CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Cybernetics

Doctoral Thesis

Robot Learning and Perception in Sensory Deprived Environment

Vojtěch Šalanský

Supervisor: Karel Zimmermann
Ph.D. Programme: P2612 Electrical Engineering and Information Technology
Branch of study: Artificial Intelligence and Biocybernetics
February 2022, Prague

Acknowledgments

I want to thank my supervisor, Karel Zimmermann, for the opportunity he gave me. He was able to light up my passion for knowledge, and his supervising style fitted well to my needs. I am thankful to him and Tomáš Svoboda for the guidance during my doctoral studies and for the senior approach to our research, which led to several publications we did together. The other people I want to thank are Tomáš Petříček, Martin Pecka, Vladimír Kubelka and Michal Reinštein, because they participated to some parts of the research I did. I want to thank also the rest of the VRAS group for the friendly environment and the discussions we had that inspired my work and ideas. I also want to thank all the team members of CTU-CRAS-NORLAB for their efforts to create the system that competed on DARPA Subterranean Challenge and for the time we spent together during the whole challenge. Last but not least, I want to thank the Department of Cybernetics and the Czech Technical University in Prague for the opportunities and environment, they gave me during the time I spent at my "Alma mater".

Over the years my research was funded by: Grant Agency of the CTU[1], Czech Science Foundation grants[2], Research Center for Informatics[3], European Union[4], and Defense Advanced Research Projects Agency (DARPA).

[1] SGS15/081/OHK3/1T/13,
SGS16/161/OHK3/2T/13,
SGS18/138/OHK3/2T/13,
SGS20/128/OHK3/2T/13

[2] 14-13876S,
17-08842S,
20-29531S

[3] CZ.02.1.01/0.0/0.0/16_019/0000765

[4] 609763 TRADR,
692455 Enable-S3

Abstract

Mobile robots have become a standard part of search-and-rescue missions in recent years. Disaster environments push the robots to face extreme conditions from the sensory (low illumination, dust, or presence of dense smoke) and mobility point of view (complex and often nonrigid terrain). Deployment of the methods that help robot's perception (terrain estimation, object segmentation) and control (terrain traversal, active perception) in such an environment is the main focus of this dissertation thesis. Knowing the robot's surroundings is a crucial feature for many robotics tasks. We present the self-supervised learning of terrain shape estimation from the robot's traversal trajectories. Our novel loss function forces the terrain prediction to be consistent with the robot poses (the robot is not in the collision and has sufficient support). An extreme case of terrain prediction without any exteroceptive data is also discussed. Moreover, we describe our active perception method that controls the special depth sensor to obtain better terrain reconstruction. Victim detection and segmentation in search-and-rescue missions are crucial tasks that motivated us to develop an active perception method for learning control and segmentation from multi-modal data. The victim segmentation accuracy is improved by cleverly controlling the pan-tilt thermal sensor unit with the limited field of view. Traversing the harsh terrain is a challenge in these scenarios. The reinforcement learning procedure of terrain traversal needs many real rollouts, which can endanger the robot. To tackle this issue, we propose a constrained policy search method that considers safety; it prevents the robot from being damaged and helps the faster convergence. The last part of this dissertation thesis describes our participation in the DARPA Subterranean Challenge, which tested the ability of our developed system to work in actual search-and-rescue-like missions.

Keywords: Scene segmentation, object detection, active perception, sparse to dense prediction, robot-terrain interaction, reinforcement learning

Abstrakt

Mobilní roboty se v posledních letech stávají běžnou součástí záchranářských misí. Během nasazení je robot testován v extrémních podmínkách, jak ze senzorického hlediska (nízké osvětlení, prach nebo přítomnost hustého kouře), tak z hlediska pohybového (komplexní a často nerigidní terén). Metody vnímání (odhad terénu, segmentace scény) a metody řízení (překonání terénu, aktivní vnímání) pro roboty v extrémním podmínkách jsou hlavními tématy této práce. Znalost okolního terénu je klíčová pro mnoho robotických úloh. V rámci této práce prezentujeme metodu učení odhadu tvaru terénu z průjezdu robotu terénem. Navrhujeme novou ztrátovou funkci, která vynucuje tvar terénu konzistentní s pozicemi robotu (robot není v kolizi s terénem a je dostatečne podepřen). Dále navrhujeme metodu odhadu tvaru terénu pro extrémní případ, kdy robot nemá žádná exteroceptivní data. Tato práce se také zabývá metodou aktivního řízení speciálního hloubkového senzoru pro lepší rekonstrukci scény. Jednou z důležitých úloh záchranářské robotiky je detekce objektů a segmentace prostředí. V rámci této disertační práce je prezentována úloha aktivní segmentace prostředí z multimodálních dat. Pomocí řízení otočného termálního senzoru s nízkým zorným polem zvyšujeme přesnost detekce obětí. Navíc je zde popsána metoda učení robotu překonávat složitý terén, se zárukami na bezpečnost trajektorii během učení. Přidáním bezpečnostních omezení navíc dosahujeme i rychlejšího učení. Poslední část disertační práce popisuje naší účast na soutěži DARPA Subterranean Challenge, která otestovala algoritmy i schopnost vyvíjeného systému fungovat v reálném nasazení.

Klíčová slova: Segmentace scény, detekce objektů, aktivní vnímání, odhad hloubkových dat, interakce robotu s terénem, posilované učení

Překlad názvu: Učení a vnímaní robotu v částečně neznámem prostředí

Contents

Figures

Tables

Chapter 1

Introduction

Robotics as a research field deals with different exciting problems from the design and construction of robots and computer systems to motion control and perception. Nowadays, robots serve not only in almost every factory, but we also use the robots at our homes to make our lives easier and safer. This thesis focuses mainly on mobile robots that can move and interact in the surrounding environment. The pushing of the boundaries of robotics allows us to use mobile robots in more challenging environments than our living room is. Therefore, robots are increasingly used in automotive, search-and-rescue scenarios, or military missions. The robots, for example, serve as an extension of first-responders in the search-and-rescue missions, which allows them to access dangerous areas quickly, locate the potential victims or send the essential measured data, which can be used to estimate the state of the disaster area before the human rescuers risk their own lives. Robots operate in disaster sites with often really harsh conditions.

Search-and-rescue scenarios yield many new challenges to perception and motion control. The complex environment can contain almost everything (e.g., severe obstacles, unstructured terrain, staircases, mud, sand, water), making the robot's mobility trouble. Learning of the traversal of such an environment can endanger the robot. Such an example of a hard-to-traverse environment is depicted in Fig. 1.1. The other problem we struggle with within the search-and-rescue scenarios is sensor measurements deprivation. The presence of dense smoke can completely blind the sensors of the robot that are essential for a successful mission. LiDAR (Light Detection And Ranging) sensors can also suffer in some other environments: surfaces with low spectral absorbance or high reflectance can cause missing measurements in the surrounding pointcloud of the robot (for example, water surface).

The deprivation of depth measurements complicates robot localization and safe navigation. Visually dependent tasks are aggravated in these scenarios as well. Low illumination or over-illumination brings new challenges to object detection or victim segmentation tasks. The particular examples of the sensor-unfriendly environment are shown in Fig. 1.2. The research behind this thesis contributed to both of the previously mentioned parts. We tackle the sensory deprived perception problems as well as motion control challenges. This thesis is divided into chapters where each one is struggling with a slightly

Figure 1.1: Example of search-and-rescue environment. Damaged church after the earthquake in Amatrice in Italy 2016. **Left:** RGB image captured by onboard cameras. **Right:** colored pointcloud obtained using rotational Sick LiDAR.

different task, from terrain reconstruction, over active perception, and motion control, to the actual search-and-rescue deployment.

Chapter 2 is motivated by our observation, that the shape of the terrain that supports the robot during its traversal – *the supporting terrain* – is an essential input for many subsequent procedures such as motion control or path planning. Since this shape cannot be measured directly, we instead learn a convolution network to predict it from lidar and camera measurements. The prediction is not straightforward since the measurements are often incomplete due to the terrain reflectivity or biased due to the terrain occlusion by a flexible (not-supporting) layer such as vegetation or water. In addition to that, it is complicated to obtain a sufficient amount of manual annotations for any fully-supervised training. Consequently, we focused on designing a self-supervised method that learns to predict the shape of supporting terrain from offline-optimized maps and robot trajectories. Since offline optimization has access to privileged information in the form of future measurements, the resulting ground truth is significantly better than what can be measured in the time of inference. While the learning from the ground-truth maps directly leads to minimizing the cross-entropy or L2 loss, the learning from ground-truth trajectories captured during the terrain traversal is non-trivial. To this end, we proposed so-called *KKT-loss* that allows us to backpropagate from ground-truth trajectories to the predicted terrain shape by measuring the physical consistency between them. The KKT-loss leverages a simple first principle model of the robot-terrain interaction that places the robot trajectory into the local minimum of its potential energy. The physical consistency is then measured as the Euclidian distance from the Karush–Kuhn–Tucker necessary conditions of this first principle model. The resulting fully-differentiable KKT-loss thus provides the additional self-supervision signal, which helps significantly, especially in cases where usual lidar fails, such as non-rigid or non-Lambertian terrain surface. The next issue we tackle in this chapter is a tactile terrain reconstruction method for the extreme case of visual and depth deprivation. In such a scenario, a robot cannot collect any valid measurements

Figure 1.2: Left: robot in dense smoke. **Middle:** robot in the dark cave staying in the water. **Right:** over-illuminated survivor in the image.

(e.g., due to dense smoke). Our method presented a combined hardware and software solution to this issue. We designed a custom-built force sensor assembled inside of the robot tracks. The robot's pose and information from the force sensors are then used to train the terrain shape prediction model.

In Chapter 3, we consider the problem of active perception with two completely different sensors: (i) the solid-state lidar that allows controlling depth-measuring rays independently, and (ii) the thermal pan-tilt camera. The solid-state lidar is used on the problem of active 3D map reconstruction. The main challenge that arises from the control of the lidar stems from the dimensionality of the action space. We tackle this problem by designing a novel method that jointly learns to predict the 3D map by the convolutional neural network and plan trajectories of the depth-measuring rays that will improve the accuracy the most. The key challenges stem from the fact that whenever we change the parameters of the map-predicting network, the result of planning is different, which also changes the training data for learning. Consequently, we cannot keep learning on the same inputs due to the distribution shift. Therefore, we iterate the process until a fixed point solution is found. The pan-tilt thermal camera is used on the problem of active victim segmentation. Our novel learnable method deploys the convolutional neural network, which simultaneously (i) segments victims from incomplete multi-modal data and (ii) actively controls the pan-tilt thermal camera to maximize the victim segmentation accuracy. This task is complex due to the limited number of real-world robot interactions for learning. To overcome this issue, we propose a self-supervised initializing of the network and optimization-based guiding of motion control learning.

Chapter 4 describes the motion control task of traversing obstacles. As it can be seen in Fig. 1.1-right and Fig. 1.2-left, the robotic platform we are working with is equipped with four subtracks called flippers which could be used to climb a complex terrain. For the operator teleoperating the robot is almost impossible to control the flippers during robot navigation; therefore, we had developed a reinforcement learning algorithm that teaches the robot to control the flippers based on the terrain shape information obtained from the aforementioned methods. Learning the policy that will lead to safe terrain

traversal can take a lot of real robot evaluation. Some of the trials may endanger the robot system or cause rapid wear. Our proposed policy search method considers safety constraints, which have to hold not only for the final policy but for the whole learning process. These additional constraints help the policy to converge safer and also faster.

The common issue behind robotic research is reproducibility. The direct and fair comparison between the published methods is often complicated due to different hardware setups or datasets. Fortunately, DARPA (Defense Advanced Research Projects Agency) organized a grand robotic challenge that allows us to deploy and test our developed systems in actual search-and-rescue-like missions and compete with the best researchers in the field of robotics. The name of the challenge was DARPA Subterranean challenge, and the objective was to send a team of semi-autonomous robots into the various underground environments to find and locate defined artifacts. We participated in this challenge, and Chapter 5 briefly describes our approach and concludes the result we achieved.

1.1 Selected publications and personal contribution

Selected publications that led to the completion of this thesis are mentioned in this section. The main ideas behind this thesis were published in the Q1 journals or top-class conferences (A* and A CORE ranking). My personal contribution to each publication is emphasized in the list below.

Articles in peer-reviewed journals

- Vojtěch Šalanský, Karel Zimmermann, Tomáš Petříček, and Tomáš Svoboda. “Pose Consistency KKT-Loss for Weakly Supervised Learning of Robot-Terrain Interaction Model”. In: *IEEE Robotics and Automation Letters* 6 (July 2021), pp. 5477–5484
  - Q1 robotic journal with impact factor 3.74.
  - Personal contribution: design and implementation of the pipeline and neural network for supporting terrain prediction, collecting dataset, and training the network. Evaluating all the experiments.
  - 40% authorship.
- Tomáš Petříček, Vojtěch Šalanský, Karel Zimmermann, and Tomáš Svoboda. “Simultaneous exploration and segmentation for search and rescue”. In: *Journal of Field Robotics* 36.4 (2019), pp. 696–709
  - Q1 robotic journal with impact factor 3.77.
  - Personal contribution: design and implementation of the pipeline, design of the neural network for active control, learning of the model, and real platform experiments.
  - 35% authorship, equal contribution within first two authors.

- Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanský, et al. *System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA Subterranean Challenge*. 2021. arXiv: `2110.05911 [cs.RO]`
  - New journal.
  - Personal contribution: object detection, human operator, system integration.
  - 5% authorship.

Conference proceedings

- Karel Zimmermann, Tomáš Petříček, Vojtech Šalanský, and Tomáš Svoboda. "Learning for Active 3D Mapping". In: *2017 IEEE Int. Conf. on Comput. Vision (ICCV)*. 2017, pp. 1548–1556
  - A* CORE Rank conference in computer vision. Average acceptance rate is 25 %. This publication was accepted for the oral presentation where the acceptance rate was 2.6 %.
  - Personal contribution: design and implementation of the pipeline, design of the neural network for 3D reconstruction from sparse measurements, learning of the model, and all experiments.
  - 20% authorship.
- Martin Pecka, Vojtěch Šalanský, Karel Zimmermann, and Tomáš Svoboda. "Autonomous flipper control with safety constraints". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2889–2894
  - A CORE Rank conference in robotics. Average acceptance rate is 45 %. Equal contribution.
  - Personal contribution: theory behind constrained reinforcement learning technique and experiments.
  - 25% authorship, equal contribution.
- Vojtěch Šalanský, Vladimír Kubelka, Karel Zimmermann, Michal Reinstein, and Tomáš Svoboda. "Touching without vision: terrain perception in sensory deprived environments". In: *21st Computer Vision Winter Workshop (pp. 1-9)*. 2016
  - International conference in computer vision.
  - Personal contribution: whole pipeline and method for terrain profile estimation. Experiments.
  - 60% authorship.

- Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanský, et al. "DARPA Subterranean Challenge: Multi-robotic Exploration of Underground Environments". In: *Modelling and Simulation for Autonomous Systems*. 2020, pp. 274–290. ISBN: 978-3-030-43890-6
  - International conference focused on autonomous systems.
  - Personal contribution: object detection, human operator, system integration.
  - 5% authorship.
- Tomas Petricek, Vojtěch Šalanský, Karel Zimmermann, and Tomas Svoboda. *Simultaneous Exploration and Segmentation with Incomplete Data*. Workshop on Action and Anticipation for Visual Learning, ECCV. 2016
  - Computer vision workshop.
  - Personal contribution: design and implementation of the pipeline, design of the neural network.
  - 35% authorship, equal contribution within first two authors.

Chapter 2

Terrain reconstruction learning

The knowledge of the surrounding terrain is an essential clue for autonomous mobile robots. Unfortunately, the real-world conditions make the dense and accurate terrain measurements almost impossible. The laser beams are reflected from some surfaces, which cause holes in the map or are corrupted by dense smoke or dust, creating false objects in the map. Moreover, the information obtained by the lidar measures the surface, which in some cases does not correspond to the rigid terrain that gives us a main clue for robot movement. For example, measuring the tall grass in the outdoor scenario will create a false step in the map that does not correspond to the actual supporting terrain (i.e., the terrain that will provide rigid support for the robot during its traversal).

This part of the thesis focus on terrain shape estimation and representation of the surrounding environment. In Section 2.1 we address the problem of self-supervised learning for predicting the shape of supporting terrain from sparse input measurements. The learning method exploits two types of ground-truth labels: dense 2.5D maps and robot poses, both estimated by a usual SLAM procedure from offline recorded measurements. We show that robot poses are required because straightforward supervised learning from the 3D maps only suffers from: (i) exaggerated height of the supporting terrain caused by terrain flexibility (vegetation, shallow water, snow, or sand) and (ii) missing or noisy measurements caused by high spectral absorbance or non-Lambertian reflectance of the measured surface. We address the learning from robot poses by introducing a novel KKT-loss, which emerges as the distance from necessary Karush-Kuhn-Tucker conditions for constrained local optima of a simplified first-principle model of the robot-terrain interaction. We experimentally verify that the proposed weakly supervised learning from ground-truth robot poses boosts the accuracy of predicted support heightmaps and increases the accuracy of estimated robot poses. All experiments are conducted on a dataset captured by a real platform. Both the dataset and codes which replicate experiments are made publicly available. The corresponding section is based on the publication "Pose Consistency KKT-Loss for Weakly Supervised Learning of Robot-Terrain Interaction Model", [1] which was published in the IEEE Robotics and Automation Letters (RA-L), the Q1 journal with the impact factor 3.74.

The extreme case where all exteroceptive sensors are blinded is discussed in Section 2.2. We propose the approach to estimating the surroundings of the robot using only his pose and tactile sensors. We demonstrate a combined hardware and software solution that enhances the sensor suite and perception capabilities of a mobile robot intended for real search-and-rescue missions. When exploring the unknown environment of a disaster site, a common fail-case is the outage or deterioration of exteroceptive sensory measurements that the robot heavily relies on, especially for localization and navigation purposes. Deprivation of visual and laser modalities caused by dense smoke motivated us to develop a novel solution comprised of force sensor arrays embedded into tracks of our platform. Furthermore, we also exploit a robotic arm for active perception in cases when the prediction based on force sensors is too uncertain. Besides integrating hardware, we also propose a framework exploiting Gaussian processes followed by Gibbs sampling to process raw sensor measurements and provide probabilistic interpretation of the underlying terrain profile. In the final, the profile is perceived by proprioceptive means only and successfully substitutes for the lack of exteroceptive measurements in the close vicinity of the robot, when traversing unknown and unseen obstacles. We evaluated our solution on real-world terrains. That section is based on the work "Touching without vision: terrain perception in sensory deprived environments" [6] published at the international Computer Vision Winter Workshop.

2.1 Self-supervised learning of pose consistent terrain reconstruction

Accurate real-time prediction of robot-terrain interaction from raw sensory measurements is crucial for many mobile robotic tasks ranging from computing traversability/costmap for high-level path planning [9] to state representation for low-level motion control [10]. Even though a usual low-level map such as ICP-aligned lidar scans (optionally discretized to voxelmap or heightmap) is often sparse and assumes terrain to be rigid, it is often used as an input to these tasks [11, 9, 12, 13, 14]. In contrast to others, we propose to predict an intermediate representation – *the shape of supporting terrain*. We show that such architecture outperforms existing state-of-the-art methods in terms of accuracy of predicted supporting terrain and consequently estimated robot poses. Since there is no straightforward way to obtain the ground-truth shape of supporting terrains without manual annotations, we propose to learn it in a self-supervised way from "future" maps and robot poses optimized in simultaneous localization and mapping pipeline (SLAM).

We define the supporting terrain as the layer of terrain, which can provide rigid support for the robot during its traversal. For example, for flexible terrain such as grass, the supporting terrain corresponds to the shape of the lidar-immeasurable rigid layer of the terrain (ground). The shape of supporting terrain is modelled by a dense 2.5D heightmap.

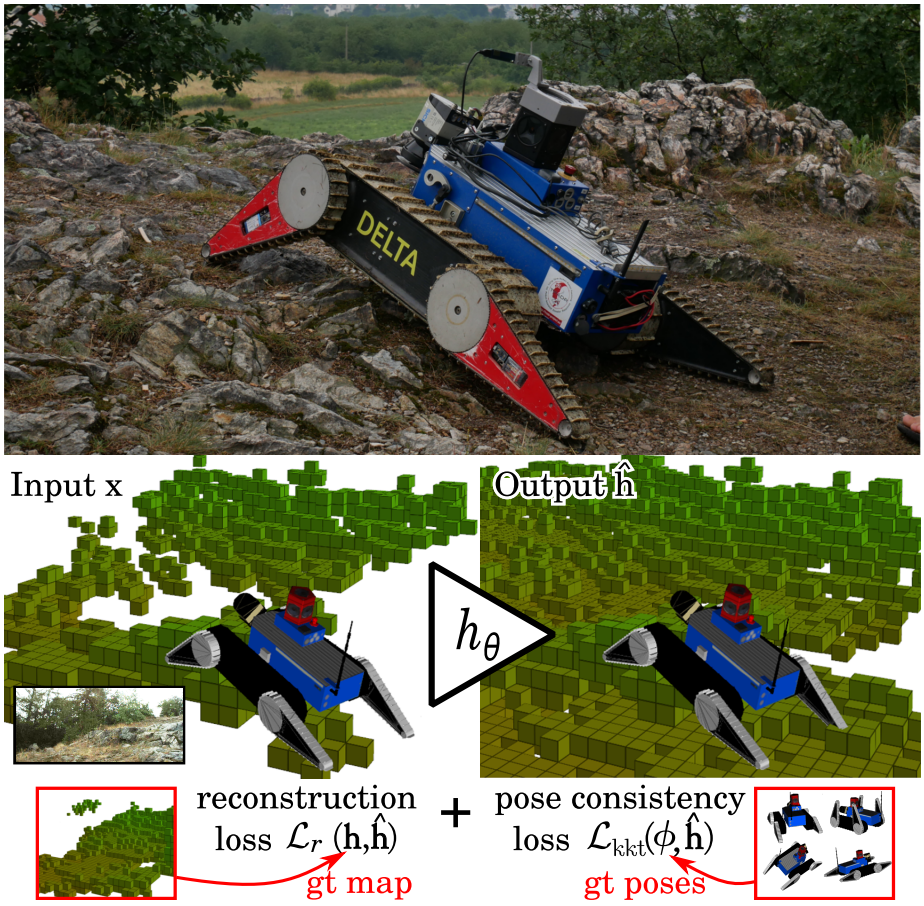


Figure 2.1: Top: The real robot platform is equipped with four independently actuated flippers, which allows traversing complex terrains. **Bottom:** The input $\mathbf{x}$ of the prediction network h_θ is the sparse heightmap obtained by projecting the ICP-aligned lidar scans on the discretized horizontal plane, enriched by visual features. The proposed approach learns to predict the support heightmaps $\hat{\mathbf{h}}$ by enhancing the reconstruction loss by the pose-consistency loss. The proposed pose-consistency loss explicitly models the robot-terrain interaction and enforces the predicted support terrain to be physically consistent with the robot poses. Ground-truth map and ground-truth poses used for learning are obtained from all the measurements along the training trajectories.

The most straightforward way of estimating the supporting heightmap is linear interpolation from ICP-aligned lidar measurement. We show that such an approach typically suffers from (i) missing measurements caused by terrain self-occlusions, (ii) the low spatial resolution of distant terrains, (iii) missing or noisy measurements caused by high spectral absorbance or non-Lambertian reflectance of the measured surface, (iv) exaggerated height of supporting terrain caused by terrain deformations (flexible vegetation, shallow water, snow or sand).

Issues (i) and (ii) could be partially overcome by learning to interpolate the terrain in a self-supervised way, where ground truth is estimated from offline-optimized 3D maps of the environment (as described in Section 3.1

and [4]). This approach partially suppresses (i) the terrain occlusions and (ii) low-spatial resolution by introducing the measurements from additional viewpoints into the learning procedure, however (iii) the lidar unfriendly surfaces and (iv) the terrain flexibility remains unresolved since the ground-truth shapes of the supporting terrain for the fully-supervised learning cannot be easily obtained.

In order to address issues (iii) and (iv), we enhance the fully-supervised reconstruction loss of (Section 3.1 and [4]) by a new pose-consistency loss; see Fig. 2.1 for the outline. The proposed architecture simultaneously optimizes two losses: Reconstruction loss $\mathcal{L}_r(\mathbf{h}, \hat{\mathbf{h}})$, where ground-truth heightmap $\mathbf{h}$ (obtained from offline-optimized maps) enforces the predicted support terrain $\hat{\mathbf{h}}$ to be close to its rigid reconstruction, and pose consistency loss $\mathcal{L}_{\text{kkt}}(\boldsymbol{\phi}, \hat{\mathbf{h}})$, where ground-truth robot pose $\boldsymbol{\phi}$ implicitly enforces the predicted supporting terrain to provide a necessary (yet collision-free) support of the robot. The pose consistency loss provides additional supervision on places, where ground-truth heightmap $\mathbf{h}$ is not available. Fig. 2.2 shows supporting terrain after optimization of the pose consistency loss.

Any direct patching of holes in ground-truth heightmaps by the robot body model would strongly bias the estimation, since it forces the predicted terrain to provide support on all points on the robot body. We avoid this shortcoming by introducing a novel pose-consistency loss. To construct the pose-consistency loss, we exploit a simple yet non-convex first-principle model, which assumes that the robot pose on an uneven terrain corresponds to a minimum of its potential energy with respect to robot-terrain collision constraints. The solution to this problem provides the physically plausible robot pose on a given heightmap. We construct the pose-consistency loss to answer the opposite question: *does the predicted heightmap make the ground-truth pose to be a solution to this problem?* To do so, we simplify the problem and search for heightmaps, for which the ground-truth pose satisfies the necessary Karush-Kuhn-Tucker (KKT) conditions [15] for the constrained local optima of this first-principle model. Obviously, the number of heightmaps consistent with a single pose is immense, since it grows exponentially with the size of the predicted heightmap. The KKT-loss avoids the explicit generation of all pose-consistent heightmaps. The proposed KKT-loss is constructed to measure the distance from these KKT conditions. Consequently, its optimization directly leads towards physically consistent heightmaps in the sense that any heightmap, which zeros the KKT-loss, is considered to be physically consistent with the ground-truth pose.

Main contribution lies in:

- introducing a novel KKT-loss, which allows for weakly-supervised learning of the supporting terrain from ground-truth poses and suggesting an algorithm for its efficient optimization
- evaluating the proposed method on a real dataset and publishing the codes and the dataset, which replicates the results reported in the experiments.[1]

[1] `github.com/ctu-vras/pose-consistency-kkt-loss`

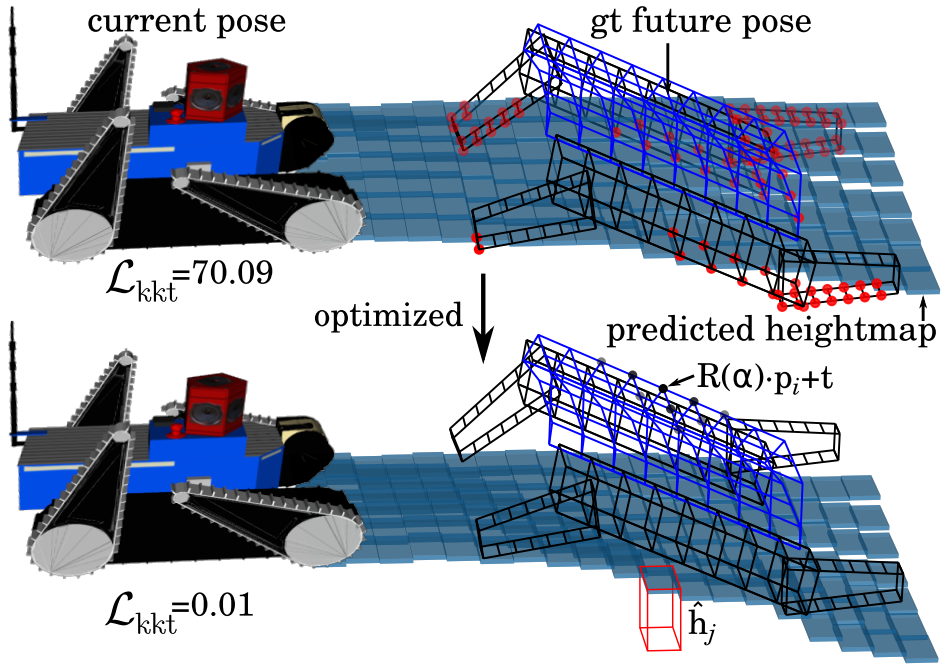


Figure 2.2: Robot is represented as a set of mass points $\mathbf{p}_i, m_i$ Terrain is represented as heightmap $\hat{h}_j$. **Top:** Network predicts the shape of the supporting terrain (blue heightmap) from measurements available in the current pose. The ground-truth pose is inconsistent with the terrain since the robot model is in collision (red points). **Bottom:** Optimization of KKT-loss for a given ground-truth pose yields physically plausible reconstruction, which does not force the terrain to copy the shape of the robot.

2.1.1 Related work

This section consists of three parts. The first part discusses approaches to the object reconstruction task. The second part summarizes methods that predict the robot-terrain interaction and discuss their relation to the proposed supporting terrain. The third part discusses learning approaches that resemble proposed KKT-loss in the sense that they learn to predict values connected with ground-truth labels through an optimization problem.

Object reconstruction

High performance of image-based models is demonstrated in [16], where a CNN pooling results from multiple rendered views outperforms commonly used 3D shape descriptors in object recognition task. Several volumetric and multi-view network architectures for object classification are compared by Qi *et al.* [17]. The authors focus on closing a performance gap between these two approaches and investigate several techniques towards this goal, such as data augmentation, or using 2D convolution with elongated kernels for projecting volumetric representation into a 2D image. We choose a similar approach in designing the mapping network. Choy et al. [18] proposed a unified approach

for single and multi-view 3D object reconstruction which employs a recurrent neural architecture. Their recurrent neural network architecture learns a map from sequences of images to object shapes in terms of 3D occupancy grid ($32 \times 32 \times 32$ voxels). The architecture they use is not suitable for dealing with high-dimensional outputs due to its high memory requirements.Model-fitting methods such as [19, 20, 21] rely on a manually-annotated dataset of models and assume that objects can be decomposed into a predefined set of parts. Besides that these methods are suited mostly for man-made objects of rigid structure, fitting of the models and their parts to the input points is computationally very expensive; e.g., minutes per input for [19, 20]. Decomposition of the scene into plane primitives as in [22] does not scale well with scene size (quadratically due to candidate pairs) and could not most likely deal with the level of sparsity we encounter. Geometrical and physical reasoning comprising stability of objects in the scene is used by [23] to improve object segmentation and 3D volumetric recovery. First, solid 3D primitives are recovered from point cloud, and then the unstable objects are grouped with the physically stable ones to minimize an energy function which includes a penalty for object (in)stability, size, geometric complexity etc. The proposed volumetric recovery is based on implicit algebraic models and the assumption of objects being aligned with coordinate axes which seems unrealistic in practice. Moreover, it is not clear how to incorporate learned shape priors for complex real-world objects which were shown to be beneficial for many tasks (e.g., in [24]). Firman et al. [25] use a structured-output regression forest to complete unobserved geometry of tabletop-sized objects.

Robot-terrain interaction models

Typical quantities, which are predicted in order to model the robot-terrain interaction, are arbitrary values that represent the expected behaviour of the robot on uneven terrain. We briefly discuss several different quantities such as the expected pose of the robot on the terrain [26], robot-terrain reaction score [14], friction/slippage coefficient [27]. Most of these methods is not a direct competitor; however, their accuracy heavily depends on the accuracy of reconstructed terrain, which is provided as an input.

Geometrical analysis: The most straightforward way of predicting the robot-terrain interaction is the direct geometrical analysis of the terrain shape, such as point cloud or heightmap. Geometrical analysis typically exploits heuristics based on manually chosen features, such as terrain normals, height difference, slope, roughness, and robot shape [28, 29]. Some approaches [30, 26] iteratively optimizes the robot-terrain transformation to obtain the contact points and static robot pose.

Self-supervised learning: Other methods learn to predict the robot-terrain interaction directly. Methods such as Suryamurthy et al. [13] learn to estimate terrain roughness from RGB images for wheeled Centauro robot, where terrain roughness labels are automatically computed from SfM optimized heightmaps. In contrast to us, such an approach inherently suffers from

the inability to assess the terrain's flexibility and the inability to reconstruct visually homogeneous terrain. Many others directly learn from the recorded behaviour of robot on the terrain. For example, Wellhausen et al. [14] estimate the ground reaction score for the legged ANYmal robot, where force-torque sensors automatically estimate labels. Similarly, Angelova et al. [27] estimate the slippage coefficient for a planetary rover. The approach proposed by Chavez et al. [31] learn to predict the terrain traversability estimated from the simulator and Nouza et al. [32] predicts robot poses. Since it is impossible to obtain real ground truth without exhaustive manual annotations, they suggest training it from the simulation. All of these methods expect to have complete heightmaps, which are typically not guaranteed in real robotic scenario due to occlusions or reflectance of the surface.

Most of the previously discussed methods heavily depend on the accuracy of provided terrain shape. Since ICP-aligned point clouds [33] or heightmaps are inaccurate, it is possible to improve their accuracy by a refinement step, which provides a more accurate estimate of a terrain shape. Methods such as [4, 34, 35] complete the sparse laser measurements using the neural networks. In contrast to our approach discussed in this chapter, these methods do not take the robot-terrain interaction (such as robot pose) into account, and therefore they can only reconstruct the rigid terrain with lidar-friendly surfaces.

In contrast to the straightforward prediction of previously mentioned quantities (terrain roughness score, slippage, ground reaction score, or the traversability), we suggest predicting the supporting terrain as a preliminary representation, which is more suitable input to these methods than a noisy 3D point clouds. This claim is experimentally verified on the pose prediction and heightmap reconstruction problem.

Learning with implicit optimization

The problem of learning the shape of support terrain from ground-truth poses belongs to the class of learning approaches, where it is not easy to obtain the ground truth for fully-supervised learning. However, it is possible to use predicted values as an input to an optimization problem, the solution of which can be compared to an alternative ground truth that is easy to obtain. It has been recently shown that a convex optimization problem could be used as a differentiable layer in neural network [36]. While various applications ranging from optimal control to signal denoising has been shown, it cannot be directly used since we need to differentiate through the non-convex first principle model of the robot-terrain interaction.

Inverse reinforcement learning (IRL) is a class of methods, which use an optimization layer to relate predicted values (costmap) to ground truth (expert trajectories). Such approaches exploit an expert driver to collect trajectories, which serves as easy-to-obtain ground truth for learning the costmap model, for which the straightforward ground truth is typically not available. The IRL assumes that the expert driver has a latent costmap, for which the executed trajectory is optimal (i.e. has the lowest sum of costs).

Given this assumption, it is possible to learn to predict this costmap by searching the optimal trajectories on predicted costmaps and then comparing them with those executed by the driver. The learning requires backpropagating through the layer that performs the search of optimal trajectories for a given costmap. Authors typically either train differentiable policies that imitate drivers' behaviour or use Dijkstra in each iteration and than backpropagate gradients along the fixed optimal trajectory only. IRL has been used on several mobile platforms. For example, Silver et al. [9] apply it on the six-wheeled Crusher military platform, while Wulfmeier et al. [12] or Zeng et al. [37] learned spatial traversability for driving an autonomous car in complex urban environments. In contrast to previous self-supervised approaches, the IRL allows to learn also the non-traversable terrain directly on the real platform from the expert behaviour. However, the connection between the expert trajectories and the traversability is often weak; therefore, the problem is typically ill-posed [38]. In addition to that, expert trajectories are often sub-optimal, which significantly harm the resulting costmap. Since IRL methods use different ground truth (the expert trajectories) and the optimization layer (search for cost-minimizing path), it cannot be easily applied for learning from ground-truth poses.

Multiple Instance Learning (MIL): One way to avoid direct backpropagation through an optimization layer is to explicitly generate all possible predictions, which make a given ground truth to be the solution of the underlying optimization problem, and then train on these predictions via MIL. For example, the proposed KKT-loss enforces the predicted support heightmap to provide a necessary (yet collision-free) support of the robot. One could achieve a similar effect by employing the MIL [39]. Such an approach would generate huge positive bags consisting of all supporting terrains, which are physically consistent with a given ground-truth pose, and then minimize the reconstruction loss from the closest sample in each positive bag. However, the number of heightmaps consistent with a single pose is immense since it grows exponentially with the predicted heightmap size. In contrast to the multiple instance learning, the proposed KKT-loss does not require to generate all pose-consistent heightmaps explicitly; it just measures the distance from KKT conditions.

Multi-task learning: Another alternative, which allows to easily backpropagate through the optimization problem, is to train a network, which directly estimates solutions of the optimization layer and use this network as a simply differentiable replacement for the optimization layer. A similar idea appears either in multitask learning [40] or weakly-supervised learning [41]. The main disadvantage is that the replacement network strongly biases the learning process by its own inaccuracy.

2.1.2 Theory

We train a convolutional network $h_\theta : \mathbb{X} \to \mathbb{H}$, which maps *sparse* heightmaps $\mathbf{x} \in \mathbb{X}$, estimated by projecting the ICP-aligned [33] lidar scans on the

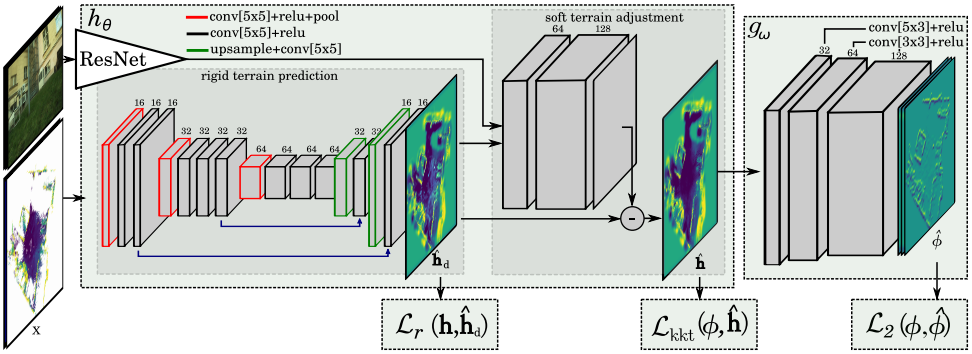


Figure 2.3: Visualization of network architectures. Network h_θ predicts the dense supporting heightmap from sparse input. Training minimizes reconstruction loss and pose consistency loss. We propose two types of pose consistency losses: (i) KKT-loss and (ii) pose predicting loss (concatenation of pretrained pose regression network g_ω predicting roll, pitch, z on the dense supporting heightmap with L2 loss on ground-truth poses).

discretized horizontal plane (optionally enriched by visual features), on *dense support* heightmaps $\hat{\mathbf{h}} \in \mathbb{H}$, where θ is the vector of network parameters, see Fig. 2.3 for an overview. In our experiments, the input $\mathbf{x}$ is either a two-channel 2D array with heights in the first channel and NaNs binary encoded in the second channel or a multichannel 2D array with heights, NaNs, and visual features. We exploit two types of self-supervised labels (i) ground-truth heightmaps $\mathbf{h} \in \mathbb{H}$, (ii) ground-truth robot poses $\boldsymbol{\phi} \in \Phi$, both estimated offline from recorded measurements by a SLAM pipeline [33]. Therefore, the ground truth for a prediction in a certain time-stamp also includes future measurements. Both types of labels are imperfect. Heightmaps are noisy, incomplete, and overestimate support heights on flexible terrains. Robot poses constrain the shape of the underlying terrain only by its physical consistency with the predicted terrain. Learning is defined as the minimization of composite loss on both types of labels.

