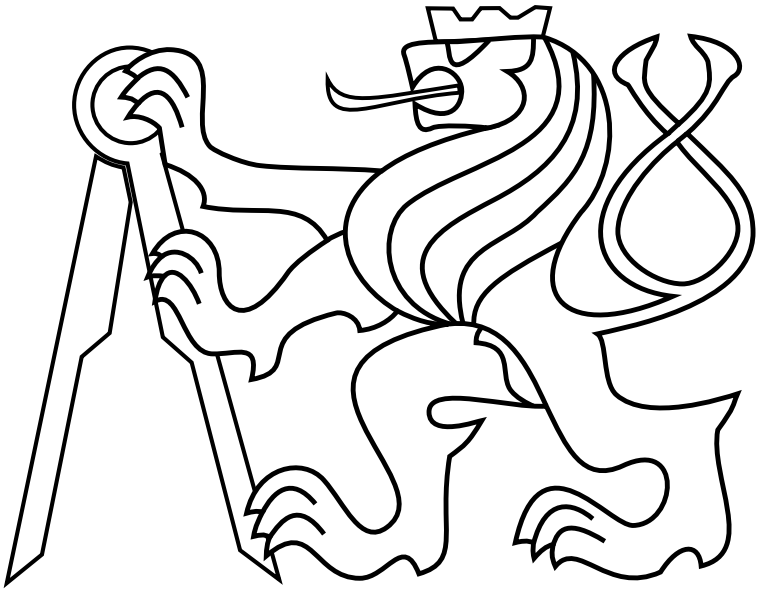
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER'S THESIS



Jasna Petrić

Mission planning for cooperative construction by a team of unmanned aerial vehicles

Department of Cybernetics

Thesis supervisor: **Dr. Martin Saska**

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Petric Jasna** Personal ID number: **471712**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Mission planning for cooperative construction by a team of unmanned aerial vehicles

Master's thesis title in Czech:

Plánování mise více autonomních helikoptér pro kooperativní stavbu zdi

Guidelines:

The goal of the thesis is to design, implement and experimentally verify a high-level mission planning approach for construction of a wall by a team of cooperating unmanned aerial vehicles (UAVs). The task is motivated by the second challenge of the MBZIRC 2020 competition, https://www.mbzirc.com/challenge/2020.
1) Design and implement a realistic environment in a Gazebo simulator under ROS for experimental verification.
2) Specify the cooperative wall assembly as a task-allocation or multi-objective optimization problem to minimize overall time of the mission, to maximise its reliability, and to maximize the overall reward obtained in the MBZIRC competition.
3) Implement the mission planning method and verify it in the simulator with 3 UAVs.
4) Compare different solutions designed for solving the task with communication available and without communication.

Bibliography / sources:

[1] V Spurny, T Baca, M Saska, R Penicka, T Krajnik, J Thomas, D Thakur, G Loianno and V Kumar. Cooperative Autonomous Search, Grasping and Delivering in a Treasure Hunt Scenario by a Team of UAVs. Accepted in Journal of Field Robotics, 2018.
[2] LaValle, S. M.. Planning Algorithms. Cambridge University Press, Cambridge, U.K., 2006.

Name and workplace of master's thesis supervisor:

Ing. Martin Saska, Dr. rer. nat., Multi-robot Systems, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **13.02.2019** Deadline for master's thesis submission: **24.05.2019**

Assignment valid until:
by the end of summer semester 2019/2020

Ing. Martin Saska, Dr. rer. nat.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

v Praze dne ....

podpis autora práce

Author statement for undergraduate thesis

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university thesis.

Prague, date ....

Signature

Acknowledgements

I would like to thank my supervisor Dr. Martin Saska for his advices and guidance throughout this thesis. Also, many thanks to Vojtěch Spurný for his valuable insights, numerous long discussions, and sharing his comments and ideas, and to everybody from the MRS group for help during the work on this thesis.

Further, I would like to express deepest gratitude to my devoted father Dragoljub, my siblings, my little nephews Maša and Lazar, and my friends, for their unconditional love and support throughout my Master studies. Last but not least, I am grateful to my beloved mother Marina, whose presence, though short, was very precious and made me who I am today.

Abstract

This thesis aims to solve the task of high level motion planning for a group of unmanned aerial vehicles (UAVs). UAVs are tasked to build a pre-defined wall structure using different types of brick shaped objects. The main focus of this work is to solve the problem of efficient and successful wall construction. The goal is to find the best sequence which defines the order in which the bricks will be placed while building the wall. A mission planner provides the hierarchically ordered states, representing the actions that the UAV is supposed to do. For a successful cooperation of the UAVs, it is important to have a proper communication. Communication can be established by WiFi, but even if the communication channel cannot be established, the approach of time windows is proposed. This thesis contributes to a solution to a task in the Mohamed Bin Zayed International Robotics Challenge (MBZIRC), that will be organized in 2020. The developed system is tested in Gazebo simulator, but part of the thesis is tested on a real UAV setup in an actual real word environment.

Keywords: unmanned aerial vehicle, multi-robot planning, aerial object manipulation

Contents

List of Figures **ix**

List of Tables **xi**

1 Introduction **1**

1.1 MBZIRC competition 2

1.2 Problem statement 3

1.3 Structure of the thesis 6

2 Related work **7**

2.1 Task allocation 7

2.2 Path planning 8

2.2.1 Graph and tree search 9

2.2.2 Cooperative path planning 10

2.3 Object manipulation 10

2.3.1 Grasping objects 10

2.3.2 Dropping objects 11

3 Cooperative wall building **13**

3.1 Processing the wall image 13

3.2 Tree build 15

3.3 Best path search 17

3.3.1 Cost formulation 17

3.3.2 Brute force 19

3.3.3 Greedy algorithm 19

3.3.4 Improved greedy algorithm 19
3.4 Approaches comparison 22

4 Motion planning **25**
4.1 Arena description 25
4.2 Mission planner 26
4.3 Communication between UAVs 29

5 Experiments **31**
5.1 Validation in simulation 31
5.1.1 Software system structure 31
5.1.2 Simulations 32
5.2 Hardware experiments 32

6 Concluding remarks and future work **41**

Bibliography **43**

Appendices **49**

Appendix List of abbreviations **53**

List of Figures

1.1 Example of drone used for last MBZIRC competition. 2

1.2 Scheme of the task solution for Challenge 2 of MBZIRC 2020. 3

1.3 The prototype of the drone built for MBZIRC 2020. 4

1.4 Red and orange brick shaped objects. 5

1.5 Blue and green brick shaped objects. 5

3.1 On the left image it can be seen example of wall image, and on the right image is a 2d array of *id*s representing the type of the bricks in the wall. . 15

3.2 All bricks in the wall assigned with their ids. 16

3.3 The exploration of the tree representation of the wall on example shown in Figure 3.2. 16

3.4 The different cases of delivery of objects. Considered situations: (the top left) there is no bricks placed yet, (the top right and the bottom left) there is one brick placed around the place the following brick should be placed, and (the bottom right) the following brick should be placed between two bricks. 18

4.1 Sketch of arena decomposition on the left side and arena decomposition from the simulator on the right side. 26

4.2 General state representation with possible outcomes. 27

4.3 Main state machine representing mission planner. 28

4.4 The wall waiting positions for UAVs are shown. 30

4.5 The division of time window. Time slots assigned to each of the UAVs in case of communication lack or failure. 30

5.1 UAVs in their start positions in Gazebo simulator. 33

5.2 UAVs in their initial waiting position close to the spot with color objects they are assigned to carry, are shown on the left images. On the right images UAVs grasping the color objects are shown. 34
5.3 UAVs in their waiting positions in front of the wall space, waiting to enter in the wall zone. 35
5.4 UAVs placing bricks in the wall. 35
5.5 The final wall, built by team of three UAVs following the planner provided order of placing bricks. 36
5.6 The drone and bricks objects used for experiments. 37
5.7 The UAVs flying to get brick shaped objects. 37
5.8 The UAVs attempting to grasp the blue and red colored bricks. On the corners of images, the views from the cameras on UAVs can be seen. 38
5.9 The UAVs in the waiting positions in front of the wall. The UAV that will enter the wall zone first is carrying the object of the color defined by planner to be placed as the next one in the wall. 38
5.10 On the top image, UAV with priority of the color of the object that is carrying, defined by the planner, is entering the wall to deliver the object. After the first UAV left the wall zone, the second UAV is entering the zone. On the left corner of the bottom image can be seen the view from the camera on the drone. 39

List of Tables

3.1 The specifications of the brick shape objects. 14

3.2 Time needed for taking particular actions with bricks. 17

3.3 The algorithms comparisons for the case of the small wall shown on the Figure 3.2. Relative speed denotes relative improvement compared to the brute force algorithm. 23

3.4 The algorithms comparisons in case of the wall dimensions $l = 5$, where number of wall levels is denoted as l, and maximum number of bricks in level is 5. 24

3.5 The algorithms comparisons in case of the wall dimensions $l = 5$, where number of wall levels is denoted as l, and maximum number of bricks on each level is 10. 24

5.1 The algorithms comparisons for the case of the final wall shown on the Figure 5.5. Wall is built by 3 UAVs with established communication. 33

1 CD Content 51

2 Lists of abbreviations 53

Chapter 1

Introduction

Contents

1.1 MBZIRC competition **2**

1.2 Problem statement **3**

1.3 Structure of the thesis **6**

Unmanned aerial vehicles (UAVs), also known as drones, have demonstrated exceptional technological advantages and the broadness of ways in which they can be applied is constantly increasing [1]. Control of UAV platforms can be done remotely by a human. Or, it can be done autonomously. Autonomous flights can be achieved using intelligent systems with on-board sensors. In this thesis, a fully autonomous system using onboard computational resources and sensors is presented.

The benefit of using drones is mainly for their ability to reach the locations that are not accessible for other types of robots. This implies their possible huge impact in the military development, photography and delivery industry [2, 3, 4]. The advance in UAV technology allows us to consider using even multiple cooperating robots simultaneously, which further increases their potential application.

One of the most common tasks in multi-robot systems is coordination of the robots and their motion planning. The definition and implementation of the motion planning task for a UAV platform is challenging. It involves dealing with uncertainty in the vehicle state and restricted knowledge about the environment caused by the limited sensor capabilities.

Using a group of UAVs raises the difficulty when the drones collaborate together while sharing the same workspace. This means that the task allocation between robots has to be included as well.

Figure 1.1: Example of drone used for last MBZIRC competition.

1.1 MBZIRC competition

The Mohamed Bin Zayed International Robotics Challenge (MBZIRC) is an international robotics competition, that have been organized in Abu Dhabi, UAE [5]. The next MBZIRC competition is scheduled for February 2020.

MBZIRC aims to inspire future robotics through innovative solutions and technological excellence. MBZIRC provides an ambitious set of challenges. The competition challenges include robots working more autonomously in dynamic and unstructured environment, while collaborating and interacting together.

MBZIRC 2020 will consist of three individual challenges focused on robotics solutions based on autonomous aerial and ground robots, accomplishing navigation and manipulation tasks, in outdoor and indoor environments. The challenge focus will be on the areas of safely neutralizing stray drones, construction automation, and urban fire-fighting:

- In Challenge 1, team of up to 3 UAVs is used to autonomously locate, track and interact with a set of objects moving in space. Challenge 1 is motivated by UAV safety.

- In Challenge 2, a team of robots containing three UAVs and one unmanned ground vehicle (UGV) will be used to solve the task that aims this team to collaborate and autonomously locate, pick, transport, and assemble different types of brick shaped objects to build pre-defined structures in an outdoor environment. This challenge is motivated by construction automation.

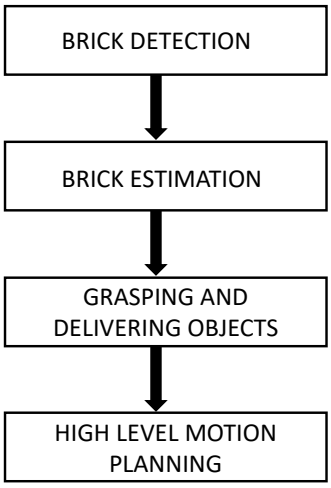


Figure 1.2: Scheme of the task solution for Challenge 2 of MBZIRC 2020.

- In Challenge 3 team of up to three UAVs and an UGV is required to collaborate to fight fire in a simulated high rise building.

