry suitable for long-term localization and navigation. To effectively cope with the uncertainty in the robot's position, a mobile robot must use exteroreceptors to sense the surrounding environment. A mobile robot position and its heading can be estimated through measurements of the surrounding environment.

Several authors of SLAM algorithms acknowledge the fact that the uncertainty of robot heading is a crucial factor affecting the quality of the map and subsequently the quality of position estimation in the localization step. The influence of the heading estimation has been evaluated both

theoretically and practically (Frese, 2006). We extend the idea of heading estimation importance and claim that for long-term mobile robot localization it is sufficient to use exteroceptive sensors for heading estimation and the Cartesian coordinate estimation can be based just on proprioceptive sensors.

1.3. Paper Overview

A minimalistic approach to monocular localization and mapping is presented in this paper. We claim that for the navigation in a known environment, a robot needs a map just to estimate its heading and can measure its position by odometry. Formulating this particular instance of the navigation mathematically, we provide a formal proof of this claim. Furthermore, several large outdoor experiments confirm the expected system performance. We think that the most important contribution of our paper is not the presented method but the convergence proof presented in Section 3. The proof would apply to several other methods (Chen & Birchfield, 2009; Guerrero et al., 2005; Zhang & Kleeman, 2009) that use vision to correct heading and lateral position errors.

The rest of this paper is organized as follows. The proposed minimalistic navigation method is described in the next section. A mathematical model of this navigation method is outlined and its properties are examined in Section 3. A theorem that claims that this method prevents position uncertainty divergence is formulated and proven in the same section. The theoretical analysis is followed by a discussion on the practical issues of the navigation method. Experimental results verifying whether the system retains the expected properties are described in Section 5. A conclusion briefly discusses the properties of the proposed navigation method and outlines possible future improvements.

2. NAVIGATION SYSTEM DESCRIPTION

The proposed navigation procedure is based on the map-and-replay technique. The idea is simple: a robot is manually driven through an environment and creates a map of its surrounding environment. After that, the map is used for autonomous navigation. A similar technique for autonomous navigation based on computation of a robot steering from positions of remembered features has been described in Chen and Birchfield (2006), Royer et al. (2007), and Zhang and Kleeman (2009). To minimize the robot sensor equipment and to satisfy the "standardness" property mentioned in Section 1.2, we consider the most available sensors. The fundamental navigation property of a mobile vehicle is the traveled distance, which can be estimated by odometry. The odometric error is cumulative and therefore can be considered precise only in the short term. Another standard available sensor that does not require additional infrastructure is a compass. A fusion of data from the compass and odometry can provide position estimation but is still unsuitable for long-term navigation, because it lacks sufficient feedback from the robot's surrounding environment. To increase robot ability to sense the environment, one of the most advantageous sensors is a camera, which can provide lots of information.

Using these three main sensors, we have proposed the following simple navigation strategy:

- The robot is navigated along a sequence of straight line segments.
- At each segment start, the robot is turned to a direction according to the compass value.
- The steering control along the straight segment is computed from matched visual features providing a so-called visual compass.
- The end of each segment is recognized according to the traveled distance, which is measured by odometry.

The crucial component of the proposed navigation procedure is a map, which is created by guiding the robot along a path consisting of straight-line segments. Each segment has its own landmark map L_i, consisting of salient features detected in images captured by the robot's forward-looking camera, the initial robot orientation α and the segment length s. Once the map is created, the robot can travel autonomously within the mapped environment. During the navigation along a segment, the robot establishes correspondences of the currently seen and previously mapped landmarks and computes differences in the expected and recognized positions for each such correspondence. The robot steers in a direction that reduces those differences while moving straight at a constant speed until its odometry indicates that the current segment has been traversed. At the end of the segment, the robot switches to the next learned segment, turns to a direction of the initial orientation of the segment, and traverses the segment while keeping its direction according to matched features.

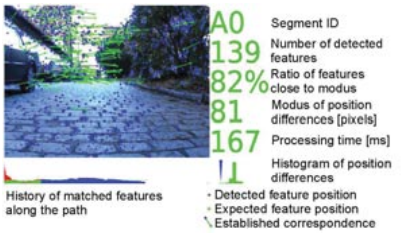
The next section describes the robot equipment and image processing. The algorithm for the map creation during the learning phase is described in Section 2.2, and the navigation algorithm is depicted in Section 2.3.

2.1. Robot Equipment

The proposed method has been verified on the P3AT robot with the Unibrain Fire-i601c camera, the TCM2 compass, and the HP 8710p laptop; see Figure 1(a). At first, the camera was equipped with a 7-mm objective with an electronically driven iris to prevent sunlight dazzle. The objective was replaced by a new one with a 4.5-mm focus length in 2008. At the same time, the electronically driven iris was substituted by a software exposure control. The laptop has Core2 Duo CPU running at 2.00 GHz and 1 GB of memory. Image processing is computationally demanding, and therefore the additional UPC70 battery had been used for longer experiments. To increase the robot action

(a) Robot configuration for experiments

(b) Image captured by robot camera and detected SURF positions



(c) Robot GUI with navigation phase data

Figure 1. Robot platform, detected features, and navigation graphical user interface (GUI).

radius, the three original batteries (connected in parallel) were replaced by one high-capacity battery and the robot's internal PC was disabled. The navigation system was implemented in C/C++ as a stand-alone Linux application.

The image processing algorithm is a critical component of the navigation system. The vision system must provide enough information to steer the robot in the right direction. Furthermore, it should be robust to real-world conditions, i.e., changing illumination, minor environment changes, and partial occlusions, and of course its performance should allow for a real-time response.

We have decided to use the speeded up robust features (SURF) (Bay, Tuytelaars, & Van Gool, 2006) method to identify landmarks in the image. The algorithm provides image coordinates of the salient features together with their descriptions. The SURF method is reported to perform better than most SIFT (Lowe, 1999) implementations in terms of speed and robustness to viewpoint and illumination changes. To achieve an additional speedup, the CPU implementation of the algorithm was adjusted to use both processor cores for parallel image processing. The captured image is horizontally divided, and the parts are processed in parallel. Later, we switched to the GPU (Cornelis & Van Gool, 2008) version of the algorithm. The GPU version has better real-time performance but is less distinctive than the CPU implementation (Svab, Krajnik, Faigl, & Preucil, 2009). Normally, the recognition of a 1,024 × 768 grayscale image provides descriptors of 150–300 features and takes 100–500 ms. The outdoor environment is usually richer in detected features, and image processing tends to be slower than indoors. A typical outdoor processed image with highlighted feature positions is shown in Figure 1(b).

2.2. Learning Phase

In the learning phase, the robot is manually guided through an environment in a turn-move manner and creates a map consisting of several straight segments. Each segment is described by its length s, its azimuth α, and a set of detected landmarks L. A landmark $l \in L$ is described by the tuple $(\mathbf{e}, k, \mathbf{u}, \mathbf{v}, f, g)$, where $\mathbf{e}$ is the SURF descriptor and k indicates the number of images in which the landmark was detected. Vectors $\mathbf{u}$ and $\mathbf{v}$ denote positions of the landmark in the captured image at the moment of its first and last detection, and f and g are the distances of the robot from the segment start in these moments.

The procedure that creates a map of one segment is shown in Algorithm 1. Before the robot starts to learn a segment, it reads compass data to establish the segment azimuth α and resets its odometric counters. After that, the robot starts to move forward, tracks detected features, and inserts them to the set L until the operator requests

Algorithm 1. Learn one segment

Input: α – an initial robot orientation (compass value)
Output: (α, s, L) – the data associated to the segment, where s is the traveled distance and L is a set of landmarks, a landmark is the tuple (k, e, u, v, f, g), where e is the SURF descriptor, k is a counter of feature detection, u and v are positions of the features in the image (at the moment of their first, resp. last occurrence), f and g denote distance from the segment start according to u, resp. v.

$L \leftarrow \emptyset$ // a set of learned landmarks
$T \leftarrow \emptyset$ // a set of tracked landmarks
$\alpha \leftarrow$ *compass value* // a robot orientation at the beginning of segment learning
repeat
  $d \leftarrow$ *current distance from the segment start*
  $S \leftarrow$ *extracted features with associated image position,* $(u, e) \in S$, *u position, e feature descriptor*
  foreach $\boldsymbol{t_i} = (e_i, k_i, u_i, v_i, f_i, g_i) \in T$ **do**
    $(u_a, e_a) \leftarrow \text{argmin}\{||e_i, e(s)|| \; | s \in S\}$ // select the best matching descriptor from S to e_i
    $(u_b, e_b) \leftarrow \text{argmin}\{||e_i, e(s)|| \; | s \in S \setminus \{(u_a, e_a)\}\}$ // select the next best matching descriptor
    if $||(e_i, e_a)|| \ll ||(e_i, e_b)||$ **then**
      $\boldsymbol{t_i} \leftarrow (e_i, k_i + 1, u_i, u_a, f_i, d)$ // update matched landmark
      $S \leftarrow S \setminus \{(u_a, e_a)\}$ // remove matched feature from the current set of detected features
    else
      $T \leftarrow T \setminus \{\boldsymbol{t_i}\}$ // remove $\boldsymbol{t_i}$ from the set of tracked landmarks
      $L \leftarrow L \cup \{\boldsymbol{t_i}\}$ // add $\boldsymbol{t_i}$ to the set of learned landmarks
  foreach $(u, e) \in S$ **do**
    $T \leftarrow T \cup \{(e, 1, u, u, d, d)\}$ // add new feature to the set of tracked landmarks
until *operator terminates learning mode*
$s \leftarrow d$ // the total traveled distance along the segment
$L \leftarrow L \cup T$ // add the current tracked landmarks to the set of learned landmarks

to stop. Images are continuously captured and processed during the movement. For each currently tracked landmark $\boldsymbol{t_i}$ (from the set T), two of the best matching features from the set of new features are found. If these two pairs are distinguishable enough (Bay et al., 2006), the best matching feature is associated to the tracked landmark, which is updated (values k, v, g). Each new feature is added to the set of tracked landmarks T, and its u and v are set to the value of the current distance from the segment start and the counter of the feature detection k is set to one. The segment description is saved at the end of the segment, and the operator can turn the robot to another direction and initiate mapping of a new segment. The format of the file, which stores the segment description, is shown in Table I.

2.3. Autonomous Navigation Mode

In the autonomous navigation mode, an operator enters a sequence of segments and indicates whether the robot should travel repeatedly. The robot is placed at the start of the first segment, loads the description of the segment, and turns itself to the segment azimuth and starts moving forward. The navigation procedure is shown in Algorithm 2. The relevant landmarks for the current robot position (i.e., according to the distance from the segment start) are selected from the set of the learned landmarks L. Correspondences between the mapped and the currently detected landmarks are established in the same way as in the learning phase. A difference in horizontal image coordinates of the features is computed for each such couple. A modus of those differences is estimated by the histogram voting method. The modus is converted to a correction value of the movement direction, which is reported to the robot's steering controller. After the robot travels a distance greater than or equal to the length of the given segment, the next segment description is loaded and the procedure is repeated. During the navigation, the robot displays the relevant states (mapped and recognized landmarks, recognition success ratio, etc.) on its graphical interface; see Figure 1(c).

An important aspect of this navigation algorithm is the fact that it does not need to explicitly localize the robot or to create a three-dimensional map of detected landmarks. It should also be noted that the proposed method is able to work in real time. Even though the camera readings are utilized only to correct the robot direction and the distance is measured by the imprecise odometry, the position uncertainty does not accumulate if the robot changes direction often enough. The stability of the proposed navigation method is discussed in the next section.

3. STABILITY OF BEARING-ONLY NAVIGATION

First, we describe in an informal way how the robot position uncertainty is changed as the robot travels a closed

Table I. A part of a segment map in a text file.

Record	Value	Meaning
Initial azimuth and length	2.13, 7.03	α, s
Landmark 0		
First position	760.74, 163.29	$\mathbf{u}_{\mathbf{l}_0}$
Last position	894.58, 54.44	$\mathbf{v}_{\mathbf{l}_0}$
Max visibility	128	k_{l_0}
First and last visible distance	0.00, 4.25	f_{l_0}, g_{l_0}
Descriptor	1, 0.116727, −0.000254, 0.000499, 0.000352, ...	$\mathbf{e}_{\mathbf{l}_0}$
Landmark 1		
First position	593.32, 381.17	$\mathbf{u}_{\mathbf{l}_1}$
Last position	689.89, 377.23	$\mathbf{v}_{\mathbf{l}_1}$
Max visibility	125	k_{l_1}
First and last visible distance	0.00, 6.73	f_{l_1}, g_{l_1}
Descriptor	−1, 0.070294, −0.006383, 0.012498, 0.006383, ...	$\mathbf{e}_{\mathbf{l}_1}$

Algorithm 2. Traverse one segment

Input: (α, s, L) – the data associated to the segment, where α is an initial angle of the robot orientation at the segment start, s is the traveled distance and L is a set of landmarks, a landmark is the tuple (e, k, u, v, f, g), where e is a SURF descriptor, k is a counter of feature detection, u and v are positions of feature in the image (at the moment of the first, resp. last, occurrence), f and g denote distances from the segment start according to u, resp. v.
Output: ω – a steering speed

```
turn(α)                                              // turn robot in the direction α
d ← current distance from the segment start
while d < s do
    T ← ∅                                            // a set of current tracked landmarks
    H ← ∅                  // a set of differences (horizontal position in the image) of matched features
    d ← current distance from the segment start
    S ← extracted features with associated image position, (u, e) ∈ S, u position, e feature descriptor
    foreach l_i = (e_i, k_i, u_i, v_i, f_i, g_i) ∈ L do
        if f_i ≥ d ≥ g_i then
            T ← T ∪ {l_i}                  // add landmark to the tracked landmarks according to the traveled distance
    while |T| > 0 do
        (e_i, k_i, u_i, v_i, f_i, g_i) ← argmax_{t∈T} k(t)        // get landmark with maximal number of occurrences k
        (u_a, e_a) ← argmin{||e_i, e(s)|| |s ∈ S}                 // select the best matching descriptor from S to e_i
        (u_b, e_b) ← argmin{||e_i, e(s)|| |s ∈ S \ {(u_a, e_a)}}          // select the next best matching descriptor
        if ||(e_i, e_a)|| ≪ ||(e_i, e_b)|| then
            p ← (v_i − u_i) (d − f_i)/(g_i − f_i) + u_i − u_a              // estimate angle to the matched landmark
            H ← H ∪ {p_x}                              // add horizontal difference to set of differences
        T ← T \ {(e_i, k_i, u_i, v_i, f_i, g_i)}                          // discard used landmark
    ω ← modus(H)                                     // determine new robot steering velocity
    report ω to steering controller
```

path. This should help to interpret the mathematical formalism describing the robot position uncertainty in geometrical terms and make the rest of this section more comprehensible. After that, we lay down a formal description of the proposed navigation method and analyze its stability. We outline a model of the robot movement and depict equations allowing the computation of the robot position uncertainty. Next, we use these equations to compute the robot position uncertainty for a closed path. Finally, we examine the properties of the proposed model and establish conditions ensuring that the robot position error does not diverge.

3.1. Geometrical Interpretation

Suppose that the learned path is a square and the robot has to travel it repeatedly. The robot is placed at a random [two-dimensional (2D) Gaussian distribution with zero mean] position near the first segment start; see Figure 2. The initial position uncertainty can therefore be displayed as a circle in which the robot is found with 90% probability. The navigation procedure is executed, and the robot starts to move along the first segment. Because it senses landmarks along the segment and corrects its heading, its lateral position deviation is decreased. However, owing to the odometric error, the longitudinal position error increases. At the end of the segment, the circle denoting position uncertainty becomes an ellipse, with a shorter axis perpendicular to the segment. Heading corrections are dependent on the value of the lateral deviation (see Section 3.2): the greater the deviation, the stronger the effect of heading corrections and therefore the lateral error decreases by a factor h for every traversed segment. The odometry error is independent of the current position deviation and is affected only by the length of the traversed segment, and therefore it can be modeled as an additive error o.

After the segment is traversed, the robot turns by 90 deg and starts to move along the next segment. The uncertainty changes again, but because of the direction change, the longer ellipse axis shrinks and the shorter is elongated due to the odometry error. This repeats for every traversed segment; the size of the uncertainty ellipse converges to a finite value. Because this particular trajectory is symmetric, axis lengths a, b of the "final" ellipse can be easily computed by the equations

$$\begin{aligned} a &= hb, \\ b &= a + o, \end{aligned} \tag{1}$$

where h is the coefficient of the lateral error reduction and o is the odometric error. The position error for $o = 1$ and $h = 0.25$ is shown in Figure 2. Though simple, this particular symmetric case gives us a basic insight into the problem. Now we will derive a broader mathematical model of the navigation, examine its properties, and show that the uncertainty does not diverge for nonsymmetrical trajectories as well.

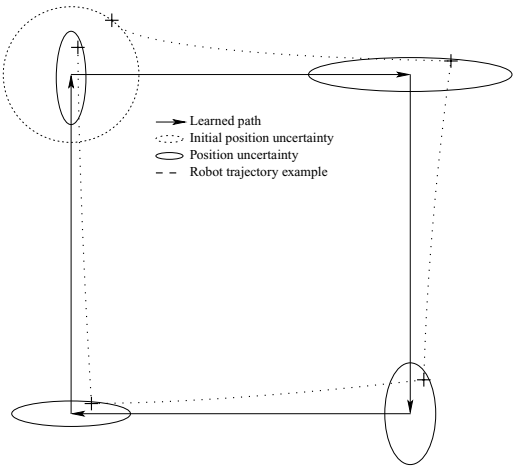


Figure 2. Position uncertainty evolution for a simple symmetric path.

3.2. Navigation

The proposed navigation method is based on the following assumptions:

- The robot moves in a plane.
- The map already exists in the form of a sequence of conjoined linear segments with landmark description.
- At least two segments of the mapped path are not collinear.
- The robot can recognize and associate a nonempty subset of mapped landmarks and determine their bearing.
- The robot can (imprecisely) measure the traveled distance by odometry.
- The camera is aimed forward, i.e., in the direction of the robot movement.

The path P consists of a sequence of linear segments p_i. The robot moves in a plane, i.e., its state vector is (x, y, φ). The robot we consider has a differential, nonholonomic drive, and therefore $\dot{x} = v\cos(\varphi)$ and $\dot{y} = v\sin(\varphi)$. For each segment p_i, there exists a nonempty subset of landmarks and a mapping between the robot position and the expected bearing of each landmark is established. At the start of each segment, the robot resets its odometry counter and turns approximately toward the segment end to sense at least one of the segment landmarks. The robot establishes correspondences of seen and mapped landmarks and computes differences in expected and recognized bearings. The robot steers in a direction that reduces these differences while moving forward until its odometry indicates that the current segment has been traversed.

Definition 1 (Closed-path stability property). *Assume that a robot navigates a closed path several times. Furthermore the robot is using an environment map only for heading corrections and measuring the distance by odometry. Then a path for which the robot position uncertainty does not diverge has the closed-path stability property.*

Theorem 1. *A path consisting of several conjoined segments retains the closed-path stability property if the assumptions in Section 3.2 are satisfied.*

3.3. Movement along One Segment

First, let us examine how a robot moves along one segment. We will focus on the position before and after traversing one segment and establish mapping from the robot position error at the segment start to the robot position error at the segment end.

To keep the model simple, we assume that the robot as well as the landmarks are positioned in a plane. We will consider having a map consisting of a single segment of length s with d landmarks, with positions represented as vectors $\mathbf{u_i}$. Because the robot is equipped with a forward-heading camera, learned landmark positions $\mathbf{u_i}$ are not assumed to be distributed uniformly along the path but rather shifted in the direction of the robot movement by a distance ρ. We can assume that $\rho \approx \frac{1}{d}\sum_{i=0}^{d-1} u_x i$, where d is the number of mapped landmarks. Let us place the segment start at the coordinate origin and the segment end at the position $[s, 0]^T$. We designate the robot position prior to the segment traversal as $\mathbf{a} = [a_x, a_y]^T$ and the final robot position as $\mathbf{b} = [b_x, b_y]^T$; see Figure 3. Let us assume that at every moment during the segment traversal, the robot recognizes a nonempty subset $\mathbf{W}$ of previously learned landmarks and the robot heads in a direction that minimizes the horizontal deviation of expected and recognized landmark positions. We denote the intersection of the robot heading with the learned segment axis as $\mathbf{w}$. At the beginning of the segment traversal, this position equals approximately $[\rho, 0]^T$ (i.e., $\mathbf{w} \approx [\rho, 0]^T$). As the robot traverses the segment, it loses sight of nearby landmarks and recognizes new ones. As new, more distant landmarks appear in the robot field of view and nearby landmarks disappear, the set $\mathbf{W}$ changes and the point $\mathbf{w}$ moves along the segment. It can be assumed that the point $\mathbf{w}$ moves approximately at the speed of the robot and therefore it is always ahead of the robot by the distance ρ.

Based on these premises, the robot position $[x, y]^T$ in terms of $y = f(x)$ can be established. The robot movement can be characterized by the following differential equation:

$$\frac{dx}{dy} = \frac{\rho}{-y}. \tag{2}$$

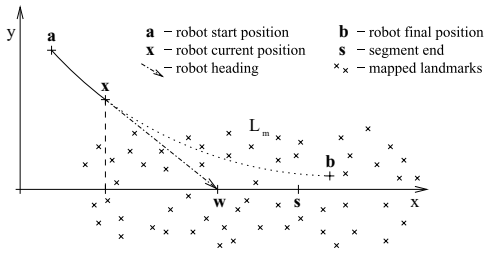


Figure 3. Robot movement model for a single path segment.

Solving Eq. (2) gives us a trajectory along which the robot moves:

$$y = c e^{-x/\rho}.$$

Considering a boundary condition $a_y = f(a_x)$, the constant c equals

$$c = \frac{a_y}{e^{-a_x/\rho}}.$$

Considering that the range of the robot's sensor is higher than the robot position uncertainty and that the segment length is higher than the robot lateral distance from the segment start (i.e., $\rho \gg a_x, s \gg |a_y|$), the constant c equals approximately a_y and the traveled distance is approximately equal to the segment length. Therefore, we can estimate the robot position after traveling a segment of length s by the following equations:

$$\begin{aligned} b_x &= a_x + s, \\ b_y &= a_y e^{-s/\rho}. \end{aligned} \tag{3}$$

We can transform Eqs. (3) to the matrix form

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} + \begin{pmatrix} s \\ 0 \end{pmatrix}. \tag{4}$$

Equation (4) is valid for an error-free odometry. If the odometry error is modeled as a multiplicative uncertainty, the equation changes to

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} + s \begin{pmatrix} 1+\upsilon \\ 0 \end{pmatrix}, \tag{5}$$

where υ is a random variable drawn from the Gaussian distribution with the zero mean and the variance ϵ. Accounting for the heading sensor noise, Eq. (5) changes to

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} + \begin{pmatrix} s+s\upsilon \\ \xi \end{pmatrix}, \tag{6}$$

where ξ is a random variable of the Gaussian distribution with the zero mean and the variance τ. Consolidating Eq. (6), we can state

$$\mathbf{b} = \mathbf{Ma} + \mathbf{s}.$$

The aforementioned movement model holds for a segment aligned with the x axis. For a segment with an arbitrary orientation α, the movement model becomes

$$\mathbf{b} = \mathbf{R^T M R a} + \mathbf{R^T s}, \tag{7}$$

where

$$\mathbf{R} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}, \qquad \mathbf{M} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & m \end{pmatrix}.$$

Equation (7) corresponds to aligning the segment with the x axis, applying $\mathbf{M}$, adding the odometric and the sensor

noise, and rotating the segment back to the direction α. In the following text, we use $\mathbf{N} = \mathbf{R^T M R}$, which shortens Eq. (7) to

$$\mathbf{b} = \mathbf{N a} + \mathbf{R^T s}. \tag{8}$$

All the aforementioned assumptions about the surrounding environment (landmark shift equal to ρ, $\rho \gg a_x$, etc.) can be relaxed as long as $m < 1$ for $s > 0$.

3.4. Position Uncertainty

Now, the dependence of the robot position uncertainty at the segment end to its uncertainty at the segment start can be examined. Consider that the robot position $\mathbf{a}$ before the segment traversal is a random variable drawn from a 2D normal distribution with the mean $\hat{\mathbf{a}}$ and the covariance matrix $\mathbf{A}$. To compute the robot position uncertainty after the segment traversal, we apply Eq. (8) to $\mathbf{a}$. Because the robot movement model in Eq. (8) has only linear and absolute terms, the robot position uncertainty after the segment traversal will constitute a normal distribution with the mean $\hat{\mathbf{b}}$ and the covariance matrix $\mathbf{B}$.

We denote $\mathbf{a} = \hat{\mathbf{a}} + \tilde{\mathbf{a}}$, where $\hat{\mathbf{a}}$ is the mean of $\mathbf{a}$ and $\tilde{\mathbf{a}}$ is a random variable of a normal distribution with the zero mean and the covariance $\mathbf{A}$. Similarly, we can denote $\mathbf{b} = \hat{\mathbf{b}} + \tilde{\mathbf{b}}$. Thus, we can rewrite Eq. (7) as follows:

$$\tilde{\mathbf{b}} = \mathbf{R^T M R}\tilde{\mathbf{a}} + \mathbf{R^T}\tilde{\mathbf{s}}.$$

We can claim that

$$\tilde{\mathbf{b}}\tilde{\mathbf{b}}^\mathbf{T} = (\mathbf{R^T M R}\tilde{\mathbf{a}} + \mathbf{R^T}\tilde{\mathbf{s}})(\mathbf{R^T M R}\tilde{\mathbf{a}} + \mathbf{R^T}\tilde{\mathbf{s}})^\mathbf{T}.$$

Because $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{a}}$ are independent and do not correlate,

$$\tilde{\mathbf{b}}\tilde{\mathbf{b}}^\mathbf{T} = \mathbf{R^T M R}\tilde{\mathbf{a}}\tilde{\mathbf{a}}^\mathbf{T}\mathbf{R^T M^T R} + \mathbf{R^T}\tilde{\mathbf{s}}\tilde{\mathbf{s}}^\mathbf{T}\mathbf{R},$$

which rewritten in terms of covariance matrices is

$$\mathbf{B} = \mathbf{R^T M R A R^T M^T R} + \mathbf{R^T S R}, \tag{9}$$

where

$$\mathbf{S} = \begin{pmatrix} s^2\epsilon^2 & 0 \\ 0 & \tau^2 \end{pmatrix}.$$

Equation (9) allows us to compute the robot position uncertainty after traversing one segment.

