CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR THESIS

Tomáš Brich

Motion Planning for Swarms of Unmanned Helicopters in Complex Environment

Department of Cybernetics

Thesis Supervisor: **Ing. Martin Saska, Dr. rer. nat.**

PRAGUE, MAY 2016

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Tomáš B r i c h

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Bachelor Project: Motion Planning for Swarms of Unmanned Helicopters in Complex Environment

Guidelines:

The goal of the thesis is to design, implement and experimentally verify a method of optimal path planning for large swarms of unmanned helicopters in complex environment. The following main tasks will be solved in the thesis.

- To understand the system of Multi-Robot Systems group at CTU being developed for control of swarms of Micro Aerial Vehicles (MAVs) [1] that are stabilized using onboard relative localization [4].
- To implement an optimal path planning algorithm based on Voronoi graph that will be suited for the bio-inspired swarm stabilization system [1].
- To design and implement an approach for evaluation of edges of the Voronoi graph based on the shape of the swarm and the environment.
- To extend the system with ability of splitting and merging of swarms based on results of the high level planning.
- To integrate the methods and verify their performance in the V-REP simulator with complex polygonal maps. In case of availability of the HW platform, to verify basic swarming abilities with real MAVs (thesis advisor will decide whether the experiment or more detailed analyses in the simulator will be conducted).

Bibliography/Sources:

[1] M. Saska, J. Vakula and L. Preucil: Swarms of Micro Aerial Vehicles Stabilized Under a Visual Relative Localization. In ICRA2014: Proceedings of 2014 IEEE International Conference on Robotics and Automation. 2014.
[2] M. Kumar, D. Garg, and V. Kumar: "Segregation of heterogeneous units in a swarm of robotic agents", IEEE Transactions on Automatic Control, vol. 55, no. 3, pp. 743-748, 2010.
[3] D. J. Bennet and C. R. McInnes: "Verifiable control of a swarm of unmanned aerial vehicles", Journal of Aerospace Engineering, vol. 223, no. 7, pp. 939-953, 2009.
[4] T. Krajnik, M. Nitsche, J. Faigl, P. Vanek, M. Saska, L. Preucil, T. Duckett and M. Mejail: A Practical Multirobot Localization System. Journal of Intelligent & Robotic Systems 76(3-4):539-562, 2014.

Bachelor Project Supervisor: Ing. Martin Saska, Dr. rer. nat.

Valid until: the end of the summer semester of academic year 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 14, 2016

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 theses.

Prague, date.....................................

Signature:.....................................

Acknowledgements:

Firstly, I would like to thank my thesis supervisor Dr. Martin Saska for a lot of good advice and his kind and friendly approach during our meetings. Further I would like to thank Dr. Miroslav Kulich for providing me with his Voronoi diagram algorithm and a path planning algorithm for swarm splitting for use in this thesis. Finally, I would like to thank my friends and my family for all the support during my studies.

Abstract

The goal of this thesis is to design and implement a method of optimal path planning for large swarms of unmanned aerial vehicles (UAVs, i.e. quadrotors) in a complex environment. The work is based on the Boids swarming model and tested using simulations in the V-Rep simulator. The planning algorithm is based on the Voronoi graph, which is created around environmental obstacles. This thesis describes the Boids model implementation, extended by simple obstacle avoidance and path following capabilities. It also describes the process of the graph's edges evaluation using an experimentally acquired heuristic and evaluation using simulation and compares these two approaches. Additionaly, the model is extended with the possibility of swarm splitting and merging.

Keywords

swarm, UAV, Boids model, path planning

Abstrakt

Tato práce se zabývá návrhem a implementací metody optimálního plánování trasy pro roj bezpilotních letounů (UAV, i.e. kvadrokoptér) v komplexním prostředí. Práce je založena na Boids modelu roje a testována pomocí simulací v simulátoru V-Rep. Plánovací algoritmus je založen na Voroného grafu, který je vytvořen okolo překážek v daném prostředí. Tato práce popisuje implementaci Boids modelu, rozšířeného o jednoduché vyhýbání se překážkám a sledování trasy. Dále popisuje proces ohodnocování hran grafu pomocí experimentálně získané heuristiky a ohodnocování pomocí simulace a porovnává tyto dva přístupy. Model je dále rozšířen o možnost dělení roje a jeho opětovném slučování.

Klíčová slova

roj, UAV, Boids model, plánovaní trasy

Contents

List of Figures

List of Tables

List of Abbreviations

CPP Chinese Postman Problem

CTU Czech Technical University (in Prague)

GPS Global Positioning System

LSM Least Squares Method

MAV Micro Aerial Vehicle

MRS Multi-robot Systems group

UAV Unmanned Aerial Vehicle (also referred to as quadrotor or boid in this work)

1 Chapter 1. Introduction

The concept of using unmanned Micro Aerial Vehicles (MAVs) for solving various everyday tasks in many different fields has received a lot of attention. Mostly thanks to their affordability and their simple handling, the use of MAVs such as quadrotors (i.e. a helicopter with four propellers) is on a rise for personal purposes, currently mostly used for high quality aerial video shooting.

Appart from recreational purposes, the possibility of deployment of large groups of MAVs working on a collective task has many potential applications in a number of fields such as surveillance [15], reconnaissance, search and rescue operations, searching for sources of pollutions or gas leaks [12, 13] and various military applications.

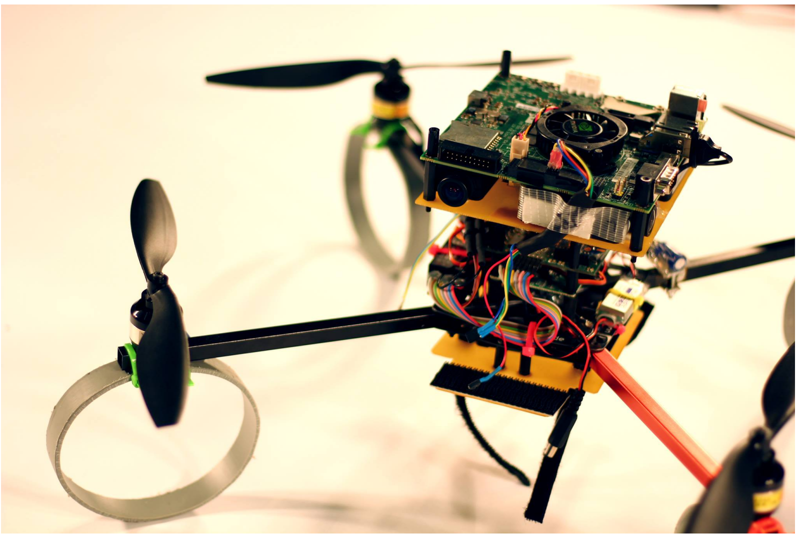


Figure 1.1.: One of the quadrotors used by MRS

Most of these tasks involve the utilization of groups of MAVs in environments with a very limited possibilities of precise localization of robots. The available global localization systems (such as GPS or visual-based localization [14]) are insufficient for determining a precise position of each quadrotor. These global localization systems are useful for path following capabilities of the swarm, where a high precision is not needed and the errors caused by such systems can even be higher than the distances between quadrotors in the group. However, more precise relative localization systems are needed for internal control of the quadrotors within a team.

One of the core research activities of the Multi-robot Systems (MRS) group [11] at Czech Technical University in Prague is the swarm robotics [19, 20], which is inspired by a behavioral model of flocks of birds. These approaches rely on an onboard visual system for relative localization of the swarm members [9, 10], which was already succesfully employed in experimental emulation of numerous formation flying [21–23] and swarm experiments [17, 24]. This relative localization is much more precise than GPS and it is crucial for collision avoidance within the group of quadrotors.

This work builds on achievements described in [19, 20] and brings an additional contribution in sense of unique trajectory planning that considers the size of the team and the environment characteristics (density of obstacles, width of corridors and narrow passages, etc.).

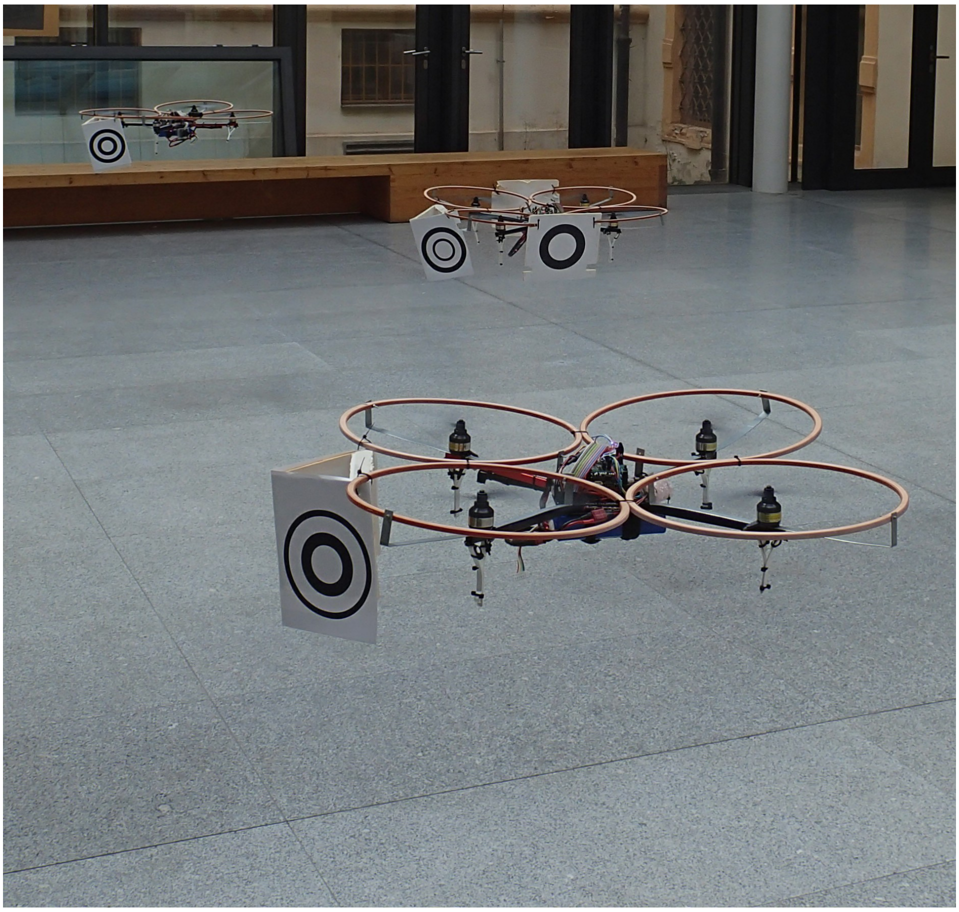


Figure 1.2.: Real world swarm of quadrotors using relative visual localization

As it is often impossible or too hardware demanding to plan a path for the swarm onboard the quadrotors, it is needed to plan the trajectory for the swarm in the given environment in advance. This work introduces a path planning method for large swarms of MAVs in a complex environment, which is based on the Voronoi graph [3]. The Boids model [2] presented by Craig Reynolds in 1986 was used as a swarming model, furtherly extended by simple obstacle avoidance and path following capabilities, as described in chapter 2. The process of the Voronoi graph evaluation is described in chapter 3. Additionaly, the system is extended with the possibility of swarm splitting and merging discussed in chapter 4.

The performace of the extended Boids model and the resulting graph evaluation is verified using a large number of simulations using the V-Rep simulator [8]. This simulator uses a simple proportional position regulator, which steers each quadrotor towards their individual targets. Videos of some of the experiments can be found on the attached CD.

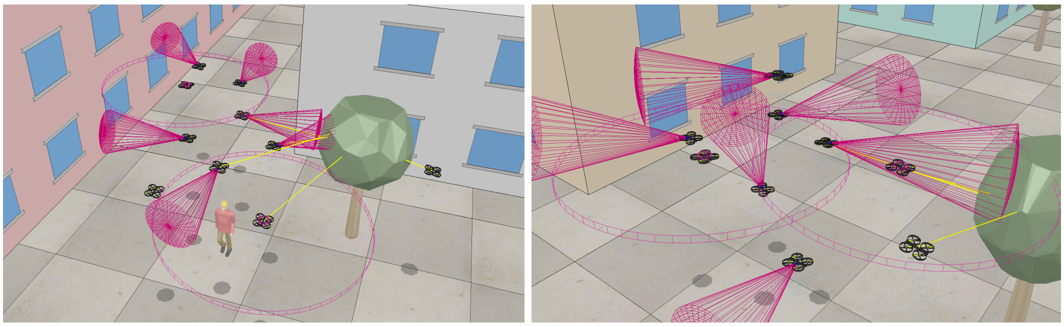


Figure 1.3.: A *Smart City* simulation of a swarm of quadrotors in the V-Rep simulator

$n \in \mathbb{N}$	Number of boids
b_i	The i-th boid
$B \coloneqq \{b_1, \ldots, b_n\}$	Set of boids (i.e. MAVs in the swarm)
$p_i \in \mathbb{R}^3$	Position of the i-th boid
$\vec{v}_i \in \mathbb{R}^3$	Velocity of the i-th boid
$r \in \mathbb{R}_0^+$	Sensoric range of all boids
R_i	Set of the detected neighbors for the i-th boid defined in 2.1
P_i	Set of boids in a platoon for the i-th boid defined in 4
W_i	Set of the waypoints for the i-th quadrotor
$w_i \in \mathbb{R}^3$	Position of the current waypoint of the i-th boid
$w_i' \in \mathbb{R}^3$	Position of the previous waypoint of the i-th boid
$d_i \in \mathbb{R}^3$	Closest point on an obstacle detected by the i-th boid
$\vec{F}_{s_i}, \vec{F}_{c_i}, \vec{F}_{a_i}, \vec{F}_{p_i}, \vec{F}_{o_i} \in \mathbb{R}^3$	Separation, cohesion, alignment, path following and obstacle avoidance forces for the i-th boid
$k_s, k_c, k_a, k_p, k_o \in \mathbb{R}_0^+$	Weights of each of the forces above
$\hat{\vec{F}}_i \in \mathbb{R}^3$	Output force without obstacle avoidance force for the i-th boid
$\vec{F}_i \in \mathbb{R}^3$	Resulting output force for the i-th boid
$s \in \mathbb{R}_0^+$	Distance between two obstacles in meters
$l_{ij} \in \mathbb{R}_0^+$	Length of the graph edge between the i-th and the j-th vertex
$t_r \in \mathbb{R}_0^+$	Reference time described in section 3.2.1
$\varepsilon(n, s, k_p, l_{ij}) \in \mathbb{R}_0^+$	Heuristic function used for the graph evaluation
$\varepsilon_r(k_p, l_{ij}) \in \mathbb{R}_0^+$	First part of the heuristic function based on the reference time t_r
$\varepsilon_i(n, s) \in \mathbb{R}_0^+$	Second part of the heuristic function based on the time increments caused by the obstacles
$K \in \mathbb{R}_0^+$	Auxiliary constant used in ε_r, described in section 3.2.1
$a_1(n), a_2(n) \in \mathbb{R}_0^+$	Auxiliary function parameters used in ε_i, described in section 3.2.1

Table 1.1.: List of used symbols and variables

Chapter 2.

2 Extended Boids model

In this chapter, the core of the UAV control is discussed. Boids swarming model was used, subsequently extended by a simple obstacle avoidance force and a force that enables the swarm of UAVs to follow a given path. The model is described using the global coordinates system instead of relative positions for each UAV to maintain the clarity of used symbols. However, the Boids model together with the obstacle avoidance force use relative localization systems as described in chapter 1.

2.1. Boids model

The Boids model, introduced by Craig Reynolds in [2] describes the swarming (flocking) behavior as a combination of three simple steering forces of each individual member of the swarm (i.e. boid). These forces are separation, cohesion and alignment.

The swarm B consists of n boids. Each boid b_i is defined by its position p_i and its current velocity $\vec{v}_i$. Boids have a limited sensoric range for their relative localization and obstacle avoidance, which is represented by a range constant r. All boids have the same sensoric range and they react only to those boids that are within their range. The set of detected neighbors for i-th boid R_i can then be written as

$$R_i = \{b_j \in B \setminus \{b_i\}\,; \forall b_j : \|p_i - p_j\| < r\}\,.$$

2.1.1. Separation force

Each boid is forced to stay away from other detected boids and to avoid potential collisions with other members of the swarm. This tendency is called the separation force $\vec{F}_s$ and for the i-th boid is computed as

$$\vec{F}_{s_i} = \sum_{\forall b_j \in R_i} \frac{p_i - p_j}{\|p_i - p_j\|^2}.$$

Each vector is divided by the square of its norm so that the closer the j-th boid is, the bigger force pulls the i-th boid away from it.

2.1.2. Cohesion force

The cohesion force $\vec{F}_c$ is oriented against the separation force and keeps the swarm members together. It pulls each boid towards the center of all detected boids and is computed as

$$\vec{F}_{c_i} = \frac{1}{|R_i|} \sum_{\forall b_j \in R_i} p_j - p_i.$$

If there are no boids detected by the i-th boid (i.e. $|R_i| = 0$), the cohesion force is set to $\vec{0}$.

2.1.3. Alignment force

The alignment force $\vec{F}_a$ makes each boid to follow other boids movements and to match its velocity to the other boids. This behavior can simply be obtained by averaging the vector velocities of all detected boids as

$$\vec{F}_{a_i} = \frac{1}{|R_i|} \sum_{\forall b_j \in R_i} \vec{v}_j.$$

Again, if there are no boids detected by the i-th boid, the alignment force is set to $\vec{0}$.