1.2 Problem statement

This thesis contributes to the Challenge 2 of Mohamed Bin Zayed International Robotics Challenge. The task that is being solved, is high level motion planning for a group of aerial vehicles meant to build a pre-defined structure using different types of brick shaped objects [6]. For MBZIRC 2020 this is a part of preliminary work of a team composed from Czech Technical University (CTU), University of Pennsylvania and New York University. Only aerial vehicles are considered in this thesis.

The goal of this thesis is to deal with collaboration of the UAVs. Hence, collaboration of three UAVs will be studied. The brick shaped objects have following colors: blue, green, orange, and red, and different sizes and weights, see Figures 1.4 and 1.5. In the case of red, blue, and green bricks, grasping and delivering of the objects can be done individually, by one drone.

The complexity of the task solution involves connection of four sub-tasks. These sub-tasks are: brick detection, brick estimation, object grasping and delivering; and high-level motion planning task. Figure 1.2 illustrates a scheme of the challenge solution steps. Except for the high-level motion planning task, the rest of the scheme sub-tasks are not solved as a part of this thesis, but as they are used in mission planning, they will be described briefly.

Brick detection is a vision detection part of the task. It allows the drone to be able

Figure 1.3: The prototype of the drone built for MBZIRC 2020.

to find the brick objects on the ground. It is programmed to detect objects using known colors and shapes of the objects.

A continuation of brick detection is brick estimation task. Once the brick is detected in the space, the estimation part will try to estimate its position and orientation. So, not just that estimation provides the location of the object, it also allows the drone to detect the orientation of the brick. This feature facilitate the next steps in the solution scheme.

Grasping part is done such that once the positions of the bricks are known by drone it aligns with the nearest brick and picks it up with a magnet. On the other hand, delivering of the objects is done by descending to an altitude above a position where the bricks should be placed and then by deactivating the magnet and dropping the object.

High-level motion planning task is the main topic of this thesis. In addition to the mission planner, the aim of this thesis is to design a communication link among the robots. This communication package is enabling the exchange of the information about performing actions between the robots, but it also provides the solution in the case when the communication between the UAVs is not available, due to the lack of WiFi connection. The planner is providing the paths for each UAV of the team.

This thesis will be focused on manipulation of the payload that involves operations that can be done by a single drone. The most important part of the thesis is dealing with the wall construction. The wall construction is based on a search for the sequence that defines how to place all the bricks. The achieved solution should give order of placing bricks on the wall, that will satisfy the constraints given on the time for the operation execution and the amount of bricks to be placed. Hereby, the brick placing order should be the solution that will provide wall building in the shortest time, and ensure that in every moment if a drone failure happens, the number of bricks placed until that moment will be maximized.

The solution presented in this thesis is based on research that is done at Multi-robot systems group (MRS) of CTU in Prague [7], [8], [9], [10], [11], [12], [13]. It is continuation on the research approaches that were done by the MRS group for previous MBZIRC com-

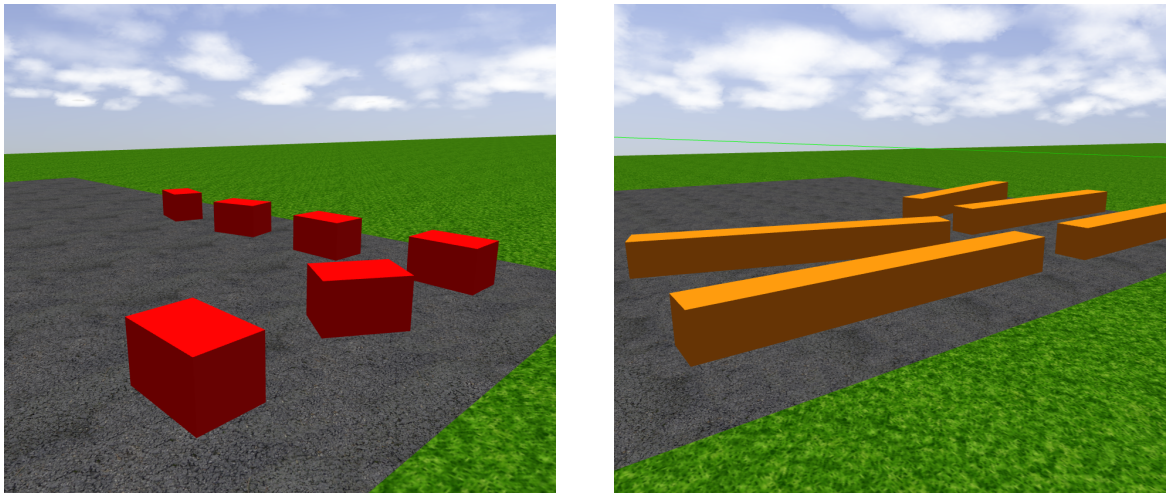


Figure 1.4: Red and orange brick shaped objects.

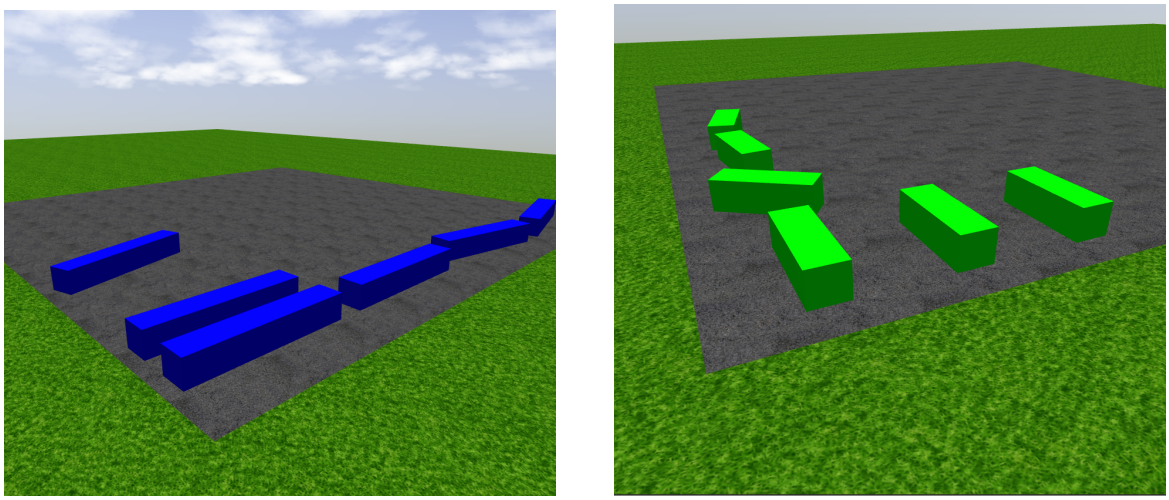


Figure 1.5: Blue and green brick shaped objects.

petition in 2017. The work done in the thesis is related to the solution done for the Treasure Hunt task of MBZIRC 2017. The Treasure Hunt task requires addressing the problems of cooperative localization, grasping and delivering of colored ferrous objects by the team of UAVs.

1.3 Structure of the thesis

- The related work is described in the Chapter 2. Most of the cited research work focus on the topics related to motion planning. Firstly, task allocation is described, i.e. how the distribution and division of tasks can be done between the robots. Furthermore, path planning approaches are described, with most attention dedicated to graph search and cooperative path planning. Moreover, the work done on the topic of the robot manipulation is described, particularly grasping and dropping of the payload.

- Chapter 3 presents the tree search approach for finding the wall building order. It introduces the constraints for defining costs of placing bricks in the wall. To find the optimal solution to go through the tree representation of the wall and achieve the sequence that ensures building the wall in the shortest time following algorithms are used and compared: brute force algorithm, greedy algorithm and improved greedy algorithm. This chapter also deals with the case of a drone failure, so the bricks with the smallest costs will be placed in the beginning, in order to maximize the reward.

- Chapter 4 puts additional focus on the planning and the communication part between the UAVs. It describes motion planer representation in the form of the state machine. Moreover, it describes how and which messages are exchanged between the UAVs, and what will happen if the communication can not be established.

- Chapter 5 introduces and gives details about the software that is used to test the approaches presented in this thesis. For verification of the results Gazebo simulator is used [14]. The results that are achieved in the simulator are described and depicted, as well as the experiments that are done in the real world environment on real drones.

- Finally, Chapter 6 provides concluding remarks for the work done so far. Also, it discusses the future work, which gives pointers on how to extend and further develop this topic and solutions of the same problem.

Chapter 2

Related work

Contents

2.1 Task allocation **7**

2.2 Path planning **8**

2.3 Object manipulation **10**

First, the task allocation problem is considered in Section 2.1, defined as a decomposition of the task among a given number of robots. Next, path planning is described in Section 2.2 as a general planner of path connecting a set of points. Path planning is mostly based on the work done in the branch of graph and tree search, and cooperating path planning algorithms. Finally, Section 2.3 is focusing on the work related to the object manipulation. Specifically, in Section 2.3, methods related to the subjects of grasping and dropping the payload are presented.

2.1 Task allocation

Task allocation can be defined as a mapping between robots and tasks. In other words, how to optimally assign a set of robots to a set of tasks in such a way that optimizes the overall system performance [15, 16]. Task allocation is one of the core issues in exploring the capabilities of cooperation of a team of UAVs. The decomposition of a overall mission in independent sub-tasks , hierarchical trees or just sequence of roles provided by the general planner is included in task allocation. Independent sub-tasks can be done simultaneously, but in the case of trees, sub-tasks are executed based on their interdependence [17]. Solution to task allocation problem can be split to centralized and distributed approaches.

Centralized approaches are trying to optimize a global cost function. They are characterized by a single control agent. The centralized allocation is the task allocation where

all decisions regarding the task are made by one single agent with global view of the system [16, 18]. The main advantage of centralized approach is the ability to produce an optimal planning since a decision making agent utilizes the relevant information from all robots. This method is suitable for time-optimal task allocation. The centralized approach, also, has some disadvantages, mainly because it needs full communication between the agent and the rest of the team constantly, and in most of the cases this is difficult to achieve. Moreover, the agent should be informed all the time about the progress of each team member, and he needs to keep the track of the changes. Also, these systems are too sensitive to the failure of any small part [19, 20].

Distributed approach, on the other hand, is assuming allocation based on decentralized collection of knowledge sources [21]. Its advantages compared to the centralized approach are manifold. It means each of the UAVs, or in general agents, is allocating itself alone tasks. It is not sensitive to single failures, and does not requires constant communication between agents. Most important there is no global control, neither global storage. It is faster, more reliable, with higher tolerance in uncertain data. But on the other side, not all problems can be decomposed well. Also, plans made for robots are based only on local information, and the solutions of the decentralized approach are often highly suboptimal [19, 22, 20].

In this thesis, centralized approach is firstly used for offline finding of the optimal sequence of bricks to be placed in the wall. This sequence is common and shared between all UAVs. Further, the task allocation for each UAV is done separately, leading to the distributed approach.

2.2 Path planning

Path planning is one of the open problems that is constantly attracting a lot of attention, and new solutions are appearing often [23]. The path planner generates one or more paths between given number of points. Path planning is a complex problem that involves dealing with physical constraints of UAVs, environment constraints and other requirements dependent on the current task. One of the surveys of motion planning algorithms for the case of autonomous UAVs is presented in [24]. Generally, for mobile robots, a large amount of research is done [25, 26, 27, 28, 29]. Path planning algorithms should give the best possible path, preferably path of minimal length. Also, the optimal path has to avoid collisions, and satisfy cooperative properties envisioned for the drone. Moreover, path planning should be implemented to be time computationally efficient so that it can run in real-time, and allow UAVs to rebuild their paths, if necessary.

2.2.1 Graph and tree search

Graph and tree search are described in many books and papers including [30, 31]. Graph and tree search may be used as one of the possible solutions of path planning problems for UAVs.

A graph $G(v, e)$ is defined as a set of vertices v that are connected by a set of edges e. In case of robotic path-planning tasks, discrete points in space are assigned as vertices and edges that are connecting them may be used to describe properties of travelling from one point to another, containing the awards for their traversal.

Graph based solution usually suffers from the problem of computation complexity, which is even more pronounced when the complexity of the task is high. One of the ways to avoid the dimensionality problem is to use a sequence of graphs. More specifically, find the optimal solution for each sequence, and then, in the neighborhood of this optimal solution, build a new graph to search for new optimal solution [32]. This is repeated until the final point is reached. In [32], Voronoi polygons are used for constructing such graph. The Voronoi diagrams are considered as graphs, with Voronoi points as vertices and edges connecting them assigned with some weights. Once the weights are assigned, the optimal path is found.