3.5. Traversing Multiple Segments

Let us consider a path consisting of n chained segments denoted by $i \in \{0, \ldots, n-1\}$ with the end of the last segment equal to the start of the first segment, i.e., the considered path is closed. We denote length and orientation of the ith segment as s_i and α_i. The robot position before and after traversing the ith segment is noted as $\mathbf{a_i}$ and $\mathbf{b_i}$. Because the robot position at the end of the ith segment equals its start position at the segment $i+1$, we can state that $\mathbf{a_{i+1}} = \mathbf{b_i}$.

The movement model (9) for the ith traveled segment is

$$\mathbf{A_{i+1}} = \mathbf{B_i} = \mathbf{R_i^T M_i R_i A_i R_i^T M_i^T R_i} + \mathbf{R_i^T S_i R_i}. \tag{10}$$

Considering $\mathbf{N_i} = \mathbf{R_i^T M_i R_i}$ and defining $\mathbf{T_i} = \mathbf{R_i^T S_i R_i}$, we can rewrite Eq. (10) as

$$\mathbf{A_{i+1}} = \mathbf{N_i A_i N_i^T} + \mathbf{T_i}.$$

One can compute the robot position uncertainty in terms of the covariance matrix after traversing i path segments in the following terms:

$$\mathbf{A_i} = \left(\prod_{\mathbf{j=i-1}}^{\mathbf{0}} \mathbf{N_j}\right) \mathbf{A_0} \left(\prod_{\mathbf{j=0}}^{\mathbf{i-1}} \mathbf{N_j^T}\right) + \sum_{\mathbf{j=0}}^{\mathbf{i-1}} \left[\left(\prod_{\mathbf{k=i-1}}^{\mathbf{j}} \mathbf{N_k}\right) \mathbf{N_j^{-1} T_j} \left(\mathbf{N_j^T}\right)^{-1} \left(\prod_{\mathbf{k=j}}^{\mathbf{i-1}} \mathbf{N_k^T}\right) \right]. \tag{11}$$

To examine how the robot position uncertainty changes after the robot travels the entire learned path i times, we define $\mathbf{C_i} = \mathbf{A_{in}}$ (e.g., $\mathbf{C_1} = \mathbf{A_n}$). Moreover, we denote

$$\check{\mathbf{N}} = \prod_{j=n-1}^{0} \mathbf{N_j}$$

and

$$\check{\mathbf{T}} = \sum_{\mathbf{j=0}}^{\mathbf{n-1}} \left[\left(\prod_{\mathbf{k=n-1}}^{\mathbf{j}} \mathbf{N_k}\right) \mathbf{N_j^{-1} T_j} \left(\mathbf{N_j^T}\right)^{-1} \left(\prod_{\mathbf{k=j}}^{\mathbf{n-1}} \mathbf{N_k^T}\right) \right] \tag{12}$$

and rewrite Eq. (11) as

$$\mathbf{C_{i+1}} = \check{\mathbf{N}}\mathbf{C_i}\check{\mathbf{N}}^\mathbf{T} + \check{\mathbf{T}}. \tag{13}$$

By proving that $\mathbf{C_i}$ converges to a finite matrix as i grows to infinity, we prove Theorem 1.

3.6. Convergence Conditions

Expression (13) is the Lyapunov discrete equation (Lyapunov, 1992). If all eigenvalues of $\check{\mathbf{N}}$ lie within a unit circle and $\check{\mathbf{T}}$ is symmetric, then $\lim_{i\to\infty} \mathbf{C_i}$ is finite and equal to $\mathbf{C}_\infty$, which can be obtained by solution of the algebraic equation

$$\mathbf{C}_\infty = \check{\mathbf{N}}\mathbf{C}_\infty\check{\mathbf{N}}^\mathbf{T} + \check{\mathbf{T}}. \tag{14}$$

Because the matrix $\mathbf{S_i}$ is symmetric, $\mathbf{T_i} = \mathbf{R_i^T S_i R_i}$ is also symmetric. The product $\mathbf{X T_i X^T}$ is symmetric for any $\mathbf{X}$ and therefore all addends in Eq. (12) are symmetric. Addition

preserves symmetry and therefore the matrix

$$\check{\mathbf{T}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n-1}} \left[\left(\prod_{\mathbf{k}=\mathbf{n-1}}^{\mathbf{j}} \mathbf{N_k} \right) \mathbf{N_j^{-1}} \mathbf{T_j} \left(\mathbf{N_j^T} \right)^{-1} \left(\prod_{\mathbf{k}=\mathbf{j}}^{\mathbf{n-1}} \mathbf{N_k^T} \right) \right]$$

is symmetric.

To prove that the eigenvalues of $\check{\mathbf{N}}$ lie within a unit circle, we exploit the positiveness of the matrices $\mathbf{M_i}$ and $\mathbf{R_i}$. Because $\mathbf{M_i}$ is positive, $\mathbf{N_i} = \mathbf{R_i^T M_i R_i}$ is also positive. Moreover, as every $\mathbf{N_i}$ equals $\mathbf{R_i^T M_i R_i}$, its eigenvalues are the same as those of $\mathbf{M_i}$ and eigenvectors are columns of $\mathbf{R_i}$. The eigenvalues of $\mathbf{N_i}$ therefore correspond to one and e^{-s_i/ρ_i}. Because the product $\mathbf{XY}$ of a positive definite matrix $\mathbf{X}$ and a symmetric positive definite matrix $\mathbf{Y}$ is positive definite, the matrix $\check{\mathbf{N}}$ is positive definite as well. Moreover, the dominant (maximal) eigenvalue of the product $\mathbf{XY}$ is lower than or equal to xy, where x and y are dominant eigenvalues of $\mathbf{X}$ and $\mathbf{Y}$. Because the dominant eigenvalue of every $\mathbf{N_i}$ is one, all eigenvalues of $\check{\mathbf{N}}$ are smaller than or equal to one. The dominant eigenvalue of $\check{\mathbf{N}}$ is equal to one if and only if the dominant eigenvalue of the product $\mathbf{N_{i+1}N_i}$ equals 1 for all i. Conditions satisfying that the eigenvalues of a product $\mathbf{N_{i+1}N_i}$ are lower than one ensure the existence of a finite solution of Eq. (14). Therefore, we have to find those conditions to support the closed-path stability property.

3.7. Convergence Proof

We will exploit the fact that a product of matrix eigenvalues equals the matrix determinant and the sum of eigenvalues equals matrix trace. Let us denote eigenvalues of the matrix product $\mathbf{N_{i+1}N_i}$ as $\lambda_{0,1}$ and the smaller eigenvalue of $\mathbf{N_i}$ as n_i ($n_i = e^{-s_i/\rho_i}$). For our convenience, we denote $j = i + 1$. Therefore

$$\det\left(\mathbf{N_j N_i}\right) = \det \mathbf{N_j} \det \mathbf{N_i} = \lambda_0 \lambda_1 = n_i n_j. \tag{15}$$

If $\lambda_{0,1} \in \langle 0, 1 \rangle$, we can state that

$$(1 - \lambda_0)(1 - \lambda_1) \geq 0, \tag{16}$$

and therefore

$$\lambda_0 \lambda_1 - \lambda_0 - \lambda_1 + 1 \geq 0. \tag{17}$$

Combining Eq. (15) and inequality (17), we obtain

$$1 + n_i n_j \geq \lambda_0 + \lambda_1.$$

Considering that the sum of eigenvalues equals matrix trace, we get

$$\operatorname{trace}\left(\mathbf{R_j^T M_j R_j R_i^T M_i R_i}\right) \leq 1 + n_i n_j. \tag{18}$$

Because trace(**AB**) is equal to trace(**BA**), we can rewrite inequality (18) as

$$\operatorname{trace}\left(\mathbf{M_j}\left(\mathbf{R_i R_j^T}\right)^{\mathbf{T}} \mathbf{M_i R_i R_j^T}\right) \leq 1 + n_i n_j. \tag{19}$$

Both matrices $\mathbf{R_i}$ and $\mathbf{R_j}$ represent rotations. The matrix $\mathbf{R_i}$ denotes rotation by the angle α_i, and $\mathbf{R_j}$ denotes rotation by the angle α_j. Their product $\mathbf{R_i R_j^T}$ denotes rotation by $\alpha_i - \alpha_j$. If we denote $\beta = \alpha_i - \alpha_j$ and $\mathbf{R}_\beta = \mathbf{R_i R_j^T}$, inequality (19) is changed to

$$\operatorname{trace}\left(\mathbf{M_j R_\beta^T M_i R_\beta}\right) \leq 1 + n_i n_j. \tag{20}$$

By expanding matrices $\mathbf{M_j R_\beta^T}$, we obtain

$$\mathbf{M_j R_\beta^T} = \begin{pmatrix} \cos\beta & -n_j \sin\beta \\ \sin\beta & n_j \cos\beta \end{pmatrix}$$

and

$$\mathbf{M_i R}_\beta = \begin{pmatrix} \cos\beta & n_i \sin\beta \\ -\sin\beta & n_i \cos\beta \end{pmatrix}.$$

Inequality (20) can be rewritten to

$$(1 + n_i n_j)\cos^2\beta + (n_i + n_j)\sin^2\beta \leq 1 + n_i n_j$$

and further reduced to

$$1 + n_i n_j - (1 - n_i - n_j + n_i n_j)\sin^2\beta \leq 1 + n_i n_j.$$

Finally, we get

$$(1 - n_i)(1 - n_j)\sin^2\beta \geq 0. \tag{21}$$

Because $n_i = e^{-s_i/\rho_i}$, $n_i \in (0, 1)$ and $n_j \in (0, 1)$, and inequality (21) is strict for $\sin\beta \neq 0$. This fact implies that inequality (16) is strict as well, which means that both λ_0 and λ_1 are lower than one. Therefore both eigenvalues of the matrix product $(\mathbf{N_i N_j})$ are smaller than one if $\beta \neq n\pi | n \in \mathcal{N}$. The matrix $\check{\mathbf{N}}$ has both eigenvalues smaller than one if and only if at least two conjoined segments of the path form an angle different from 0 or π.

Because all eigenvalues of $\check{\mathbf{N}}$ lie within a unit circle and $\check{\mathbf{T}}$ is symmetric, the covariance matrix $\mathbf{C}_\infty$ denoting the robot position uncertainty at the start of the first path segment is finite and obtainable by a solution of the algebraic equation

$$\mathbf{C}_\infty = \check{\mathbf{N}} \mathbf{C}_\infty \check{\mathbf{N}}^{\mathbf{T}} + \check{\mathbf{T}}.$$

□

3.8. Convergence Proof Overview

We have established Eqs. (7) describing the movement of a robot using a navigation method, which is described in Section 2. Equation (10) allowed us to examine the robot position uncertainty evolution as the robot travels through a known environment. Modifying Eq. (10) to closed trajectories, we could rewrite it as Eq. (13). By examining the conditions under which Eq. (13) has a finite solution, we have proven Theorem 1 for closed paths that have at least two noncollinear segments. The existence of a finite solution of Eq. (13) means that if a mobile robot traverses repeatedly a

closed polygonal path using our method, its position error at every point of the traversed trajectory will stabilize at a certain value.

4. PRACTICAL ISSUES

The theoretical proof of convergence stands on several assumptions, which might not always be met. In this section, we will outline possible issues that might arise from incorrect assumptions and discuss their impact on navigation stability and accuracy. Moreover, we will discuss method requirements in terms of computational power, memory, and disk storage.

4.1. Convergence Proof from an Engineer's Point of View

Though elegant and useful, mathematical models and formal proofs are often based on a simplification of reality. A good mathematical model picks out the essence of the modeled problem and concentrates on an examination of the model properties. Some properties of the real system are not included in the model and therefore are not considered. An experienced engineer must be aware of these properties and realize the difference between math and reality. This applies to the proof presented in Section 3 as well.

In practice, extremely elongated (imagine a rectangle with sides 1,000 and 0.001 m long) paths will not retain the closed-path stability property because the movement along the shorter side will not compensate odometry errors accumulated over the long side. Moreover, the model does not cover the probability that the robot will not establish enough correct correspondences because its position error would grow too high. This might easily happen in case the robot has to avoid large obstacles as well as for paths with very long segments.

Moreover, the fact that the position error does not diverge might not be really useful in real robotic systems. In practice, the real precision is more important. The precision can be estimated using Eq. (13) if the learned path shape, landmark distribution ρ, camera noise τ, and odometry noise ϵ are known. Using $\rho = 20$ (i.e., most of the sensed landmarks are 20 m in front of the robot), the sensor noise $\tau = 0.1$, and the odometry noise $\epsilon = 0.0005$ for a square, 1-km-long path, the predicted navigation repeatability (i.e., computed from eigenvalues of $\mathbf{C}_\infty$) is 0.15 m. This value is in good accordance with the experimental results presented in Section 5, where the measured repeatability in outdoor scenarios was 0.14 m.

4.2. Reliance on Odometry

Odometry is regarded as unsuitable for long-term localization due to cumulative errors. Its error model is usually multiplicative with a precision around 1%. The error of the odometric pose estimation is caused mainly by the fact that the robot heading cannot be properly determined. On the other side, odometry can be very precise for traveled distance measurements. Moreover, an odometric error is usually systematic, which can be solved by precise calibration. Our experience with the P3AT robot shows that repeatability of its odometric measurements of the traveled distance on paved roads is better than 0.1%. This means that in the case of precise heading estimation, the robot would be able to travel 1 km with a position error lower than 1 m.

Our approach relies on the fact that the robot changes direction often enough. If the robot would travel in a straight direction for a long distance, its position error might grow beyond an acceptable level. This might be avoided either by forcing the robot to change directions during the learning phase or by complementing the distance measurement by methods without a long-term drift. An example of such a method might be global positioning system or a vision-based localization used in methods in Chen and Birchfield (2009), Royer et al. (2007), and Zhang and Kleeman (2009).

4.3. False Correspondences

The most troublesome issue is that correct correspondences might not be established. However, our algorithm works even in cases of a large number of outliers. Consider a situation in which the system is navigating and all of its established correspondences are false. The horizontal position deviation of detected and mapped features would be basically a random variable. Therefore a histogram H, which is built in order to establish the robot turning speed, will have its bins (approximately) equally filled. The robot turning speed will therefore be random. Now consider that there are a few correctly established correspondences. Each correctly established correspondence increases the value of the bin, which corresponds to the robot's true heading deviation. Therefore the probability that the correct bin has a maximal value increases with each correct correspondence.

This is different from the work presented in Chen and Birchfield (2009) and Segvic et al. (2007), where the authors choose a mean of horizontal differences instead of the modus. The modus is more invariant to the incorrect correspondences than the mean, which makes our method more precise and robust.

In reality, we get 80%–90% correctly established correspondences if the navigation phase follows mapping immediately. As the map gets older, the ratio of correct correspondences tends to drop. The rate of "map decay" depends on the environment and is caused mainly by two factors: short-term lighting changes caused by a change in the position of the sun and the current weather conditions and long-term environment changes caused by seasonal factors. Both illumination and long-term changes are not so significant in indoor environments, because lighting is typically artificial and seasonal changes do not happen. So it

is expected that illuminations and seasonal changes would play an important role in outdoor environments.

To evaluate the system robustness to lighting changes, we made an all-day experiment in which the robot traversed a 1-km-long path in an outdoor environment; see Section 5.5. To evaluate our system robustness to seasonal environment changes, we mapped a 50-m-long path in a park. The path was autonomously navigated and then relearned one month later. This was done in five consecutive months; see Section 5.4. The results of both experiments show that the system is robust to both long-term and short-term environment changes.

Dynamic objects and occlusions cause only a temporary and slight decrease in the ratio of correctly established correspondences. During the experiments, we did not notice any problems with moving objects in the robot field of view.

4.4. Obstacle Avoidance

The proposed navigation method itself does not include obstacle avoidance. However, it can be complemented by a collision avoidance module, which takes control of the robot whenever an obstacle in the robot's course is detected. Such a module guides the robot around the obstacle until the area between the robot and its path is clear again. After that, the visual-based navigation takes control and guides the robot by Algorithm 2.

Because obstacles have finite dimensions, the robot position error will grow by a finite value every time it passes an obstacle. From the theoretical point of view, random obstacles in the robot path can be modeled by the addition of a random vector with a zero mean to $\mathbf{s}$ in Eq. (8). Although the addition will increase the matrix $\mathbf{S}$ in Eq. (9), the symmetry of $\mathbf{S}$ will be preserved. Obstacles would therefore increase the matrix $\check{\mathbf{T}}$ in Eqs. (13) and (14), but because $\check{\mathbf{T}}$ remains symmetric, Eq. (14) will have a unique solution. However, the robot position uncertainty, represented by the matrix $\mathbf{C}_\infty$, will increase. Therefore, obstacle avoidance would decrease the precision of the robot navigation, but it should remain stable. This assumption is experimentally verified in Section 5.3.

It is clear that there exists a size of obstacles for which the algorithm will fail, because after circumnavigating the obstacle, the robot will not find previously mapped features.

4.5. Systematic Errors

Because the navigation algorithm relies on two sensors, there are two sources of systematic errors in our algorithm: the odometry and the camera.

The systematic error of the odometry means that if the robot traverses the distance d, it will report that the traveled distance is $d(1+\eta)$. Let us consider that the robot has 900% odometric error, i.e., $\eta = 9$. During mapping phases, the error will cause the segment lengths s and landmark data f and g to be 10 times higher in the map than in reality. However, in the navigation phases, the odometry error will give 10 times higher values of the robot distance from the segment start, and therefore errors in the map and robot position error will suppress each other.

The systematic error of the camera might be caused by the misalignment of the camera optical axis and the robot body. This causes the positions of landmarks $\mathbf{u}$, $\mathbf{v}$ in the map to be different from a case with an ideal camera. However, when the robot encounters the same location, detected landmark positions will be shifted the same way as in the learning phase. The systematic error will therefore cancel out as in the previous case.

A different case would be a change of the odometric error η or the camera angle θ between the learning and navigation phases. From a theoretical point of view, this would cause a change of the vector $\mathbf{s}$. Unlike in the previous case, the vector $\mathbf{s}$ will not be modified by a random vector but a fixed one. This means that $\tilde{\mathbf{s}}$ will remain the same and matrix $\check{\mathbf{T}}$ will preserve symmetricity. Systematic errors would cause the robot to traverse a trajectory slightly different from the learned one but should not affect navigation stability. However, the algorithm will fail to establish correct correspondences between the mapped and detected features if the systematic errors are too high.

The experimental evaluation of the influence of systematic errors on the navigation stability is described in Section 5.2. Even though the systematic errors were set to high values during the experimental evaluation, the navigation stability was preserved.

4.6. Necessity of a Compass

Relying on only a compass for heading estimation was shown to be a weak point during experiments performed in 2008 and 2009. During learning phases, the compass noise caused an incorrect azimuth estimation of some segments.

Therefore, we considered replacing the absolute azimuth measurements by relative ones. So, instead of recording α_i for the ith segment in the learning phase, the azimuth relative to the previous segment (i.e., $\Delta_i = \alpha_i - \alpha_{i-1}$) is recorded in the map. The relative azimuth Δ_i can be estimated either by odometry or by tracking of the features during transitions to the next segment. This approach is applicable in cases in which a robot is taught a path that is supposed to be traversed later. However, sometimes it is necessary to create a more complex, graph-like map; see Section 5.7.

In more complex cases, the robot creates a map of the large environment in several mapping runs and traverses the given sequence of segments in an order different from the mapped sequence. In this case, sole knowledge of the relative segment azimuths is not sufficient, because angles between nonconsecutive segments are not known to us. Of course an angle between the ith and $(i+j)$th segments can

be estimated by summing all relative angles between segments i and $i + j$, but because every Δ_i contains a small error, the error of the sum is too large for high j.

To deal with these more complex cases, we have implemented a simple Kalman filter, which fuses data from odometry and compass. The filter suppresses the compass noise and causes the absolute heading measurements to be more reliable. However, the filter was implemented at the end of 2009, so in the previous experiments the compass noise caused trouble.

4.7. Computational and Storage Requirements

To estimate computational and storage requirements, we have used data from the experiment described in Section 5.5.

We evaluated the computational requirements in terms of required computational time spent in various stages of the algorithm. The most computationally intensive stage of the algorithm is the feature extraction, which takes 260 ms on average. About 30 ms is taken by establishing proper correspondences. The histogram voting time is less than 1 ms. During experiments, the camera image and additional parameters of the algorithm were displayed for debugging purposes. Drawing these data on a computer screen takes about 60 ms. Thus, the entire control loop takes about 350 ms.

The average landmark density is about 140 landmarks per meter of the path. The map is stored on the hard drive in a text format (see Table I), and one landmark occupies about 800 bytes. Therefore, the disk storage needed for 1 km of path is about 112 MB. Once loaded to the computer memory, a landmark is represented in binary and occupies less than 300 bytes. Thus, a segment 1 km long would take 42 MB of computer memory.

5. EXPERIMENTS

The assumptions formed in Sections 3 and 4 were verified in several real-world experiments. The experimental evaluation was performed in seven different scenarios examining the following:

1. Convergence for two types of paths–with and without the closed-path stability property
2. The impact of systematic errors to the navigation precision
3. Feasibility of complementing the method by the collision avoidance module
4. Robustness to environment changes and variable lighting conditions
5. Performance in environments with landmark deficiency
6. Navigation efficiency for long paths in an outdoor environment with diverse terrain
7. Real deployment of the navigation procedure in RoboTour 2008 and RoboTour 2009 contests (Iša & Dlouhý, 2010)

During these scenarios, the robot autonomously traversed more than 3 km of indoor and more than 25 km of outdoor paths. The P3AT platform with the configuration described in Section 2 was used in all testing scenarios.

The robot learned different closed paths and was requested to navigate these paths several times in the first six scenarios. The relative position $\mathbf{c_i}$ of the robot to the path start was measured every time the robot completed the ith path loop. To evaluate the quality of the navigation algorithm, accuracy and repeatability values as in Chen and Birchfield (2009) were used. The accuracy ε_{acc} and the repeatability ε_{rep} are computed as the rms of the Euclidean distance or the standard deviation of the robot's final positions from the path start:

$$\varepsilon_{\text{acc}} = \sqrt{\frac{1}{n-j}\sum_{i=j}^{n}\|\mathbf{c_i}\|^2}, \quad \varepsilon_{\text{rep}} = \sqrt{\frac{1}{n-j}\sum_{i=j}^{n}\|\mathbf{c_i}-\mu\|^2}, \tag{22}$$

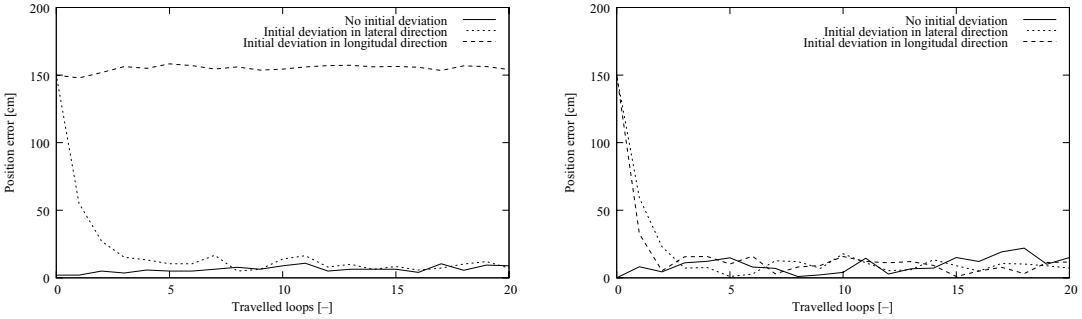
where $\mathbf{c_i}$ is the robot position relative to the path start after completing the ith loop and $\mu = \sum_{i=j}^{n}\mathbf{c_i}/(n-j)$. In most scenarios, the initial robot position was intentionally changed to be 1.5 m apart from the learned path start. In these cases, we do not set j to 1 but wait five loops until the initial position error diminishes. Thus, the repeatability and the accuracy are computed for $j = 5, n = 20$ in the first four scenarios.

5.1. Stability of Robot Position for Different Types of Paths

The scenario examines Theorem 1 for paths with and without the closed-path stability property. The conclusions made in Section 3 indicate that paths with all collinear segments do not retain the closed-path stability property, whereas other paths do. The following paths have been considered: a path with only two collinear segments (i.e., a "back and forth" line path) and a square path. At first, the robot was taught these closed paths in an indoor hall. After that, the robot was placed either directly at the path start or 1.5 m away and requested to navigate along the learned path 20 times. The robot position $\mathbf{c_i}$ was measured after each completed loop.

The first (degenerate) path was formed of two collinear, 5-m-long, segments. The square path was composed of four segments, each 5 m long, with the end of the last segment identical to the segment at the start of the path. The distance of the robot from the path start after each loop (i.e., $\|\mathbf{c_i}\|$) is shown in Figure 4.

The values indicate that for square trajectories, the robot was able to correct the position error that was introduced at the beginning of navigation along the learned path. Only was the error in the y coordinate (i.e., the coordinate axis normal to path segments) partially corrected for collinear trajectories, while the x coordinate remained



(a) The position error for line (degenerate) path (b) The position error for square (normal) path

Figure 4. The position error for paths without and with the stability property.

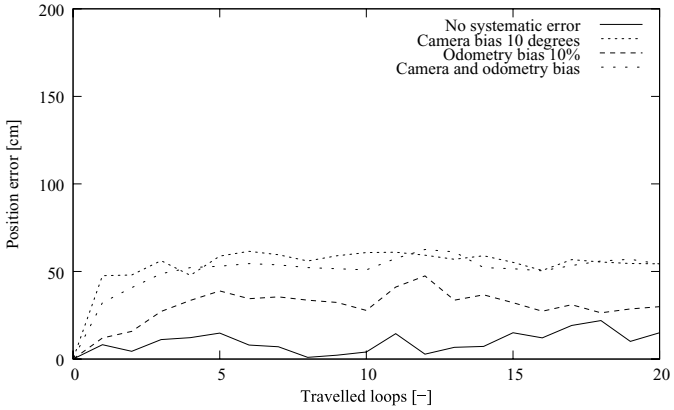


Figure 5. Systematic errors effect: the position error for different types of systematic errors.

uncorrected. These experimental results confirm the theoretical assumptions described in Section 3.8, stating that the navigation is unstable for paths with only collinear segments.