Learning from ground-truth heightmaps

Given a ground-truth heightmap $\mathbf{h}$ and predicted supporting heightmap $\hat{\mathbf{h}}$, we use (optionally asymmetric) L2-loss $\mathcal{L}_r(\mathbf{h},\hat{\mathbf{h}}) = \|\max\{a(\mathbf{h}-\hat{\mathbf{h}}), \hat{\mathbf{h}}-\mathbf{h}\}\|_2^2$, which optionally provides decreased loss for underestimated heights. Especially, if $a = 0$ the reconstruction loss is quadratic upper bound, if $a = 1$ the reconstruction loss becomes a usual L2 loss.

Learning from ground-truth poses

We introduce an additional penalty for predicting heightmaps, which are physically inconsistent with the ground-truth robot pose, such as terrains that either does not provide sufficient stability of all degrees of freedom of the robot or colliding with the robot body. To enforce the penalty, we propose two different pose-consistency losses: (i) the KKT-loss $\mathcal{L}_{kkt} : \Phi \times \mathbb{H} \to \mathbb{R}$,

which is purely based on the first-principle model (Section 2.1.2.2) and (ii) pose-predicting loss $\mathcal{L}_p : \Phi \times \mathbb{H} \to \mathbb{R}$ (Section 2.1.2.3), which contains pose-predicting regressor g_ω followed by L2 loss on predicted $\hat{\boldsymbol{\phi}}$ and ground-truth pose $\boldsymbol{\phi}$: $\mathcal{L}_p(\boldsymbol{\phi}, \hat{\mathbf{h}}) = \mathcal{L}_2(\boldsymbol{\phi}, g_\omega(\hat{\mathbf{h}}))$. The pose-predicting regressor g_ω has to be trained in advance.

2.1.2.1 Architecture

We study two different cases: (i) strictly rigid terrain, on which the supporting terrain is equivalent to its dense reconstruction, and (ii) partially flexible terrain, on which the predicted support terrain is lower or equal to its dense reconstruction.

Strictly rigid terrain. In this case, the input $\mathbf{x}$ is only the sparse heightmap (2D array with NaNs). The learning minimizes both losses simultaneously $\mathcal{L}_r(\mathbf{h}, \hat{\mathbf{h}}) + \mathcal{L}_{\text{kkt}}(\boldsymbol{\phi}, \hat{\mathbf{h}})$. For the reconstruction loss, we use $a = 1$, since the output should directly correspond to ground truth $\mathbf{h}$.

Partially flexible terrain. In this case, the input $\mathbf{x}$ is the sparse heightmap enriched by visual features obtained by projecting outputs of RGB network on the heightmap. We observed that the simultaneous minimization of both losses on a flexible terrain suffers from undesirable interference. The reconstruction loss pulls predicted heights towards lidar-measured heights on flexible terrain, while the pose consistency loss enforces lower heights that do not collide with the robot body. To avoid such undesirable behaviour, we propose to divide h_θ into two sub-networks. The first sub-network (denoted as "rigid terrain prediction" in Fig. 2.3) is responsible for reconstructing the terrain $\hat{\mathbf{h}}_d$ as it has been rigid, the second sub-network (denoted as "soft terrain adjustment" in Fig. 2.3) predicts the terrain flexibility. The reconstruction loss is connected between these sub-networks, while the pose consistency loss is connected in the end. The learning minimizes compound loss: $\mathcal{L}_r(\mathbf{h}, \hat{\mathbf{h}}_d) + \mathcal{L}_{\text{kkt}}(\boldsymbol{\phi}, \hat{\mathbf{h}}) + \mathcal{L}_2(\hat{\mathbf{h}}_d, \hat{\mathbf{h}})$, where the last term slightly encourages the predicted supporting terrain to be similar to the rigid reconstruction $\hat{\mathbf{h}}_d$, see Fig. 2.3. The reconstruction loss naturally enforces that the rigid reconstruction $\hat{\mathbf{h}}_d$ is a necessary part of the predicting process without influencing the second sub-network. In this scenario, we experimented with asymmetric reconstruction loss (i.e. with $a < 1$). Eventually, it turned out that the best is to enforce the upper bound directly into the h_θ architecture by making the "soft terrain adjustment" sub-network only able to decrease $\hat{\mathbf{h}}_d$ by subtracting the non-negative outputs from $\hat{\mathbf{h}}_d$.

2.1.2.2 KKT loss

In this section we firstly introduce a simple first-principle model, allows to estimate robot pose $\boldsymbol{\phi}$, given a predicted heightmap $h_\theta(\mathbf{x})$. This model is employed in the next part to construct the KKT loss. Then we describe

efficient estimation of KKT loss by the inner-loop minimization of its KKT multipliers.

First-principle model. Let us assume that a set of mass points represents the robot $(\mathbf{p}_i, m_i)\ i = 1 \dots N$, where $\mathbf{p}_i \in \mathbb{R}^3$ are 3D coordinates and $m_i \in \mathbb{R}$ are weights, see Fig. 2.2. We omit modeling the passive compliance[2] of flipper-motors and assume, that the positions of four independently articulated flippers in a given time are fixed in a measured position, therefore flipper points become a rigid part of the model. Robot pose $\boldsymbol{\phi} = [\boldsymbol{\alpha}, \mathbf{t}]$ is uniquely determined by its roll, pitch, yaw angles denoted by $\boldsymbol{\alpha}$ and translation vector $\mathbf{t} \in \mathbb{R}^3$. Matrix $\mathbf{R}(\boldsymbol{\alpha})$ denotes rotation matrix corresponding to rotation angles $\boldsymbol{\alpha}$. Given a predicted heightmap $\hat{\mathbf{h}} = h_\theta(\mathbf{x}) \in \mathbb{H}$, we define robot pose estimation problem as the minimization of its potential energy

$$\sum_i m_i \cdot g \cdot [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z \tag{2.1}$$

with respect to collision constraints

$$\hat{h}_i - [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z \leq 0 \quad \forall_i,$$

where $[.]_z$ denotes the z-coordinate of the vector inside the brackets and $\hat{h}_i$ is the height corresponding to point $\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}$. Note, that if the pose needs to optimized (which is not the case of this section) then the point-height correspondences have to be re-established in each iteration.

Consistency of predicted heightmap with ground-truth pose. We express the consistency of the predicted heightmap $\hat{\mathbf{h}}$ with a ground-truth pose $\boldsymbol{\phi}$ as the L2-distance from the necessary conditions for the constrained local optimality of $\boldsymbol{\phi}$ in this model. In particular, given the Lagrangian of the pose estimation model

$$\begin{aligned} L(\boldsymbol{\alpha}, \mathbf{t}, \mathbf{h}, \boldsymbol{\lambda}) = &\sum_i m_i \cdot g \cdot [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z + \\ &+ \sum_i \lambda_i (\hat{h}_i - [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z), \end{aligned} \tag{2.2}$$

where $\boldsymbol{\lambda}$ denotes the KKT multipliers, we express necessary optimality conditions:
Stationarity conditions

$$\frac{\partial L(\boldsymbol{\alpha}, \mathbf{t}, \mathbf{h}, \boldsymbol{\lambda})}{\partial \boldsymbol{\alpha}, \mathbf{t}} = \sum_i (m_i g - \lambda_i) \frac{\partial [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z}{\partial \boldsymbol{\alpha}, \mathbf{t}} = 0\,,$$

complementary slackness conditions

$$\lambda_i \cdot (\hat{h}_i - [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z) = 0\,,$$

[2] By passive compliance, we refer to flipper motors inability to lift the body without any additional support of the main tracks, which results in a passively smooth motion over the terrain.

primal feasibility conditions

$$\hat{h}_i - [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z \leq 0 \quad \forall_i \,,$$

dual feasibility conditions

$$\boldsymbol{\lambda} \geq \mathbf{0} \,.$$

While stationarity together with complementary slackness assure physical stability of the robot on the predicted heightmap $\hat{\mathbf{h}}$ in all considered degrees of freedom $\boldsymbol{\phi}$, the primal feasibility assures that the predicted heightmap does not collide with the robot model.

The KKT-loss of the predicted heightmap $\hat{\mathbf{h}}$ for a ground-truth pose $\boldsymbol{\phi}$ is then defined as follows

$$\begin{aligned} \mathcal{L}_{\text{kkt}}(\boldsymbol{\phi}, \hat{\mathbf{h}}) = \min_{\boldsymbol{\lambda}} \Bigg\{ & \left\| \sum_i (m_i g - \lambda_i) \frac{\partial [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z}{\partial \boldsymbol{\alpha}, \mathbf{t}} \right\|_2^2 + \\ & + \sum_i \Big(\lambda_i \cdot (\hat{h}_i - [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z) \Big)^2 + \\ & + C \cdot \Big(\max\{0, \hat{h}_i - [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z\} \Big)^2 \;\Big|\; \boldsymbol{\lambda} \geq 0 \Bigg\}, \end{aligned} \tag{2.3}$$

where $C > 0$ is a learning hyper-parameter. In our experiments, we use $C = 100$, since it was a reasonable compromise between resulting primal feasibility and obtaining an optimization-friendly landscape of the KKT-loss. However we have not noticed any significant impact on the results for $C = 10$ or $C = 1000$ (if the learning rate was correspondingly adjusted).

Estimating the KKT-loss by efficient inner-loop optimization. Since the Lagrangian $L(\boldsymbol{\alpha}, \mathbf{t}, \mathbf{h}, \boldsymbol{\lambda})$ is linear in KKT multipliers, the minimization over $\boldsymbol{\lambda} > 0$ reduces to the following non-negative least squares problem:

$$\begin{aligned} \boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}} \Bigg\{ & \left\| \sum_i (m_i g - \lambda_i) \frac{\partial [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z}{\partial \boldsymbol{\alpha}, \mathbf{t}} \right\|_2^2 + \\ & + \sum_i \Big(\lambda_i \cdot (\hat{h}_i - [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z) \Big)^2 \;\Big|\; \boldsymbol{\lambda} \geq 0 \Bigg\}, \end{aligned} \tag{2.4}$$

which is known to have an efficient solution [42]. For the sake of simplicity we skip the terms which does not depend on λ_i. Learning of parameters $\boldsymbol{\theta}$ with the KKT-loss is summarized in the following algorithm:

We provide fully differentiable implementation of the KKT-loss, which allows for learning through standard PyTorch interfaces such as `kkt_loss(net).backward()`.

2.1.2.3 Pose-predicting loss

We suggest to follow the idea of multi-task learning for semantic segmentation, where a classifier of another task is trained in advance and then used to

Algorithm 1: Estimating the gradient of KKT-loss

Input: Training batch $(\mathbf{x}_j, \boldsymbol{\phi}_j),\ j = 1 \dots M$
Robot model $p_i, m_i,\ i = 1 \dots N$
for $j = 1 \dots M$ **do**
  Predict heights (feedforward pass): $\hat{\mathbf{h}} = h_\theta(\mathbf{x}_j)$
  Estimate $\boldsymbol{\lambda}^*$ by solving the non-negative least squares Problem (2.4).
  Construct KKT-loss $\mathcal{L}_{\text{kkt}}(\boldsymbol{\phi}_j, h_{\boldsymbol{\theta}}(\mathbf{x}_j))$ by substituting inner loop minimizer $\boldsymbol{\lambda}^*$ into eq. (2.3).
  Cumulate gradient $\frac{\partial \mathcal{L}_{\text{kkt}}(\boldsymbol{\phi}_j, h_{\boldsymbol{\theta}}(\mathbf{x}_j))}{\partial \boldsymbol{\theta}}$ with respect to parameters $\boldsymbol{\theta}$

estimate labels for the original task. Similarly, we start by learning the pose regressor $g_{\boldsymbol{\omega}} : \mathbb{H} \to \Phi$, which predicts the robot pose $\boldsymbol{\phi}$ given the heightmap $\mathbf{h}$; see Fig. 2.3 for network architecture details.

Since the robot observes a significantly larger area than it physically visits, there are many heightmap patches for which the pose is unknown. We denote J to be the set of indices of heightmap patches for which the real ground-truth pose is known and K to be the set of all indices. Since J is relatively small for training a reliable pose regressor, we decided to exploit also the real heightmap patches $K \setminus J$ for which the real pose is unknown by estimating synthetic robot poses on them.

In particular, given a pretrained reconstruction network $h_{\boldsymbol{\theta}}(\mathbf{x})$, we first reconstruct each dense heightmap $\hat{\mathbf{h}} = h_{\boldsymbol{\theta}}(\mathbf{x})$ and then find corresponding ground-truth pose as the solution of the first principle model:

$$\begin{aligned} \boldsymbol{\phi} = &\arg\min_{\boldsymbol{\alpha},\mathbf{t}} \sum_i m_i \cdot g \cdot [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z \\ &\text{subject to: } \hat{\mathbf{h}}_i - [\mathbf{R}(\boldsymbol{\alpha}) \cdot \mathbf{p}_i + \mathbf{t}]_z \leq 0 \quad \forall_i, \end{aligned} \tag{2.5}$$

The solution is searched by the steepest gradient method with a quadratic exterior penalty function.

Given these synthetically estimated poses we learn the pose predicting regressor $g_{\boldsymbol{\omega}}$ on mixed training data K, consisting of both real and synthetically estimated poses

$$\arg\min_{\omega} \sum_{k \in K} \mathcal{L}_2(\boldsymbol{\phi}_k, g_{\boldsymbol{\omega}}(\mathbf{h}_k)). \tag{2.6}$$

Finally, we construct the pose predicting loss $\mathcal{L}_p$ as the L2 error between ground-truth pose $\boldsymbol{\phi}$ and predicted pose $\hat{\boldsymbol{\phi}} = g_{\boldsymbol{\omega}}(\hat{\mathbf{h}})$

$$\mathcal{L}_p(\boldsymbol{\phi}, \hat{\mathbf{h}}) = \mathcal{L}_2(\boldsymbol{\phi}, g_{\boldsymbol{\omega}}(\hat{\mathbf{h}})) = \mathcal{L}_2(\boldsymbol{\phi}, \hat{\boldsymbol{\phi}}) \tag{2.7}$$

and train the heightmap predicting network h_θ by minimizing pose predicting loss on real ground-truth poses $\boldsymbol{\phi}_j, j \in J$.

$$\sum_{j \in J} \mathcal{L}_p(\boldsymbol{\phi}_j, h_{\boldsymbol{\theta}}(\mathbf{x}_j)) \tag{2.8}$$

Since the change in parameters θ changes the distribution of reconstructed heightmaps $\hat{\mathbf{h}}_k$, the pose regressor should be retrained. Theoretically, we should iterate this process until a fixed point is reached; however, we observed in the practice that a good initialization of h_θ is sufficient.

The pose predicting loss can be understood as a counterpart of the proposed KKT-loss. In contrast to KKT-loss, the pose predicting loss suffers from the regressor bias. The bias is strong since flexible terrains and lidar-unfriendly surfaces cannot be used for the regressor training. In the other hand, the pose-predicting loss does not require any inner-loop optimization, therefore its backpropagation is faster[3].

2.1.3 Experiments

We evaluate the performance of the proposed methods on a real-world dataset. The achieved results are compared with two state-of-the-art methods [32, 4]. For the sake of a fair comparison, all methods are implemented and trained from scratch on the same network architectures, training/testing subsets and hyper-parameters[4].

Dataset

Dataset consists of trajectories collected by the tracked robot during the traversal of various rigid and flexible obstacles. The robot was equipped with a spherical camera PointGrey Ladybug3, LiDAR SICK LMS-151, and inertial measurement unit (IMU) Xsens MTi-G. Internal camera parameters were factory calibrated, camera-lidar calibration was based on [43]. The data consist of the sparse input heightmaps $\mathbf{x}$ enriched by projected RGB-ResNet [44] features, ground-truth heightmaps $\mathbf{h}$ and ground-truth robot poses $\boldsymbol{\phi}$. All heightmaps are obtained as follows: Given a set of lidar scans, odometry measurements, and IMU data, we estimate a 3D point cloud map by a SLAM approach [33]. We filter out the ceiling automatically and discretize the resulting 3D point cloud into a heightmap with 10-cm bins. This heightmap is transformed into a robot-centric frame with gravity-aligned z-axis (i.e., the frame with roll and pitch equal to zero). The whole dataset consists of 871 training, 335 validation, and 745 testing heightmaps each of size $25.6\,\mathrm{m}\times25.6\,\mathrm{m}$. It contains over $4\cdot10^6$ predictable heights (usually only a part of the heightmap is known) and more than $2\cdot10^4$ robot poses (there are multiple robot poses for each heightmap). The 2/3 of the dataset (used for the "Strictly rigid terrains" experiment) consist only of rigid terrain, collected indoor with robot traversing over various obstacles and driving on uneven terrain in experimental mines (see Fig. 2.4). The remaining 1/3 of the dataset (used in the "Partially flexible terrains" experiment) contains both rigid and flexible terrains. This part of the dataset was collected outdoor on various

[3]The backpropagation over the KKT-loss for a single robot pose takes about 35 ms on a quad-core 2.3 GHz Intel i7 processor with 16 GB RAM.

[4]All the approaches were trained using Adam optimizer with learning rate 10^{-4}, and the final weights were picked using the validation set.

terrains such as cobblestones, rocks, paths, grass, or clay (see Fig. 2.5). While the proposed methods could be adapted to different robot models, the dataset we collected is specific to the robot and sensor model.

Input x: The sparse input heightmap at time t is the heightmap estimated from lidar, IMU, and odometry measurements captured until this time by the procedure described above. We extend the heightmap by RGB features obtained from the camera image captured in time t. The RGB features are estimated from the camera image by a pre-trained ResNet network. The resulting 150 features are upsampled to original image resolution and projected onto the heightmap. The input for terrain reconstruction $\mathbf{x}$ has dimension $256 \times 256 \times 152$. The sparse array with terrain heights is in the first channel (see Fig. 2.4a), the binary mask of unknown measurements is in the second channel and the camera features in the subsequent 150 channels. In the experiment showing rigid terrain reconstruction, we use only the first two channels consisting of a sparse measurement and a corresponding mask.

Ground truth: The ground-truth label $\mathbf{h}$ at time t is the heightmap estimated from lidar, IMU, and odometry measurements captured along the whole recorded trajectory by the procedure described above. In contrast to sparse inputs, the ground truth also contains future measurements, therefore it is significantly denser and more accurate (compare Fig. 2.4a and Fig. 2.4b). Nevertheless, the ground-truth heightmaps are still often inaccurate and incomplete, and they suffer from discretization along all axes (see Fig. 2.4b). Each ground-truth pose $\boldsymbol{\phi}$ contains roll and pitch angles of the robot and z coordinate of the position, resulting from the same SLAM procedure.

Methods

We evaluate the performance of proposed methods (r+kkt and r+p), which minimize compound loss described in Section 2.1.2.1, with two state-of-the-art methods (li [32] and r Section 3.1[4]) in terms of the supporting terrain reconstruction accuracy and pose predicting accuracy. All compared methods use the same SLAM procedure [33] to align the input measurements and the pose is always predicted from estimated supporting terrain $\hat{\mathbf{h}}$ by pose predicting network g_ω. g_ω is trained[5] to minimize $\mathcal{L}_2(\boldsymbol{\phi}, g_\omega(\hat{\mathbf{h}}))$. A detailed description of implemented methods is in the following paragraphs.

li: This method is based on work [32], which predicts the robot poses from heightmaps by linear and Gaussian process regression on linearly interpolated heightmaps. We replicate this work, by filling the missing measurement by **l**inear **i**nterpolation and then predicting the pose by g_ω.

r: This method replaces the linear interpolation by deep convolution network h_θ trained by minimizing **r**econstruction loss $\mathcal{L}_r(\mathbf{h}, h_\theta(\mathbf{x}))$ similarly to [4]. The pose estimation remains the same as in work [32].

r+p: Method corresponds to learning with pose-predicting loss (Section 2.1.2.3). This method first pretrains the h_θ and g_ω network in the same

[5]To train the g_ω network we used $5 \cdot 10^5$ terrain-poses, and the testing error evaluated on ground-truth heightmaps was 0.070.

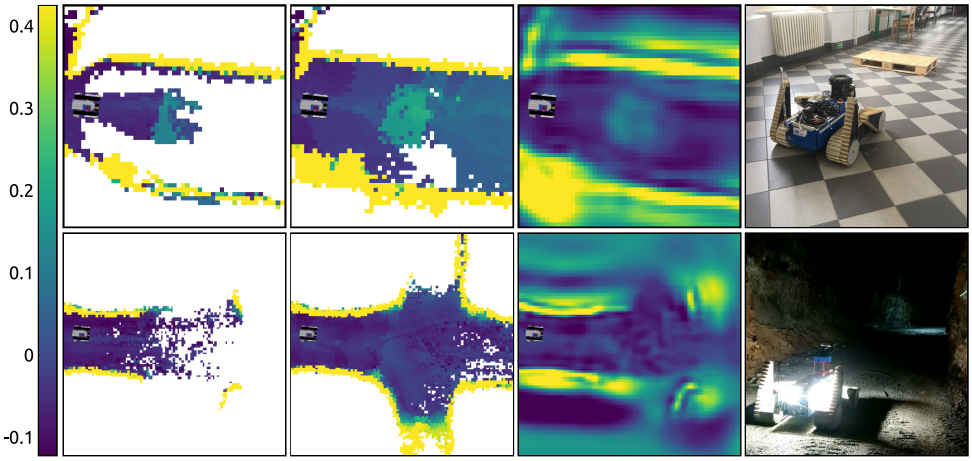


Figure 2.4: Example crop from testing data. **First row:** Robot approaching the obstacle in the hallway. The scale is in meters. Yellow pixels are walls, dark blue denotes a floor an light blue blob in the middle is the obstacle. **Second row:** Robot in the experimental mine has noisy and missing data on the floor, our network fill the gaps.

way as in previous methods. Then we connect the computational graphs of both networks h_θ and g_ω and train only θ parameters by backpropagating the **p**ose-predicting errors of g_ω and **r**econstruction errors of h_θ. In particular we minimize the weighted sum $\mathcal{L}_r(\mathbf{h}, \hat{\mathbf{h}}_d) + \mathcal{L}_p(\boldsymbol{\phi}, \hat{\mathbf{h}}) + \mathcal{L}_2(\hat{\mathbf{h}}, \hat{\mathbf{h}}_d)$.

r+kkt: Method corresponds to learning with KKT-loss (Section 2.1.2.2). The pre-training procedure is the same, the final training minimizes weighted sum: $\mathcal{L}_r(\mathbf{h}, \hat{\mathbf{h}}_d) + \mathcal{L}_{\text{kkt}}(\boldsymbol{\phi}, \hat{\mathbf{h}}) + \mathcal{L}_2(\hat{\mathbf{h}}, \hat{\mathbf{h}}_d)$.

	errors	li [33][32]	r [33][4][32]	r+p (Sec.2.1.2.3)	r+kkt (Sec.2.1.2.2)
measured	roll [rad]	0.113	0.077	0.075	**0.073**
	pitch [rad]	0.075	0.069	0.065	**0.064**
	z [m]	0.067	**0.049**	0.072	0.057
all	roll [rad]	0.140	0.082	0.078	**0.075**
	pitch [rad]	0.191	0.089	0.072	**0.071**
	z [m]	0.140	0.080	0.082	**0.068**
	heightmap [m]	0.202	0.118	0.096	**0.089**

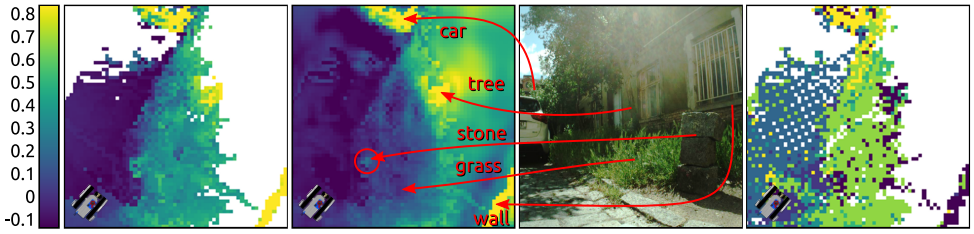
Table 2.1: Mean average terrain reconstruction errors and pose estimation errors for rigid terrain reconstruction.

Evaluation and discussion

Strictly rigid terrains. This experiment demonstrates the ability of proposed methods to reconstruct the heightmaps and predict poses from sparse inputs

	errors	li [33][32]	r [33][4][32]	r+p (Sec.2.1.2.3)	r+kkt (Sec.2.1.2.2)
measured	roll [rad]	0.089	0.084	0.056	**0.048**
	pitch [rad]	0.075	0.064	0.048	**0.042**
	z [m]	0.145	0.142	0.056	**0.045**
	heightmap*[m]	0.124	0.131	**0.057**	0.059
all	roll [rad]	0.093	0.098	0.063	**0.057**
	pitch [rad]	0.071	0.084	0.051	**0.044**
	z [m]	0.131	0.134	0.067	**0.055**
	heightmap*[m]	0.114	0.129	**0.075**	**0.075**

Table 2.2: Mean average terrain reconstruction errors and pose estimation errors on partially flexible terrain



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

Figure 2.5: Example crop from testing data. Robot measured an input heightmap (a) and predicts the supporting terrain (b) using the ResNet features (d). The scene is shown on image (c). Only maximum channel of ResNet features is shown in the image (d). The heightmaps (a) and (b) are in meters.

on rigid terrains; see Fig. 2.4 for testing examples and for the qualitative results. Since terrains are assumed to be rigid, we have omitted the "soft terrain adjustment" part of h_θ for this experiment (see Fig. 2.3). The quantitative results on the testing data are depicted in Table 2.1. The first three rows show Root-Mean-Square error (RMS) on *fully-measured* heightmaps (i.e. those, where input heightmap $\mathbf{x}$ does not contain any NaNs). The second part of the table shows how methods perform on *all* heightmaps, including also incomplete input heightmaps $\mathbf{x}$.

The results demonstrate that the supporting terrain reconstruction is an important intermediate step for pose prediction since the linear interpolation (li) suffers from sparsity of input heightmaps. We can also see that the backpropagation from ground-truth poses via proposed pose consistency losses ($\mathcal{L}_{kkt}$ and $\mathcal{L}_p$), improves the accuracy of the predicted terrains and poses. The results show that the proposed methods decrease the reconstruction error and pose predicting error by approximately 20%. Since the ground-truth height of extremely high and obviously untraversable obstacles such as walls, trees, or buildings is not well defined (e.g. due to unreliability of ceiling removal procedure), the heightmap reconstruction error is evaluated only on bins, where ground-truth height is lower than 0.3m.

Partially flexible terrains. This experiment evaluates the ability to predict the shape of supporting terrain on both types of terrain (flexible and rigid); see Fig. 2.5 for an example of the testing data. In this particular example, the sparse input heightmap contains the heights measured by lidar on the high grass. Since these heights are misleading for predicting the robot pose on this type of terrain, the height of the grass is suppressed by the "soft terrain adjustment" part of the network (see Fig. 2.3). The rigid objects such as cars, stones, trees, roads, or buildings remain at the original level. The results are summarized in Table 2.2. Since the ground-truth supporting terrain is generally unknown, we semi-manually annotated a selected subset of terrains, by fitting the local ground plane between robot-terrain contact points. The heightmap reconstruction error is denoted with * to point out that it is computed on the manually annotated subset of the dataset.

The benefit of backpropagating the pose consistency loss to the heightmap reconstructing network is even more obvious since it allows to predict the correct height on flexible terrains. Consequently, the pose predicting error and reconstruction error of proposed methods (r+p and r+kkt) is reduced by 50% with respect to the state-of-the-art methods (r and li). The accuracy of r+kkt slightly outperforms r+p, since it is not biased by the g_ω regressor.

2.1.4 Conclusion

We have addressed the problem of learning to predict supporting terrain from SLAM-optimized 3D maps and robot poses. We have demonstrated that using the pose-consistency loss, which expresses the physical consistency of predicted terrain with the robot pose, improves the accuracy of predicted heightmaps and poses. In particular, two different pose consistency losses have been proposed: (i) KKT-loss and (ii) pose-predicting loss. The KKT-loss has been defined as the distance from the necessary conditions of the first principle model for a given ground-truth pose. The pose-predicting loss, first learns the pose predicting regressor and then minimizes the distance between predicted pose and ground-truth pose. In contrast to KKT-loss, the pose predicting loss requires prior learning and suffers from the regressor bias. The bias is strong since flexible terrains and lidar-unfriendly surfaces cannot be used for the training of the regressor. On the other hand, the pose-predicting loss does not require any inner-loop optimization, therefore its backpropagation is faster. The experimental comparison shows that the proposed end-to-end differentiable architecture with proposed pose consistency losses reduces the error of predicted terrains and robot poses by 50% on the partially flexible dataset and 20% on the strictly rigid dataset when compared to state-of-the-art methods. Generalization of the trained model depends on a particular type of the robot model and a variety of training terrains. We believe that KKT-loss is directly usable for a broad range of mobile platforms such as wheeled or skid-steer robots, however its generalization for legged robots is unclear.

The dataset and codes replicating the reported experiments are made

publicly available. We also provide the fully differentiable implementation of the KKT-loss, which can be simply included in any future learning methods through the standard PyTorch interfaces such as `kkt_loss(net, robot).backward()`.

2.2 Tactile terrain reconstruction

Advances in robotic technology allow mobile robots to be deployed in gradually more and more challenging environments. However, real-world conditions often complicate or even prohibit the adoption of classical approaches to localization, mapping, navigation, or teleoperation. When rescuers operate a UGV (unmanned ground vehicle) during joint experiments in the TRADR project[3], which develops novel software and technology for human-robot teams in disaster response efforts [45], we have to deal with such problems.

One of the crucial fail-cases is the presence of dense smoke that blocks camera view and spoils laser measurements, creating false obstacles in front of the robot (Fig. 2.7). Without exteroceptive measurements, classical approaches to robot SLAM cannot be used. Localization can only be in the dead-reckoning sense, and the operator of the robot has to rely solely on

[3] `http://www.tradr-project.eu`

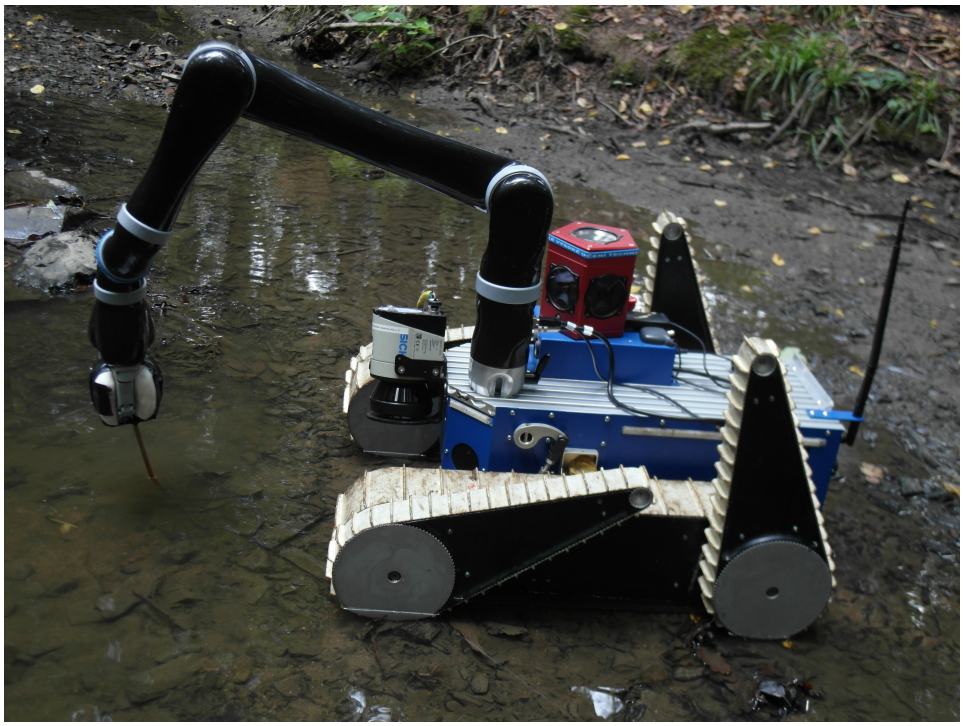


Figure 2.6: Search-and-rescue UGV developed as part of the TRADR projects, equipped with a *Point Grey Ladybug 3* omni-directional camera, a rotating *SICK LMS-151* laser range finder and a *Kinova Jaco* robotic arm, in our case used for contact exploration of terrain with a wooden stick tool.

the maps created up to the point of the sensor outage. In an industrial environment consisting of many hazardous areas, driving blind can lead to damage or loss of the robot.

Therefore, we propose a combined hardware and software solution to predict the profile of terrain underneath and in front of the tracked robot. The algorithm exploits a prototype of a force sensor array installed inside a track of the robot, a robotic arm attached to the robot, proprioceptive measurements from joints and (see Fig. 2.6) an inertial measurement unit (IMU), and information learned from a dataset of traversed terrains. The prototype of the force sensor (Fig. 2.8, 2.9) is suitable for tracked robots and is installed between rubber track and its support, allowing it to serve as a tactile sensor. The arm is used to measure height of terrain outside the reach of the force sensor as contact between the arm end-effector and the terrain. The height of terrain that cannot be measured directly is estimated by sampling from a joint probability distribution of terrain heights, conditioned by proprioceptive measurements (geometric configuration of the robot, torques in joints and attitude of the robot) and learned from a training dataset consisting of real-world examples of traversed terrains. The estimates of terrain profile are used as a partial substitute for missing laser range-finder data that would reveal obstacles or serve as an input our adaptive traversability solution (Chapter 4).

Our **contribution** is twofold: we designed a new force sensor suitable for tracked robots as well as an algorithm that uses proprioceptive and tactile measurements to estimate terrain shape in conditions that prohibit usage of cameras and laser range-finders. We extended this solution with robotic arm to deal with special cases when the predictions have too high uncertainty.

2.2.1 Related work

The problem of terrain characterization primarily using proprioceptive sensors, but also by sonar/infra-red range-finders and by a microphone is discussed in [46]. The authors exploit neural networks trained for each sensor and demonstrate that they are able to recognize different categories: gravel, grass, sand, pavement and dirt surface. Furthermore, they present a concept of terrain-characteristic curves that establish relationship between currents in motors driving the main wheels and resulting angular rate of the robot. In [47] they took a similar approach to train a regression function that maps from a space of features extracted from inertial sensors to parameters that compensate slippage in track odometry. In both cases the aim was to improve localization and control of the robot. This chapter focuses more on the actual terrain profile prediction, necessary for successful traversal.

Lack of sufficient visual information related to danger of collision with obstacles is addressed in [48]: decision whether it is safe to navigate through vegetation is based on wide-band radar measurements since it is impossible to detect solid obstacle behind vegetation from laser range-finder or visual data. Artificial whiskers offer an alternative solution; they mimic facial whiskers

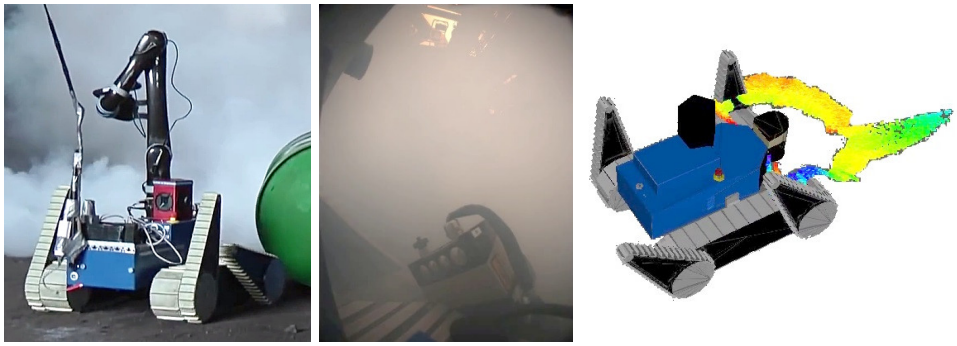


Figure 2.7: Left: UGV robot approaches smoke area. **Middle:** Example of visual information that the operator sees inside a cloud of smoke. **Right:** Output of the laser range-finder (rainbow-colored point cloud in the right half of the image). Laser beams are randomly reflected by smoke particles. The resulting 3D point cloud is just noise close to the robot.

of animals and using them as a tactile sensor is a promising way to explore areas, which are prohibitive to standard exteroceptive sensors. Work of [49] presents a way to use array of actively actuated whiskers to discriminate various surface textures. In [50], similar sensor is used for a SLAM task. Two sensing modalities—the whisker sensor array and the wheel odometry are used to build a 2D occupancy map. Robot localization is then performed using particle filter with particles representing one second long "whisk periods". During these periods, the sensor actively builds local model of the obstacle it touches. Unfortunately, design of our platform does not allow using such whiskers due to rotating laser range-finder.

Relation between shape of terrain that we are interested in and configuration of the flippers is investigated in [51]. The authors exploit the knowledge about robot configuration and torques in joints to define a set of rules for climbing and descending obstacles not observed by exteroceptive sensors. We investigated this problem in Chapter 4 by introducing the adaptive traversability algorithm based on machine learning. We collected features from both proprioceptive and exteroceptive sensors to learn a policy that ensures safe traversal over obstacles by adjusting robot morphology. Our motivation coincided with [51], aiming primarily to lower the cognitive load of the operator.

On contrary to the approaches exploiting only simple contact sensors, we extend our sensory suite with a robotic arm for further active perception for cases if necessary. Related to the active perception, relevant ideas and techniques come from the field of haptics. The work of [52] proposes to create models of objects in order to be able to grasp them. The idea is to complement visual measurements by tactile ones by strategically touching the object in areas with high shape uncertainty. For this purpose they use Gaussian processes (GP, [53, 54]) to express the shape of the object. We take a similar approach: we choose parts of terrain to be explored by the robotic arm based on uncertainty of the estimate resulting from the sampling process

(Sec. 2.2.3). Probabilistic approach to express uncertainty in touched points is also described in [55], where only tactile sensors of a robotic hand are used to reconstruct the shape of an unknown object.

2.2.2 Sensors

Prototype of force sensor

To obtain well-defined contact points with the ground, we decided to take advantage of the flippers that can reach in front of the robot and are designed to operate on dirty surfaces or sharp edges. The original mechatronics of the robot allows to measure torque in flipper servos and thus detect physical contact between flippers and the environment. To be able to locate the contact point on the flipper exactly, we designed a thin force sensor between the rubber track and its plastic support (see Fig. 2.8, 2.9). Since it is a first prototype, we use it only in one flipper and consider only symmetrical obstacles or steps. The sensor construction is a sandwich of two thin stripes of steel with *FSR 402* sensing elements between them which allows the rubber track to slide over it while measuring forces applied onto the track. There are six force sensing elements; the protecting sheet of steel distributes the force among them, the sensor is thus sensitive along its whole length.

The *FSR 402* sensing elements are passive sensors that exhibit decrease in resistance with increasing force; the force sensitivity range is $0.1 - 10\,\mathrm{N}$. To measure the resistance, we connect them in series with a fixed reference resistor forming a voltage divider. We apply $5\,\mathrm{V}$ to this divider and measure voltage on the reference resistor. We use an analog-to-digital converter expansion board for the Raspberry Pi computer to read the six voltages. We calibrate the voltage values for initial bias caused by the sandwich construction.