2.2. Path following force

With the Boids model alone, the swarm wanders randomly in the environment. For the purpose of making the swarm to follow a given path, the path following force $\vec{F}_p$ was introduced. It directs the UAVs towards a single point on the given path. The points in the path are switched as the swarm progresses towards the end of the path. The given path points then serve as waypoints to steer the quadrotors. The force is computed as

$$\vec{F}_{p_i} = \frac{w_i - p_i}{\|w_i - p_i\|},$$

where w_i is the position of the current waypoint of the i-th UAV. The force is normalized to unit length independently to the distance to the waypoints.

In this work, two different approaches were used for following the path: the *leader-followers* approach and the *all UAVs following path* approach.

2.2.1. Leader-followers approach

In the *leader-followers* approach, the path following force is applied to only one chosen UAV (a leader). The leader is chosen as the closest quadrotor to the first waypoint on the path. The rest of the swarm then follows the leader due to the cohesion force of the Boids model. When the leader gets close to its target, it switches its target to the next waypoint.

The advantage of this approach is that the path following force does not interfere with the Boids model, which makes the swarm better organized. On the other hand, the swarm moves slower, because the leader has to pull the whole swarm only using the cohesion force. This problem is especially amplified when the swarm is supposed to fly through a narrow corridor, since then also the obstacle avoidance force (described in section 2.3) works against the path following force. Therefore, the *leader-followers* approach is better suited for smaller swarms (i.e. three UAVs or less), where the cohesion force that slows down movement of the leader is smaller.

2.2.2. All UAVs following path approach

In this approach, all quadrotors use the path following force. Each quadrotor has own individual target for its steering. Again, when the particular MAV reaches its target, the following waypoint is chosen.

The advantage of this approach is that each member of the swarm flies through the given path, which is useful in maze-like environments, where the swarm cannot spontaneously split when they encounter a narrow corridor that is hard to fly through. Another advantage is that the MAV can fly towards the goal or to the rest of the team if it is splitted from the group (i.e. it can no longer detect any other quadrotors).

The disadvantage of this approach is that it interferes with the Boids model, because all the quadrotors fly towards a single waypoint, which works against the separation force. It is only a small disadvantage, since the closer the UAVs are, the bigger is the separation force and it only results in the quadrotors flying closer to each other.

The waypoint switching process cannot be done based only on the distance to their target as in *leader-followers* approach. This is due to the fact that in larger swarms, the quadrotors on the edge of the swarm never get close enough to the waypoints, but they still need to switch their target to the next waypoint. Because of this, an additional waypoint switching method was introduced. For this method, vector $\vec{u}_w = w_i^{'} - w_i$, where $w_i^{'}$ is the previous waypoint of the i-th quadrotor, and vector $\vec{u}_q = p_i - w_i$ are used. Additionaly to the target switching based on the distance to the current target, the target is also switched to the next waypoint when the angle between those two vectors is bigger than $\frac{\pi}{2}$. Now it does not matter how far the quadrotor is to its current target. It switches to the next waypoint when it passes the target.

Since the *leader-followers* approach is not suitable for larger swarms, the *all UAVs following path* approach is used as the main path following method in the rest of this work.

2.3. Obstacle avoidance force

As each member of the swarm needs to avoid the obstacles in the workspace, a simple obstacle avoidance force $\vec{F}_o$ was introduced in to the model. It pulls each UAV away from the closest detected obstacle and is computed as

$$\vec{F}_{o_i} = \frac{p_i - d_i}{\|p_i - d_i\|^2},$$

where d_i is the closest detected point by the i-th UAV. The vector is divided by the square of the distance to the obstacle, so that the obstacle avoidance force is stronger when the detected obstacle is closer.

2.4. Output force

The output force $\vec{F}$ that controlls each UAV is computed in two steps, first without the obstacle avoidance force as

$$\hat{\vec{F}_i} = k_s\vec{F}_{s_i} + k_c\vec{F}_{c_i} + k_a\vec{F}_{a_i} + k_p\vec{F}_{p_i},$$

where k_s, k_c, k_a and k_p are weights of each force. These constants can be set by the user to meet the desired properties of the swarm. Changing the separation and cohesion constants k_s

and k_c changes the distance between quadrotors at which the separation and cohesion forces are balanced. Increasing these constants also increases the reaction of quadrotors when they deviate from the equilibrium. Decreasing the alingment constant k_a makes the swarm movement more chaotic, because each member of the swarm will react less to the direction of other quadrotors around them. Since the alignment force interferes with the obstacle avoidance, it is better to set the alignment constant to a small number compared to the other constants, or it can be set to zero, which cancels the alignment force completely. The path following constant k_p changes the speed of quadrotors while following the path. It should not be set too high compared to the other forces, because it would cause the quadrotors to crash into obstacles or each other while trying to get closer to their current target.

Now, let us define angle α between $\hat{\vec{F}_i}$ and the obstacle avoidance force $\vec{F}_{o_i}$. The obstacle avoidance force is then added to the output force only if $\alpha < \frac{\pi}{2}$. This ensures that the members of the swarm ignore obstacles that are already behind them while following the path. If influence of these obstacles is added into the output force, the quadrotors tend to accelerate excessively because of the cumulative efect of the obstacle avoidance force and the path following force, which both are added into the standard Boids model. The resulting output force for the i-th boid is computed as

$$\vec{F}_i = \hat{\vec{F}_i} + k_o \vec{F}_{o_i},$$

where k_o is a weigth constant for obstacle avoidance force, which can be set by the user. It should not be set too low to ensure that the quadrotors do not crash into obstacles, but at the same time, setting this parameter too high might result in the swarm behavior that does not allow to fly through narrow corridors such as doors.

Chapter 3.

Path planning

The path planning process is splitted into three steps. First a 2D graph is created around obstacles in the environment using the Voronoi [3] diagram algorithm. Then the Voronoi graph edges are evaluated based on the swarm properties. After the graph evaluation, it is searched for one or more shortest paths around obstacles. For the purpose of graph searching, the K-Shortest path routing algorithm was used. The implementation of the algorithm can be seen in [1].

3.1. Voronoi diagram

For fast computing of the Voronoi diagram algorithm, implementaion developed at CTU was used. This approach is based on the C++ Boost [4] library and its Voronoi diagram implementation, further extended by the possibility of using polygon shapes as inputs, which makes the algorithm more suitable for real world obstacles. In this work, the bounding box edges of each obstacle are used as an input for the Voronoi algorithm. Examples of the used V-Rep simulator [8] environments and the resulting Voronoi diagrams can be seen on figures 3.1 and 3.2.

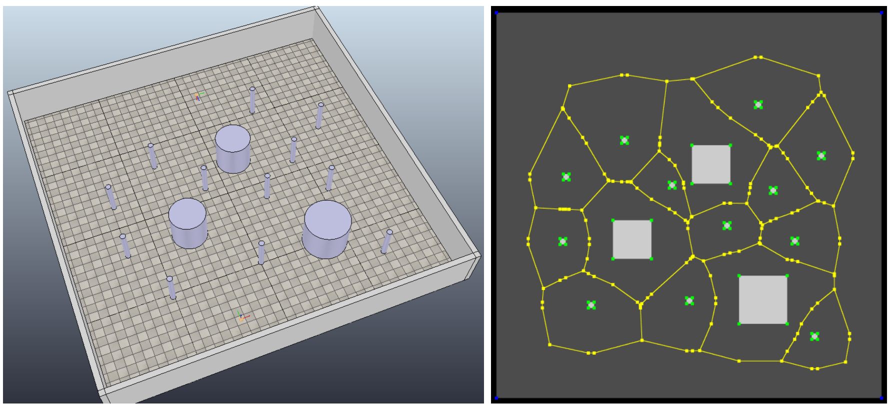


Figure 3.1.: Simple environment in the V-Rep simulator and the Voronoi algorithm output

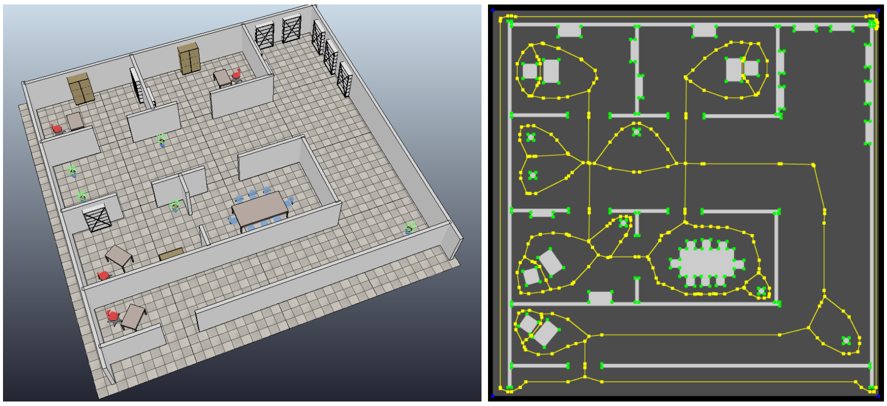


Figure 3.2.: Office environment in the V-Rep simulator and the Voronoi algorithm output

Each of the obstacles are additionaly sampled and the distance between obstacles (i.e. the free space for the swarm) is determined for each of the graph edges. This information is then used in the graph edges evaluation process.

3.2. Graph edges evaluation

For the graph edges evaluation, two approaches were used in this work. The first approach uses a heuristic function obtained experimentally in V-Rep simulator. The second approach uses the same heuristic for graph evaluation, but the graph evaluated using heuristic is then used only for acquiring a path in the graph that covers all edges. After that, simulation in V-Rep is started to experimentally determine the times it takes the swarm to fly over each of the graph edges. These times are then used as the resulting graph evaluation.

3.2.1. Evaluation using heuristic

The goal of this section is to acquire a heuristic function for the graph edges evaluation based on the number of swarm members and space between obstacles related to the corresponding edge. To find such a function, a set of experiments were made in the V-Rep simulator using the scene shown on figure 3.3 with two obstacles between which the swarm of various sizes is supposed to fly through.

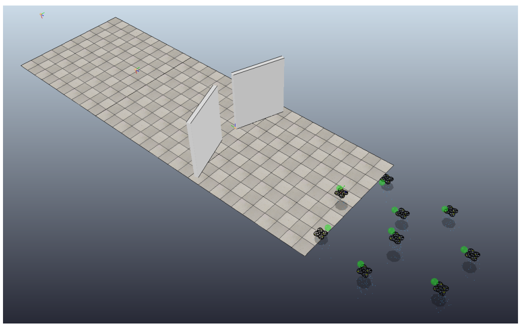


Figure 3.3.: V-Rep environment for determining the heuristic function for edges evaluation

First, an experiment was made to determine the time it takes the swarm to reach the end of the path without any obstacle in the environment. This time was measured for each swarm size and it is used as a reference to determine the time increment caused by the obstacles. The average reference times (the time was measured five times for each swarm size) can be seen in table 3.1. The average time increment measured for various distances between the obstacles for different swarm sizes can be seen in table 3.2. Based on the experimental observations, we propose a heuristic function in the form

$$\varepsilon(n, s, k_p, l_{ij}) = \varepsilon_r(k_p, l_{ij}) + \varepsilon_i(n, s),$$

where ε_r is the time it takes the swarm to fly over an edge without obstacles, ε_i is the time increment caused by the obstacles, n is the number of swarm members, s is the distance between obstacles, k_p is the path following constant introduced in section 2.4, and l_{ij} is the length of edge between the i-th and the j-th vertex.

Number of swarm members n	Reference times t_r $[s]$
1	11.74 ± 0.17
2	11.76 ± 0.07
3	11.71 ± 0.06
4	11.73 ± 0.10
5	11.71 ± 0.09

Table 3.1.: Reference time for each swarm size

For the first part of the heuristic function $\varepsilon_r(\cdot)$, the reference times shown in table 3.1 were used. As the differences between the measured time values for each swarm size are smaller than their individual variances, we can say that the times do not depend on the swarm size according to the obtained statistics. The reference times were measured as the difference between times when the last member of the swarm passed the start and the end of the measured path section. Since the whole swarm flies at the same speed as a single quadrotor, it will always take the last quadrotor the same time. The function $\varepsilon_r(\cdot)$ then depends only on the length of the measured edge l_{ij} and the path following constant k_p. Since the path following force stays constant, the maximum speed at which the swarm follows the path depends linearly on the weight constant k_p (described in 2.4) and the function $\varepsilon_r(\cdot)$ is in the form

$$\varepsilon_r(k_p, l_{ij}) = K\frac{l_{ij}}{k_p}.$$

Since the reference times in table 3.1 were measured with path following constant $k_p = 0.1$ on a path of length $l = 10m$, all the times were adjusted accordingly. The constant K was then determined as an average of the adjusted reference times and the resulting function for $\varepsilon_r(\cdot)$ is then

$$\varepsilon_r(k_p, l_{ij}) = 0.117\frac{l_{ij}}{k_p}.$$

	Number of swarm members n				
Distance between obstacles s $[m]$	1	2	3	4	5
1.0	1.75	2.91	3.89	4.65	5.80
1.1	1.70	2.93	3.57	4.13	4.97
1.2	1.58	2.36	3.18	3.58	4.17
1.3	1.35	2.04	2.60	3.05	3.82
1.4	1.33	1.87	2.15	2.63	3.47
1.5	1.23	1.53	1.88	2.45	2.92
1.6	1.20	1.40	1.80	2.23	2.67
1.7	0.98	1.23	1.60	1.90	2.52
1.8	0.95	1.18	1.50	1.73	2.27
1.9	0.88	1.10	1.43	1.58	1.72
2.0	0.85	0.95	1.30	1.45	1.67

Table 3.2.: Average measured time increment in seconds for different swarm sizes and different environments

The values of time increment caused by the obstacles measured in the experiment were then used to determine an exponential dependence of the time increment on the distance between obstacles for each swarm size using the Least Squares Method (LSM) [5]. The graphs of LSM can be seen on figure 3.4. The resulting functions for each swarm size can be seen in table 3.3.

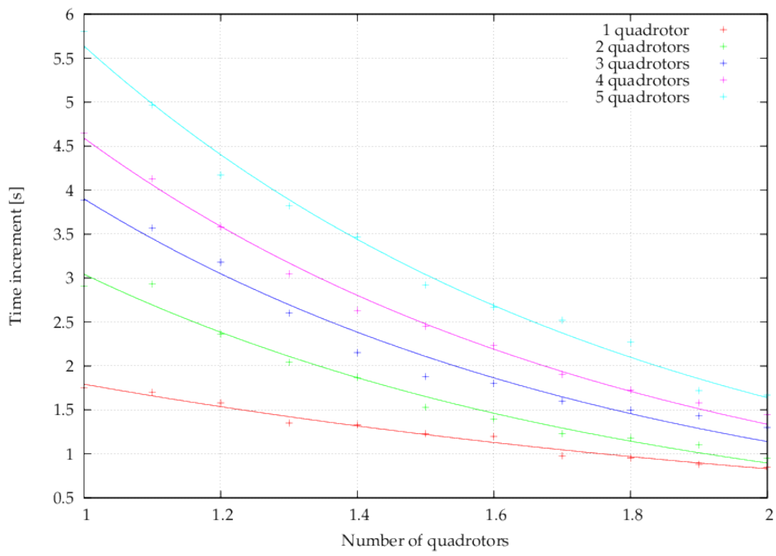


Figure 3.4.: Graph of LSM for each swarm size

Number of swarm members n	Function acquired by LSM
1	$\varepsilon_i(1,s) = 3.87e^{-0.77s}$
2	$\varepsilon_i(2,s) = 10.33e^{-1.222s}$
3	$\varepsilon_i(3,s) = 13.34e^{-1.230s}$
4	$\varepsilon_i(4,s) = 15.76e^{-1.233s}$
5	$\varepsilon_i(5,s) = 19.38e^{-1.235s}$

Table 3.3.: Exponential functions for each swarm size using LSM

As seen in table 3.3, all the functions are in the form

$$\varepsilon_i(n,s) = a_1(n)e^{-a_2(n)s}$$

with two parameters a_1 and a_2, which change with the swarm size. While a linear dependence of these parameters on the swarm size can be seen for the swarms (i.e. two or more quadrotors), the function for only one quadrotor deviates from this dependence. As one quadrotors is not affected by the Boids model, it tends to perform differently from the swarms. For that reason, the heuristic for only one quadrotor is computed separately.

To determine the dependence of parameters a_1 and a_2 on the swarm size and to derive the final function that is used for the evaluation, the LSM was used once again. The graphs of LSM for both parameters can be seen on figure 3.5 and the resulting functions are

$$a_1(n) = 4.353 + 2.957n,$$

$$a_2(n) = 1.215 + 0.004n.$$

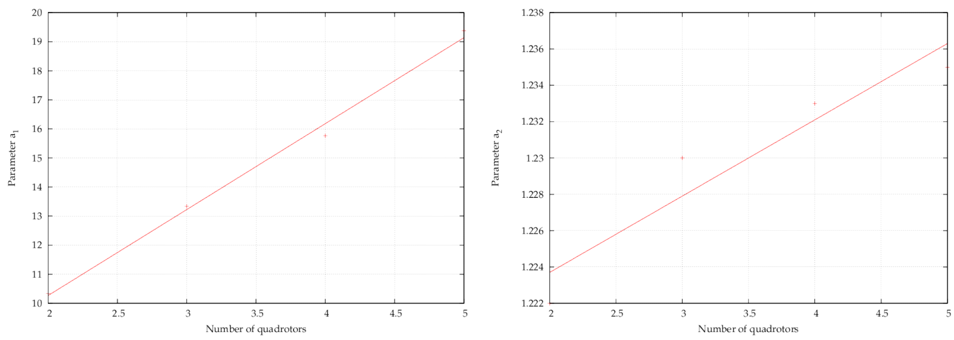


Figure 3.5.: Graphs of LSM for function parameters a_1 and a_2

The time increment due to obstacles can be estimated by heuristic

$$\varepsilon_i(n,s) = (4.353 + 2.957n)\, e^{-(1.215+0.004n)s}. \tag{3.1}$$

The comparison between measured values of time increment and the estimated values using (3.1) can be seen in table 3.4.