Nowdays, many approaches can be found in literature that are dealing with the path planning task. In [33, 34], visibility graphs are used to plan the paths for mobile robots in combination with the well known A* algorithm. Also, A* is used in [27, 35]. Algorithms, like rapidly-exploring random tree (RRT) can be used as well, with all their amendments, [36, 37, 38]. Also, potential field approach is widely used in this problem statement [39, 40].

A tree is a directed graph that is connected and does not contain cycles [41, 42, 43]. A tree is a structure of hierarchically linked nodes where each node represents a particular state. Nodes have none, one or more child nodes. Tree search algorithms attempt to find a solution by traversing the tree structure starting from the root node and expanding the child nodes in a systematic way. These algorithms are split in two groups: blind search algorithms and best-first search algorithms.

The blind algorithms are also known as uniformed algorithms. They work only with information to distinguish the goal state from non-goal states. All information available to blind search algorithms is the state, the successor function, the goal test, and the path cost. Different algorithms from this group differ only by the way how they expand the nodes, and that can influence the performance dramatically. In this group are: Depth-First search algorithm (DFS) [44, 45] and Breath-First search (BFS) algorithm [44, 46].

The best-first search algorithms are known as informed search algorithms. The algorithms from this group use heuristics to decide which adjacent is most promising and then explore. These algorithms are done with the priority queues. These priority queues are typically used to store the costs of the nodes. Best-first search algorithms are often used for path finding in combinatorial search studies. Between these algorithms are Greedy algorithm [47] and A* algorithm [27, 35] .

This manuscript uses a tree search approach. For traversing the tree and finding the optimal solution, following algorithms are used: brute force, greedy algorithm, and a proposed improved greedy algorithm.

2.2.2 Cooperative path planning

The general advantage of coordination between UAVs is their benefit in decreasing the period of the time needed to accomplish some task. Moreover, if a damage happens on a single UAV, it will not necessarily cause the whole mission to fail [48]. Different path planning algorithms for a team of UAVs were studied, and tested in various environment. Between them, most notable are: Dijkstra's algorithm, and A* algorithm. They were compared so that there can be monitored and established the path for communication. The best results were given by A* algorithm, with guarantees to achieve the goal and find the shortest path.

There are many studies dealing with the simultaneous arrival of a group of UAVs on the target position. In [49], the main task is to achieve the flyable and safe paths. The trajectory for each of the UAVs is introduced with constraints to achieve the safety in path following as minimum distance between the non-intersecting paths and in the end to achieve the target positions by paths of the same length .

The wall task in this work is defined as a tree search problem. Three UAVs are used to build the wall structure. The successful cooperation is ensured by having a reliable communication channel. Otherwise, when the communication is unavailable, a fallback routine is used, that does not depend on establishing the communication channel.

2.3 Object manipulation

In this section the topics of grasping and delivering of the objects will be discussed. These parts are not directly developed in the thesis, but they are considered as important step in motion planning task definition thus they will be described in following sections.

2.3.1 Grasping objects

This grasping task is defined as a unity of three sub-tasks [50]. It involves detection and estimation of objects, and finally the grasping. Object recognition is a wide field of a machine learning and computer vision research area. Some of the authors use neural networks [51], for detection based on the prior knowledge of environment, as well as on the sensor data. To estimate position of the object, authors in [52, 53] use depth information received from the depth sensors. Also, online detection and estimation of objects position based on features from images is proposed in [54, 55].

Generally, the grasping by the aerial robots is done when some of the features are known, such as color, some pre-defined mark [56], or even known object position. The ability to pick up and transport payload is valuable for aerial vehicles, and it is consistent on many difficulties. There are increasing number of examples of interaction between aerial vehicles and objects. We have some examples of slung load attachments that could be transported individually or cooperatively [57, 58, 59]. The objects are not attached automatically.

The grasping grippers are depending on the material of the interacting objects. So there could be two possible ways: to attach a magnet on the probe in the case of ferromagnetic objects; or to use a hook to take the object. In [60], the quadrotor helicopters with several grippers and the ability to grasp and manipulate number of items are shown. They used recursive least-square methods to identify the mass of the object, center of the mass-offset, and the moment of inertia.

The grasping part is done based on the solution proposed by our team from the previous MBZIRC 2017 [7]. There, from the known object size, color and camera parameters, the relative positions of the objects to the drone are calculated. Furthermore, these positions are transformed in the global frame. All these positions are added to the map. After the map of the objects is created, the trial to grasp is attempted. The difference compared to the previous MBZIRC challenge is that the positions of the spots with objects are known, and all the brick objects are static. In MBZIRC 2017, the task was dealing with static and dynamic objects, located on unknown positions.

2.3.2 Dropping objects

Nowadays, UAVs should be able to autonomously and precisely deliver object on a given position. The dropping objects task is based and implemented similarly as landing approaches. The precision of landing task solutions is indispensable for autonomous missions that involve carrying objects. There are different situations that are investigated on the topic of delivery payload on given position. In the paper [61], authors are presenting the aircraft sliding the payload along the cable down to the ground, while aircraft stays in the air.

Probably the easiest method is to release the object from an UAV, when the UAV is above its target position, without taking care about precise positioning, letting it fall unguided close to the ground position. This is done in [62], a work describing the payload delivery on an iceberg. They released the object when they were certain that the iceberg is under the aircraft.

The part related to the delivery of the objects is done based on the solution proposed by MRS team in [7]. Compared to the previous MBZIRC 2017, where the objects were dropped above the known position (inaccuracy in dropping position $\sim 0.5m$ did not have impact on the result), the proposed solution needs to place the objects on the precise positions.

Chapter 3

Cooperative wall building

Contents

3.1 Processing the wall image **13**

3.2 Tree build **15**

3.3 Best path search **17**

3.4 Approaches comparison **22**

Approaches designed to efficiently solve the wall building task are described in this chapter. The goal is to achieve the order of placing bricks that will result in a solution with the smallest cost in total. This approach assumes that each action related to the manipulation with a brick assigns a cost to the brick. This cost is defined as difficulty of action execution and time that particular action takes.

Firstly, wall building as a tree search problem, and procedures to build and search such a tree are explained. Further, the cost assigned for each brick is introduced. Finally, approaches for finding the best path to traverse the tree representation of the wall are described. The best path represents the solution and the order in which the wall will be built.

3.1 Processing the wall image

It is assumed that shortly before each competition trial an image of the required wall shape will be provided by organizers. This image has to be processed to obtain the shape of the wall. Therefore, the initial phase of our proposed system is an image processing algorithm that automatically transfers the provided image into a 2d array. This array contains indexes of bricks depending on their color. As it was mentioned before, the bricks

differ by their color c, but also by their weight m and size len. The specification of each brick type is shown in the Table 3.1. Orange bricks will not be considered here, as we only consider brick actions performed by a single UAV in this thesis.

By applying the indicators type according to the Table 3.1 on the wall image we can get the processed image that is easily represented. See Figure 3.1 for an example. On the left side, an example of a wall image can be seen. On the right side, the final wall representation in the form of a 2-d array of brick color ids is shown.

Firstly, in our image processing procedure, the image is expressed by RGB color values. The whole image is thus presented as a 2d array of pixels P, where each pixel is represented using 3 values. Let $P[i][j]$ be one pixel in array P. Further, pixel is assigned to be the type of object by its color. The classification can be done using Algorithm 1.

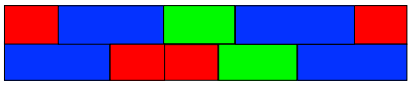
Algorithm 1 Pixels to *object_type*s indexes representing colors of the bricks.

1: $pixel \leftarrow P[i][j]$ ▷ $P[i][j] = (r, g, b)$
2: $red = P[i][j][0]$, $green = P[i][j][1]$, $blue = P[i][j][2]$
3: **if** $((red > 2 \cdot green)$ and $(red > 1.5 \cdot blue))$ **then**
4: $object\_type = 1$
5: **else if** $((green > 1.5 \cdot red)$ and $(green > 1.5 \cdot blue))$ **then**
6: $object\_type = 2$
7: **else if** $((blue > 1.5 \cdot red)$ and $(blue > 1.5 \cdot red))$ **then**
8: $object\_type = 3$
9: **else**
10: $object\_type$ = -1
11: **end if**

The advantage of this method is that the background and borders between bricks are as well identified. They are used to define how many pixels represent each brick. In other words, every time that pixels with $object\_type = -1$ are detected, that would mean the brick is found and can be saved. After the pixels are connected to the bricks, all pixels with this *object_type* value are not further processed.

Table 3.1: The specifications of the brick shape objects.

Bricks	*weight [kg]*	*size [m]*	*color id*	*rgb value*
Redbrick	≤ 1	0.3	1	(255, 0 ,0)
Greenbrick	≤ 1	0.6	2	(0, 255, 0)
Bluebrick	≤ 1.5	1.2	3	(0, 0, 255)
Orangebrick	≤ 2	1.8	4	(255,165,0)



1	3	2	3	1
3	1	1	2	3

Figure 3.1: On the left image it can be seen example of wall image, and on the right image is a 2d array of *id*s representing the type of the bricks in the wall.

3.2 Tree build

Each position for a brick in the tree is assigned by identification number (id) . The *id* is counted as a order from left to right, based on level l of the wall in which the slot for a brick is located and based on position p in this level as:

$$id = 100 \cdot l + p, \tag{3.1}$$

where 100 is taken as the smallest number that is a power of 10, but it is higher than the maximum number of bricks that can appear in any level. In the Figure 3.2, the wall representation example shows how the *ids* are assigned to each brick.

Each of these slots for bricks in the wall is presented as a node in the tree. Every node is containing information about the color of the brick, *id* of a brick that was one step placed before, and *ids* of bricks that need to be placed before this brick. Having these nodes that contain all important information about bricks that have to be placed in each slot in the wall, the tree structure is achieved. For every node in the tree it is known its children *ids* and *ids* of all bricks that have to be places before the brick which is relevant to the node.

Root node is assigned as *empty* node, not containing any brick, whose children are all nodes that represent the bricks on the first level in the wall. Therefore, the first brick that can be placed in the wall can be any of the bricks from the first level, Figure 3.3. As the leaf node, tree can have any of the nodes representing bricks located in the last level in the wall. There is not precisely specified which from the nodes from the last level needs to be the leaf node.

The approach of not specifically assigning the root node and leaf node guarantees that every possible path through the tree will be explored, knowing that the path can start from any node in the first level and end in any node in the last level.

Additionally, all the nodes relevant to the bricks that are placed before the current one are saved as its predecessors in the parents list. Children of each node are the children of the current node, but also the children of all predecessors nodes that are not contained in a parent list. This gives us tree structure as it is shown on Figure 3.3, for example of the wall shown of Figure 3.2.

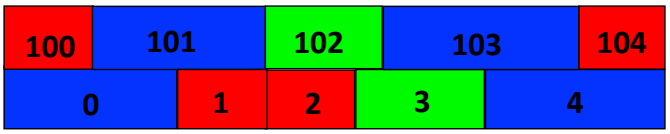


Figure 3.2: All bricks in the wall assigned with their ids.

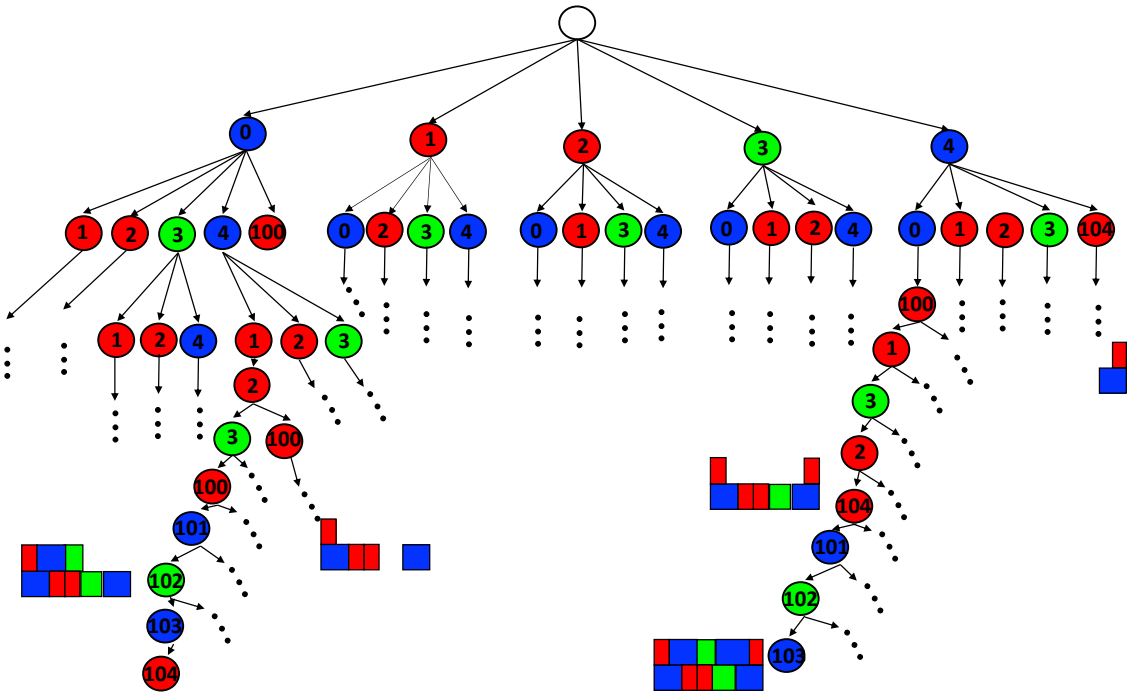


Figure 3.3: The exploration of the tree representation of the wall on example shown in Figure 3.2.