Fig. 2.10 shows three examples of the sensor readings. The first case consists of a flipper touching flat floor. Although one would expect to see more or less equal distribution of the contact force along the flipper track, the torque generated by the flipper actually lifts the robot slightly and thus, most of the force concentrates at its tip (element n. 6). Compare this case

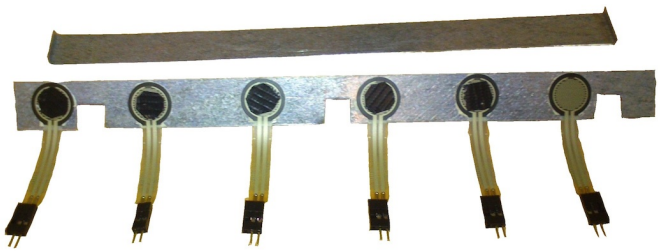


Figure 2.8: Prototype of the flipper force sensor: array of six sensing elements (FSR 402) is covered by a stripe of steel, forming a thin sensor that fits between the rubber track and the plastic track support. The stripe of steel protects the sensors from the moving rubber track and distributes measured force amongst them.

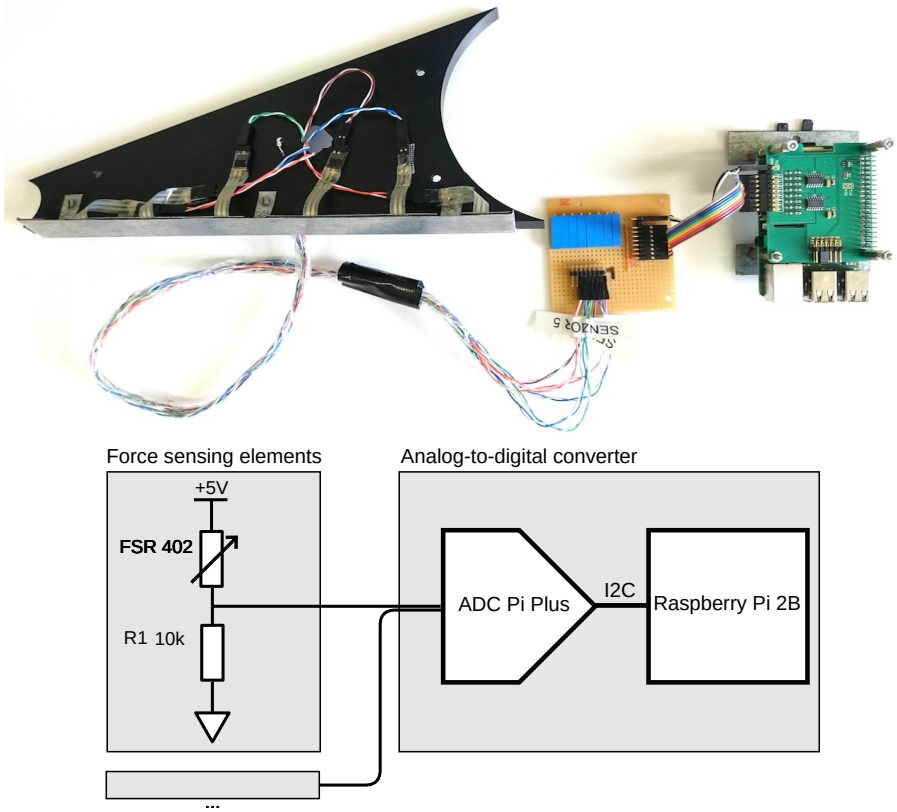


Figure 2.9: Top: The sensor mounted to the plastic track support. **Bottom:** The sensing elements are passive sensors that exhibit decrease in resistance with applied force. For each sensing element, we use a reference resistor to form a voltage divider; we obtain voltage inversely proportional to the resistance of the FSR 402 elements.

with the third one, where the pose of the robot prohibits the lifting effect, and we therefore see the expected result. The second case (middle) shows an example of a touch in one isolated point.

Robotic arm

The UGV is equipped with a *Kinova Jaco* robotic arm[6], see Fig. 2.7-left. It is a 6-DOF manipulator (with one extra DOF in each finger) capable of lifting 1.5 kg. For our approach, it is used for tactile exploration of surroundings up to cca. 50 cm around the robot. For the terrain sensing, robotic arm holds a tool with a wooden stick—this setup protects its fingers from being broken when pushing against ground. It also allows the robot to measure the height of terrain in a chosen point by gradually lowering the arm until upsurge of actuator currents indicates contact with ground (there are currently no touch sensors) [56]. Accuracy of the measurement is 3 cm (standard deviation). However, the process of unfolding the arm, planning and execution of the

[6] http://www.kinovarobotics.com/service-robotics/products/robot-arms

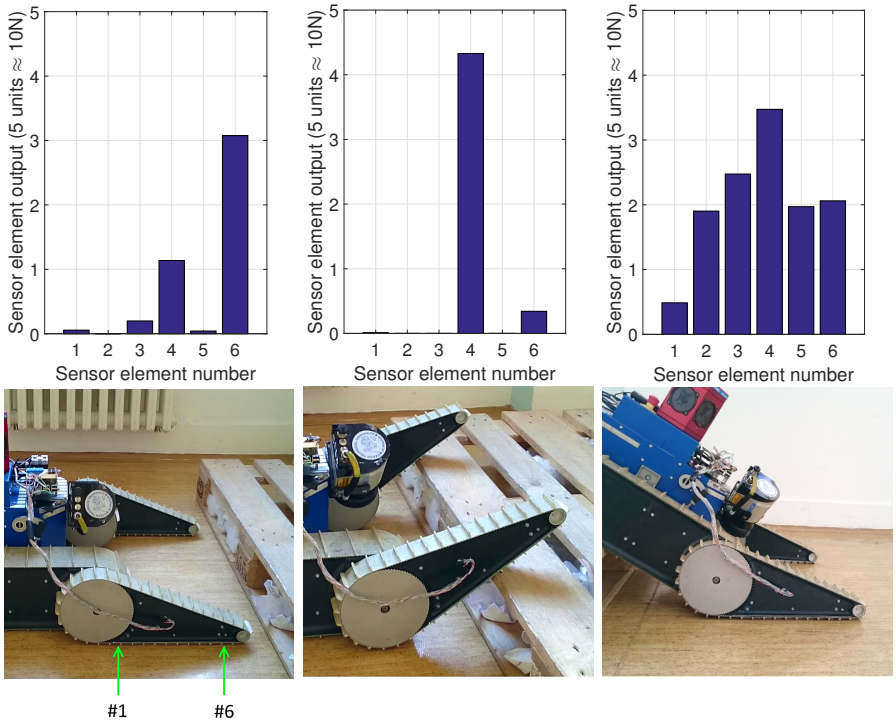


Figure 2.10: Examples of the force sensor readings. **Top:** raw readings of each sensing element, only corrected for bias. **Bottom:** the photos of the moments of the acquisition of the reading.

desired motion and finally folding back to home position can easily take 45 s. Therefore, it is practical to use the arm for this purpose only in situations when the gain from the additional information overweights the cost of time spent to get it.

2.2.3 Terrain shape reconstruction

When robot is teleoperated operator's awareness is based on camera images and the map. In the presence of smoke, both of these modalities are useless, see output of the operator console in the presence of smoke shown in Fig. 2.7. We propose active tactile exploration mode (ATEM), in which flippers and robotic arm autonomously explores the terrain shape in close vicinity of the robot. Estimated terrain shape and expected reconstruction accuracy are eventually displayed to the operator.

If ATEM is requested by the operator, robot first adjusts flippers to press against the terrain and capture proprioceptive measurements. Then the initial probabilistic reconstruction of the underlying terrain shape is estimated from the captured data. If the reconstruction is ambiguous, the robotic arm explores the terrain height in the most inaccurate place. Eventually, the probabilistic reconstruction is repeated. As a result, reconstructed terrain shape with estimated variances is provided. The ATEM procedure is summarized in

Algorithm 2. The rest of this section provides detailed description of particular steps.

Algorithm 2: Active tactile exploration mode for terrain shape reconstruction.

Variables: **h** - vector of terrain bin heights,
v - vector of height variances,
s - vector of proprioceptive measurements.

while *ATEM is requested* **do**
  stop_robot;
  `// Invoke flipper exploration mode`
  while torque_in_front_flippers < threshold **do**
    push_flippers_down;
  s = capture_proprioceptive_measurements();
  `// Perform kinematic reconstruction`
  [**h**, **v**] = kinematic_reconstruction(**s**);
  `// Perform probabilistic reconstr.`
  [**h**, **v**] = probabilistic_reconstruction(**h**, **v**, **s**);
  `// Invoke arm exploration`
  if any(**v** > threshold) **then**
    [**h**, **v**] = arm_exploration(**h**, **v**);
    [**h**, **v**] = probabilistic_reconstruction(**h**, **v**, **s**);
  move_forward;

Flipper exploration mode

As soon as the ATEM is requested, the robot halts driving and adjusts angles of front flippers towards ground until they reach an obstacle or the ground. They keep pressing against it by defined torque while vector of proprioceptive measurements **s** is captured. We measure: (i) pitch of the robot (estimated from IMU sensor), (ii) angles of flippers, (iii) currents in flipper engines, and iv) 6-dimensional output of the force sensor.

Kinematic reconstruction

The terrain shape is modeled by Digital Elevation Map (DEM), which consists of eleven 0.1 m-wide bins. If there is only one isolated contact point sensed by the force sensor and the force surpasses experimentally identified threshold (see Fig. 2.10, second case), the height $\mathbf{h}_i$ of the terrain in the corresponding bin i is estimated by a geometric construction from known robot kinematics, using the attitude of the robot, configuration of joints and the position of the contact point on the flipper. Variance $\mathbf{v}_i$ for the corresponding force sensor is

set to an experimentally estimated value. The remaining $\mathbf{h}_i$ and $\mathbf{v}_i$ values are set to non-numbers.

Probabilistic reconstruction

In the probabilistic reconstruction procedure, the vector of heights $\mathbf{h}$ and the vector of variances $\mathbf{v}$ are estimated by the Gibbs sampling [57]. Let us denote the set of all bins J and the set of all bins in which the reconstruction is needed by I (i.e. those which height was not estimated in the kinematic reconstruction procedure or measured by the robotic arm). We use the Gibbs sampling to obtain height samples $\mathbf{h}_I^k$, $k = 1 \dots K$ from the joint probability distributions $p(\mathbf{h}_I|\mathbf{h}_{J\setminus I}, \boldsymbol{s})$ of all missing heights $\mathbf{h}_I$. Missing heights $\mathbf{h}_I$ are reconstructed as the mean of generated samples, variances $\mathbf{v}_I$ are estimated as the variance of samples.

In the beginning, the missing heights $\mathbf{h}_I$ are randomly initialized. The k-th sample $\mathbf{h}_I^k$ is obtained by iterating over all unknown bins $i \in I$ and generating their heights $\mathbf{h}_i^k$ from conditional probabilities $p(\mathbf{h}_i|\mathbf{h}_{J\setminus i}, \mathbf{s})$. The conditional probability is modeled by Gaussian process [53, 54, 58] with a squared exponential kernel.

To train the conditional probabilities, we collected real-world trajectories with i) sensor measurements $\mathbf{s}^u$ and ii) corresponding terrain shapes $\mathbf{h}^u$ estimated from the 3D laser map for $u = 1 \dots U$. The i-th conditional probability $p(\mathbf{h}_i|\mathbf{h}_{J\setminus i}, \mathbf{s})$ is modeled by one Gaussian process learned from the training set $\{[(\mathbf{h}_{J\setminus i}^1, \mathbf{s}^1)^\top, \mathbf{h}_i^1], \dots, [(\mathbf{h}_{J\setminus i}^U, \mathbf{s}^U)^\top, \mathbf{h}_i^U]\}$.

Modeling the bin height probabilities as normal distributions is a requirement laid by the Gaussian process. However, it allows samples of the bin height that collide with the body of robot, which is of course physically impossible. We propose to use Gaussian distribution truncated by known kinematic constraints, in which the samples are constrained by the maximal height that does not collide with the body of the robot. We discuss impact of this modification in the Section 2.2.4.

Active arm exploration

We use the robotic arm to measure the height of the terrain in bins the flippers cannot reach. The measurement taken by the robotic arm is reasonably accurate and precise but in its current state it takes about 45s to complete. If the probabilistic reconstruction contains bins with variance $\mathbf{v}$ higher than a user-defined threshold, the robotic arm is used to measure the height in the most uncertain bin, i.e. the bin $j = \arg\max_i \mathbf{v}_i$. The height sensed in the given bin is then fixed and the probabilistic reconstruction process is repeated.

2.2.4 Experiments

In qualitative experiments, we focus on typical cases of terrain profile shapes and discuss performance of different settings of our algorithm. In quantitative

experiments, we present performance statistics over the whole testing dataset.

The *training dataset* consists of 28 runs containing driving on flat terrain, approaching obstacles of two different heights, traversing them and descending from them back to flat ground. Shape of obstacles selected for the dataset reflects the industrial accident scenario of the TRADR project - the environment mostly consists of right-angle-shaped concrete and steel objects. From the recorded runs, we have extracted approximately 1400 individual terrain profile measurements for training. The whole training dataset was recorded indoors on flat hard surfaces. The *testing dataset* was recorded outdoors and combines uneven grass, stone and rough concrete surfaces. It contains more complex obstacles with various heights (different from those seen in the training dataset). The testing dataset consists of more than 300 terrain profiles with the corresponding sensory data. Ground truth necessary for training and testing was created manually by sampling scans from the laser range-finder recorded during the experiments.

We compare four different algorithms for terrain profile prediction. The baseline approach [59] uses only the IMU sensor and angles of flippers, we call it PA (pitch + angle of flippers) for short. The second setup uses the same data and adds the probability of terrain height being adapted in the way described in Section 2.2.3. If the sampled height collides with the robot, the sample is set to the maximal possible height that is not in collision. The approach is called PAc (pitch + angle of flippers; constrained). The third approach adds the flipper force sensor; measured data are used in two ways. If the force measured by a sensor element exceeds a threshold (experimentally set on 2 units), then the height of the bin is computed from kinematics of the robot (pitch and flipper angles and position of the sensor element) and the bin is fixed and excluded from the Gibbs sampling step. It should be noted however, that the measured forces are used even if they are not bigger than the threshold – they are part of the proprioceptive data $\mathbf{s}$. The approach is called as PAFc (pitch + angle of flippers + flipper force sensor; constrained). The fourth approach adds direct terrain measurement: we simulate use of the robotic arm for measurements the terrain height in bins with high uncertainty [56]. The fourth approach is called as PAFAc (pitch + angle of flippers + flipper force sensor + robotic arm; constrained).

Qualitative Evaluation

In this section we present typical terrain profiles and robot actions: flat ground, two steps with different height, climbing up a step and stepping down of a step. We compare performance of two algorithms: i) PAc uses the kinematic constraints when sampling sampling but does not use the force sensors (light blue line in the plots) ii) PAFc algorithm which uses the force sensors (green line and bars). The last two bars marked yellow in order to emphasize the predictions are learnt from training dataset and we do not have enough information to correct the predictions from the sensing by flippers. We use mean of the (Gibbs) samples as the predicted value (connected by lines)

and 0.1 and 0.9 quantiles for displaying dispersion of samples (errorbars). The point $(0, 0)$ coincides with the location of the IMU sensor inside the robot body. The depicted sketch of the robot: the pitch is estimated by IMU, flipper angle is directly measured. When the robot lies on a flat ground, Fig. 2.11, contact point is sensed by the sixth element. The force measurement reduces uncertainty mainly in positions $0.3 - 0.7$ m.

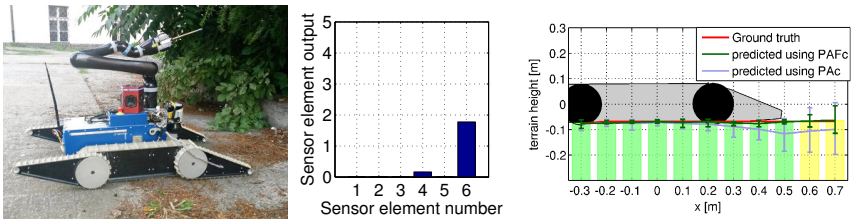


Figure 2.11: Left: photo of the robot on a concrete ground. **Middle:** measured forces. **Right:** terrain reconstruction.

Climbing up a step cases are depicted in Fig. 2.12. The higher 0.28 m step obstacle is on top. The fifth sensor element measures the force that is bigger than threshold and the height in the bin 0.4 is fixed and not sampled. Note that algorithm PAc which does not use force sensor cannot predict the exact edge location. This fact is indicated by big dispersion of samples in bins 0.3 and 0.4. The second situation shown in Fig. 2.12 is the lower step. The height of the lower step 0.2 m was correctly measured by the sixth element of the force sensor.

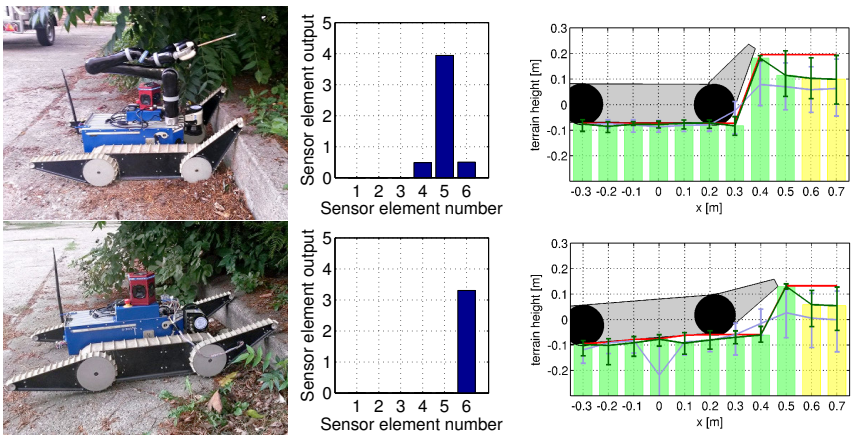


Figure 2.12: Top: 0.28 m step. **Bottom:** 0.2 cm step. Note the reduced uncertainty for the PAFc – green line and errorbars.

Climbing up and stepping down cases are displayed in the Fig. 2.13. Variances in the bins that are underneath the robot are high because we do not have enough information to estimate the correct heights. Still, the means are correct due to models learnt from the training data.

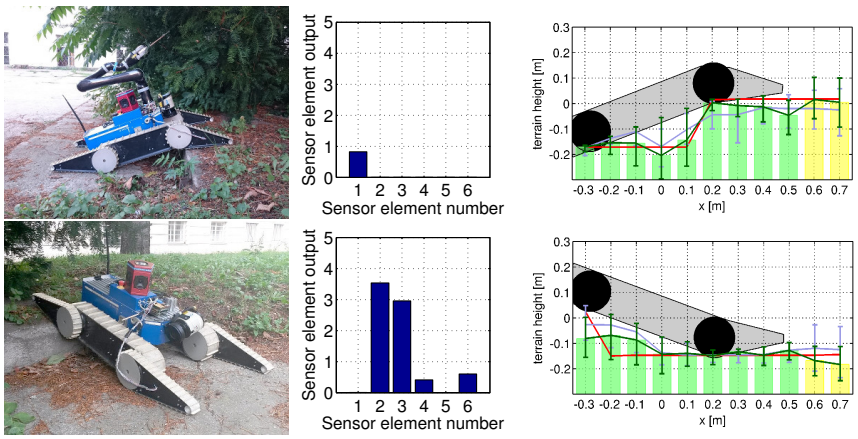


Figure 2.13: Top: climbing up a step. **Bottom:** stepping down of a step.

Quantitative Evaluation

At first, we measure the direct effect of the force measurement on the accuracy of the height estimates. The graph on Fig 2.14-left, shows the height error frequency of the DEM-bins that are underneath the front flipper. Note that the attribute "underneath the front flipper" is not fixed, it depends the flipper angle. The force sensor indeed improves the accuracy over the using flipper angle only. The second experiment studies the statistics for all the DEM-bins individually, see Fig 2.14-right. Adding the kinematic constraint c naturally improves the estimates of the bins underneath the robot body $(-0.3 \ldots 0.2)$. Using the force sensors (PAFc) improves height estimates of the DEM-bins underneath the front flipper $(0.3 \ldots 0.5)$. The bins in front of the flippers, i.e. (0.6 and 0.7) are directly measurable only by the arm exploration. It is thus obvious that including the measurement by arm (PAFAc) has the dominant effect.

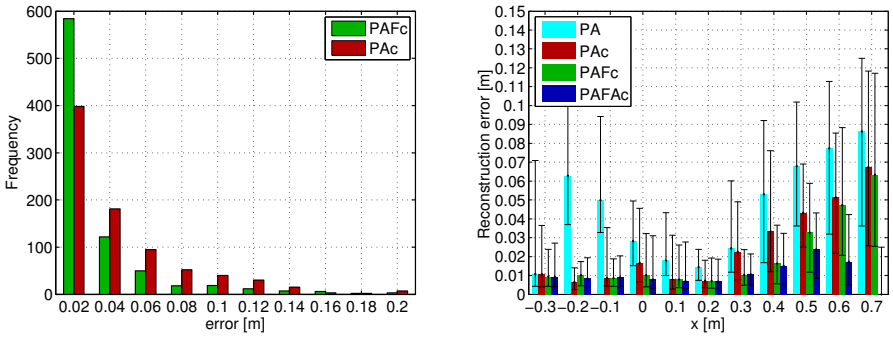


Figure 2.14: Left: Evaluation of reconstruction quality in the bins that are under the flippers. **Right:** Quantitative evaluation of terrain profile reconstruction – for all the DEM bins. Median, 1st quartile and 3rd quartile of errors are shown

2.2.5 Conclusion

We demonstrated a combined hardware and software solution that enhances sensor suite and perception capabilities of our mobile robot. We focused our efforts on enabling proprioceptive terrain shape prediction for cases when vision and laser measurements are unavailable or deteriorated (such as in presence of a dense smoke). To evaluate our proposed solution experimentally, we designed and compared four algorithms—four possible approaches for proprioceptive terrain shape reconstruction: simple kinematics based approach, constrained kinematics, constrained kinematics with force sensors, and constrained kinematics with both force sensors and robotic arm—intended for special cases, where terrain prediction reaches very high uncertainty. From the presented qualitative and quantitative experimental evaluation we can clearly see that enhancing the sensor suite with force sensor array proves to be superior. The proposed algorithm, which combines Gaussian processes followed by Gibbs sampling, was successfully implemented on-board the robot to process the raw force measurements and perform the actual terrain shape prediction in a probabilistic manner. We certainly do not claim this is the only and best way to perform such terrain prediction, but, it definitely serves as sufficiently robust and accurate proof of concept for intended deployment. As part of this concept, the integration of robotic arm for active perception in cases when the prediction based on force sensors is too uncertain proved to be important.

Chapter 3

Active perception

The state of the robot environment is typically not fully known. The robot follows the policy that leads to the objective defined by the task and collects the additional data from the environment that helps to achieve the goal. The special case, when the objective is obtaining knowledge of the environment, is called active perception. Consider time-critical scenarios such as autonomous driving or search-and-rescue missions. In such scenarios, if the sensor's field of view is limited or the collected data are sparse, we have to select which data to collect to get the most information about the surrounding. We are proposing active perception methods within this chapter.

In Section 3.1 we propose an active perception method for depth sensors, which allow individual control of depth-measuring rays, such as the newly emerging solid-state lidars. The method simultaneously (i) learns to reconstruct a dense 3D occupancy map from sparse depth measurements, and (ii) optimizes the reactive control of depth-measuring rays. To make the first step towards the online control optimization, we propose a fast prioritized greedy algorithm, which needs to update its cost function in only a small fraction of possible rays. An experimental evaluation on the subset of the KITTI dataset demonstrates significant improvement in the 3D map accuracy when learning-to-reconstruct from sparse measurements is coupled with the optimization of depth measuring rays. The Section 3.1 is based on the previously published work "Learning for Active 3D mapping", which was published on IEEE International Conference on Computer Vision (ICCV) [4]. This conference has an A* CORE Rank and average acceptance rate 25%. Moreover this publication was accepted for the oral presentation where the acceptance rate was 2.6%.

Section 3.2 covers our work on the active perception problem of victim segmentation during a search-and-rescue exploration mission. The robot is equipped with a multi-modal sensor suite consisting of a camera, lidar, and pan-tilt thermal sensor. The robot enters unknown scene, builds 3D model incrementally, and the proposed method simultaneously (i) segments the victims from incomplete multi-modal measurements and (ii) controls the motion of the thermal camera. Both of these tasks are difficult due to the lack of natural training data and the limited number of real-world trials. In particular, we overcome the absence of training data for the segmentation

task by employing a manually designed generative model, which provides the semi-synthetic training dataset. The limited number of real-world trials is tackled by self-supervised initialization and optimization-based guiding of the motion control learning. In addition to that, we provide a quantitative evaluation of the proposed method on several real testing scenarios using the real search-and-rescue robot. We also provided a dataset which will allow for further development of algorithms on the real data. The Section 3.2 is based on the submission "Simultaneous Exploration and Segmentation for Search and Rescue" [2] that was published in Journal Field of Robotics (Q1 journal with impact factor 3.77).

3.1 Active 3D mapping

Development of autonomous vehicles such as self-driving cars or ground robots has attracted substantial attention of the robotics community in the last few years. One of the reasons is that an 3D perception, which is an essential component for many fundamental capabilities such as emergency braking, predictive active damping or self-localization from offline maps [60], has finally become accurate enough.

All autonomous vehicles require a sensor providing high resolution and long range 3D measurements. Since state-of-the-art rotating lidars are very expensive, heavy and contain moving parts prone to mechanical wear, several manufacturers have announced the development of cheaper, lighter, smaller and motionless solid-state lidars (SSL), which should become available soon [61]. In contrast to rotating lidars, the SSL uses an optical phased array as a transmitter of depth measuring light pulses. Since the built-in electronics can independently steer pulses of light by shifting its phase as it is projected through the array, the SSL can focus its attention on the parts of the scene important for the current task. Task-driven reactive control steering hundreds of thousands of rays per second using only an on-board computer is a challenging problem, which calls for highly efficient parallelizable algorithms. As a first step towards this goal, we propose an active mapping method for SSL-like sensors, which simultaneously (i) learns to *reconstruct a dense 3D voxel-map* from sparse depth measurements and (ii) optimize the reactive *control of depth-measuring rays*, see Fig. 3.1. The proposed method is evaluated on a subset of the KITTI dataset [62], where sparse SSL measurements are artificially synthesized from captured lidar scans, and compared to a state-of-the-art 3D reconstruction approach [18].

The main contribution lies in proposing a computationally tractable approach for very high-dimensional active perception task, which couples learning of the 3D reconstruction with the optimization of depth-measuring rays. Unlike other approaches such as active object detection [63] or segmentation [64], SSL-like reactive control has significantly higher dimensionality of the state-action space, which makes a direct application of unsupervised reinforcement learning [63] prohibitively expensive. Unlike the active SLAM, we are mostly interested in a local 3D map, which allows for reactive control

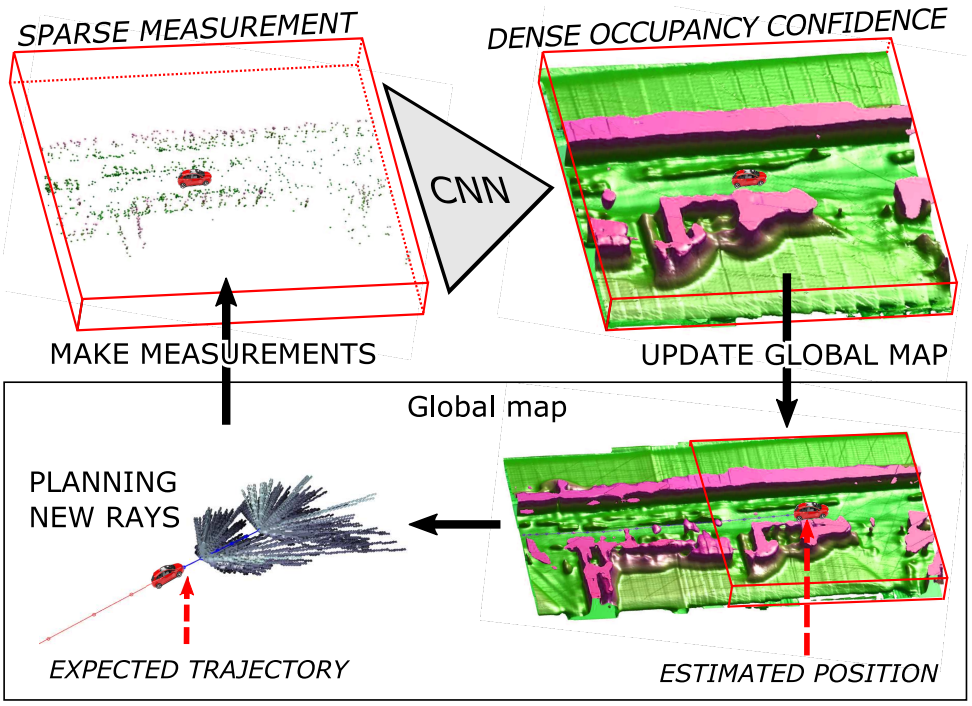


Figure 3.1: Iteratively learned deep convolutional network reconstructs local dense occupancy map from sparse depth measurements. The local map is registered to a global occupancy map, which in turn serves as an input for the optimization of depth-measuring rays along the expected vehicle trajectory.

of the vehicle, rather than the global map which would require global motion rectifications such as loop closures. In such a short horizon, the motion estimated from odometry, IMU and GPS is sufficiently accurate, therefore localization is not an issue. In contrast to the discriminative voxel reconstruction approaches, the map being reconstructed has also significantly higher dimensionality. Keeping the on-board reactive control in mind, we propose prioritized greedy optimization of depth measuring rays, which in contrast to a naïve greedy algorithm re-evaluates only 1/500 rays in each iteration.

The 3D mapping is handled by an iteratively learned convolution neural network (CNN), as CNNs proved their superior performance in [18, 65]. The iterative learning procedure stems from the fact that both (i) the directions in which the depth should be measured and (ii) the weights of the 3D reconstruction network are unknown. We initialize the learning procedure by fixing the sparse depth measurements in randomly generated directions. The sparse depth measurements input the learning of the 3D mapping network, which estimates the probability of each particular voxel being occupied. The iterative learning then fixes the weights of the 3D mapping network and uses the expected reconstruction (in)accuracy in each voxel to optimize directions of all depth-measuring rays along the expected vehicle trajectory. We cannot directly reconstruct the map on the optimized depth measurements, because the training distribution of the 3D mapping network does not correspond to

the one obtained by the planning. In order to reduce the training-planning discrepancy, the mapping network is re-learned on optimized sparse measurements and the whole process is iterated until it converges.

3.1.1 Related work

Active perception has been widely studied in many robotics applications ranging from exploration and active SLAM [66] to active object detection [63] and segmentation [64]. In contrast to these applications, the active SSL-mapping has significantly higher dimensionality of the action space, which makes a direct application of known approaches such as unsupervised reinforcement learning [63] prohibitively expensive. Unlike the active SLAM, we are mostly interested in a local 3D map, which allows for reactive control of the vehicle, rather than the global map which would require global motion rectifications such as loop closures. In such a short horizon, the motion estimated from odometry, IMU and GPS is sufficiently accurate, therefore localization is not an issue. In contrast to the discriminative voxel reconstruction approaches, the map being reconstructed has also significantly higher dimensionality. Nevertheless we follow the main paradigm, which achieved state-of-the-art performance in the most of the active perception tasks: *discriminative learning of the target perception task coupled with the active component*.

A generative model proposed by [65], termed Deep Belief Network, learns joint probability distribution $p(\mathbf{x}, y)$ of complex 3D shapes $\mathbf{x}$ across various object categories y. Their model assumes that all cameras are registered in a common reference frame so that 2.5D images can be converted to a 3D occupancy grid ($30 \times 30 \times 30$ voxels). The authors suggest to use the model for Next-Best-View prediction via rendering view hypotheses by Gibbs sampling and selecting the view maximizing mutual information between class label y and the newly observed voxels conditioned on current observation.

End-to-end learning of stochastic motion control policies for active object and scene categorization is proposed by [63]. Their CNN policy successively proposes views to capture with RGB camera to minimize categorization error. The authors use a look-ahead error as an unsupervised regularizer on the classification objective. Andreopoulos et al. [67] solve the problem of an active search for an object in a 3D environment. While they minimize the classification error of a single yet apriori unknown voxel containing the searched object, we minimize the expected reconstruction error of all voxels. Also, their action space is significantly smaller than ours because they consider only local viewpoint changes at the next position while the SSL planning chooses from tens of thousands of rays over a longer horizon.

3.1.2 Overview of the active 3D mapping

In this section we introduce notation and give an overview of the active mapping pipeline depicted in Fig. 3.1. We assume that the vehicle follows a

known path consisting of L discrete positions and a depth measuring device (SSL) can capture at most K rays at each position. The set of rays to be captured at position l is denoted J_l.

We denote $\mathbf{Y}$ the global ground-truth occupancy map, $\hat{\mathbf{Y}}$ its estimate, and $\mathbf{X}$ the map of the sparse measurements. All these map share common global reference frame corresponding to the first position in the path. For each of these maps there are local counterparts $\mathbf{y}_l, \hat{\mathbf{y}}_l$, and $\mathbf{x}_l$, respectively. Local maps corresponding to position l all share a common reference frame which is aligned with the sensor and captures its local neighborhood. The global ground-truth map $\mathbf{Y}$ is used to synthesize sensor measurements $\mathbf{x}_l$ and to generate local ground-truth maps $\mathbf{y}_l$ for training.

The active mapping pipeline, consisting of a measure-reconstruct-plan loop, is depicted in Fig. 3.1 and detailed in Alg. 3. Neglecting sensor noise, the set

Algorithm 3: Active mapping

1. Initialize position $l \leftarrow 0$ and select depth-measuring rays randomly.
2. Measure depth in the directions selected for position l and update global sparse measurements $\mathbf{X}$ and dense reconstruction $\hat{\mathbf{Y}}$ with these measurements.
3. Obtain local measurements $\mathbf{x}_l$ by interpolating $\mathbf{X}$.
4. Compute local occupancy confidence $\hat{\mathbf{y}}_l = \mathbf{h}_\theta(\mathbf{x}_l)$ using the mapping network $\mathbf{h}_\theta$.
5. Update global occupancy confidence $\hat{\mathbf{Y}} \leftarrow \hat{\mathbf{Y}} + \hat{\mathbf{y}}_l$.
6. Plan depth-measuring rays along the expected vehicle trajectory over horizon L given reconstruction $\hat{\mathbf{Y}}$.
7. Repeat from line 2 for next position $l \leftarrow l + 1$.

of depth-measuring rays obtained from the planning, the measurements $\mathbf{x}_l$, and the resulting reconstruction $\hat{\mathbf{Y}}$ can all be seen as a deterministic function of mapping parameters θ and $\mathbf{Y}$. If we assume that that ground-truth maps $\mathbf{Y}$ come from a probability distribution, both learning of θ and planning of the depth-measuring rays approximately minimize common objective

$$\mathbb{E}_{\mathbf{Y}}\{\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}(\theta, \mathbf{Y}))\}, \tag{3.1}$$

where $\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = \sum_i w_i \log(1 + \exp(-Y_i \hat{Y}_i))$ is the weighted logistic loss, $Y_i \in \{-1, 1\}$ and $\hat{Y}_i \in \mathbb{R}$ denote the elements of $\mathbf{Y}$ and $\hat{\mathbf{Y}}$, respectively, corresponding to voxel i. In learning, $w_i \geq 0$ are used to balance the two classes, *empty* with $Y_i = -1$ and *occupied* with $Y_i = 1$, and to ignore the voxels with unknown occupancy. We assume independence of measurements and use, for corresponding voxels i, additive updates of the occupancy confidence $\hat{Y}_i \leftarrow \hat{Y}_i + h_i(\mathbf{x}_l)$ with $h_i(\mathbf{x}_l) \approx \log(\Pr(Y_i = 1|\mathbf{x}_l)/\Pr(Y_i = -1|\mathbf{x}_l))$. $\Pr(Y_i = 1|\mathbf{x}_l)$ denotes the conditional probability of voxel i being occupied given measurements $\mathbf{x}_l$ and $\sigma(\hat{Y}_i) = 1/(1 + e^{-\hat{Y}_i})$ is its current estimate.

3.1.3 Learning of reconstruction network

The learning is defined as approximate minimization of Equation 3.1. Since (i) the result of planning $\mathbf{x}_l(\theta, \mathbf{Y})$ is not differentiable with respect to θ and (ii) we want to reduce variability of training data[1], we locally approximate the criterion around a point θ^0 as

$$\mathbb{E}_{\mathbf{Y}}\{\sum_l \mathcal{L}(\mathbf{y}_l, \mathbf{h}_\theta(\mathbf{x}_l(\theta^0, \mathbf{Y})))\} \tag{3.2}$$

by fixing the result of planning in $\mathbf{x}_l(\theta^0, \mathbf{Y})$. We also introduce a canonical frame by using the local maps instead of the global ones, which helps the mapping network to capture local regularities. The learning then becomes the following iterative optimization

$$\theta^t = \arg\min_\theta \mathbb{E}_{\mathbf{Y}}\{\sum_l \mathcal{L}(\mathbf{y}_l, \mathbf{h}_\theta(\mathbf{x}_l(\theta^{t-1}, \mathbf{Y})))\}, \tag{3.3}$$

where minimization in each iteration is tackled by Stochastic Gradient Descent. Learning is summarized in Alg. 4.

Algorithm 4: Learning of active mapping

1. Initialize $t \leftarrow 0$ and obtain dataset $D_0 = \{(\mathbf{x}_l, \mathbf{y}_l)\}_l$ by running the pipeline with the rays being selected randomly, instead of using the planner.
2. Train the mapping network on D_t to obtain $\mathbf{h}_{\theta^t}$ with parameters θ^t.
3. Obtain $D_{t+1} = \{(\mathbf{x}_l(\theta^t, \mathbf{Y}), \mathbf{y}_l)\}_l$ by running Alg. 3 and using $\mathbf{h}_{\theta^t}$ for mapping.
4. Set $t \leftarrow t + 1$ and repeat from line 2 until validation error stops decreasing.

Note, that in order to achieve (i) local optimality of the criterion and (ii) statistical consistency of the learning process (i.e., that the training distribution of sparse measurements $\mathbf{x}_l$ corresponds to the one obtained by planning), one would have to find a fixed point of Equation 3.3. Since there are no guarantees that any fixed point exists, we instead iterate the minimization until validation error is decreasing.

The mapping network consists of 6 convolutional layers with 5×5 kernels followed by linear rectifier units (element-wise $\max\{x, 0\}$) and, in 2 cases, by max pooling layers with 2×2 kernels and stride 2, see Fig. 3.2. In the end, there is an fourfold upsampling layer so that the output has same size as input. The network was implemented in *MatConvNet* [68].

3.1.4 Depth measuring rays planning

Planning at position l searches for a set of rays J, which approximately minimizes the expected logistic loss $\mathcal{L}(\mathbf{Y}, \mathbf{h}_{\theta^t}(\mathbf{x}_{l+L}))$ between ground truth

[1] We introduce a canonical frame by using the local maps instead of the global ones, which helps the mapping network to capture local regularities.