The robot traversed more than 1.8 km in this scenario. Both the accuracy and the repeatability for the 20-m-long square paths were 0.10 m.

5.2. Effect of Systematic Errors

The effect of the systematic errors on navigation precision was evaluated in this scenario. Two sources of systematic errors were considered: the camera and the odometry. An error of the camera can be caused by its optical axis deviation, and an odometric error can be caused by a tire pressure change. To show the effect of the parameter change, it is necessary to modify these parameters between the learning and the navigation phases; otherwise a path is learned with the systematic errors, and therefore the errors do not have an effect.

The following experiments were performed to verify that small-scale systematic errors do not affect navigation stability. At first, the robot camera was panned by 10 deg, and the robot was requested to traverse the square path learned during scenario 5.1 20 times. Then, a 10% systematic odometry error was introduced.[1] Finally, the robot was requested to traverse the path 20 times with both 10% odometry and 10-deg camera bias. As in the previous cases, the robot position $\mathbf{c_i}$ was measured each time the robot reached the learned path start. The measured distances from the path start are shown in Figure 5.

[1]This was done in software—a distance of the robot from the segment start measured by odometry was multiplied by a factor of 1.1 before it was passed to the navigation algorithm.

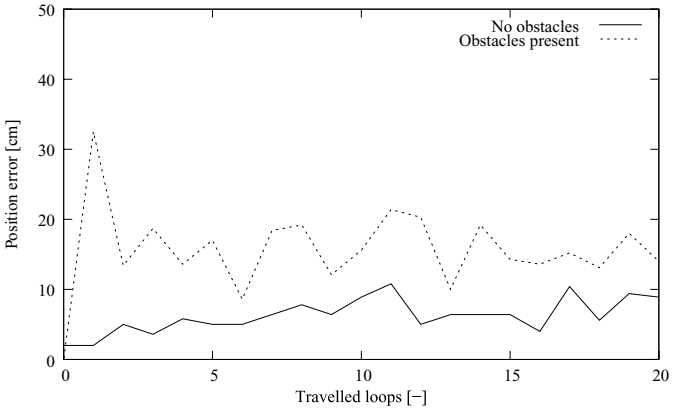


Figure 6. Obstacle avoidance experiment: the position errors with and without obstacles.

The results show that the odometric and the camera biases cause errors in robot positioning but the error does not diverge. After a few loops, the position error does not grow anymore and the system reaches a steady state. The overall accuracy is lower than without the systematic errors, but repeatability remains similar to the previous scenarios.

The robot traversed more than 1.2 km in this scenario. The average accuracy with the camera bias was 0.58 m, and the odometry bias caused the accuracy to change to 0.34 m. When both the odometry and the camera were biased, the accuracy was 0.55 m. The average repeatability was lower than in the previous scenario, i.e., 0.06 m.

5.3. Obstacle Avoidance

A simple collision avoidance module (based on the robot sonars) was activated in this scenario. The collision avoidance is based on the Tangent Bug algorithm with a finite range sensor (Choset, Lynch, Hutchinson, Kantor, Burgard, et al., 2005). When the robot detects an obstacle on its course, the visual-based navigation is suppressed and the robot starts to circumnavigate the detected obstacle. During the circumnavigation, odometry is used to determine the robot position and the sonar data are used to estimate the obstacle position. The visual navigation algorithm is resumed when the path between the robot and the end of the current segment is clear.

The robot was taught a square path similar to the one used in scenario 5.1. After that, one obstacle[2] was placed on each path segment, and the robot navigated the path 20 times. The robot autonomously navigated approximately 0.4 km with an accuracy of 0.16 m and a repeatability of 0.08 m. It is clear that the position precision was affected but did not diverge. See Figure 6.

[2]Obstacle dimensions were approximately half of the robot size.

5.4. Environment and Lighting Changes

The effects of variable lighting conditions and long-term environment changes were examined in this scenario. At first, the robot was taught a closed, 50-m-long path consisting of five segments in the Stromovka park located in the city of Prague. One month later, the robot was placed 1.5 m away from the path start and was requested to navigate the path 20 times. This procedure was repeated five times, i.e., the test was done every month from November 2009 until April 2010. In each experiment, the robot used a map created in a previous month. The measured distances are shown in Figure 7.

Not only did the lighting conditions differ every time but also the environment went through seasonal changes. To document these changes, a picture from the onboard camera was stored every time the mapping was initiated; see Figure 8. There were considerably fewer correct correspondences between recognized and learned features. With a map created just before the navigation, the robot usually correctly recognizes 70%–90% of learned landmarks. Using a 1-month-old map, the ratio of the correctly recognized landmarks drops to 10%–40%. Nevertheless, the robot was able to correct its initial position error and was able to traverse the path faultlessly in all cases.

The robot autonomously navigated more than 6 km with an average accuracy of 0.24 m in this scenario. Unlike in previous scenarios, we did not measure the robot position $\mathbf{c_i}$ after completion of each loop; we recorded the robot distance only from the path start, i.e., $\|\mathbf{c_i}\|$. Therefore, the repeatability cannot be calculated by Eqs. (22). Except for winter months, pedestrians regularly crossed the robot path and moved into the robot's field of view.

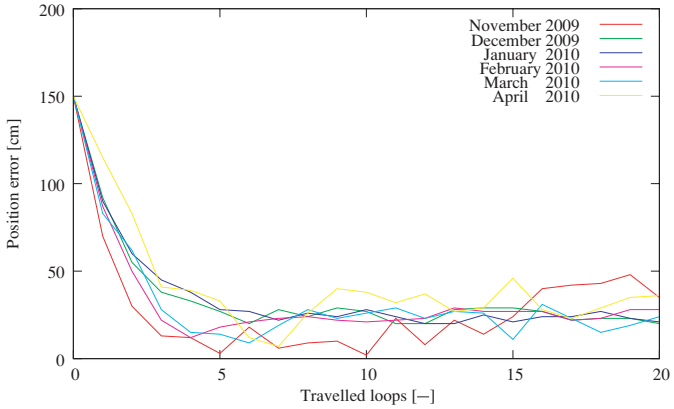
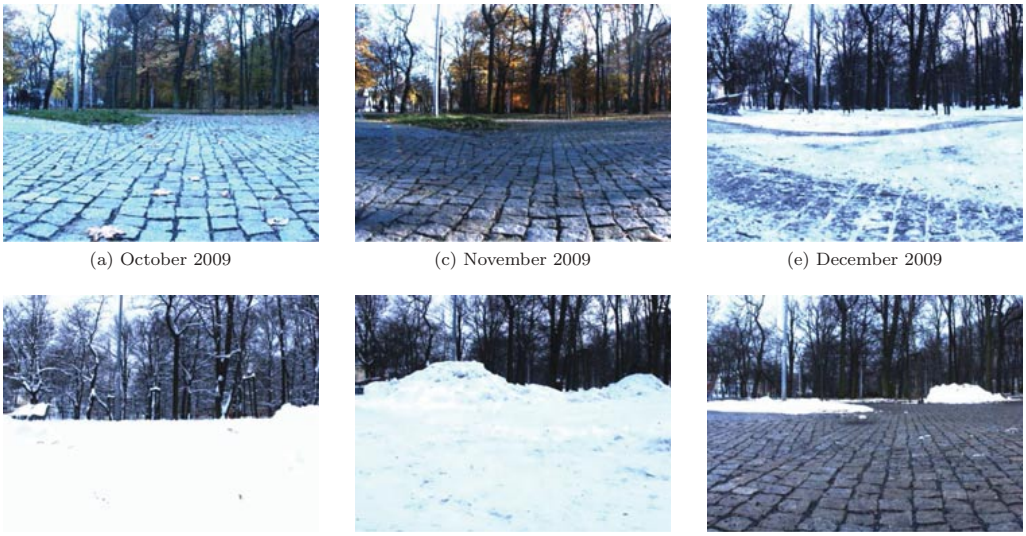


Figure 7. The position errors in different months of the long-term experiment.



(a) October 2009 (c) November 2009 (e) December 2009

(b) January 2010 (d) February 2010 (f) March 2010

Figure 8. Long-term experiment: the view from the robot's camera at the path start in different months.

5.5. One-Day Outdoor Experiment

The system performance for long paths was evaluated in a realistic outdoor environment. The experiment was performed around the Proboštov pond[3] in Proboštov, Czech Republic, at the end of March 2010. The robot was taught a 1-km-long path around the pond in the morning. The path went through a variable nonflat terrain with asphalt paths, dirt roads, footpaths, and grass terrain in an approximately equal ratio (see Figure 9). After the path was taught, the robot was placed 1.5 m away from the path start and requested to traverse it repeatedly. Every time it reached the path start, its position was measured and its batteries replaced (the robot was not moved during the battery exchange). It took approximately 1 h for the robot to traverse the learned path, and the battery replacement took 15 min. The weather changed from cloudy/light rain to partly cloudy/sunny during the experiment. In the afternoon, a lot of pedestrians showed up and either entered the robot's field of view or crossed its path.

Nevertheless, the robot was able to complete the learned path six times before nightfall. The robot traversed

[3] 50°39′58.716″N, 13°50′18.35″E.

Figure 9. One-day experiment: the path around Proboštov pond and the dirt road terrain example.

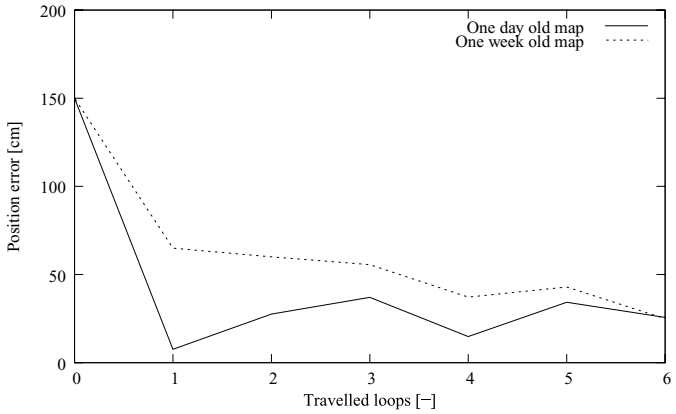


Figure 10. The position errors of the one-day experiment.

6 km with an accuracy[4] of 0.26 m and a repeatability of 0.14 m. The experiment was repeated (without the learning phase) 1 week later, and the robot traversed the path six times with an accuracy of 0.31 m and a repeatability of 0.20 m. The measured distances are shown in Figure 10.

5.6. Landmark Deficiency Experiment

We have claimed that the system is able to operate in an environment that contains a low number of landmarks. To verify this assumption, we taught the robot an outdoor path during night and let it navigate using only streetlamp lights. The robot was taught a 0.3-km-long path on paved roads in a residential area. The onboard camera iris was fully opened, and the camera exposure time was set to 0.37 s. The path was taught at midnight, so more than 90% of the mapped landmarks were streetlamps and illuminated windows.

After the path was learned, the robot was placed 1.5 m from the path start and requested to traverse it 10 times. As opposed to in the daytime experiments, in which the robot detected typically 150–300 landmarks, during the nighttime, the typical number of landmarks was 3. The robot traversed 3 km with an accuracy[5] of 0.32 m and a repeatability of 0.16 m. See Figure 11.

5.7. The RoboTour Outdoor Delivery Challenge

The RoboTour contest (Dlouhy & Winkler, 2009; Iša & Dlouhý, 2010) is an international autonomous robot delivery challenge organized by robotika.cz. The participating

[4]In this case, ε_{acc} and ε_{rep} were computed with $j = 3$ and $n = 6$ in Eqs. (22).

[5]In this case, ε_{acc} and ε_{rep} were computed with $j = 3$ and $n = 10$ in Eqs. (22).

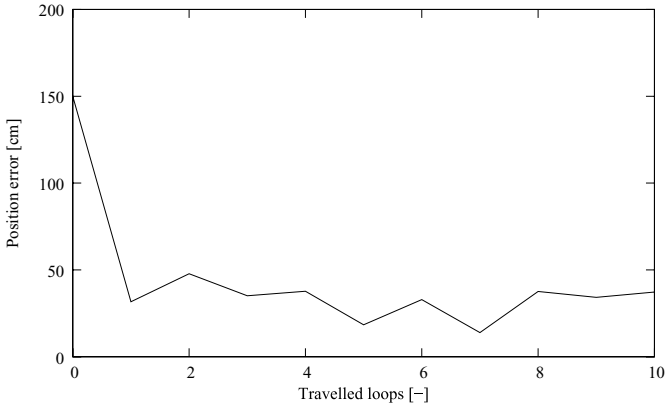
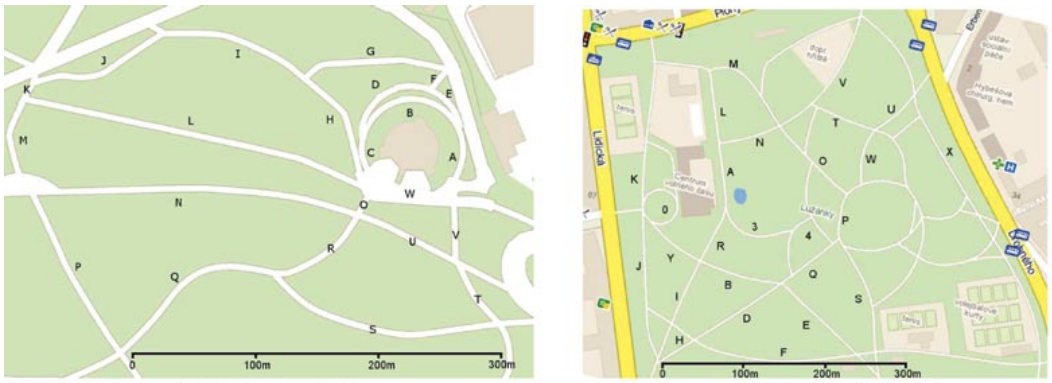


Figure 11. The position errors of the landmark deficiency (night) experiment.



(a) RoboTour 2008 (b) RoboTour 2009

Figure 12. RoboTour 2008/2009 pathway maps.

teams are mostly from Czech and Slovak universities. The competition is a perfect event for an independent verification of system functions and comparison with other navigation methods. However, not only are the navigation methods evaluated, but also the complete systems including all hardware parts are tested.

Fully autonomous robots have to travel a random path in a park, stay on the pavements, and detect randomly placed obstacles in this challenge. A map of the park with its pathways (designated by letters) is given to the teams in advance. The competition consists of several rounds, each with a different path. Thirty minutes before each round, referees choose a random closed path and announce it as a sequence of letters. Competing teams place their robots at the starting positions and execute their autonomous navigation algorithms. Robots must travel without leaving the pathway and without colliding with any random obstacles. The robot score is determined according to its traveled distance. In 2008, the competition was held in Stromovka park[6] in Prague, Czech Republic. One year later, the contest moved to park Lužánky[7] in Brno, Czech Republic.

The Stromovka park pathways were mapped 2 days prior to the competition. The competition had five rounds with different pathways; see Table II and Figure 12(a). The robot completed the required path four times of these five attempts. During two of these attempts, the robot did not leave the pathway at all, and during two others, the robot had partially left (i.e., with two side wheels) the pathway. In

[6] 50°6′18.778″N, 14°25′33.395″E.
[7] 49°12′25.516″N, 16°36′29.81″E.

Table II. RoboTour 2008 and 2009 paths.

Pathway sequence length (m)			
RoboTour 2008		RoboTour 2009	
ABCW	250	ALK0JIR	800
ORQPMKJIH	850	BESQDGHJ0Y	1,050
ABCHGFDCW	500	34PWTMLA	750
HGFEAWOUVW	600	0YR34SFHJ	600
UTSROCDEBCW	700		
Total	2,900	Total	2,200

cases when the robot left the pathway partially, it was left to continue moving (without additional scores) and reached the goal area. One failed attempt was caused by a battery failure.

The competition in Lužánky park was performed in a larger part of the environment; hence the mapping took 3 days. The total length of the mapped pathways was 8 km; the map consisted of approximately one million landmarks, which took 834 MB of disk space. The competition had four rounds; see Table II and Figure 12(b). The robot was able to complete the required paths two times. One attempt failed due to a wrong compass reading during the path learning, but after a manual correction of the robot heading, the robot caught up and reached the goal area. The other failed attempt was caused by a human factor.

Although the performance was not perfect, the robot was able to travel the required trajectory, and our team reached the first rank for both events in 2008 and 2009.

5.8. Experiment Summary

The results of the aforementioned experiments not only confirm theoretical assumptions stated in Section 3, but they also show that our method is feasible for use in real-world conditions. The proposed method is able to cope with diverse terrain, dynamic objects, obstacles, systematic errors, variable lighting conditions, and seasonal environment changes. The summary of experiments in Table III indicates that the localization precision of our method is slightly worse than in closely related methods presented in Royer et al. (2007) and Zhang and Kleeman (2009). Lower precision is probably caused by heavy reliance on odometry and suboptimal use of visual information.

Compared to the similar method presented in Chen and Birchfield (2009), our method accuracy and repeatability is better in outdoor environments. A probable reason for this is that we used a modus to determine robot heading. Chen and Birchfield (2009) use a mean of horizontal deviations, which is less robust to data association errors.

6. CONCLUSION

A simple navigation method based on bearing-only sensors and odometry was presented. In this method, a robot navigating a known environment uses a map of the environment and a camera input to establish its heading, while measuring the traveled distance by odometry. We claim that this kind of navigation is sufficient to keep the robot position error limited. This claim is formulated as a closed-path stability property and proved for polygonal paths with at least two noncollinear segments. The property allows us to estimate the robot position uncertainty based on the landmark density, robot odometry precision, and path shape.

The proposed method was experimentally verified by a mobile robot with a monocular camera. The robot builds a SURF-based (Bay et al., 2006; Cornelis & Van Gool, 2008) landmark map in a guided tour. After that, it uses the aforementioned method to autonomously navigate in the mapped environment.

We conducted experiments indicating that theoretical results and assumed conditions are sound.

The proposed navigation method has surprising properties different from the properties of other navigation and localization methods, mainly the following:

- The robot can perform 2D localization by heading estimation, which is a one-degree-of-freedom method.
- If the robot travels between two points, it is better to use a "zigzag" trajectory rather than a straight one.
- Traveling a closed trajectory might reduce the robot position uncertainty.

Table III. Proposed method accuracy and repeatability in various scenarios.

	Indoor		Outdoor		
	Clear	Obstacles	Long term	1 day	Night
Accuracy (m)	0.10	0.16	0.24	0.25	0.32
Repeatability (m)	0.10	0.08	N/A	0.14	0.16
Loop length (m)	20	20	50	1,040	330

We believe that the convergence proof does not apply only to our system but is valid for many other algorithms. The proof suggests that any algorithm that decreases the lateral position error of a robot is stable for closed polygonal trajectories. This might be the case for even simpler and faster methods, such as the one presented in Zhang and Kleeman (2009). However, this is merely a hypothesis, which needs to be thoroughly examined.

The fundamental limitation of our method is its reliance on odometric measurements. Other visual-based navigation methods use odometry only as an auxiliary measurement or do not require odometry at all. Therefore, these methods would perform better in scenarios in which wheel slippages have to be taken into account. Although our method is limited in application and its precision is lower compared to methods presented in Royer et al. (2007) and Zhang and Kleeman (2009), we believe that it is interesting from an academic point of view.

In the future, we would like to test our algorithm with robots that have only imprecise odometry or inaccurate dead reckoning. The preliminary tests conducted with the AR-Drone quadrotor helicopter, which estimates the traveled distance by accelerometers, seem to be promising.

7. APPENDIX A: INDEX TO MULTIMEDIA EXTENSIONS

The video is available as Supporting Information in the online version of this article.

Extension	Media type	Description
1	Video	Algorithm 1 implemented on a UAV

8. APPENDIX B

This Appendix presents values measured during experiments.

8.1. Stability of Robot Position for Different Types of Paths

Table B.I contains data measured during the first experimental scenario presented in Section 5.1. It shows the measured positions for the paths that do and do not retain the stability property. Note that line (degenerate) paths do not correct the longitudal position deviation, which is in accordance with the convergence proof presented in Section 3.

Table B.I. Convergence for degenerate and normal paths: indoors.

	Position relative to start (m)					
	Line (degenerate) path			Square (normal) path		
Loop	$d_{0,0}$	$d_{0,1.5}$	$d_{1.5,0}$	$d_{0,0}$	$d_{0,1.5}$	$d_{1.5,0}$
00	0.00, 0.00	0.00, 1.50	1.50, 0.00	0.00, 0.00	0.00, 1.50	1.50, 0.00
01	−0.02, 0.00	0.05, 0.55	1.46, −0.24	0.08, −0.02	0.12, 0.58	0.32, −0.06
02	−0.03, −0.04	−0.03, 0.27	1.49, −0.30	0.02, 0.04	0.02, 0.23	0.05, −0.02
03	−0.03, −0.02	−0.06, 0.14	1.53, −0.32	−0.02, 0.11	0.04, 0.06	0.10, 0.12
04	−0.05, 0.03	−0.03, 0.13	1.53, −0.25	0.10, 0.07	−0.07, −0.03	0.05, 0.15
05	−0.05, 0.00	−0.03, 0.10	1.56, −0.27	0.10, 0.11	−0.01, 0.00	0.05, 0.09
06	−0.04, 0.03	−0.03, 0.10	1.55, −0.25	0.08, 0.00	−0.02, 0.02	0.00, −0.16
07	−0.05, 0.04	−0.05, 0.16	1.53, −0.22	0.01, 0.07	0.04, −0.12	−0.03, 0.00
08	−0.05, 0.06	−0.05, 0.00	1.54, −0.25	0.00, −0.01	0.05, 0.11	0.07, 0.04
09	−0.04, 0.05	−0.06, 0.02	1.53, −0.15	−0.01, 0.02	0.05, −0.05	0.07, 0.06
10	−0.04, 0.08	−0.05, 0.13	1.53, −0.21	0.00, −0.04	−0.08, −0.16	0.09, 0.13
11	−0.06, −0.09	−0.04, 0.16	1.54, −0.25	−0.04, −0.14	0.02, −0.11	0.12, 0.02
12	−0.05, 0.01	−0.04, −0.07	1.54, −0.31	0.02, −0.02	0.05, 0.02	0.02, −0.11
13	−0.05, 0.04	−0.08, 0.06	1.55, −0.27	0.03, −0.06	0.00, 0.06	−0.01, −0.12
14	−0.04, 0.05	−0.06, 0.02	1.53, −0.31	0.06, 0.04	−0.09, −0.10	0.02, −0.09
15	−0.05, −0.04	−0.06, 0.06	1.55, −0.21	−0.01, −0.15	−0.01, 0.09	0.01, 0.00
16	−0.04, 0.00	−0.05, −0.03	1.54, −0.24	−0.02, −0.12	−0.05, 0.01	0.01, 0.05
17	−0.03, 0.10	−0.07, 0.02	1.52, −0.21	−0.03, −0.19	0.07, 0.08	0.05, −0.06
18	−0.04, 0.04	−0.09, −0.05	1.55, −0.24	0.02, −0.22	0.09, 0.05	0.01, 0.03
19	−0.03, 0.09	−0.11, 0.05	1.56, −0.10	−0.02, −0.10	0.08, 0.04	0.07, 0.09
20	−0.04, 0.08	−0.07, 0.02	1.54, −0.07	0.01, −0.15	0.02, 0.07	0.04, −0.11

Appendix 6 - Krajník, T. - Faigl, J. - Vonásek, V. - Košnar, K. - Kulich, M. - et al.: *Simple, Yet Stable Bearing-Only Navigation* [52], referenced on page 21.

Table B.II. Effect of systematic errors on navigation convergence.

	Position relative to start (m) Systematic error (bias)			
Loop	None	Camera	Odom.	Both
00	0.00, 0.00	0.00, 0.00	0.00, 0.00	0.00, 0.00
01	0.08, −0.02	−0.30, 0.37	0.08, 0.09	−0.16, 0.28
02	0.02, 0.04	−0.28, 0.39	0.09, 0.13	−0.15, 0.38
03	−0.02, 0.11	−0.35, 0.44	0.13, 0.24	−0.16, 0.46
04	0.10, 0.07	−0.29, 0.38	0.15, 0.30	−0.15, 0.50
05	0.10, 0.11	−0.31, 0.50	0.17, 0.35	−0.03, 0.53
06	0.08, 0.00	−0.33, 0.52	0.13, 0.32	−0.24, 0.49
07	0.01, 0.07	−0.34, 0.49	0.19, 0.30	−0.14, 0.52
08	0.00, −0.01	−0.32, 0.46	0.13, 0.31	−0.18, 0.49
09	−0.01, 0.02	−0.33, 0.49	0.12, 0.30	−0.13, 0.50
10	0.00, −0.04	−0.36, 0.49	0.10, 0.26	−0.14, 0.49
11	−0.04, −0.14	−0.35, 0.50	0.13, 0.39	−0.33, 0.47
12	0.02, −0.02	−0.32, 0.50	0.18, 0.44	−0.39, 0.49
13	0.03, −0.06	−0.35, 0.45	0.13, 0.31	−0.38, 0.48
14	0.06, 0.04	−0.33, 0.49	0.14, 0.34	−0.18, 0.49
15	−0.01, −0.15	−0.32, 0.45	0.14, 0.29	−0.13, 0.50
16	−0.02, −0.12	−0.31, 0.40	0.11, 0.25	−0.12, 0.49
17	−0.03, −0.19	−0.36, 0.44	0.11, 0.29	−0.16, 0.51
18	0.02, −0.22	−0.31, 0.46	0.11, 0.24	−0.15, 0.54
19	−0.02, −0.10	−0.35, 0.42	0.12, 0.26	−0.21, 0.53
20	0.01, −0.15	−0.32, 0.44	0.13, 0.27	−0.12, 0.53

8.2. Effect of Systematic Errors

Table B.II contains data from the experimental scenario in Section 5.2, in which effects of the camera and the odometry bias were measured.

8.3. Obstacle Avoidance

Table B.III contains data from the experiment scenario described in Section 5.3, in which we verified the methods ability to deal with obstacles.

8.4. Environment and Lighting Changes

Table B.IV contains data from the experiment scenario described in Section 5.4, in which the algorithm was tested in an outdoor environment with long-term environment changes.

8.5. One-Day Outdoor Experiment

Table B.V contains data from experiment scenario 5.5, in which the algorithm was tested in an outdoor environment with variable terrain.