3.3 Best path search

In this section will be shown the solution of the tree search that ensures achievement of the best path through the tree. The best path gives the sequence of the order how the bricks should be placed in the wall, in the shortest time period.

First, the cost definition is described in Section 3.3.1. After, algorithms for tree search are explained: brute force, greedy algorithm, and an improved version of greedy algorithm. In the end, all of the methods are compared and results are shown in Section 3.4.

3.3.1 Cost formulation

Lets define cost c as a sum of grasping cost t_g, flying cost t_f, and delivery cost t_d. Cost t_g represents time necessary for grasping of the object, t_f represents time necessary for flying with the object, and t_d represents time necessary for dropping the object. As the bricks have different weights and dimensions, and they are as well placed in different spots on the wall, time that takes particular action with the brick will differ based on the brick color.

Thus the proposed cost that defines cost of placing one brick is formulates as follows:

$$c = (1 + e^{-\alpha \cdot l}) \cdot (t_g + t_f + t_d), \tag{3.2}$$

where α parameter controls how big the decay of exponential function will be, and it is set to be $\alpha = 0.1$. Parameter l is denoted as the level where the brick is placed.

In the Table 3.2 it is shown how much time in average takes to grasp each type of the bricks object in simulation. Moreover, the grasping cost t_g can be influenced by detection of the bricks. The UAV can easily manage to detect the brick and go to take it, but it can happen as well that during the grasping maneuver it will lose it from its view. It can happen that the drone does not manage to grasp the brick. In this case, drone will repeat the grasping action again. Let n be the amount of drone repetition of doing the grasping action on the same brick. This means the grasping cost should be computed as $n \cdot t_g$, which represents the worst case. This could make the cost of grasping the brick really high.

The flying cost is the time that takes for the drone to carry object from one point in the space to another. It contains the time to go for a object, and once an object is grasped,

Table 3.2: Time needed for taking particular actions with bricks.

Brick	*Grasping time* $[s]$	*Flying time* $[s]$	*Dropping time* $[s]$
Red	14	22.5	8
Green	26	18.5	9
Blue	16	22.5	15

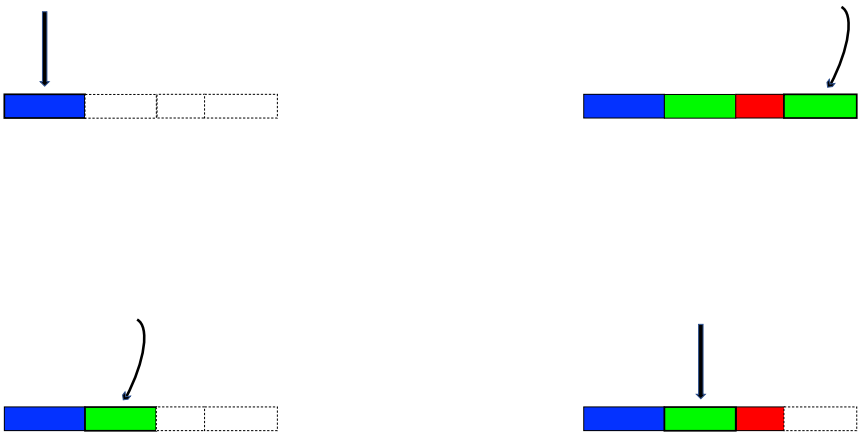


Figure 3.4: The different cases of delivery of objects. Considered situations: (the top left) there is no bricks placed yet, (the top right and the bottom left) there is one brick placed around the place the following brick should be placed, and (the bottom right) the following brick should be placed between two bricks.

it is measured what is the duration of the flight from spot with bricks, to the drones waiting positions in front of the wall. This component is also dependent on the type of the brick that drone is carrying, see the Table 3.2.

The delivery-cost is defined as the time that dropping action takes. Cost of delivering objects is measured from the moment UAV enters the wall space until it drops the object on certain position. This parameter is mostly dependent on the position of the brick, where brick should be placed in the wall. There are considered cases such that, i.e., there are no bricks around the slot the carried brick should be placed, there is one brick next to it, and that the brick need to be placed into slot between two bricks. Figure 3.4 shows these situations.

Another aspect that needs to be considered in the competition is to maximize the amount of bricks with the smallest cost that will be placed in the beginning. There can be many solutions for finding the optimal path through the tree that have the same costs in total. But some of these solutions can give the sequence of order for placing the bricks such that in the beginning are placed bricks with higher costs and after the bricks with smaller costs. This is situation that should be avoided, especially if there is considered the application of the solution will be with real UAVs. Basically, the aim is to achieve in the competition as high score as the possible. The grading of the achievements will be based on the amount of bricks that are placed, in other words the area of the wall that is built. Due to there is possibility of drones failure, goal is to start with placing bricks with the

smallest cost. Because of this issue, the exponential part in equation (3.2) is emphasizing the costs at lower levels, while decreasing the importance of costs at higher levels.

3.3.2 Brute force

The total cost of building the wall is sum of the costs of traversed nodes. So, total sum of the cost for traversing the tree choosing particular path from any of the root children nodes to any of the leafs, can be calculated. For all possible ways of going through the tree, the final sum of the node costs is saved. The cheapest path to go through a tree is selected as the best path. The algorithm is working as it is shown in Algorithm 2.

The algorithm is done in form of recursive function $brute\_force$. Current node contains all the nodes that were placed before it, in its parents list. All nodes that can be explored next, after the current node, are stored in the children list of the current node. After traversing each node, the total sum of the costs is increased for the cost of the travelling the current node. This function procedure is repeated for each of the children nodes in a recursive way. If the number of nodes that are traversed is equal to the total number of the bricks, the whole height of the tree is explored and that path is saved. In the end, the path with smallest cost is chosen as the optimal.

3.3.3 Greedy algorithm

In greedy algorithm the growth of the tree is evaluated using some heuristic. The cheapest children of the node is then selected and expanded. The heuristic can be based on the distance, how close the current node is to the goal. The nodes that are considered the closest to the goal node are traversed the first. In the case of wall build task the heuristic can be defined by the costs of the nodes. In other words, as the next node to be visited it is chosen the node that has the smallest cost from all the children nodes.

The greedy algorithm is shown in Algorithm 3. Firstly, for current node are saved all the predecessors, and found the children. Then, the child with the smallest cost among children is found and recursive function $greedy\_algorithm$ is then executed for this child. The greedy algorithm provides only one path as final way of the traversing the tree representing the wall.

3.3.4 Improved greedy algorithm

Improved greedy algorithm is a combination of the standard greedy approach and the brute force solution. This approach allows to choose how many children nodes will be explored. For example, brute force explores every child of the current node, while greedy picks only the best. In order to satisfy the condition that the bricks at the bottom levels are more important because they are easily built, we choose to only explore one child in the

Algorithm 2 Brute force algorithm

```
1: procedure FIND BEST PATHS
2:     brick ← node
3:     cost_sum ← 0
4:     n ← number of bricks
5:     final_paths ← []
6:     parents_list ← []
7:     function BRUTE_FORCE (BRICK, PARENTS_LIST, FINAL_PATHS = [])
8:         copy parents_list in my_parents_list
9:         expand my_parents_list with brick.id
10:        cost_sum = cost_sum + brick.cost
11:        children = find_children(brick)
12:
13:        if length of my_parents_list equal to n then
14:            expand final_paths with (my_parents_list, cost_sum)
15:        end if
16:        for child in children do
17:            brute_ force (child, my_parents_ list, final_paths)
18:        end for
19:        return final_paths
20:    end function
21:    final_paths = brute_force(root, [], [])
22:    find path of minimum cost in final_paths
23: end procedure
```

Algorithm 3 Greedy algorithm

```
1: procedure FIND BEST PATH
2:     brick ← node
3:     cost_sum ← 0
4:     n ← number of bricks
5:     final_path ← []
6:     parents_list ← []
7:     function MIN_COST(CHILDREN)
8:         for child in children do
9:             find child with min_cost
10:        end for
11:    end function
12:    function GREEDY_ ALGORITHM (BRICK, PARENTS_LIST, FINAL_PATH = [])
13:        copy parents_list in my_parents_list
14:        expand my_parents_list with brick.id
15:        cost_sum = cost_sum + brick.cost
16:        children = find_children(brick)
17:        if length of my_parents_list equal to n then
18:            expand final_path with (my_parents_list, cost_sum)
19:        end if
20:        c = min_cost(children)
21:        greedy_algorithm (c, my_parents_ list, final_path)
22:        return final_path
23:    end function
24:    final_path = greedy_algorithm(root, [], [])
25: end procedure
```

first n levels of the wall, and then, for the levels higher than n, we can explore k children with the lowest cost. In this manner, we will avoid costly full tree exploration that the brute force does, but we will still explore more solutions than just a single one that greedy does.

Two parameters can be adjusted here, n and k, both allowing algorithm to be faster at the cost of less optimal solution, or, if we want, choosing stricter values to get more optimal solution by paying the price of traversing a bigger part of the tree. This is an additional benefit, as we might have more time to spend on tree search at the beginning of the challenge, but less towards the end, if new paths need to be found.

Hence, due to unfeasibility of current solution (decreased number of drones for accomplishing the mission due to technical error), the ability to set different values depending on how far we are in the challenge, is another positive aspect of this approach.

3.4 Approaches comparison

The approaches used for finding the best path are compared based on the time for execution of the algorithm, and based on the total cost of the optimal path that defines the order in which bricks are placed in the wall. We experiment with two different wall setups, that have different complexities. A simple toy example is the wall depicted in Figure 3.2, that has 2 levels and 10 bricks in total. Additionally, we perform detailed analysis and comparison on the wall that has 5 levels and 25 bricks in total; 5 levels and 50 bricks in total. The second and third wall examples are more realistic and closer to the one used in the challenge.

Brute force algorithm executes every possible path to go through the tree of wall representation. Based on that fact that it explores every possible solution, brute force will find optimal solution always. The drawback of this algorithm is that is slow and takes time to go through all possible solutions, especially if there is larger number of bricks in the wall.

The main advantage of the Greedy algorithm is the speed of its execution. This algorithm gives the solution in the shortest amount of time from all compared algorithms. But the solution may not be the optimal one in most of the cases. The solution achieved with greedy algorithm is not even in the best 10 solutions of the brute force algorithm. This approach does not guarantee to find the shortest path solution, and may happen that it will not traverse all the nodes in the tree.

Improved greedy algorithm is dependent on the desire if the algorithm is wanted to be faster or to be more optimal. The price of the choosing the algorithm to be faster is that solution will be less optimal, and opposite, if it is wanted to achieve more optimal solution, the algorithm will be slower based on the number of nodes it considers in each of the next steps. The approach will still give better solution than greedy algorithm and faster comparable to brute force algorithm.

The results of the comparison of these approaches for the example of the wall shown on Figure 3.2 are presented in Table 3.3. Comparison is done relative to the brute force algorithm. The algorithms time of execution in relative and absolute values is presented in a table. Furthermore, it is presented the total cost of the best path found by the algorithms. The improved greedy algorithm, is done such that on the first level is chosen as the next node the node with the smallest cost, $n = 1$, and on the second level cases $k = 2$, $k = 3$, $k = 4$, $k = 5$ are considered.