	Number of swarm members n			
Distance between obstacles $[m]$	2	3	4	5
1.0	2.91 / 3.02	3.89 / 3.88	4.65 / 4.72	5.80 / 5.57
1.1	2.93 / 2.67	3.57 / 3.43	4.13 / 4.18	4.97 / 4.92
1.2	2.36 / 2.37	3.18 / 3.03	3.58 / 3.69	4.17 / 4.35
1.3	2.04 / 2.09	2.60 / 2.68	3.05 / 3.27	3.82 / 3.84
1.4	1.87 / 1.85	2.15 / 2.37	2.63 / 2.89	3.47 / 3.40
1.5	1.53 / 1.64	1.88 / 2.10	2.45 / 2.55	2.92 / 3.00
1.6	1.40 / 1.45	1.80 / 1.86	2.23 / 2.26	2.67 / 2.65
1.7	1.23 / 1.28	1.60 / 1.64	1.90 / 2.00	2.52 / 2.34
1.8	1.18 / 1.14	1.50 / 1.45	1.73 / 1.76	2.27 / 2.07
1.9	1.10 / 1.01	1.43 / 1.29	1.58 / 1.56	1.72 / 1.83
2.0	0.95 / 0.89	1.30 / 1.14	1.45 / 1.38	1.67 / 1.62

Table 3.4.: Comparison between measured values of time increment / computed values using ε_i

The resulting heuristic functions for one quadrotor and for swarms are then

$$\varepsilon(1, s, k_p, l_{ij}) = 0.117\frac{l_{ij}}{k_p} + 3.87e^{-0.77s},$$

$$\varepsilon(n, s, k_p, l_{ij}) = 0.117\frac{l_{ij}}{k_p} + (4.353 + 2.957n)\, e^{-(1.215+0.004n)s};\ n = 2, 3, 4, \ldots.$$

3.2.2. Evaluation using simulation

The evaluation by simulation is an extension of the evaluation using heuristic. In this approach, the resulting heuristic shown in section 3.2.1 is used for an initial evaluation of the Voronoi graph. After the evaluation, a path that covers all the edges of the graph is found. The problem of finding such a path is called the Chinese Postman Problem (CPP) and there are multiple algorithms for finding an optimal solution in graphs of different properties [6, 7]. Since there is no need for the path to be optimal, as well as it does not have to start and end in the same node, a simple and fast algorithm for finding a path was introduced instead. The initial heuristic evaluation is used so that the algorithm prefers shorter unvisited edges. Due to this, the algorithm tends to first search the smaller domains of the graph before progressing further, which results in shorter paths. The algorithm written in pseudocode is:

```
path = [starting node]
while there are unvisited edges:
 if there are unvisited edges connected to the current node:
  next_node = closest node connected to the current node by unvisited edge
  path = path + next_node
 else:
  shortest_path = path from the current node to the closest unvisited edge
  path = path + shortest_path
return path
```

After the path is found, a simulation in the V-rep simulator is started. In the simulation, a swarm of quadrotors flies over the found path. Each edge is then evaluated by the time required for the whole swarm to reach the ending point of the edge. If one edge was used multiple times in the found path, the evaluation uses the smallest value from the times measured on this edge.

Chapter 4.

4 Swarm splitting and merging

For the possibility of swarm splitting and merging, a path finding algorithm developed at CTU was used. It takes a graph evaluated for 1 to N quadrotors as an input and finds a path for each of the quadrotors. An example of the algorithm output for the environment shown on figure 5.2 can be seen on figure 4.1.

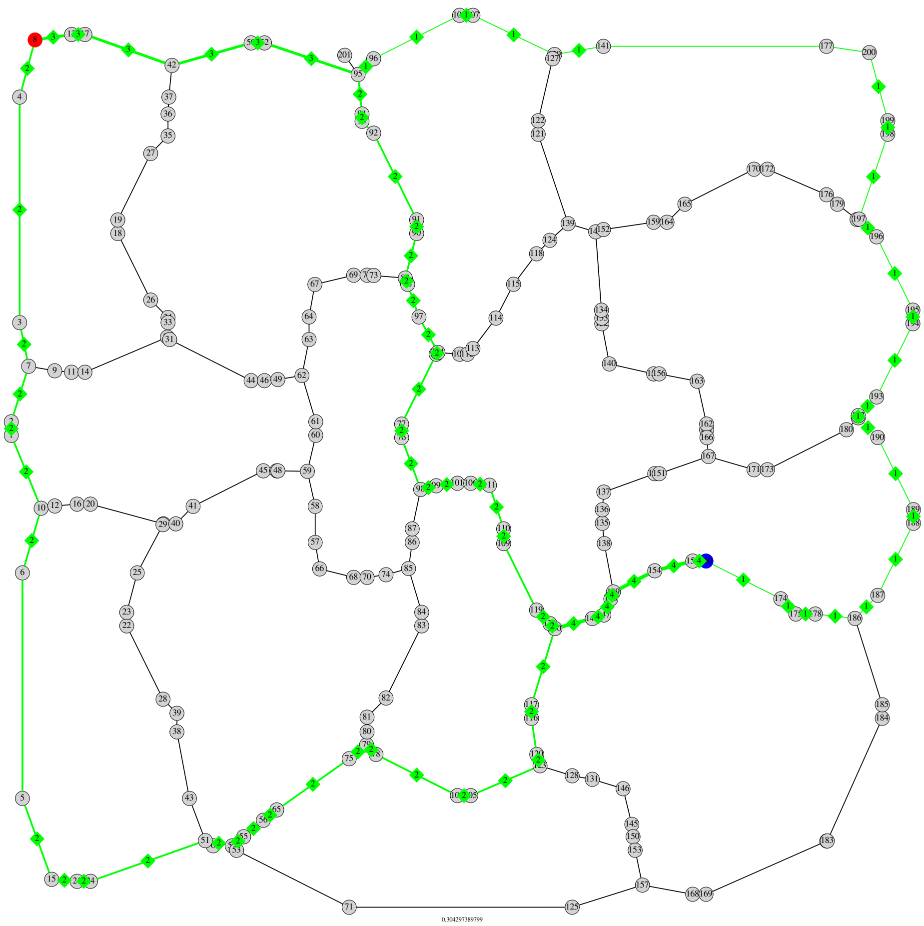


Figure 4.1.: Swarm splitting algorithm output example

On the figure, the grey circles are the graph nodes. The algorithm finds a path from the blue node to the red node. The edges of the path are denoted by the green marks. The numbers in the green squares are then the numbers of quadrotors that are supposed to fly on the particular edge.

For the purpose of swarm splitting, the Boids model discussed in section 2.1 needs to be adjusted. Additionally to the set of detected quadrotors R_i, a set of swarm members in a platoon (i.e. the subswarm that is supposed to fly together after swarm splitting) P_i was introduced. This set for the i-th quadrotor can be written as

$$P_i = \{b_j \in R_i; \forall b_j : w_i \in W_j \wedge w_j \in W_i\},$$

where W_i is a set of waypoints for the i-th quadrotor.

When a detected neighbor is not in a platoon with the i-th quadrotor, it is not used to compute the cohesion force $\vec{F_c}$ and its contribution to the separation force $\vec{F_s}$ is lowered. This ensures that the swarm spliting is slow and smooth.

Chapter 5.
5 Experiments

To test both the extended Boids model and the graph evaluation, the experiments were made in three different environments in the V-Rep simulator. The first environment (seen on figure 3.1) is very simple, with only a small number of obstacles in the form of cylinders. The second environment (seen on figure 5.1) is very dense, with a high number of obstacles. The third environment (seen on figure 5.2) is a maze-like environment well suited for the swarm splitting experiments.

The parameters of the Boids model used in the experiments can be seen in table 5.1.

Sensoric range r $[m]$	k_s	k_c	k_a	k_p	k_o
3	0.3	0.3	0.0	0.2	0.2

Table 5.1.: Parameters for the Boids model used in the experiments

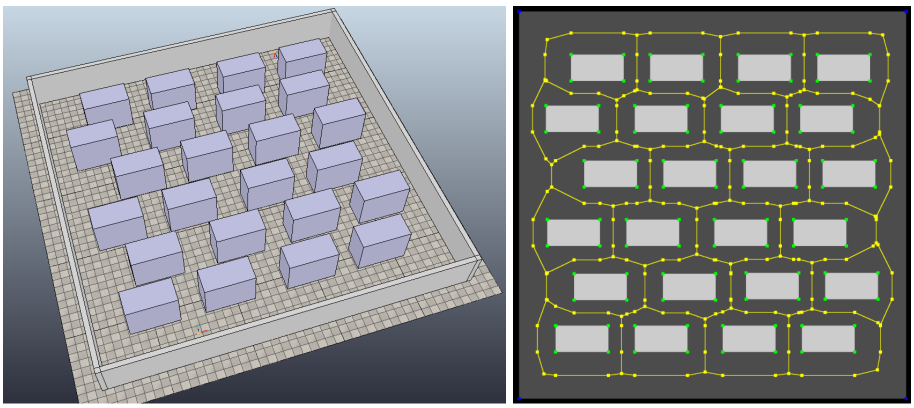


Figure 5.1.: Dense environment in the V-Rep simulator and the Voronoi algorithm output

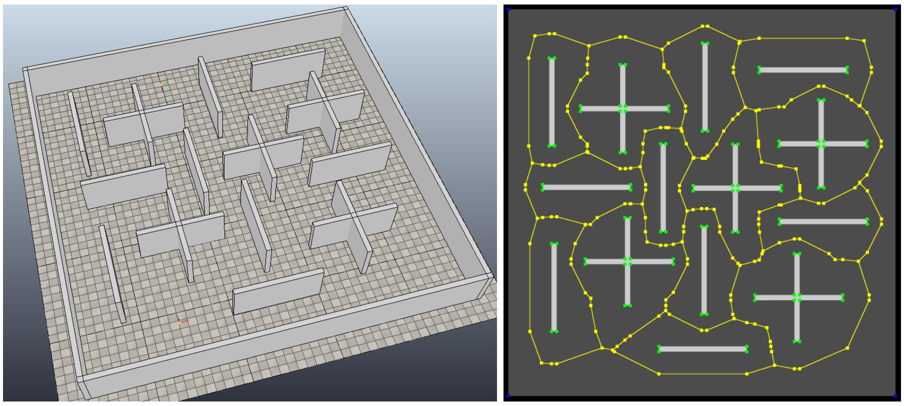


Figure 5.2.: Maze environment in the V-Rep simulator and the Voronoi algorithm output

The experiments were done for swarms with one to five members. To see the performance of the extended Boids model discussed in chapter 2, some experiments are accompanied by graphs showing the time development of velocities of each quadrotor, the distance to the closest detected obstacle and their closest neighbor. These graphs are shown in the appendices. The starting and ending points for the path finding algorithm are shown on the related figures in each section as yellow spheres. The found path is shown on the figures as a blue line with blue spheres representing the graph nodes.

5.1. Simple environment

In this section, the results of the experiments in the simple environment seen on figure 3.1 are shown. The shortest found paths in the simple environment for each swarm size can be seen on figure 5.3 for the heuristic evaluation and on figure 5.4 for the evaluation by simulation. For most of the cases, the two evaluation approaches found the same path. For one quadrotor, the paths were very similar, but the small difference resulted in a significant time increase in the evaluation by simulation.

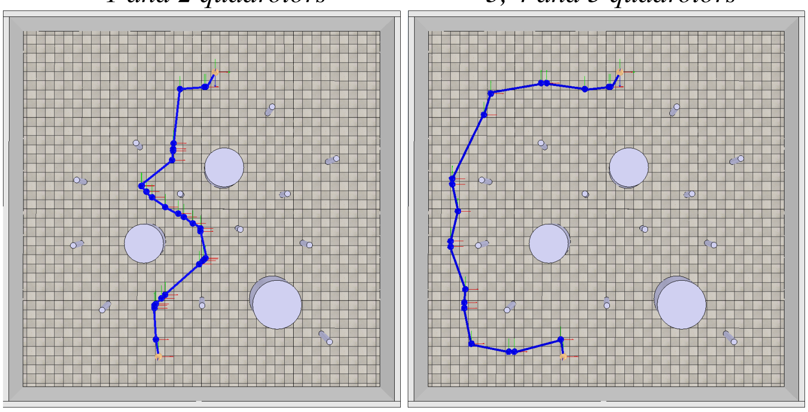


Figure 5.3.: Found paths in the simple environment using the heuristic evaluation

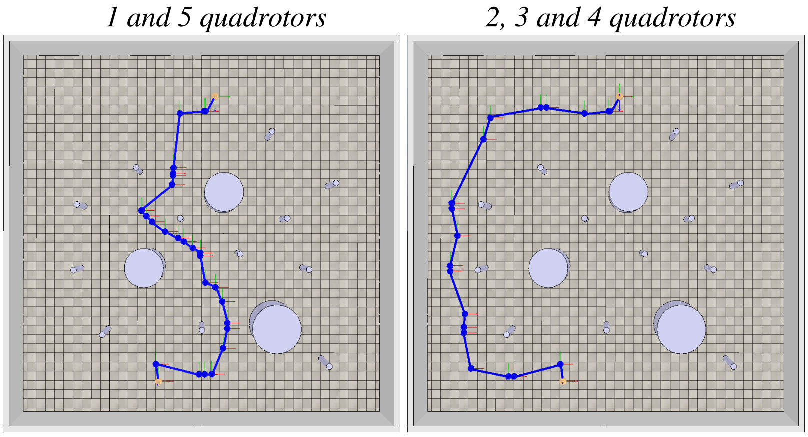


Figure 5.4.: Found paths in the simple environment using the evaluation by simulation

The times required for the swarm to reach the end of the paths found using the evaluation by heuristic and the paths found using the evaluation by simulation can be seen in table 5.2.

	Heuristic evaluation			Evaluation by simulation		
Swarm size n	Estimated times $[s]$	Simulation times $[s]$	Error	Estimated times $[s]$	Simulation times $[s]$	Error
1	13.943	22.050	36.8%	42.545	39.899	6.6%
2	13.602	29.750	54.3%	48.143	40.065	20.2%
3	17.097	37.149	54.0%	44.992	40.400	11.4%
4	17.395	38.099	54.3%	48.844	41.250	18.4%
5	17.684	38.699	54.3%	34.246	38.469	11.0%

Table 5.2.: Estimated and resulting times in the simple environment

As seen in the table, the heuristic fails to estimate the times it takes the swarm to reach the goal. The estimation error stays constant for all the swarm sizes except for one quadrotor, which has a separate heuristic function. This error is probably caused by the fact that in an environment with a low number of obstacles, the quadrotors have more space and they tend to wander off the given path. This is especially problematic when the path contains sharp turns, in which the swarm fails to react quickly enough and it takes the swarm a long time to return on the right path. This does not happen in the dense environment, since the high number of obstacles forces the swarm to fly exactly on the given path.

Another problem is that the heuristic function was based on experiments where the swarm was following a straight line. It means that the swarm flies at its maximum velocity when there are either no obstacles present or when there is enough space beween the obstacles. But in the simulation, even in environment without obstacles, the swarm is slowed down by the sharp turns of the path.

The times estimated by the evaluation using simulation are much more accurate than in the heuristic evaluation, but the estimation errors are still significant. Generally, the behavior of the swarm in an environment with a low number of obstacles tends to be very random and it is hard to predict the time it will take the swarm to reach the desired position.

Even though the time estimated by simulation was always more accurate than the estimations using heuristic, the heuristic always found either the same or a faster path. This is important, since the goal of this work is to find an optimal path. Additionaly, the evaluation by simulation takes much more time to compute than the heuristic evaluation. While the heuristic evaluation takes around 1 millisecond to compute in the simple environment, the evaluation by simulation takes up to 15 minutes depending on the swarm size.

5.2. Dense environment

In this section, the results of the experiments in the dense environment seen on figure 5.1 are shown. The heuristic evaluation resulted in the same path for all swarm sizes. This path can be seen on figure 5.5. The shortest paths found while using the evaluation by simulation can be seen on figure 5.6.

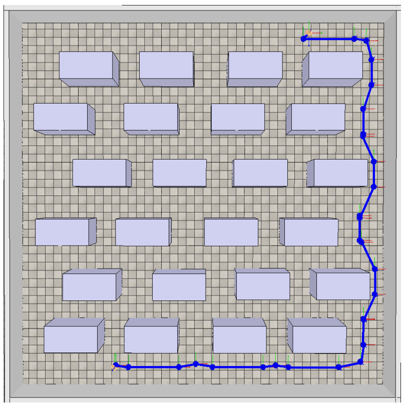


Figure 5.5.: Found path in the dense environment using the heuristic evaluation for all swarm sizes

The times required for the swarm to reach the end of the path found using the evaluation by heuristic and the paths found using the evaluation by simulation can be seen in table 5.3.