Table B.III. Positioning errors with and without obstacles.

	Position relative to start (m)	
Loop	Clear path	Obstacles
00	0.00, 0.00	0.00, 0.00
01	0.08, −0.02	0.24, 0.22
02	0.02, 0.04	0.12, 0.06
03	−0.02, 0.11	0.17, 0.08
04	0.10, 0.07	0.11, −0.08
05	0.10, 0.11	0.15, −0.08
06	0.08, 0.00	−0.05, −0.07
07	0.01, 0.07	0.14, −0.12
08	0.00, −0.01	0.15, −0.12
09	−0.01, 0.02	0.02, −0.12
10	0.00, −0.04	0.14, −0.07
11	−0.04, −0.14	0.17, −0.13
12	0.02, −0.02	0.20, −0.04
13	0.03, −0.06	0.08, −0.06
14	0.06, 0.04	0.19, −0.03
15	−0.01, −0.15	0.14, −0.03
16	−0.02, −0.12	0.13, 0.04
17	−0.03, −0.19	0.15, −0.03
18	0.02, −0.22	0.02, −0.13
19	−0.02, −0.10	0.17, −0.06
20	0.01, −0.15	0.14, −0.01

Table B.IV. Long-term algorithm reliability.

	Distance to path start (m)					
Loop	Nov	Dec	Jan	Feb	Mar	Apr
00	1.50	1.50	1.50	1.50	1.50	1.50
01	0.70	0.92	0.90	0.87	0.83	1.15
02	0.30	0.55	0.60	0.50	0.62	0.83
03	0.13	0.38	0.45	0.22	0.28	0.41
04	0.12	0.33	0.38	0.12	0.15	0.39
05	0.03	0.27	0.28	0.18	0.14	0.33
06	0.18	0.20	0.27	0.21	0.09	0.12
07	0.06	0.28	0.22	0.23	0.19	0.07
08	0.09	0.24	0.26	0.24	0.28	0.26
09	0.10	0.29	0.24	0.22	0.23	0.40
10	0.02	0.27	0.28	0.21	0.26	0.38
11	0.23	0.20	0.24	0.22	0.29	0.32
12	0.08	0.20	0.20	0.23	0.23	0.37
13	0.22	0.28	0.20	0.29	0.27	0.27
14	0.14	0.29	0.25	0.27	0.26	0.29
15	0.24	0.29	0.21	0.27	0.11	0.46
16	0.40	0.27	0.24	0.27	0.31	0.28
17	0.42	0.22	0.24	0.22	0.23	0.23
18	0.43	0.23	0.27	0.23	0.15	0.29
19	0.48	0.23	0.23	0.28	0.19	0.35
20	0.45	0.20	0.21	0.28	0.24	0.36

Table B.V. One-day outdoor experiments.

	Position relative to start (m) by map age	
Loop	Up to date	1 week
00	0.00, 1.50	0.00, 1.50
01	−0.07, 0.03	0.63, 0.15
02	−0.21, 0.18	0.59, 0.11
03	−0.34, 0.15	0.53, 0.17
04	−0.05, −0.14	0.32, 0.19
05	0.34, 0.05	−0.09, 0.22
06	−0.25, 0.06	0.15, 0.20

Table B.VI. Landmark deficiency experiment.

Loop	Position relative to start (m)
00	0.00, 1.50
01	−0.13, −0.29
02	0.21, −0.43
03	−0.09, −0.34
04	−0.32, −0.20
05	0.12, −0.14
06	−0.08, −0.32
07	−0.05, −0.13
08	−0.29, −0.24
09	−0.09, −0.33
10	−0.10, −0.36

8.6. Landmark Deficiency Experiment

Table B.VI contains data from the experiment scenario described in Section 5.6, in which the algorithm was tested during night in low-visibility conditions.

ACKNOWLEDGMENTS

We would like to thank our colleagues for valuable remarks and friends for help with the outdoor tests. The work presented in this paper has been supported by the Ministry of Education of the Czech Republic under projects MSM 6840770038 and 2C06005, by CTU research grant SGS10/185/OHK3/2T/13, and by the FP7-ICT project "REPLICATOR" No. 216240.

REFERENCES

Bay, H., Tuytelaars, T., & Van Gool, L. (2006, May). SURF: Speeded up robust features. In Proceedings of the Ninth European Conference on Computer Vision, Graz, Austria.

Blanc, G., Mezouar, Y., & Martinet, P. (2005, April). Indoor navigation of a wheeled mobile robot along visual routes. In Proceedings of International Conference on Robotics and Automation, Barcelona, Spain.

Bosse, M., Newman, P., Leonard, J. J., & Teller, S. (2004). SLAM in large-scale cyclic environments using the Atlas framework. International Journal of Robotics Research, 23(12), 1113–1139.

Burschka, D., & Hager, G. (2001, May). Vision-based control of mobile robots. In Proceedings International Conference on Robotics and Automation, Seoul, Korea.

Chaumette, F., & Hutchinson, S. (2006). Visual servo control, part I: Basic approaches. IEEE Robotics and Automation Magazine, 13(4), 82–90.

Chaumette, F., & Hutchinson, S. (2007). Visual servo control, part II: Advanced approaches. IEEE Robotics and Automation Magazine, 14(1), 109–118.

Chen, Z., & Birchfield, S. T. (2006, May). Qualitative vision-based mobile robot navigation. In 2006 (ICRA), IEEE International Conference on Robotics and Automation (ICRA), Orlando, Florida (pp. 2686–2692).

Chen, Z., & Birchfield, S. T. (2009). Qualitative vision-based path following. IEEE Transactions on Robotics and Automation, 25(3), 749–754.

Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E., & Thrun, S. (2005). Principles of robot motion: Theory, algorithms, and implementations. Cambridge, MA: MIT Press.

Civera, J., Davison, A. J., & Montiel, J. (2008). Inverse depth parametrization for monocular SLAM. IEEE Transactions on Robotics, 24(5), 932–945.

Clemente, L. A., Davison, A. J., Reid, I. D., Neira, J., & Tardós, J. D. (2007). Mapping large loops with a single hand-held camera. In W. Burgard, O. Brock, and C. Stachniss (Eds.), Robotics: Science and systems. Cambridge, MA: MIT Press.

Cornelis, N., & Van Gool, L. (2008, June). Fast scale invariant feature detection and matching on programmable graphics hardware. In CVPR 2008 Workshop (June 27th), Anchorage AK.

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6), 1052–1067.

DeMenthon, D., & Davis, L. S. (1992, May). Model-based object pose in 25 lines of code. In ECCV '92: Proceedings of the Second European Conference on Computer Vision, Santa Margherita Ligure, Italy.

Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. IEEE Transactions on Robotics and Automation, 17(3), 229–241.

Dlouhy, M., & Winkler, Z. (2009). Robotour outdoor delivery challenge. http://robotika.cz/competitions/en. Accessed June 26, 2010.

Estrada, C., Neira, J., & Tardós, J. D. (2005). Hierarchical SLAM: Real-time accurate mapping of large environments. IEEE Transactions on Robotics, 21(4), 588–596.

Frese, U. (2006). A discussion of simultaneous localization and mapping. Autonomous Robots, 20(1), 25–42.

Gibbens, P., Dissanayake, G., & Durrant-Whyte, H. (2000, December). A closed form solution to the single degree

Appendix 6 - Krajník, T. - Faigl, J. - Vonásek, V. - Košnar, K. - Kulich, M. - et al.: *Simple, Yet Stable Bearing-Only Navigation* [52], referenced on page 21.

of freedom simultaneous localisation and map building (SLAM) problem. In Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia (pp. 191–196).

Guerrero, J. J., Martinez-Cantin, R., & Sagüés, C. (2005). Visual map-less navigation based on homographies. Journal of Robotic Systems, 22(10), 569–581.

Holmes, S., Klein, G., & Murray, D. W. (2008, May). A square root unscented Kalman filter for visual monoSLAM. In 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA (pp. 3710–3716).

Iša, J., & Dlouhý, M. (2010, September). Robotour—Robotika.cz outdoor delivery challenge. In Proceedings of the 1st Slovak–Austrian International Conference on Robotics in Education, Bratislava, Slovakia (to appear).

Julier, S., & Uhlmann, J. (2001, May). A counter example to the theory of simultaneous localization and map building. In 2001 IEEE International Conference on Robotics and Automation, Seoul, Korea (pp. 4238–4243).

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. Transactions of the ASME—Journal of Basic Engineering, 82(Series D), 35–45.

Kidono, K., Miura, J., & Shirai, Y. (2000, July). Autonomous visual navigation of a mobile robot using a human-guided experience. In Proceedings of 6th International Conference on Intelligent Autonomous Systems, Venice, Italy (pp. 620–627).

Lowe, D. G. (1999, September). Object recognition from local scale-invariant features. In ICCV '99: Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece (p. 1150).

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91–110.

Lyapunov, A. (1992). The general problem of stability in motion. International Journal of Control, 55(3), 531–773.

Martinelli, A., Tomatis, N., & Siegwart, R. (2005, August). Some results on SLAM and the closing the loop problem. In International Conference on Intelligent Robots and Systems, Edmonton, Canada (pp. 334–339).

Matsumoto, Y., Inaba, M., & Inoue, H. (1996, April). Visual navigation using view-sequenced route representation. In Proceedings of the International Conference on Robotics and Automation, Minneapolis, MN.

Montiel, J., Civera, J., & Davison, A. (2006, August). Unified inverse depth parametrization for monocular SLAM. In Proceedings of Robotics: Science and Systems, Philadelphia, PA.

Mourikis, A., & Roumeliotis, S. I. (2004, September). Analysis of positioning uncertainty in simultaneous localization and mapping (SLAM). In Proceedings of the IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS), Sendai, Japan.

Remazeilles, A., & Chaumette, F. (2007). Image-based robot navigation from an image memory. Robotics and Autonomous Systems, 55(4), 345–356.

Royer, E., Lhuillier, M., Dhome, M., & Lavest, J.-M. (2007). Monocular vision for mobile robot localization and autonomous navigation. International Journal of Computer Vision, 74(3), 237–260.

Segvic, S., Remazeilles, A., Diosi, A., & Chaumette, F. (2007, June). Large scale vision based navigation without an accurate global reconstruction. In IEEE International Conference on Computer Vision and Pattern Recognition, Minneapolis, MN.

Segvic, S., Remazeilles, A., Diosi, A., & Chaumette, F. (2009). A mapping and localization framework for scalable appearance-based navigation. Computer Vision and Image Understanding, 113(2), 172–187.

Svab, J., Krajnik, T., Faigl, J., & Preucil, L. (2009, November). FPGA-based speeded up robust features. In 2009 IEEE International Conference on Technologies for Practical Robot Applications, Boston, MA.

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., & Tardós, J. (2009). A comparison of loop closing techniques in monocular SLAM. Robotics and Autonomous Systems, 57(12), 1188–1197.

Wilson, W., Hulls, C., & Bell, G. (1996). Relative end-effector control using Cartesian position based visual servoing. Robotics and Automation, 12(5), 684–696.

Zhang, A. M., & Kleeman, L. (2009). Robust appearance based visual route following for navigation in large-scale outdoor environments. International Journal of Robotics Research, 28(3), 331–356.

Appendix 7 - Faigl, J. - Krajník, T. - Vonásek, V. - Přeučil, L.: *Surveillance Planning with Localization Uncertainty for UAVs* [30], referenced on page 22.

Surveillance Planning with Localization Uncertainty for UAVs

Jan Faigl*, Tomáš Krajník, Vojtěch Vonásek, Libor Přeučil
Department of Cybernetics, Faculty of Electrical Engineering
Czech Technical University in Prague
{xfaigl,tkrajnik,vonasek,preucil}@labe.felk.cvut.cz, *corresponding author

***Abstract*— This paper presents a new multi-goal path planning method that incorporates the localization uncertainty in a visual inspection surveillance task. It is shown that the reliability of the executed found plan is increased if the localization uncertainty of the used navigation method is taken into account during the path planning. The navigation method follows the map&replay technique based on a combination of monocular vision and dead-reckoning. The mathematical description of the navigation method allows efficient computation of the evolution of the robot position uncertainty that is used in the proposed path planning algorithm. The algorithm minimizes the length of the inspection path while the robot position error at the goals is decreased. The effect of the decreased localization uncertainty is examined in several scenarios.**

The presented experimental results indicate that probability of the goals visits can be increased by the proposed algorithm. Thus, the proposed approach opens further research directions in the increasing reliability of the autonomous navigation by the path planning using efficient and sufficiently informative heuristics of the localization error evolution.

I. Introduction

The problem of autonomous navigation of a mobile robot is addressed by various localization methods in order to achieve sufficiently reliable navigation. The methods use various techniques based on different assumptions and environment constraints. One of the popular technique is the so-called SLAM in which no prior information about the environment is known and the localization is performed on the basis of the simultaneously created map. Even though a significant progress has been made in this field, more reliable localization techniques use a priori known map of the robot surrounding environment. Moreover, the authors of SLAM algorithms tends to consider only the immediate localization error, and it is not exceptional that robots are navigated manually during SLAM examination. On the other side of the navigation methods, higher precision and reliability (in a long term) is achieved at the cost of the previously created map. In this sense, reliable navigation methods are based on the visual servoing techniques using the map&replay scenario [1], [2].

Beside improvements of the localization methods, the reliability of the autonomous navigation may be increased by consideration of environment properties and the localization/navigation method in the preparation of the plan/path for the navigation. The idea is simple: the robot identifies areas, where the localization method would be too imprecise and avoids these places. Although the idea is simple, the problem is not easily tractable as it depends on appropriate (realistic) environment models. Moreover the problem domain has to be extended by the "uncertainty" dimensions (e.g. pose×uncertainties [3]) that increase the computational complexity of the planning methods. Models of the uncertainty evolution have to be sufficiently informative otherwise the intended plan will be more likely useless. For example a model of the increasing odometry error simply leads to minimization of the planned path length, but it does not provide a way to "correct" the robot pose estimation. Therefore such model cannot be efficiently used in a long-term planning. The models based on performing a simulation of the localization within the map of the robot surrounding environment are too computationally intensive and therefore approximations have to be used. However, such simplified models may lead to violate the required stability assumptions of the localization methods [4], [5], [6].

In this paper, we consider a heuristic function describing the evolution of the localization uncertainty in a surveillance path planning problem that deals with visiting a set of areas of interest (AoIs). The function is derived from the model of a simple navigation principle [7] that uses the map&replay technique. The principle is based on a detection of salient objects and dead-reckoning measurements, e.g. an odometry, and it is similar to [8], but its localization error is bound and theoretically proven. The heuristic function is used in the modified competitive rule of the self-organizing map (SOM) approach for the Traveling Salesman Problem (TSP) [9] that is applied to the multi-goal path planning problem, i.e. the problem of finding a shortest path visiting given set of goals [10]. The proposed method finds a path with lower localization uncertainty at the visited AoIs than in the case of methods minimizing only the path length.

This paper is organized as follows. The localization uncertainty model and the derived heuristic function are described in Section II together with an overview of the used SOM scheme. The problem formulation is presented in Section III The proposed planning method is presented in Section IV. Experimental results of the proposed method are presented in Section V. A discussion of the proposed approach and remarks about future work are presented in the conclusion.

II. Related Work

A. Navigation Method and Uncertainty Model

The used navigation method [7] is based on heading corrections only while the traveled distance is estimated by a rel-

atively imprecise odometry. The heading corrections utilize salient objects (Speeded Up Robust Features (SURF) [11]) recognized in the environment in the map&replay approach. At first, the robot is tele-operated in the environment along straight line segments in the mapping phase. For each segment a set of visual landmarks is remembered and the local length of the segment is measured by the odometry, thus the long term instability of the odometry is not an issue. In the replay phase, the robot is placed at the starting segment and requested to travel the learned path. The current visible landmarks are matched with the learned ones by the histogram voting method, which realizes the so-called visual compass. Based on the heading deviation the control law steers the robot in the desired direction. Once the traveled distance reaches the segment length, the robot is turned into the direction of the next segment by the compass, and the next segment is traversed in the same manner. Even though the navigation method is very simple, it allows the robot to continuously travel closed paths more than one kilometer long in real outdoor environments in variable lighting conditions and seasonal changes, see experimental results in [7].

To establish the localization uncertainty model a simple case of a robot navigating along a single segment aligned with the x axis can be considered, see Fig. 1. Let the robot

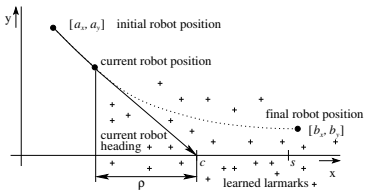


Fig. 1: Robot navigation model

start at position $[a_x, a_y]$ and learned landmarks are in front of the robot. During the robot movement, the nearby landmarks disappear and more distant landmarks become visible, and therefore landmarks can be considered in a constant distance ρ ahead of the robot. The robot movement can be characterized by the differential equation $dx/dy = \rho/-y$. Assuming $c \approx a_y$ and boundary conditions the final robot position is $[b_x, b_y] = [a_x + s, a_y \exp(-s/\rho)]$. The model of the robot movement can be augmented by an odometry error ν and heading sensor noise ξ (random variables drawn from the Gaussian distribution with the zero mean and the variances η and τ respectively) resulting in

$$\begin{bmatrix} b_x \\ b_y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-\frac{s}{\rho}} \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} + \begin{bmatrix} s + s\nu \\ \xi \end{bmatrix}, \quad (1)$$

which rewritten to a matrix form denotes $\boldsymbol{b} = \boldsymbol{M}\boldsymbol{a} + \boldsymbol{s}$. For an arbitrary orientation of the segment the matrix equation can be complemented by the rotation matrix $\boldsymbol{R}$:

$$\boldsymbol{b} = \boldsymbol{R}^T \boldsymbol{M} \boldsymbol{R} \boldsymbol{a} + \boldsymbol{R}^T \boldsymbol{s}. \quad (2)$$

The evolution of the robot position uncertainty for a single segment is based on consideration of the robot position as a random variable drawn from 2D normal distribution with the mean $\hat{\boldsymbol{a}}$ and the covariance matrix $\boldsymbol{A}$. Due to linear and absolute terms of Equation (2) the uncertainty at the segment end is a normal distribution with the mean $\hat{\boldsymbol{b}}$ and the covariance matrix $\boldsymbol{B}$. To investigate the evolution of the robot position covariance matrix, the robot position can be denoted to $\boldsymbol{a} = \hat{\boldsymbol{a}} + \tilde{\boldsymbol{a}}$, where $\hat{\boldsymbol{a}}$ is the mean of $\boldsymbol{a}$ and $\tilde{\boldsymbol{a}}$ denotes a random variable with the zero mean and the covariance matrix $\boldsymbol{A}$. Regarding to independence of $\tilde{\boldsymbol{s}}$ and $\tilde{\boldsymbol{a}}$

$$\tilde{\boldsymbol{b}}\tilde{\boldsymbol{b}}^T = \boldsymbol{R}^T \boldsymbol{M} \boldsymbol{R} \tilde{\boldsymbol{a}} \tilde{\boldsymbol{a}}^T \boldsymbol{R}^T \boldsymbol{M}^T \boldsymbol{R} + \boldsymbol{R}^T \tilde{\boldsymbol{s}} \tilde{\boldsymbol{s}}^T \boldsymbol{R}. \quad (3)$$

For a sequence of segments, the end of the segment i is the start of the segment $i+1$, i.e. $\boldsymbol{a}_{i+1} = \boldsymbol{b}_i = \boldsymbol{R}_i^T \boldsymbol{M}_i \boldsymbol{R}_i \boldsymbol{a}_i + \boldsymbol{R}_i^T \boldsymbol{s}_i$. The evolution of the uncertainty is a recurrent form of Equation (3) that is in terms of covariance matrices

$$\boldsymbol{A}_{i+1} = \boldsymbol{R}_i^T \boldsymbol{M}_i \boldsymbol{R}_i \boldsymbol{A}_i \boldsymbol{R}_i^T \boldsymbol{M}_i^T \boldsymbol{R}_i + \boldsymbol{R}_i^T \boldsymbol{S}_i \boldsymbol{R}_i, \quad (4)$$

where

$$\boldsymbol{M}_i = \begin{bmatrix} 1 & 0 \\ 0 & e^{-\frac{s_i}{\rho}} \end{bmatrix}, \boldsymbol{S}_i = \begin{bmatrix} s_i^2 \eta^2 & 0 \\ 0 & \tau^2 \end{bmatrix}$$

The length of the traveled segment is s_i, ρ represents density of the landmarks in the environment (an "average" distance of the landmarks to the robot), η and τ represent precision of the odometry and the heading sensor.

Equation (4) provides estimation of the robot position uncertainty at the particular segment end and can be used as a heuristic function of the uncertainty evolution in a path planning algorithm. The stability (boundness) of the navigation method for closed paths consisting of several conjoined straight line segments is proven in [7] for a robot moving in a plane with an imprecise measurements of the traveled distance, a forward aimed camera capable recognizing a nonempty subset of mapped landmarks and paths with at least two noncollinear segments.

B. Self-Organizing Map Based Multi-Goal Path Planning

The multi-goal path planning problem is formulated as the TSP, which allows to use any TSP solver. However, self-organizing map (SOM) approach is useful, because it is flexible enough to consider the evolution of the localization uncertainty in a straightforward way. The SOM algorithm for the TSP [9] has been selected as the main adaptation schema being modified. The algorithm is Kohonen's unsupervised neural network in which nodes are organized into a cycle and a solution of the TSP is represented by synaptic weights of nodes that are adapted to the goals (cities) during the self-adaptation process.

The adaptation consists of two phases: *competitive* and *cooperative*. In the *competitive* phase, goals are presented to the network in a random order. For each goal a winner (the closest) node is found according to its Euclidean distance to the goal. The *cooperative* phase is an adaptation of nodes (weights) to the presented goal. A winner node and its neighbouring nodes are adapted (moved) towards the presenting goal g by the adaptation rule $\nu_j' = \nu_j + \mu f(\sigma, l)(g - \nu_j)$, where ν_j and g are coordinates of the node and the presented

goal, μ is the fractional learning rate and $f(\sigma, l)$ is the neighbouring function. The function is $f(\sigma, l) = exp(-l^2/\sigma^2)$ for $l < d$ and $f(\sigma, l) = 0$ otherwise, where σ is the gain parameter (also called the neighbourhood function variance), l is the distance in a number of nodes measured along the ring, d is the size of the winner node neighbourhood that is set to $d = 0.2m$, where m is the number of nodes. After the complete presentation of all goals (one adaptation step) the gain is decreased by $\sigma = (1 - \alpha)\sigma$, where α is the gain decreasing rate. The adaptation process is repeated until the distance of a winner to the city is lower than given threshold, e.g. 0.001. An inhibition mechanism [9] is used to avoid nodes to win too often. The initial value of σ is set according to $\sigma_0 = 0.06 + 12.41n$, where n is the number of goals. The learning and decreasing rates are $\mu = 0.6$, $\alpha = 0.1$. The number of nodes m is set to $m = 2.5n$.

III. Problem Statement

The problem addressed in this paper is motivated by an instance of the inspection task that is a problem of visiting given set of goals. The goals' positions in the environment are known in advance. A mobile robot is requested to repeatably visit the goals, due to its restricted sensing capabilities, i.e. all goals are not visible from a single place. The problem is to find a sequence of goals visits with a minimal inspection period. The robot is capable to be navigated by the method described in Section II-A. The robot uses imperfect sensors, therefore its position estimation is imprecise.

Even though the stability of the used navigation method has been theoretically proven and experimentally verified in real-world environments, it does not mean that the localization error is sufficiently low. Therefore, to increase the probability of visiting the requested goals, the localization uncertainty at the goal positions should be as small as possible.

For simplicity, the environment is assumed to be obstacle free, thus a path can be composed from straight line segments connecting the goals. Once a path is found, the robot is navigated along the path in the tele-operated manner in order to create a map of the environment. After that, the robot is requested to periodically visit the goals using the mapped path.

IV. Planning with Localization Uncertainty

The proposed multi-goal path planning algorithm combines Equation (4) and the SOM adaptation schema for the TSP. The idea is based on evolution of the localization uncertainty that depends on particular values of η, τ and ρ. The imprecise odometry increases the error in the direction of the robot movement, while the error is suppressed by heading corrections in the lateral direction. For a robot with single forward looking camera, it means that a lower localization uncertainty is achieved by a moving along "zig-zag" trajectories instead of a single straight line segment. So, the direction from which the robot arrives to the goal is crucial in the uncertainty decreasing process. In other words, a simple straight line segment path between two goals g_1 and g_2 increases the error in the segment (longitudinal) direction due to the imprecise odometry. To decrease the uncertainty at g_2 caused by the odometry, the robot can be navigated to an auxiliary navigation point close to g_2. From a point at some perimeter around g_2, the robot movement to g_2 diminishes the previously increased error in the g_1–g_2 longitudinal direction. The situation is demonstrated in Fig. 4. A radius of the perimeter can be selected according to the odometry error and profuseness of landmarks in the environment that is represented by ρ.

Based on the observation the planning problem is formulated as the following modification of the TSP. Each goal g is represented by a group of points $P_g = \{p_{g,1}, \ldots, p_{g,k}\}$ at the perimeter d_p and the problem is to select a single point from each group such that the uncertainty at each goal is minimized and the total route length is minimized as well. The SOM competitive and cooperative adaptation rules are modified to consider the evolution of the localization uncertainty using Equation (4). The ring of nodes must be oriented in order to used the equation during the adaptation process. It is achieved by the adaptation of the first and the last ring nodes (in fact any two neighbouring nodes can be used) to the selected starting goal prior presentation of other goals to the network. During the ring evolution, each node has associated the localization uncertainty represented by the covariance matrix A_ν. The covariance matrix at the first node is computed from the connection of the node with the starting goal by a straight line segment. The initial uncertainty at the starting goal is set to zero. The matrix A_{ν_i} is computed from $A_{\nu_{i-1}}$ directly by Equation (4) where s is the Euclidean distance between the nodes ν_{i-1}, ν_i. The modified rules are as follows.