The Table 3.4 shows the result of the comparison of the described approaches for the wall containing 5 levels and 25 bricks in total. The brute force algorithm takes more than 10 minutes to find a solution and based on that it is not considered for comparison. It can be proofed, the algorithm is good for problems of smaller task complexity, but in case of higher task complexity it takes too long to find a solution. The results from the greedy algorithm, as well the improved greedy algorithm for different cases of n and k parameter are presented.

Last Table 3.5 shows the result of the comparison of the approaches for the wall containing 5 levels and 50 bricks in total. The brute force algorithm is not included in comparison again because the solution could not be found up to 10 minutes. The results from greedy algorithm and the improved greedy algorithm for considering different cases of n and k parameters again are presented.

Table 3.3: The algorithms comparisons for the case of the small wall shown on the Figure 3.2. Relative speed denotes relative improvement compared to the brute force algorithm.

Algorithm	*Relative speed*	*Absolute time*$[s]$	*Total cost*
$Bruteforce$	x1	1.5914	336.22
Greedy algorithm	x3183	0.0005	341.16
Improved greedy $[n = 1, k = 2]$	x758	0.0021	340.03
Improved greedy $[n = 1, k = 3]$	x408	0.0039	339.55
Improved greedy $[n = 1, k = 4]$	x408	0.0039	338.87
Improved greedy $[n = 1, k = 5]$	x398	0.0040	338.87

Table 3.4: The algorithms comparisons in case of the wall dimensions $l = 5$, where number of wall levels is denoted as l, and maximum number of bricks in level is 5.

Algorithm	*Absolute time*$[s]$	*Total cost*
$Brute force$	–	–
Greedy algorithm	0.0020	1546.99
Improved greedy $[n = 4, k = 2]$	0.0028	1545.32
Improved greedy $[n = 4, k = 3]$	0.0033	1543.76
Improved greedy $[n = 3, k = 3]$	0.1902	1536.75
Improved greedy $[n = 3, k = 4]$	0.2876	1533.41
Improved greedy $[n = 2, k = 2]$	1.7343	1533.27
Improved greedy $[n = 2, k = 3]$	44.1548	1522.89
Improved greedy $[n = 2, k = 4]$	108.5388	1522.89

Table 3.5: The algorithms comparisons in case of the wall dimensions $l = 5$, where number of wall levels is denoted as l, and maximum number of bricks on each level is 10.

Algorithm	*Absolute time*$[s]$	*Total cost*
$Brute force$	–	–
Greedy algorithm	0.0100	3061.80
Improved greedy $[n = 4, k = 2]$	0.1622	3059.22
Improved greedy $[n = 4, k = 3]$	0.6459	3054.39
Improved greedy $[n = 4, k = 5]$	1.7765	3048.39
Improved greedy $[n = 4, k = 10]$	1.8965	3048.39
Improved greedy $[n = 3, k = 2]$	199.6822	3036.63
Improved greedy $[n = 3, k = 3]$	220.7225	3033.84

Chapter 4

Motion planning

Contents

4.1 Arena description 25

4.2 Mission planner 26

4.3 Communication between UAVs 29

The chapter is organized such that the arena that was used as a testing environment is first introduced in Section 4.1. In the next section, the motion planner presented in the form of the state machine it is described, paying attention on execution of every state in the state machine. As the last part, the task allocation between the UAVs is explained . Especially, two approaches are considered. The first approach is describing the communication channel if the WiFi connection is available. The second approach is telling about the backup plan if communication link can not be established or it failed.

4.1 Arena description

The operating area is divided in six parts. The first part is reserved for drones start positions. Then, four square spots are reserved for the bricks shaped objects. Each of the brick spots is containing just bricks of one color. And the last part of the arena is related to the place where the wall construction is supposed to be built. On the Figure 4.1 can be seen on the left side the sketch of the arena decomposition, and on the right side is shown how the decomposition of the arena looks in the Gazebo simulator.

In the work presented in this thesis, for each UAV is reserved position in which drone can safely wait for performing some action. Namely, in front of the each spot with bricks there is assigned a initial position for drone where it can fly first and hover until it makes a decision about doing the next action. Once the planner for building wall is run

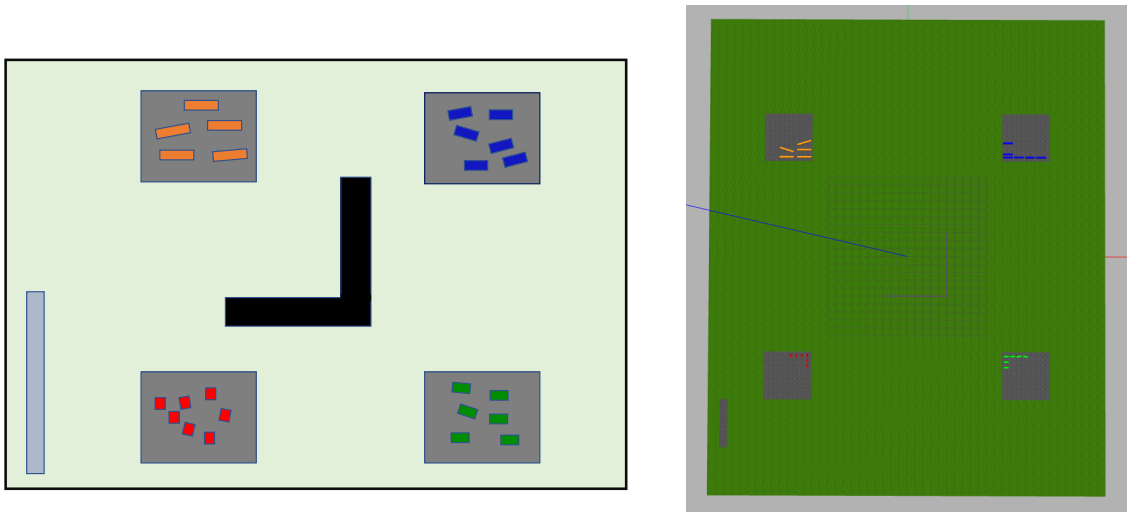


Figure 4.1: Sketch of arena decomposition on the left side and arena decomposition from the simulator on the right side.

the sequence of the bricks how they should be places is known. Based on this sequence of bricks, each UAV knows the brick of which color it is going to take. And flies firstly to initial position close to the spots with objects of that color. Moreover, the wall is shared part for all UAVs. Drones can not operate on the construction at the same time. To be ensured that the collisions of UAVs will be avoided, there are assigned specific position close to the wall for each UAV. These waiting positions in front of the wall are meant to be spots where drones can safely wait for their turn to release the payload. In the Section 4.3 will be minutely described when the drones are allowed to enter in the space of the wall construction.

4.2 Mission planner

Mission planner is done in the form of the state machine. To build a state machine SMACH is used. SMACH is a ROS Python library, that is integrated in the ROS framework [63]. It is allowing the task to be split in finite amount of sub-tasks. SMACH is designed as a set of states. Each state represents a step of execution with some set of potential outcomes. On the Figure 4.2 is shown how one state is presented with its possible outputs.

State machine can be described as graph or diagram. Each state of executions is one node. It means that each node is some action that robot is supposed to do. To be able to traverse from one state to another there are edges connecting all states. Edges are used to express the outcome from the current state. Next state in the state machine will be executed based on the outcome from the current state.

The mission planner of this thesis is presented as a hierarchical state machine. The scheme of the planner is shown on the Figure 4.3. The square blocks are representing the

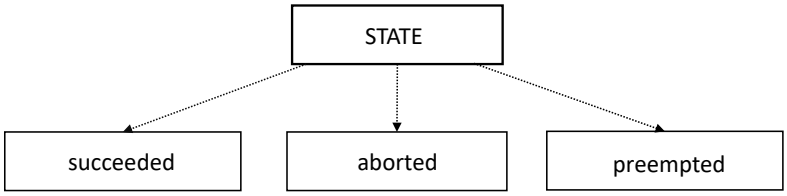


Figure 4.2: General state representation with possible outcomes.

states, and arrows are used to represent possible transitions from one state to the another state.

1. The state machine execution starts with the checking of the state of the UAV. Once the drone is ready to be launched, automatic take off is called. This means the drone is successfully performed the action assigned by this state and it can proceed to the next step.

2. The next state in the planner is reserved for the selection of the color of the object which drone is going to take. Namely, the solution for building a wall explained in the Chapter 3, is used here. That solution provides 2d array of bricks *id*s in which order they should be placed in the wall. Each UAV is assigned for one element of that array. Based on the *id* of the brick that drone is going for, the color is checked. From this moment UAV is responsible for placing brick of that particular color in the wall, and execution of this state is done.

3. Drone is flying to the position where the bricks, of the color him assigned, are located. Here, first the detection of the bricks is done. The bricks are detected based on the color, shape, and their size.

   Once the bricks are detected, their position are estimated. This is done by using known position of the drone in global frame and by projection of detected object from camera to the ground level. Estimation thus provides position of detected object in global frame in which the drone is also controlled.

4. When the drone succeed to estimate the position of the closest brick it flies above it.

5. In the moment when the drone is above the closest brick it is going to grasp it. The grasping itself is a separate state machine, that is using the same strategy as in [7].

   Basically, it is working such that a drone is trying first to align with the object it is trying to grasp. Once UAV is aligned with the object it is descending to the altitude, trying to align again, and slowly descend to the altitude until the magnet is not attached on the object. Once the object is attached, drone can pass to the next step, and perform next action.

   In case the drone does not manage to grasp the object it is going back to the step 3 and does the procedure again.

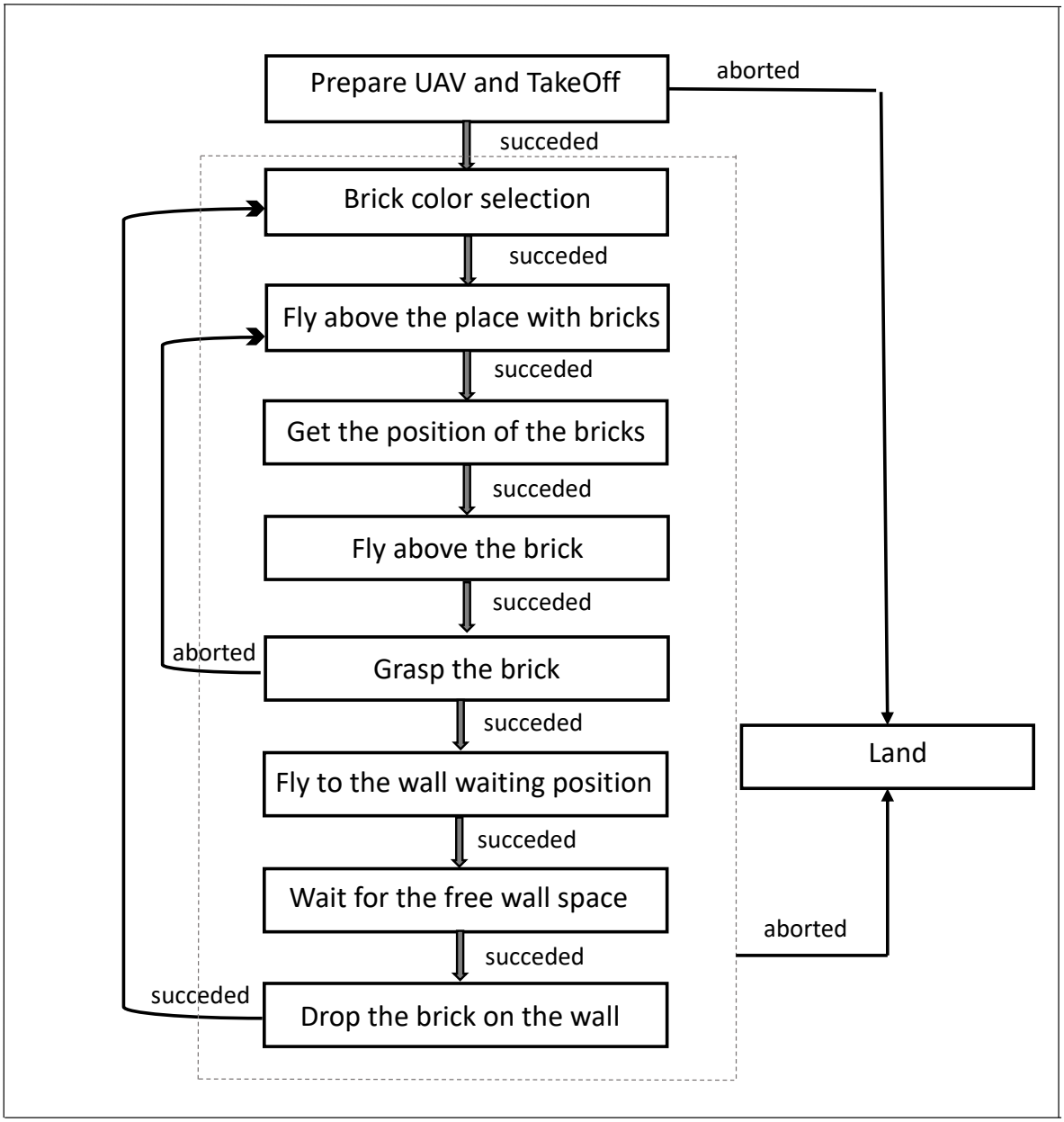


Figure 4.3: Main state machine representing mission planner.