	Heuristic evaluation			Evaluation by simulation		
Swarm size n	Estimated times $[s]$	Simulation times $[s]$	Error	Estimated times $[s]$	Simulation times $[s]$	Error
1	37.498	38.249	2.0%	60.886	53.560	13.7%
2	36.296	41.900	13.4%	57.343	49.150	16.7%
3	40.047	42.100	4.9%	65.793	50.050	31.5%
4	43.733	43.900	0.4%	68.991	43.900	57.2%
5	47.350	47.751	0.8%	78.291	47.751	64.0%

Table 5.3.: Estimated and resulting times in the dense environment

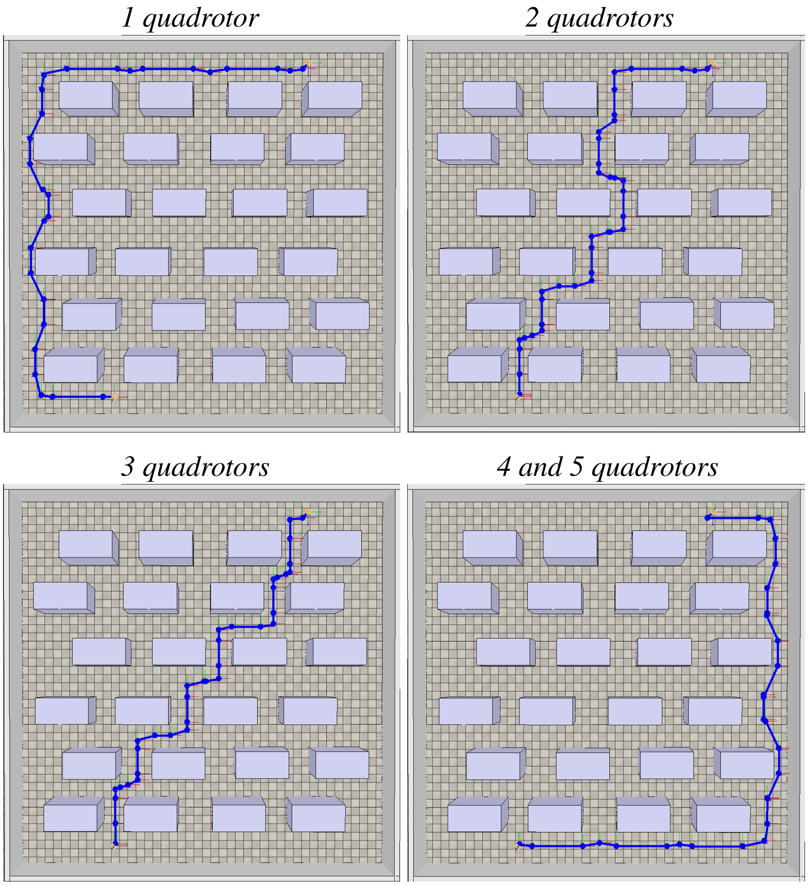


Figure 5.6.: Found paths in the dense environment using the evaluation by simulation

The times estimated by the heuristic in the dense environment were very accurate, especially for the swarms of four and five quadrotors. This is due to the fact that in dense environment, the swarm does not have much space to maneuver, which makes the movements of the swarm more deterministic than in the simple environment. There are also almost no edges where the heuristic evaluation expects the swarm to fly at a maximum speed. Thanks to this, the errors caused by the path curvature are much less dominant than in the simple environment.

In contrast to the very accurate heuristic evaluation, the evaluation using simulation has much worse results. That is because the simulation is very dependant on the chosen path itself. For example, it can take the swarm only a small amount of time to fly over one edge in case the swarm approaches the edge from one direction, but a lot of time if it approaches the edge from a different direction. In order to determine the times correctly, the evaluation using simulation would have to cover not just all edges, but all possible ways to approach each edge, which would significantly increase the time of the evaluation. The high number of obstacles and cramped spaces in the dense environment make the differences between possible approaches to each edge from multiple directions more significant. Since a small swarm is not influenced so much by a small amount of space to manuever, the estimation error while using the evaluation by simulation tends to be the higher for bigger swarms.

5.3. Maze environment

In this section, the results of the experiments in the maze environment seen on figure 5.2 are shown. The heuristic evaluation resulted in the same path for all swarm sizes. This path can be seen on figure 5.7. In order to achieve more variety in the found paths, the experiments using the evaluation by simulation have different ending point than the experiments using heuristic evaluation. For that reason, the measured times for the two evaluation approaches cannot be compared in the maze environment. The shortest paths found while using the evaluation by simulation can be seen on figure 5.8.

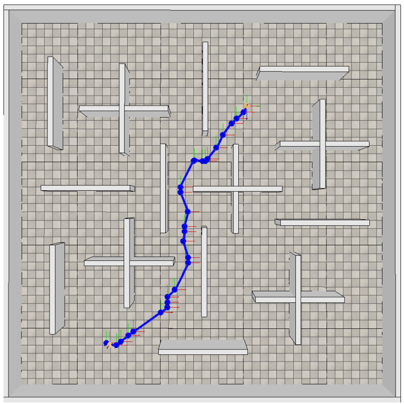


Figure 5.7.: Found path in the maze environment using the heuristic evaluation for all swarm sizes

The times required for the swarm to reach the end of the path found using the evaluation by heuristic and the paths found using the evaluation by simulation can be seen in table5.4.

	Heuristic evaluation			Evaluation by simulation		
Swarm size n	Estimated times $[s]$	Simulation times $[s]$	Error	Estimated times $[s]$	Simulation times $[s]$	Error
1	23.336	23.399	0.3%	53.347	51.800	3.0%
2	24.760	23.100	7.2%	48.748	32.449	50.2%
3	28.554	24.350	17.3%	46.398	33.600	38.1%
4	32.294	30.300	6.6%	53.737	41.999	27.9%
5	35.978	29.750	20.9%	52.237	44.199	18.2%

Table 5.4.: Estimated and resulting times in the maze environment

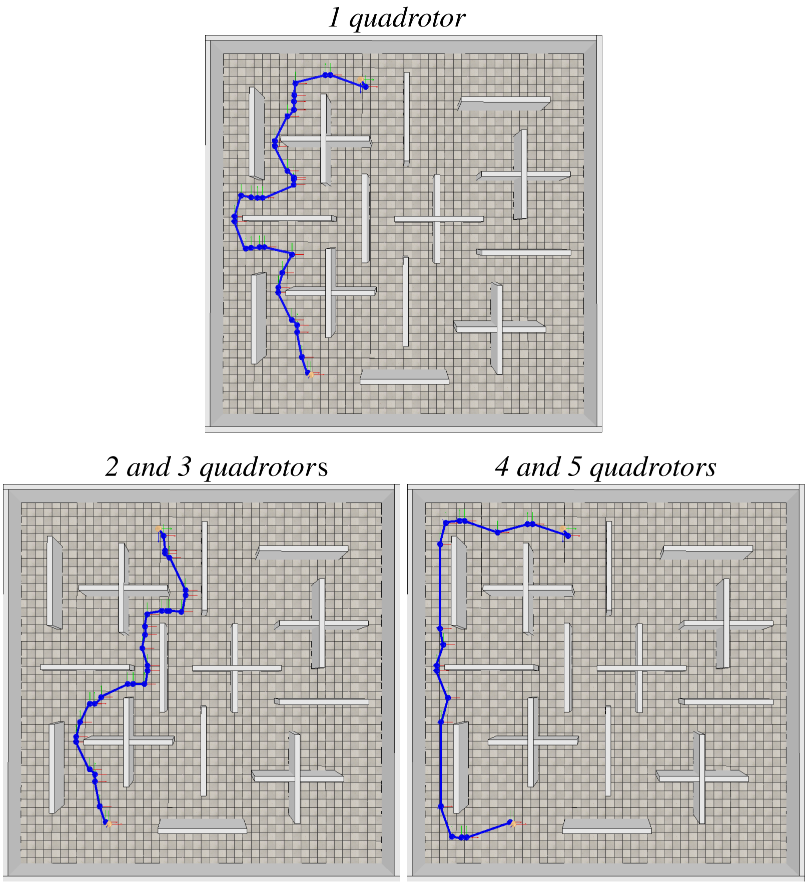


Figure 5.8.: Found paths in the maze environment using the evaluation by simulation

As seen in the table, the time estimations of the evaluation using heuristic in the maze environment tend to be worse than in the dense environment, but still better than in the simple environment. The evaluation using simulation performs worse than in the simple environments, but better than in the dense environment. For that reason, it is safe to assume that generally the evaluation using heuristic works better in environments with a high number of obstacles with less space, while the evaluation by simulation performs better in environments with a low density of obstacles. This applies only to the estimates of the time required for the swarm to reach the end of the path.

5.4. Experiments with the possibility of swarm splitting

In this section, the results of the experiments with a swarm capable of splitting and merging are discussed. The experiments were made with a swarm of five members with a Boids model adjusted for the purpose of swarm splitting, discussed in chapter 4. The estimated times shown in the tables presented in this section are the times it takes the last of the quadrotors to reach the end of the path. Since the results of the path planning algorithm for swarm splitting occupy a lot of space in order to be legible, they were placed in appendix E.

5.4.1. Simple environment

In the simple environment, two starting and ending positions for both evaluation approaches were used. Since the time increments caused by the obstacles are very small in this environment, the swarm tends to stay together instead of splitting. The time development of one of the simulations can be seen on figure 5.9. The times required for the swarm to reach the end of the path found using the evaluation by heuristic and the paths found using the evaluation by simulation can be seen in table5.5.

Heuristic evaluation			Evaluation by simulation		
Estimated times $[s]$	Simulation times $[s]$	Error	Estimated times $[s]$	Simulation times $[s]$	Error
Path from node 15 to node 133 (figure E.1)					
18.860	37.500	49.7%	34.947	36.850	5.2%
Path from node 29 to node 110 (figure E.2)					
17.562	37.350	53.0%	32.251	37.600	14.2%

Table 5.5.: Experiments in the simple environment with the possibility of swarm splitting

As seen in the table, the precision of the estimates follows the same trend as in the experiments in the simple environment without the possibility of swarm splitting. The heuristic function is unable to correctly estimate the time it will take the swarm to reach the end of the path, while the evaluation using simulation has much better results. Without the possibility of splitting, the heuristic evaluation always found a path that was faster in the end than the evaluation using simulation though. In this case, the resulting finishing times are very similar for both evaluation approaches and the time estimations made by the evaluation using simulation were much more accurate.

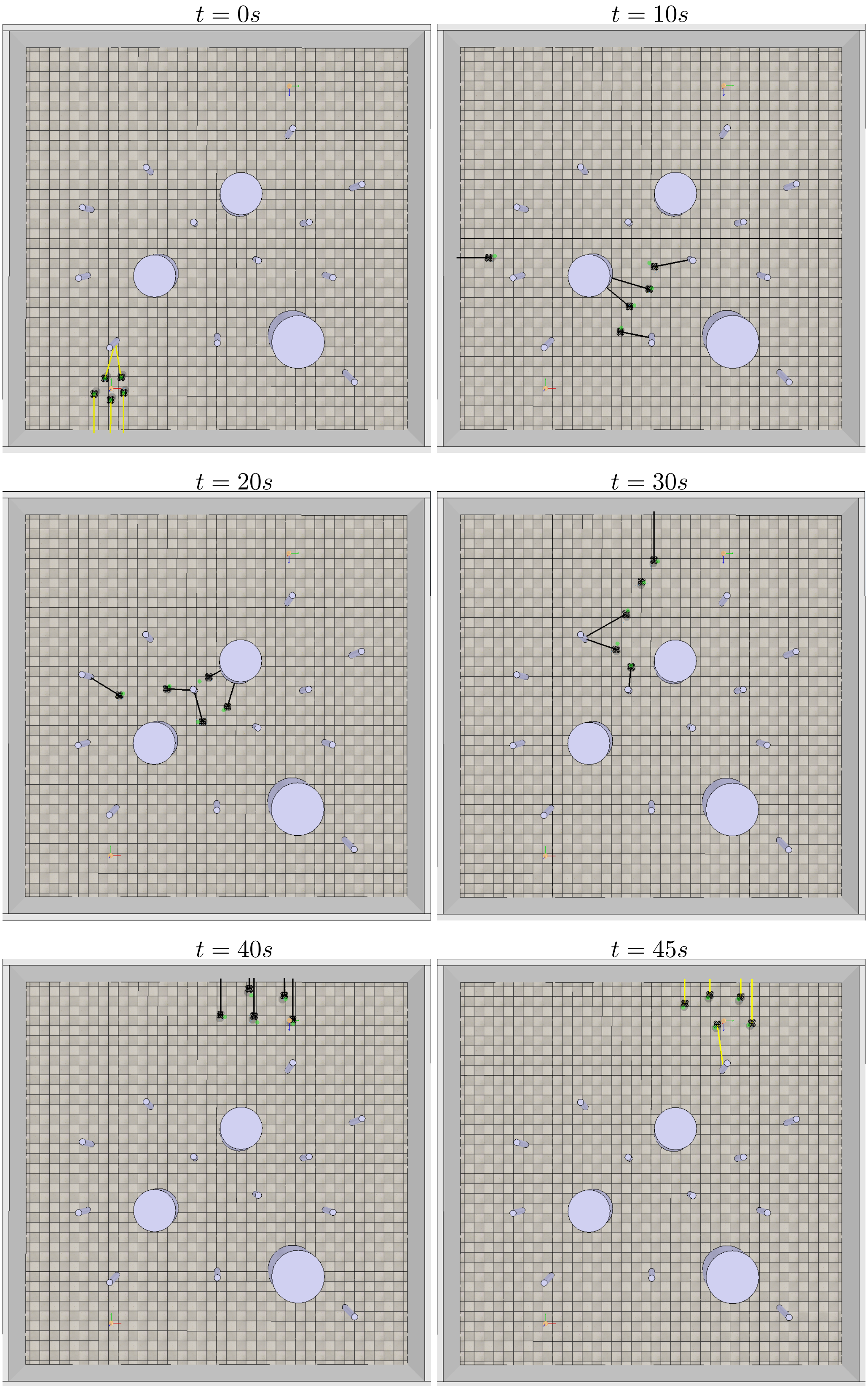


Figure 5.9.: V-Rep experiment screenshots using evaluation by simulation for path from node 29 to node 110 in the simple environment

5.4.2. Dense environment

In the dense environment, three starting and ending positions for both evaluation approaches were used. The time development of one of the simulations can be seen on figure 5.10. The times required for the swarm to reach the end of the path found using the evaluation by heuristic and the paths found using the evaluation by simulation can be seen in table 5.6.

Heuristic evaluation			Evaluation by simulation		
Estimated times $[s]$	Simulation times $[s]$	Error	Estimated times $[s]$	Simulation times $[s]$	Error
Path from node 42 to node 171 (figure E.3)					
50.486	51.099	1.2%	52.098	53.999	3.5%
Path from node 91 to node 171 (figure E.4)					
47.396	52.899	10.4%	51.742	51.149	1.2%
Path from node 152 to node 81 (figure E.5)					
48.022	52.449	8.4%	43.243	45.750	5.5%

Table 5.6.: Experiments in the dense environment with the possibility of swarm splitting

In the dense environment, the evaluation using heuristic tends to be very accurate as in the case without swarm splitting. The evaluation using simulation is much more accurate while using swarm splitting though. This is probably caused by the fact that in this environment, the swarm tends to split a lot. As seen on the figure E.3, the quadrotors usually formed platoons of only two or three quadrotors, or they were even flying alone. The estimated times by the evaluation using simulation are then much more accurate, since the evaluation works better for these small swarms. Thanks to the smaller swarms, the evaluation by simulation generally tends to be more accurate in all experiments with the possibility of swarm splitting.

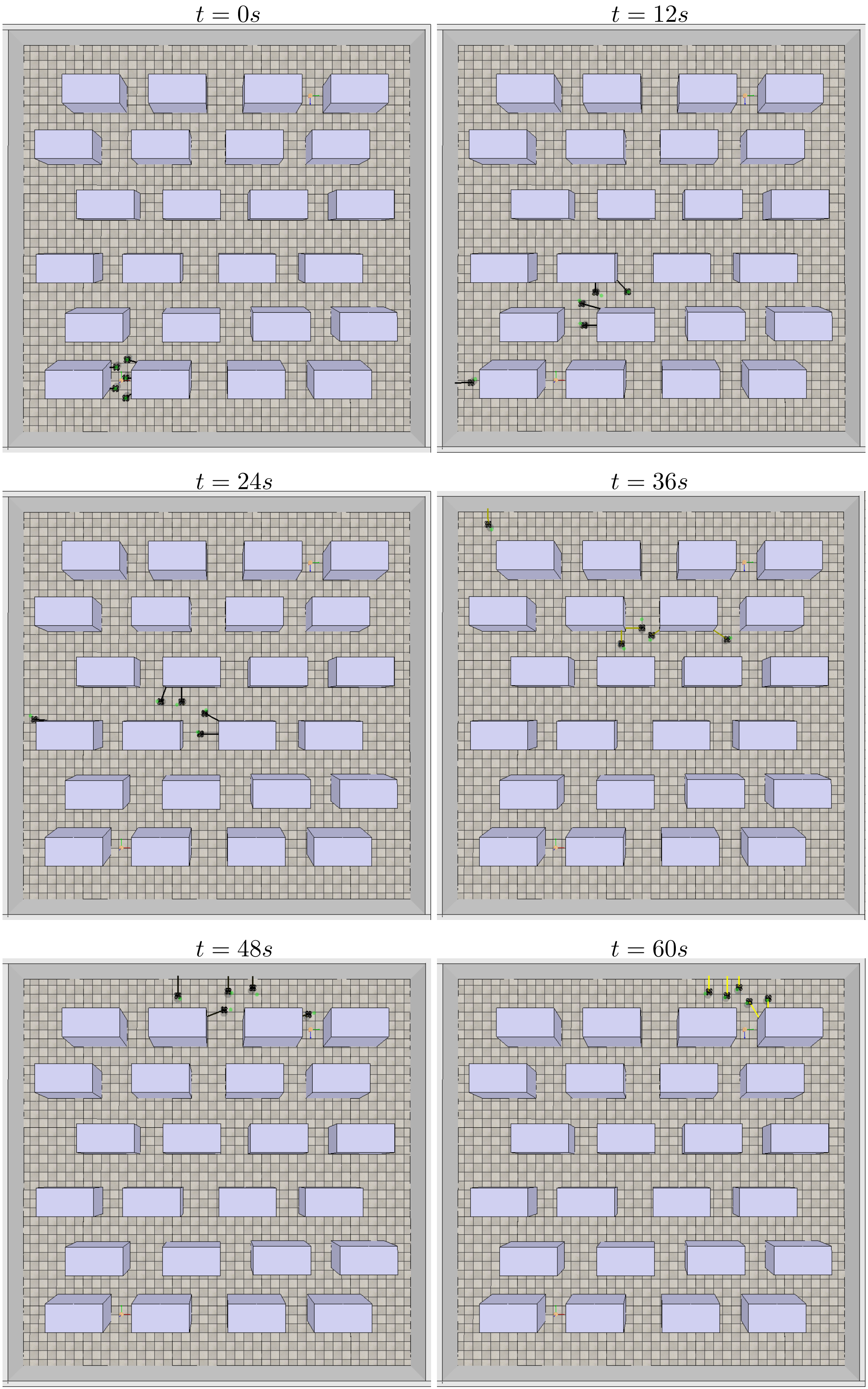


Figure 5.10.: V-Rep experiment screenshots using evaluation by simulation for path from node 42 to node 171 in the dense environment

5.4.3. Maze environment

In the maze environment, two starting and ending positions for both evaluation approaches were used. The time development of one of the simulations can be seen on figure 5.11. The times required for the swarm to reach the end of the path found using the evaluation by heuristic and the paths found using the evaluation by simulation can be seen in table 5.7.