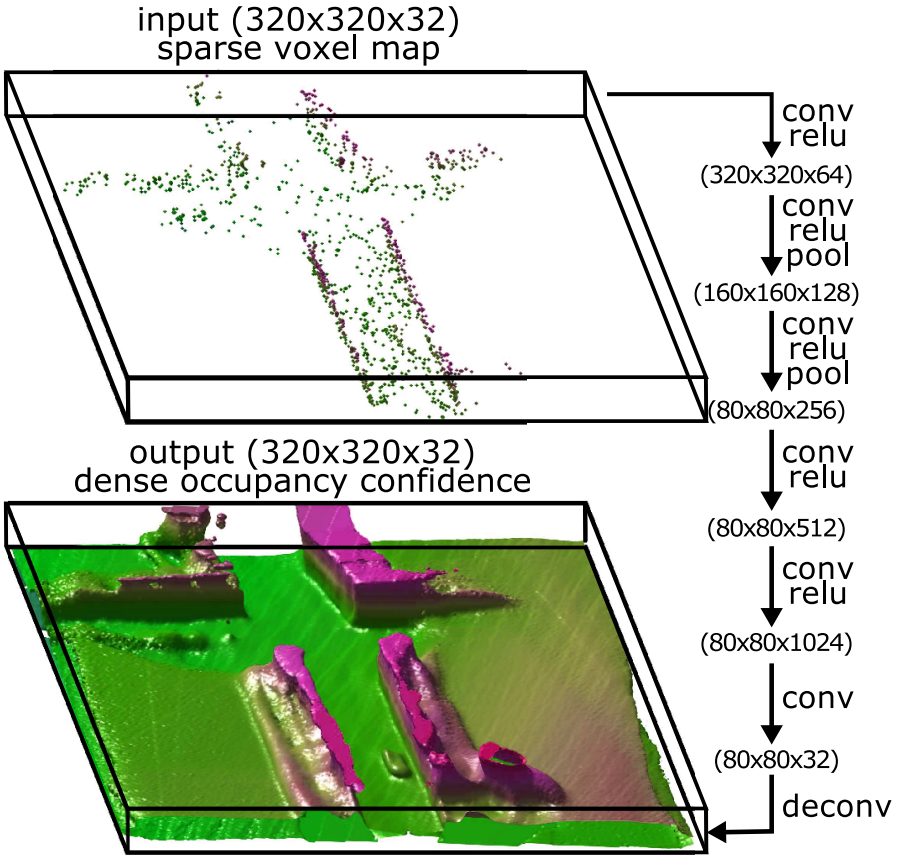


Figure 3.2: Architecture of the mapping network. **Top:** An example input with sparse measurements, showing only the occupied voxels. **Bottom:** The corresponding reconstructed dense occupancy confidence after thresholding. **Right:** Schema of the network architecture, composed from the convolutional layers (denoted *conv*), linear rectifier units (*relu*), pooling layers (*pool*), and upsampling layers (*deconv*).

map $\mathbf{Y}$ and reconstruction obtained from sparse measurements $\mathbf{x}_{l+L}$ at the horizon L. The result of planning is the set of rays J, which will provide measurements for a sparse set of voxels. This set of voxels is referred to as *covered* by J and denoted as $C(J)$. While the mapping network is trained *offline* on the ground-truth maps, the planning have to search the subset of rays *online* without any explicit knowledge of the ground-truth occupancy $\mathbf{Y}$. Since it is not clear how to directly quantify the impact of measuring a subset of voxels on the reconstruction $\mathbf{h}_{\theta^t}(\mathbf{x}_{l+L})$, we introduce simplified reconstruction model $\hat{\mathbf{h}}(J, \hat{\mathbf{Y}})$, which predicts the loss based on currently available map $\hat{\mathbf{Y}}$. This model conservatively assumes that the reconstruction in covered voxels $i \in C(J)$ is correct (i.e. $\mathcal{L}(Y_i, \hat{h}_i(J, \hat{\mathbf{Y}})) = 0$) and reconstruction of not covered voxels $i \notin C(J)$ does not change (i.e. $\mathcal{L}(Y_i, \hat{h}_i(J, \hat{\mathbf{Y}})) = \mathcal{L}(Y_i, \hat{Y}_i)$).

Given this reconstruction model, the expected loss simplifies to:

$$\sum_i \mathcal{L}(Y_i, \hat{h}_i(J, \hat{\mathbf{Y}})) = \sum_{i \notin C(J)} \mathcal{L}(Y_i, \hat{Y}_i) \tag{3.4}$$

Since the ground-truth occupancy of voxels is apriori unknown, neither the voxel-wise loss nor the coverage are known. We model the expected loss in voxel i as

$$\mathcal{L}(Y_i, \hat{Y}_i) \approx \mathbb{E}_{Y_i \sim \mathrm{B}(\sigma(\hat{Y}_i))} \mathcal{L}(Y_i, \hat{Y}_i) = \mathcal{H}(\mathrm{B}(\sigma(\hat{Y}_i))) = \epsilon_i, \tag{3.5}$$

where $\mathcal{H}(\mathrm{B}(p))$ is the entropy of the Bernoulli distribution with parameter p, denoting the probability of outcome 1 from the possible outcomes $\{-1, 1\}$. The vector of concatenated losses ϵ_i is denoted $\boldsymbol{\epsilon}$.

The length of particular rays is also unknown, therefore coverage $C(J)$ of voxels by particular rays cannot be determined uniquely. Consequently, we introduce probability p_{ij} that voxel i will not be covered by ray $j \in J$. This probability is estimated from currently available map $\hat{\mathbf{Y}}$ as the product of (i) the probability that the voxels on ray j which lie between voxel i and the sensor are unoccupied and (ii) the probability that at least one of the following voxels or the voxel i itself are occupied. If ray j does not intersect voxel i, then $p_{ij} = 1$. The vector of probabilities p_{ij} for ray j is denoted $\mathbf{p}_j$. Assuming that rays J are independent measurements, the expected loss is modeled as $\boldsymbol{\epsilon}^\mathsf{T} \prod_{j \in J} \mathbf{p}_j$.

The planning searches for the set $J = J_1 \cup \cdots \cup J_L$ of subsets $J_1 \dots J_L$ of depth-measuring rays for the following L positions, which minimize the expected loss, subject to budget constraints $|J_1| \leq K, \dots |J_L| \leq K$

$$J^* = \arg\min_J \ \boldsymbol{\epsilon}^\mathsf{T} \prod_{j \in J} \mathbf{p}_j, \text{ s.t. } |J_1| \leq K, \dots |J_L| \leq K, \tag{3.6}$$

where $|J_l|$ denotes cardinality of the set J_l.

This is a non-convex combinatorial problem[2] which needs to be solved online repeatedly for millions of potential rays. We tried several convex approximations, however the high-dimensional optimization has been extremely time consuming and the improvement with respect to the significantly faster greedy algorithm was negligible. As a consequence of that, we have decided to use the greedy algorithm. We first introduce its simplified version (Alg. 5) and derive its properties, the significantly faster prioritized greedy algorithm (Alg. 6) is explained later.

We denote the list of available rays at position l as V_l. At the beginning, the list of all available rays is initialized as follows $V = V_1 \cup \cdots \cup V_L$. Alg. 5 successively builds the set of selected rays J. In each iteration the best ray j^* is selected, added into J and removed from V. The position from which the ray j^* is chosen is denoted l^*. If the budget K of l^* is reached, all rays from V_{l^*} are removed from V.

[2] In our experiments, the number of possible combinations is greater then 10^{2000}.

In order to avoid multiplication of all selected rays at each iteration, we introduce the vector $\mathbf{b}$, which keeps voxel loss. Vector $\mathbf{b}$ is initialized as $\mathbf{b} = \boldsymbol{\epsilon}$ and whenever ray j is selected, voxel losses are updated as follows $\mathbf{b} = \mathbf{b} \odot \mathbf{p}_j$, where $\odot$ denotes element-wise multiplication.

Algorithm 5: Greedy planning

Require: Set of available rays V and budget K
$J \leftarrow \emptyset$ ▷ Initialization
$\mathbf{b} \leftarrow \boldsymbol{\epsilon}$
while $\neg(V = \emptyset)$ **do**
  $j^* \leftarrow \arg\min_{j \in V} \mathbf{b}^\mathsf{T} \mathbf{p}_j$ ▷ Add the best ray
  $J \leftarrow J \cup j^*$
  $\mathbf{b} \leftarrow \mathbf{b} \odot \mathbf{p}_j$ ▷ Update voxel costs
  $V \leftarrow V \setminus j^*$ ▷ Remove j^* from V
  if $|J_{l^*}| = K$ **then**
    $V \leftarrow V \setminus V_{l^*}$ ▷ Close position
Return: Set of selected rays J

3.1.5 Prioritized greedy planning

In practice we observed a significant speed up of the greedy planning (Alg. 5) by imposing prioritized search for $\arg\min_j \mathbf{b}^\mathsf{T} \mathbf{p}_j$. Namely, let us denote Δ_j^k the decrease of the expected reconstruction error achieved by selecting ray j in iteration k, $\Delta_j^k = \sum_i (b_i^{k-1} - b_i^k) = \sum_i b_i^{k-1}(1 - p_{ij})$, and show that it is non-increasing. For $p_{ij}, p_{ij'} \in [0, 1]$ and $b_i^{k-1} \geq 0$ it follows that $b_i^{k-1}(1 - p_{ij}) \geq b_i^{k-1} p_{ij'}(1 - p_{ij})$. Summing the inequalities for all voxels i, we get

$$\Delta_j^k = \sum_i b_i^{k-1}(1 - p_{ij}) \geq \sum_i b_i^{k-1} p_{ij'}(1 - p_{ij}) = \Delta_j^{k+1} \tag{3.7}$$

for an arbitrary ray j' selected in iteration k. Note that $\Delta_j^k \geq \Delta_j^{k+a}$ for any $a \geq 1$.

Now, when we search for j maximizing Δ_j^k in decreasing order of $\Delta_j^{k-a_j}$, $a_j \geq 1 \; \forall j$, we can stop once $\Delta_j^k > \Delta_{j'}^{k-a_{j'}}$ for the next ray j' because none of the remaining rays can be better than j. Moreover, we can take advantage of the fact that all the remaining rays including j remained sorted when updating the priority for the next iteration. The proposed planning is detailed in Alg. 6.

The number of re-evaluations of Δ_j in Alg. 6 was approximately 500× smaller than in Alg. 5. Despite the sorting took about a 1/10 of the computation time, the prioritized planning was about 30× faster and took 0.3s on average using a single-threaded implementation.

Algorithm 6: Prioritized greedy planning

Require: Set of rays $V = \{1, \ldots, N\}$ at positions L, budget K, voxel costs $\mathbf{b}$, probability vectors $\mathbf{p}_j \; \forall j \in V$, mapping from ray to position $\lambda\colon V \mapsto L$

$J_l \leftarrow \emptyset \; \forall l \in L$ ▷ No rays selected

$\Delta_j \leftarrow \infty \; \forall j \in V$ ▷ Force recompute

$S \leftarrow (1, \ldots, N)$ ▷ Sequence of ray indices, $S(n)$ denotes the nth element in the sequence, $S(m{:}n)$ the subsequence from the mth to the nth element.

while $S \neq \emptyset$ **do**

  for $n \in (1, \ldots, |S|)$ **do**

    $\Delta_{S(n)} \leftarrow \mathbf{b}^\mathsf{T}(\mathbf{1} - \mathbf{p}_{S(n)})$

    if $n < |S| \wedge \Delta_{S(n)} \geq \Delta_{S(n+1)}$ **then**

      break

  Sort subsequence $S(1:n)$ s.t. $\Delta_{S(n')} \geq \Delta_{S(n'+1)}$

  Merge sorted subsequences $S(1:n-1)$ and $S(n:|S|)$

  $j^* \leftarrow S(1), l^* \leftarrow \lambda(j^*)$

  $J_{l^*} \leftarrow J_{l^*} \cup \{j^*\}$ ▷ Add the best ray

  $\mathbf{b} \leftarrow \mathbf{b} \odot \mathbf{p}_{j^*}$ ▷ Update voxel costs

  if $|J_{l^*}| = K$ **then**

    $S \leftarrow S \setminus \{j : \lambda(j) = l^*\}$ ▷ Close position

  else

    $S \leftarrow S \setminus \{j^*\}$ ▷ Remove j^* from S

Return: Selected rays J_l at every position $l \in L$

3.1.6 Experiments

Dataset

All experiments were conducted on selected sequences from categories *City* and *Residential* from the KITTI dataset [62]. We first brought the point clouds (captured by the Velodyne HDL-64E laser scanner) to a common reference frame using the localization data from the inertial navigation system (OXTS RT 3003 GPS/IMU) and created the ground-truth voxel maps from these. The voxels traced from the sensor origin towards each measured point were updated as empty except for the voxels incident with any of the end points which were updated as occupied for each incident end point. The dynamic objects were mostly removed in the process since the voxels belonging to these objects were also many times updated as empty while moving. All maps used axis-aligned voxels of edge size 0.2 m.

For generating the sparse measurements, we consider an SSL sensor with the field of view of 120° horizontally and 90° vertically discretized in $160 \times 120 = 19200$ directions. At each position, we select $K = 200$ rays and ray-trace in these directions until an occupied voxel is hit or the maximum distance of 48m is reached. Only the rays which end up hitting an occupied voxel produce valid measurements, as is the case with the time-of-flight sensors. Local maps $\mathbf{x}_l$ and $\mathbf{y}_l$ contain volume of 64m × 64m × 6.4m discretized into $320 \times 320 \times 32$ voxels.

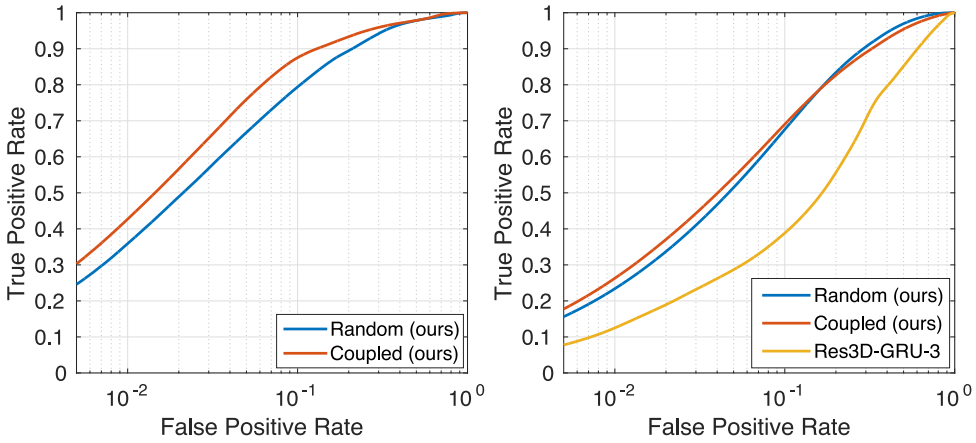


Figure 3.3: ROC curves of occupancy prediction from active 3D mapping on test sets. **Left:** *Random* denotes the global occupancy $\hat{\mathbf{Y}}$ obtained by using $\mathbf{h}_{\theta^0}$ with random sparse measurements, *Coupled* the occupancy obtained by using $\mathbf{h}_{\theta^3}$ with the prioritized greedy planning. The voxels which are more than 1m from what could possibly be measured are excluded, together with the false positives which can be attributed to discretization error (in 1-voxel distance from an occupied voxel). **Right:** *Random* denotes the local occupancy maps $\hat{\mathbf{y}}_l$ obtained by using $\mathbf{h}_{\theta^0}$, *Coupled* the maps obtained by using $\mathbf{h}_{\theta^1}$, and *Res3D-GRU-3* denotes the reconstruction obtained by the network adapted from [18].

Active 3D mapping

In this experiment, we used 17 and 3 sequences from the *Residential* category for training and validation, respectively, and 13 sequences from the *City* category for testing. We evaluate the iterative planning-learning procedure described in Sec. 3.1.3. For learning the mapping networks, we used learning rate $\alpha = 10^{-3}(1/8)^{\lceil i/10 \rceil}$ based on epoch number i, batch size 1, and momentum 0.99. Networks $\mathbf{h}_{\theta^0}, \ldots, \mathbf{h}_{\theta^3}$ were trained for 20 epochs.

The ROC curves shown in Fig. 3.3 (left) are computed using ground-truth maps $\mathbf{Y}$ and predicted global occupancy maps $\hat{\mathbf{Y}}$. The performance of the $\mathbf{h}_{\theta^3}$ network (denoted *Coupled*) significantly outperforms the $\mathbf{h}_{\theta^3}$ network (*Random*), which shows the benefit of the proposed iterative planning-mapping procedure. Examples of reconstructed global occupancy maps are shown in Fig. 3.4. Note that the valid measurements covered around 3% of the input voxels.

Comparison to a recurrent image-based architecture

We provide a comparison with the image-based reconstruction method of Choy *et al.* [18]. Namely, we modify their residual *Res3D-GRU-3* network to use sparse depth maps of size 160×120 instead of RGB images. The sensor pose corresponding to the last received depth map was used for reconstruction. The number of views were fixed to 5, with $K = 200$ randomly selected depth-measuring rays in each image. For this experiment, we used

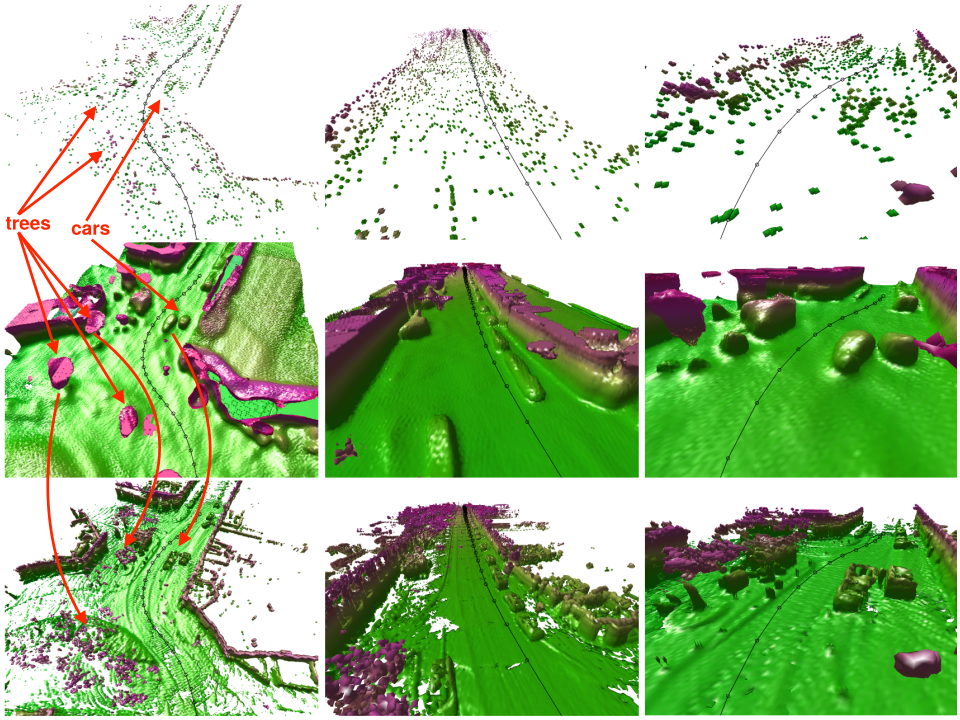


Figure 3.4: Examples of global map reconstruction. **Top:** Sparse measurement maps $\mathbf{X}$. **Middle:** Reconstructed occupancy maps $\hat{\mathbf{Y}}$ in form of isosurface. **Bottom:** Ground-truth maps $\mathbf{Y}$. The black line denotes trajectory of the car.

20 sequences from the *Residential* category—18 for training, 1 for validation and 1 for testing. Since the *Res3D-GRU-3* architecture is not suited for high-dimensional outputs due to its high memory requirements, we limit the batch size to 1 and the size of the maps to $128 \times 128 \times 32$, which corresponds to $16 \times 16 \times 4$ recurrent units. Our mapping network was trained and tested on voxel maps instead of depth images.

The corresponding ROC curves, computed from local maps $\mathbf{y}_l$ and $\hat{\mathbf{y}}_l$, are shown in Fig. 3.3 (right). Both $\mathbf{h}_{\theta 0}$ and $\mathbf{h}_{\theta 1}$ networks outperforms the *Res3D-GRU-3* network. We attribute this result mostly to the fact that our method is implicitly provided the known trajectory, while the *Res3D-GRU-3* network is not. Another reason may be the ray-voxel mapping which is also known implicitly in our case, compared to [18].

3.1.7 Conclusion

We have proposed a computationally tractable approach for the very high-dimensional active perception task. The proposed 3D-reconstruction CNN outperforms a state-of-the-art approach by 20% in recall, and it is shown that when learning is coupled with planning, recall increases by additional 8% on the same false positive rate. The proposed prioritized greedy planning algorithm seems to be a promising direction with respect to on-board reactive

control since it is about 30× faster and requires only 1/500 of ray evaluations compared to a naïve greedy solution.

3.2 Learning for multi-modal active segmentation

An integral part of any search-and-rescue mission is a time-critical search for potential victims in a disaster area. Typical disaster area contains dangerous zones in which human rescuers are not allowed to enter - red zones. Exploration of these red zones is usually assured by a teleoperated robot [69]. In the teleoperated exploration scenario a human operator determines a coarse exploration path. The robot follows the path and collects a multi-modal measurements, such as camera, depth and thermal images. Since these measurements are difficult to be processed reliably by a human operator in a time-critical scenario, the real-time autonomous processing, which allows to guide operator's attention and decrease his cognitive load, is highly desirable functionality [70]. Proposed autonomous processing delivers a concise 3D semantic map which visualizes topological relations and emphasizes potential victims, see Fig. 3.7. Resulting semantic map is critical for further reasoning and decision-making, therefore its quality is essential.

Since sensors have a limited field of view and the exploration time is restricted, the resulting coverage of the disaster environment by the sensor measurements is incomplete, which compromises the accuracy of the resulting semantic map. We consider the setup in which a robotic platform is equipped with an omnidirectional camera, a rotation laser scanner (together denoted

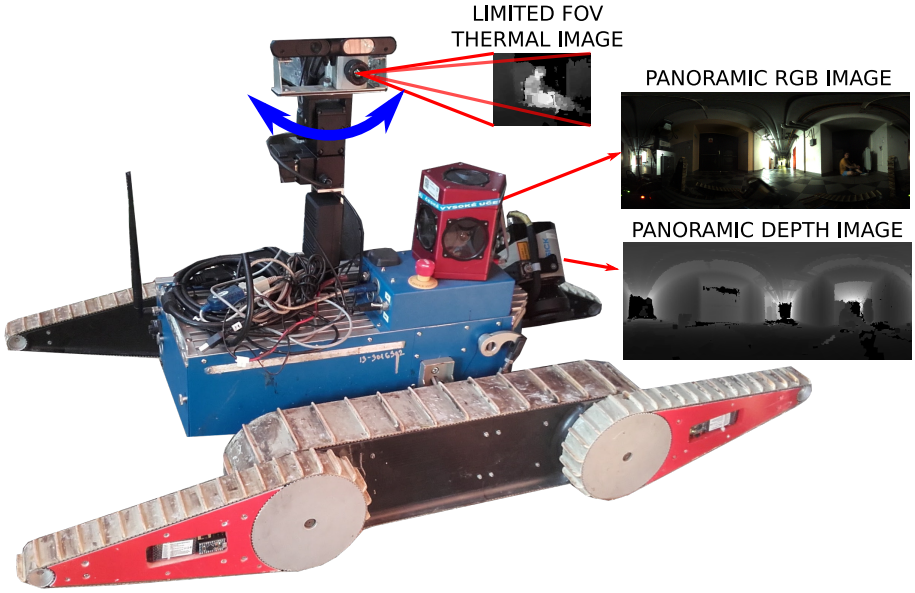


Figure 3.5: Skid-steer search-and-rescue robot with a panoramic RGBD sensor consisting of an omnidirectional camera and a rotation laser scanner; Thermal (T) sensor, with limited narrow field of view mounted on a pan-tilt unit.

as RGBD sensor) and a pan-tilt thermal camera with a small field of view, see Fig. 3.5. Since temperature is an essential cue for detecting humans in search and rescue, a segmentation-friendly control of the pan-tilt unit is needed for compensating the limited sensor coverage and maintain accurate segmentation. We proposed active victim segmentation algorithm which simultaneously (i) segments victims in incomplete RGBDT data and (ii) controls the thermal camera to minimize segmentation error.

One possible solution is to define a cost function reflecting the expected error of the semantic map and plan the thermal measurements to minimize it. However, since the dimensionality of the underlying state-action space is enormous, the real-time (re)planning is prohibitively time-consuming. Consequently, we propose a novel learnable method, which employs deep convolutional neural networks (CNNs) for both tasks: (i) the victim detection (provided by purely segmentation nets), and (ii) the thermal camera control (determined by a Q-value policy network). Proposed learning method minimize common objective, therefore segmentation nets are trained to work well on the temperature measurements provided the policy network, while the policy network is trained to provide temperature measurements which maximize the segmentation accuracy.

CNNs have recently been shown to be a powerful representation for both the classification and the motion control. However, the success of CNNs is usually conditioned either by (i) the existence of a huge number of training examples [71, 72], or (ii) a careful initialization [73, 74]. Both of these conditions are difficult to achieve due to the lack of natural training data and the inherently limited number of trials, which can be performed with the real robot. In particular, we overcome the absence of training data for the segmentation task by employing a manually designed generative model, which provides the semi-synthetic training dataset. The high-dimensionality of the underlying control problem is tackled by self-supervised initialization and coverage-planning-based guiding of the motion control learning.

In particular, we first extend Long's segmentation CNN [73] by depth and both depth and thermal modalities, and retrain it on our own human/background-annotated RGBDT dataset. These segmentation CNNs are further used for self-supervised training of a control sub-network, on data without any annotation, which estimates potential impact of thermal measurements on the classification error. The control sub-network is further extended by sub-sampling and fully connected layers and trained to predict the long-term impact of possible thermal-camera motions on the classification error. To train the control CNN efficiently, we propose a guided Q-learning algorithm, which uses optimal trajectories estimated by a Mixed Integer Linear Programming (MILP) planner to guide the exploration of the Q-learning and consequently avoids poor local optima.

The main contribution of this work lies in (i) showing that in contrast to general reinforcement learning tasks, the active segmentation problem allows for a self-supervised policy initialization, (ii) introducing coverage-planning-based guiding of the motion control learning, (iii) conducting quantitative

evaluation of the proposed method on several real testing scenarios using the real search-and-rescue robot, (iv) providing the active search and rescue dataset which allows further development of algorithms on the real data[3].

3.2.1 Related work

Real search and rescue missions are discussed in [69]. Due to several reasons, notably due to extremely harsh conditions and lack of human to robot trust, the robots were teleoperated in all the successful missions. The victim search were done by human operators. Rescue scenarios have been simulated for many years within annual RoboCup events emphasizing the importance [75]. The DARPA Subterranean Challenge[4], confirms out that scenaros like this are still actual and automatic scene understanding or segmentation in real mission remains largely unsolved challenge.

The problem of active perception from computational perspective was recently surveyed in [76]. Employing purposive sensing for the task of object recognition can be dated back to [77], where the authors divided the task into two subproblems—moving the camera and primary light source to a position called a *standard view* of the unknown object, and two-dimensional recognition problem. With articulated objects in arbitrary pose, as in our scenario, this decomposition is impractical because the standard view may not be reachable from the current position or due to possibly large self-occlusions which may occur even when the standard view is reached.

Many methods have been proposed for selecting the next best view with respect to the object recognition task at hand. [78] addressed the view selection itself as a classification problem and proposed a boosting-based algorithm combining three types of cues, including a similarity measure based on an implicit shape model. [79] proposed a multi-view classifier based on an ensemble of random decision trees where the view selection is inherent in the classification process. The proposed framework was applied to the task of autonomously unfolding clothes by a robot, addressing the problem of best view selection in classification, grasp point and pose estimation of garments. These two approaches, nevertheless, are closely connected to the classification task with a single object being presented and cannot be adapted to segmentation of multiple articulated objects in arbitrary poses. [80] decompose the multi-view object classification task into a set of independent classification tasks, each dealing with a single image pair. They then use this pair-wise decomposition in trajectory optimization—they seek a path of a given length in the corresponding undirected graph of neighboring views which maximizes estimated cross-entropy scores for unobserved views. After each new view observed, the scores are updated and the trajectory is re-planned. Their approach, nevertheless, becomes impractical for longer trajectories, because the number of pairs increases quadratically with the number of views.

Shubina et al. [81] proposes a strategy for finding a target in a space with

[3] `http://ptak.felk.cvut.cz/tradr/data/human_seg/`

[4] `https://www.subtchallenge.com`

unknown inner structure but with known dimensions. The search space is tessellated into a 3D grid of non-overlapping cubic elements. Also the space of possible robot position is tessellated as 2D 1×1m grid. An operation on the search space 3D grid consists of taking an image according to camera (robot) position and orientation and analyzing it to find out where the target is present. The cost function for the operation includes moving the sensor, acquiring the data, and running the recognition algorithm, and updating the grid. A sensed sphere [82] is used to represent the surrounding of the camera. The global sensor move planning is NP-complete [82] hence, a greedy one-step where to look and move next planning strategy is proposed.

Andreopoulos et al. [67] share many concepts with Shubina et al. [81], notably the concept of 3D search space grid, here called target confidence map. Their work adds an obstacle map and a multi-view visual detector. The core contribution is a probabilistic update of both the target confidence and the obstacle map. The planning is greedy—next best view and position (of a humanoid robot) is selected.

The active visual segmentation approach proposed by [64] understands the activity very much differently from us. The authors propose an automatic segmentation method *given* a fixation on an object or a scene part. An initial fixation is further refined by choosing certain points on the skeleton of the segmented object.

Semantic segmentation have been traditionally formulated as energy minimization in graphical models, as in [83, 84], employing approximations both in learning and inference to maintain tractability. Recently, deep convolutional neural networks (CNN) achieved competitive results. Long et al. [73] adapt several classification models for the semantic segmentation task, introducing skip connections to maintain spatial fidelity of the output. We use their *FCN-32s* model as a basis of our multi-modal segmentation models.

A multi-modal human body segmentation was recently proposed by [85]. Their method, nevertheless, relies on background subtraction using a learned Gaussian mixture model and the camera being static which is not applicable in our settings. They also present a new `RGBDT` dataset with annotated human bodies which is similar to the dataset we published as for the represented modalities and object of interest. The dataset was used to evaluate a human body co-segmentation approach proposed in [86], using uncalibrated but static sensors within indoor scenes. Our dataset, nevertheless, exhibits higher variability of background scenes and human poses, motivated by search-and-rescue scenarios.

Paper from Mnih et al. [72] shows method how to control agent using CNN. They tested it on Atari games where only pixels and game score was given as an input and action of the agent was an output. They propose slightly modified Q-learning to learn weights of the CNN. In this work, we show that training DQN policies can benefit from being provided with guiding samples obtained from an optimal planner.

Levine et al. [74] also shows how to control a humanoid robot to solve contact rich manipulation tasks such as screwing a cap onto a bottle. The

policy is represented by CNN that maps raw images observations directly to torques at the robot's motors. In contrast to [72], [74] we solve active segmentation task where segmentation error is tightly coupled with reward.

Recently [63] proposed to use the reinforcement learning for active object and scene categorization, in which a learned CNN policy successively selects viewpoints of `RGB` camera to minimize categorization error. In contrast to this task, we solve the task of active 3D segmentation from incomplete `RGBDT` data captured online in a structured 3D environment. Hence, the learned policy has to infer both (i) the expected segmentation errors and (ii) the occlusions preventing future acquisition of thermal data. To tackle such complex task we propose self-supervised initialization and provide optimal trajectories to guide the reinforcement learning.

3.2.2 Theory

The sensory suite of our mobile robot consists of (i) the Point Grey Ladybug 3 panoramic camera providing `RGB` images, (ii) the SICK LMS-151 laser scanner on a rotating mount providing depth measurements `D` and (iii) the thermal camera Micro-Epsilon thermoIMAGER TIM 160 with a small field of view mounted on a pan-tilt unit and providing thermal measurements `T`.[5] The robot follows a known short-horizon path consisting of several discrete positions into an unknown environment. As the robot explores the environment, it simultaneously builds a 3D voxel map of occupancy and localizes itself within the map. In addition to that, temperature of some voxels can be measured by the thermal camera. Our goal is (i) human/background segmentation of the 3D voxel map from captured `RGBDT` data and (ii) simultaneous control of the thermal camera in order to capture such thermal measurements which facilitate the segmentation the most.

Measurements and voxel map

The result of the proposed pipeline (see Fig. 3.6) is 3D voxel map, which accumulates occupancy, temperature and segmentation confidence. At each position, the voxel map is reprojected into the `RGB` camera coordinate frame to create depth and thermal image, respectively, of the same resolution as the `RGB` images. Concatenation of the `RGB` image with depth image `D` is denoted by $\mathbf{x}$, the thermal image is denoted by $\mathbf{z}$. Especially, we introduce state X, which consist of $\mathbf{x}, \mathbf{z}$, current viewpoint of the thermal camera and the position of the robot on the exploration path.

Human segmentation

The probability of human presence/absence in particular pixels is estimated by two segmentation networks. The first segmentation network $S_\theta(\mathbf{x})$ provides estimates without using any temperature measurements, the second

[5] The extrinsic camera calibration w.r.t. the laser was obtained by [43].

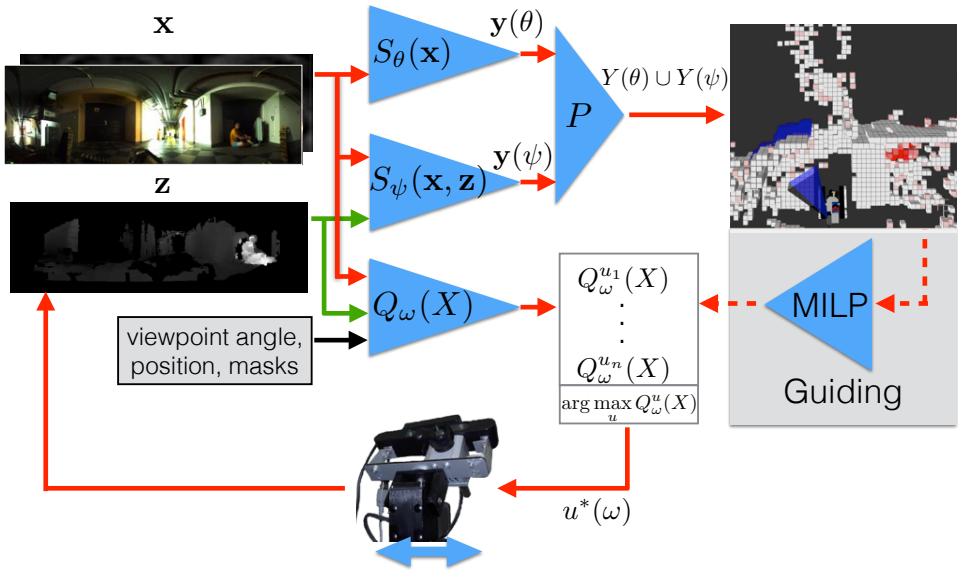


Figure 3.6: Learning outline. Human presence/absence in particular pixels is determined by two segmentation networks $S_\theta(\mathbf{x})$ and $S_\psi(\mathbf{x},\mathbf{z})$. Motion of the thermal camera is controlled by state-action value function network $Q_\omega(X)$. While learning of the segmentation networks is tackled by SGD (Stochastic Gradient Descent), learning of the Q-network is guided by the optimal Q-values provided by the MILP (Mixed Integer Linear Program)-based planner.

segmentation network $S_\psi(\mathbf{x},\mathbf{z})$ use the available temperature measurements. Network parameters are denoted by θ and ψ, respectively. Outputs of these networks, $\hat{\mathbf{y}}(\theta)$ and $\hat{\mathbf{y}}(\psi)$, are projected by mapping P onto the existing 3D voxel map to update the respective probability estimates in the corresponding voxels, denoted by $\hat{Y}(\theta)$ and $\hat{Y}(\psi)$.

Motion of the thermal camera is determined by state-action value function network $Q_\omega(X)$ with parameters ω, which assigns Q-values $Q_\omega^{u_1}(X),\ldots,Q_\omega^{u_n}(X)$ to n discrete control actions. At each state X, the best available action $u^* = \arg\max_u Q_\omega^u(X)$ is chosen to control the motion of the thermal camera. The proposed measuring-classification-control loop is summarized in Alg. 7.

At each position on the exploration path, the thermal camera captures a single thermal image from a defined viewpoint. Viewpoint at position k is denoted i_k (viewpoints outlined as light blue cones in Fig. 3.7-right). We assume that the motion dynamics of the thermal camera is constrained and that viewpoint i_k is given as $i_k = f(i_{k-1}, u_k)$, where f is the motion model and u_k is a control action at position k. Given a fixed training scenario, resulting voxel segmentation map $\hat{Y}(\theta,\psi,i_1\ldots i_K)$ estimated at position K is uniquely determined by the segmentation parameters θ, ψ and by the

Algorithm 7: The active segmentation algorithm.

1. Capture `RGB`, `D`, and `T` data and update the corresponding 3D voxel maps.
2. Construct $\mathbf{x}$ and $\mathbf{z}$ from the `RGB` camera image and the current voxel maps of occupancy and temperature.
3. Estimate local pixel-wise human probability $\hat{\mathbf{y}}(\psi) = S_\psi(\mathbf{x}, \mathbf{z}), \hat{\mathbf{y}}(\theta) = S_\theta(\mathbf{x})$.
4. Update the corresponding voxel maps $\hat{Y}(\psi)$ and $\hat{Y}(\theta)$ using mapping P.
5. Estimate new control $u^* = \arg\max_u Q_\omega(X, u)$.
6. Simultaneously move the robot towards the next position on the exploration path and the thermal camera by control signal u^* towards the viewpoint to be captured at the next position.
7. Repeat from the beginning.

captured viewpoints $i_1 \dots i_K$.

Learning is defined as a search for parameters θ, ψ, and ω which minimize the cross-entropy loss $\mathcal{H}(Y, \hat{Y}(\theta, \psi, i_1 \dots i_K))$ between estimated global voxel map $\hat{Y}(\theta, \psi, i_1 \dots i_K)$ at a final position K and ground-truth voxel map Y subject to the motion constrains of the thermal camera,

$$\begin{aligned} &\arg\min_{\psi,\theta,\omega} \sum_v \mathcal{H}\left(Y_v, \hat{Y}_v(\theta, \psi, i_1 \dots i_K)\right) \\ &\text{s.t. } i_k = f\left(i_{k-1}, u_k(\omega)\right) \ \forall_{k \in \{1,\dots,K\}}, \end{aligned} \tag{3.8}$$

where Y_v, $\hat{Y}_v$ denotes elements (voxels) of voxel maps Y, $\hat{Y}$, respectively, and initial viewpoint i_0 is a constant assumed to be known in advance. This optimization problem is approximately solved as successive minimization over θ, ψ, and ω.