6. When the drone managed to grasp the object drone is flying to the waiting position in front of the wall, see Figure 4.4.

   If the wall is occupied by other UAV, drone keeps waiting in this position until is free. More details about this step in the Section 4.3.

7. Once the wall is not occupied, drone is flying into the wall space and drop the brick. That is done by slowly decreasing the altitude. Once it is on the precised height above the desired location for brick placement it deactivates the magnets and drops the brick.

   If the UAV managed to succeed to finish this step and there is still more bricks to be placed it is going back to step 2, and repeats the procedure. If there are no more bricks the outcome of the state will be aborted and drone will safely land. Aslo, if any step in the state machine went wrong so that is not possible to continue with the mission, safe land is called.

4.3 Communication between UAVs

In order to achieve a cooperation of the UAVs, it is necessary to have reliable communication channel. Having available communication link is especially important in situations when the UAVs are sharing common space, as in our case the wall construction spot, see Figure 4.4. With the available communication link, drones are able to send the messages and share the information. The information that are shared in the exchanged messages are containing: the position of the drone in global coordinate system, the state in the state machine the drone is, and the planned trajectory.

Once the UAV reaches the waiting position in front of the common zone, drone remains hovering until the moment when there is no other UAV in the zone. If the WiFi communication is available, the preference of going into the zone is based on the queue of the UAVs already being in their waiting positions and the brick color the UAV is carrying. The color of the brick that should be placed as the next one is assigned by the wall building bricks sequence. In other words, if this condition is satisfied and the zone is not occupied, the UAV is allowed to access the zone.

Based on our experience from previous MBZIRC competition, it cannot be assumed that complete communication channel will constantly be available. Therefore, it is necessary to make a system able to deal with lack of WiFi communication. So, in case that communication between drones cannot be established or it is lost they need to switch to another strategy. The strategy should be a safe solution to prevent the possible collisions in the case the drones are operating on the common area and cannot exchange information between each other.

The proposed strategy is based on assigning the time windows for each of the drones. Based on the measured time that takes to UAV to perform action of going into the common

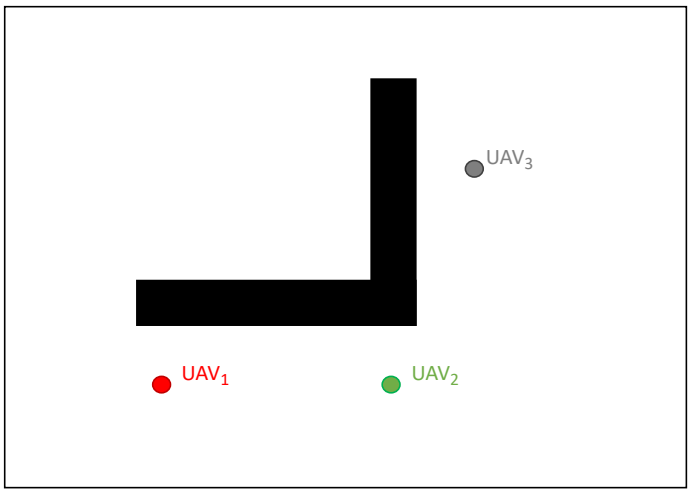


Figure 4.4: The wall waiting positions for UAVs are shown.

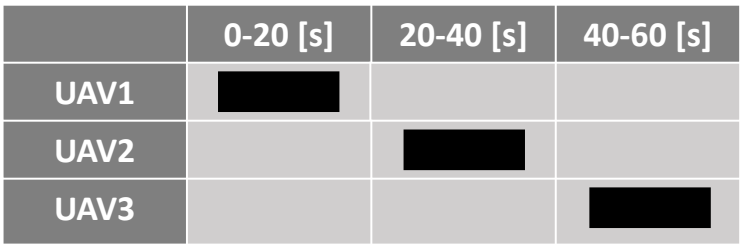


Figure 4.5: The division of time window. Time slots assigned to each of the UAVs in case of communication lack or failure.

space, dropping the payload and leaving it, the time window is divided as it is shown on the Figure 4.5.

For entering, the color of the brick that drone is carrying needs to match with the brick color defined to be placed next by wall building order. This means when the drone reach the waiting position in front of the wall, it is allowed to enter the wall zone only if it is its time slot provided by the time window, and a color of carried brick is defined as next to be placed by high-level motion planner for building the wall. This approach provides a safe strategy in case of issues with communication channel.

Chapter 5

Experiments

Contents

5.1 Validation in simulation 31

5.2 Hardware experiments 32

The experimental results described in this chapter are divided in two parts. The first part is related to the results that are achieved in the simulator, details in Section 5.1. The second part is related to the hardware experiments on the real UAVs.

5.1 Validation in simulation

Testing the approaches that are described in a thesis are done using Robotic operating system (ROS) and Gazebo simulator. ROS is used to enable splitting the task into separate nodes representing subtasks and to allow the communication between the UAVs. The Gazebo simulator is used as testbed for the proposed solution.

5.1.1 Software system structure

Robotic operating system is a flexible framework for writing robot software. ROS is a collection of tools and libraries that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. Different sets of ROS-based processes are represented in an architecture where processing tasks are placed in nodes. So, each node represents one process running in the ROS graph.The existence of nodes simplifies possible complex robotics tasks allowing their splitting in separate subtasks.

The communication between nodes is provided by ROS Master. Every node before doing action first needs to be registered on ROS Master. There is a possibility to exchange

data publishing to a topic, so each node that is subscribing a certain topic is announced when new messages appear on the topic. Moreover, each of the nodes can call services of other nodes, and provide them by themselves. These messages and service calls are not passing through the Master directly, but enables communication between all node processes.

Moreover, to be able to verify the provided solution of the given task, the Gazebo simulator is used as a satisfactory testbed in realistic scenarios. Simulator enables the creation of different 3D scenarios with robots, including a physical engine for illumination, gravity, inertia, etc. The main advantage of using the Gazebo simulator is that the changes verified in the simulator, mostly ensure that in real hardware experiments there should not be any serious problems, and can be expected the same behaviour.

5.1.2 Simulations

The wall that meant to be build in the simulation experiments is containing 2 levels of bricks, with 5 bricks on each level. The algorithms developed in this thesis are tested and the results are shown in Table 5.1. The improved greedy algorithm is run for $n = 1, k = 4$. The experiments are tested in case when the communication link is available.

On the beginning of the scenario drones are spawned on start positions, Figure 5.1, and each of the UAVs is assigned a color of the brick that should be placed on the wall provided by offline computed plan. Based on the conditions for flying are established (operators enable autonomous model), UAVs are flying to their initial waiting positions. Initial waiting positions are located around the spots with bricks. Once the UAV reached this position, it performs next action. Namely, the bricks are detected, their positions are estimated, and then UAV is going to grasp the brick, as it is shown on Figure 5.2.

Once the drones managed to take the objects they are flying with them attached on grippers to the wall waiting positions. UAV is hovering in the waiting position until it is allowed to go inside the wall space, Figure 5.3. When the UAV is allowed to go to wall space, it is going to place the brick on the wall, Figure 5.4. This procedure is repeated until the all objects are placed on the wall, Figure 5.5.

The video of the proposed system contains experiment done in simulator for the first MBZIRC 2020 qualification video report. This video can be found on a link [64]. The motion planning task is done as collaboration of two UAVs. Furthermore, the link contains video from building the wall by the team of three UAVs.

5.2 Hardware experiments

The last part of the experiment section presents real world deployment. These experiments are done with two UAVs. That is sufficient number for presenting the functionality

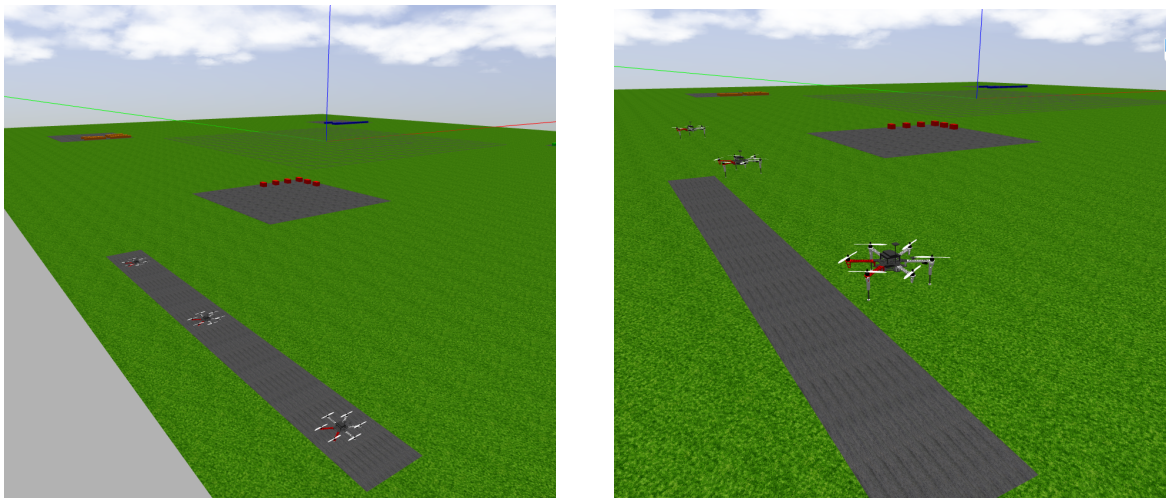


Figure 5.1: UAVs in their start positions in Gazebo simulator.

and also more UAVs were not available at the time of experiments. The UAV platforms are previously used for MBZIRC 2017 competition. These platforms are based on hexacopter DJI F550 frame.

Basic stabilization of the UAV is done by the PixHawk flight control unit. The powerful computer running the ROS is provided on each of the drones. This computer is used for solving the UAV coordination, detection of the objects, state estimation and motion planning. Communication between UAVs is done using WiFi module that is embedded in the PC. Real Time Kinetic(RTK) satellite navigation was employed to achieve high precision in localization. Furthermore, drone is equipped with camera that is providing vision feedback for detection of the objects, and with the laser rangefinder for precise altitude control above the ground. The drones are able to lift up to 2 kg heavy objects, that fits into the competitions requirements.

The hardware experiments were done before the integration of all parts of the solution for the competition (grasping and dropping were not ready). Based on that, just proposed motion planning part is tested. The experiments were prepared in the way that the bricks of three colors were located on three different positions, showing the spots where each type of bricks should be grasped. Furthermore, place where the wall should be build was marked with several objects in the row. At the beginning, the motion planning algorithm is run. The result from the planner is used as a optimal way how to build a wall. The order of

Table 5.1: The algorithms comparisons for the case of the final wall shown on the Figure 5.5. Wall is built by 3 UAVs with established communication.