The winner node to a goal g is selected from not inhibited nodes according to the Euclidean distance between a node and the goal. The orientation of the ring defines forward and backward neighbourhoods of the winner node, which are utilized in the cooperative phase. The backward neighbouring nodes of the winner node $\nu^\star$ are adapted to the perimeter point $p_{g,\star}$, while $\nu^\star$ and its forward neighbouring nodes are adapted towards g. The perimeter point $p_{g,\star}$ is selected from P_g according to

$$p_{g,\star} = \mathrm{argmin}_{p \in P_g}(||\boldsymbol{A}_g||^2), \tag{5}$$

where $||\boldsymbol{A}_g||^2$ denotes the norm of the covariance matrix, i.e. the maximal eigenvalue of $\boldsymbol{A}_g \boldsymbol{A}_g^T$. The particular matrix $\boldsymbol{A}_g$ is computed from the straight line segments $(\nu^{\star-1}, p_{g,.})$ and $(p_{g,.}, g)$, where $\nu^{\star-1}$ is the first backward node of the winner. After the adaptation, the nodes $\nu^\star$ and $\nu^{\star-1}$ are marked as inhibited for the rest of the current adaptation step. The proposed algorithm is called *dadapt* from the "*double adaptation*", because of two performed adaptations: towards $p_{g,\star}$ and g.

The final path can be constructed by two methods. The first variant is called *dadapt-ring*, because it directly uses the ring as the final path. The second variant consider only the winner nodes associated to the goals and their first backward

nodes associated to the perimeter points, the variant is called *dadapt-perim*.

V. Experiments

The proposed multi-goal path planning algorithm has been experimentally verified in several scenarios. Four path construction variants are considered in the algorithm examination. The first variant is a solution of the TSP without consideration of the localization uncertainty and it is refereed as *simple*. The second variant represents straightforward decreasing of the longitudinal error and it is used as a reference method to examine quality of SOM solutions. The path is constructed from a TSP solution where an additional perimeter point p is placed before each visited goal. The point is placed at the perimeter in such a way, that line segments $(g_{i-1}, p$ and (p, g_i) form the right angle and one of the two solutions is randomly selected. Finally two SOM algorithm variants *dadapt-perim* and *dadapt-ring* are used.

The quality of solution is characterized by the length of the found path L and the maximal localization uncertainty at the visited goals E_g computed from the covariance matrix

$$E_{max} = \max_{g \in \boldsymbol{G}} \sqrt{(||\boldsymbol{A}_g||^2)}.$$

Due to randomization of the SOM algorithm fifty solutions are found by the particular algorithm variant and the quality metrics are computed as average values. Besides, the best solution with the lowest E_{best} is used to estimate algorithm capability to find good solutions.

Fig. 2: AR Drone quadcopter.

The considered robot is the AR Drone quadcopter [12] shown in Fig. 2. The following values for the outdoor environment have been set: η=0.1 m, τ=0.1 m and ρ=20 m. In the case of an indoor environment, landmarks are closer, therefore $\rho = 5.5$ has been used. The used SOM parameters are $\sigma_0 = 12.14n + 0.6, \mu = 0.6, \alpha = 0.1, \delta = 0.001, d = 0.2m$, where n is the number of goals and m is the number of nodes that is set to $m = 4n$. To examine the effect of the perimeter, solutions have been found for several perimeter radiuses. Each goal perimeter is sampled by 48 equidistantly placed points in the proposed SOM algorithm with dual adaptation.

A. Planned paths for Outdoor Environments

Experimental results for the *square* scenario are depicted in Fig. 3 and examples of found paths for perimeter radius 19 meters are shown in Fig. 4. The results indicate that considering localization uncertainty leads to more than two

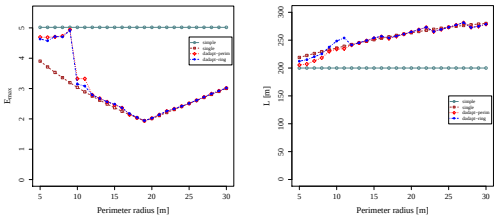
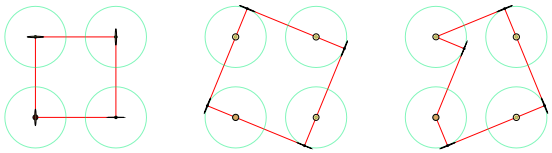


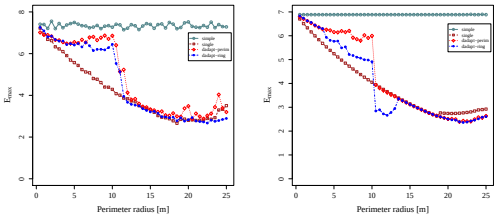
Fig. 3: Influence of the perimeter to the solution quality, *square* scenario, $\eta = 0.1, \tau = 0.1, \rho = 20$.



(a) *simple*, L=200 m, E_{max}=5 m (b) *single-perim*, L=261 m, E_{max}=2 m (c) *dadapt-ring*, L=261 m, E_{max}=2 m

Fig. 4: Solutions for the *square* scenario, perimeter at 19 m. The green disks are goals, the green circles denote perimeters, the found path is in red and ellipses denote localization uncertainty at goals (yellow) and perimeter points (blue).

times lower localization uncertainty at the goals, while the length of the path is about thirty percents longer. The proposed SOM algorithm with the dual adaptation provides competitive results to the *single-perim* variant in this simple scenario. The lowest E_{max}=1.94 m is achieved for the perimeter at 19 m. The variances of the computed quality metrics are very small due to simple configuration of the goals, the highest values are for the *dadapt* variants, but they are typically less than one meter for L and they are in hundredths for E_{max}.



(a) maximal localization uncertainty E_{max} (b) E_{max} of the best found solution

Fig. 5: Influence of the perimeter to the solution quality, *square* scenario, $\eta = 0.1, \tau = 0.1, \rho = 20$.

To examine capabilities of the proposed algorithm in a larger environment fourteen goals have been placed within Charles's Square location. Experimental results are presented in Fig. 5. The *dadapt* algorithms provide worse average solutions up to perimeter around twelve meters. However,

consideration of the best found solution provides interesting observation. The *dadapt-ring* algorithm provides better solutions for perimeters between ten and fourteen meters. In these cases, the found solutions have localization uncertainty about one meter lower than for the *single-perim* variant. The perimeter at 18.5 m provides the lowest values of E_{max}. The best solution found by the *dadapt-ring* algorithm for additional twenty new runs and perimeter at twelve meters is depicted in Fig. 6. The best solution provided by the *single-perim* algorithm variant has quality metrics L=641.5 m with E_{max}=3.65 m.

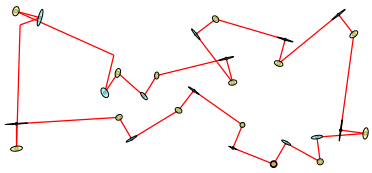


Fig. 6: The best found solution for *Charles's square* scenario, *dadapt-ring*,d_p=12 m,L=674.4 m, E_g=2.85 m.

B. Computational Requirements

The algorithms have been implemented in C++, compiled by the G++ 4.2 with the '-O2' optimization flag and executed at workstation with 2 GHz CPU. The *simple* and *single-perim* algorithms variants provide solution in units of milliseconds (including the solution of the TSP). The *dadapt* algorithm is more computationally intensive due the used number of perimeters points, particularly solutions for the *square* scenario have been found in tens of milliseconds and less than four hundred milliseconds in the case of *Charles's square* scenario. Even though the covariance matrix is computed for each node after each adaptation of the winner node, the simplicity of Equation (4) allows to plan several paths and select the best found solution or plan in real-time.

C. Real Indoor Experiments

Two scenarios have been used in real experiments within indoor environment. In these scenarios, a white color card with dimensions 85.6× 54 mm has been placed at each goal. The AR drone has been manually navigated along the given path, then it has been requested to traverse the closed path autonomously. The drone has been placed at the last goal approximately in the directions of the first goal or the perimeter point at each run. At each goal (over the card) the bottom camera has been used to take a snapshot of the card. The success of the navigation has been measured by the number of observed cards, i.e. a detected card in the snapshot over the goal area. The view angle of the vertical camera is approximately sixty degrees and the high of the drone has been in 1.0 - 1.5 m above the goal.

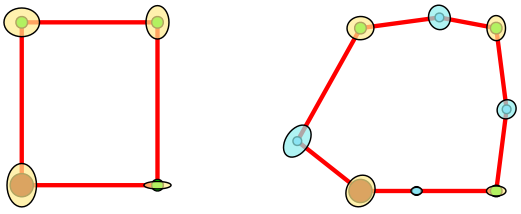
Scenario A - In the first scenario, goals form a rectangle with dimensions 6.25 × 3.75 m. Experimental results from five runs are presented in Table I. The *simple* path is a direct connection of the goals, while perimeter points have been manually set before each goal in the *perim* path.

TABLE I: Scenario A

Path Planning Method	Success Rates%				
	g_1	g_2	g_3	g_4	overall
simple	20	20	40	40	30
perim at 1.77m	60	60	40	20	45

Scenario B - A rectangle with 3.750 × 4.375 m and perimeter at 2.2 m has been used in this scenario. Beside the *simple* path, the best found solution by the method *dadapt-perim* has been used. Several snapshots of the goal area have been taken for the card detection, because of the following control issue. The images from the on-board cameras are transfered to the laptop for the image processing, i.e. SURF computation and landmark matching. The WiFi connection with the AR Drone caused unpredictable delays. These delays did not significantly affect the main navigation loop, however the command to take a snapshot of the goal area has been sometimes delayed. Even though the AR Drone tries to stabilize its position, under certain circumstances it is slightly moved. Thus, to avoid possible miss detection of the goal area from a single snapshot, several images have been captured and the goal visit has been considered as successful if the card has been detected in at least one snapshot.

The used paths are shown in Fig. 7. Notice that the perimeter point is needed only for the last goal g_4. The first perimeter point lies on the straight line segment from the start goal the g_1. In other cases the perimeter points are at distance 30 cm from the straight line segment, which in fact is very close to the expected localization precision. Success rates of the detected goals from tens runs are presented in Table II. The AR Drone during experiments is shown in Fig. 8.



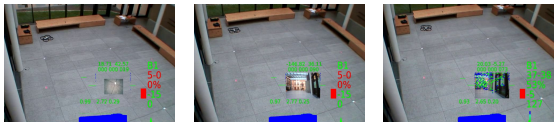
(a) *simple*, E_{max}=0.58m (b) *dadapt-perim*, E_{max}=0.45m

Fig. 7: Found paths by the *simple* and *dadapt-perim* algorithm variants and for the perimeter at 2.2 m in the scenario B.

The *single-perim* algorithm variant is not suitable for this scenario as distances between goals are relatively small and perimeter points are not necessary for first goals. The best found solutions has E_{max}=0.5 m and is about three meters longer than the *dadapt-perim* solution.

TABLE II: Scenario B

Path Planning Method	Success Rates%				
	g_1	g_2	g_3	g_4	overall
simple	100	100	60	70	82.5
dadapt-perim	100	100	90	90	95.0



(a) goal snapshot (b) front camera view (c) detected landmarks

Fig. 8: The AR Drone during experiment, the user interface overlays the real scene, the small picture is from the AR drone on-board cameras.

VI. Discussion and Conclusion

The presented results show that the expected localization uncertainty can be decreased by consideration of heuristic function describing evolution of the localization error. The used heuristic function is based on simple and stable navigation method, thus regarding to its real performance [7] and presented experimental results it seems that the function is computationally efficient and sufficiently informative. The proposed multi-goal path planning algorithm shows how the function can be used in the path planning task. Even though the results show uncertainty reduction, the benefit of the used SOM schema is mainly in providing solution of the multi-goal path planning problem. The straightforward *single-perim* variant provides better results in most cases with less required computational times. The efficiency of perimeter point placement by the *single-perim* method is caused by the dominant longitudinal error, because the right angle constraint is sufficient to decrease this type of error. In a general case, consideration of other orientation of the error, i.e. axes of the error ellipse, can lead to higher reduction. Also the axes directions depend on the order of goals in the path, thus the post-processing procedure can provide sub-optimal solutions. From this point of view, the best found solution found by the *dadapt* algorithm can be a motivation for further investigation of SOM application in the formulated multi-goal path planning problem. Moreover, for the small indoor environment, the *dadapt* method is able to find solutions in which all perimeter points are not needed. The expected benefit of SOM is in its flexibility to deal with various problem variants, e.g. considering obstacles in the environment [13], restrictions of directions to reach goals or in the case of several mobile robots [14].

Our future intention is to examined the proposed method in a real-world outdoor scenario[1]. The main idea of the uncertainty reduction is to increase the reliability of the inspection. The found plan represents only expected localization uncertainty, thus it is clear that real performance will be different and a robot will unlikely be navigated precisely to the goal position. On the other side, it is sufficient if the robot is navigated to the goal vicinity, because a robot can be then locally navigated to the goal by another navigation method. For an UAV, it means that forward looking camera can be used for the navigation to the goal vicinity, while the vertical camera is used for the local navigation to the goal. The execution of the plan has one important aspect relating to the localization uncertainty. If the goal is successfully recognized and its position is known, it can be used to localize the robot. Thus, additional reduction of the localization error is achieved, which increases the probability of the next goal visit. These ideas will be investigated in our future work.

[1]We have planed experiment in Charles Square park, however due to weather conditions and low turbine power of the UAV it was not possible. Even small wind made the robot control unpredictable.

Acknowledgments

This work was supported by the Grant Agency of the Czech Technical University in Prague, grants No. SGS10/185 and No. SGS10/195 and by the Ministry of Education of the Czech Republic under program "National research program II" by the projects 2C06005 and by the project No. 7E08006. The support of EU under the project ICT-2007-1-216240 is also acknowledged. We would also like to thank the Parrot company for providing AR drones to us.

References

[1] A. Remazeilles and F. Chaumette, "Image-based robot navigation from an image memory," *Robot. Auton. Syst.*, vol. 55, no. 4, pp. 345–356, 2007.

[2] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette, "A mapping and localization framework for scalable appearance-based navigation," *CVIU*, vol. 113, no. 2, pp. 172–187, February 2009.

[3] A. Censi, D. Calisi, A. D. Luca, and G. Oriolo, "A Bayesian framework for optimal motion planning with uncertainty," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Pasadena, CA, May 2008.

[4] U. Frese, "A discussion of simultaneous localization and mapping," *Auton. Robots*, vol. 20, no. 1, pp. 25–42, 2006.

[5] A. Martinelli, N. Tomatis, and R. Siegwart, "Some results on SLAM and the closing the loop problem," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Canada, 2005, pp. 334–339.

[6] S. J. Julier and J. K. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *IEEE/RSJ Int. Conf. Robotics and Automation*, 2001, pp. 4238–4243.

[7] T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, and L. Přeučil, "Simple yet stable bearing-only navigation," *Journal of Field Robotics*, vol. 27, no. 5, pp. 511–533, 2010.

[8] Z. Chen and S. T. Birchfield, "Qualitative vision-based path following," *Trans. Rob.*, vol. 25, no. 3, pp. 749–754, 2009.

[9] S. Somhom, A. Modares, and T. Enkawa, "A self-organising model for the travelling salesman problem," *Journal of the Operational Research Society*, pp. 919–928, 1997.

[10] M. Saha, T. Roughgarden, J.-C. Latombe, and G. Sánchez-Ante, "Planning Tours of Robotic Arms among Partitioned Goals," *Int. J. Rob. Res.*, vol. 25, no. 3, pp. 207–223, 2006.

[11] H. Bay, T. Tuytelaars, and L. Gool, "SURF: Speeded up robust features," in *Proc. 9th European Conf. Comput. Vision*, Graz, Austria, 2006, May.

[12] "Parrot AR Drone," http://ardrone.parrot.com/parrot-ar-drone/en, cited July 2010.

[13] J. Faigl, M. Kulich, V. Vonásek, and L. Přeučil, "An Application of Self-Organizing Map in the non-Euclidean Traveling Saleman Problem," *Neurocomputing*, to appear.

[14] J. Faigl, "Multi-goal path planning for cooperative sensing," Ph.D. dissertation, Czech Technical University in Prague, 2010.

Appendix 8 - Faigl, J. - Krajník, T. - Vonásek, V. - Přeučil, L.: *On Localization Uncertainty in an Autonomous Inspection* [34], referenced on page 24.

2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

On Localization Uncertainty in an Autonomous Inspection

Jan Faigl, Tomáš Krajník, Vojtěch Vonásek, Libor Přeučil

***Abstract*— This paper presents a multi-goal path planning framework based on a self-organizing map algorithm and a model of the navigation describing evolution of the localization error. The framework combines finding a sequence of goals' visits with a goal-to-goal path planning considering localization uncertainty. The approach is able to deal with local properties of the environment such as expected visible landmarks usable for the navigation. The local properties affect the performance of the navigation, and therefore, the framework can take the full advantage of the local information together with the global sequence of the goals' visits to find a path improving the autonomous navigation. Experimental results in real outdoor and indoor environments indicate that the framework provides paths that effectively decreases the localization uncertainty; thus, increases the reliability of the autonomous goals' visits.**

I. Introduction

This paper concerns a problem of finding a reliable path for an autonomous mobile robot in the inspection task, i.e., a problem of finding a path to visit a set of goals. Having a model of the robot's workspace the required mission objective is to maximize the frequency of the goals' visits, which leads to minimize the inspection path length. However, due to a localization uncertainty, the path length is not the only criterion, and a precision of navigation is also considered during the path planning. The proposed approach follows the basic idea of the planning approaches considering robot position uncertainty that is to design a sequence of robot's actions to fulfill the desired mission objective while minimizing the position uncertainty. Let us briefly review previous approaches in this field.

The motion planning problem with uncertainty has been addressed using the "Sensory Uncertainty Field" notion in [1]. The approach provides estimation of possible errors in robot position computed by the localization function for every possible robot configuration. Then, a combination of the expected error with the path length is minimized during the path planning. In [2], authors formulate a problem of bearing-only target localization as an optimization problem of finding an optimal observer trajectory. Their approach is based on the Fisher information matrix (FIM) representing information contained in a sequence of measurements.

Authors of [3] proposed a path planning algorithm that finds a safe path in an uncertain-configuration space using a localization function based on the Kalman filter technique (EKF). Safe paths are also studied in [4], where authors address the problem of computationally intractable stochastic control problem as a path planning problem in a Bayesian framework using extended planning space that is created as a Cartesian product of robot poses and covariances. A safe path is determined by a state space searching algorithm, which finds a path as a sequence of state transitions using FIM. Although the approach is well formalized, the authors noted it is computationally demanding if states in the open list cannot be easy discarded due to a dominated state, i.e., the case in which states are a totally ordered set, and therefore, all relations have to be evaluated.

High computational requirements are also the case of the general framework for planning with uncertainty based on the partially observable Markov decision process (POMDP) [5]. On the other hand, sampling based techniques based on an extension of the configuration space by an "uncertainty dimension" seem to provide computationally feasible solution [6]. However, the key issue is an efficient determination of the collision probability with position uncertainty [7].

An alternative to the belief space planning approach has been presented in [8], where Belief Roadmap (BRM) approach is proposed. The BRM is a variant of the Probabilistic Roadmap algorithm for linear Gaussian systems, where nodes of the graph built have associated information about belief estimation. Thus, the roadmap allows to plan a trajectory regarding its length and uncertainty along it.

The aforementioned planning approaches consider uncertainty in sensors' measurements or in a robot position estimation using a regular localization function, e.g., based on the update step of the EKF. In the proposed approach, we rather consider a model of the localization uncertainty evolution based on a mathematical formulation of the navigational method [10]. The main advantage of the proposed approach is an efficient determination of the robot position uncertainty along a path consisting of a sequence of straight line segments. The basic formula describing the position uncertainty is a simple matrix equation that allows us to consider the path planning problem as an instance of the multi-goal path planning problem (MTP). The MTP is formulated as the well-know Traveling Salesman Problem (TSP) [11], [9], which is known to be NP-hard. The TSP stands to find a best sequence of goals visits; however, an order of goals' visits affects the precision of the goals' visits. Thus, we propose a planning framework for the MTP with the localization uncertainty. Although the graph based TSP solver can be eventually used with the BRM, the framework proposed does not require explicit construction of a graph before solving the TSP; thus, a solution found does not depend on sampling strategy as PRM based approaches.

Authors are with the dept. of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27 Prague, Czech Republic {xfaigl}@labe.felk.cvut.cz

978-1-4673-1404-6/12/$31.00 ©2012 IEEE

We introduced the initial idea of the framework in [12], where the Parrot AR.Drone has been utilized in the verification of the main principle of the proposed MTP solver. The real experimental results show that the proposed planning improves the success rate of the goals' visits from 82.5 % to 95 %. However, this initial work is rather limited, and therefore, in this paper, we extend the framework to deal with environments with obstacles. Moreover, we generalize the model of the autonomous navigation to consider local properties of the environment and develop local models of visible landmarks used for the navigation. The generalized framework and its experimental verification in real outdoor environment are the main contribution of this paper.

The paper is organized as follows. The problem addressed is specified in the next section. In Section III, the main idea of the localization uncertainty decreasing is presented. The proposed MTP framework is described in Section IV. The experimental results are presented in Section V, and concluding remarks are dedicated to Section VI.

II. Problem Statement

The problem addressed is an instance of the inspection path planning for a mobile robot operating in a planar environment. The map of the environment is a priori known, and it is represented as a polygon with holes $\mathcal{W}$. Obstacles of $\mathcal{W}$ are enlarged to respect dimensions of the robot and the required clearance. It is assumed the robot has differential drive; thus, a point robot model is considered in $\mathcal{W}$. A set of n goals $\boldsymbol{G} = \{g_1, \ldots, g_n\}$ representing areas of interest to be visited is given, and each goal $g \in \boldsymbol{G}$ is reachable by the mobile robot, $g \in \mathcal{W}$. The considered multi-goal path planning problem is as follows: *Find a closed shortest path in $\mathcal{W}$ visiting all goals of $\boldsymbol{G}$ while the localization error of the robot at the goals is minimized.* Without loss of generality the starting point is assumed to be $g_n \in \boldsymbol{G}$.

The problem addressed is a variant of the well-known Traveling Salesman Problem (TSP), in which not only the path length is considered, but also the localization error is taken into account. As two criteria are minimized, it is clear that only Pareto optimality can be achieved. The problem is how the length of the path and the localization uncertainty at the goals relate, and what improvements can be achieved.

A. Models of the Localization Uncertainty

An efficient and informative heuristic function is needed to incorporate the localization uncertainty into the path planning algorithm. We assume that the mobile robot performing the inspection is navigated by the method presented in [10], where a proof of stability of the method and its experimental validation can be found. The method uses a map of salient objects of the environment, and matches the current seen objects to steer the robot in the desired direction. Assuming distance and heading estimations are independent and not correlated (e.g., using odometry for distance measurement and vision based heading estimation), the position uncertainty can be expressed in terms of covariance matrices [10]. For a robot navigated along a straight line segment (with the length s_i) from the position a_i, the covariance matrix $\boldsymbol{A}_{i+1}$ at the end of the segment i (position a_{i+1}) can be computed using the formula[1]:

$$\boldsymbol{A}_{i+1} = \boldsymbol{R}_i^T \boldsymbol{M}_i \boldsymbol{R}_i \boldsymbol{A}_i \boldsymbol{R}_i^T \boldsymbol{M}_i^T \boldsymbol{R}_i + \boldsymbol{R}_i^T \boldsymbol{S}_i \boldsymbol{R}_i, \quad (1)$$

where

$$\boldsymbol{M}_i = \begin{bmatrix} 1 & 0 \\ 0 & m(a_i, a_{i+1}, \mathcal{M}) \end{bmatrix}, \boldsymbol{S}_i = \begin{bmatrix} s_i\eta^2 & 0 \\ 0 & \tau^2 \end{bmatrix},$$

$m(a_i, a_{i+1}, \mathcal{M})$ represents a model of visible landmarks used for the navigation, $\boldsymbol{R}$ is the rotation matrix, and η, τ represent precision of the odometry and the heading sensor, respectively.

1) Global model of visible landmarks: For homogeneously distributed landmarks the model can be characterized using a single parameter ρ representing an "average" distance of the landmarks ahead of the robot. In this case, the function m can be expressed as [10]:

$$m(a_i, a_{i+1}, \mathcal{M}) = e^{-\frac{s_i}{\rho}}. \quad (2)$$

2) Local model of visible landmarks: Alternatively a map of the landmarks $\mathcal{M}$ can be utilized to compute expected distance of the closest landmark to the robot traveling from the position a_i towards the position a_{i+1}. Let the distance of such a landmark be d. Then, the model can have a form:

$$m(a_i, a_{i+1}, \mathcal{M}) = \frac{d}{s_i + d}, \quad (3)$$

which describes position of the robot moving towards a particular landmark. A particular value of d can be computed using various approaches depending on the representation of $\mathcal{M}$, e.g., point or polygonal map of landmarks. An expected benefit of the local model is a more precise estimation of the localization uncertainty; however, it is clear that it can be more computationally demanding than the global model.

B. Solution Quality Metrics

A natural quality metric of the inspection path visiting the given goals is a length of the path L. However, a robot can miss the goal due to imprecise navigation. Hence, an additional quality metric can be the maximal expected localization uncertainty at the goals. Using the navigational method [10], the uncertainty can be described by Eq. 1, and the expected position error at a goal g can be computed using the maximal eigenvalue of the matrix $\boldsymbol{A}_g$. Thus, the maximal localization error is

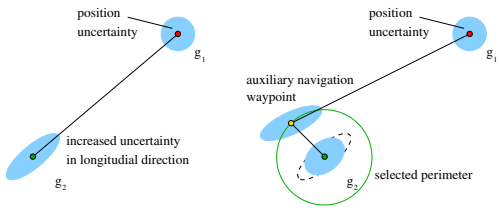
$$E_{max} = \max_{g \in \boldsymbol{G}} \sqrt{(||\boldsymbol{A}_g||^2)}. \quad (4)$$

Beside the quality metrics L and E_{max} of an inspection path found, the real performance of the autonomous inspection can be measured by real distances of the robot to the goals, when the robot announces that it reaches the particular goal. The real distances are examined in the experimental evaluation of the proposed planning method in Section V.

[1] Here, it is worth to mention that s_i in $\boldsymbol{S}_i$ is not in power of two, which is accidentally presented in [10].

III. Principle of Uncertainty Decreasing

A principle of the uncertainty decreasing utilized in the proposed multi-goal path planning is based on a geometrical interpretation of Eq. 1. The idea is as follows. Assume a robot moving from a goal g_1 to a goal g_2 along a straight line segment. The error in the longitudinal direction caused by the odometry is increased, while the error in the lateral direction is decreased due to heading corrections, see Fig. 1a where the corresponding covariance matrices $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$ are visualized as ellipses [13]. In Fig. 1b, it is shown how the error can be decreased by an auxiliary navigation waypoint placed at a selected perimeter around g_2.