Optimization over ψ and θ is formulated as SGD minimizing of the cross entropy of pixel-wise updates $\hat{y}_i(\theta)$, $\hat{y}_i(\psi)$ with respect to pixel-wise ground-truth y_i.

$$\arg\min_\theta \sum_i \mathcal{H}\left(y_i, \hat{y}_i(\theta)\right), \tag{3.9}$$

$$\arg\min_\psi \sum_i \mathcal{H}\left(y_i, \hat{y}_i(\psi)\right). \tag{3.10}$$

Optimization over ω

Let us denote V the set of all voxels and $V(i_1, \dots, i_K)$ its subset containing the voxels visible with the thermal camera in any of the K views $i_1, \dots, i_K$ selected along the path (see Fig. 3.7, on the right). Resulting voxel segmentation $\hat{Y}(\theta, \psi, i_1 \dots i_K)$ is composed from $\hat{Y}(\theta)$ and $\hat{Y}(\psi)$ as follows:

$$\hat{Y}_v(\theta, \psi, i_1 \dots i_K) = \begin{cases} \hat{Y}_v(\theta), & v \in V(i_1, \dots, i_K) \\ \hat{Y}_v(\psi), & v \notin V(i_1, \dots, i_K) \end{cases} \tag{3.11}$$

Consequently, the optimization over ω is simplified as follows:

$$\begin{aligned}
&\arg\min_{\omega} \sum_{v\in V(i_1,\dots,i_K)} \mathcal{H}\left(Y_v, \hat{Y}_v(\psi)\right) + \sum_{v\notin V(i_1,\dots,i_K)} \mathcal{H}\left(Y_v, \hat{Y}_v(\theta)\right) \\
&\qquad \text{s.t. } i_k = f(i_{k-1}, u_k(\omega)) \; \forall_{k\in\{1,\dots,K\}} \\
&= \arg\min_{\omega} \sum_{v\in V(i_1,\dots,i_K)} \mathcal{H}\left(Y_v, \hat{Y}_v(\psi)\right) - \sum_{v\in V(i_1,\dots,i_K)} \mathcal{H}\left(Y_v, \hat{Y}_v(\theta)\right) + \sum_{v\in V} \mathcal{H}\left(Y_v, \hat{Y}_v(\theta)\right) \\
&\qquad \text{s.t. } i_k = f(i_{k-1}, u_k(\omega)) \; \forall_{k\in\{1,\dots,K\}} \\
&= \arg\max_{\omega} \sum_{v\in V(i_1,\dots,i_K)} \underbrace{\mathcal{H}\left(Y_v, \hat{Y}_v(\theta)\right) - \mathcal{H}\left(Y_v, \hat{Y}_v(\psi)\right)}_{\Delta\mathcal{H}_v(\theta,\psi)} \qquad (3.12) \\
&\qquad \text{s.t. } i_k = f(i_{k-1}, u_k(\omega)) \; \forall_{k\in\{1,\dots,K\}},
\end{aligned}$$

where difference

$$\mathcal{H}\left(Y_v, \hat{Y}_v(\theta)\right) - \mathcal{H}\left(Y_v, \hat{Y}_v(\psi)\right) = \Delta\mathcal{H}_v(\theta,\psi) \qquad (3.13)$$

denotes the reduction of the cross-entropy loss in voxel v when the temperature becomes known at this particular voxel—we call this quantity *gain*. The motion and budget constraints bind the control $u_1(\omega), \dots, u_K(\omega)$ over the whole horizon K and the optimization cannot be decoupled. Given fixed segmentation parameters, we learn state-action value function $Q_\omega(X)$, which estimates the expected gain. The guided Q-learning algorithm for optimization of ω is detailed in Section 3.2.3.

Figure 3.7: Left: Panoramic `RGB` image with segmented humans outlined by green (`RGBD` data segmentation) and magenta (`RGBDT` data segmentation) contours. The reprojected thermal measurements collected up to the current time are emphasized by blue overlay. **Right:** Reconstructed and segmented voxel map with accumulated thermal measurements displayed in blue color. Light red denotes the voxels marked as corresponding to human based on `RGBD` data only, dark red denotes the voxels marked as human based on the data with additional temperature measurements. Robot path with positions is denoted by black arrow with dots and selected thermal viewpoints are outlined by blue cones. The thermal camera is controlled to maximize the long-term sum of $\Delta\mathcal{H}$.

3.2.3 Learning of the control network

If (i) the visibility of all voxels in all viewpoints along the robot path is available in advance, (ii) the gain is known for all voxels, and (iii) the control signals are discrete, then the optimal control corresponds to the weighted maximum coverage problem with limited budget and motion constraints. Such formulation is an instance of the following MILP:

$$\begin{aligned} \arg\max_{\mathbf{u},\mathbf{v}} \quad & \mathbf{v}^{\mathsf{T}} \Delta\mathcal{H}(\psi, \theta) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{u} \geq \mathbf{v} \\ & \mathbf{B}\mathbf{u} = \mathbf{1} \\ & \mathbf{C}\mathbf{u} \leq \mathbf{1} \\ & \mathbf{v} \in [0,1]^{V} \\ & \mathbf{u} \in \{0,1\}^{KN}, \end{aligned} \tag{3.14}$$

where $\mathbf{A}$ is a sparse binary matrix which captures visibility of the voxels in the available viewpoints along the planning horizon, $\mathbf{B}$ is a sparse binary matrix determined by the budget constraints (single viewpoint per position), $\mathbf{C}$ is a sparse binary matrix which captures the motion constraints, V is the number of voxels in the map, K is the planning horizon (i.e., the number of positions along the path), N is the number of the available actions (viewpoints). The result of this optimization are two vectors $\mathbf{u}$ and $\mathbf{v}$; vector $\mathbf{u}$ specifies control signal along the path and vector $\mathbf{v}$ is an auxiliary variable which denotes visibility of particular voxels in the thermal camera.

Since an unknown environment is typically explored, neither the map nor the gain $\Delta\mathcal{H}_v$ are known in a testing scenario, which makes direct online optimization of problem (3.14) impossible. On the other hand, complete voxel maps with corresponding voxel gains are available for the annotated training sequences. Since a direct optimization of ω would require recurrent estimation of the gain with respect to the considered horizon K, which is both computationally demanding and prone to get stuck in a poor local minimum, we instead use MILP to directly optimize the control $\mathbf{u}$ on the training sequences. Optimal Q-values eventually guide the learning of parameters ω, see Sec. 3.2.3.2 for details.

Since the raw sensory measurements are high-dimensional, learning of deep Q-value network $Q_\omega(X, u)$ from randomly initialized weights would require a huge amount of training samples. To avoid such a demanding training procedure, we suggest to divide the $Q_\omega(X)$ network into two sub-networks:
(i) $g_{\omega_1}(\mathbf{x})$ network which estimates $\Delta\mathcal{H}$ from $\mathbf{x}$ and
(ii) $q_{\omega_2}(\Delta\mathcal{H}, X)$ network which predicts the Q-values from the estimated gain $\Delta\mathcal{H}$ and state X.
These networks are first trained independently, then concatenated as $Q_\omega(X) = q_{\omega_2}(g_{\omega_1}(\mathbf{x}), X)$ and fine-tuned as the one network (see Fig. 3.8). Learning of the Q-value network is summarized in the three following steps.

1. **Train gain predicting sub-network** g_{ω_1} from supervised and self-supervised $\Delta\mathcal{H}$ annotations. In the supervised setting, human/background

annotations are available. In such case, $\Delta\mathcal{H}$ annotations are just the difference of cross entropies of segmentation networks (Eq. 3.13). In the self-supervised setting, we exploit unlabeled `RGBDT` data. In such a case, $\Delta\mathcal{H}$ annotations are approximated as Kullback-Leibler divergence of the outputs of segmentation networks, see Sec. 3.2.3.1 for details. The g_{ω_1} sub-network predicts the expected reduction of the cross-entropy loss as a result of measuring temperature at particular pixels.

2. **Train Q-value sub-network** $q_{\omega_2}(\Delta\mathcal{H}, X)$ by the proposed guided Q-learning algorithm. The guided Q-learning first use the MILP planner to estimate optimal trajectories which maximize $\Delta\mathcal{H}$-weighted coverage of voxels from the explored environment. These trajectories are used to normalize the Q-values and to guide the exploration. Learned policy approximates these optimal trajectories and consequently minimize the segmentation error, see Section 3.2.3.2 for details.

3. **Connect previously trained sub-networks** into the final Q-value network $Q_\omega(X, u) = q_{\omega_2}(g_{\omega_1}(\mathbf{x}), X)$ and fine-tune its parameters ω. Note, that the fine-tuned Q_ω network does not predict the gain $\Delta\mathcal{H}$ anymore.

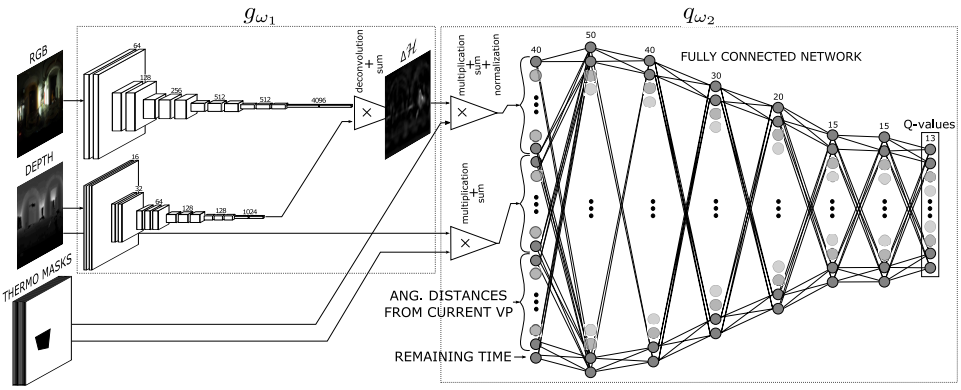


Figure 3.8: Structure of Q_ω network: The policy $Q_\omega(X) = q_{\omega_2}(g_{\omega_1}(\mathbf{x}), X)$ is composed from two subnetworks: (i) gain predicting sub-network $g_{\omega_1}(\mathbf{x})$, and (ii) Q-value sub-network $q_{\omega_2}(\Delta\mathcal{H}, X)$, with an interconnecting subsampling layer in the middle.

3.2.3.1 Self-Supervised Policy Initialization

Training sequence consists of $\mathbf{x}$ and $\mathbf{z}$ images. For some of these images human/background labels $\mathbf{y}$ are available, some images are unlabeled. When annotations are available, supervised learning of the gain predicting network g_{ω_1} is straightforward. We collect training pairs $[\mathbf{x},\ \mathcal{H}(\mathbf{y}, \hat{\mathbf{y}}(\theta)) - \mathcal{H}(\mathbf{y}, \hat{\mathbf{y}}(\psi))]_k$ for fixed parameters θ and ψ, and learn a regression network minimizing the square loss. In addition to this, we also suggest a self-supervised learning

setup, in which arbitrary unlabeled data are used. In this setting, we approximate the gain using outputs of segmentation networks $\hat{\mathbf{y}}(\theta)$ and $\hat{\mathbf{y}}(\psi)$ as the expected difference of the cross entropy losses under the best current estimate $\hat{\mathbf{y}}(\psi)$ of truth labels as follows

$$\begin{aligned}
&\mathbb{E}_{y_i \sim B(\hat{y}_i(\psi))} \left\{ \Delta \mathcal{H} \left(\psi, \theta \right) \right\} \\
&= \mathbb{E}_{y_i \sim B(\hat{y}_i(\psi))} \left\{ \mathcal{H} \left(y_i, \hat{y}_i(\theta) \right) - \mathcal{H} \left(y_i, \hat{y}_i(\psi) \right) \right\} \\
&= \mathcal{H} \left(\hat{y}_i(\psi), \hat{y}_i(\theta) \right) - \mathcal{H} \left(\hat{y}_i(\psi) \right) \\
&= \mathcal{H} \left(\hat{y}_i(\psi) \right) + D_{\mathrm{KL}} \left(\hat{y}_i(\psi) \| \hat{y}_i(\theta) \right) - \mathcal{H} \left(\hat{y}_i(\psi) \right) \\
&= D_{\mathrm{KL}} \left(\hat{y}_i(\psi) \| \hat{y}_i(\theta) \right)
\end{aligned} \tag{3.15}$$

where $B(p)$ is the Bernoulli distribution with parameter p, $\mathcal{H}(p)$ is the entropy of such a Bernoulli distribution, and $\mathcal{H}(p, q)$ and $D_{\mathrm{KL}}(p \| q)$ denote the cross entropy and Kullback-Leibler divergence, respectively, of the respective distributions. Predicted gain for a testing image is shown in Fig. 3.9.

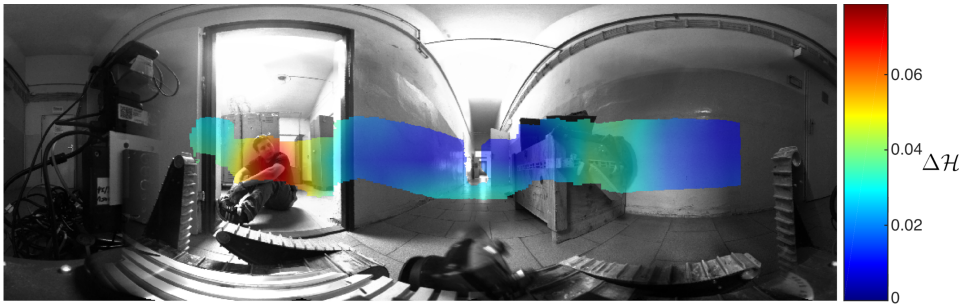


Figure 3.9: Output of gain predicting network $g_{\omega_1}(\mathbf{x})$ estimates the expected per-pixel gain derived in Eq. (3.15). Colorbar encodes values of the visualized gain.

3.2.3.2 Coverage-planning-based Guided Q-Learning

Given fixed segmentation networks, we formalize control of the thermal camera as MDP with the following states, actions, rewards and transition probability.

State X_k is concatenation: $X_k = (\mathbf{x}_k, \mathbf{z}_k, \mathbf{m}_k, \angle(I, i_k), K - k)$, where $\angle(I, i_k)$ denotes the angular distance of all viewpoints I from current viewpoint i_k, $K - k$ is the remaining number of the positions, $\mathbf{m}_k$ denotes the thermal masks determining coverage of pixels by temperature for the allowed viewpoints $i \in I$.

Action u_k corresponds to the motion of the thermal camera from the current viewpoint to the one of discrete viewpoints in the close neighbourhood.

Reward r_k for performing the action is given state is equaled to the gain of newly covered voxels.

Transition probability between states is synthesized from reconstructed training maps with full thermal measurements, which has been captured offline.

We propose guided Q-learning algorithm, which is used for training of (i) the second sub-network q_{ω_2}, as well as (ii) the whole Q-value network Q_ω during fine-tuning. Proposed algorithm Alg. 8 successively collects training transitions from available maps and learns to predict Q-values, which corresponds to the expected gain of covered voxels, when action u is applied in state X and then controlling optimally.

The guided Q-learning first estimates gain for all voxels. The optimal control u^* of the thermal camera and the optimal gain coverage q^* is determined by solving the corresponding MILP instance from the current state. Then it evaluates the sum of gains q' achievable for all possible controls u' by successively applying each control u' and solving the corresponding MILP instance from the following state X'. All these transitions (X_k, u', Q) are stored in the dataset $\mathcal{D}$. We have considered (and experimentally evaluated, see Fig. 3.11) three different types of Q-values:

1. *raw* sum of covered gain-values: $Q = q'$,
2. *absolute loss* in the sum of covered gain values: $Q = q' - q^*$,
3. *relative loss* in the sum of covered gain values: $Q = q'/q^*$.

Eventually, either the optimal control u^* or Q-value-driven control $\arg\max_u Q^u_\omega(X_k)$ is applied and the process continues from the following state X_{k+1}. When a sufficient number of transitions is collected, SGD is performed on weights ω of the regression network Q_ω, until the validation error stops decreasing.

In contrast to the standard Q-learning, the guided Q-value network is not forced to predict the absolute sum of $\Delta\mathcal{H}$ which is often loosely connected with features observed in the current state. Guided Q-learning predicts rather the expected impact on the optimality. Another advantage stems from guiding the exploration of the state-action space close to the optimal trajectories. In the experiments, guiding probability p linearly decreases from 1 towards 0.

Algorithm 8: The guided Q-learning algorithm.

Input: Initial viewpoint i_0
for $k \in \{1, \ldots, K\}$ **do**
  $(q^*, u^*) \leftarrow \texttt{MILP}(X_k)$ # Optimal control from the current state X_k
  for $u' \in \{1, \ldots, N\}$ **do**
    $(X', R') \leftarrow \texttt{act}(X_k, u')$ # Apply action u and get reward R' and the following state X'
    $q' \leftarrow R' + \texttt{MILP}(X')$ # Estimate Q-value for doing action u in state X_k
    $\mathcal{D} \leftarrow \mathcal{D} \cup \left(X_k, u', \frac{q'}{q^*}\right)$
  $u_k \leftarrow \begin{cases} u^* & \text{with prob. } p \\ \arg\max_u Q^u_\omega(X_k) & \text{with prob. } 1-p \end{cases}$
  $X_{k+1} \leftarrow \texttt{act}(X_k, u_k)$ $\omega \leftarrow \texttt{SGD}(Q_\omega, \mathcal{D})$.

3.2.4 Learning of the Multimodal CNN Models

Convolutional neural networks are expressive models which allow efficient element-wise prediction for inputs of variable size. They are composed of multiple processing layers forming a directed acyclic graph. The *bottom* layer has the source data as its input, the *top* layer yields the target prediction or a task-specific scalar loss for training.

We minimize the loss using stochastic gradient descent (SGD) with Nesterov's accelerated gradient (NAG) [87, 88] which yields the following weight updates:

$$\mathbf{v}_{t+1} = \mu\mathbf{v}_t - \alpha\nabla\ell(\mathbf{w}_t + \mu\mathbf{v}_t), \tag{3.16}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{v}_{t+1}, \tag{3.17}$$

with $\mathbf{w}_t$ being the model parameters at iteration t and $\mathbf{v}_t$ their preceding update, $\nabla\ell(\mathbf{w})$ being the gradient of loss function ℓ at $\mathbf{w}$, $\alpha > 0$ being the learning rate, and $\mu \in [0, 1)$ the momentum coefficient. The segmentation models use the multinomial logistic loss for training, the regression model uses the Euclidean loss.

All the models having `RGB` as input reuse the 16-layer VGG net [89] as adapted and fine-tuned by [73], namely the *FCN-32s* variant. Since annotated depth and thermal data are much scarcer, and no suitable pretrained models are available for these modalities, we employ smaller models, with similar structure but having four times less output channels in each convolutional layer to prevent overfitting.

The multimodal models are composed by summing up the outputs of the corresponding deconvolution layers, directly before the final softmax layer.

First, we train the segmentation networks using extra modalities—one using depth, the other using depth and the thermal modality. These are then combined with the pre-trained `RGB` segmentation network [73] and fine-tuned to provide the S_θ and S_ψ networks used in the experimental evaluation in Sec. 3.2.6. After fine-tuning, FCN-32s model achieved the average precision of 0.56 on test images, compared to 0.61 achieved by S_θ. Outputs $\hat{\mathbf{y}}(\theta)$ and $\hat{\mathbf{y}}(\psi)$ are used to train gain-predicting network $\Delta\mathcal{H}_{\omega_1}$, once with ground-truth labels $\mathbf{y}$ to predict $\Delta\mathcal{H}(\theta, \psi)$ directly and once with not annotated data to predict its estimate in form of the Kullback-Leibler divergence from Eq. (3.15). Finally, the gain-predicting network is merged with the control sub-network q_{ω_2} and fine-tuned on guiding trajectories.

For learning parameters of the models, we use training subsets from the two datasets described below, where we replaced the missing measurements in case of the depth and thermal modalities by their nearest valid neighbors. The validation subset of the panoramic dataset were used for early stopping and to select models for test. The reported results in Sec. 3.2.6 are obtained on the test sequences from the panoramic dataset.

We performed 10^5 parameter updates with momentum coefficient $\mu = 0.99$, linearly decaying learning rate from $\alpha = 10^{-4}$ to zero, and a single example per batch. An additional L_2 regularization on weights was used

with coefficient $\lambda = 5 \times 10^{-4}$. The parameters of the models learned from scratch were initialization using the procedure from [90]. The parameters of the segmentation networks S_θ and S_ψ, and the gain-predicting network $\Delta\mathcal{H}_{\omega_1}$ were selected to minimize the loss on the validation set.[6] The CNN-based models were implemented in the *Caffe* framework [91].

3.2.5 Datasets

Semi-Synthetic Human Body Dataset

In order to obtain a large number of images with accurate ground-truth segmentation for training and evaluation we chose to create a semi-synthetic dataset[7] in the following way. First, positive examples with humans in various poses were captured in the lab, in front of the green screen to simplify their annotation. Second, background images were captured in a real-life environment, both outdoor and indoor, without the need to constraint the scene conditions much. Finally, semi-synthetic images were composed by placing annotated humans onto the background images, using the depth information to avoid implausible configurations and to impose realistic occlusions.

For a pair of images, object configurations (i.e., rotation, translation, and scale) were sampled from a uniform distribution until a plausible configuration was found, as measured by an ad-hoc criterion which rewards contact at boundary pixels and penalizes object pixels behind the background. The process is illustrated in Fig. 3.10(a)–(d), showing the source images and the resulting composition, along with several examples of synthetic images. Small occlusions are often generated along the bottom boundary of the object, as if it were partially submerged in mud or fine rubble; in some cases, there are occlusions generated from vertical structures which are part of the background, such as poles, staircases etc. Employing semantic scene analysis techniques would be needed to generate major occlusions while maintaining plausibility of the result.

We used Asus Xtion PRO LIVE to capture the `RGBD` data and IMAGER TIM 160 to capture the thermal data `T`. The source images were split into training, validation, and test sets prior to composition. The number of images in every group is summarized in Table 3.1.

Data set	Training	Validation	Test
Human	1617	539	539
Background	369	123	122
Composed	4022	1381	1294

Table 3.1: Number of images in the semi-synthetic segmentation dataset.

[6] Namely the parameters θ from iteration 8×10^3, ψ from 14×10^3, ω_1 for true $\Delta\mathcal{H}$ prediction from iteration 64×10^3, and ω_1 for D_{KL} prediction from iteration 90×10^3 were selected.

[7] `http://ptak.felk.cvut.cz/tradr/data/human_seg/`

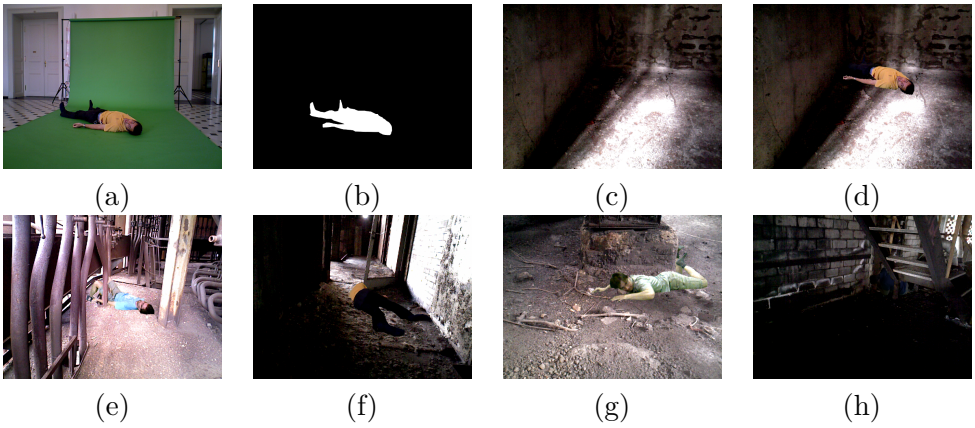


Figure 3.10: Semi-synthetic human body dataset. (a) An image of human body image of a human with (b) the ground-truth segmentation; (c) a background image; (d) a semi-synthetic image composed from the source images. (e)–(h) examples of semi-synthetic images.

Panoramic Human Body Dataset

The panoramic human body dataset[8] was captured indoors using the mobile search-and-rescue platform depicted in Fig. 3.5. During data capture, the robot localized itself using the ICP-based SLAM method from [92, 93], fusing IMU measurements and odometry during dead reckoning. We recorded 24 sequences in total (see Table 3.2 for a summary) with the robot following a mostly straight path[9] during which it was stopping regularly to capture data, including the thermal images for 13 discretized camera views. The dataset allows generating instances of the simultaneous exploration and segmentation task outlined above, i.e., it allows generating sensor data very similar to what would be observed during corresponding online experiments by employing a given thermal camera control policy. The panoramic `RGB` images from the Ladybug 3 camera are 1024×512 pixels in size, the depth and thermal images are rendered in the same resolution from captured data and corresponding voxel maps—see Fig. 3.12 for an example.

Data set	Training	Validation	Test
Human / Background	225 (15)	60 (4)	60 (4)

Table 3.2: Number of images (sequences) in the panoramic segmentation dataset from the search-and-rescue platform.

[8] `http://ptak.felk.cvut.cz/tradr/data/active_seg`

[9] During our joint exercises with firefighters, we observed that robot operators often controlled the robot to follow a straight path towards a checkpoint which lies in the currently observable free space.

3.2.6 Experiments

The experiments are divided into evaluation on synthetic data, which mainly shows the influence of various hardware and learning setups, and real data, which compares the behavior of the learned policy with the greedy algorithm on the search-and-rescue platform. We provide a comparison of our `RGBDT→Q` control policy with a reactive control similar to [81], here denoted by `greedy`, which at each position choses the viewpoint maximizing gain of the voxels.

Synthetic experiments

This section provides the comparison of the proposed guided Q-learning method (referred as `GQ-policy`) in terms of the total $\Delta\mathcal{H}$ of covered voxels. We provide the comparison on 64 randomly generated maze-like maps for the following methods:

- `greedy` reactive control similar to [81], which at each position choses the viewpoint maximizing $\Delta\mathcal{H}$ of voxels
- `Q-policy` reactive control learned by Q-learning similar to [72].
- `optimal` control estimated as a solution of MILP by the CPLEX solver. It creates a theoretical upper bound for the case in which the map, gain and visibility of all voxels along the whole robot's path is known in advance. This method is mainly used to normalize the results and make maps with significantly different sum of gains comparable.
- `optimal-incomplete` control estimated as repeated optimization of MILP by the CPLEX solver on the so far available incomplete map. It requires to update the map and recompute the visibility of voxels and re-plan the trajectory at each robot's position.
- the `A`* control estimated as a `A`*-like search of the optimal trajectory, which solves the same task as the MILP for the `optimal` control, but the number of expanded nodes is limited 10^5. Again, it is assumed that the map, $\Delta\mathcal{H}$ and visibility of all voxels along the whole robot's path is known in advance.

`GQ-policy` and `Q-policy` policies are modeled by the CNN with the same number of hidden and output layers and neurons, only the number of inputs is different if influence of possible features is evaluated. Considered features are denoted as follows: `D` is sub-sampled layer of pixel depths, $\Delta\mathcal{H}$ is sub-sampled layer of per-pixel-$\Delta\mathcal{H}$ multiplied by depth `D`, which makes it proportional to the sum of per-voxel-$\Delta\mathcal{H}$ in particular viewpoints. Eventually, $\Delta\mathcal{H}\texttt{cog} \approx \frac{\sum \texttt{D}\cdot\Delta\mathcal{H}}{\sum \Delta\mathcal{H}}$ is the center of gravity of $\Delta\mathcal{H}$, which provides the approximate depth in which the voxels with significant $\Delta\mathcal{H}$ are located. Note that for real experiments the `GQ-policy (`$\Delta\mathcal{H}$`+D)` was used.

Policies and features: Table 3.3 compares all these methods, especially for `GQ-policy` the influence of alternative features is shown. The performance is measured by the relative sum gain ($\Delta\mathcal{H}$) of covered voxels defined as

$$\texttt{rs}\Delta\mathcal{H} = \frac{\texttt{achieved\_sum\_of\_}\Delta\mathcal{H}}{\texttt{optimal\_sum\_of\_}\Delta\mathcal{H}}.$$

Value of $\texttt{rs}\Delta\mathcal{H} \in [0; 1]$ captures ability of a particular method to cover the voxel with high gain values. Especially, $\texttt{rs}\Delta\mathcal{H} = 1$ is theoretical maximum which has been achieved by the optimal planning on the full map with gain of all voxels known in advance; $\texttt{rs}\Delta\mathcal{H} = 0.5$, means that the method covered only half of the gain than the optimal planning.

Method	$\texttt{rs}\Delta\mathcal{H}$
`GQ-policy` ($\Delta\mathcal{H}$)	0.807 ± 0.109
`GQ-policy` ($\Delta\mathcal{H}$+D)	0.846 ± 0.109
`GQ-policy` ($\Delta\mathcal{H}$+D+$\Delta\mathcal{H}$cog)	0.884 ± 0.077
`optimal-incomplete`	0.847 ± 0.100
`greedy`	0.657 ± 0.114
`Q-policy` ($\Delta\mathcal{H}$+D+$\Delta\mathcal{H}$cog)	0.722 ± 0.114
`A`* `with` 10^5 `nodes`	0.943 ± 0.055
`optimal`	1.000 ± 0.000

Table 3.3: Comparison of policies and features. The hardware setup corresponds to the one used in real experiments. Each row corresponds to the results achieved by particular method on 64 synthetically generated testing maps.

Method	$\texttt{rs}\Delta\mathcal{H}$
`GQ-policy 7 viewpoints,` 180°	0.853 ± 0.118
`GQ-policy 13 viewpoints,` 180°	0.846 ± 0.109
`GQ-policy 25 viewpoints,` 180°	0.821 ± 0.109
`GQ-policy 24 viewpoints,` 360°	0.853 ± 0.089
`greedy 7 viewpoints,` 180°	0.772 ± 0.141
`greedy 13 viewpoints,` 180°	0.657 ± 0.114
`greedy 25 viewpoints,` 180°	0.676 ± 0.114
`greedy 24 viewpoints,` 360°	0.628 ± 0.126

Table 3.4: Influence of different hardware setups: Table demonstrates influence of action discretization and range of thermal camera for the proposed `GQ-policy` and `greedy` policy.

Fig. 3.11 shows that learning the `GQ-policy` with *relative* Q-values outperforms learning with *absolute* or *not normalized* Q-values, see Section 3.2.3.2 for Q-values definition. Consequently proposed `GQ-policy` is learned with *relative* Q-values in all experiments.

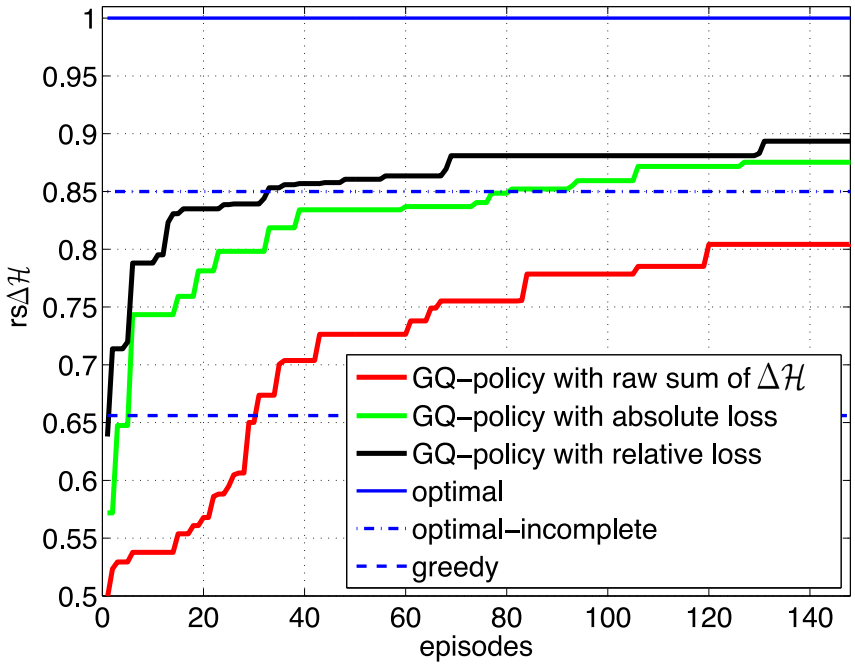


Figure 3.11: Relative sum of $\Delta\mathcal{H}$ as a function learning episodes.

Action discretization and range: We also evaluate the influence of action discretization and range within which the thermal camera operates. The corresponding results are summarized in Table 3.4. The action discretization is given by the number of distinguished viewpoints. Range 180° corresponds to the thermal camera operating in two frontal quadrants. Range 360° corresponds to the thermal camera operating in all four quadrants with allowed turn over. This experiment reveals, that when proposed `GQ-policy` is used, then the resulting relative coverage does not depend on the discretization or the range. On the other hand, `greedy-policy` tends to get stuck for finer discretizations.

Real experiments using a search-and-rescue platform

The control policies were also evaluated on the mobile search-and-rescue platform and the test sequences from the panoramic dataset described in Section 3.2.5. In spite of the data being pre-recorded to allow fair experimental comparison of all methods, the generated data very much correspond to what would be captured during an online experiment with the same platform. Minor variations may be due to a slightly longer time needed for the recording session compared to an online experiment, as all thermal images had to be captured instead of just one at each position. These variations include possibly larger temperature changes in the vicinity of human participants during this extended time period and mostly negligible changes in pose of these participants between capturing individual thermal images. We believe that the resulting effects are negligible; in any case, these effects are same for

all methods being evaluated. As in the synthetic experiments, the robot was following a path discretized into 14 positions at which viewpoints were to be selected. Viewpoint i_k at position k was selected based on the observations from the preceding position $k-1$. Example images from the experiment are shown in Fig. 3.12.

We compared the following control policies:

- *RGBD* uses only the segmentation from $S_\theta(\mathbf{x})$ and thus no thermal measurements. It provides a loose lower bound on the performance since the additional thermal modality provides an important cue with respect to the segmentation task and improves the performance in general, no matter what views are selected.
- *DQN* provides reactive control similar to Mnih *et al.* [72] with the double DQN extension from [94] and the prioritized experience replay from [95].
- *Greedy* D_{KL} corresponds to the $\Delta\mathcal{H}_{\omega_1}$ network predicting the gain obtained through self-supervision. The predicted pixel-wise gain is accumulated by viewpoint kernels and the maximum within the motion constraints is selected for the next action.
- GQ_0 D_{KL} corresponds to the Q_ω network obtained from the self-supervised policy initialization.
- GQ_1 $\Delta\mathcal{H}$ corresponds to the Q_ω network fine-tuned on the guiding trajectories ($p=1$) with ω_1 previously trained to predict true gain $\Delta\mathcal{H}$.
- *Optimal* uses additional information of true $\Delta\mathcal{H}$ to plan the optimal trajectory by solving instances of MILP.

DQN usually needs millions of examples to achieve satisfying results. The computational complexity of our task does not allow to sample such a number of training data. Consequently, we modified some parameters to accommodate our setting.[10] The optimization was carried out in the Tensorflow library [96] using SGD with gradient clipping to maximum norm of 10. The DQN network used the same architecture as our Q_ω network but without normalizing the gain prior to the fully-connected control sub-network as it must predict absolute expected rewards. The gain-predicting sub-network was initialized with the same parameters ω_1 as GQ_1 $\Delta\mathcal{H}$ prior to fine-tuning, the control sub-network was initialized with random weights according to [90]. During learning, 10^4 experience examples were gathered in total. Finally, the model achieving the highest rewards on the validation sequences was selected for testing.

[10] Training parameters of DQN:

batch size	1	replay memory size	10^3
learning rate	10^{-4}	replay start size	50
gradient momentum	0.99	initial exploration prob.	0.9
target network update freq.	100	final exploration	0.1
discount factor	0.99	final exploration frame	5000

Method	AP	Advantage over RGBD	Relative w.r.t. Optimal
RGBD	0.454	0.0 %	89.3 %
DQN	0.486	7.1 %	95.6 %
Greedy D_{KL}	0.489	7.9 %	96.3 %
GQ_1 $\Delta\mathcal{H}$	0.490	8.1 %	96.5 %
GQ_0 D_{KL}	**0.498**	**9.8 %**	**98.1 %**
Optimal	0.508	12.0 %	100.0 %
Complete RGBDT	0.591	30.3 %	116.3 %

Table 3.5: Average precision (AP) for resulting human-background segmentation of the voxel maps from 20 instances of the simultaneous exploration and segmentation task. The instances were generated from 4 full test sequences by randomly selecting starting position k, viewpoint i_k, and planning horizon K. Besides absolute AP, we also list the relative advantage over the `RGBD`-only segmentation and relative AP with respect to the *Optimal* policy. Note that the *Optimal* policy uses the ground-truth $\Delta\mathcal{H}$ gains to guide the planning; this information is not available to other policies.

Our control policies GQ_0 D_{KL} and GQ_1 $\Delta\mathcal{H}$ were initialized using the model parameters learned in Sec. 3.2.4. The GQ_1 $\Delta\mathcal{H}$ network was further fine-tuned on 2198 training examples from optimal plans provided by the CPLEX solver as solutions to the corresponding instances of MILP. From the guiding trajectories, 15 were of full length (i.e., 14 planned viewpoints) and 29 were of varying length ≥ 5 generated from the same source data. To reduce the planning time, planning horizon $K = 6$ were used, which still allowed to plan one full sweep ahead. The model which achieved the lowest error on 578 guiding examples from 4 validation sequences was selected for comparison.

Since the *RGBD* and *Optimal* policies provide loose bounds on the performance from both sides, we are actually interested in evaluating the relative performance with respect to these bounds. In Table 3.5, we list average precision for 20 instances of the simultaneous exploration and segmentation task. Besides absolute AP values, we also list the relative performance w.r.t. these bounds to allow easy comparison.

Using temperature as an additional modality improves the performance and the extent of such improvement varies with policy, due to different thermal images captured. Using self-supervised gain-predicting sub-network *Greedy* D_{KL} alone already provided a competitive alternative to *DQN*. Using complete Q_ω networks further improved the average precision, with GQ_0 D_{KL} having an advantage of another 1.7% over the GQ_1 $\Delta\mathcal{H}$ policy. A possible reason for the self-supervised control network outperforming the GQ_1 $\Delta\mathcal{H}$ policy may be a larger tendency to overfit on our panoramic dataset, which still has a rather limited size.

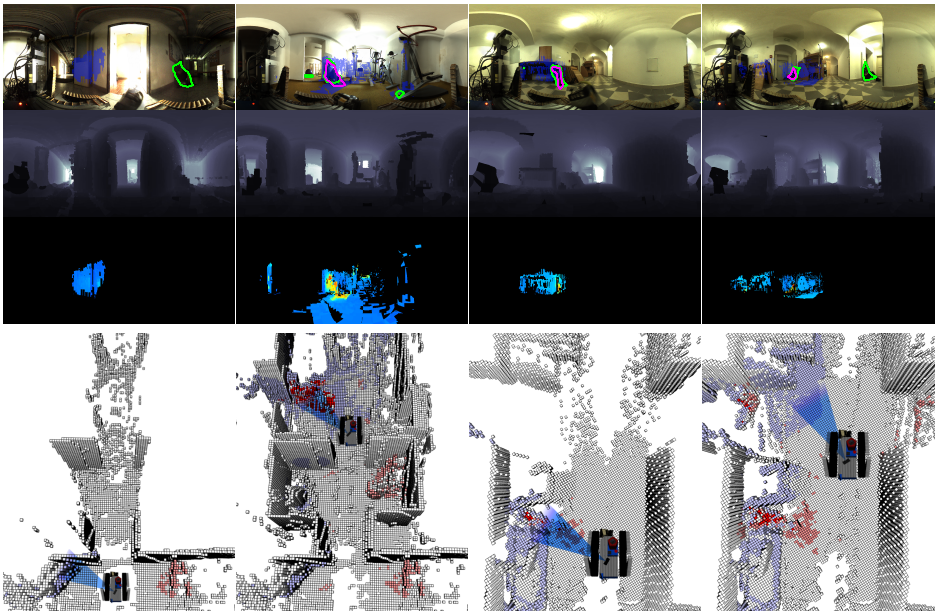


Figure 3.12: Panoramic images and corresponding voxel maps from experiment on test data with the mobile search-and-rescue platform. **Top:** three panoramic images: (top row) `RGB` image with humans delineated by green and magenta contours, as given by segmentation from the $S_\theta(\mathbf{x})$ and $S_\psi(\mathbf{x}, \mathbf{z})$ networks, respectively, and blue overlay denoting the accumulated temperature measurements, (second row) depth image and (third row) thermal image, both rendered from the voxel map. **Bottom:** reconstructed and segmented voxel map with accumulated thermal measurements in blue. Light red denotes the voxels marked as human by the $S_\theta(\mathbf{x})$ network based on the `RGBD` data only, dark red denotes the voxels marked as human by the $S_\psi(\mathbf{x}, \mathbf{z})$ network based on the data with additional temperature measurements.