Algorithm	*Bruteforce*	*Greedy algorithm*	*Improved greedy*
Time[*min*]	5 : 10	5 : 55	5 : 40

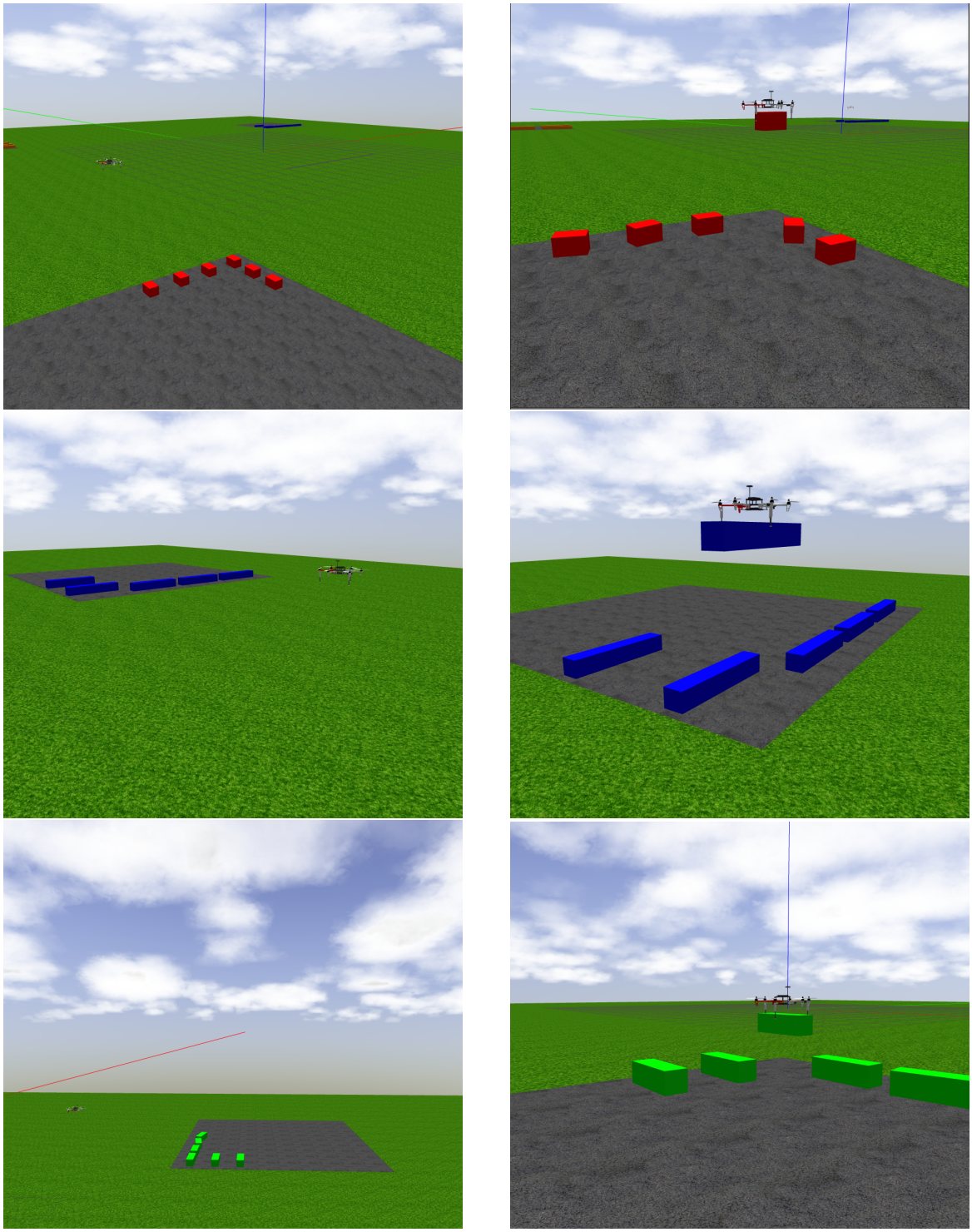


Figure 5.2: UAVs in their initial waiting position close to the spot with color objects they are assigned to carry, are shown on the left images. On the right images UAVs grasping the color objects are shown.

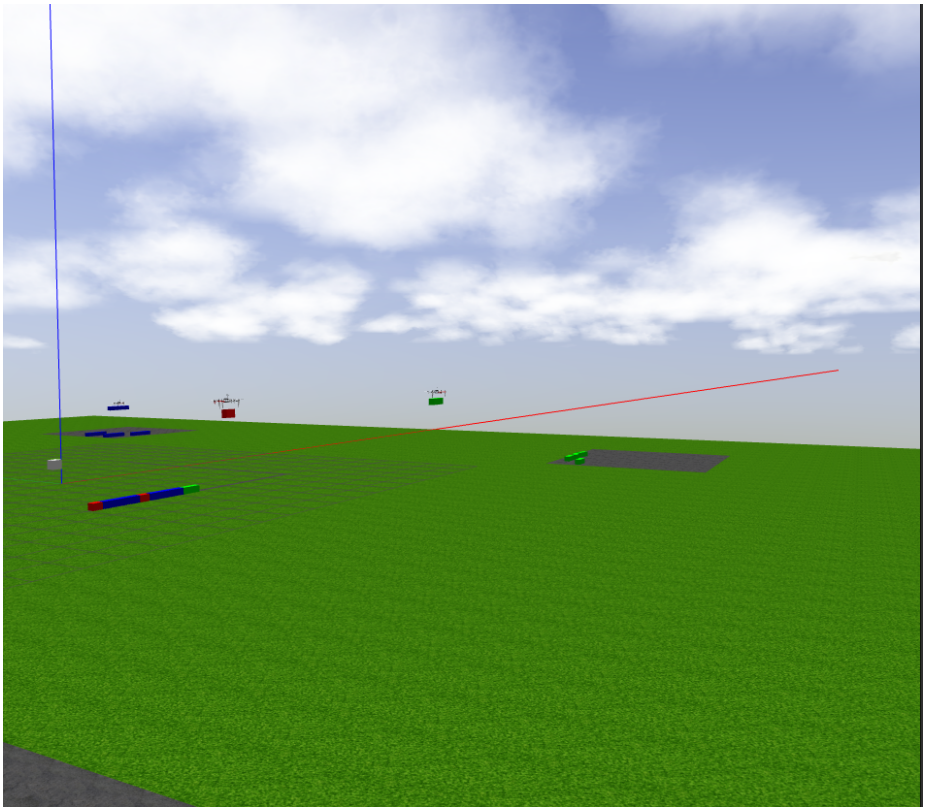


Figure 5.3: UAVs in their waiting positions in front of the wall space, waiting to enter in the wall zone.

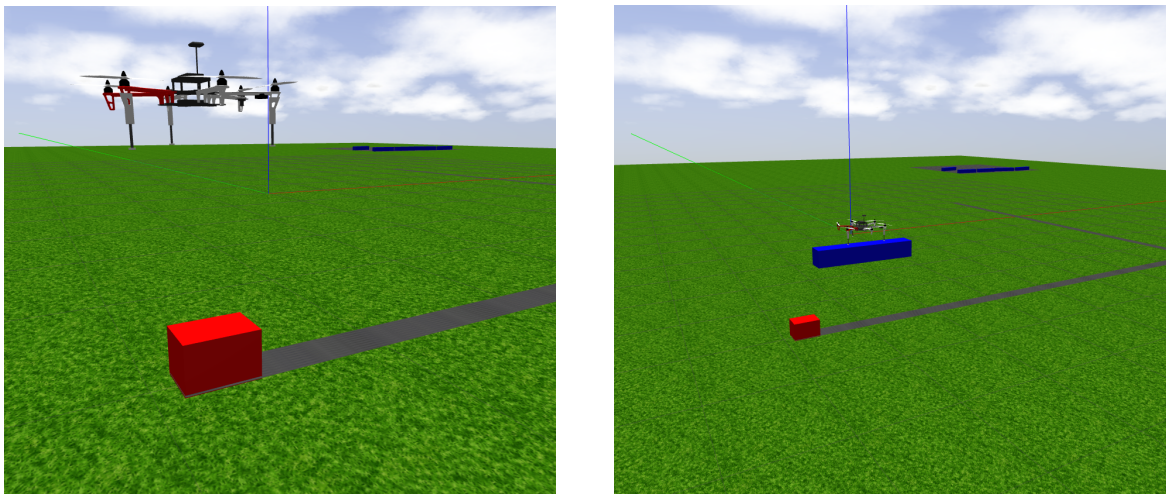


Figure 5.4: UAVs placing bricks in the wall.

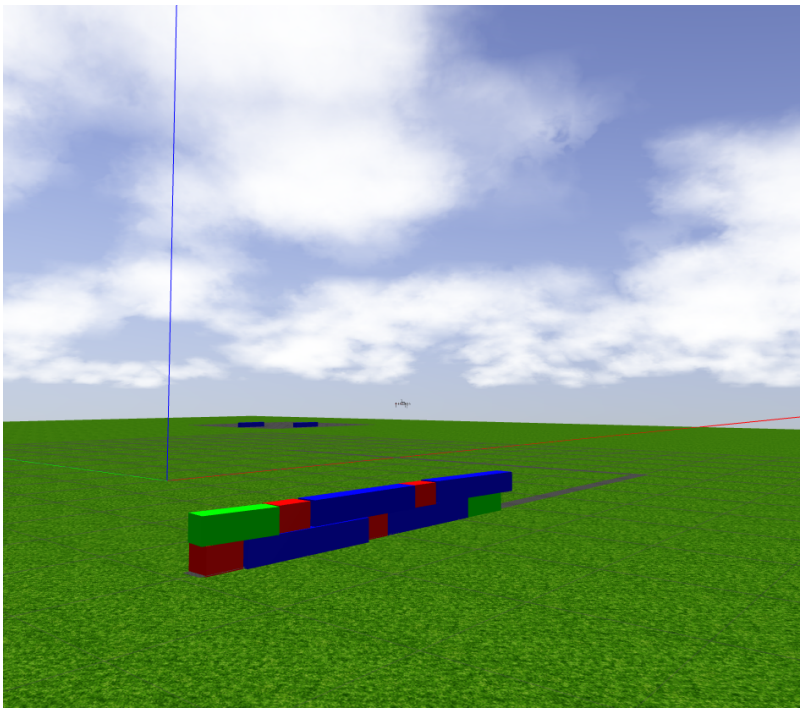


Figure 5.5: The final wall, built by team of three UAVs following the planner provided order of placing bricks.

placing the bricks in the construction was defined in this case as a task for two drones. Once the drone knows which color object it should place on the wall, it is flying to initial waiting position close to the spot with bricks, then goes for grasping, and finally goes to the wall waiting position. There, drone is waiting until its turn to go into the wall zone, and drop the object. Drones are doing this procedure in loop until all bricks are placed. The whole video of the experiments can be seen on the link [64].

As the prototype of the drone for the MBZIRC 2020, was not ready yet for the experiments, the bricks attachment to drone cannot be seen. The aim of the experiment was to show the functionality of motion planner, and to show that drones are satisfactory following the plan mission (given sequence of bricks), and as well execution of the mission.

Figure 5.6: The drone and bricks objects used for experiments.

Figure 5.7: The UAVs flying to get brick shaped objects.

Figure 5.8: The UAVs attempting to grasp the blue and red colored bricks. On the corners of images, the views from the cameras on UAVs can be seen.

Figure 5.9: The UAVs in the waiting positions in front of the wall. The UAV that will enter the wall zone first is carrying the object of the color defined by planner to be placed as the next one in the wall.

Figure 5.10: On the top image, UAV with priority of the color of the object that is carrying, defined by the planner, is entering the wall to deliver the object. After the first UAV left the wall zone, the second UAV is entering the zone. On the left corner of the bottom image can be seen the view from the camera on the drone.

Chapter 6

Concluding remarks and future work

This thesis deals with a problem of high-level motion planning for a group of UAVs that are employed to build a wall structure. Wall structure is composed from different types of brick shaped objects, also denoted as bricks throughout the thesis. Mission planner for such a task is done in a form of a state machine. As the additional part of the mission planner, the communication link between the robots is designed.

The approaches developed in this thesis are tested in the arena environment. Such arena environment contains the starting spots for the drones, spots with colored brick objects, spot where the wall is supposed to be built, and waiting spots assigned to each of the UAVs. An example operating arena is used for the purposes of experimental validation of the developed and proposed approaches, under the Robotics Operating System (ROS).

The wall assembly task is cast as a tree search problem in this thesis. An image of the wall to be build is read and from it, a tree of all possible building orders is constructed. Each node of the tree is representing a brick that will be placed in a specific order. We employ tree search algorithms to find an optimal path from root to leafs, as such path will represent the perfect building order. We implement several algorithms to search the optimal path in the tree: brute force search, greedy algorithm and improved greed algorithm. We conclude that the brute force algorithm, that traverses every path in the tree, is always finding the best possible solution. On the other hand, this algorithms complexity exponentially grows with the size of the wall. We experiment and validate that such an algorithm would use too much time, that is valuable in the competition. The greedy algorithm finds the solution in the shortest amount of time. However, it only explores one possible path in the tree, the one in which the next step is always with smallest cost, and it doesn't guarantee globally optimal solution. Thus, the solution found is almost always not optimal, in fact, we observe it is quite far from the optimal one. Finally, we propose to use the improved version of the greedy algorithm, which will explore more than just a single path, and has adjustable parameters. We conclude that the benefit here is twofold: (i) possibility to find a more optimal solution at the beginning of the challenge, when we have more time available;

(ii) possibility to find a quick solution in the situations when something goes wrong, or if a change is required towards the end of the challenge. The future work would be related to the use of other tree search algorithms and experimenting on how much benefit they would provide time-wise and optimality-wise.