Heuristic evaluation			Evaluation by simulation		
Estimated times $[s]$	Simulation times $[s]$	Error	Estimated times $[s]$	Simulation times $[s]$	Error
Path from node 15 to node 200 (figure E.6)					
34.970	42.800	18.3%	59.136	46.700	26.6%
Path from node 160 to node 8 (figure E.7)					
37.877	47.500	20.3%	54.844	59.599	8.0%

Table 5.7.: Experiments in the maze environment with the possibility of swarm splitting

As in the experiments without the possibility of swarm splitting, the time estimates made by the evaluation using simulation were more accurate than in the simple environment, but still less accurate than the estimates in the dense environment. The same applies for the evaluation by simulation for the path from node 160 to node 8, but the estimate for the path from node 15 to node 200 was very inaccurate compared to other experiments with the possibility of swarm splitting.

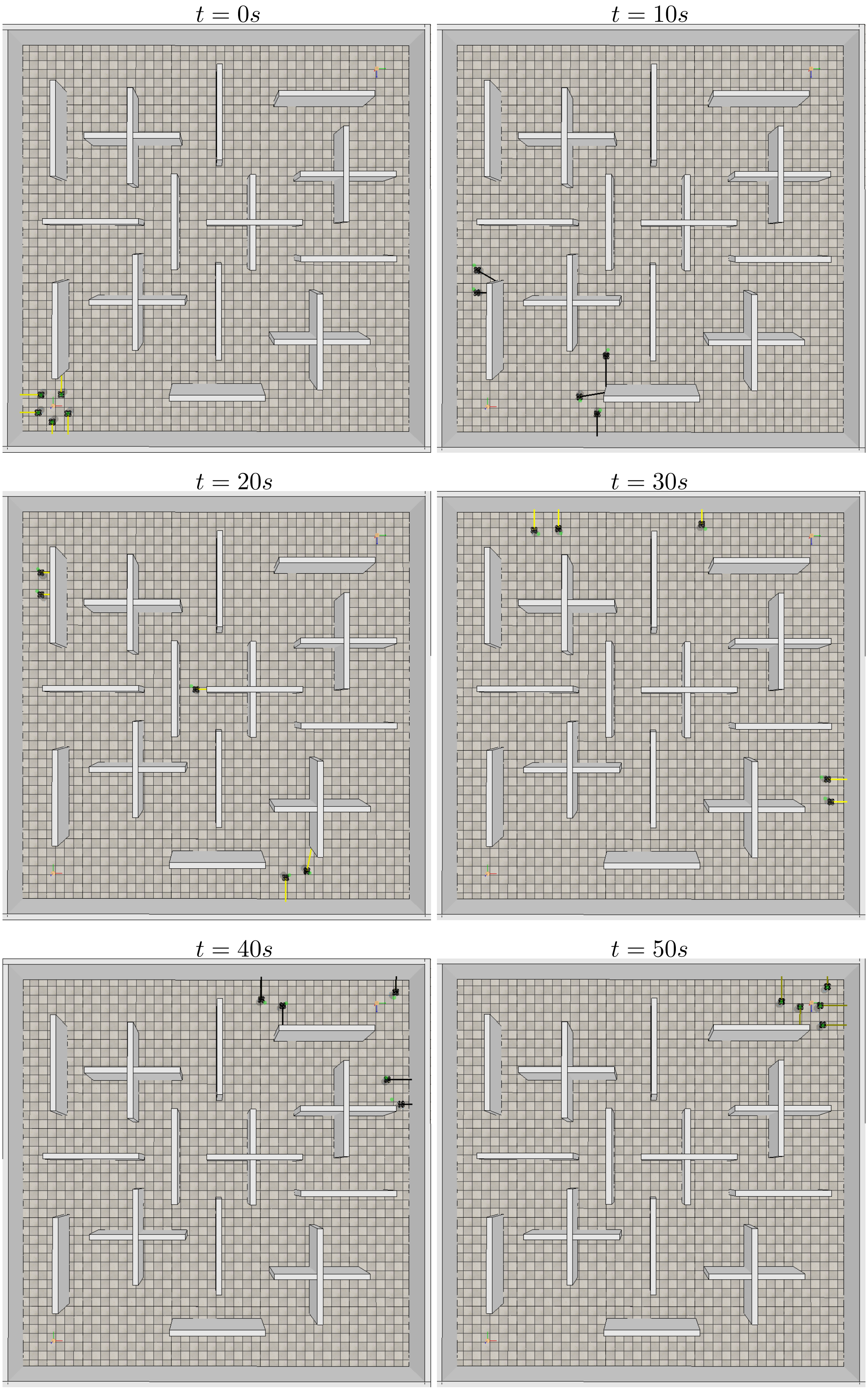


Figure 5.11.: V-Rep experiment screenshots using heuristic evaluation for path from node 15 to node 200 in the maze environment

6 Chapter 6. Conclusion

All the thesis assignment tasks have been met. Even though the real world experiment was not accomplished in the end, the implemented system was verified by a large set of experiments in different environments to fully determine its capabilities.

The functionality of the extended Boids model can be well seen on the graphs provided in the appendices. The Boids model works very well together with the simple obstacle avoidance force, while following a given path. The path following force has a problem with sharp turns of the path. This is caused by the simple proportional position regulator used in the V-Rep simulator for the quadrotors, which tends to react very slowly to extensive direction changes. For this reason, the performance of the extended Boids model can be significantly increased by using a more complex position regulator in a future work, such as the onboard model predictive control method [16], which is used in the multi-MAV platform at CTU in Prague [17].

Two approaches to the Voronoi graph evaluation were implemented. The first approach uses a heuristic function based on the number of quadrotors in a swarm and the distance between obstacles. The other approach uses the simulation to evaluate the edges based on the time it takes the quadrotors to fly over each edge.

The heuristic evaluation generally works very well in environments with a high density of obstacles, where it can usually estimate the time it will take the swarm to reach the desired position with the accuracy to 5%. The accuracy of this evaluation decreases significantly with a small number of obstacles. Even without an accurate time estimation, this approach usually finds a better or the same path as the evaluation using simulation. Additionally, the heuristic evaluation takes much less time to compute. While the graph evaluation using heuristic takes around 1 millisecond to compute in the environments used in the experiments, the evaluation by simulation usually takes over 10 minutes, depending on the swarm size. The comparison between the two evaluation approaches in terms of finding a better path in the experiments can be seen in table 6.1.

	Without swarm splitting	With swarm splitting
Both approaches found the same path	4 times	0 times
Heuristic found a better path	5 times	4 times
Simulation found a better path	1 time	3 times

Table 6.1.: Comparison between the results of both evaluation approaches

As seen in the table, the evaluation by simulation fails to find an optimal path in most of the cases without the possibility of swarm splitting. Because it performs better with smaller swarms, the accuracy of the time estimation by this approach grows significantly with the possibility of swarm splitting and merging, since it estimates times for smaller subswarms. The heuristic evaluation still outperforms the evaluation by simulation even with the possibility of swarm splitting, though.

The evaluation by simulation is insufficient at the current state. In order to make this evaluation approach more accurate, the simulations would have to test the swarm behavior when approaching the graph edges from different directions. This would require each edge to be tested separately, which the V-Rep simulator does not allow. In a future work, we intend to use the Gazebo simulator [18], which will allow to test the edges separately and to furherly develop the evaluation by simulation. This evaluation can also be made faster and much more accurate by using a more complex quadrotor position regulator to make the Boids model more deterministic, as mentioned above.

References

[1] Wikipedia. *K-Shortest path routing algorithm*. https://en.wikipedia.org/wiki/K_shortest_path_routing

[2] Craig Reynolds website. *Boids swarming model*. http://www.red3d.com/cwr/boids/

[3] Wikipedia. *Voronoi diagram*. https://en.wikipedia.org/wiki/Voronoi_diagram

[4] Boost library. *Voronoi diagram C++ implementation*. http://www.boost.org/doc/libs/1_54_0/libs/polygon/doc/voronoi_main.htm

[5] Wikipedia. *Least Squares method*. https://en.wikipedia.org/wiki/Least_squares

[6] E. Minieka. *The Chinese postman problem for mixed networks. Management Science, 25 (1979), pp. 643–648*, 1979.

[7] Yves Nobert, Jean-Claude Picard. A*n optimal algorithm for the mixed Chinese postman problem. Networks 27(2): 97-108*, 1996.

[8] Coppelia Robotics website. *V-Rep Simulator*. http://www.coppeliarobotics.com/

[9] J. Faigl, T. Krajnik, J. Chudoba, L. Preucil and M. Saska. *Low-Cost Embedded System for Relative Localization in Robotic Swarms. In Proceedings of 2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[10] T. Krajnik, M. Nitsche, J. Faigl, P. Vanek, M. Saska, L. Preucil, T. Duckett and M. Mejail. *A Practical Multirobot Localization System. Journal of Intelligent & Robotic Systems 76(3-4):539-562*, 2014.

[11] Multi-robot Systems group at CTU. *Swarm Robotics*. http://mrs.felk.cvut.cz/research/swarm-robotics

[12] X. Cui, C. T. Hardin, R. K. Ragade, and A. S . Elmaghraby. *A Swarm Approach for Emission Sources Localization. In ICTAI 2004: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, 2004.

[13] M. Saska, J. Langr and L. Preucil. *Plume Tracking by a Self-stabilized Group of Micro Aerial Vehicles. In Modelling and Simulation for Autonomous Systems,* 2014.

[14] J. Chudoba, M. Kulich, M. Saska, T. Báča and L. Přeučil. *Exploration and Mapping Technique Suited for Visual-features Based Localization of MAVs. Journal of Intelligent & Robotic Systems. First online., pages 1–19*, 2016.

[15] 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. First online*, 2016.

[16] T. Baca, M. Saska. *Embedded Model Predictive Control of Micro Aerial Vehicles, International Conference on Methods and Models in Automation and Robotics*, 2016.

[17] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno and V. Kumar. *System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization. Autonomous Robots. First online*, 2016.

[18] Robot Operating System (ROS) website. *The Gazebo simulator*. `http://wiki.ros.org/gazebo`

[19] M. Saska. *MAV-swarms: unmanned aerial vehicles stabilized along a given path using onboard relative localization. In Proceedings of 2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015.

[20] M. Saska, J. Vakula and L. Preucil. *Swarms of Micro Aerial Vehicles Stabilized Under a Visual Relative Localization. In Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA),* 2014.

[21] M. Saska, V. Vonasek, T. Krajnik and L. Preucil. *Coordination and Navigation of Heterogeneous MAV–UGV Formations Localized by a 'hawk-eye'-like Approach Under a Model Predictive Control Scheme. International Journal of Robotics Research 33(10):1393–1412*, 2014.

[22] M. Saska, T. Krajnik, V. Vonasek, Z. Kasl, V. Spurny and L. Preucil. *Fault-Tolerant Formation Driving Mechanism Designed for Heterogeneous MAVs-UGVs Groups. Journal of Intelligent and Robotic Systems 73(1-4):603–622*, 2014.

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

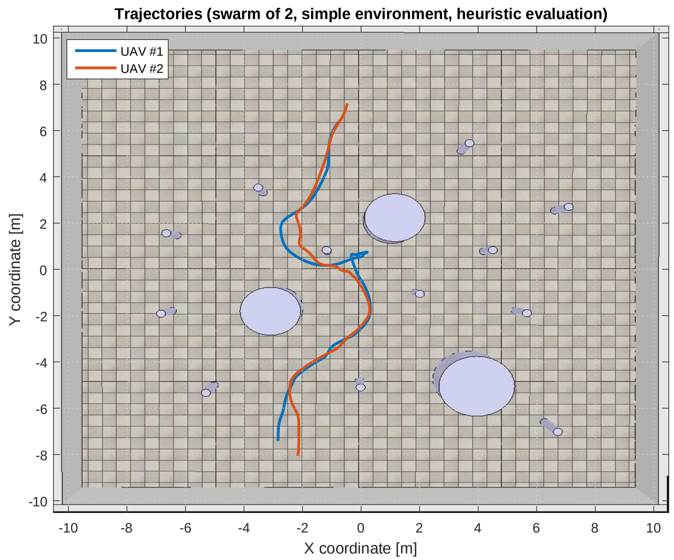
[24] M. Saska, J. Chudoba, L. Preucil, J. Thomas, G. Loianno, A. Tresnak, V. Vonasek and V. Kumar. *Autonomous Deployment of Swarms of Micro-Aerial Vehicles in Cooperative Surveillance. In Proceedings of 2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014.

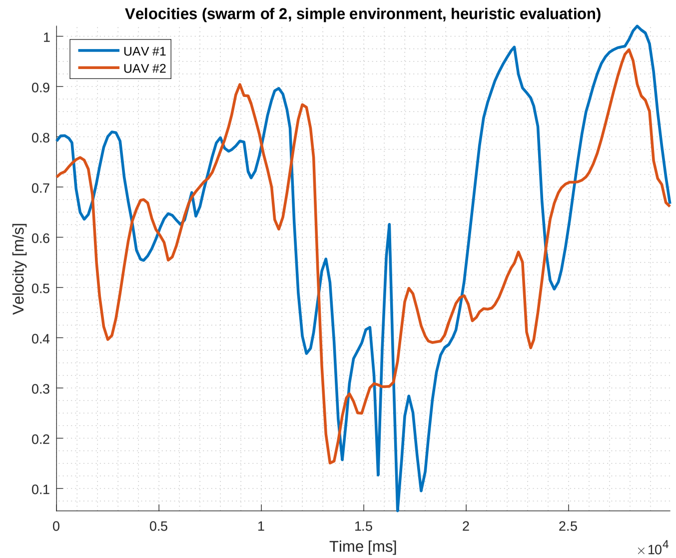
Appendix A.

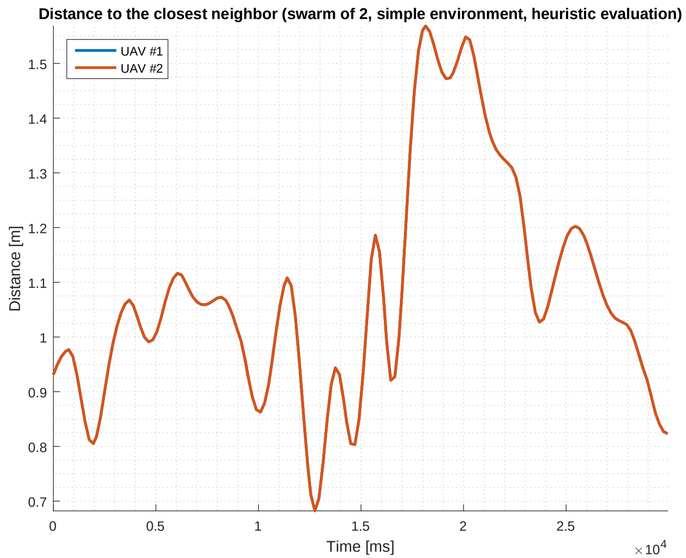
A Contents of the attached CD

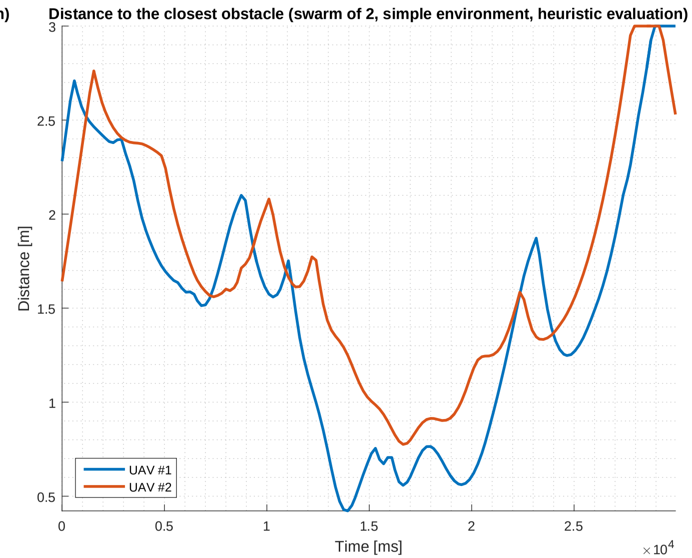
Folder or File	Description
captured_data/	Captured data from the experiments
sourcecodes/	The python project sourcecodes together with the C++ sourcecodes of the Voronoi diagram algorithm
thesis_project/	Lyx project of this thesis together with the used pictures
videos/	Videos of selected experiments (captured separately, the data from the experiments may differ from the videos)
vrep_scenes/	V-Rep scenes used for the experiments
Brich_BP_2016.pdf	Electronic version of this thesis