3.2.7 Conclusion

We have proposed a guided self-supervised learning method for active semantic mapping with a `RGBD` sensor (calibrated camera with lidar) and a pan-tilt thermal camera. Thorough experimental evaluation on synthetic data justified all design choices such as used features, Q-value normalization, discretization and range of the action space. The best performing setup has been qualitatively and quantitatively evaluated on the real platform and compared to standard baselines on challenging real-world search and rescue exploration scenarios, which intentionally comprised many confusing objects such as dolls, pile of empty coats heated on the human temperature, strongly over/under-illuminated areas or heavily occluded humans. Since full thermal scans has been recorded for all robot positions in all 23 scenarios, the resulting *active* search and rescue dataset, which has been made publicly available, could serve for a fair and repeatable comparison and further development of active semantic mapping methods.

Comparison of evaluated policies reveals that using actively controlled

thermal camera yields 7% higher average precision than the pure `RGBD` sensor. The best performing setup ($\mathrm{GQ}_0 D_{\mathrm{KL}}$) has achieved 3% higher average precision with respect to the DQN baseline. While DQN often suffer from slow convergence or from the convergence to a local minimum due to the delayed rewards, proposed guided Q-learning overcome these problems by generating the guiding samples. Estimation of guiding samples is formulated as the optimization problem, which requires transition probability. Both the transition probability and the optimization itself may not be tractable in some cases.

We have been able to achieve average precision around 0.5 on the provided active search-and-rescue dataset, which gives a large space for future improvements. The main source of errors stems from (i) robot motion estimation inaccuracy, which caused multiple responses of a single human in the resulted semantic map; and (ii) insufficient number of real training data, which would sufficiently represent visual diversity of real search and rescue missions. Although we aim at close to real-time method we are not there yet. Real-time usage on our current platform is mainly limited by the depth measurement speed, one complete laser scan takes about 3 seconds, and the absence of GPU on board.

Chapter 4

Terrain traversal learning

One of the crucial tasks of the mobile robot is the ability to traverse the terrain. In the following section, we focused on controlling the augmented robot tracks (called flippers) of the Tradr UGV platform mentioned in the previous sections. Clever control of these flippers allows the robot to traverse a lot of different obstacles. It is an important function not only when the robot is fully autonomous but also in cases when the robot is teleoperated. The additional degrees of freedom is a big load for the operator and it is highly desirable to let the human control only the direction or the speed of the robot. Therefore we propose the method that autonomously controls the flippers.

We have extended the reinforcement learning method [97] by adding constraints, which have to hold not only for the final policy but for the whole learning process. Learning the policy to control the flippers should take an immense number of evaluations which can lead to wear or damage of the robot. The constraints we propose help to converge faster and safer. In the following section, we show this method working with the safety constraints (defined by the pose of the robot), but the constraints could be defined in a lot of different ways, for example, observability constraints. This chapter is based on our publication "Autonomous Flipper Control with Safety Constraints" which was published at the IROS conference (International Conference on Intelligent Robots and Systems) [5]. This conference has A CORE ranking in robotics.

4.1 Constrained relative entropy policy search

The task of Reinforcement Learning (RL) [98] is to search through the space of policies $\pi : S \rightarrow A$, which map agent states S to possible actions A; the actions are then applied either in reality or using a transition model, and the agent reaches a new state (this description is usually known as Markov Decision Process, MDP). RL expects that the agent is rewarded for being in state s with a reward $R(s) \in \mathbb{R}$, and searches for a policy that maximizes the expected reward over all trajectories the agent might execute. It is different from supervised learning, which maximizes the immediate reward and not the long-term one.

Policy Gradient (PG) methods [99] stochastically optimize the expected sum of rewards by direct sampling in the policy parameter space. Learning performance of PG methods does not depend on the complexity of controlled systems. Such property makes them suitable for learning of controllers for robotic systems for which robust real behavior prediction using the first-principle models is difficult (such as closed form equations describing kinematic and/or dynamic behaviors for rover-terrain interaction, or Navier-Stokes aerodynamic laws). Unfortunately, PGs usually require many trials which endanger the real system or cause its excessive wear. Therefore, they are usually not used directly on the real system, but on data-driven models [97, 100, 101], or in simulators [102].

Learning performance of PG methods depends only on the number of policy parameters being optimized [101]. Since it is not easy to find a useful low-dimensional policy parameterization, the sampling is often time-consuming, even if the model is implemented on a GPU [97]. We argue that constraining the PG on a smaller subset of possible policies leads to more efficient policy sampling. We suggest to include various implicit constraints, such as safety or maximum joint angles, into the PG search.

GPREPS uses a stochastic upper-level policy which generates deterministic lower-level policy samples. The expected sum of rewards of these samples is evaluated on an existing Gaussian Process (GP) model and used to estimate the upper-level policy gradient. In contrast to the original GPREPS, the proposed method called *Constrained-REPS* (CREPS) evaluates also other properties of the generated samples and successively constrains the upper-level policy distribution to sample from a bounded distribution, which (i) reduces the number of needed samples/iterations and consequently speeds-up the learning process of the model, and (ii) provides a safe policy when used with the real system.

While deciding if joint angles are in given bounds is obvious, evaluating safety is more complicated. Since it is difficult to provide any guarantees on data-driven models created from real-world samples without any prior knowledge about the underlying physics [103, 104], we replace the GP model from GPREPRS by a *cautious* physics-based simulator [105] that certifies the safety of generated samples. The cautious model of a system is defined as a model that generates unsafe trajectories for all policies which are unsafe on the real system, but could also generate unsafe trajectories for policies which are safe on the real system.

The major contribution of this work is twofold: (i) We propose extending GPREPS [97] to a new constrained Policy Gradient method by including implicit constraints which cannot be derived explicitly from first-principle models. (ii) The new algorithm is evaluated on an autonomous flipper control task on real search-and-rescue rover platform (see Fig. 4.1) and the results are compared with two existing methods [97] and [106].

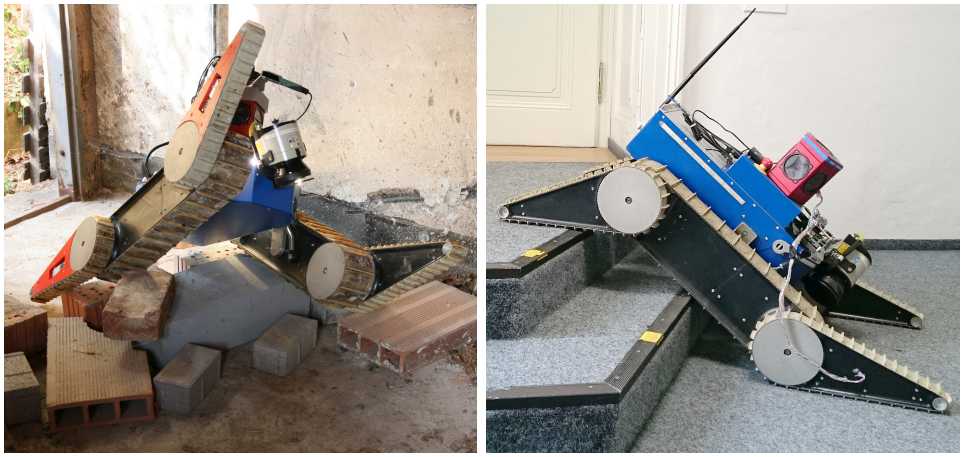


Figure 4.1: Tradr UGV. Track flippers visible on the side of the robot. A policy governs active tilting of the flippers assuring a *safe traversal*.

4.1.1 Related work

Policy Gradient methods proved useful for systems which are not easy to model. For example, Kupcsik et al. [97] demonstrate data-driven PG learning of the ball throwing problem with a robotic arm, and Tedrake et al. [101] argues that Policy Gradient learning for aerial maneuvers with an ornithopter may be very efficient in fact. Transeth et al. [107] show that for snake-like robots with significant side-slip, no closed form expression of the snake's motion exists, therefore policy learning must resort to simulation.

Constraints have been imposed into several PG methods. Uchibe and Doya [108] propose constrained policy search by for GPOMDP [109]. However, the GPOMDP belongs to early PG algorithms which use the likelihood-ratio trick to compute the gradient of the expected sum of rewards and then update the policy parameters by a user defined learning rate. In contrast to [108], we impose constraints into the GPREPS algorithm. GPREPS bounds the KL divergence which typically yields uniform convergence in the whole parameter space and increased learning speed. Prashanth [110] propose constrained PG method for Stochastic Shortest Path problem with inequality constraints on Conditional Value-at-Risk (CVaR) as a risk measure. This method does not allow to include implicit constraints and cannot be easily extended for general episode-based rewards such as minimum distance of the trajectory from a target positions.

Despite the fact that safe exploration becomes a key issue ([111, 100]), it has remained almost neglected in the general reinforcement learning community [112]. Akametalu et al. [104] propose Policy Gradient with safety metrics based on reachability analysis and demonstrate the algorithm on experimental quadrotor application. In contrast to us, they restrict the model to the class of control-affine systems with locally Lipschitz continuous functions.

Bagnell [111] encodes safety into uncertainty of the dynamics model, and assigns negative rewards for leaving an area close to already visited states.

Generally, connecting safety and rewards into a single function is popular [113, 106, 112]. However, as it is shown in e.g. [114, 115], separating safety and rewards into independently optimized measures simplifies the learning process and allows for re-using the safety constraints for multiple different tasks. It is also usually unclear how to choose the weight vector to sum up the reward and safety terms, and what is the impact of negative numbers with high magnitude on the performance of policy search methods.

Moldovan and Abbeel [116] define safe policies as those preserving *ergodicity*, which means from any state there exists a policy that returns the robot to the initial state. This definition is too strict, since it discards a whole class of problems where inverse actions do not exist or returning to the starting state is not possible or even desired. A real word example is the *Safe Traversal* (ST) task defined in section 4.1.3. In this task, the policy does not control the forward speed, which is constant; therefore, it can never reach the starting state again, and still there are safe policies for this task.

4.1.2 Theory

Preliminaries

Model-free policy search algorithms usually follow these steps: (i) generate trajectories from the real-world system, (ii) compute a policy maximizing the expected sum of rewards on the so-far-generated trajectories, (iii) use the policy to generate a new real-world trajectory, (iv) repeat from (ii). Contextual REPS [97] adds a task-dependent context $\mathbf{s}$ (a changing property of the environment, e.g. the height of an obstacle), from which it extracts a feature vector $\boldsymbol{\phi}(\mathbf{s})$. This feature vector is an input of the stochastic *upper-level* policy $q(\mathbf{s}, \boldsymbol{\omega})$, which generates parameter vectors $\boldsymbol{\omega}$ which, in turn, define the lower-level policy. Samples $\boldsymbol{\omega} \simeq q(\mathbf{s})$ are evaluated on the model (which is called a *rollout*) and the corresponding sums of collected rewards (or a single episodic reward) $\mathcal{R}_{\mathbf{s}\boldsymbol{\omega}}^{[i]}$ are recorded. Using this data, Contextual REPS searches for a new upper-level distribution $p(\mathbf{s}, \boldsymbol{\omega})$, which maximizes the expected sums of rewards while staying close to the so-far-generated trajectories. The distance of trajectories is measured by Kullback-Leibler (KL) divergence. Bounding the KL divergence between $p(\mathbf{s}, \boldsymbol{\omega})$ and $q(\mathbf{s}, \boldsymbol{\omega})$ as follows:

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \log \frac{p(\mathbf{s}, \boldsymbol{\omega})}{q(\mathbf{s}, \boldsymbol{\omega})} \leq \epsilon,$$

where ϵ specifies the trade-off between exploration and exploitation, was shown to lead to uniform convergence in the whole parameter space [117]. Another constraint imposed on the upper-level distribution in Contextual REPS is to preserve the average distribution of context features:

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \boldsymbol{\phi}(\mathbf{s}) = \hat{\boldsymbol{\phi}},$$

where $\hat{\boldsymbol{\phi}}$ is the average feature value.

Additional constraints

We extend Contextual REPS with additional constraints. In particular, for systems which are not inherently safe, or which are prone to wear, we use a cautious physics-based simulator to determine a rollout safety $S_{\mathbf{s}\boldsymbol{\omega}}$. The safety equals to 1 if policy $\boldsymbol{\omega}$ generated a safe trajectory for context $\mathbf{s}$, and equals to 0 otherwise. We force the upper-level distribution $p(\mathbf{s}, \boldsymbol{\omega})$ to have expected safety bigger than user-defined threshold δ

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega})(1 - S_{\mathbf{s}\boldsymbol{\omega}}) \leq \delta \tag{4.1}$$

Another source of additional constraints is a prior knowledge of physical limits such as the maximal joint angles (which are the control actions in our experiment). Violating such constraint is usually not safety-critical, because reaching an impossible pose is often prevented by some low-level motor drivers. However, evaluating many impossible samples naturally slows down the learning process.

Since all of these constraints have the same form, we compose a vector $\mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}$ as a collection of evaluated quantities (e.g. safety and mechanical constraints) and vector $\boldsymbol{\delta}$ as a collection of corresponding bounds. Such notation yields the following set of inequalities

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega})(\mathbf{1} - \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}) \leq \boldsymbol{\delta} \tag{4.2}$$

where $\mathbf{1}$ denotes a vector with all-ones of a corresponding dimension.

Constrained REPS

Constrained REPS searches for an upper level policy distribution $p(\mathbf{s}, \boldsymbol{\omega})$ corresponding to the solution of the following optimization problem:

$$\begin{aligned}
&\max_{p} \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \mathcal{R}_{\mathbf{s}\boldsymbol{\omega}}, \\
&s.t. \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \log \frac{p(\mathbf{s}, \boldsymbol{\omega})}{q(\mathbf{s}, \boldsymbol{\omega})} \leq \epsilon, \\
&\quad \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega})(\mathbf{1} - \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}) \leq \boldsymbol{\delta}, \\
&\quad \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \boldsymbol{\phi}(\mathbf{s}) = \hat{\boldsymbol{\phi}}, \\
&\quad \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) = 1.
\end{aligned} \tag{4.3}$$

We follow the same derivation as proposed in [97] and solve the problem by the method of Lagrange multipliers (the detailed derivation is provided in [97] and is not given here due to space constraints). By setting the gradient of the corresponding Lagrangian with respect to $p(\mathbf{s}, \boldsymbol{\omega})$ to zero, we obtain the closed form solution

$$p(\mathbf{s}, \boldsymbol{\omega}) \propto q(\mathbf{s}, \boldsymbol{\omega}) \exp\left(\frac{\mathcal{R}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right) \tag{4.4}$$

where $\boldsymbol{\gamma}, \boldsymbol{\theta}$ and η are solutions of the following dual problem

$$\begin{aligned} &\max_{\eta,\boldsymbol{\gamma},\boldsymbol{\theta}} \; g(\eta,\boldsymbol{\gamma},\boldsymbol{\theta}) \\ &s.t. : \boldsymbol{\gamma} > 0, \\ &\qquad \boldsymbol{\eta} > 0, \end{aligned} \tag{4.5}$$

where

$$\begin{aligned} g(\eta,\boldsymbol{\gamma},\boldsymbol{\theta}) =& \eta \log \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} q(\mathbf{s},\boldsymbol{\omega}) \exp\left(\frac{\mathcal{R}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right) \\ &+ \eta\epsilon + \boldsymbol{\theta}^\top \hat{\boldsymbol{\phi}} + \boldsymbol{\gamma}^\top \boldsymbol{\delta}. \end{aligned} \tag{4.6}$$

Using a dataset $\mathcal{D} = [\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}, \mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}, \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}]_{i=1,\dots,N}$ where samples are picked from distribution $q(\mathbf{s},\boldsymbol{\omega})$, we can rewrite the previous equation as

$$\begin{aligned} g(\eta,\boldsymbol{\gamma},\boldsymbol{\theta};\mathcal{D}) =& \eta \log \left[\frac{1}{N}\sum_{i=1}^{N} \exp\left(\frac{\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right)\right] \\ &+ \eta\epsilon + \boldsymbol{\theta}^\top \hat{\boldsymbol{\phi}} + \boldsymbol{\gamma}^\top \boldsymbol{\delta}, \end{aligned} \tag{4.7}$$

Dual problem (4.5) is a convex function with lower bound constraints. We achieved the fastest convergence with the interior point algorithm [118] with supplied gradients:

$$\begin{aligned} \frac{\partial g}{\partial \eta} =& \epsilon + \log \frac{1}{N}\sum_{i=1}^{N} Z(\mathbf{s}^{[i]},\boldsymbol{\omega}^{[i]}) + \\ &- \frac{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]},\boldsymbol{\omega}^{[i]})(\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}})}{\eta \sum_{i=1}^{N} Z(\mathbf{s}^{[i]},\boldsymbol{\omega}^{[i]})}, \end{aligned} \tag{4.8}$$

$$\frac{\partial g}{\partial \boldsymbol{\gamma}} = \boldsymbol{\delta} + \frac{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]},\boldsymbol{\omega}^{[i]}) \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]},\boldsymbol{\omega}^{[i]})} - \mathbf{1}, \tag{4.9}$$

$$\frac{\partial g}{\partial \boldsymbol{\theta}} = \hat{\boldsymbol{\phi}} - \frac{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]},\boldsymbol{\omega}^{[i]}) \boldsymbol{\phi}(\mathbf{s}^{[i]})}{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]},\boldsymbol{\omega}^{[i]})}, \tag{4.10}$$

where $Z(\mathbf{s}^{[i]},\boldsymbol{\omega}^{[i]}) = \exp\left(\frac{\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma} \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right)$.

Probabilities $p^{[i]}$ of the new upper-level distribution are estimated from the optimal dual variables $\boldsymbol{\theta}, \boldsymbol{\gamma}, \eta$:

$$p^{[i]} \propto \exp\left(\frac{\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right) \tag{4.11}$$

To generate samples from this distribution, we either use weighted maximum likelihood to fit a normal distribution into samples $(\boldsymbol{\omega}^{[i]}, \mathbf{s}^{[i]})$ weighted by probabilities $p^{[i]}$ as suggested in [97] or we use importance sampling to generate samples from the non-parametric distribution. Constrained REPS is described in Algorithm 9.

Algorithm 9: Constrained REPS

Input: maximal information loss ϵ, vector of constraints $\boldsymbol{\delta}$, context distribution $\mu(\mathbf{s})$, initial upper-level policy $\pi(\boldsymbol{\omega}|\mathbf{s})$, number of policy updates K, number of samples N.

for $k = 1, \ldots, K$ **do**

  for $i = 1, \ldots, N$ **do**

    Observe $\mathbf{s}^{[i]}$ from $\mu(\mathbf{s})$.

    Generate parameters $\boldsymbol{\omega}^{[i]}$ from $\pi(\boldsymbol{\omega}|\mathbf{s}^{[i]})$.

    Using $\boldsymbol{\omega}^{[i]}$ execute lower-level policy on the model and collect $\mathcal{R}_{\mathbf{s}\boldsymbol{\omega}}^{[i]}$, $\mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}^{[i]}$.

  Fill in dataset

  $\mathcal{D} = [\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}, \mathcal{R}_{\mathbf{s}\boldsymbol{\omega}}^{[i]}, \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}^{[i]}]_{i=1,\ldots,N}$.

  Optimize dual function

  $[\eta, \boldsymbol{\gamma}, \boldsymbol{\theta}] = \operatorname{argmin}_{\eta', \boldsymbol{\gamma}', \boldsymbol{\theta}'} g(\eta, \boldsymbol{\gamma}, \boldsymbol{\theta}; \mathcal{D})$.

  Compute weights $p^{[i]}$ for all samples in $\mathcal{D}$, Eq. (4.11).

  Update upper-level policy $\pi(\boldsymbol{\omega}|\mathbf{s})$.

Fig. 4.2 shows a toy example. We generate 1000 samples from one-dimensional normal distribution $q(\boldsymbol{\omega})$ with both mean and variance equal to 0.3. We have intentionally chosen the position of rewards maximum into $\boldsymbol{\omega} = 0$, safety equal to one for $\boldsymbol{\omega} > 0.5$ and mean of q into 0.3 to make all constraints active. We set the upper bound on KL-divergence $\epsilon = 0.1$ and the lower bound on safety $\delta = 0.6$. We verify the average safety of samples generated from the distribution given by (4.11) is 0.6064, which is indeed above the required safety bound.

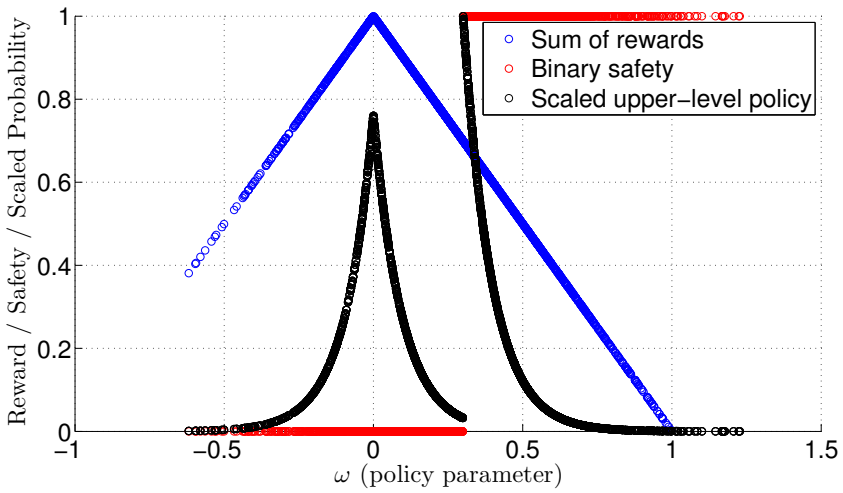


Figure 4.2: Toy example demonstrating solution of problem (4.3). Resulting upper-level probability distribution (equation (4.11)) is in black. Please notice the maximum reward is located in an unsafe area. Therefore, CREPS computes a distribution that prefers safe, though suboptimal choices.

Safety function

One of the additional constraints uses the term $S_{s,\omega}$, which denotes the safety of the rollout (1 is safe, 0 unsafe), and is computed by a *cautious* physics-based simulator. In simple cases, it can be an equation (even implicit), that checks some constraints of arbitrary order. When modeling a complex system, standard software physics and dynamics simulators can be used. The only requirement is that the simulation has to fulfill the cautiousness requirement. From the implementation point of view, the *cautiousness* can be a core part of the simulator design or it is achieved by adding noise to the inputs and outputs, and testing more possible values of uncertain parameters (such as track-soil interaction). This way, it should be possible to create cautious simulations of most of the real-world systems (given the simulator can simulate all the important interactions and influences).

Since the simulator is only approximate (and contrastingly to GPs, it cannot be easily updated by new samples), real-world verification of its reward estimates has to be performed. Therefore, after finding an optimal policy in the simulator, a few more policy search iterations are performed on the real robot. Safety is still a concern, so every sampled lower-level policy is first tested for safety in the simulator, and if it is safe, it is executed on the real robot, and the real-world reward is collected (instead of the one reported by the simulator).

4.1.3 Experiments

Safe Traversal task description

The robot has to learn a *flipper control policy* that would allow it to traverse an obstacle without any prior knowledge about the correct traversal strategy (see Fig. 4.3). This task has been chosen because it very well separates good and bad policies (as well as safe and unsafe). A bad policy is not even able to get the robot on top of the obstacle, and therefore the robot gets stuck in

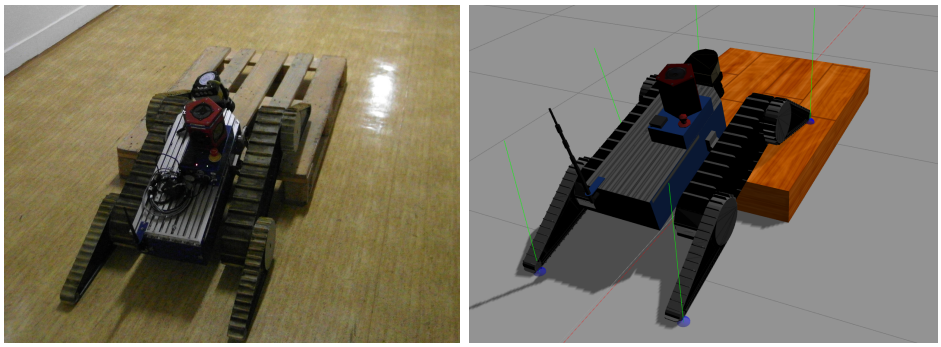


Figure 4.3: The task is to learn how to traverse safely a pallet without any prior knowledge about the correct policy. There are more substantially different policies satisfying our constraints on safety and forward speed. **Left:** real robot executing a safe policy. **Right:** visualization of the simulated robot

front of it and travels only a short distance (receiving low reward). This task also allows for a wide variety of unsafe policies.

Some results of this experiment are compared to a similar task called *Adaptive Traversability* (AT) presented in [106] and improved in [59]. The goal of that task is to find a policy to control the flippers so that the robot maximizes a weighted sum of rewards (and minimizes penalties). The training process is a combination of supervised and reinforcement learning and requires a large set of manually annotated data. Since the forward speed is constant in the task, we compare the policies using the penalties for high pitch angle and for high acceleration.

We use a similar environment for the *Safe Traversal* (ST) task. The tracked robot starts in front of a standard wooden EUR 1 pallet. The robot is automatically driven forward by a constant speed, and the experiment ends after 30 seconds. For an illustration, see Fig. 4.3.

States. States of the ST task are: (i) robot body pitch, and (ii) height of the terrain approximately 20 cm in front of the robot body (read from an octomap built online from laser scans).

Actions. ST policy controls independently the pairs of front and rear flippers using positional control. Therefore, the action space is continuous and 2-dimensional.

Rewards and Safety. In the AT task, safety is not modeled separately, and some of the safety features are part of the reward. The reward for the AT task is a weighted sum of (i) manually assigned safety penalty, (ii) high pitch/roll angle penalty, (iii) penalty for excessive flipper motion, (iv) robot forward speed reward, and (v) motion roughness penalty measured by accelerometers [106].

In the ST task, the reward is simply the distance traveled in 30 seconds over the pallet (the choice of policy influences e.g. track slippage and motor stress, which lower the speed). Safety is modeled explicitly by the cautious simulator, which marks as unsafe all rollouts in which the robot tops over, hits hard on the ground or obstacle (measured as deceleration), or hits objects with delicate parts of its body (e.g. sensors).

Policy. We use a policy that is linear in the states, and that controls front and rear flippers separately. The state vector is 2-dimensional, which yields 3 parameters per action, summing up to 6 policy parameters, $\omega = (\omega_1, \ldots, \omega_6)$, to be learned.

Context. In this experiment, we did not make use of the context—it was always set to zeros. This helped to keep the experiment simple, and was also needed to keep the possibility of comparing ST and AT results. The theory, however, supports the use of context.

Experiment Setup

The experiment follows the pipeline described in Section 4.1.2. First, 10 to 30 iterations of the CREPS algorithm are done using only the simulator (until the policy converges). It returns an upper-level policy, from which we can draw lower-level policies that are safe and lead to high expected rewards. In each iteration, we test approximately 150 simulated rollouts to estimate the sum in the dual function (4.7).

If improvement by real-world samples is intended, the experiment continues with 10 lower-level policies sampled from the optimal upper-level policy and checked in the simulator for safety. If they are safe, they are executed on the real robot. Otherwise, different policies are sampled until the desired number of safe policies is reached.

These real samples are further used to update the upper-level policy in the CREPS algorithm. This way we can correct the policy if the simulator estimated the rewards incorrectly.

Results

We tested the algorithm with several settings to show it consistently converges to high rewards in safe regions. The settings differed in e.g. the initial policy, upper-level policy representation (either a multivariate Gaussian distribution or the Importance Sampling mechanism as described in Equation 4.1.2), and the expected safety lower bound (generally 0.8, but in one experiment it was set to 0.0 to simulate unconstrained REPS).

The probability distribution of safe/unsafe policies showed to be very complex during the experiments, and it is far from being Gaussian or uniform. As can be seen in Fig. 4.4, most of the time, the mean safety of rollouts is below the desired threshold of 0.8. However, it still tries to reach the threshold. In this case, the Importance Sampling method yields better results, as it better represents the complex distribution.

In the two experiments with unconstrained REPS, comparing Fig. 4.4 and Fig. 4.5, it is clear the algorithm strived for the highest rewards possible and safety quickly dropped to almost zero (which means the traversal was faster, but the robot hit ground too hard during the rollout). Interestingly, one of the unconstrained experiments reach a level of expected rewards not seen in any of the safety-constrained cases, which suggests that the best policies are unsafe and CREPS correctly avoids these maxima.

Fig. 4.4 and Fig. 4.5 show that the CREPS algorithm maximizes the rewards in cca the first 10 iterations, and then it holds the good rewards and tries to satisfy the expected safety constraint. As we discussed earlier in this section, since the safe policy distribution is difficult to represent, the expected safety constraint is often broken. However, it does not mean that the robot could be damaged because of this imperfection. It only means that we probably need to sample more lower-level policies until a safe one is found (which is always tested in the simulator before real execution).

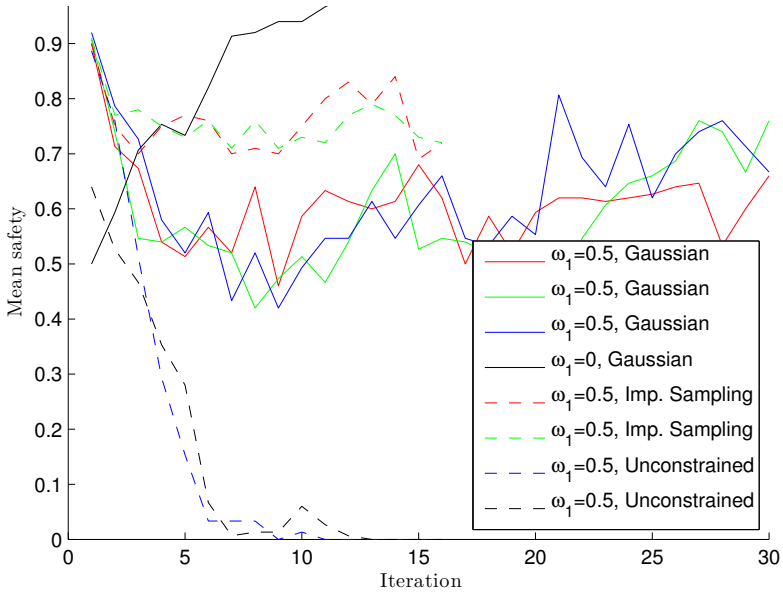


Figure 4.4: Mean safety during rollouts. ω_1 is the initial value of the first element of the policy parameter vector, and it showed to have large influence on safety. The solid black curve is from a policy initialized close to a local maximum of the safety function. Compare with Fig.4.5 to see that the reward increased very slowly with expected safety higher than the desired threshold of 0.8. The last two policies were not constrained by safety at all (behaving like Contextual REPS), and they quickly found the best rewards lie in the unsafe space. Also note the Importance Sampling experiments tend to achieve higher expected safety.

We compared the high-pitch and high-acceleration penalties gathered by both an optimal policy for the ST task and also for the AT task. We performed 10 rollouts with each of the methods, and compared the penalties; results are shown in Fig. 4.6. The pitch histogram was essentially the same for both tasks, so only the acceleration histograms are shown in Fig. 4.6. The figure illustrates that the CREPS policy doesn't generate more dangerous trajectories than the AT policy (which was however trained with a large set of manually annotated data).

Last, we closed the loop improving one of the best policies found in the simulator by real-world reward samples. We executed two CREPS iterations, each with 10 samples. Safety was always checked in the simulator, then the sampled policy was executed, and the real-world reward collected. After two gradient search steps, the expected reward is higher than the best reward achieved in the simulator, as is shown in Tab. 4.1. It is important to note that the policy search now cannot reuse samples from the simulator, since the reward estimate may be biased by imperfection of the simulator. The converged simulated policy performance was lower when used in the real world, but after only 2 real-world iterations, the CREPS algorithm converged to the real-world optimum (which is different from the simulated optimum, since the simulator is only approximate). The reader should notice that the

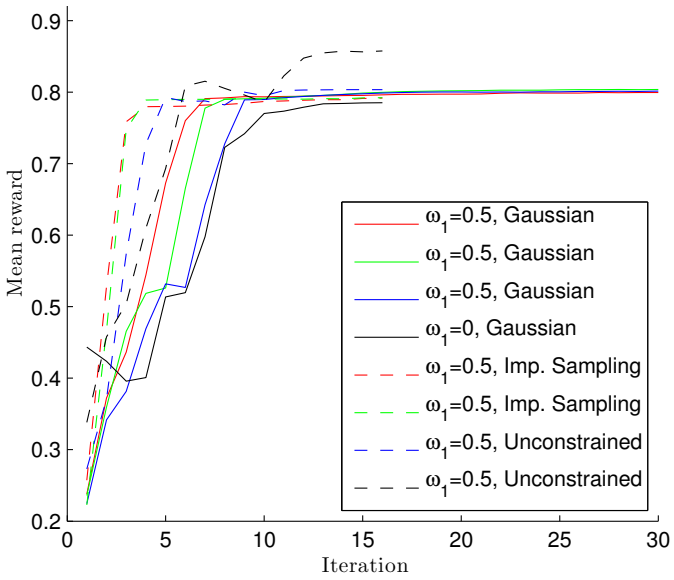


Figure 4.5: Mean reward during rollouts. The last policy not constrained by safety converged to an optimum with the highest expected reward.

simulated optimum had to be close to the real-world optimum, since CREPS doesn't allow large changes of the policy.

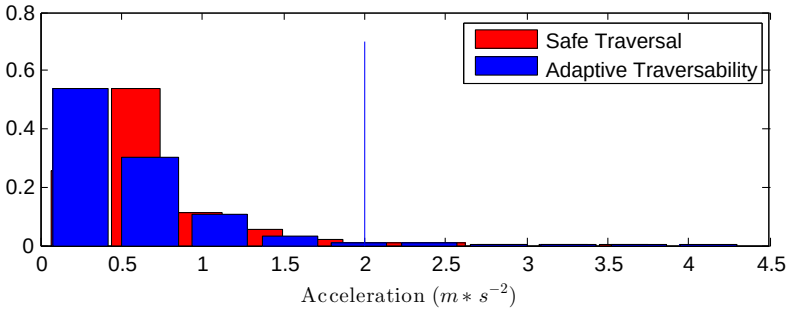


Figure 4.6: Comparison of acceleration during 10 rollouts with AT and with ST (shown as relative histograms). Values over 2 (right to the blue line) are penalized in the AT task.

4.1.4 Conclusion

We extend an existing Gradient Policy search algorithm (Contextual REPS)[97] by adding implicit constraints that help keeping the gradient in a promising direction. We call the algorithm Constrained REPS (CREPS). One of the additional constraints includes robot (system) safety which is determined by a cautious simulator. We presented the basic equations needed for implementation of this algorithm, and we have shown correctness of the algorithm both on a toy example and on a task with real robot. In a small number of iterations (and with about 2000 simulated trajectories), the robot learned

Iteration	Kind	Policy Converged	Mean Reward
29	Simulated	yes (in sim.)	0.80 ± 0.01
30	Real	no	0.73 ± 0.10
31	Real	yes (in real)	0.85 ± 0.05

Table 4.1: Execution in the real world

how to traverse safely a previously unknown obstacle, and even the learning process itself was safe—thanks to the cautious simulator. We show that the cautious simulator can be designed in such a way that it does not constrain the possible actions too much and the robot is still able to reach the safe optimal rewards.

Chapter 5

DARPA Subterranean challenge

Similar to other fields of science, robotics is facing a crisis of research reproducibility. The cost of experiments rises with the amount of logistic and technical issues which need to be solved for any field experiments. The high cost of field experiments causes the evaluations to be performed on datasets often gathered for a different purpose, thus making them either not sophisticated enough or not able to capture the uncertainty and unpredictability of real scenarios. Another reason is the complexity of the systems, where most robots comprise many smaller submodules from which are usually tested separately. Thus, method interoperability, compatibility, and their impact on the whole system's efficiency often remain neglected. The last reason is that failures of experiments are often blamed on technical issues, and the reliability and robustness of the methods are not reported. Instead, scientific papers focus on issues of accuracy or computational complexity of the individual methods rather than the reliability of the integrated systems [119][120]

With all the mentioned problems, the performance of robotic systems cannot be optimal in real-world situations which are impacted by the robustness of the deployed systems. Robotic contests offer a potential solution to the problem. The results of many of these contests, such as MIROSOT [121], Eurobot [122], RoboTour [123], RockIn [124] or MBZIRC [125], show that the success depends more on reliability and interoperability than other aspects.

Figure 5.1: Left: Tradr UGV in the simulated metro station. **Right:** Boston Dynamics Spot robot in cave.

In 2018, Defense Advanced Research Projects Agency (DARPA) announced the Subterranean Challenge (SubT). The primary goal of the challenge was to discover innovative solutions that can navigate, map and search complex underground environments. The DARPA SubT Challenge was organized not only to fund but also to motivate the development of robotics systems that are capable of supporting search-and-rescue operations without any supporting infrastructure in the underground environments (see Fig. 5.1 for examples). The contest emphasizes the reliability and efficiency of complete integrated systems rather than the efficiency of the individual modules and components. In particular, the performance of robotic teams is evaluated by their ability to quickly and accurately locate relevant objects in underground sites with a variable degree of complexity.

This chapter describes the participation of our team called CTU-CRAS-NORLAB (see Fig. 5.2) in this competition and briefly outlines our approach. Since the whole system is vast and complex, only the main parts are described, focusing on topics I have contributed. I was the main person responsible for object detection, and I was the team operator (the only person allowed to communicate with the robots during the missions). This chapter is based on the published reports "System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA Subterranean Challenge" [3], that was accepted in Field Robotics journal, and "Darpa subterranean challenge: Multi-robotic exploration of underground environments" [7], published at Modelling and Simulation for Autonomous Systems International conference.

Figure 5.2: CTU-CRAS-NORLAB team at urban circuit.

5.1 Challenge specification

In the SubT Challenge, a team of mobile robots has to actively search an environment to locate and report the positions and types of specific artifacts in a previously unknown, subterranean environment with limited human interaction. The performance of the robotic team is evaluated by the number of objects detected and correctly localized [126]. The competition was divided into three preliminary rounds, where each round tests the system in a different environment and one final round, which was a mixture of all environments. See Table 5.1 for places and dates of the rounds. The challenges presented by subterranean environments can vary across subdomains. There were three different subdomains presented in the Subterranean Challenge. The first environment where the challenge took place was a human-made tunnel network; the second challenge was in urban underground, and the third environment presented within the challenge was the natural cave environment. The tunnels can be multiple kilometers long with constrained passages, often with hardly traversable terrain with mud, dust, and partially flooded passages. Urban environments are complex, multilevel structures with stairs and holes in the ground with constrained passages such as doors. The natural cave could contain irregular geological structures, with both constrained passages as well as large caverns. The topologies are often unpredictable, and the passages are hardly traversable. All these subdomains have common issues such as an absence of GPS signal, low communication-signal throughput, hardly traversable terrain, and low or no illumination. DARPA added dynamic obstacles and generated smoke in some passages to make the conditions even harder. A virtual component of the same challenge was running in parallel, where the whole system took place in a simulation and had to be strictly autonomous. We attended all system circuits (the real robot system deployment) and two virtual rounds (cave and finals).