The mission planner is implemented in the form of the hierarchical state machine. The whole planner is tested in Gazebo simulator. The UAV explores the states in the state machine based on the outcome of the current state as it is at the respective moment. If any state execution fails and the drone cannot continue with the mission the safe land is issued. The verification is done with 3 UAVs collaborating together.

In order to be able to establish a proper collaboration between the UAVs two approaches are considered. First, if the communication link using WiFi connection is successfully established, drones are communicating by exchanging the messages between each other. If the communication link is not available, not successfully established, or it fails sometime during the mission, the drones switch to a safe approach of splitting time windows.

The additional work, which is out of the scope of this thesis, being done while performing thesis research, is the contribution for the first MBZIRC report. The motion planning task is implemented as a collaboration of two drones. Full report and video can be seen in [64]. Also, hardware experiments were performed. Because the hardware experiments were done before the full integration of all necessary parts, that are the complete solution for the competition, we tested only the motion planning part.

Bibliography

[1] G. Pajares, "Overview and current status of remote sensing applications based on unmanned aerial vehicles (uavs)," *Photogrammetric Engineering & Remote Sensing*, vol. 81, no. 4, pp. 281–330, 2015.

[2] D. Glade, "Unmanned aerial vehicles: Implications for military operations," Air Univ Press Maxwell Afb Al, Tech. Rep., 2000.

[3] H. Eisenbeiss *et al.*, "A mini unmanned aerial vehicle (uav): system overview and image acquisition," *International Archives of Photogrammetry. Remote Sensing and Spatial Information Sciences*, vol. 36, no. 5/W1, pp. 1–7, 2004.

[4] P. Grippa, D. A. Behrens, C. Bettstetter, and F. Wall, "Job selection in a network of autonomous uavs for delivery of goods," *arXiv preprint arXiv:1604.04180*, 2016.

[5] "Mohamed bin zayed international robotics challenge (mbzirc)." [Online]. Available: http://www.mbzirc.com

[6] "Mohamed bin zayed international robotics challenge video description." [Online]. Available: https://www.youtube.com/watch?v=l5aPjTNYcpc&list=PL1WtPvuCDpc-s4mplf6KNI8BsEH8x7eZl

[7] V. Spurnỳ, T. Báča, M. Saska, R. Pěnička, T. Krajník, J. Thomas, D. Thakur, G. Loianno, and V. Kumar, "Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles," *Journal of Field Robotics*, vol. 36, no. 1, pp. 125–148, 2019.

[8] J. Faigl, P. Vana, R. Penicka, and M. Saska, "Unsupervised Learning based Flexible Framework for Surveillance Planning with Aerial Vehicles," *Accepted in Journal of Field Robotics*, 2018.

[9] G. Loianno, V. Spurny, T. Baca, J. Thomas, D. Thakur, T. Krajnik, A. Zhou, A. Cho, M. Saska, and V. Kumar, "Localization, grasping, and transportation of magnetic objects by a team of mavs in challenging desert like environments," *IEEE Robotics and Automation Letters*, 2018.

[10] M. Saska, V. Spurny, and V. Vonasek, "Predictive control and stabilization of nonholonomic formations with integrated spline-path planning," *Robotics and Autonomous Systems*, vol. 75, no. Part B, pp. 379–397, 2016.

[11] M. Saska, V. Vonásek, J. Chudoba, J. Thomas, G. Loianno, and V. Kumar, "Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles," *Journal of Intelligent & Robotic Systems.*, vol. 84, no. 1, pp. 469–492, 2016.

[12] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[13] M. Saska, Z. Kasl, and L. Preucil, "Motion Planning and Control of Formations of Micro Aerial Vehicles," in *19th World Congress of the International Federation of Automatic Control (IFAC)*. IFAC, 2014.

[14] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[15] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," in *Cooperative Robots and Sensor Networks 2015*. Springer, 2015, pp. 31–51.

[16] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[17] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multirobot coordination," *IEEE transactions on robotics and automation*, vol. 18, no. 5, pp. 758–768, 2002.

[18] A. Farinelli, L. Iocchi, and D. Nardi, "Multirobot systems: a classification focused on coordination," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 5, pp. 2015–2028, 2004.

[19] J. van der Horst and J. Noble, "Distributed and centralized task allocation: When and where to use them," in *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop*. IEEE, 2010, pp. 1–8.

[20] X. Jia and M. Q.-H. Meng, "A survey and analysis of task allocation algorithms in multi-robot systems," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2013, pp. 2280–2285.

[21] H. Hanna, "Decentralized approach for multi-robot task allocation problem with uncertain task execution," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 535–540.

[22] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating uavs," in *Cooperative control: models, applications and algorithms*. Springer, 2003, pp. 23–41.

[23] J.-P. Laumond *et al.*, *Robot motion planning and control*. Springer, 1998, vol. 229.

[24] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous uav guidance," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1-4, p. 65, 2010.

[25] K. H. Sedighi, K. Ashenayi, T. W. Manikas, R. L. Wainwright, and H.-M. Tai, "Autonomous local path planning for a mobile robot using a genetic algorithm," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, vol. 2. IEEE, 2004, pp. 1338–1345.

[26] A. Ismail, A. Sheta, and M. Al-Weshah, "A mobile robot path planning using genetic algorithm in static environment," *Journal of Computer Science*, vol. 4, no. 4, pp. 341–344, 2008.

[27] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[28] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *International journal of physical sciences*, vol. 7, no. 9, pp. 1314–1320, 2012.

[29] D.-q. Zhu and M.-z. Yan, "Survey on technology of mobile robot path planning," *Control and Decision*, vol. 25, no. 7, pp. 961–967, 2010.

[30] A. Kaufmann, "Graphs dynamic programming and finite games," Tech. Rep., 1967.

[31] D. Shasha, J. T. Wang, and R. Giugno, "Algorithmics and applications of tree and graph searching," in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2002, pp. 39–52.

[32] S. A. Bortoff, "Path planning for uavs," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 1, no. 6. IEEE, 2000, pp. 364–368.

[33] J. D. Contreras, F. Martínez *et al.*, "Path planning for mobile robots based on visibility graphs and a* algorithm," in *Seventh International Conference on Digital Image Processing (ICDIP 2015)*, vol. 9631. International Society for Optics and Photonics, 2015, p. 96311J.

[34] T. Kito, J. Ota, R. Katsuki, T. Mizuta, T. Arai, T. Ueyama, and T. Nishiyama, "Smooth path planning by using visibility graph-like method," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 3. IEEE, 2003, pp. 3770–3775.

[35] W. Y. Loong, L. Z. Long, and L. C. Hun, "A star path following mobile robot," in *2011 4th International conference on mechatronics (ICOM)*. IEEE, 2011, pp. 1–7.

[36] M. Kothari, I. Postlethwaite, and D.-W. Gu, "Multi-uav path planning in obstacle rich environments using rapidly-exploring random trees," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 3069–3074.

[37] M. Kothari and I. Postlethwaite, "A probabilistically robust path planning algorithm for uavs using rapidly-exploring random trees," *Journal of Intelligent & Robotic Systems*, vol. 71, no. 2, pp. 231–253, 2013.

[38] Y. Dong, C. Fu, and E. Kayacan, "Rrt-based 3d path planning for formation landing of quadrotor uavs," in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2016, pp. 1–6.

[39] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, vol. 1. IEEE, 2000, pp. 256–263.

[40] Y.-b. Chen, G.-c. Luo, Y.-s. Mei, J.-q. Yu, and X.-l. Su, "Uav path planning using artificial potential field method updated by optimal control theory," *International Journal of Systems Science*, vol. 47, no. 6, pp. 1407–1420, 2016.

[41] S. J. Rasmussen and T. Shima, "Branch and bound tree search for assigning cooperating uavs to multiple tasks," in *2006 American Control Conference*. IEEE, 2006, pp. 6–pp.

[42] ——, "Tree search algorithm for assigning cooperating uavs to multiple tasks," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 18, no. 2, pp. 135–153, 2008.

[43] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[44] M. A. Kaur, M. P. Sharma, and M. A. Verma, "A appraisal paper on breadth-first search, depth-first search and red black tree," *International Journal of Scientific and Research Publications*, vol. 4, no. 3, pp. 1–3, 2014.

[45] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[46] M. Kurant, A. Markopoulou, and P. Thiran, "On the bias of bfs (breadth first search)," in *2010 22nd International Teletraffic Congress (lTC 22)*. IEEE, 2010, pp. 1–8.

[47] M. Alighanbari, "Task assignment algorithms for teams of uavs in dynamic environments," Ph.D. dissertation, Massachusetts Institute of Technology, 2004.

[48] B. M. Sathyaraj, L. C. Jain, A. Finn, and S. Drake, "Multiple uavs path planning algorithms: a comparative study," *Fuzzy Optimization and Decision Making*, vol. 7, no. 3, p. 257, 2008.

[49] M. Shanmugavel, A. Tsourdos, B. White, and R. Żbikowski, "Co-operative path planning of multiple uavs using dubins paths with clothoid arcs," *Control Engineering Practice*, vol. 18, no. 9, pp. 1084–1092, 2010.

[50] P. Ramon Soria, B. Arrue, and A. Ollero, "Detection, location and grasping objects using a stereo sensor on uav in outdoor environments," *Sensors*, vol. 17, no. 1, p. 103, 2017.

[51] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[52] N. Roy, P. Newman, and S. Srinivasa, "Recognition and pose estimation of rigid transparent objects with a kinect sensor," *Robotics: Science and Systems*, 2013.

[53] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, "Single image 3d object detection and pose estimation for grasping," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3936–3943.

[54] P. Ramon Soria, B. Arrue, and A. Ollero, "Detection, location and grasping objects using a stereo sensor on uav in outdoor environments," *Sensors*, vol. 17, no. 1, p. 103, 2017.

[55] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 48–55.

[56] M. Laiacker, M. Schwarzbach, and K. Kondak, "Automatic aerial retrieval of a mobile robot using optical target tracking and localization," in *2015 IEEE Aerospace Conference*. IEEE, 2015, pp. 1–7.

[57] M. Bisgaard, J. D. Bendtsen, and A. L. Cour-Harbo, "Modeling of generic slung load system," *Journal of guidance, control, and dynamics*, vol. 32, no. 2, pp. 573–585, 2009.

[58] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous Robots*, vol. 30, no. 1, pp. 73–86, 2011.

[59] M. Bernard and K. Kondak, "Generic slung load transportation system using small size helicopters," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3258–3264.

[60] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, "Design, modeling, estimation and control for aerial grasping and manipulation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 2668–2673.

[61] P. Williams and P. Trivailo, "Cable-supported sliding payload deployment from a circling fixed-wing aircraft," *Journal of aircraft*, vol. 43, no. 5, pp. 1567–1570, 2006.

[62] P. McGill, K. Reisenbichler, S. Etchemendy, T. Dawe, and B. Hobson, "Aerial surveys and tagging of free-drifting icebergs using an unmanned aerial vehicle (uav)," *Deep Sea Research Part II: Topical Studies in Oceanography*, vol. 58, no. 11-12, pp. 1318–1326, 2011.

[63] J. Bohren and S. Cousins, "The smach high-level executive [ros news]," *IEEE Robotics & Automation Magazine*, vol. 17, no. 4, pp. 18–20, 2010.

[64] "Video material from the experimental verification of proposed method." [Online]. Available: http://mrs.felk.cvut.cz/theses/petric2019

Appendices

CD Content

In Table 1 are listed names of all root directories on CD.

Directory name	**Description**
thesis	the thesis in pdf format
thesis_latex	latex source codes
source_codes	implementation source codes

Table 1: CD Content

List of abbreviations

In Table 2 are listed abbreviations used in this thesis.

Abbreviation	Meaning
UAV	Unmaned Aerial Vechicle
UGV	Unmaned Ground Vechicle
MBZIRC	Mohamed Bin Zayed International Robotics Challenge
ROS	Robot Operating System
BFS	Breadth First Search
RRT	Rapidly-exploring Random tree
DFS	Depth First Search

Table 2: Lists of abbreviations