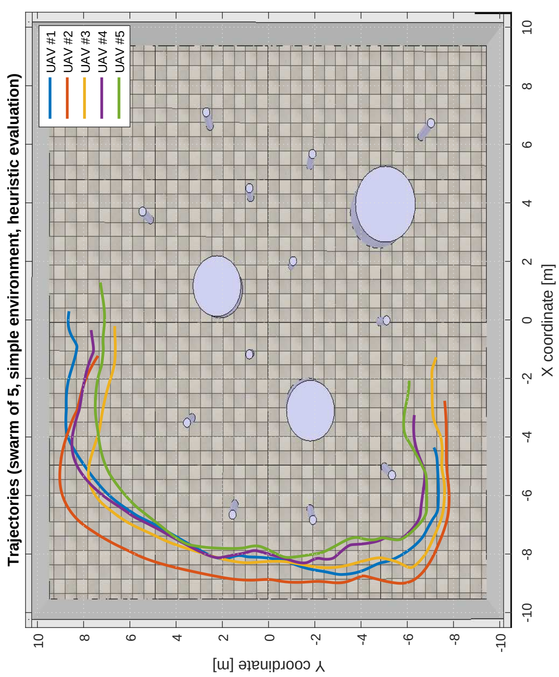
Appendix B. Graphs for the simple environment

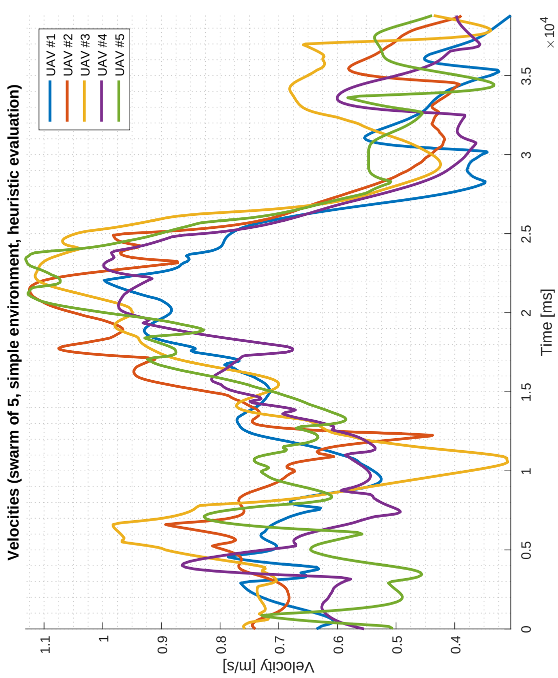

Trajectories (swarm of 2, simple environment, heuristic evaluation)
UAV #1
UAV #2
Y coordinate [m]
X coordinate [m]
10
8
6
4
2
0
-2
-4
-6
-8
-10

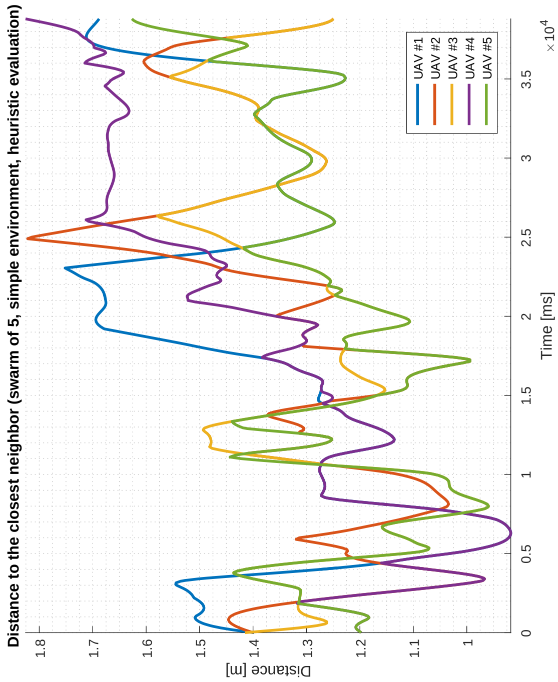

Velocities (swarm of 2, simple environment, heuristic evaluation)
UAV #1
UAV #2
Velocity [m/s]
Time [ms]
×10^4
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.5
1
1.5
2
2.5

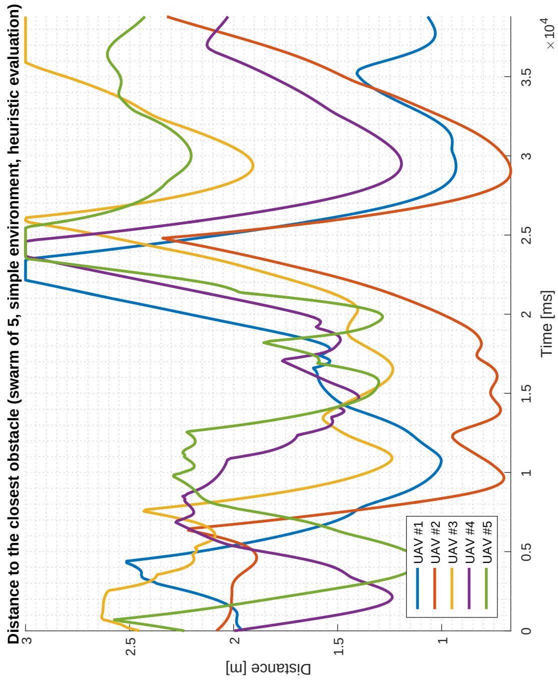

Distance to the closest neighbor (swarm of 2, simple environment, heuristic evaluation)
UAV #1
UAV #2
Distance [m]
Time [ms]
×10^4
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0
0.5
1
1.5
2
2.5

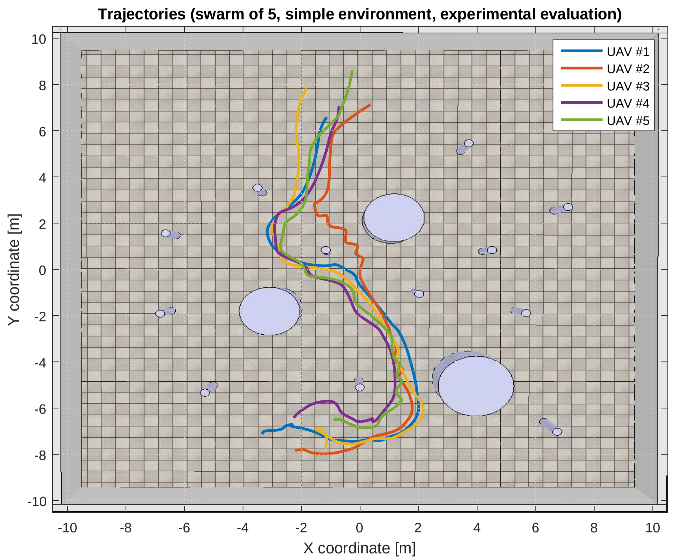

Distance to the closest obstacle (swarm of 2, simple environment, heuristic evaluation)
UAV #1
UAV #2
Distance [m]
Time [ms]
×10^4
3
2.5
2
1.5
1
0.5
0
0.5
1
1.5
2
2.5

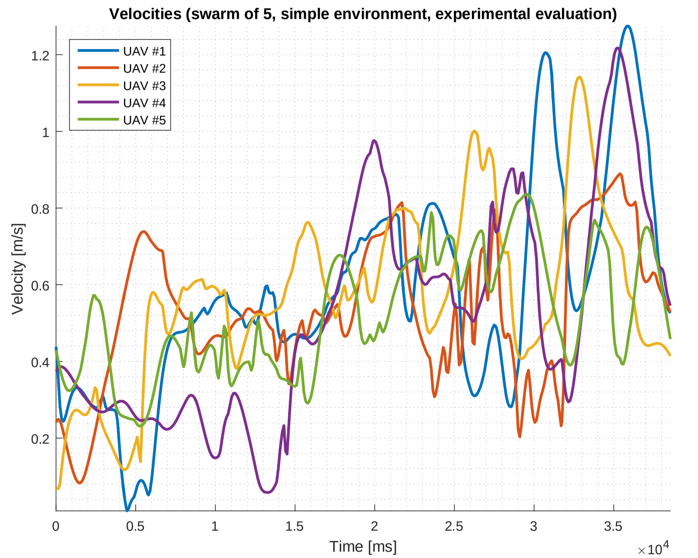

Trajectories (swarm of 5, simple environment, heuristic evaluation)
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
X coordinate [m]
Y coordinate [m]

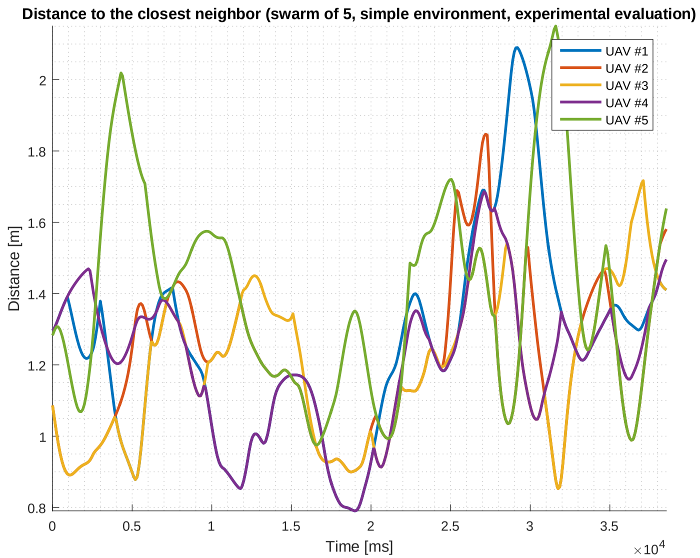

Velocities (swarm of 5, simple environment, heuristic evaluation)
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
Time [ms]
Velocity [m/s]
×10^4

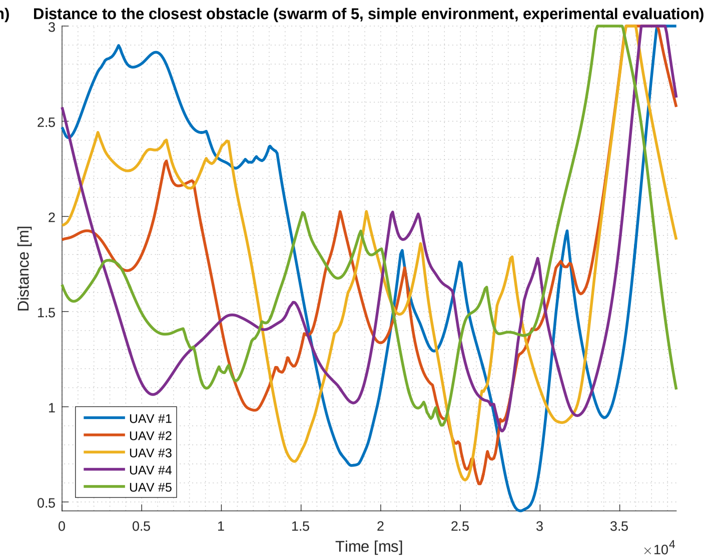

Distance to the closest neighbor (swarm of 5, simple environment, heuristic evaluation)
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
Time [ms]
Distance [m]
×10^4


Distance to the closest obstacle (swarm of 5, simple environment, heuristic evaluation)
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
Time [ms]
Distance [m]
×10^4


Trajectories (swarm of 5, simple environment, experimental evaluation)
X coordinate [m]
Y coordinate [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5


Velocities (swarm of 5, simple environment, experimental evaluation)
Time [ms]
Velocity [m/s]
$\times 10^4$
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

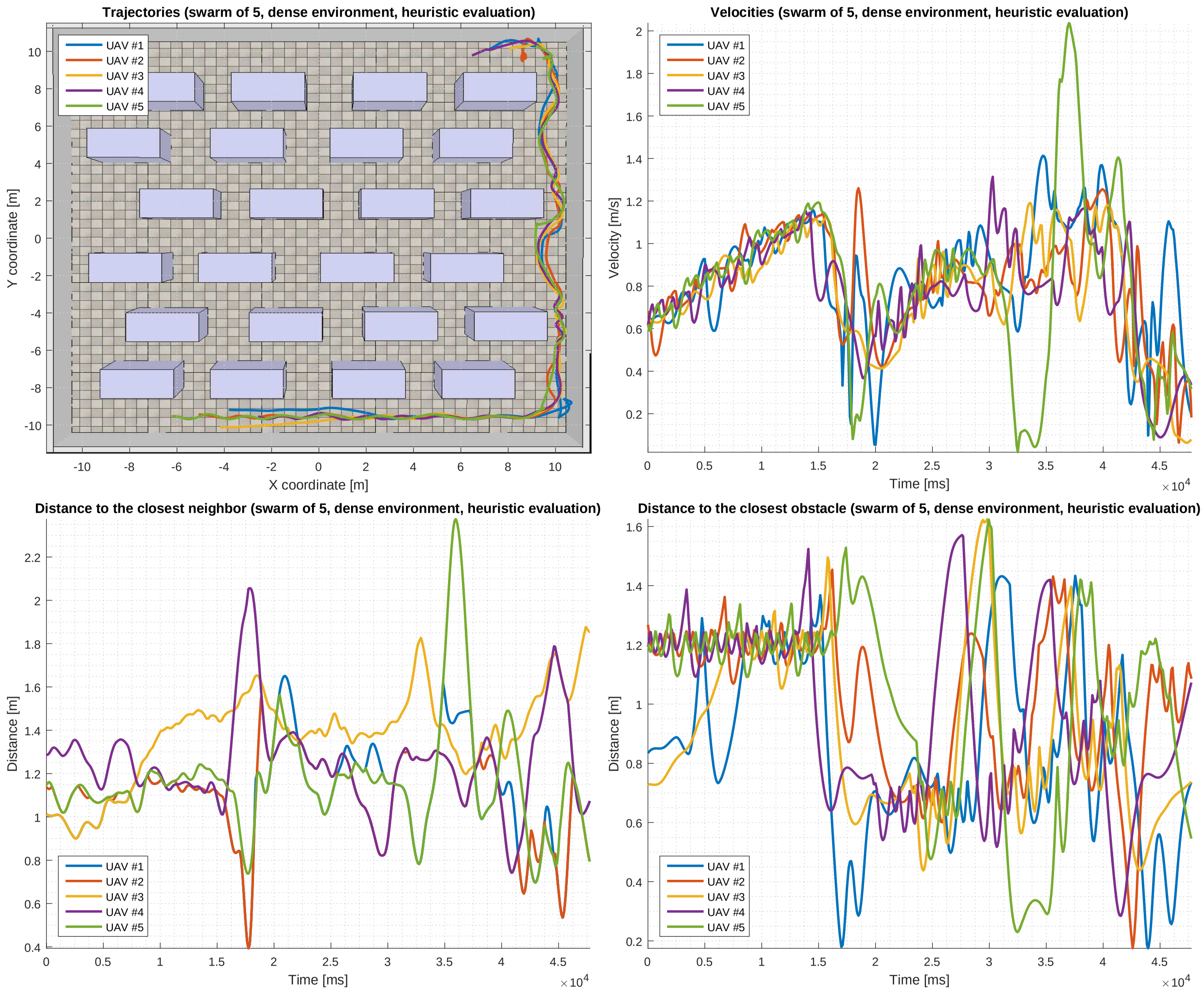

Distance to the closest neighbor (swarm of 5, simple environment, experimental evaluation)
Time [ms]
Distance [m]
$\times 10^4$
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

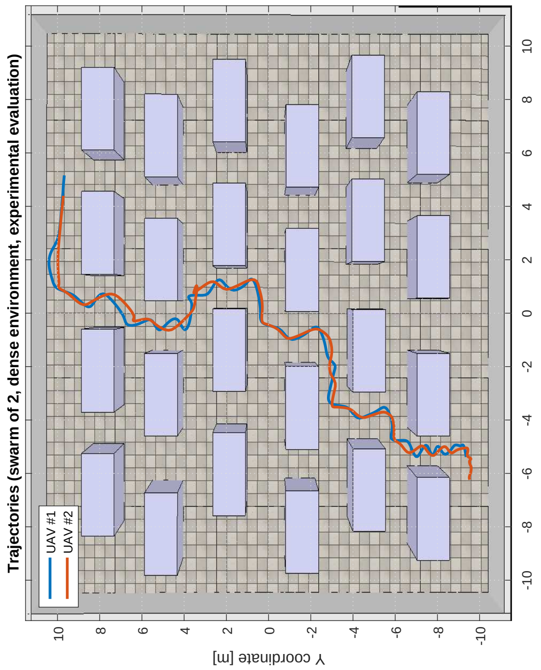

Distance to the closest obstacle (swarm of 5, simple environment, experimental evaluation)
Time [ms]
Distance [m]
$\times 10^4$
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

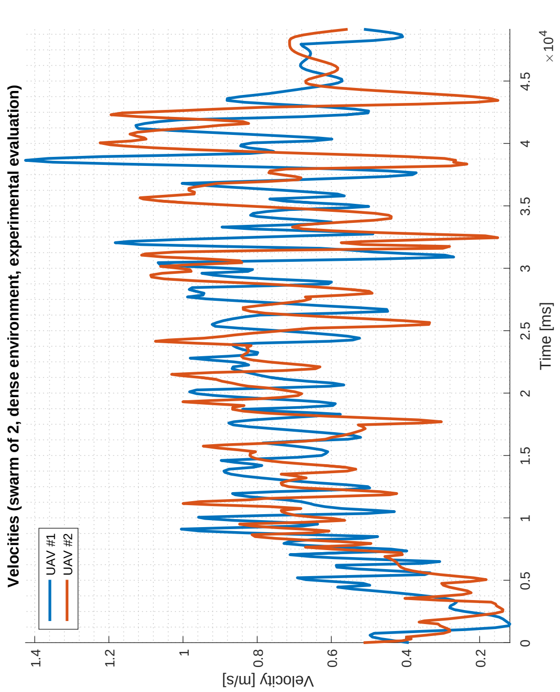
C

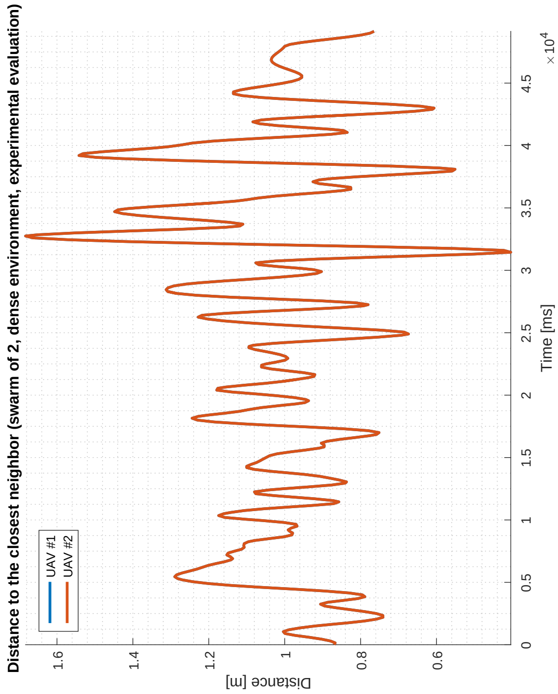
Appendix C.