(a) direct connection of goals (b) auxiliary navigation waypoint

1: A principle of the localization uncertainty decreasing.

This principle motivates us to consider visitation of an auxiliary navigation waypoint prior each goal visit. The auxiliary waypoint should be placed at a location that will suppress the error in a direction corresponding to the eigenvector of the maximal eigenvalue of the matrix $\boldsymbol{A}_i$. However, due to non-linearity of Eq. 1, such a location is not easy to compute. Moreover, the location can be unreachable due to obstacles. Therefore, eventual auxiliary waypoints are spread around each goal, and an appropriate waypoint is determined during the multi-goal path planning.

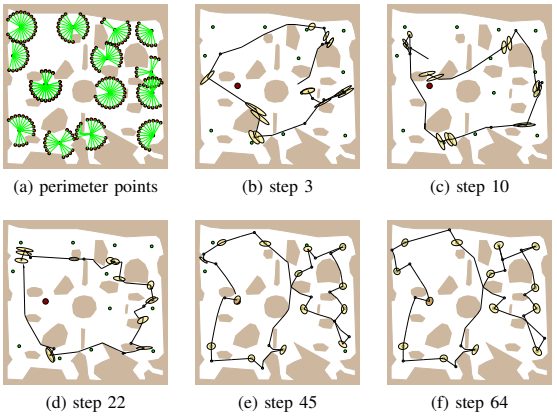
IV. Multi-Goal Path Planning with Localization Uncertainty

Herein, we extend the approach [12] to problems with obstacles using the algorithm [14]. The planning algorithm proposed is a type of unsupervised learning procedure that is a two-layered competitive learning network. An input vector i represents coordinates (g_{i1}, g_{i2}) of the goal g_i, and m output units form the output layer where neurons' weights (ν_{j1}, ν_{j2}) of the node ν_j are points in $\mathcal{W}$. The output units are organized into a unidimensional structure that prescribes a sequence of nodes representing a path in $\mathcal{W}$ in which the first node ν_1 and the last node ν_m denote the orientation of the path. For each goal g a set of auxiliary navigation waypoints is created, e.g., using a perimeter with radius d_p like in [12], but in this case only points inside $\mathcal{W}$ are associated to the goal, $P_g = \{p_{g,i} | p_{g,i} \in \mathcal{W}\}$.

During the learning, the goals are presented to the network in a random order, and for each goal a winning neuron is found using the length of approximate shortest path in $\mathcal{W}$ [14]. To ensure a path will be closed at the desired final goal g_n, the end nodes (ν_1 and ν_m) are adapted to g_n without competition. The node ν_1 and its neighbouring nodes ν_j (for $j > 1$) are adapted by a regular adaptation to g_n, while ν_m and its neighbouring nodes are adapted by the double adaptation rule, called *dadapt*.

The *dadapt* rule adapts nodes to the presented goal and a particular auxiliary navigation waypoint. Each node has associated the localization uncertainty represented by the covariance matrix A_ν. The matrix A_{ν_i} is computed from $A_{\nu_{i-1}}$ by Eq. 1, where a path among obstacles between two nodes is considered, i.e., the path between the nodes is a sequence of straight line segments; thus, each segment of the path is considered using Eq. 1. The matrix A_{ν_1} is computed from the approximate path from g_n to the node because ν_1 can be far from g_n during the adaptation. The initial uncertainty for g_n is set to zero. For a winner node $\nu^\star$ of the goal g, the *dadapt* rule is performed as follows. The backward neighbouring nodes of $\nu^\star$ are adapted to the perimeter point $p_{g,\star}$, while $\nu^\star$ and its forward neighbouring nodes are adapted to g. A point $p_{g,\star}$ is selected from the set P_g to minimize the dominant eigenvalue of $\boldsymbol{A}_g$. Because the adaptation changes positions of the nodes, the covariance matrices are recomputed prior each selection of the perimeter point.

An example of the network evolution is shown in Fig. 2. In all presented figures in this paper, the error ellipses are four times enlarged to show the effect of the uncertainty decreasing.



(a) perimeter points (b) step 3 (c) step 10

(d) step 22 (e) step 45 (f) step 64

2: An example of the path evolution, the green disks represent goals, blue disks are nodes associated to the auxiliary navigation waypoints, the error ellipses are drawn for the winner nodes. The starting point (g_n) is shown as the brown disk.

The advantage of the propose algorithm is the ability to deal with obstacles while minimizing the both quality metrics. The obstacles cause that a path between two goals consists of several segments, which can eventually increase the precision of the navigation. However, the real benefit is not clear, and therefore, the performance of the real autonomous navigation using the proposed planning algorithm

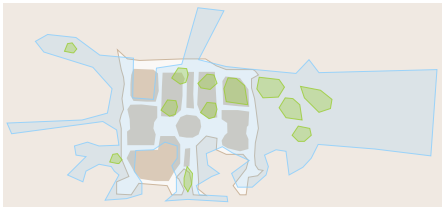
is compared with a simple solution of the TSP (without localization uncertainty) in the experimental part of this paper.

A. Placement of Auxiliary Waypoints

In [12], we place auxiliary waypoints at a selected perimeter and an influence of perimeter radius to the solution quality has been shown. Also in the presence of obstacles, the radius affects the solution quality. Although an appropriate radius can be found experimentally (i.e., solving the problem individually for selected perimeters several times), the flexibility of the underlying self-organizing map algorithm allows to consider a general set of waypoints at various perimeters.

B. Local Models of Visible Landmarks

Eq. 3 pre-scribes how visible landmarks affect the localization uncertainty. A practical implementation of the formula can be based on a polygonal map of the visible landmarks or point landmarks can be directly used. In the first case, visible objects form obstacles and the distance d to the closest visible landmark can be found using an intersection of the supporting line of the path's segment with a segment forming the polygonal representation of the landmarks $\mathcal{M}_{polyg}$. An example of the superimposed landmark map is shown in Fig. 3, the map has been created from the ortophotomap shown in Fig. 4. Thus, the obstacles in this landmark map represent objects, where eventual landmarks can be seen.



3: An example of the polygonal map representing landmarks; the blue polygon is a boundary polygon of the surrounding area and the green polygons represents visible objects within the experimental site, e.g., trees.

Alternatively, landmarks can be represented by a set of points. In this case, it is necessary to consider a field of view (FoV) of the forward looking camera used for the navigation because a point landmark will not likely be placed exactly at the supporting line. An example of such a model is visualized in Fig. 6.

Similarly, the FoV can be considered also for $\mathcal{M}_{polyg}$. Assume a path segment (a_i, a_{i+1}), then the expected landmark is found as an intersection point p of a half-line started at a_{i+1} with a segment of $\mathcal{M}_{polyg}$. Such a point p must also be within the FoV defined by the segment (a_i, a_{i+1}), which provides a rough approximation of the expected landmark. The distance d in (3) is computed as $d = ||(a_i, p)||$. Moreover, this model allows additional visibility constraints, e.g., distance constraints regarding a texture of the objects.

It may happened that an expected landmark is not found by the proposed models due to local properties of the environment model $\mathcal{M}$ around the position a_i. In such a case, the function m has value 1, which corresponds to a landmark placed at infinity, regarding (2).

C. Computational requirements

The complexity of the proposed planning algorithm is proportional to the square of the number of goals multiplied by a number of navigational waypoints associated to one goal. The solutions presented in this paper are typically found in tens of milliseconds for waypoints on a single perimeter and in hundreds of milliseconds for several perimeters using C++ implementation and 3 GHz single core workstation.

V. Experiments

The idea of the uncertainty decreasing presented in Section III has been examined in real experiments with two types of robots, see Fig. 4b and Fig. 4d. All robots have been navigated using the method [10]. First, the benefit of the proposed planning method in real environment with obstacles (a city park) has been examined. Then, the effect of the local models has been examined in a simple scenario using four goals.

In all experiments, the parameters of the navigational model have been estimated for the particular robot and environment. Two paths visiting the given set of goals are used in each experimental scenario. The first path is a solution of the TSP without localization uncertainty denoted as *simple*. The second path is found using the proposed algorithm that is denoted as *dadapt*. For each method, the robot is taught the found path first. Then, the robot is requested to traverse the path several times while its position to the particular goal has been measured whensoever the robot announced it reached the goal. Supplementary materials describing the experiments can be found in [15].

A. Scenario 1 - Autonomous Navigation in a City Park

The P3AT mobile robot has been used to verify the proposed method in a real outdoor experiment in which five goals have been placed within the Charles Square location, see Fig. 4. For each algorithm variant twenty solutions have been found for parameters ρ=15, τ=0.001, ϵ=0.05, d_p=5 m, and a polygonal map created on top of the park orthophotomap. Impassable terrain has been marked as obstacles, which have been enlarged by a small distance to reflect the robot's dimensions and its possible localization error. The best found paths (regarding E_{max}) are depicted in Fig. 5. After the planning, the robot position at the goals' locations has been marked on the ground by a chalk during the path learning. Then, the robot has been requested to traverse the path autonomously for five times. Average robot distances to the goals are presented in Table I. The sample variances of the distances are 0.37 m and 0.32 m for the *simple* and *dadapt* methods respectively.

In this experimental scenario, the robot autonomously traveled about two kilometers. Although the path is found

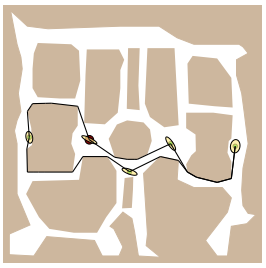
(a) a map of the Charles Square park in Prague, the goals are represented by small yellow disks

(b) the P3AT mobile robot during experiment

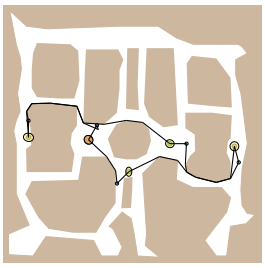
(c) indoor testing site

(d) MMP-5 platform

4: The outdoor and indoor experimental sites.



(a) *simple*, L=184 m, E_{avg}=0.57, E_{max}=0.63



(b) *dadapt*, L=202 m, E_{avg}=0.35, E_{max}=0.37

5: Best solutions found for the Charles Square scenario, d_p=5 m, and the global landmarks model.

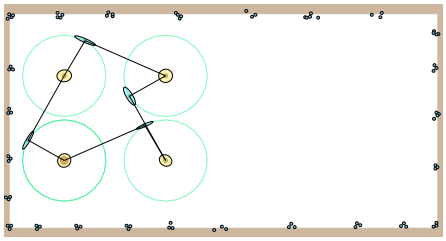
I: Real P3AT distances to the goals

Planning method	**Average distances to the goals** [m]					
	g_1	g_2	g_3	g_4	g_5	overall
simple	0.71	0.92	0.94	0.97	0.93	0.89
dadapt	0.33	0.61	0.71	0.55	0.70	0.58

regarding the dimensions of the robot (using the Minkowski sum), the robot leaved the pathway occasionally due to imprecise localization. These errors do not cause a collision with obstacles, as far landmarks are used for the navigation, and a grass terrain is mostly around the pathways. The robot has been manually moved to the pathway only once and just its lateral position has been changed, i.e., the navigation has been paused, the robot has been moved, and requested to continue the navigation without any additional settings. In this particular case, the robot has been completely on the grass, in other cases at least one wheel remains on the pathway.

B. Scenario 2 - Small Low-Cost Wheeled Mobile Robot

The second experiment has been performed in an indoor environment without obstacles with a MMP5 platform shown in Fig. 4d. A single camera is used for the navigation with the on-board processing using the NVidia ION platform. The identified parameters of the navigational model are ρ=7, η=0.05, and τ=0.01 meters. The goals form a rectangle with 3.750 $\times$ 4.375 m and the proposed *dadapt* method provides the same best paths regarding the lowest E_{max} over several runs and selected perimeters regardless models of landmarks described in Section IV. However, the expected values of E_{max} are different for different landmarks models, and generally the local models provide lower values than the global model. The best found path is visualized in Fig. 6. Real average distances of the robot positions from the goals computed over 10 runs of autonomous navigation are presented in Table II.



6: The best path found using point based local model of landmarks. The blue disks represent landmarks.

II: Real MM5 distances to the goals

Planning method	**Average distance to the goal** [cm]				
	g_1	g_2	g_3	g_4	overall
simple	12.9	14.3	20.4	18.7	16.6
dadapt	9.7	12.6	12.8	16.2	12.8

The expected ratio of the localization uncertainty decrease is about 0.7 using E_{max} for the paths found by the *single* and *dadapt* methods, while the achieved ratio is 0.8. The real average localization errors are 24.8 cm and 20.2 cm for the *single* and *dadapt* paths, respectively, and the maximal errors over all goals and runs are 34 cm and 27 cm, respectively. The cost of the uncertainty decrease is a bit longer path, which is approximately proportional to the uncertainty decrease, i.e., the length ratio is about 1.3.

C. Discussion

Although the expected uncertainty decrease (ratio of the E_{max} for the *single* and *dadapt* methods) differs from the real achieved ratio, the experimental results confirms the benefit of the planning method providing a path leading to a more precise visitations of the goals (about 20 %). The results show that even in the environment with obstacles, the proposed planning is beneficial.

The differences between the expected and achieved results are caused by several factors. First, all the parameters are only estimated. In addition, the global model is only a rough estimation providing average expected results, which in part holds also for the local models. Besides, it's clear that the real robot path cannot be exactly the planned path, unless precise (that means expensive and unpractical) navigation is used for learning the path. Despite that the planned paths provide real valuable guideline suggesting an order of the goals visits and the navigation waypoints.

On the other hand, the proposed generalizations of the planning method considering more auxiliary waypoints and local models of the landmarks do not provide significant benefit in a comparison with a single selected perimeter and the global model. This is mainly due to considered scenarios, which are rather simple. However, these generalizations form fundamental extensions towards a planning framework allowing to consider local properties of the environment and specific sensing device used for the navigation. It is expected that a proper local model will provide a more precise estimation of E_{max}, which will be closer to the really achieved error, as a more sophisticated model can be proposed. For example in the current local model, the expected visible landmark for a segment of the path is found using the segment end. It is assumed that such a landmark will be visible during the traversing the segment. The robot is navigated towards the landmark, and therefore, its distance to the landmark is decreasing. It is obvious that for a very long segment, this approximation is only rough, and a more sophisticated local models can be proposed, e.g., considering the fact that, in reality, closer landmarks along the segment are often used for the navigation. In addition, the eigenvalue used for E_{max} is not exactly the distance measured in real experiments.

VI. Conclusion

An extension of the multi-goal path planning with localization uncertainty for environments with obstacles has been presented. Moreover, generalization of our previous work to deal with a local model of landmarks has been introduced. The planning algorithm can use more navigational waypoints at different perimeters allowing to automatically find the best perimeter waypoint individually for each goal. The proposed approaches have been experimentally verified in real outdoor and indoor scenarios. Although the expected characteristics of the navigation for the planned path differ from the real performance, the benefit of the approaches to uncertainty decrease is evident from the results. All together, the generalized approach presented forms a suitable framework (with low computational requirements) for a further research in path planning with focus on surveillance tasks.

Our future aim is to improve the model of landmarks to achieve closer expected and real performance characteristics of the autonomous navigation to provide more realistic expectations.

Acknowledgments

This work has been supported by the Ministry of Education of the Czech Republic under Projects No. LH11053 and No. 7E08006, and by the EU Project No. 216342.

References

[1] H. Takeda, C. Facchinetti, and J.-C. Latombe, "Planning the motions of a mobile robot in a sensory uncertainty field," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, pp. 1002–1017, October 1994.

[2] Y. Oshman and P. Davidson, "Optimization of observer trajectories for bearings-only target localization," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 35, no. 3, pp. 892 –902, July 1999.

[3] A. Lambert and D. Gruyer, "Safe path planning in an uncertain-configuration space," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, vol. 3, sep. 2003, pp. 4185–4190.

[4] A. Censi, D. Calisi, A. D. Luca, and G. Oriolo, "A Bayesian framework for optimal motion planning with uncertainty," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Pasadena, CA, May 2008.

[5] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, pp. 99–134, May 1998.

[6] R. Pepy, M. Kieffer, and E. Walter, "Reliable robust path planning with application to mobile robots," *Int. J. Appl. Math. Comput. Sci.*, vol. 19, no. 3, pp. 413–424, 2009.

[7] Y. Huang and K. Gupta, "Collision-probability constrained prm for a manipulator with base pose uncertainty," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2009, pp. 1426 –1432.

[8] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *Int. J. Rob. Res.*, vol. 28, pp. 1448–1465, November 2009.

[9] M. Saha, T. Roughgarden, J.-C. Latombe, and G. Sánchez-Ante, "Planning Tours of Robotic Arms among Partitioned Goals," *Int. J. Rob. Res.*, vol. 25, no. 3, pp. 207–223, 2006.

[10] T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, and L. Přeučil, "Simple yet stable bearing-only navigation," *Journal of Field Robotics*, vol. 27, no. 5, pp. 511–533, 2010.

[11] S. N. Spitz and A. A. G. Requicha, "Multiple-Goals Path Planning for Coordinate Measuring Machines," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2000, pp. 2322–2327.

[12] J. Faigl, T. Krajník, V. Vonásek, and L. Přeučil, "Surveillance planning with localization uncertainty for mobile robots," in *Proceeding of the 3rd Israeli Conference on Robotics*, 2010.

[13] R. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[14] J. Faigl, M. Kulich, V. Vonásek, and L. Přeučil, "An Application of Self-Organizing Map in the non-Euclidean Traveling Salesman Problem," *Neurocomputing*, vol. 74, no. 5, pp. 671–679, 2011.

[15] "On localization uncertainty in an autonomous inspection with bearing-only navigation," videos from experiments and supplementary material, http://purl.org/faigl/planning.

2011 IEEE International Conference on Robotics and Automation
Shanghai International Conference Center
May 9-13, 2011, Shanghai, China

On Distance Utility in the Exploration Task

Miroslav Kulich, Jan Faigl, and Libor Přeučil

***Abstract*— Performance of exploration strategies strongly depends on the process of determination of a next robot goal. Current approaches define different utility functions how to evaluate and select possible next goal candidates. One of the mostly used evaluation criteria is the distance cost that prefers candidates close to the current robot position. If this is the only criterion, simply the nearest candidate is chosen as the next goal. Although this criterion is simple to implement and gives feasible results there are situations where the criterion leads to wrong decisions. This paper presents the distance cost that reflects traveling through all goal candidates. The cost is determined as a solution of the Traveling Salesman Problem using the Chained Lin-Kernighan heuristic. The cost can be used as a stand-alone criterion as well as it can be integrated into complex decision systems. Experimental results for open-space and office-like experiments show that the proposed approach outperforms the standard one in the length of the traversed trajectory during the exploration while the computational burden is not significantly increased.**

I. INTRODUCTION

The exploration can be understood as a process of autonomous navigation of a mobile robot in an unknown environment in order to build a model of the environment. An exploration algorithm can be defined as an iterative procedure consisting of a selection of a new goal and a navigation to this goal. Such an algorithm is terminated whenever the defined condition (mission objective) is fulfilled. In this paper, the mission objective is building of a complete map of the environment. Besides, the usage of resources (e.g. the exploration time, the length of the trajectory) is optimized. In other words, the exploration strategy determines the next robot goal in each exploration iteration (one exploration step) with respect to the actual robot position, the current knowledge of the environment, and a selected optimization criterion.

Several exploration strategies have been proposed over last decades. The strategies differ in the way how candidates for the next goal are generated and in the criterion how the best candidate is selected. Yamauchi [1] introduced a frontier-based strategy that guides the robot to the nearest frontier, i.e. the boundary between a free and an unexplored space. It has been shown [2], [3] that this strategy produces reasonably short trajectories for graph-like environments with upper bound $O(|V|\log(|V|))$, where $|V|$ is the number of vertices of the graph. The authors of [4] discussed two simple heuristics improving Yamauchi's approach. The first one uses Voronoi diagrams to prefer exploration of the whole room in office-like environments before leaving it, while the second one repetitively re-checks whether the currently approached goal is still a frontier. When it is not, a new goal is determined. A strategy selecting the leftest candidate according to a robot position and orientation with a defined distance to obstacles is described in [5].

Other works generate several candidates in a free space (typically near to frontiers) and combine the distance cost (the utility evaluating effort needed to reach the goal) with other criteria. This concept has been introduced in [6] where measure $A(q)$ of an unexplored region of the environment, which is potentially visible from the candidate q, is combined with the distance cost $L(q)$ to get the overall utility of q:

$$g(q) = A(q)e^{-\lambda L(q)},$$

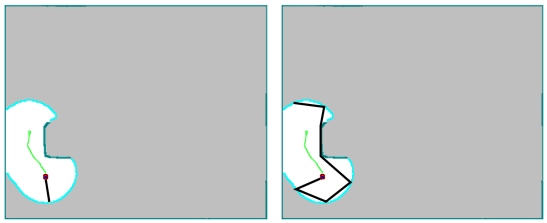
where λ is a positive constant. A utility of the next action as the weighted sum of the distance cost and expected information gain computed as a change of entropy after performing the action is presented in [7]. Another strategy taking into account the distance cost and the information gathered (based on the relative entropy) is introduced in [8] together with solid mathematical foundations. The strategy in [9] samples points near each candidate and filters samples according to selected criteria. The candidate with the highest number of samples that passed the filters is then chosen. Moreover, the localization utility can be integrated into the overall utility to prefer places traveling to them improves information about the robot pose [10]. Criteria forming the overall utility are not typically independent. General approach that reflects dependency among the criteria based on multi-criteria decision making is used in [11].

The aforementioned approaches evaluate the distance cost simply as the length of the trajectory from the current robot position to the next goal position. Such defined cost prefers candidates close to the robot without considering subsequent actions. In this paper, we present more sophisticated approach that is based on the observation that the robot should pass (or go nearby) all the goal candidates and define the distance cost for a candidate q as a minimal length of the path starting at the current robot position, continuing to the candidate q at first and then to all other candidates. We show that the introduced cost can reduce the exploration time significantly and leads to more feasible trajectories.

A similar approach is described in [12] where several exploration steps ahead are also considered. The state space of all possible paths consisting of several exploration steps is searched for the best alternative using the branch and bound algorithm. The branch and bound is a general technique that greedily searches relatively large state spaces without

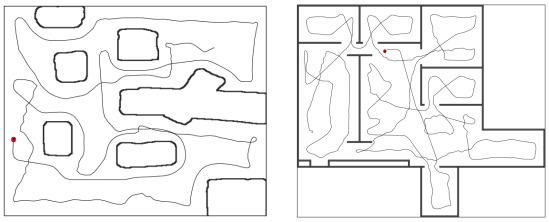
The authors are with Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technicka 2, 166 27 Prague 6, Czech Republic {kulich, xfaigl, preucil}@labe.felk.cvut.cz

978-1-61284-380-3/11/$26.00 ©2011 IEEE



(a) (b)

Fig. 1: Next goal selection by (a) the greedy approach, (b) taking into account all goal candidates.



(a) (b)

Fig. 2: Typical trajectories for the greedy approach.

a priory information about the solved problem. It can be therefore time consuming and the quality of found solutions heavily depends on the defined depth of pruning.

In our approach, we define the distance cost as the Traveling Salesman Problem (TSP). The problem formulation is described in the next section. Fast evaluation of the proposed distance cost is addressed by a heuristic algorithm for the TSP that finds a feasible solution quickly. Thus, the required computational time to solve the TSP is negligible in comparison to other parts of the exploration as it is shown in Section III presenting the experimental results. Finally, the concluding remarks are presented in Section IV.

II. Exploration with the TSP distance cost

Let the robot be equipped with a distance sensor with a fixed range (e.g. laser rangefinder) and the map the robot builds during exploration be modeled as the occupancy grid. The proposed exploration strategy is based on Yamauchi's frontier based approach. The key idea of the approach is to detect *frontier cells*, i.e. the reachable free grid cells (the cells representing free regions) that are adjacent with at least one cell that has not been explored yet. The *frontier* is a continuous set of frontier cells such that each frontier cell is a member of exactly one frontier. Once all frontiers are detected, the most appropriate frontier cell is selected as a new robot goal according to the defined criteria. This process is executed repeatedly at defined time steps until there is a frontier cell reachable by the robot.

As mentioned above, the current approaches compute the distance cost as the length of the path from the current robot position to the next goal, which can lead to selection of an inappropriate goal. An example of such a selection is demonstrated in Fig. 1a). In the shown situation, the robot moves down (the trajectory represented by the green curve) and a new goal has to be determined. The greedy approach selects the nearest frontier cell using the path showed as the black straight-line segment. It is obvious that in this situation a much better selection is to travel to the left first and then continue as it is illustrated in Fig. 1b).

Unfortunately, the described situation is not rare, and therefore the greedy approach produces superfluously long trajectories, see Fig. 2. To avoid this behavior, we propose a more informed approach to the distance cost using the TSP distance cost and consisting of two steps.

At first, frontier cells are filtered to get a set of representatives approximating the frontier cells such that each frontier cell is detectable by the robot sensor from at least one representative. This guaranties that all frontiers will be explored (i.e. it will be detected whether frontier lies in a free space or in any obstacle) after visiting all representatives. An algorithm for selection of representatives based on k-means is depicted in Algorithm 1.

Algorithm 1: Representatives selection for occupancy grids

Input: $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_n\}$ - the set of frontiers
Input: D - the range of the used sensor (in grid cells)
Output: $\mathcal{R} = \{r_1, \ldots, r_m\}$ - the set of representatives

$\mathcal{R} = \{\}$
foreach $\mathcal{Q}_I$ **do**
  Set an appropriate number of representatives:
  $$N = 1 + \frac{|\mathcal{Q}_I|}{2D}$$
  Find N means using k-means clustering:
  $$\mathcal{A} = \texttt{k-means}(\mathcal{Q}_I, N)$$
  $\mathcal{R} = \mathcal{R} \cup \mathcal{A}$

Having the representatives, the second step is to decide in which order they will be visited with respect to the minimal length of the traveled trajectory. Let the robot position be s_0, the set of representatives $\mathcal{S} = \{s_1, \ldots, s_n\}$, and $d(a, b)$ denote the length of the path between cells a and b. The aim is to find a permutation $\Pi = \{\pi_1, \ldots, \pi_n\}$ of $\mathcal{I} = \{1, \ldots, n\}$ such that $d(s_0, s_{\pi_1}) + \sum_{i=1}^{n-1} l(s_{\pi_i}, s_{\pi_{i+1}})$ is minimal over all permutations of $\mathcal{I}$. The representative s_{π_1} is then selected as the next robot goal.