Event	Date	Place	Enviroment
Tunnel Circuit	08/2019	Pittsburgh, Pennsylvania	Coal Mine
Urban Circuit	02/2020	Satsop, Washington	Nuclear Plant
Cave Circuit	Canceled	Canceled	Natural Cave
Final Event	09/2021	Louisville, Kentucky	Cavern

Table 5.1: Summary of DARPA organised events for SubT with their environments. Cave circuit was canceled due to covid-19 pandemic. The final circuit's cavern contains artificially made all environment subdomains.

5.1.1 Competition rules

With limited time (30 minutes) and personnel (maximum 5 people at final round), each team has to prepare a robotic system to explore a previously unknown environment inaccessible to the personnel. Fig. 5.3 depicts the

staging area where the team prepares the robot for the mission. The robotic fleet has a mission to find, locate and then report back to base the position of the previously given objects called artifacts within a limited time (usually 1 hour). During the mission only one person (team operator) is allowed to control any aspect of the mission and view the data reported back by the robots.

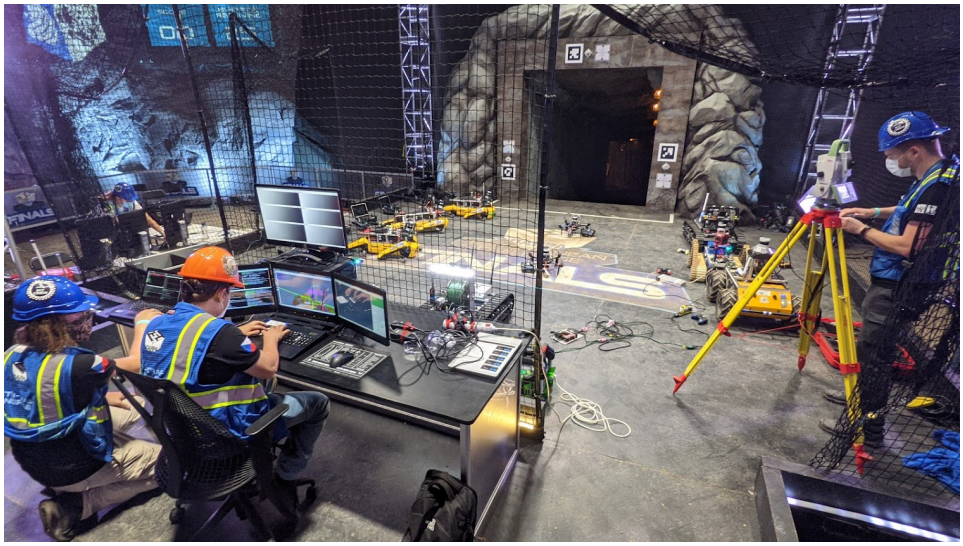


Figure 5.3: Staging area at finals during the preparation for the scored run. Robots are already on their place. Their exact position is measured by the person at right side of the image. Base station is prepared by the team operator (the person with orange helmet).

Artifacts and scoring

All artifacts that have to be found change for each circuit and are specified by DARPA beforehand. New domain specific artifacts were added in each round. See Fig. 5.4 where the artifacts are depicted. The common artifacts were presented in each round extended by domain specific artifacts. In the final round all ten artifacts were presented. One of the artifacts was detectable only by CO2 sensor – gas.

For a team to obtain a point, it is required to

- find and recognize the correct type of artifacts,
- localize the artifact with a maximum of 5m of euclidian distance from the ground truth in the global coordinates,
- and report the correct type and position back to home base.

There are between 20/40 artifacts on the course. Each team can send 40 reports during one run, making it not possible to brute force the problem. If any teams would have the same number of points, other factors such as

how fast the artifacts were reported or how far teams traveled are then used as a tie-breaker. More information can be found on the DARPA SubT website [126].

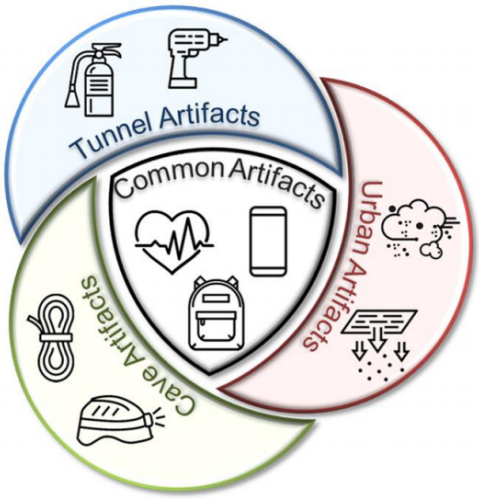


Figure 5.4: Artifacts divided by subdomains.
Common artifacts: backpack, survivor, cellphone.
Tunnel artifacts: fire extinguisher, drill.
Urban artifacts: gas, vent.
Cave artifacts: rope, helmet.
Final artifact: transparent color-changing SubT logo.

5.2 CTU-CRAS-NORLAB solution

This section describes part of the solution of the CTU-CRAS-NORLAB team. Because the whole system is really vast and complex only the main ideas are mentioned below. The main focus is on the parts of the system that are related to my contribution.

5.2.1 Hardware

Our robotic fleet changed a bit between the circuits, but some of the robots were with us from the beginning till the final. The main robotic platforms that contributed to the challenge are described below.

Tradr UGV

Tracked robotic platform developed during the TRADR project. This robot was already mentioned in previous chapters, but we upgraded the hardware during the SubT Challenge (see Fig. 5.5-left). The rotational 2D LiDAR SICK LMS-151 was replaced by the Ouster OS0 with 128 beams. This allows

us to perform the faster and more precise movement because later on we were limited by rotation of laser, which takes 3 seconds to measure complete scene. The camera PointGrey Ladybug 3 omnicamera is used to obtain RGB pictures. From a computational point of view, we extended the robot by GPU unit NVidia Xavier AGX, which allows us to run the detector onboard on 6-8 Hz at the 608×608 image size. As the computational demands of our algorithms increase, we have also added an offloading computer (Intel NUC8-i7) that can process the mapping and other CPU-intensive operations. The main advantage of this robot is the flippers which allow the robot to access the areas not accessible by the wheeled robots (stairs or rugged terrain).

Husky A200

The Husky A200 from Clearpath Robotics is a differentially driven wheeled platform (see Fig. 5.5-right). The platform is built to be rugged and capable of traversing mud, gravel, light rocks, and steep declines/inclines. The main advantage of this platform is the speed in comparison with tracked UGV mentioned above. The sensor equipment changed during the competition a lot. In the first rounds, we used the Robosense 3D lidar with 32 lines at the final competition; we replaced it with the Ouster OS0 with 128 lines. The global shutter of the cameras is a must-have feature in low illumination scenarios. The images are deformed while using a rolling shutter with the long exposure (which is needed to obtain enough light) In the preliminary rounds, we used Bluefox MLC200 cameras that were replaced by Basler Ace 2 a2A1920-51gcPRO for the final round. The robot uses the Xsens MTI-30 inertial measurement unit. NVidia Xavier AGX and Intel NUC8-i7 served sufficient computational power.

Spot

Spot robot by Boston Dynamics is a state-of-the-art walking robot that allows exploring various environments fastly (see Fig. 5.6-left). It can traverse almost all kinds of obstacles, which was the main reason why almost all the teams deployed spots in the final round. We have used the same sensor set as on the

Figure 5.5: Left: Upgraded tracked Tradr UGV. **Right:** Husky A200 wheeled robot.

husky payload. We use Ouster OS0 with 128 lines for depth measurements and five Basler Ace 2 a2A1920-51gcPRO which gives us visual information from the surroundings. The computational resources consist of NVidia Xavier AGX and Intel NUC8-i7.

X500 UAV

The Multi Robot System Group from CTU in Prague made a custom UAV (unmanned areal vehicle) for flying indoors. During the competition, the drone developed a lot, and changes were made between all the rounds. The final version of the drone can be seen on Fig. 5.6 right. It is equipped with Ouster OS0 with 128 scanning lines with an extended 90-degrees field of view. Two Intel Realsense RGBD cameras are used to cover blind spaces of the lidar below and above the drone. The platform is equipped with two dedicated artifact detection cameras, the Basler Dart daa1600. The UAV has an outstanding flight time of 25 minutes. This developed drone is capable of flying in dense indoor environments, even in tight vertical shafts, while being able to localize itself with the required accuracy. This platform was also introduced into the virtual competition, and it was used by almost every team due to its capabilities. Some parts of the system (e.g., mapping, navigation) differ between UGVs and UAVs. Since my personal interest and contribution are mostly in ground robots, the following text is related mainly to them.

5.2.2 Communication channels

During search-and-rescue situations, interweaved hallways, underground tunnels, shielded rooms, and thick steel-reinforced concrete walls make hard or even impossible to maintain any stable radio link. To partially overcome this issue, we designed a combined hardware and software solution. All the platforms were equipped with three types of communication channels (see Table 5.2). WiFi for direct control and debugging when the robots are on line-of-sight. Long-range mesh network Mobilicom sends all essential data and custom-made motes, which could be dropped from the robots that send

Figure 5.6: Left: Spot by Boston Dynamics. **Right:** X500 UAV developed by Multi Robot System Group from CTU

some small critical messages between robots. Motes could help, for example, in cases where the robots went out of the range of the Mobilicom nodes, and the operator wanted them to return to the signal. The software solution is shortly discussed in Section 5.2.4.

Type	WiFi	Mobilicom	MOTE
Range	Short	Medium	Large
Frequency	5GHz	2.3GHz	900MHz
Bandwidth	50 Mbit/s	1 Mbit/s	432 bits/s

Table 5.2: Communication methods comparison.

5.2.3 Software architecture

While appropriate hardware is necessary for successful participation in the challenge, it is the software that brings the entire system alive. To stress the importance of the software, DARPA SubT has a parallel virtual track where the contest occurs only in software simulations of the deployment sites. While not all the teams participated in the virtual track, most of them utilized the simulators as an integral part of the development process since tests in a simulation are much less tedious than real-world testing. The software modules of all robots had to tackle localization, mapping, navigation, object detection, exploration, and multi-robot coordination

All of the systems used in the competition were running the Robot Operating System (ROS) [127] in its Melodic version on Ubuntu 18.04. Choosing ROS allowed us the benefit from the multitude of packages developed by the open-source community around ROS; thus, we could concentrate on the software that is specific for our platforms. For example, the whole driver stack of the Husky robot up to EKF-based localization, RGB camera, and lidar drivers was using just the publicly available code from ROS community. The community-provided packages do not, however, have only advantages. ROS does not (yet) provide any terms of software quality measures, so it happens that as we are using a package, we discover it is either buggy or not written in an optimized manner. Thankfully, as all the packages are open-source, we can always fix the issues locally and then offer the fixes to the upstream repositories.

One of the benefits ROS brings to us is the ease of decision on where should some code be running. As mentioned in Section 5.2.1, most robots carry more computers on board. The various types of algorithms we use and their constant development require high flexibility in terms of computational power, and with ROS, running the CPU- or GPU-intensive programs on different computers is just a matter of a few configuration lines and some one-off provisioning. ROS also provides high-quality visualization tools like Rqt and RViz, which we utilize in our system both during development and for the operator console (described further in Section 5.2.10).

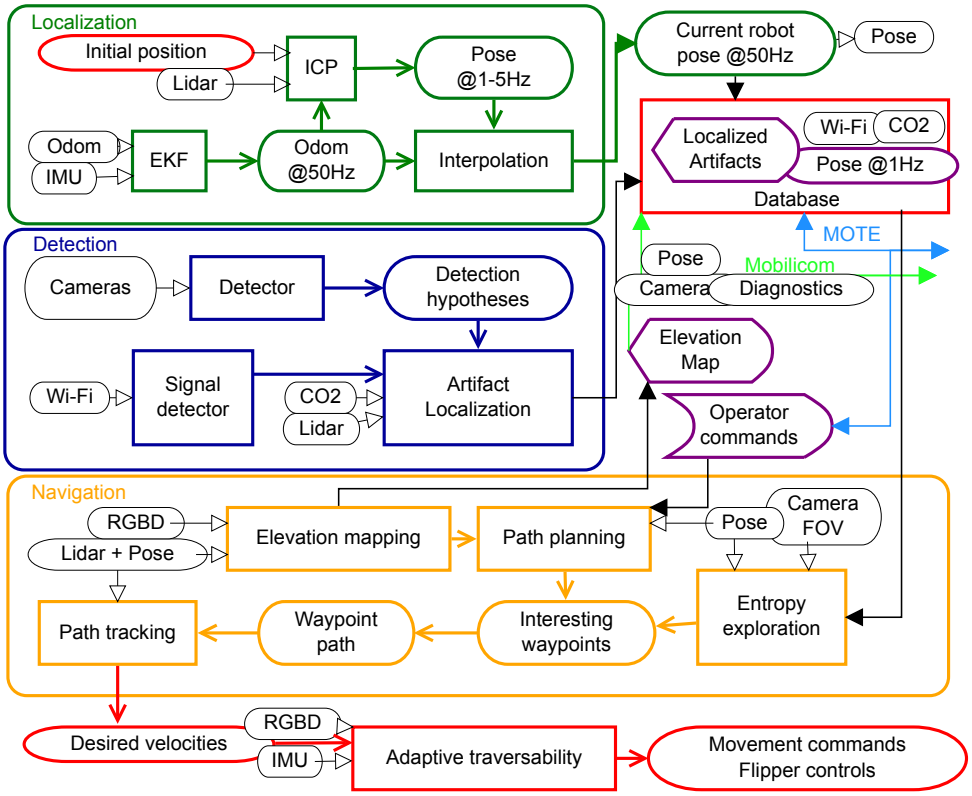


Figure 5.7: Block diagram of software modules on a single UGV platform with three main sub-systems: localization (green), detection (blue) and navigation (yellow). Purple boxes and database indicate the main I/O of a robotic platform that is transmitted to other agents via the MOTE (light blue) and Mobilicom (light green) networks. Internal robot states and sensory information are represented as bubbles while modules are represented as rectangles. Initial position is obtained from the pit crew during setup.

5.2.4 Database

To get a resilient and distributed communication architecture where some links may disappear during the mission, we decided not to use the standard single-master mode ROS supports. Instead, we use a multi-master solution based on a custom build database (running on RocksDB no-SQL backend), which stores both live and historical data. Each robot runs a ROS master on its main computer, so on the robot's local network, all computational nodes and other supporting programs are running as in a standard single-master ROS system. The database is running on each robot and base station and uses broadcast messages to send its own data to neighboring nodes.

The goal is to replicate all data to all nodes in the system. To this end, we employ an asynchronous group replication strategy, that is, the state may not be propagated immediately and any node can serve any data to any other node. To receive missing data, each node regularly broadcasts requests and fills in gaps by using any data it receives. To this end, each

database node maintains a set of active requests for all nodes and regularly serves these requests with the data it has. Assigning messages IDs from consecutive integers allows requesting a consecutive sub-sequence of messages which comes into play when a robot joins other robots after some time of autonomous exploration out of signal. Large messages are divided into chunks and reconstructed on receivers when they are complete, with the chunks having their byte offsets appended to their IDs.

Our implementation allows configuring several parameters of each transmitted topic – e.g. whether the topic is live-only (no history, e.g. camera streams) or the synchronization priority (so that confirmed detections are transmitted earlier than e.g. maps). An important feature is also explicit bandwidth limitation of the synchronization traffic, so that the physical layer (Mobilicom) is not overwhelmed and forced to start buffering.

5.2.5 Localization

The ground robots deployed the localization and mapping solution developed by NORLAB (Laval University). The mapper (*norlab_icp_mapper*[1]) is primarily intended for data captured by 3D lidars. It gains from the stability and speed of the *libpointmatcher* library[2] that implements a variety of Iterative Closest Point (ICP) algorithm variants. This library and the *norlab_icp_mapper* are continuously developed and maintained by NORLAB and publicly available on Github. The performance of the mapper, based on the ground truth provided by the competition organizers, has been satisfactory. Fig. 5.8 presents the expected mapping performance given the initial frame alignment. The presented map is based on the post-event data recorded by the tracked Tradr robot.

5.2.6 Navigation

Besides the map for the localization, each ground robot builds a separate map for navigation and exploration purposes. More in depth principles of navigational maps are based on published [128, 129, 130]. The navigation maps were periodically analyzed for terrain traversability to determine how to navigate the robot to the specified location. Even we have successfully tested the method proposed in Section 2.1, in the final, we decided to use simplified estimation from the roughness of the terrain to ease the complexity of the system. The Dijkstra algorithm produces plans from the robot position to the given goal location. The cost field generated using the Distance transform algorithm [131] has been utilized to keep a safe distance from the obstacles or to avoid rough but traversable terrain if there is other choice.

[1] https://github.com/norlab-ulaval/norlab_icp_mapper_ros
[2] https://github.com/norlab-ulaval/libpointmatcher_ros

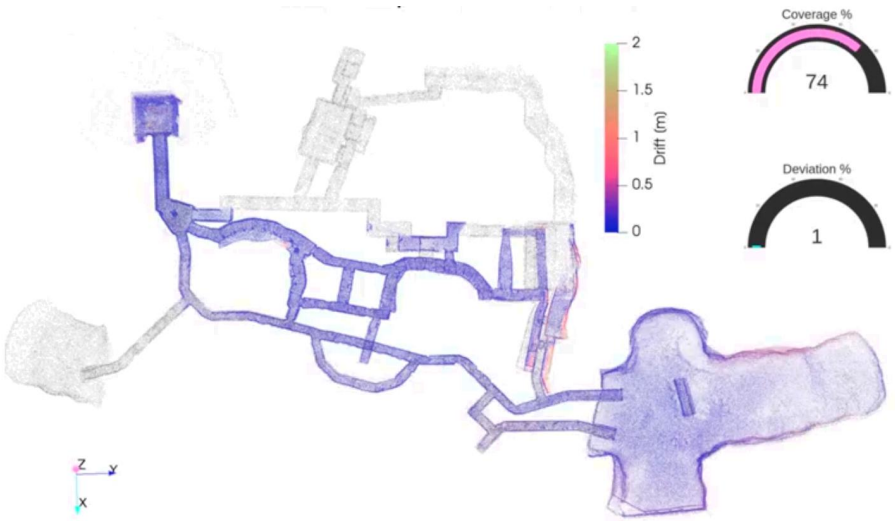


Figure 5.8: The mapper performance during the post-event testing. The robot was teleoperated manually. The mapper settings are identical to the final scored run.

5.2.7 Exploration

The robots create a topological map, which they share with each other, besides the metric map created for local planning. This global topological map is used for decentralized exploration, where each robot explores a different area of the environment. The way the next exploration waypoint is determined from all possible waypoints is based on the approach [132]. In case the waypoint is far from the robot's actual position, the merged topological map from all robots is used for high-level planning. For more information about exploration see [130, 132, 128]. Exploration is invoked when the robots are in autonomous mode. For the virtual track of the competition, they were only autonomous mode. In the system track, the operator sometimes took over the control of the robot.

5.2.8 Adaptive terrain traversal

The tracked robots incorporate the information from their depth sensors and position their flippers accordingly (Chapter 4) to improve their ability to overcome adverse terrain. Thus, flipper control was done most of the time autonomously. This is because manual control would cause excessive cognitive load on the operator, especially when operating several platforms simultaneously.

The approach used in the competition consists of a state machine, shown in Fig. 5.9, where four primary states of obstacle traversability are defined. The neutral state means folded flippers in such a way as to minimize blocking the Lidar. Yellow states define configurations of the flippers which allow the robot to safely climb and descend obstacles by using the front and back

flippers as support. These are triggered when the robot is moving forward on the flat ground, and a significant height increase or decrease is sensed in front of the robot in a designated region. For example, in the *Climbing rear* state, the rear flippers push down to raise the rear while keeping the front flippers extended forward to shift the weight balance maximally forward and dampen the robot when it hinges forward. Whenever the robot finishes the ascent or descent onto the flat ground, the state resets back into neutral. The same state machine is used for stair traversal with a modification that the robot cannot turn while on stairs. When setting flippers for any action, we use the natural tactile feedback of the ground by simply pressing down with low torque. This is better than strictly conforming the flippers to a height map which can be prone to errors.

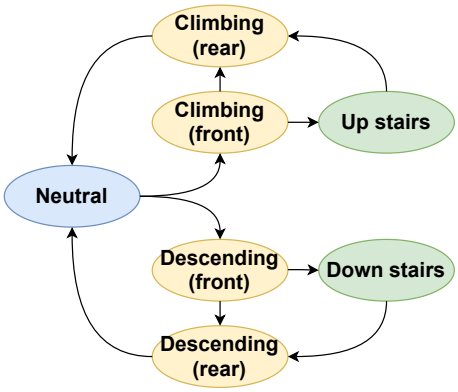


Figure 5.9: Finite state machine for controlling flippers of the Absolem platform, capable of traversing difficult obstacles and stairs. Blue and green states signify longer lasting modes, whereas yellow states are short lasting.

5.2.9 Object detection

The purpose of the entire system is to detect and locate artifacts that represent potential victims or provide a cue of their location (see Fig. 5.4 for artifacts definition). Therefore, every robot of our team performs the object detection task. We have deployed a state-of-the-art neural-based computer vision method called YOLOv3 [133, 134] for object detection on ground robots. On UAVs, the MobileNet2 [135] was used due to limited computational resources. The MobileNet2 serves as the feature extractor for a Single-Shot-Detector (SSD) that, in the end, produces bounding boxes. This is a similar approach with the YOLO, where you only look once without iterating through the parts of the image. These detectors take an RGB image as an input and predict the bounding boxes with their confidences, sizes, coordinates, and class probabilities. After the forward pass of the neural network, only the detections with confidence above the threshold are further processed. The Fig. 5.11 depicts detected objects and false positive examples.

We obtain x and y coordinates (in the camera frame) as the center of the

bounding box. To estimate the z coordinate, we use the known transform between the lidar sensor and camera to project the pointcloud in the camera frame (see Fig. 5.10). From all the points within the bounding box, we estimate the z as a median point. All the points from the pointcloud within the bounding box are used to compute the covariance matrix used later for merging the detections. In case that pointcloud from the actual scan is too sparse and no points correspond to the bounding box, we use a local map to get the depth points. Before we process the detection further, we compare the size of the actual objects with the predicted size of the bounding box, and in case the expected size differs a lot, we throw away the detection.

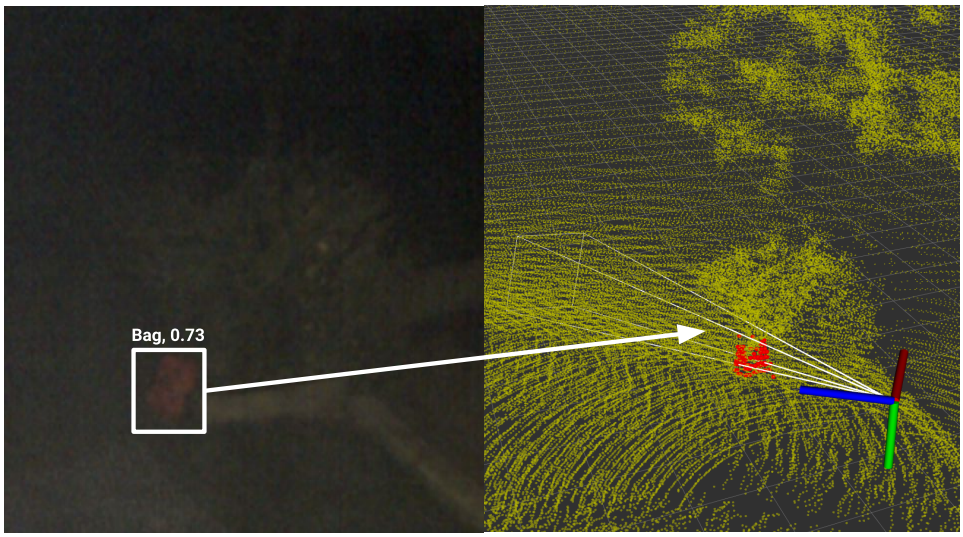


Figure 5.10: Detection of the bounding box and projection of the points inside a camera frame.

Then, the detections are processed by the final position estimator [136, 137], which fuses detections from different images that correspond to the same objects using a principle similar to multi-target tracking. It helps to improve the precision of the estimated position of the artifacts and rejects some false positives. Each new detection creates a new hypothesis (described by state, position estimation covariance, and the number of corresponding measurements) or is associated with existing hypotheses based on maximal likelihood and a gating distance. The final position of the hypotheses is established by the application of Kalman filtering. Only if the hypothesis is certain (covariance lower than threshold and the satisfying number of measurements), the robot sends position, established class, and the most representative image of the detected object to the database. This prevents flooding of the communication network with image data and helps to lower the number of false positives detections send to the operator.

To ease the RGB detections and to improve the chance of finding phones, the Jetson computer was utilized to search surrounding WiFi signals and, using weighted trilateration, estimate the position of the artefact [138]. Since it was necessary to detect gas in the form of elevated CO2, we used Sensorion

(a) : Survivor

(b) : Cellphone

(c) : Backpack

(d) : Drill

(e) : Fire extinguisher

(f) : Vent

(g) : False Backpack

(h) : False Cellphone

Figure 5.11: Detection of artifacts during the challenge. Two false positives are depicted, Backpack as the red sign in water reflection and cellphone as a piece of paper.

SDC30, the sensor dedicated to measuring just CO2. Using sensor data, we extrapolated and showed values to the operator and eventual guesses where the CO2 artifact might be. Both methods are explained in [139].

Detection dataset

We gathered a dataset from various underground environments to train the object detectors. It contains images from the Josef experimental mine, the Prague subway station, the Dobrosov military fortress, the Bull Rock Cave, and the university hallway and basement. All data was collected using the cameras of our robots while another testing was being carried out on the platforms or through cameras that were temporarily dismantled from the robots. The dataset was annotated manually. See Table 5.3 for number of annotated artifacts in the different environments. We made an automatic data generator for the virtual competition, which generates around 100000 images. More images were added iteratively during the tests by selecting the false positive detections.

	Total	Tunnel	Urban	Cave	Misc.
Images	40373	10454	10123	7727	12069
Negative	6132	2205	2790	103	1034
Survivors	7597	2476	1559	1037	2525
Cellphones	5092	1010	1109	1171	1802
Backpacks	5635	1191	1301	1080	2063
Drills	3779	2317	12	38	1412
Extinguishers	7637	1869	247	30	5491
Vents	4747	0	3876	31	840
Ropes	4996	0	0	3575	1421
Helmets	5504	0	0	2639	2865
Cubes	2338	0	0	332	2006

Table 5.3: Dataset details with the number of annotated objects. Negative row corresponds to the images containing only the background.

5.2.10 User interface

As the challenge rules allow only a single human supervisor (team operator) to manage all the systems deployed in the mission, it was necessary to design a reliable and easy-to-use graphical user interface (GUI) to supervise and control the robots' behavior and report back found artifacts. The base station GUI is built on top of standard ROS visualization components, namely custom rqt plugins and rviz tool. The two major parts of the GUI are realized in the base station. One for controlling the robots (control GUI) and the second for reporting the artifacts (detection GUI). Besides them, the operator uses one screen to visualize the camera outputs from the robots and one screen for debugging and necessary scripting. See Fig. 5.12 for basestation layout.

Figure 5.12: The base station and its user interface used in the final circuit event.

Robot control GUI uses an interactive marker server and multiple custom plugins, e.g., to capture keyboard events, in rviz for control of individual robots. It allows to show and hide the constructed maps of individual robots and their horizontal cross-sections to deliver the human supervisor the spatial awareness and assign high-level commands using an interactive marker and keyboard shortcuts. The main commands that can be sent are *Explore*, *Follow given* (robot will plan the path to the selected waypoint), *Force follow* (robot is going straight to the waypoint without planning - this is useful when the robot is stuck somewhere), *Stop*, *Clear topological map*, *Clear metric map* and Add abandoned area (which prohibits planning trough this area). The UAVs have in addition the *Land* command, which forces the UAV to land immediately.

The detection GUI serves to accumulate the detections from all the robots and present these detections to the human supervisor that verifies them and sends them to the DARPA scoring server. The GUI is realized as a custom rqt plugin and a ROS node that visualizes the individual detections using the interactive marker server in rviz. As the detections are transmitted to the base station with their image, the rqt plugin presents this image to the human supervisor. The human supervisor may iterate through the detections, confirm or reject them, change any parameter of the detection, and finally he may send the detection to the DARPA scoring server. The rviz visualizes the individual robots, their path, the detections, and also the strength of the WiFi signal emitted by the cell-phone artifact and the CO_2 levels essential to detect the gas artifact that is measured by the robots along their path. The WiFi and CO_2 levels are visualized as color-coded point clouds along the

robot path. The rqt plugin allows switching the visualization of the individual elements for each of the robots in rviz on and off. Further, the detections visualized as interactive markers are color- and shape-coded based on their status, e.g., detections confirmed and rejected by DARPA are green and red spheres respectively, detections rejected by the human supervisor are small grey cubes. This allows the human supervisor to recognize the state of the detections immediately. Last but not least, the human supervisor is able to manipulate the spatial positions of the detections, and thus manually correct for the localization drift of the robots prior to sending the artifact positions to the DARPA scoring server.

5.3 Conclusion

Our team developed a multi-robot robust system that can operate in various extreme environments far behind domains defined by DARPA Subterranean Challenge. From the small team of people with old robots in the Tunnel circuit (see Fig. 5.13) we grew up to the relatively big team with the newest state-of-the-art robots and sensors (see Fig. 5.14). In the beginning, we were focused on a single-agent robust solution. During the competition, we start to build the fully autonomous multi-robot system on the robust base solution we already developed. As it can be seen in Table 5.4 we performed well. We have been 3rd in the first two rounds which convinced DARPA to give us additional funding that helps us to get new hardware. The improvement of the autonomy of our system is confirmed by the 2nd place in the virtual challenge, where the whole system was fully autonomous.

Figure 5.13: CTU-CRAS-NORLAB (formerly CTU-CRAS) team at Tunnel circuit.

Round	Position	Points	Price
Tunnel circuit (system)	3rd	10	$200,000
Urban circuit (system)	3rd	10	$500,000
Final circuit (system)	6th	7	
Cave circuit (virtual)	6th	90	
Final circuit (virtual)	2nd	215	$500,000

Table 5.4: CTU-CRAS-NORLAB positions at SubT Challenge.

Figure 5.14: CTU-CRAS-NORLAB team with the robots before the final event.

5.3.1 Lessons learned

Participation in the DARPA Subterranean Challenge was one of the best parts of my doctoral study. It allowed us to join the theoretical research with the practical deployment of the system. It was a pleasure to see how our goals and system developed during the whole competition, which took three years. In the first experiments, we focused on single robot control. We were still developing the system of communication, and we were far from multi-robot coordination and autonomy as well.

When the robot has to operate in extreme conditions for one hour, it rarely happens that the robot is still alive after the end of the mission. The motivation for winning the competition pushes the robots to their limits which is another reason for the failure. The common fail cases were stuck or dropped the track, a reboot of the system due to insufficient power source, big heat, or dropped water into the robot. We also had software failures during the missions. We learned from the fails, and our robots started to be robust enough to be able to explore for a longer time. We implemented a restarting procedure that allows the robot to connect to the system even after reboots during the mission.

From the operator's point of view, we made a vast improvement. The first detection GUI had only a few functions. We were able only to see the detected artifacts and choose whether to send them to the DARPA server

or not. The last versions allowed us to change the positions, classes, display different images from the same artifact, adjust the brightness of the pictures, filter all detection from some areas, and so on. That is all in an intuitive way. The key feature in control GUI was a key binding we implemented which saved a lot of time for the operator.

Another important thing during the mission was to drill the deployment procedure. Setting the system with multiple computers and robots can take a lot of time and stress. At the final rounds, we were able to set the system quickly, which allowed us to check the whole system before starting the mission.

There is also the drawback of the development. We always wanted to push our system towards being better and having new features. Unfortunately, every new update of the part of the system could potentially destroy something different, and the testing of the system is really time-consuming. Last week's changes could not be tested well, and it can cause problems during the actual missions. Unfortunately, adding the new features made our system less robust, which cost us points in the final event. In the ideal case, it would be best to freeze new updates for the last few weeks to test the system and debug all the parts.

Chapter 6

Conclusion and future work

This dissertation thesis discusses several problems in search-and-rescue robotics. The research behind this thesis was motivated by real problems we dealt with during our projects. The absence of dense and accurate data is a common issue in many robotic tasks, not only in search-and-rescue scenarios.

Autonomous robots need to have dense and accurate information about terrain in their surroundings. We proposed the approach to actively control the newly coming solid-state lidar, which allows the users to control the directions that will be measured. Our approach simultaneously predicts the dense 3D map from sparse depth measurements and actively controls the sensor with immense action space. The rays are cast to obtain the most information for the reconstruction network. Prediction accuracy is improved by iteratively learning the network and casting the rays in the directions defined by reconstruction confidence. In the case of flexible terrain (e.g., tall vegetation), the output of the reconstruction network does not provide relevant information for terrain traversal methods. Therefore we extended the prediction network input by segmentation features from images, and we enriched labels by robot poses. Using the robot poses as labels, we can predict the supporting terrain that will support the robot body during traversal. Our novel KKT-loss allows us to optimize the reconstruction network parameters to predict terrain that supports the robot body, but it is not in collision. The loss is defined as a distance from the satisfaction of Karush-Kuhn-Tucker conditions. One of the crucial fail cases is the presence of dense smoke that blinds the camera and spoils the laser measurements. Therefore we presented a combined hardware and software solution that reconstructs the nearest surrounding of the robot using our custom-built touch sensors and robot pose.

One of the main topics presented within this thesis is an active perception problem. Besides above mentioned active 3D reconstruction method for terrain reconstruction, we presented an end-to-end learnable method for active victim segmentation from multi-modal data. Victim detection and segmentation are the main problems in search-and-rescue missions. We proposed a method that actively controls the sensor to obtain the thermal information that will enrich the RGBD modalities obtained by other sensors. We implemented the neural network, which predicts the gain we get by measuring the temperature of given voxels. This information serves as an input for our proposed guided

Q-learning method. Our method controls the narrow field-of-view thermal sensor to increase the accuracy of the segmented victims.

Another topic discussed within this thesis is learning of policy for traversing terrain. The proposed method autonomously controls the flippers to allow the robot to traverse a complex terrain. Classical reinforcement learning techniques need to evaluate the immense number of policies that have no guarantees of safety. We have extended the relative entropy policy search by the constraints on safety that have to be satisfied during the learning procedure. These additional safety constraints prevent destroying the robot and lead to faster policy convergence. The robot can safely learn to traverse a previously unknown obstacle after only a few training iterations.

We were able to test and deploy our methods during the DARPA Subterranean Challenge. Chapter 5 concludes our participation and results in this challenge. With our team CTU-CRAS-NORLAB, we developed and deployed a multi-robotic autonomous system for searching objects in search-and-rescue missions. We were also able to take place on the podium in the competition of the word best researchers.

The methods presented within this dissertation thesis contributed to the robotics research field, in particular to the search-and-rescue robotics. All the methods are well tested and are useable in real applications. Moreover, the theoretical background is transferable to other research fields as well. I see the biggest potential in the application of our novel KKT-loss. This loss function can be used to solve different non-convex and constrained optimization problems, without the necessity of finding explicitly its optimal solution.