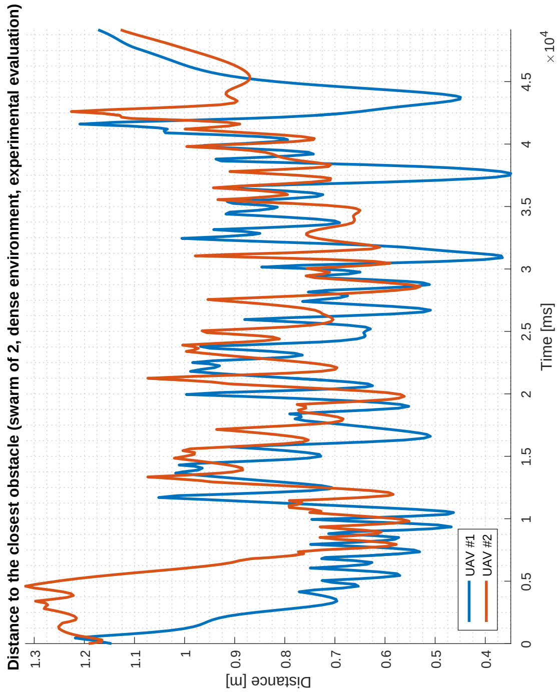
Graphs for the dense environment

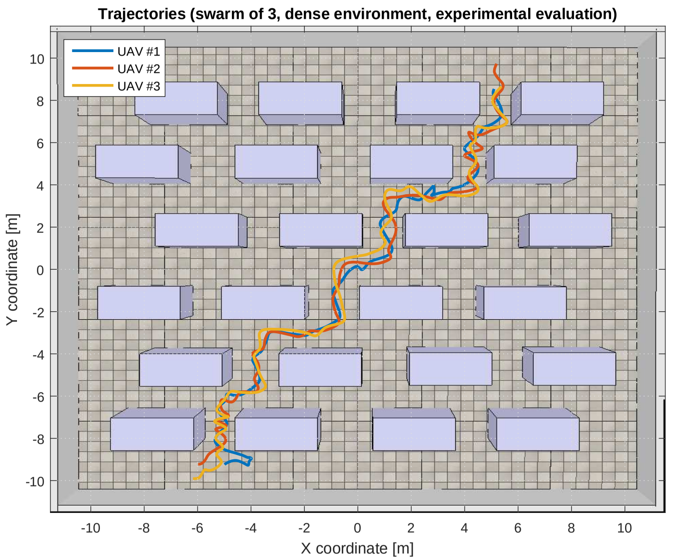

Trajectories (swarm of 5, dense environment, heuristic evaluation)
X coordinate [m]
Y coordinate [m]
Velocities (swarm of 5, dense environment, heuristic evaluation)
Velocity [m/s]
Time [ms]
Distance to the closest neighbor (swarm of 5, dense environment, heuristic evaluation)
Distance [m]
Time [ms]
Distance to the closest obstacle (swarm of 5, dense environment, heuristic evaluation)
Distance [m]
Time [ms]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

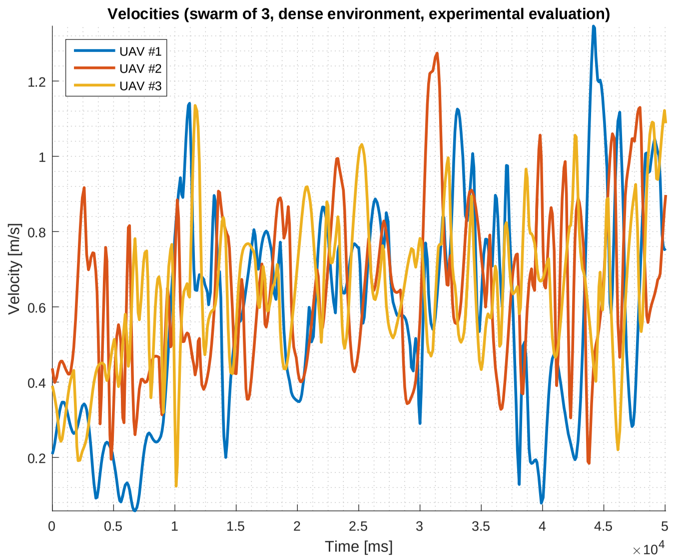

Trajectories (swarm of 2, dense environment, experimental evaluation)
UAV #1
UAV #2
X coordinate [m]
Y coordinate [m]

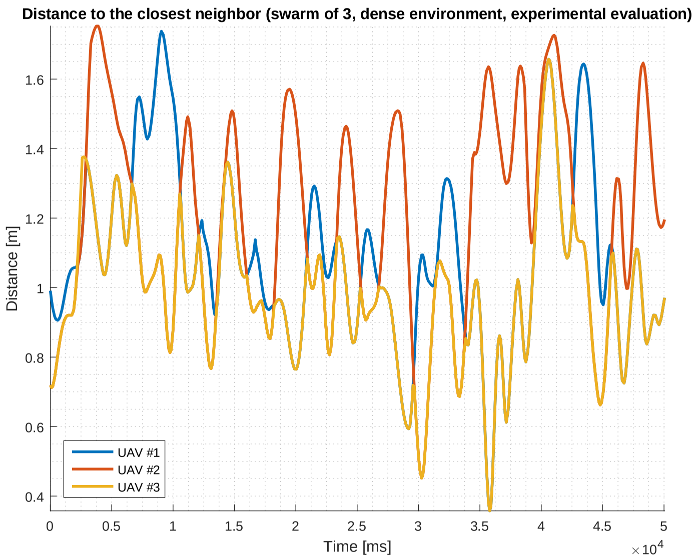

Velocities (swarm of 2, dense environment, experimental evaluation)
UAV #1
UAV #2
Time [ms]
Velocity [m/s]

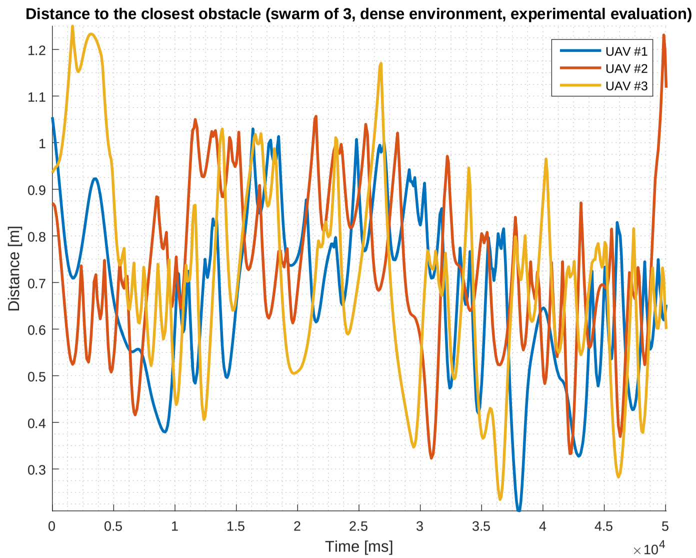

Distance to the closest neighbor (swarm of 2, dense environment, experimental evaluation)
UAV #1
UAV #2
Time [ms]
Distance [m]


Distance to the closest obstacle (swarm of 2, dense environment, experimental evaluation)
UAV #1
UAV #2
Time [ms]
Distance [m]


Trajectories (swarm of 3, dense environment, experimental evaluation)
UAV #1
UAV #2
UAV #3
Y coordinate [m]
X coordinate [m]
10
8
6
4
2
0
-2
-4
-6
-8
-10


Velocities (swarm of 3, dense environment, experimental evaluation)
UAV #1
UAV #2
UAV #3
Velocity [m/s]
1.2
1
0.8
0.6
0.4
0.2
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time [ms]
$\times 10^4$

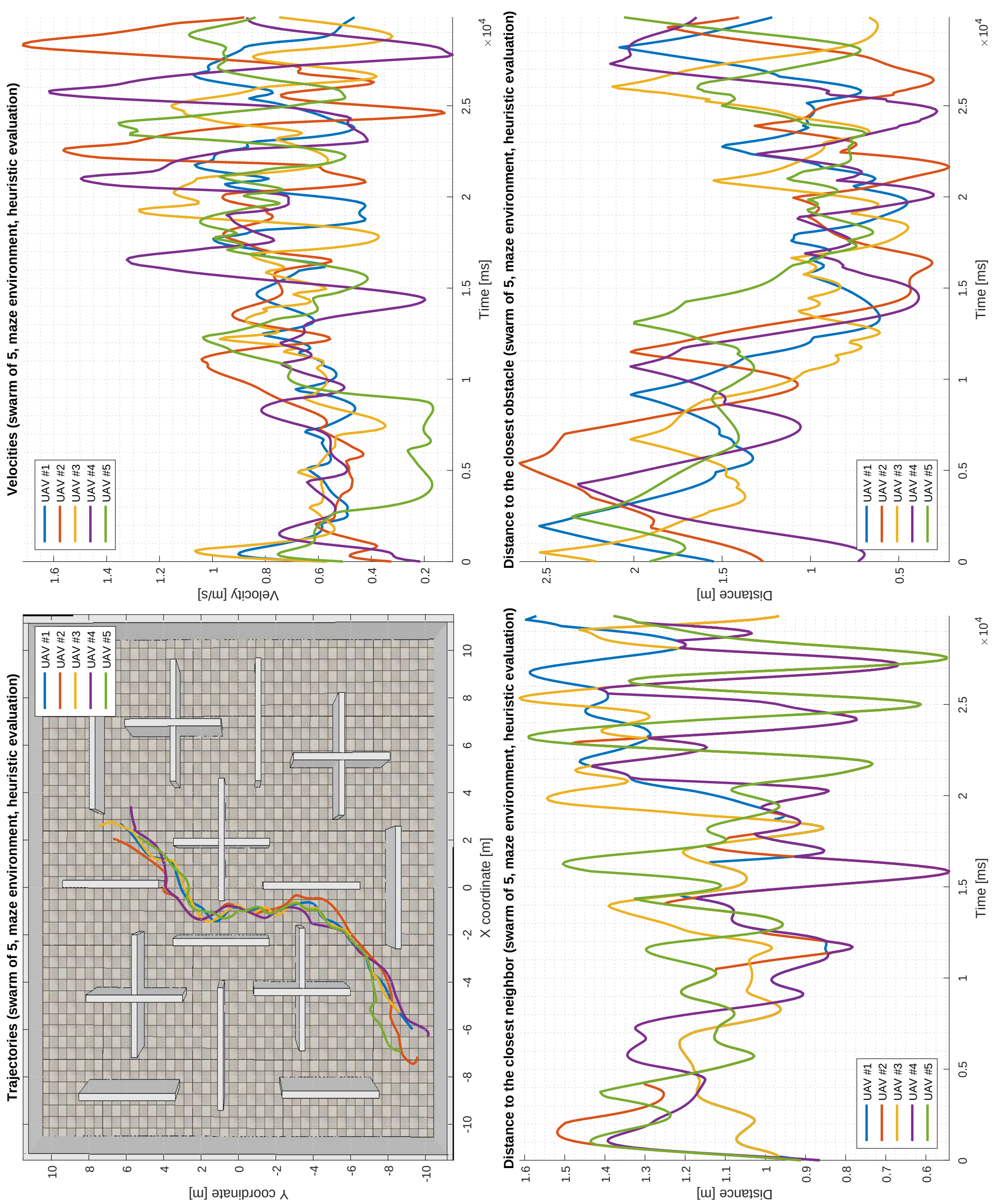

Distance to the closest neighbor (swarm of 3, dense environment, experimental evaluation)
UAV #1
UAV #2
UAV #3
Distance [m]
1.6
1.4
1.2
1
0.8
0.6
0.4
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time [ms]
$\times 10^4$

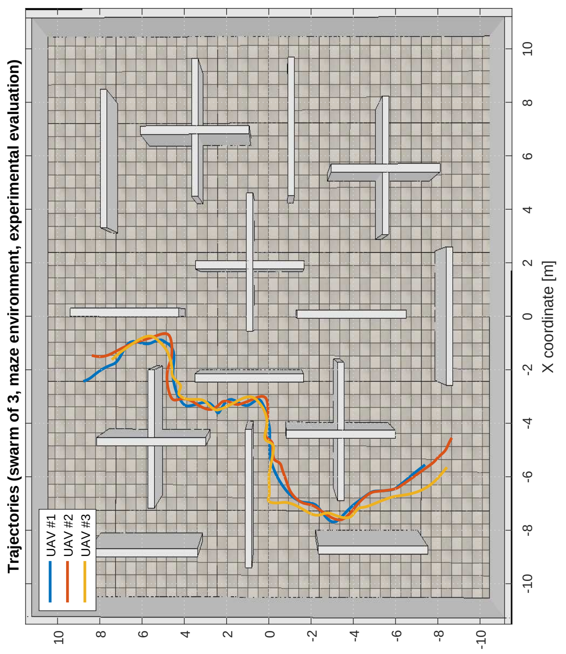

Distance to the closest obstacle (swarm of 3, dense environment, experimental evaluation)
UAV #1
UAV #2
UAV #3
Distance [m]
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time [ms]
$\times 10^4$

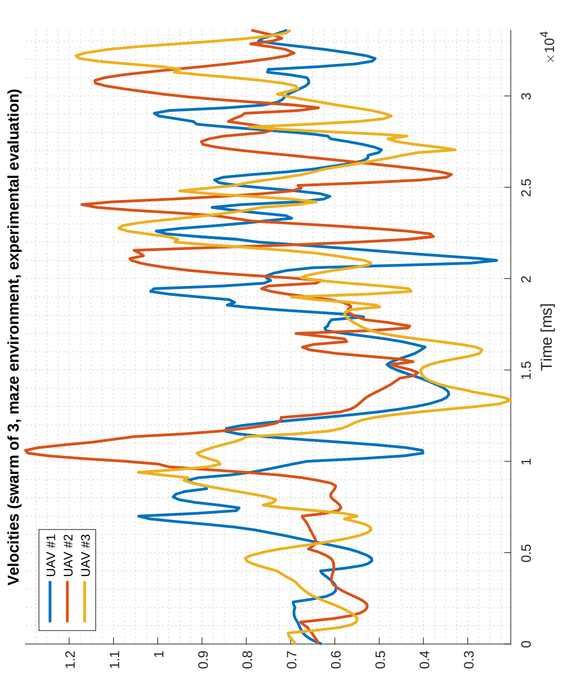
D

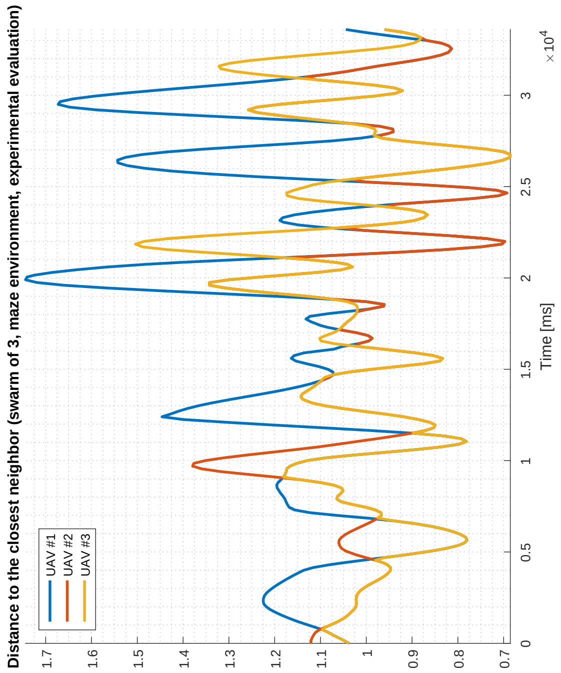
Appendix D.