The problem to find the best permutation is similar to the TSP that is known to be NP-hard [13]. While finding an optimal solution of the TSP can be computationally demanding, there are many approximate algorithms. One of the most powerful ones is the Chained Lin-Kernighan heuristic, which gives near-optimal results (up to 1% of the optimum) in a reasonable time [14]. The TSP can be defined on a graph $G(V, E)$, where V is a set of vertices and E are edges connecting the vertices and representing the connection cost. The objective is to find a closed tour with the minimal cost connecting all vertices in V.

A straightforward approach to formulate the permutation problem as the TSP is to construct $G(V, E)$, where V is the set of representatives, and E is the set of the all shortest

paths between the representatives. E can be computed by n calls of Dijkstra's algorithm on the adjacency graph of cells in the occupancy grid. Note that the TSP is formulated to find the best closed tour on the graph while a sequence of representatives ending in an arbitrary representative is requested for the distance cost. This discrepancy can be addressed by adding a fictive vertex s_∞ to V together with edges to all other vertices in V, whereas $d(s_\infty, s_0) = 0$ and $\forall i \in \{1, n\} : d(s_\infty, s_i) = \omega$, and ω is a large number (i.e. larger than the longest possible tour). This ensures that the TSP solver finds a solution where s_0 and s_∞ are neighbors in the tour. A solution of the permutation problem is found as a part of the tour starting from the s_0 and removing s_∞ and both its adjacent edges.

Algorithm 2: Frontier based exploration with the TSP distance cost.

repeat
- Get the updated map built from sensor readings
- Detect all frontiers from the actual map
- Select representatives
- Build the graph $G(V, E)$
- Solve the TSP on $G(V, E)$
- Set the neighbor of s_0 other than s_∞ as the next goal

until *accessible frontier found*

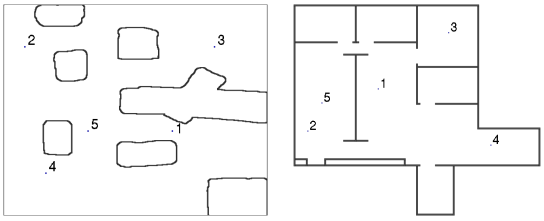
If the TSP distance cost is the only criterion for the goal selection it can be used as described above. An overview of the exploration procedure with this cost is depicted in Algorithm 2. In the case, the distance cost is combined with other costs (localizability, information gain) like in [10] [15], evaluation of the TSP distance cost is needed for each representative. Therefore a solution of the TSP for each representative s_j has to be found in order to compute the distance cost. The cost for s_j is determined using the graph $G_j(V_j, E_j)$ that is constructed in the following way:

1) $V_j = \{s_1, \ldots, s_n, s_\infty\}$, i.e. V_j does not contain s_0.
2) $d(s_\infty, s_j) = 0$ and $\forall i \in \{1, n\} \setminus \{j\} : d(s_\infty, s_i) = \infty$.
3) Costs of all other edges represent shortest paths between adjacent vertices.

In other words, G_j is constructed similarly to the previous case, however s_j has the role of s_0. The distance cost for s_j is simply computed as the sum of $d(s_0, s_j)$ and the length of the sequence created from the TSP result in the similar way as in the previous case.

III. EXPERIMENTAL RESULTS

Performance of the proposed distance cost has been evaluated and compared to the standard greedy approach in two types of environments in simulations using the Player/Stage framework [16]. The first one is an open space represented by the *cave* map, which has been scaled to 25×20 m. The second one is an office-like environment represented by a map of the Autonomy Lab (*autolab*) scaled to 35×35 m. The environments are visualized in Fig. 3. Five positions where the robot starts the exploration have been chosen for each environment as shown in Fig. 3 and described in Tab. I.



(a) Cave (b) Autolab

Fig. 3: Testing environments. The numbers correspond to the starting positions presented in Table I

All experiments were performed within the same computational environment: a workstation with the Intel®Core2 Duo CPU E6850 at 3 Ghz, 4 GB RAM running Sabayon 5.2 operating system with the Linux kernel 2.6.35. The algorithms have been implemented in C++ as client programs for the Player/Stage in version 3.0 and compiled by the GCC 4.4.2 with -O2 optimization flag. Simulation of the Pioneer 2DX robot equipped with SICK LMS200 with $180°$ field of view has been used as the robotic platform, while the occupancy grid with cell size 0.1×0.1 m has been chosen to represent the working environment. VFH+ algorithm [17] implemented in the Player has been used to control the robot motion and to avoid obstacles. The TSP solver used is the Chained Lin-Kernighan heuristic from the Concorde package [13].

TABLE I: Description of robot start positions in testing environments. The positions are in meters, the orientation of the robot is $0°$ for all positions.

Map	1	2	3	4	5
Cave	$[16, 8]$	$[2, 16]$	$[20, 16]$	$[4, 4]$	$[8, 8]$
Autolab	$[12, 18]$	$[2, 12]$	$[22, 26]$	$[28, 10]$	$[4, 16]$

The algorithm for the new goal selection (i.e. the body of the loop in Algorithm 2) is run every 1000 ms. It means that a new goal can be selected before the old one is reached. Moreover, the sensor range has been limited to 2, 3, and 5 meters. For each experimental setup consisting of the map, the starting robot position and the range, 30 runs have been performed for both the greedy approach and the proposed distance cost, which gives 1800 experiments in total. The particular experiment run took from 4 minutes for the *cave* map with 5 m range up to about 15 minutes for the *autolab* map and 2 m sensor range.

The experimental results are depicted in Tab. II for the *cave* map and in Tab. III for the *autolab* map. The solution quality is measured as the ratio of $avg_{TSP}/avg_{Greedy} \cdot 100\%$. Furthermore, the best found solutions of the proposed algorithm are shown in Fig. 6 and Fig. 7. The results show that the proposed TSP based approach outperforms the greedy selection in all cases. Generally speaking, the best improvement is achieved for smaller sensor ranges. The only

exception is for the open space and high sensor range where the trajectories generated by the TSP approach can be shorter up to 72% comparing to the greedy approach. The greedy approach is not able to explore the space systematically, it leaves some places unexplored and they have to be visited later, which is not the case of the TSP based approach. Another interesting observation is that the standard deviation for the TSP case is significantly smaller than for the greedy approach. It is an expected result, because the TSP is more robust to small local changes in representatives' positions as demonstrated in Fig 5.

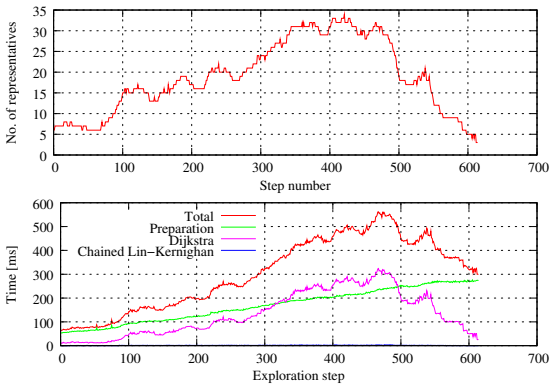


Fig. 4: The number of frontiers (upper) and the required computational time for the particular parts of the algorithm (bottom) during the exploration.

The required computational time of particular parts of the exploration algorithm in the *autolab* map and 2 m range is shown in Fig. 4. The blue curve, almost identical to the x-axis, denotes the computational time of the TSP solver. Regarding the times, the solution time of the TSP is negligible to other parts of the exploration algorithm.

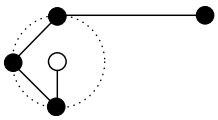


Fig. 5: If several goal candidates (the black disks) are in the similar distance to the robot and another one is far enough, small changes in goal candidates' positions do not change the shape of the TSP solution and thus the next goal will be preserved.

IV. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this paper, we present a novel approach to determine the distance cost in the exploration task. The key idea of the proposed approach is to select an appropriate set of goal candidates (representatives), which are expected to be visited by the robot. Then, the near optimal tour connecting all the representatives is found from which the next robot goal is determined. Although the introduced evaluation of the distance cost is primarily intended as a standalone criterion, a variant of the cost to be used as one of many criteria in complex systems has been presented as well.

A huge set of experiments has been performed in two different environments. These show that the presented method provides better results then the widely used greedy approach. The approach was presented for occupancy grids as the working environment representation. However a modification for a geometrical representation is straightforward.

B. Future Works

To the best of our knowledge there is no comprehensive comparison of exploration strategies in literature. Some attempts were made for example in [4], [11], [18], and [19]. Unfortunately, experiments in these papers are performed in different environments, with different robots and sensors and the number of experimental runs is relatively small. Moreover, the description of the experimental setup is not complete in many cases, which does not allow to repeat and compare described experiments. Therefore a detailed comparison of the current approaches (including the one presented in this paper), which can be repeated and enhanced by everyone, is one of our future goals.

The presented cost has been designed for a single-robot exploration. The next natural step is to extend the cost evaluation for the case of multiple robots that leads to solving a variant of the TSP called the Multiple Traveling Salesman Problem with MinMax criterion.

Finally, we would like to verify the results obtained in a simulation by experiments performed with real robots in the SyRoTek system [20].

V. ACKNOWLEDGMENTS

This work has been supported by the Ministry of Education of the Czech Republic, under the Project No. 6840770038.

REFERENCES

[1] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. of IEEE Int. Symposium on Computational Intelligence in Robotics and Automation*. IEEE Comput. Soc. Press, 1997, pp. 146–151.

[2] S. Koenig, C. Tovey, and W. Halliburton, "Greedy mapping of terrain," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, vol. 4, 2001, pp. 3594 – 3599 vol.4.

[3] S. Koenig, "Improved analysis of greedy mapping," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Las Vegas, NV*, 2003, pp. 3251–3257.

[4] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the Efficiency of Frontier-Based Exploration Strategies," in *Proc. of the Joint Conf. of the 41st Int. Symposium on Robotics and the 6th German Conference on Robotics*, Munich, Germany, Jun. 2010, pp. 36–43.

[5] Y. Mei, Y. hsiang Lu, C. S. G. Lee, and Y. C. Hu, "Energy-efficient mobile robot exploration," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2006.

[6] H. H. Gonzalez-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, Oct. 2002.

[7] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters," in *Proc. of Robotics: Science and Systems*, Cambridge, MA, USA, 2005.

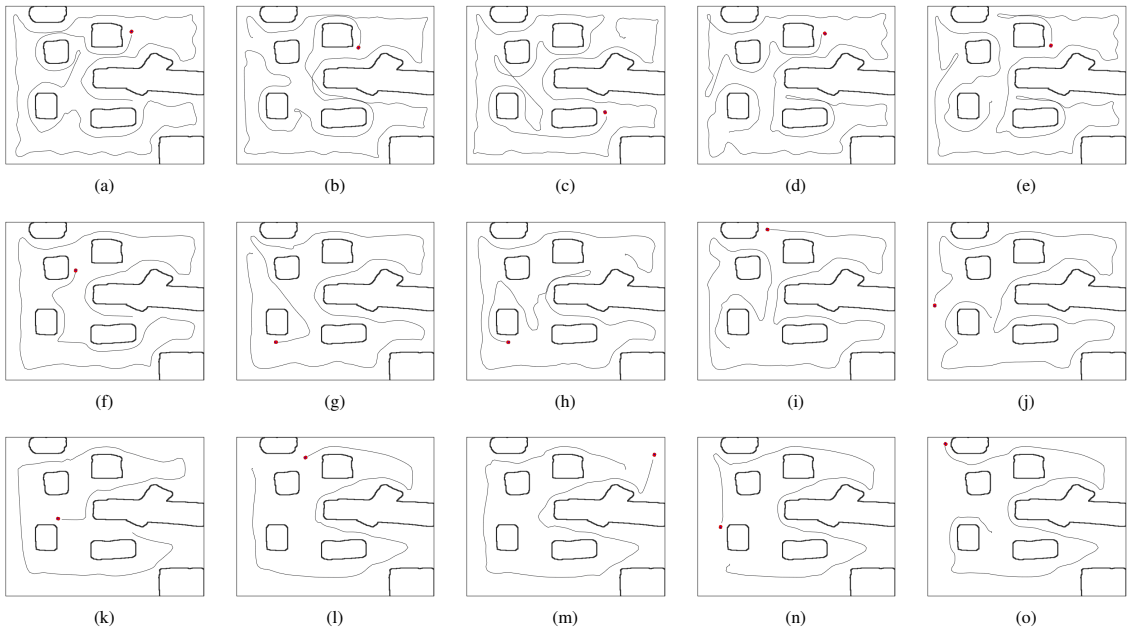


Fig. 6: The best results found with the TSP distance cost in the open space environment, the first row for $r = 2$, the second row for $r = 3$ and the third row for $r = 5$. The columns number equals to the position numbers.

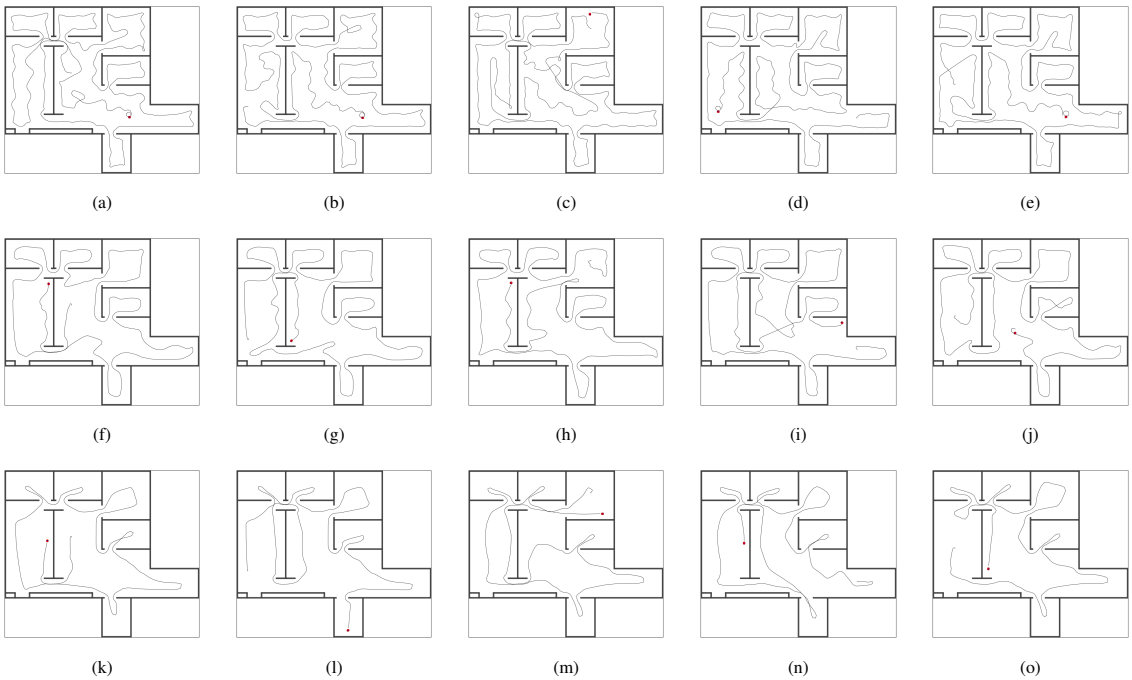


Fig. 7: The best results found with the TSP distance cost in the office-like environment, the first row for $r = 2$, the second row for $r = 3$ and the third row for $r = 5$. The columns number equals to the position numbers.

TABLE II: Comparison of the greedy approach and the proposed algorithm for cave environment. Pose numbers correspond to Table I.

Range	Pose	Greedy				TSP				Ratio
[m]	number	avg[m]	min[m]	max[m]	stdev[m]	avg[m]	min[m]	max[m]	stdev[m]	%
	1	212.13	184.94	239.39	13.49	174.66	168.18	181.30	3.77	82.34
	2	200.41	184.39	227.08	10.32	177.09	171.62	185.95	3.76	88.36
2.0	3	199.14	183.68	224.79	9.40	181.04	175.49	191.31	3.78	90.91
	4	206.98	187.35	226.92	9.99	184.86	172.13	195.83	5.65	89.31
	5	201.42	185.41	216.47	9.13	176.41	170.28	185.10	4.06	87.58
	1	146.59	125.72	166.32	10.94	129.49	122.91	136.52	2.69	88.33
	2	129.84	108.03	160.68	10.72	125.53	118.34	137.34	5.16	96.68
3.0	3	148.93	133.23	163.06	8.12	137.79	126.22	147.43	4.37	92.52
	4	148.21	121.36	183.64	14.16	132.06	125.69	143.60	5.87	89.10
	5	134.95	118.90	162.45	11.35	125.93	115.17	136.44	5.53	93.32
	1	116.43	99.87	129.80	6.05	84.01	80.69	89.77	2.08	72.16
	2	108.49	90.12	125.40	10.30	78.45	77.50	79.79	0.67	72.30
5.0	3	112.19	94.99	133.52	11.27	89.01	87.18	91.32	0.86	79.34
	4	110.69	99.64	124.45	10.95	88.32	80.66	104.97	5.87	79.79
	5	116.53	97.75	135.01	8.94	86.91	84.68	90.63	1.48	74.59

TABLE III: Comparison of the greedy approach and the proposed algorithm for autolab environment. Pose numbers correspond to Table I.

Range	Pose	Greedy				TSP				Ratio
[m]	number	avg[m]	min[m]	max[m]	stdev[m]	avg[m]	min[m]	max[m]	stdev[m]	%
	1	345.57	307.87	383.82	19.37	301.76	293.63	314.45	5.97	87.32
	2	355.02	315.52	404.69	21.32	300.44	281.57	314.96	7.57	84.63
2.0	3	351.16	320.93	385.65	16.98	311.48	302.10	320.74	4.97	88.70
	4	354.13	318.52	393.98	16.38	299.03	276.92	314.20	11.53	84.44
	5	343.95	311.52	378.76	17.37	295.06	277.32	308.13	8.27	85.79
	1	244.08	222.14	274.80	13.36	217.71	212.03	226.03	3.86	89.20
	2	248.44	233.99	267.84	10.07	219.82	215.36	224.43	2.24	88.48
3.0	3	247.51	222.44	276.87	13.92	231.25	221.22	237.13	3.36	93.43
	4	254.97	220.45	275.90	10.50	227.56	223.34	236.42	2.90	89.25
	5	251.67	235.38	284.67	11.94	223.20	217.37	234.58	4.78	88.69
	1	167.72	151.19	192.29	10.46	159.82	156.45	163.40	1.72	95.29
	2	165.77	150.70	183.28	9.71	157.57	151.89	171.67	4.46	95.06
5.0	3	182.63	164.36	198.81	10.12	165.42	162.81	168.11	1.56	90.58
	4	187.53	168.57	199.01	8.13	167.36	159.17	172.70	4.46	89.25
	5	163.26	147.85	185.09	10.93	153.77	149.32	167.67	5.12	94.19

[8] F. Amigoni and V. Caglioti, "An information-based exploration strategy for environment mapping with mobile robots," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 684 – 699, 2010.

[9] P. M. Newman, M. Bosse, and J. J. Leonard, "Autonomous feature-based exploration." in *IEEE Int. Conf. on Robotics and Automation*, Taiwan, Sep. 2003.

[10] A. A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte, "An experiment in integrated exploration," in *IEEE/RSJ Int. Conf. on Intelligent Robots and System*. IEEE, 2002, pp. 534–539.

[11] N. Basilico and F. Amigoni, "Exploration strategies based on multi-criteria decision making for an autonomous mobile robot," in *Proc. of the Fourth European Conference on Mobile Robots*. KoREMA, 2009, pp. 259–264.

[12] B. Tovar, L. Muñoz-Gómez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, and S. Hutchinson, "Planning exploration strategies for simultaneous localization and mapping," *Robotics and Autonomous Systems*, vol. 54, no. 4, pp. 314 – 331, 2006.

[13] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. (2010, Sep.) Concorde TSP Solver.

[14] D. Applegate, W. Cook, and A. Rohe, "Chained Lin-Kernighan for large traveling salesman problems," *Informs J. on Computing*, vol. 15, no. 1, pp. 82–92, 2003.

[15] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai, "A practical, decision-theoretic approach to multi-robot mapping and exploration," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 4, Oct. 2003, pp. 3232 – 3238 vol.3.

[16] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage project: Tools for multi-robot and distributed sensor systems," in *Proc. of the 11th Int. Conf. on Advanced Robotics*, 2003, pp. 317–323.

[17] I. Ulrich and J. Borenstein, "VFH+: Reliable obstacle avoidance for fast mobile robots," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, vol. 2, 1998, pp. 1572–1577.

[18] N. Basilico and F. Amigoni, "On evaluating performance of exploration strategies for an autonomous mobile robot," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Workshop on Performance Evaluation and Benchmarking for Intelligent Robots and Systems*, 2008.

[19] F. Amigoni, "Experimental evaluation of some exploration strategies for mobile robots," in *IEEE Int. Conf. on Robotics and Automation*, 2008, pp. 2818–2823.

[20] J. Faigl, J. Chudoba, K. Košnar, M. Kulich, M. Saska, and L. Přeučil, "SyRoTek - A Robotic System for Education," in *Proc. of Int. Conf. on Robotics in Education, Bratislava*. Bratislava: Slovak University of Technology in Bratislava, 2010, pp. 37–42.

2012 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

Goal Assignment using Distance Cost in Multi-Robot Exploration

Jan Faigl, Miroslav Kulich, Libor Přeučil

***Abstract*— In this paper, we discuss the problem of goal assignment in the multi-robot exploration task. The presented work is focused on the underlying optimal assignment problem of the multi-robot task allocation that is addressed by three state-of-the art approaches. In addition, we propose a novel exploration strategy considering allocation of all current goals (not only immediate goal) for each robot, which leads to the multiple traveling salesman problem formulation. Although the problem is strongly NP-hard, we show its approximate solution is computationally feasible and its overall requirements are competitive to the previous approaches. The proposed approach and three well-known approaches are compared in series of problems considering various numbers of robots and sensor ranges. Based on the evaluation of the results the proposed exploration strategy provides shorter exploration times than the former approaches.**

I. Introduction

The mobile robot exploration is a complex task in which a mobile robot is autonomously navigated in an unknown environment in order to create a map of the environment. The exploration can be defined as an iterative process determining a new goal for the robot and its navigation towards the goal. The process is terminated whensoever a complete map of the environment is created. Having a team of robots, an efficient allocation of exploration targets among the team is a natural way how to reduce the required time to collect information about an unknown environment.

The problem of Multi-Robot Exploration (MRE) is a kind of the Multi-Robot Task Allocation (MRTA) [1] in which tasks are new goal locations towards which robots are navigated. The fundamental way how to determine candidates for goal locations is the frontier based approach proposed by Yamauchi in 1998 [2] and further extended by many researchers later, e.g., see one of the recent work [3].

Having a set of candidate positions the robot's next goal can be determined regarding a selected criteria. A unifying concept of how to evaluate candidate positions is based on the goal utility. Although various utility functions have been proposed, all of them basically combine information gain (or expected benefit [4]) together with the required travelling distance to the goal [5]. Then, the robot's next goal is repeatably selected from goal candidates. Such an assignment of the next robot goal is called a next-best-view approach and it represents the fundamental stream in exploration [6].

The next-best-view approach can be formulated as the optimal assignment problem studied in operational research [1]. The problem is to find the best assignment of n goals to m robots maximizing the overall utility, i.e., to find one goal for each robot. The problem can be solved in polynomial time using the Hungarian algorithm. The algorithm has been applied to MRE in [7], where authors use Voronoi Graphs of the current known environment to explore a single room by one robot.

A distributed assignment algorithm called *Broadcast of Local Eligibility* (BLE) has been proposed in [8]. A pair $\langle robot, task \rangle$ with the highest utility is considered to assign the task to the robot without tasks. The BLE algorithm works iteratively until each robot has assigned a task; thus, the algorithm is also called iterative assignment.

Another stream of distributed MRE solutions is based on market (or auction) based approaches in which a robot (auctioneer) offers a task and other robots bid. If any robot bids with a higher price than the auctioneer's offer, the task is exchanged. This approach is used in [9], where a robot considers its goals in a tour and new (exchanged) goal is inserted into the tour regarding minimization of the tour's cost, i.e., the problem is a variant of the traveling salesman problem (TSP).

A selection of the next navigational goal considering the TSP distance cost has been studied in our previous work [10]. The cost is computed as the length of the shortest path connecting the robot with the candidate goal and all remaining goals. The Chained Lin-Kernighan heuristic [11] is considered to find a solution of the TSP, which provides sufficiently good solution without expensive computational requirements. Considering visitation of all current goals leads to about 30 percentage points shorter exploration path than using the standard greedy approach.

In this paper, we examine the TSP approach in MRE as the multiple traveling salesman problem (MTSP). The encouraging results presented in [10] motivate us to consider similar approach also in MRE; however, here, the key issue is how to determine and assign particular set of all goals to each robot in order to compute the distance cost as the length of the tour visiting all goals in the set using a solution of the related TSP. We propose to cluster the goals into m clusters first, where m is the number of robots. After that, a $\langle cluster, robot \rangle$ pair is evaluated using the TSP distance cost [10] for determining the next robot's goal.

The proposed approach is similar in the TSP aspect with the approach [9]. The main difference is that our approach is focused on the explicit MTSP formulation and the proposed solution is compared with the greedy approach [12], iterative assignment [8], and the Hungarian algorithm [7], which (to the best of our knowledge) has not been published yet.

Authors are with the dept. of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27 Prague, Czech Republic {xfaigl,kulich,preucil}@labe.felk.cvut.cz

978-1-4673-1735-1/12/S31.00 ©2012 IEEE

Moreover, during an experimental verification of the tested approaches we have found out that the studied performance metric significantly depends on particular components of the whole navigational system, especially on a local path planner. Therefore, inspired by the methodology described in [13] we designed a multi-robot exploration framework in which we can isolate the assignment problem and fix the navigation issues. In consequence, the framework provides same conditions for all evaluated methods during the whole exploration process. However, it is clear that real benefits of the exploration strategy should be verified in real experiments. Therefore, the methods have been also evaluated in selected problems using the Player/Stage framework [14], like in the aforementioned approaches.