Appendix A

Citations of author's publications

- Vojtěch Šalanský, Karel Zimmermann, Tomáš Petříček, and Tomáš Svoboda. "Pose Consistency KKT-Loss for Weakly Supervised Learning of Robot-Terrain Interaction Model". In: *IEEE Robotics and Automation Letters* 6 (July 2021), pp. 5477–5484
  - Maximilian Stolzle, Takahiro Miki, Levin Gerdes, Martin Azkarate, and Marco Hutter. "Reconstructing occluded Elevation Information in Terrain Maps with Self-supervised Learning". In: *IEEE Robotics and Automation Letters* (2022)
  - Ruslan Agishev, Tomas Petricek, and Karel Zimmermann. "Trajectory Optimization using Learned Robot-Terrain Interaction Model in Exploration of Large Subterranean Environments". In: *IEEE Robotics and Automation Letters* (2022)
- Tomáš Petříček, Vojtěch Šalanský, Karel Zimmermann, and Tomáš Svoboda. "Simultaneous exploration and segmentation for search and rescue". In: *Journal of Field Robotics* 36.4 (2019), pp. 696–709
  - Adrián Banuls, Anthony Mandow, Ricardo Vázquez-Martín, Jesús Morales, and Alfonso García-Cerezo. "Object detection from thermal infrared and visible light cameras in search and rescue scenes". In: *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2020, pp. 380–386
  - Jesús Morales, Ricardo Vázquez-Martín, Anthony Mandow, David Morilla-Cabello, and Alfonso García-Cerezo. "The UMA-SAR Dataset: Multimodal data collection from a ground vehicle during outdoor disaster response training exercises". In: *The International Journal of Robotics Research* 40.6-7 (2021), pp. 835–847
  - Songqun Gao, Yulong Ding, and Ben M Chen. "A Frontier-Based Coverage Path Planning Algorithm for Robot Exploration in Unknown Environment". In: *2020 39th Chinese Control Conference (CCC)*. IEEE. 2020, pp. 3920–3925

- Karel Zimmermann, Tomáš Petříček, Vojtech Šalanský, and Tomáš Svoboda. "Learning for Active 3D Mapping". In: *2017 IEEE Int. Conf. on Comput. Vision (ICCV)*. 2017, pp. 1548–1556
  - Xinjing Cheng, Peng Wang, and Ruigang Yang. "Depth estimation via affinity learned with convolutional spatial propagation network". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 103–119
  - Xinjing Cheng, Peng Wang, and Ruigang Yang. "Learning depth with convolutional spatial propagation network". In: *IEEE transactions on pattern analysis and machine intelligence* 42.10 (2019), pp. 2361–2379
  - Philipp Rosenberger, Martin Holder, Marina Zirulnik, and Hermann Winner. "Analysis of real world sensor behavior for rising fidelity of physically based lidar sensor models". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 611–616
  - Luis Roldao, Raoul De Charette, and Anne Verroust-Blondet. "3D semantic scene completion: a survey". In: *arXiv preprint arXiv:2103.07466* (2021)
  - Luis Roldão, Raoul de Charette, and Anne Verroust-Blondet. "Lmscnet: Lightweight multiscale 3d semantic completion". In: *2020 International Conference on 3D Vision (3DV)*. IEEE. 2020, pp. 111–119
  - Mohammed Bennamoun, Yulan Guo, Federico Tombari, Kamal Youcef-Toumi, and Ko Nishino. "Guest editors' introduction to the special issue on rgb-d vision: Methods and applications". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10 (2020), pp. 2329–2332
  - Ruslan Agishev, Tomas Petricek, and Karel Zimmermann. "Trajectory Optimization using Learned Robot-Terrain Interaction Model in Exploration of Large Subterranean Environments". In: *IEEE Robotics and Automation Letters* (2022)
  - Maximilian Stolzle, Takahiro Miki, Levin Gerdes, Martin Azkarate, and Marco Hutter. "Reconstructing occluded Elevation Information in Terrain Maps with Self-supervised Learning". In: *IEEE Robotics and Automation Letters* (2022)
- Martin Pecka, Vojtěch Šalanský, Karel Zimmermann, and Tomáš Svoboda. "Autonomous flipper control with safety constraints". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2889–2894
  - Maxim Sokolov, Ilya Afanasyev, Roman Lavrenov, Artur Sagitov, Leysan Sabirova, and Evgeni Magid. "Modelling a crawler-type UGV for urban search and rescue in Gazebo environment". In: *Artificial Life and Robotics (ICAROB 2017), International Conference on*. 2017, pp. 360–362

- Maxim Sokolov, Ilya Afanasyev, Alexandr Klimchik, and Nikolaos Mavridis. "HyperNEAT-based flipper control for a crawler robot motion in 3D simulation environment". In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2017, pp. 2652–2656
- Martin Pecka, Karel Zimmermann, and Tomáš Svoboda. "Fast simulation of vehicles with non-deformable tracks". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 6414–6419
- Ilya Moskvin, Roman Lavrenov, Evgeni Magid, and Mikhail Svinin. "Modelling a crawler robot using wheels as pseudo-tracks: model complexity vs performance". In: *2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA)*. IEEE. 2020, pp. 1–5
- Vladimír Kubelka, Michal Reinstein, and Tomáš Svoboda. "Tracked robot odometry for obstacle traversal in sensory deprived environment". In: *IEEE/ASME Transactions on Mechatronics* 24.6 (2019), pp. 2745–2755
- Giuseppe Paolo, Lei Tai, and Ming Liu. "Towards continuous control of flippers for a multi-terrain robot using deep reinforcement learning". In: *arXiv preprint arXiv:1709.08430* (2017)
- Andrei Mitriakov, Panagiotis Papadakis, Serge Garlatti, et al. "Staircase traversal via reinforcement learning for active reconfiguration of assistive robots". In: *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE. 2020, pp. 1–8
- Andrei Mitriakov, Panagiotis Papadakis, Serge Garlatti, et al. "Staircase negotiation learning for articulated tracked robots with varying degrees of freedom". In: *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2020, pp. 394–400
- Yijun Yuan, Letong Wang, and Sören Schwertfeger. "Configuration-space flipper planning for rescue robots". In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2019, pp. 37–42
- Yijun Yuan, Qingwen Xu, and Sören Schwertfeger. "Configuration-Space Flipper Planning on 3D Terrain". In: *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2020, pp. 318–325
- George Mason. "Safe Reinforcement Learning Using Formally Verified Abstract Policies". PhD thesis. University of York, 2018
- Andrei Mitriakov, Panagiotis Papadakis, Jérôme Kerdreux, and Serge Garlatti. "Reinforcement Learning Based, Staircase Negotiation Learning: Simulation and Transfer to Reality for Articulated Tracked Robots". In: *IEEE Robotics & Automation Magazine* 28.4 (2021), pp. 10–20

- Afanasev Ilya Mikhajlovich, Sokolov Maksim Alekseevich, Lavrenov Roman Olegovich, Sagitov Artur Gazizovich, Sabirova Leysan Nailevna, Magid Evgeni, et al. *Modelling a crawler-type UGV for urban search and rescue in Gazebo environment.* 2017
- Moskvin Ilya Sergeevich, Svinin Mikhail, Magid Evgeni Arkadevich, Lavrenov Roman Olegovich, et al. *Modelling a Crawler Robot Using Wheels as Pseudo-Tracks: Model Complexity vs Performance.* 2020

- Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanský, et al. "DARPA Subterranean Challenge: Multi-robotic Exploration of Underground Environments". In: *Modelling and Simulation for Autonomous Systems.* 2020, pp. 274–290. ISBN: 978-3-030-43890-6
  - Jorge Peña Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, et al. "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision". In: *Ieee Access* 8 (2020), pp. 191617–191643
  - Matěj Petrlík, Tomáš Báča, Daniel Heřt, Matouš Vrba, Tomáš Krajník, and Martin Saska. "A robust UAV system for operations in a constrained environment". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2169–2176
  - Tomas Baca, Matej Petrlik, Matous Vrba, Vojtech Spurny, Robert Penicka, Daniel Hert, et al. "The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles". In: *Journal of Intelligent & Robotic Systems* 102.1 (2021), pp. 1–28
  - Shenghai Yuan, Han Wang, and Lihua Xie. "Survey on localization systems and algorithms for unmanned systems". In: *Unmanned Systems* 9.02 (2021), pp. 129–163
  - Petr Štibinger, George Broughton, Filip Majer, Zdeněk Rozsypálek, Anthony Wang, Kshitij Jindal, et al. "Mobile manipulator for autonomous localization, grasping and precise placement of construction material in a semi-structured environment". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2595–2602
  - Vojtech Spurny, Vaclav Pritzl, Viktor Walter, Matej Petrlik, Tomas Baca, Petr Stepan, et al. "Autonomous firefighting inside buildings by an unmanned aerial vehicle". In: *IEEE Access* 9 (2021), pp. 15872–15890
  - Jorge Pena Queralta, Li Qingqing, Fabrizio Schiano, and Tomi Westerlund. "VIO-UWB-based collaborative localization and dense scene reconstruction within heterogeneous multi-robot systems". In: *arXiv preprint arXiv:2011.00830* (2020)
  - Daniel S Drew. "Multi-agent systems for search and rescue applications". In: *Current Robotics Reports* 2.2 (2021), pp. 189–200

- Anushri Dixit, Mohamadreza Ahmadi, and Joel W Burdick. "Risk-sensitive motion planning using entropic value-at-risk". In: *2021 European Control Conference (ECC)*. IEEE. 2021, pp. 1726–1732
- Vít Krátký, Pavel Petráček, Tomáš Báča, and Martin Saska. "An autonomous unmanned aerial vehicle system for fast exploration of large complex indoor environments". In: *Journal of field robotics* 38.8 (2021), pp. 1036–1058
- Michael T Ohradzansky, Eugene R Rush, Danny G Riley, Andrew B Mills, Shakeeb Ahmad, Steve McGuire, et al. "Multi-agent autonomy: Advancements and challenges in subterranean exploration". In: *arXiv preprint arXiv:2110.04390* (2021)
- Chris Yu Hsuan Lee, Graeme Best, and Geoffrey A Hollinger. "Optimal sequential stochastic deployment of multiple passenger robots". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 8934–8940
- Tomas Baca, Petr Stibinger, Daniela Doubravova, Daniel Turecek, Jaroslav Solc, Jan Rusnak, et al. "Gamma radiation source localization for micro aerial vehicles with a miniature single-detector compton event camera". In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2021, pp. 338–346
- Jan Bayer and Jan Faigl. "Handheld localization device for indoor environments". In: *2020 4th International Conference on Automation, Control and Robots (ICACR)*. IEEE. 2020, pp. 60–64
- Kshitij Goel, Wennie Tabib, and Nathan Michael. "Rapid and high-fidelity subsurface exploration with multiple aerial robots". In: *International Symposium on Experimental Robotics*. Springer. 2020, pp. 436–448
- Li Qingqing, Yu Xianjia, Jorge Peña Queralta, and Tomi Westerlund. "Adaptive lidar scan frame integration: Tracking known mavs in 3d point clouds". In: *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE. 2021, pp. 1079–1086
- Danny G Riley Ii and Eric W Frew. "Assessment of Coordinated Heterogeneous Exploration of Complex Environments". In: *2021 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE. 2021, pp. 138–143
- Thulio Amorim, Tiago Nascimento, Pavel Petracek, Giulia De Masi, Eliseo Ferrante, and Martin Saska. "Self-organized UAV flocking based on proximal control". In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2021, pp. 1374–1382
- Jan Bayer and Jan Faigl. "Decentralized task allocation in multi-robot exploration with position sharing only". In: *International Symposium on Swarm Behavior and Bio-Inspired Robotics (SWARM)*. 2021

- Tomáš Krajník, Tomáš Vintr, George Broughton, Filip Majer, Tomáš Rouček, Jiří Ulrich, et al. "CHRONOROBOTICS: Representing the structure of time for service robots". In: *Proceedings of the 2020 4th International Symposium on Computer Science and Intelligent Control*. 2020, pp. 1–8
- Michael T Ohradzansky and J Sean Humbert. "Lidar-Based Navigation of Subterranean Environments Using Bio-Inspired Wide-Field Integration of Nearness". In: *Sensors* 22.3 (2022), p. 849
- Mihir Kulkarni, Mihir Dharmadhikari, Marco Tranzatto, Samuel Zimmermann, Victor Reijgwart, Paolo De Petris, et al. "Autonomous Teamed Exploration of Subterranean Environments using Legged and Aerial Robots". In: *arXiv preprint arXiv:2111.06482* (2021)
- Mihir Dharmadhikari, Huan Nguyen, Frank Mascarich, Nikhil Khedekar, and Kostas Alexis. "Autonomous Cave Exploration using Aerial Robots". In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2021, pp. 942–949
- Yun Chang. "Robust and Lightweight Localization and Dense Mapping for Multi-Robot Systems". PhD thesis. Massachusetts Institute of Technology, 2021
- Jiří Kubík, Petr Čížek, Rudolf Szadkowski, and Jan Faigl. "Experimental Leg Inverse Dynamics Learning of Multi-legged Walking Robot". In: *International Conference on Modelling and Simulation for Autonomous Systems*. Springer. 2020, pp. 154–168
- George Broughton, Pavel Linder, Tomáš Rouček, Tomáš Vintr, and Tomáš Krajník. "Robust Image Alignment for Outdoor Teach-and-Repeat Navigation". In: *2021 European Conference on Mobile Robots (ECMR)*. IEEE, pp. 1–6
- Chris Yu Hsuan Lee. *Deployment Planning for Multiple Marsupial Robots*. Master thesis, Oregon State University. 2021
- Andriy Dmytruk. *A perception-aware NMPC for collision avoidance and control of a multi-rotor UAV with visual localization constraints*. Bachelor Thesis, Ukrainian Catholic University. 2021
- Adam Norton, Peter Gavriel, Brendan Donoghue, and Holly Yanco. "Test Methods to Evaluate Mapping Capabilities of Small Unmanned Aerial Systems in Constrained Indoor and Subterranean Environments". In: *2021 IEEE International Symposium on Technologies for Homeland Security (HST)*. IEEE. 2021, pp. 1–8
- Carlos Azevedo, António Matos, Pedro U Lima, and Jose Avendaño. "Petri Net Toolbox for Multi-Robot Planning under Uncertainty". In: *Applied Sciences* 11.24 (2021), p. 12087
- Martin Zoula, Miloš Prágr, and Jan Faigl. "On Building Communication Maps in Subterranean Environments". In: *International Conference on Modelling and Simulation for Autonomous Systems*. Springer. 2020, pp. 15–28

- Akshay Prabhakar Shetty. "Trajectory planning under motion and sensing uncertainties: reachability analysis and connectivity maintenance". PhD thesis. University of Illinois at Urbana-Champaign, 2021
- Shaza Kaoud Abdelaziz et al. "Vision-aided Inertial System for Near-real-time Positioning and Navigation of Unmanned Ground Survey Vehicles in GNSS-denied Environments". PhD thesis
- Bianca Bendris and Julián Cayero Becerra. "Design and Experimental Evaluation of an Aerial Solution for Visual Inspection of Tunnel-like Infrastructures". In: *Remote Sensing* 14.1 (2022), p. 195
- Mark Selden, Jason Zhou, Felipe Campos, Nathan Lambert, Daniel Drew, and Kristofer SJ Pister. "BotNet: A Simulator for Studying the Effects of Accurate Communication Models on Multi-agent and Swarm Control". In: *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE. 2021, pp. 101–109
- Jorge Peña Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, et al. "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision". In: 8 (2020), pp. 191617–191643

Appendix B

Bibliography

[1] Vojtěch Šalanský, Karel Zimmermann, Tomáš Petříček, and Tomáš Svoboda. "Pose Consistency KKT-Loss for Weakly Supervised Learning of Robot-Terrain Interaction Model". In: *IEEE Robotics and Automation Letters* 6 (July 2021), pp. 5477–5484.

[2] Tomáš Petříček, Vojtěch Šalanský, Karel Zimmermann, and Tomáš Svoboda. "Simultaneous exploration and segmentation for search and rescue". In: *Journal of Field Robotics* 36.4 (2019), pp. 696–709.

[3] Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanský, Teymur Azayev, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Vojtěch Spurný, Vít Krátký, Pavel Petráček, Dominic Baril, Maxime Vaidis, Vladimír Kubelka, François Pomerleau, Jan Faigl, Karel Zimmermann, Martin Saska, Tomáš Svoboda, and Tomáš Krajník. *System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA Subterranean Challenge*. 2021. arXiv: `2110.05911 [cs.RO]`.

[4] Karel Zimmermann, Tomáš Petříček, Vojtech Šalanský, and Tomáš Svoboda. "Learning for Active 3D Mapping". In: *2017 IEEE Int. Conf. on Comput. Vision (ICCV)*. 2017, pp. 1548–1556.

[5] Martin Pecka, Vojtěch Šalanský, Karel Zimmermann, and Tomáš Svoboda. "Autonomous flipper control with safety constraints". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2889–2894.

[6] Vojtěch Šalanský, Vladimír Kubelka, Karel Zimmermann, Michal Reinstein, and Tomáš Svoboda. "Touching without vision: terrain perception in sensory deprived environments". In: *21st Computer Vision Winter Workshop (pp. 1-9)*. 2016.

[7] Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanský, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Vojěch Spurný, François Pomerleau, Vladimír Kubelka, Jan Faigl, Karel Zimmermann, Martin Saska, Tomáš Svoboda, and Tomáš Krajník. "DARPA Subterranean Challenge: Multi-robotic Exploration of Under-

ground Environments". In: *Modelling and Simulation for Autonomous Systems*. 2020, pp. 274–290. ISBN: 978-3-030-43890-6.

[8] Tomas Petricek, Vojtěch Šalanský, Karel Zimmermann, and Tomas Svoboda. *Simultaneous Exploration and Segmentation with Incomplete Data*. Workshop on Action and Anticipation for Visual Learning, ECCV. 2016.

[9] David Silver, J. Andrew Bagnell, and Anthony Stentz. "Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain". In: *The Int. J. of Robotics Research* 29.12 (2010), pp. 1565–1592.

[10] Martin Pecka, Karel Zimmermann, Michal Reinstein, and Tomas Svoboda. "Controlling Robot Morphology From Incomplete Measurements". In: *IEEE Transactions on Industrial Electronics* 64.2 (2017), pp. 1773–1782.

[11] Philipp Krüsi, Paul Furgale, Michael Bosse, and Roland Siegwart. "Driving on Point Clouds: Motion Planning, Trajectory Optimization, and Terrain Assessment in Generic Nonplanar Environments". In: *Journal of Field Robotics* 34.5 (2017), pp. 940–984.

[12] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. "Large-scale cost function learning for path planning using deep inverse reinforcement learning". In: *The International Journal of Robotics Research* 36.10 (2017), pp. 1073–1087.

[13] Vivekanandan Suryamurthy, Vignesh Sushrutha Raghavan, Arturo Laurenzi, Nikos G. Tsagarakis, and Dimitrios Kanoulas. "Terrain Segmentation and Roughness Estimation using RGB Data: Path Planning Application on the CENTAURO Robot". In: *2019 IEEE-RAS 19th Int. Conf. on Humanoid Robots (Humanoids)*. 2019, pp. 1–8.

[14] Lorenz Wellhausen, Alexey Dosovitskiy, René Ranftl, Krzysztof Walas, Cesar Cadena, and Marco Hutter. "Where Should I Walk? Predicting Terrain Properties From Images Via Self-Supervised Learning". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1509–1516.

[15] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[16] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. "Multi-view Convolutional Neural Networks for 3D Shape Recognition". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 945–953.

[17] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. "Volumetric and Multi-view CNNs for Object Classification on 3D Data". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 5648–5656.

[18] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction". In: *ECCV 2016: 14th European Conference on Computer Vision*. 2016, pp. 628–644. ISBN: 978-3-319-46484-8.

[19] Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. "Structure Recovery by Part Assembly". In: *ACM Trans. Graph.* 31.6 (Nov. 2012), 180:1–180:11. ISSN: 0730-0301.

[20] Minhyuk Sung, Vladimir G. Kim, Roland Angst, and Leonidas Guibas. "Data-driven Structural Priors for Shape Completion". In: *ACM Trans. Graph.* 34.6 (Oct. 2015), 175:1–175:11. ISSN: 0730-0301.

[21] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem. "Completing 3D object shape from one depth image". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 2484–2493.

[22] Aron Monszpart, Nicolas Mellado, Gabriel J. Brostow, and Niloy J. Mitra. "RAPter: Rebuilding Man-made Scenes with Regular Arrangements of Planes". In: *ACM Trans. Graph.* 34.4 (July 2015), 103:1–103:12. ISSN: 0730-0301.

[23] B. Zheng, Y. Zhao, J. C. Yu, K. Ikeuchi, and S. C. Zhu. "Beyond Point Clouds: Scene Understanding by Reasoning Geometry and Physics". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3127–3134.

[24] D. T. Nguyen, B. S. Hua, M. K. Tran, Q. H. Pham, and S. K. Yeung. "A Field Model for Repairing 3D Shapes". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 5676–5684.

[25] M. Firman, O. M. Aodha, S. Julier, and G. J. Brostow. "Structured Prediction of Unobserved Voxels from a Single Depth Image". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 5431–5440.

[26] Stefan Fabian, Stefan Kohlbrecher, and Oskar Von Stryk. "Pose Prediction for Mobile Ground Robots in Uneven Terrain Based on Difference of Heightmaps". In: *2020 IEEE Int. Symp. on Safety, Security, and Rescue Robotics (SSRR)*. 2020, pp. 49–56.

[27] A. Angelova, L. Matthies, D. Helmick, G. Sibley, and P. Perona. "Learning to predict slip for ground robots". In: *2006 IEEE International Conference on Robotics and Automation*. 2006, pp. 3324–3331.

[28] Igor Bogoslavskyi, Olga Vysotska, Jacopo Serafin, Giorgio Grisetti, and Cyrill Stachniss. "Efficient traversability analysis for mobile robots using the Kinect sensor". In: *2013 European Conferenc on Mobile Robots*. 2013, pp. 158–163.

[29] Jiajun Gu, Qixin Cao, and Yi Huang. “Rapid Traversability Assessment in 2.5D Grid-based Map on Rough Terrain”. In: *International Journal of Advanced Robotic Systems* 5.4 (2008), pp. 389–394. ISSN: 1729-8806.

[30] Michael Brunner, Torsten Fiolka, Dirk Schulz, and Christopher M. Schlick. “Design and comparative evaluation of an iterative contact point estimation method for static stability estimation of mobile actively reconfigurable robots”. In: *Robotics and Autonomous Systems* 63 (2015), pp. 89–107. ISSN: 0921-8890.

[31] R. Omar Chavez-Garcia, Jérôme Guzzi, Luca M. Gambardella, and Alessandro Giusti. “Learning Ground Traversability From Simulations”. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1695–1702.

[32] Tomáš Nouza. *Safe Adaptive Traversability Learning for Mobile Robots*. Master thesis, Czech Technical University in Prague. 2014. URL: `http://hdl.handle.net/10467/24431`.

[33] Francois Pomerleau, Philipp Krüsi, Francis Colas, Paul Furgale, and Roland Siegwart. “Long-term 3D map maintenance in dynamic environments”. In: *2014 IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2014, pp. 3712–3719.

[34] Wouter Van Gansbeke, Davy Neven, Bert De Brabandere, and Luc Van Gool. “Sparse and Noisy LiDAR Completion with RGB Guidance and Uncertainty”. In: *2019 16th International Conference on Machine Vision Applications (MVA)*. 2019, pp. 1–6.

[35] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. “Self-Supervised Sparse-to-Dense: Self-Supervised Depth Completion from LiDAR and Monocular Camera”. In: *2019 Int. Conf. on Robot. and Autom. (ICRA)*. 2019, pp. 3288–3295.

[36] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. “Differentiable Convex Optimization Layers”. In: Curran Associates Inc., 2019.

[37] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. “End-To-End Interpretable Neural Motion Planner”. In: *2019 IEEE/CVF Conf. on Comput. Vision and Pattern Recognit. (CVPR)*. 2019, pp. 8652–8661.

[38] Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. “Inverse Reinforcement Learning with Locally Consistent Reward Functions”. In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015.

[39] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. “Multiple instance learning: A survey of problem characteristics and applications”. In: *Pattern Recognition* 77 (2018), pp. 329–353. ISSN: 0031-3203.

[40] Sebastian Ruder. *An Overview of Multi-Task Learning in Deep Neural Networks*. arXiv:1706.05098 [cs.LG]. 2017. arXiv: 1706.05098 [cs.LG].

[41] Zilong Huang, Xinggang Wang, Jiasi Wang, Wenyu Liu, and Jingdong Wang. "Weakly-Supervised Semantic Segmentation Network with Deep Seeded Region Growing". In: *2018 IEEE/CVF Conf. on Comput. Vision and Pattern Recognit.* 2018, pp. 7014–7023.

[42] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995.

[43] Jan Brabec. *Automated Camera Calibration from Laser Scanning Data in Natural Environments*. Bachelor thesis, Czech Technical University in Prague, 2014. 2014. URL: http://hdl.handle.net/10467/24152.

[44] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. "Semantic understanding of scenes through the ade20k dataset". In: *International Journal of Computer Vision* 127.3 (2019), pp. 302–321.

[45] Ivana Kruijff-Korbayová, Francis Colas, Mario Gianni, Fiora Pirri, Joachim de Greeff, Koen V. Hindriks, Mark A. Neerincx, Petter Ögren, Tomás Svoboda, and Rainer Worst. "TRADR Project: Long-Term Human-Robot Teaming for Robot Assisted Disaster Response". In: *KI* 29.2 (2015), pp. 193–201.

[46] Lauro Ojeda, Johann Borenstein, Gary Witus, and Robert Karlsen. "Terrain characterization and classification with a mobile robot". In: *Journal of Field Robotics* 23.2 (2006), pp. 103–122.

[47] Michal Reinstein, V. Kubelka, and Karel Zimmermann. "Terrain Adaptive Odometer for Mobile Skid-steer Robots". In: *Proc. IEEE Int Robotics and Automation (ICRA) Conf.* 2013, pp. 4691–4696.

[48] Juhana Ahtiainen, Thierry Peynot, Jari Saarinen, and Steven Scheding. "Augmenting traversability maps with ultra-wideband radar to enhance obstacle detection in vegetated environments". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 5148–5155.

[49] J.C. Sullivan, B. Mitchinson, M.J. Pearson, M. Evans, N.F. Lepora, C.W. Fox, C. Melhuish, and T.J. Prescott. "Tactile Discrimination Using Active Whisker Sensors". In: *IEEE Sensors Journal* 12.2 (2012), pp. 350–362.

[50] Martin J. Pearson, Charles Fox, J. Charles Sullivan, Tony J. Prescott, Tony Pipe, and Ben Mitchinson. "Simultaneous localisation and mapping on a multi-degree of freedom biomimetic whiskered robot". In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 586–592.

[51] Kazunori Ohno, Shouich Morimura, Satoshi Tadokoro, Eiji Koyanagi, and Tomoaki Yoshida. "Semi-autonomous control system of rescue crawler robot having flippers for getting Over unknown-Steps". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 3012–3018.

[52] Mats Bjorkman, Yasemin Bekiroglu, Virgile Hogman, and Danica Kragic. "Enhancing visual perception of shape through tactile glances". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 3180–3186.

[53] Anthony O'Hagan and JFC Kingman. "Curve fitting and optimal design for prediction". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1978), pp. 1–42.

[54] Christopher KI Williams and Carl Edward Rasmussen. "Gaussian processes for regression". In: *Advances in Neural Information Processing Systems 8*. The MIT Press, 1996, pp. 514–520.

[55] Martin Meier, Matthias Schopfer, Robert Haschke, and Helge Ritter. "A Probabilistic Approach to Tactile Shape Reconstruction". In: *IEEE Transactions on Robotics* 27.3 (2011), pp. 630–635.

[56] Vojtěch Šalanský. *Contact terrain exploration for mobile robots*. Master thesis, Czech Technical University in Prague. 2015. URL: `http://hdl.handle.net/10467/62094`.

[57] Stuart Geman and Donald Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.6 (1984), pp. 721–741.

[58] C. E. Rasmussen and H. Nickisch. *GPML Matlab Code*. URL: `http://www.gaussianprocess.org/gpml/code/matlab/doc/`.

[59] Karel Zimmermann, Petr Zuzánek, Michal Reinstein, Tomáš Petříček, and Václav Hlaváč. "Adaptive traversability of partially occluded obstacles". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 3959–3964.

[60] Heiko Seifa and Xiaolong Hub. "Autonomous Driving in the iCity—HD Maps as a Key Challenge of the Automotive Industry". In: *Autonomous Robots* 2.2 (2016), pp. 159–162.

[61] Evan Ackerman. "Quanergy Announces $250 Solid-State LIDAR for Cars, Robots, and More". In: *IEEE Spectrum*. 2016. URL: `http://spectrum.ieee.org/cars-that-think/transportation/sensors/quanergy-solid-state-lidar`.

[62] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. "Vision meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research* 32.11 (Sept. 2013), pp. 1231–1237.

[63] Dinesh Jayaraman and Kristen Grauman. "Look-Ahead Before You Leap: End-to-End Active Recognition by Forecasting the Effect of Motion". In: *ECCV 2016: 14th European Conference on Computer Vision*. 2016, pp. 489–505. ISBN: 978-3-319-46454-1.

[64] A. K. Mishra, Y. Aloimonos, L. F. Cheong, and A. Kassim. "Active Visual Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.4 (2012), pp. 639–653. ISSN: 0162-8828.

[65] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. "3D ShapeNets: A deep representation for volumetric shapes". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1912–1920.

[66] Ruben Martinez-Cantin, Nando de Freitas, Eric Brochu, José Castellanos, and Arnaud Doucet. "A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot". In: *Autonomous Robots* 27.2 (2009), pp. 93–103. ISSN: 1573-7527.

[67] Alexander Andreopoulos, Stephan Hasler, Heiko Wersing, Herbert Janssen, John K. Tsotsos, and Edgar Körner. "Active 3D object localization using a humanoid robot". In: *IEEE Transactions on Robotics* 27.1 (2011), pp. 47–64. ISSN: 15523098.

[68] Andrea Vedaldi and Karel Lenc. "MatConvNet: Convolutional Neural Networks for MATLAB". In: *Proceedings of the 23rd ACM International Conference on Multimedia*. MM '15. ACM, 2015, pp. 689–692. ISBN: 978-1-4503-3459-4.

[69] Robin R. Murphy. *Disaster Robotics*. The MIT Press, 2014. ISBN: 0262027356, 9780262027359.

[70] Geert-Jan M Kruijff, Ivana Kruijff-Korbayová, Shanker Keshavdas, Benoit Larochelle, Miroslav Janíček, Francis Colas, Ming Liu, François Pomerleau, Roland Siegwart, Mark Neerincx, et al. "Designing, developing, and deploying systems to support human-robot teams in disaster response". In: *Advanced Robotics* 28.23 (2014), pp. 1547–1570.

[71] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances In Neural Information Processing Systems* (2012), pp. 1–9. ISSN: 10495258.

[72] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533.

[73] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440. ISBN: 9781467369640.

[74] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-End Training of Deep Visuomotor Policies". In: *Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[75] Raymond Sheh, Sören Schwertfeger, and Arnoud Visser. "16 Years of RoboCup Rescue". In: *Künstliche Intelligenz* 30.3–4 (2016), pp. 267–277.

[76] Ruzena Bajcsy, Yiannis Aloimonos, and John K. Tsotsos. "Revisiting active perception". In: *Autonomous Robots* (2017). ISSN: 1573-7527.

[77] D. Wilkes and J. K. Tsotsos. "Active object recognition". In: *1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1992, pp. 136–141.

[78] Zhaoyin Jia, Yao-Jen Chang, and Tsuhan Chen. "A general boosting-based framework for active object recognition". In: *British Machine Vision Conference*. 2010.

[79] Andreas Doumanoglou, Tae-Kyun Kim, Xiaowei Zhao, and Sotiris Malassiotis. "Active Random Forests: An Application to Autonomous Unfolding of Clothes". In: *ECCV*. 2014. ISBN: 978-3-319-10602-1.

[80] Edward Johns, Stefan Leutenegger, and Andrew J. Davison. "Pairwise Decomposition of Image Sequences for Active Multi-View Recognition". In: *CoRR* abs/1605.08359 (2016).

[81] Ksenia Shubina and John K. Tsotsos. "Visual search for an object in a 3D environment using a mobile robot". In: *Computer Vision and Image Understanding* 114.5 (2010), pp. 535–547. ISSN: 10773142.

[82] Yiming Ye and John K. Tsotsos. "Sensor Planning for 3D Object Search". In: *Computer Vision and Image Understanding* 73.2 (1999).

[83] Philipp Krähenbühl and Vladlen Koltun. "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials". In: *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2011, pp. 109–117.

[84] J. Yao, S. Fidler, and R. Urtasun. "Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 702–709.

[85] Cristina Palmero, Albert Clapés, Chris Bahnsen, Andreas Møgelmose, Thomas B. Moeslund, and Sergio Escalera. "Multi-modal RGB–Depth–Thermal Human Body Segmentation". In: *International Journal of Computer Vision* (2016), pp. 1–23. ISSN: 1573-1405.

[86] Sungil Choi, Seugryong Kim, Kihong Park, and Kwanghoon Sohn. "Multispectral Human Co-Segmentation via Joing Convolutional Neural Networks". In: *IEEE International Conferecen on Image Processing*. 2017.

[87] Yurii Nesterov. "A method of solving a convex programming problem with convergence rate $O(1/k^2)$". In: *Soviet Mathematics Doklady*. Vol. 27. 2. 1983, pp. 372–376.

[88] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. "On the importance of initialization and momentum in deep learning". In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. Vol. 28. 3. JMLR Workshop and Conference Proceedings, May 2013, pp. 1139–1147.

[89] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014). URL: `http://arxiv.org/abs/1409.1556`.

[90] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *International conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[91] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *Proceedings of the ACM International Conference on Multimedia*. MM '14. 2014, pp. 675–678. ISBN: 978-1-4503-3063-3.

[92] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. "Comparing ICP variants on real-world data sets". In: *Autonomous Robots* 34.3 (2013), pp. 133–148. ISSN: 0929-5593.

[93] J. Simanek, M. Reinstein, and V. Kubelka. "Evaluation of the EKF-Based Estimation Architectures for Data Fusion in Mobile Robots". In: *IEEE/ASME Transactions on Mechatronics* 20.2 (2015), pp. 985–990. ISSN: 1083-4435.

[94] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. AAAI Press, 2016, pp. 2094–2100.

[95] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. "Prioritized Experience Replay". In: *International Conference on Learning Representations*. 2016.

[96] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. "TensorFlow: A System for Large-Scale Machine Learning". In: *12th*

USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016, pp. 265–283. ISBN: 978-1-931971-33-1.

[97] Andras Kupcsik, Marc Peter Deisenroth, Jan Peters, Loh Ai Poh, Prahlad Vadakkepat, and Gerhard Neumann. "Model-based contextual policy search for data-efficient generalization of robot skills". In: *Artificial Intelligence* (2014).

[98] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. "Reinforcement Learning: A Survey". In: *Journal of Artificial Intelligence Research* 4 (1996), pp. 237–285.

[99] J Peters, S Vijayakumar, and S Schaal. "Reinforcement learning for humanoid robotics". In: *Proceedings of the third IEEE-RAS International Conference on Humanoid Robots*. 2003.

[100] Marc Deisenroth and Carl E. Rasmussen. "PILCO: A model-based and data-efficient approach to policy search". In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 465–472.

[101] Russ Tedrake, Zack Jackowski, Rick Cory, John Roberts, and Warren Hoburg. *Learning to Fly like a Bird*. Jan. 2009.

[102] Richard Primerano, David Wilkie, and William C. Regli. "A case study in system-level physics-based simulation of a biomimetic robot". In: *IEEE Transactions on Automation Science and Engineering* 8.3 (2011), pp. 664–671. ISSN: 15455955.

[103] S Ross and JA Bagnell. "Agnostic system identification for model-based reinforcement learning". In: *Proceedings of the 29th International Conference on Machine Learning*. 2012.

[104] Anayo K. Akametalu, Jaime F. Fisac, Jeremy H. Gillula, Shahab Kaynama, Melanie N. Zeilinger, and Claire J. Tomlin. "Reachability-based safe learning with Gaussian processes". In: *53rd IEEE Conference on Decision and Control*. 2014, pp. 1424–1431.

[105] M Pecka, K Zimmermann, and T Svoboda. "Safe Exploration for Reinforcement Learning in Real Unstructured Environments". In: *20th Computer Vision Winter Workshop*. 2015.

[106] Karel Zimmermann, Petr Zuzanek, Michal Reinstein, and Vaclav Hlavac. "Adaptive Traversability of unknown complex terrain with obstacles for mobile robots". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 5177–5182.

[107] Aksel Transeth, K.Y. Pettersen, and Pål Liljebäck. "A survey on snake robot modeling and locomotion". In: *Robotica* 27 (Dec. 2009), pp. 999–1015.

[108] Eiji Uchibe and Kenji Doya. "Constrained reinforcement learning from intrinsic and extrinsic rewards". In: *2007 IEEE 6th International Conference on Development and Learning*. 2007, pp. 163–168.

[109] Jonathan Baxter and Peter L. Bartlett. "Infinite-Horizon Policy-Gradient Estimation". In: *Journal of Artificial Intelligence Research (JAIR)* 15 (2001), pp. 319–350.

[110] L. A. Prashanth. "Policy Gradients for CVaR-Constrained MDPs". In: *International Conference Algorithmic Learning Theory*. 2014, pp. 155–169.

[111] JA Bagnell. "Learning decisions: Robustness, uncertainty, and approximation". PhD thesis. Carnegie Mellon University, PA, USA, 2004, p. 160.

[112] Jens Kober and Jan Peters. "Reinforcement learning in robotics: a survey". In: *International Journal of Robotics Research* (2013).

[113] Borja Fernandez-Gauna, Manuel Graña, Jose Manuel Lopez-Guede, Ismael Etxeberria-Agiriano, and Igor Ansoategui. "Reinforcement Learning endowed with safe veto policies to learn the control of Linked-Multicomponent Robotic Systems". In: *Information Sciences* April (2015). ISSN: 00200255.

[114] Philipp Ertle, Michel Tokic, Richard Cubek, Holger Voos, and Dirk Soffker. "Towards learning of safety knowledge from human demonstrations". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5394–5399. ISBN: 978-1-4673-1736-8.

[115] Martin Pecka and Tomas Svoboda. "Safe Exploration Techniques for Reinforcement Learning – An Overview". In: *Modelling and Simulation for Autonomous Systems*. Lecture Notes in Computer Science 8906. Springer, 2014.

[116] Teodor Mihai Moldovan and Pieter Abbeel. "Safe Exploration in Markov Decision Processes". In: *Proceedings of the 29th International Conference on Machine Learning*. 2012. arXiv: 1205.4810.

[117] Christian Daniel, Gerhard Neumann, and Jan Peters. "Hierarchical Relative Entropy Policy Search". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. Apr. 2012.

[118] Richard H. Byrd, Mary E. Hribar, and Jorge Nocedal. "An Interior Point Algorithm for Large-Scale Nonlinear Programming". In: *SIAM Journal on Optimization* 9.4 (1999), pp. 877–900. ISSN: 1052-6234.

[119] Florian Lier, Marc Hanheide, Lorenzo Natale, Simon Schulz, Jonathan Weisz, Sven Wachsmuth, and Sebastian Wrede. "Towards automated system and experiment reproduction in robotics". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 3298–3305.

[120] Rüdiger Dillmann. "Ka 1.10 benchmarks for robotics research". In: *European Robotics Network (EURON), IST-2000-26048* (2004).

[121] Ching-Chang Wong, Wei-Wen Wang, and Ya-Ling Lee. "Soccer robot design for FIRA MiroSot League". In: *IEEE International Conference on Mechatronics, 2005. ICM'05.* IEEE. 2005, pp. 457–460.

[122] David Obdrzalek. "Eurobot Junior and Starter-A Comparison of Two Approaches for Robotic Contest Organization". In: *Robotics in Education, Bratislava* (2010).

[123] Jiří Iša and Martin Dlouhý. "Robotour-robotika.cz outdoor delivery challenge". In: *Proceedings of the 1-st Slovak-Austrian International Conference on Robotics in Education*. 2010.

[124] Sven Schneider, Frederik Hegger, Aamir Ahmad, Iman Awaad, Francesco Amigoni, Jakob Berghofer, Rainer Bischoff, Andrea Bonarini, Rhama Dwiputra, Giulio Fontana, et al. "The rockin@ home challenge". In: *ISR/Robotik 2014; 41st International Symposium on Robotics*. VDE. 2014, pp. 1–7.

[125] Vojtěch Spurný, Tomáš Báča, Martin Saska, Robert Pěnička, Tomáš Krajník, Justin Thomas, Dinesh Thakur, Giuseppe Loianno, and Vijay Kumar. "Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles". In: *Journal of Field Robotics* 36.1 (Oct. 2018), pp. 125–148.

[126] DARPA. *Challenge Rules*. 2021. URL: `https://www.subtchallenge.com/resources/SubT_Challenge_Finals_Rules.pdf`.

[127] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software* 3.3.2 (2009), p. 5.

[129] Jan Bayer and Jan Faigl. "On Autonomous Spatial Exploration with Small Hexapod Walking Robot using Tracking Camera Intel RealSense T265". In: *2019 European Conference on Mobile Robots (ECMR)*. 2019, pp. 1–6.

[130] Jan Bayer and Jan Faigl. "Speeded Up Elevation Map for Exploration of Large-Scale Subterranean Environments". In: *Modelling and Simulation for Autonomous Systems*. Mar. 2020, pp. 190–202. ISBN: 978-3-030-43889-0.

[131] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. "Distance Transforms of Sampled Functions". In: *Theory of Computing* 8 (2012), pp. 415–428.

[132] Jan Bayer and Jan Faigl. "Decentralized Task Allocation in Multi-robot Exploration with Position Sharing Only". In: *International Symposium on Swarm Behavior and Bio-Inspired Robotics (SWARM)*. 2021.

[133] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[134] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[135] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition*. June 2018, pp. 4510–4520.

[136] Matouš Vrba, Daniel Hert, and Martin Saska. "Onboard Marker-Less Detection and Localization of Non-Cooperating Drones for Their Safe Interception by an Autonomous Aerial System". In: *IEEE Robotics and Automation Letters* PP (July 2019), pp. 1–1.

[137] Matouš Vrba and Martin Saska. "Marker-Less Micro Aerial Vehicle Detection and Localization Using Convolutional Neural Networks". In: *IEEE Robotics and Automation Letters* PP (Feb. 2020), pp. 1–1.

[138] Paula Tarrío, Ana Barbolla, and José Casar. "Weighted Least Squares Techniques for Improved Received Signal Strength Based Localization". In: *Sensors (Basel, Switzerland)* 11 (Dec. 2011), pp. 8569–92.

[139] Tomáš Rouček. *Sensor fusion for object localisation in adverse conditions for mobile robots*. Master thesis, Czech Technical University in Prague. 2020. URL: `http://hdl.handle.net/10467/88208`.