The reminder of the paper is organized as follows. The problem statement is presented in Section II and a brief description of the examined methods in Section III. In Section IV, the MRE framework used in the evaluation is described. The proposed solution of the assignment problem based on the MTSP formulation is presented in Section V. Evaluation of the results and discussions of the MRE issues are presented in Section VI. Finally, Section VII is dedicated to concluding remarks.

II. Problem Statement

Although the evaluated approaches are general and not necessarily restricted to the particular sensors or map building techniques, we consider laser range finder sensor and occupancy grid approach for building a map of the unknown environment. The addressed problem of the multi-robot exploration (MRE) stands for building a map of the unknown environment using a team of m identical robots equipped with a laser range finder. The map $\mathcal{M}$ is formed from the occupancy grid using threshold values for probability that the grid's cell is occupied or free [2]. Thus, a cell in the navigational grid represents freespace, obstacle, or unknown part of the environment.

The exploration algorithm is an iterative procedure that is terminated once the navigational grid does not contain a reachable cell with an unknown value. At each exploration step, the robots' goals are determined from a set of candidate positions that are found as representatives of frontier cells.

The goals are assigned to robots using the exploration strategy that can be formalized as follows. *Let the current n goals be located at positions $\boldsymbol{G} = \{g_1, \ldots, g_n\}$ and the current robot poses be $\boldsymbol{R} = \{r_1, \ldots, r_m\}$. The problem is to determine a goal $g \in \boldsymbol{G}$ for each robot $r \in \boldsymbol{R}$ that will minimize the total required time, which can be approximated by the maximal travelled distance by an individual robot, to explore the whole environment.* The assignment is performed according to the particular strategy using defined utility and cost functions. In this paper, we consider only a distance cost $\mathcal{L}$ for evaluating the goal assignment; however, the examined assignment strategies are general and can also be used with a combined value of distance and utility costs.

For the standard strategies (described bellow) the distance cost $\mathcal{L}(g_i, r_j)$ (where $g_i \in \boldsymbol{G}$ and $r_j \in \boldsymbol{R}$) is the length of the shortest collision free path from the robot r_j to the goal g_i, e.g., found by the Distance Transform algorithm [15]. The proposed MTSP based assignment strategy utilizes the TSP distance cost [10].

In this paper, we consider the total distance travelled by a robot as the performance metric. The main motivation for utilizing several robots is expected reduction of the total required time to explore the whole environment, therefore we are looking for the maximal distance travelled as short as possible. Thus, having m robots with the distances travelled $l_1, l_2, \ldots, l_m$ the distance metric is $L = \max\{l_1, l_2, \ldots, l_m\}$.

III. Standard Goal Assignment Strategies

Greedy Assignment – The greedy assignment is based on the approach proposed by Yamauchi in [12]; however, it is modified to avoid assignment of the same goal to two robots because a centralized approach is considered here. The modification is that a random permutation of the robots $\Pi(\boldsymbol{R})$ is created first. Then, for each robot from $r \in \Pi(\boldsymbol{R})$ the best not assigned goal from $\boldsymbol{G}$ is found. The complexity of this assignment algorithm can be bounded by $O(nm)$.

Iterative Assignment – The iterative assignment follows the BLE algorithm [8], but for simplicity it is also implemented in a centralized manner. First, all robot-goal pairs $p = \langle r, g \rangle$ are created and ordered using the distance cost $\mathcal{L}$, i.e., $\mathcal{L}(p_1) \leq \mathcal{L}(p_2), \ldots \leq \mathcal{L}(p_l)$. After that, the ordered sequence is traversed starting from its first element, and the first not already used goal is iteratively assigned to a robot without the goal. The complexity of the iterative assignment can be bounded by $O((nm)\log(nm))$.

Hungarian Method – The Hungarian algorithm provides the optimal assignment of the n goals to m robots with the time complexity $O(n^3)$ for $n \geq m$. Similarly to the iterative assignment the cost matrix is determined using the distance cost $\mathcal{L}$, where rows stand for robots and columns for goals. In particular we consider the C implementation of the algorithm developed by Cyrill Stachniss [16].

IV. Multi-Robot Exploration Framework

Similarly to the approach [13] we consider a simulator for a focused investigation of exploration strategies. The simulator is based on a grid map that provides discrete timing of navigation and sensing operations. In particular, the motion consists of independent turning and moving steps using the grid cells (e.g., a robot visits all grid cells during its motion along a straight line segment) while the sensing is performed at each such a motion step. The framework also allows to easily switch the simulator with some robot control framework like Player/Stage or ROS; thus, MRE strategies developed can be easily deployed to control real robots.

A schema of the exploration loop is depicted in Algorithm 1. First, the initial robots surroundings are sensed and the occupancy grid is updated accordingly (Line 3). Then, the navigation grid is created from the occupancy grid and all frontiers are detected. The frontiers are particular grid cells (a set of freespace cells that are incident with cells with the unknown value using 8-neighbourhood) that form a

Algorithm 1: MRE Framework

Input: $\boldsymbol{R}$ - a set of m robots
Input: s_{max} - the maximal number of the performed navigation steps before new assignment
Output: $\mathcal{M}$ - a map of the explored environment

1 Initialization of the occupancy grid Occ
2 $\mathcal{M} = \emptyset$ `// the environment is unknown`
3 **foreach** $r_j \in \boldsymbol{R}$ **do** update(Occ, robot_sense(r_j))
4 **repeat**
5 $\mathcal{M} = \text{create\_navigation\_grid(Occ)}$
6 $\boldsymbol{F} = \text{detect\_all\_frontiers}(\mathcal{M})$
7 $\boldsymbol{G} = \text{determine\_representatives}(\mathcal{F})$
8 $(\langle r_1, g_{r_1}\rangle, \ldots, \langle r_m, g_{r_m}\rangle) = \text{assign\_goal}(\boldsymbol{R}, \boldsymbol{G}, \mathcal{M})$
9 fix_assignment($\boldsymbol{G}, \langle r_1, g_{r_1}\rangle, \ldots, \langle r_m, g_{r_m}\rangle$)
10 Create navigation plan $\boldsymbol{P}_i$ for each pair $\langle r_i, g_{r_i}\rangle$.
11 $l = \min\{|\boldsymbol{P}_1|, \ldots, |\boldsymbol{P}_m|\}$ `// the plan length`
12 $k = \min\{l, s_{max}\}$
13 **for** $i = 1..k$ **do**
14 **foreach** $r_j \in \boldsymbol{R}$ **do**
15 move($r_j, \boldsymbol{P}_j(i)$)
16 update(Occ, robot_sense(r_j))
17 **until** $|\boldsymbol{G}| == 0$

set of connected components. The goals for the assignment are found as representatives of the connected components using the K-means algorithm. The number of representatives n_r of a single component F with $f = |F|$ frontier cells is determined as

$$n_r = 1 + \left\lfloor \frac{f}{1.8D} + 0.5 \right\rfloor, \tag{1}$$

where D is the sensor range (in grid cells). A detailed description of the selection procedure can be found in [10].

Once the goals are determined the selected exploration strategy (Line 8) is used to assign a goal to each robot. It may happen that in the resulting assignment a robot can be without the associated goal, e.g., for $|\boldsymbol{G}| < m$. In such a case, the assignment is fixed (Line 9) using the closest goal to the robot, because it is desirable to utilize all the robots for the whole exploration period in order to minimize the total required time of the exploration (here, we assume the explored environment is a single connected component).

The assigned goals are used to determine the execution plan consisting of simple operations. The plan is then executed up to s_{max} steps (Lines 13–16). This part of the loop is replaced by adding goals to a local path planner if the algorithm is used with real navigation system, e.g., using the Player framework. Finally, the exploration loop is terminated if all reachable parts of the environment are explored.

It should be noted that the set $\boldsymbol{G}$ contains only representatives that are reachable by at least one robot, i.e., a collision free path exists in $\mathcal{M}$. The paths are found using the Distance Transform algorithm [15] that are then simplified by a greedy ray-shooting method using Bresenham's algorithm. The simplification does not affect the length of the path (on a grid) but the path is smoother.

V. Proposed MTSP based Assignment

The proposed exploration strategy is based on formulation of the goals' assignment problem as the MTSP. Having the set of goals $\boldsymbol{G}$ and m robots at positions $r_i \in \boldsymbol{R}$ for $i = 1..m$, the problem is to find m tours starting at the robots positions r_i such that each goal $g \in \boldsymbol{G}$ is contained in at least one tour and the length of the longest tour is minimal. It is known the MTSP problem is NP-hard, and therefore, we consider approximate solution of the problem based on an assignment of m clusters of the goals to robots, i.e., a kind of cluster-first, route-second heuristic approach. The proposed MTSP based assignment can be summarized in the following steps.

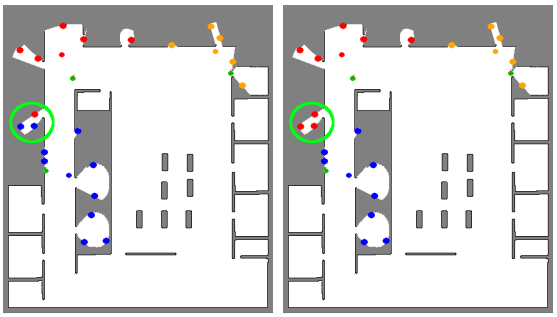
1) Find m clusters $\boldsymbol{C} = \{C_1, \ldots, C_m\}$, where $C_i \subseteq \boldsymbol{G}$.
2) Determine the TSP distance cost for each pair $\langle C_i, r_i\rangle$, where $C_i \in \boldsymbol{C}$ and $r_i \in \boldsymbol{R}$.
3) Extract the first goal $g \in \boldsymbol{G}$ of the TSP tour from each non-empty cluster C_i assigned to the robot r_i.
4) Fix goals' assignment if there is an empty set C_i.

Although the proposed clustering based solution of the MTSP is fairly common, we suggest (regarding the context of MRE) the following particular solutions of the clustering and a direct assignment of the clusters to the robots.

A. Goals Clustering

Various methods of clustering can be used. One of the popular algorithms is K-means; however, a regular variant of this algorithm is based on the Euclidean distance between samples. The distances between goals on frontiers in the map of the environment being explored are rather geodesic due to presence of obstacles (or missing information). Therefore, using the Euclidean distance provides clusters for which real paths to the goals are significantly longer than the expected.

Regarding this fact a more general variant of the K-means algorithm can be used, e.g., [17]. Alternatively, goals can be transformed to the Euclidean space where their mutual distances are preserved using SAMCOF (Scaling by Maximizing a Convex Function) [18]. Then, a regular K-means algorithm can be used. In this paper, we consider



(a) regular K-means (b) K-means with SAMCOF

Fig. 1. An example of the found clusters within the *jh* environment using a regular K-means algorithm on goals and transformed goals. The goals in the clusters are shown as red, blue, and orange disks. Unknown parts of the environment are in gray. The current robot positions are shown as green disks and the green circle highlights the effect of SAMCOF utilization.

the SAMCOF transformation and an example of the found clusters can be seen in Fig. 1. The K-means algorithm is initialized using position of the robots, and therefore, also the robots' positions are transformed using SAMCOF as well.

B. Fixing Goals' Assignment

The utilized initialization of the K-means algorithm results to the clustering where a cluster C_i is formed in the vicinity of r_i. Thus, the clustering prefers assignment of goals that are close to the robot, which follows the greedy strategy that is advantageous in the case of separable clusters and robots faraway each other. On the other hand, it may happen that two robots are close, and therefore, one cluster dominates over the other, which results in a situation when all the goals are in the first cluster and the second cluster is empty. In such a situation, it is important that the robot with an empty cluster is moved towards unexplored part of the environment. Otherwise (e.g., when the robots stops its motion), it will be more and more far from new goals, which may result the robot will not actively participate on the exploration.

Various strategies how to determine a goal for a robot with an empty cluster can be proposed. Regarding the considered TSP distance cost, we proposed to assign a goal according to the expected time when the goal will be visited. Therefore, once a tour visiting all goals in the cluster is determined for each robot r_i with $C_i > 0$, a length of the path from the particular robot to the goal (along the tour) denotes the expected time of visit. Then, the goals are ordered using the time and goals with higher times are sequentially assigned to the robots without already assigned goal.

This procedure assigns a goal to each robot, and therefore, it replaces the `fix_assignment` in Algorithm 1 (Line 9).

VI. Results

Three standard approaches and the proposed MTSP based approach have been evaluated in the developed MRE framework first. The framework allows focused study of exploration strategies that can be fully controlled by the trial setup. Herein presented experimental evaluation has been performed using a map of the environment (called *jh*) representing a real administrative building with dimensions 21 m×24 m. The environment is large enough to exhibit performance of the MRE using several robots while it also contains cycles and long corridors with several rooms. Thus, it provides representative office-like environment for verifying feasibility of the proposed MTSP strategy.

We followed recommendations of benchmarking the exploration strategies presented in [13] and considered small perturbations in the initial positions of the robots forming 20 variants of each problem defined by the sensor range ρ, the number of robots m, and the maximal planning period given by s_{max}. The iterative assignment and Hungarian exploration strategies are completely deterministic, while the Greedy and the proposed MTSP methods are stochastic. Therefore for each problem variant a single trial is considered for the deterministic ones and 20 trials are performed for the stochastic methods. The studied performance metric is then computed over all perturbations (problem variants) and trials as average values (denoted as L) and standard deviations (s_L). All presented distance values are in meters.

The used sensor is a laser range finder HOKUYO with 270° field of view. The occupancy and navigational grids (map) have identical dimensions with the cell size 0.05 m×0.05 m.

A. Comparison of the Assignment Strategies

The exploration strategies are compared using s_{max}=7 that provides a good trade-off between the quality of solution and computational requirements, see Fig. 3. The considered numbers of robots are $m \in \{3, 5, 7, 10\}$ and the sensor range is selected from the set $\rho \in \{3, 4, 5\}$ meters, which results in 10 080 trials in total for this evaluation.

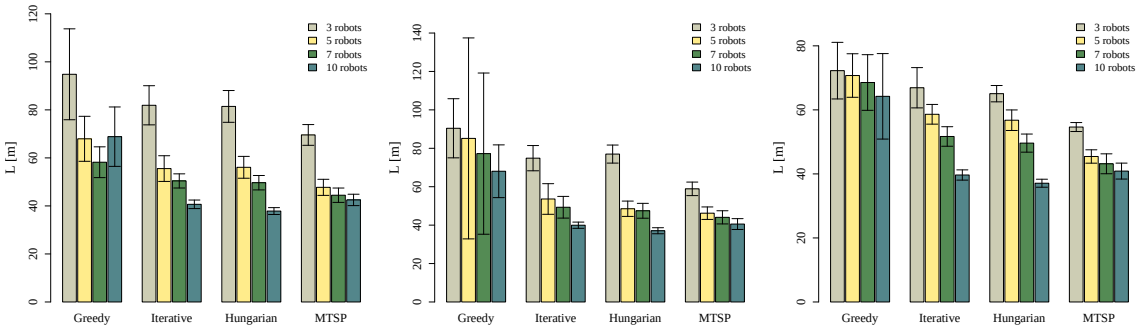
TABLE I
MAXIMAL TRAVELLED DISTANCE, s_{max}=7

ρ	m	**Greedy**		**Iterative**		**Hungarian**		**MTSP**	
		L	s_L	L	s_L	L	s_L	L	s_L
3.0	3	94.8	18.9	81.9	8.1	81.4	6.6	69.6	4.3
3.0	5	68.0	9.4	55.6	5.3	56.1	4.5	47.7	3.3
3.0	7	58.2	6.4	50.4	3.0	49.7	3.0	44.5	3.0
3.0	10	68.8	12.4	40.7	1.8	37.9	1.4	42.5	2.4
4.0	3	90.4	15.4	74.9	6.6	77.0	4.7	58.9	3.5
4.0	5	85.2	52.3	53.6	8.0	48.5	4.0	46.2	3.3
4.0	7	77.2	42.0	49.3	5.7	47.5	3.9	44.1	3.4
4.0	10	68.0	13.7	40.0	1.6	37.1	1.6	40.6	2.8
5.0	3	72.2	8.8	66.9	6.3	65.1	2.5	54.6	1.4
5.0	5	70.7	6.8	58.6	3.1	56.8	3.2	45.4	2.1
5.0	7	68.6	8.7	51.7	3.0	49.6	2.8	43.2	3.1
5.0	10	64.2	13.4	39.7	1.6	37.1	1.2	40.9	2.5

The results are shown in Fig. 2 and detailed results in Table I. The MTSP provides shorter exploration paths than other strategies; however, with increasing m the benefits of the MTSP strategy is not evident from the average values. Therefore we performed statistical evaluation using a null hypothesis that the algorithms provide statistically identical results. We consider the Wilcoxon test for the evaluation, because we assume the distributions are not Gaussian (based on the Shapiro-Wilk test).

The strategies are considered different if the P-values obtained by the Wilcoxon test are less than 0.001, which indicates the difference between L is statistically significant and a strategy providing lower L is considered as providing better results. Results of the statistical comparison are shown in Table II. All the p-values are very small, therefore characters '–', '+', and '=' are used to denote that the particular strategy provides longer, shorter or statistically identical L.

Regarding the results the considered range does not significantly affect L because of relatively small open parts while the rooms must be explicitly visited. Notice the standard deviation s_L for the greedy strategy. It indicates the performance is varying and sometimes the solution can be very close to the solution found by the MTSP. However, in average, it is worse than all other strategies. Although, the Greedy strategy is stochastic, we found that the performance



(a) ρ=3 m (b) ρ=4 m (c) ρ=5 m

Fig. 2. Scaling of the exploration strategy with the increasing number of robots and for laser range ρ.

depends on the initial conditions, i.e., a small perturbation leads to significantly different performance. This is not the case of the other methods, and especially for the MTSP, which provide more stable solutions as s_L is low.

TABLE II
COMPARISON OF THE MRE STRATEGIES

ρ	m	**Iterative vs Greedy**	**Hungarian vs Iterative**	**MTSP vs Hungarian**
3.0	3	+	=	+
3.0	5	+	=	+
3.0	7	+	=	+
3.0	10	+	+	-
4.0	3	+	=	+
4.0	5	+	=	=
4.0	7	+	=	+
4.0	10	+	+	-
5.0	3	+	=	+
5.0	5	+	=	+
5.0	7	+	=	+
5.0	10	+	+	-

1) Influence of the planning period: The influence of s_{max} to the performance of the exploration is depicted in Fig. 3. The results have been obtained for m=7 and ρ=3 m, and for each value of s_{max} all problem variants have been considered as well as 20 trials for each problem variant and the stochastic strategy. The total number of the performed trials in this evaluation is 14 280. The results indicate that a smaller value of s_{max} generally provides better results, but for all tested values of s_{max} the proposed MTSP method provides superior results. Regarding the required computational time the simple greedy or iterative strategies are computationally less intensive, but due to a longer exploration time, all the methods are competitive in the total required computational time.

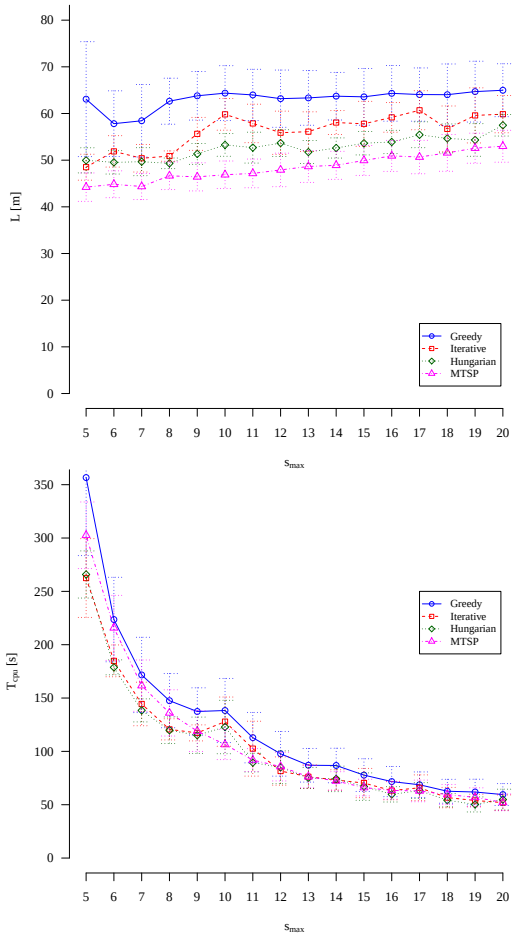


Fig. 3. Influence of the planning period s_{max} to the exploration performance for m=7 and ρ=3 m; *upper*: average values of the maximal travelled distance; *bottom*: average values of the required computational times using C++ and a workstation with 3.2 GHz CPU running FreeBSD 9.

B. Results using real navigational framework

Performance of the tested exploration strategies have also been evaluated using Player/Stage framework, in which additional components of the navigational architecture play role. The robot configuration and the environment is same as in the previous tests. The main difference is that a robot is controlled using the SND driver [19] for the robot motion and obstacle avoidance.

TABLE III
PERFORMANCE OF THE STRATEGIES USING PLAYER/STAGE AND ρ=3 m

m	Greedy		Iterative		Hungarian		MTSP	
	L	s_L	L	s_L	L	s_L	L	s_L
3	73.8	9.8	66.5	4.6	65.2	4.8	56.0	7.1
5	55.5	9.3	47.0	8.2	49.2	9.0	49.2	7.0

In this test, the replanning frequency has not been restricted, i.e., it is limited only by the hardware used. Therefore, the presented results provide estimation of the real benefits of more computational demanding methods over simple and faster methods. The results are shown in Table III.

C. Discussion

The presented results indicate that the proposed MTSP method provides more efficient tasks allocation than the former approaches. Hence, the results support the idea to consider a longer planning horizon rather than just an immediate goal. However, for ten robots the benefit of the method is not evident from the presented results. It is probably due to a relatively small environment and the used clustering initialized by the robots' position, which can lead to few dominant clusters and a greedy assignment.

Performance of the Hungarian and Iterative strategies is very similar, therefore, the main advantage of the Iterative strategy is its simpler implementation. Besides, the Iterative strategy can also be easily deployed in a distributed environment, which is not the case of the Hungarian algorithm. On the other hand, computational requirements of the more sophisticated Hungarian and MTSP approaches are competitive to the simple greedy algorithm; thus, they should be preferred in the applicable scenarios.

During the experimental evaluation, we have noticed, the navigational framework, in particular the local planner, affects the performance of the exploration. This is mostly visible in a situation where a robot is approaching a narrow passage (e.g., doors), where its velocity is slow. Differences in the robot average velocities affect the total required time. Therefore, the expected distance cost to reach the goal is only approximation, which in consequence means that the exploration strategy does not provide the expected benefit.

VII. CONCLUSION

In this paper, we further developed our previous work on exploration strategy using the TSP distance cost to the multi-robot exploration. The proposed strategy is compared with three standard approaches and the results show the proposed novel multi-robot exploration provides better results while its total computational requirements are competitive. Although only relatively small number of robots has been considered, the results indicated that for a higher number of robots the Iterative and Hungarian algorithms provides similar results to the proposed MTSP based strategy.

Regarding the found insights, the real performance of exploration is not affected only by the used strategy, but also by the low-level motion control. Therefore, we are aiming to consider more realistic estimation of the travelling cost towards the goal in the assignment problem. Besides, we also intend to evaluate different exploration strategies using real robots to verify the results presented in this paper.

ACKNOWLEDGMENTS

This work has been supported by the Technology Agency of the Czech Republic under Project No. TE01020197 and by the Ministry of Education of the Czech Republic under Project No. LH11053.

The access to computing and storage facilities provided under the National Grid Infrastructure MetaCentrum, provided under project No. LM2010005 funded by Ministry of Education of Czech Republic is highly appreciated.

REFERENCES

[1] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. of Robotic Research*, vol. 23, no. 9, pp. 939–954, 2004.

[2] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, ser. AGENTS '98. NY, USA: ACM, 1998, pp. 47–53.

[3] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the efficiency of frontier-based exploration strategies," in *ISR/ROBOTIK*, 2010, pp. 1–8.

[4] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, Jun. 2005.

[5] F. Amigoni and V. Caglioti, "An information-based exploration strategy for environment mapping with mobile robots," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 684–699, 2010.

[6] N. Basilico and F. Amigoni, "Exploration strategies based on multi-criteria decision making for searching environments in rescue operations," *Autonomous Robots*, vol. 31, no. 4, pp. 401–417, 2011.

[7] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 1160–1165.

[8] B. B. Werger and M. J. Mataric, "Broadcast of local eligibility for multi-target observation," in *Distributed Autonomous Robotic Systems 4*, L. E. Parker, G. Bekey, and J. Barhen, Eds. Springer-Verlag, 2001, pp. 347–356.

[9] R. Zlot, A. Stentz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, 2002, pp. 3016 –3023.

[10] M. Kulich, J. Faigl, and L. Preucil, "On distance utility in the exploration task," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 4455–4460.

[11] D. Applegate, W. Cook, and A. Rohe, "Chained Lin-Kernighan for large traveling salesman problems," *Informs J. on Computing*, vol. 15, no. 1, pp. 82–92, 2003.

[12] B. Yamauchi, "Decentralized coordination for multirobot exploration," *Robotics and Autonomous Systems*, vol. 29, pp. 111–118, 1999.

[13] F. Amigoni, "Experimental evaluation of some exploration strategies for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, may 2008, pp. 2818 –2823.

[14] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage project: Tools for multi-robot and distributed sensor systems," in *Proc. of the 11th Int. Conf. on Advanced Robotics*, 2003, pp. 317–323.

[15] R. A. Jarvis, "On distance transform based collision-free path planning for robot navigation in known, unknown and time-varying environments," in *Advanced Mobile Robots*, Y. F. Zang, Ed. World Scientific Publishing Co. Pty. Ltd., 1994, pp. 3–31.

[16] C. Stachniss, "C implementation of the hungarian method," 2004, [cited 29 Feb 2012]. [Online]. Available: http://www.informatik.uni-freiburg.de/~stachnis/misc/libhungarian-v0.1.2.tgz

[17] N. Asgharbeygi and A. Maleki, "Geodesic K-means clustering," in *19th Int. Conf. on Pattern Recognition (ICPR)*, 2008, pp. 1–4.

[18] A. Elad and R. Kimmel, "On bending invariant signatures for surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1285 – 1295, oct. 2003.

[19] J. W. Durham and F. Bullo, "Smooth nearness-diagram navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 690–695.