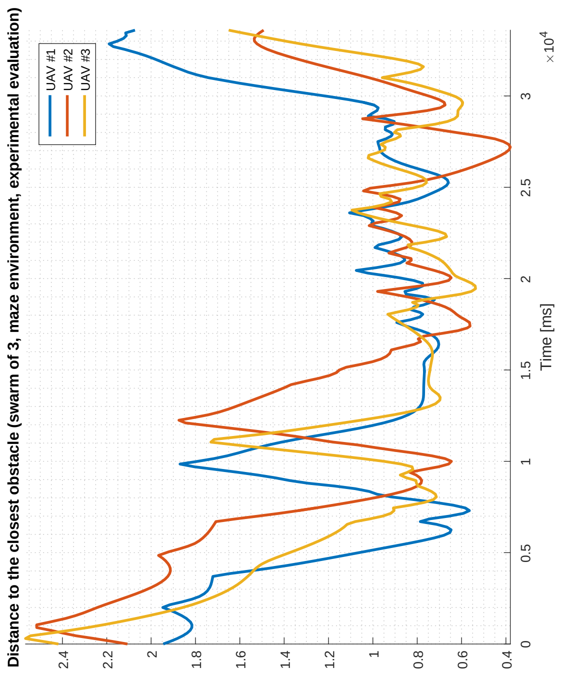
Graphs for the maze environment

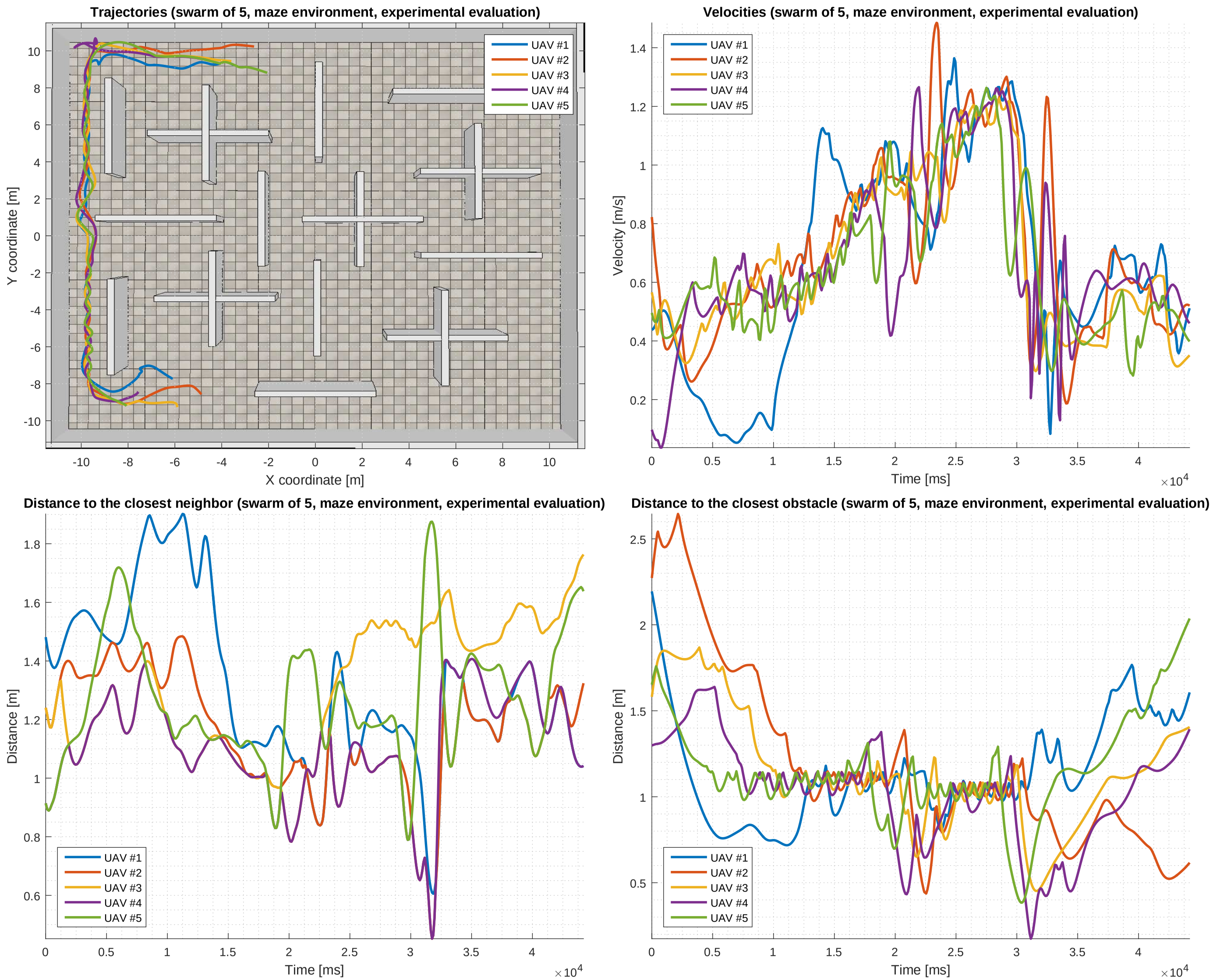

Trajectories (swarm of 5, maze environment, heuristic evaluation)
X coordinate [m]
Y coordinate [m]
Velocities (swarm of 5, maze environment, heuristic evaluation)
Time [ms]
Velocity [m/s]
Distance to the closest neighbor (swarm of 5, maze environment, heuristic evaluation)
Distance to the closest obstacle (swarm of 5, maze environment, heuristic evaluation)
Distance [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5


Trajectories (swarm of 3, maze environment, experimental evaluation)
UAV #1
UAV #2
UAV #3
X coordinate [m]
Y coordinate [m]


Velocities (swarm of 3, maze environment, experimental evaluation)
UAV #1
UAV #2
UAV #3
Time [ms]
Velocity [m/s]
×10^4


Distance to the closest neighbor (swarm of 3, maze environment, experimental evaluation)
UAV #1
UAV #2
UAV #3
Time [ms]
Distance [m]
×10^4


Distance to the closest obstacle (swarm of 3, maze environment, experimental evaluation)
UAV #1
UAV #2
UAV #3
Time [ms]
Distance [m]
×10^4


Trajectories (swarm of 5, maze environment, experimental evaluation)
Y coordinate [m]
X coordinate [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
Velocities (swarm of 5, maze environment, experimental evaluation)
Velocity [m/s]
Time [ms]
×10^4
Distance to the closest neighbor (swarm of 5, maze environment, experimental evaluation)
Distance [m]
Distance to the closest obstacle (swarm of 5, maze environment, experimental evaluation)

E

Appendix E.

Results of the swarm splitting algorithm

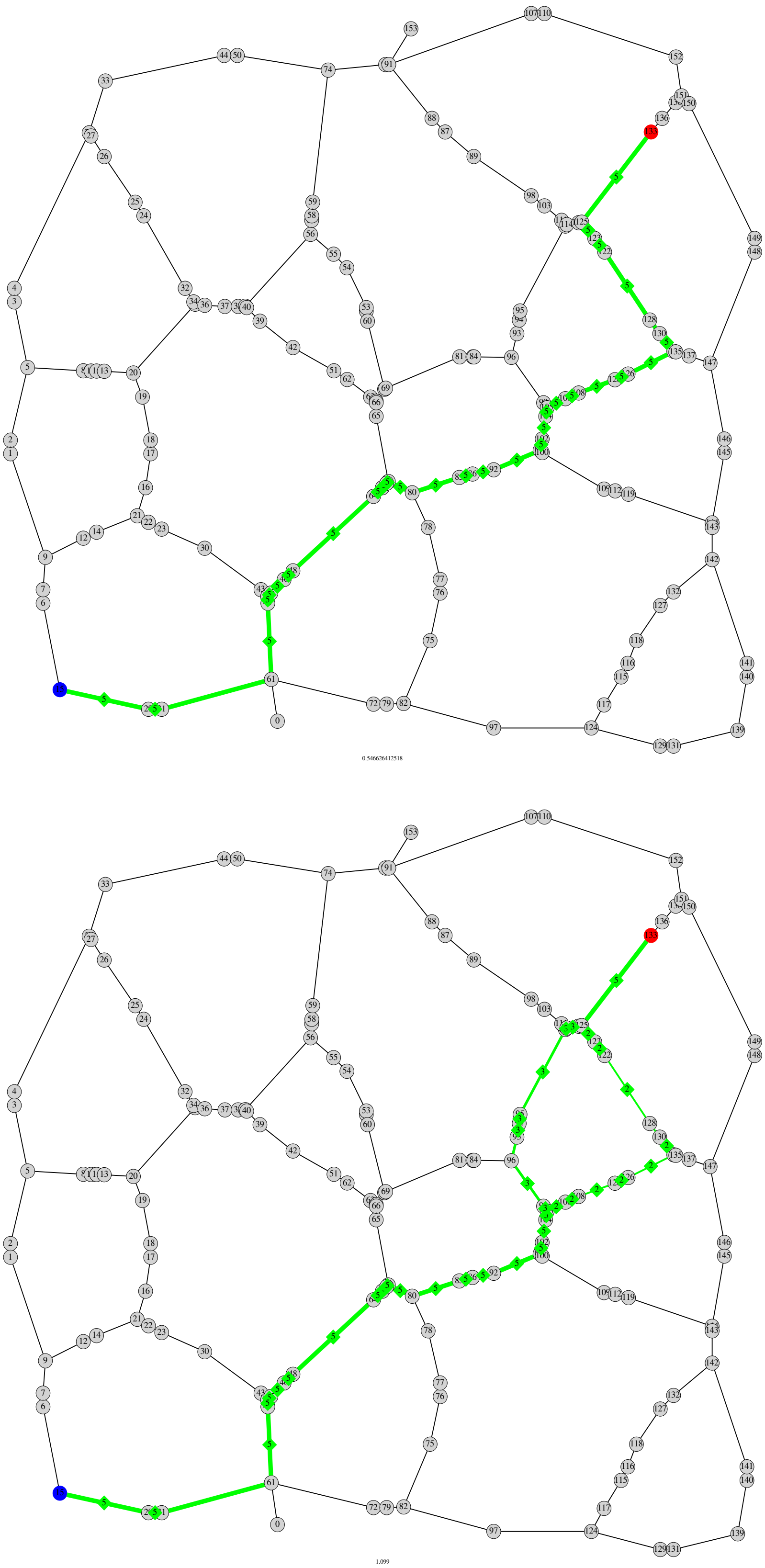


Figure E.1.: Swarm splitting algorithm result (simple environment, heuristic evaluation (top) / evaluation by simulation (bottom), path from node 15 to node 133)

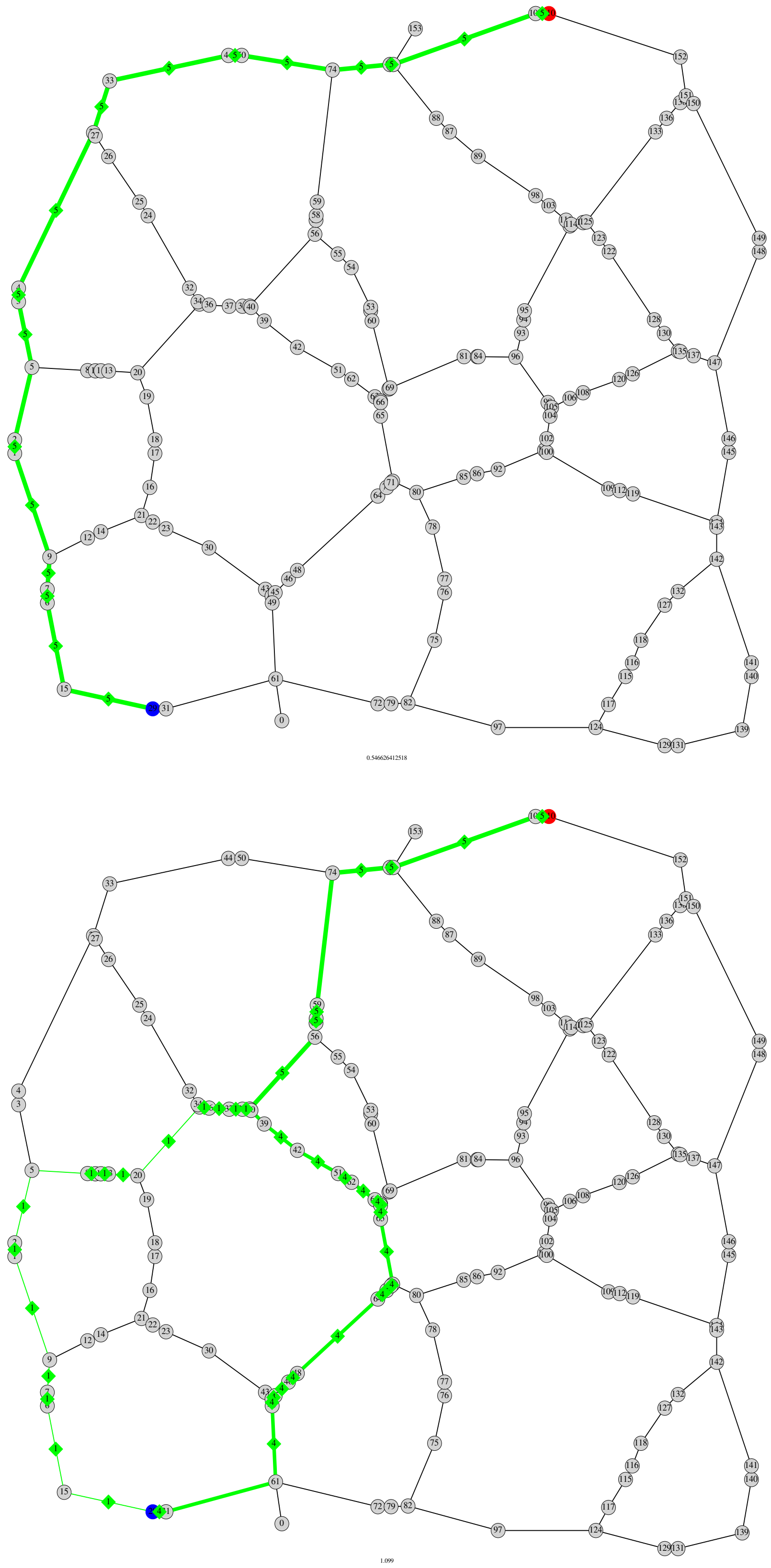


Figure E.2.: Swarm splitting algorithm result (simple environment, heuristic evaluation (top) / evaluation by simulation (bottom), path from node 29 to node 110)

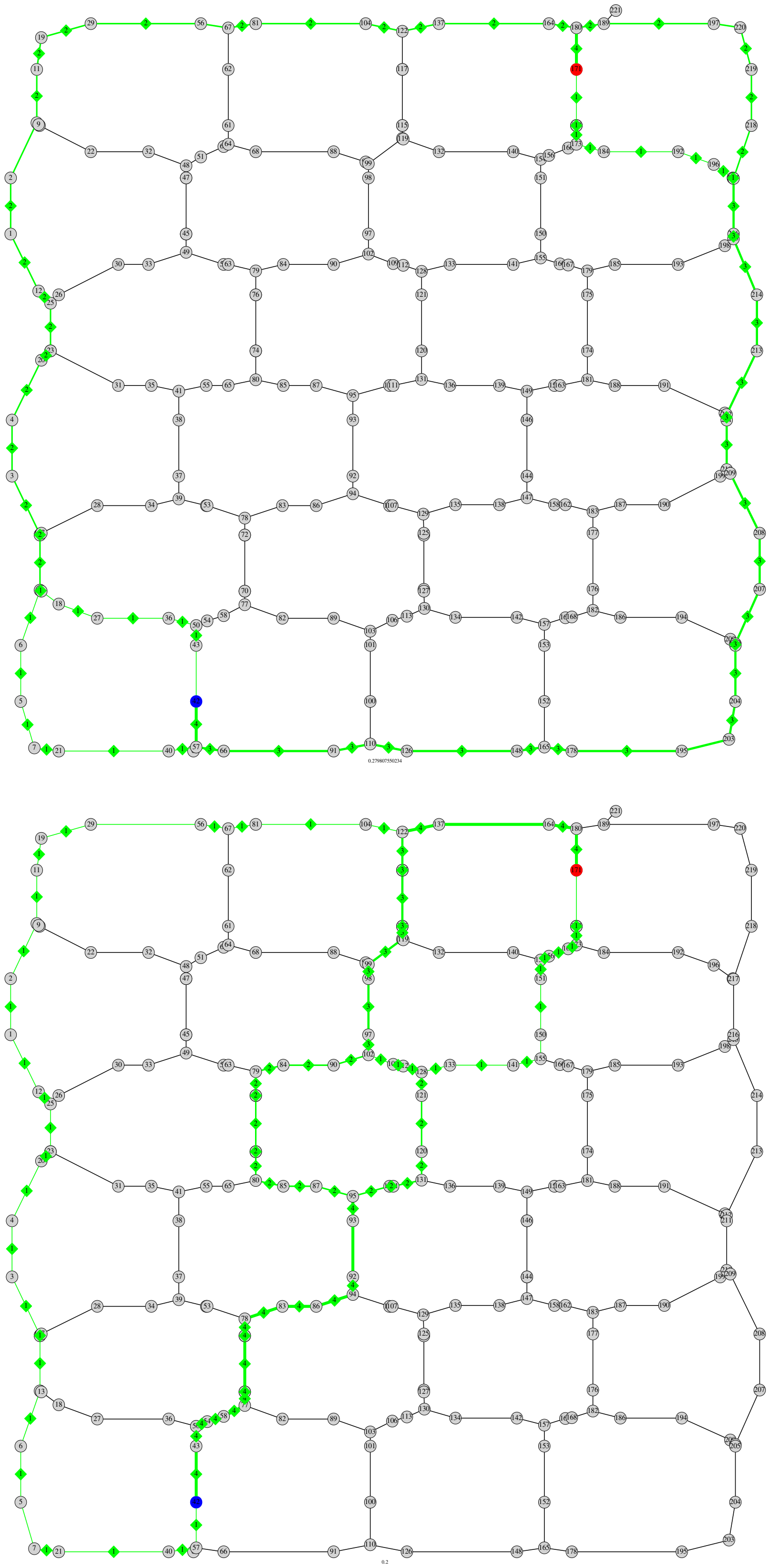


Figure E.3.: Swarm splitting algorithm result (dense environment, heuristic evaluation (top) / evaluation by simulation (bottom), path from node 42 to node 171)

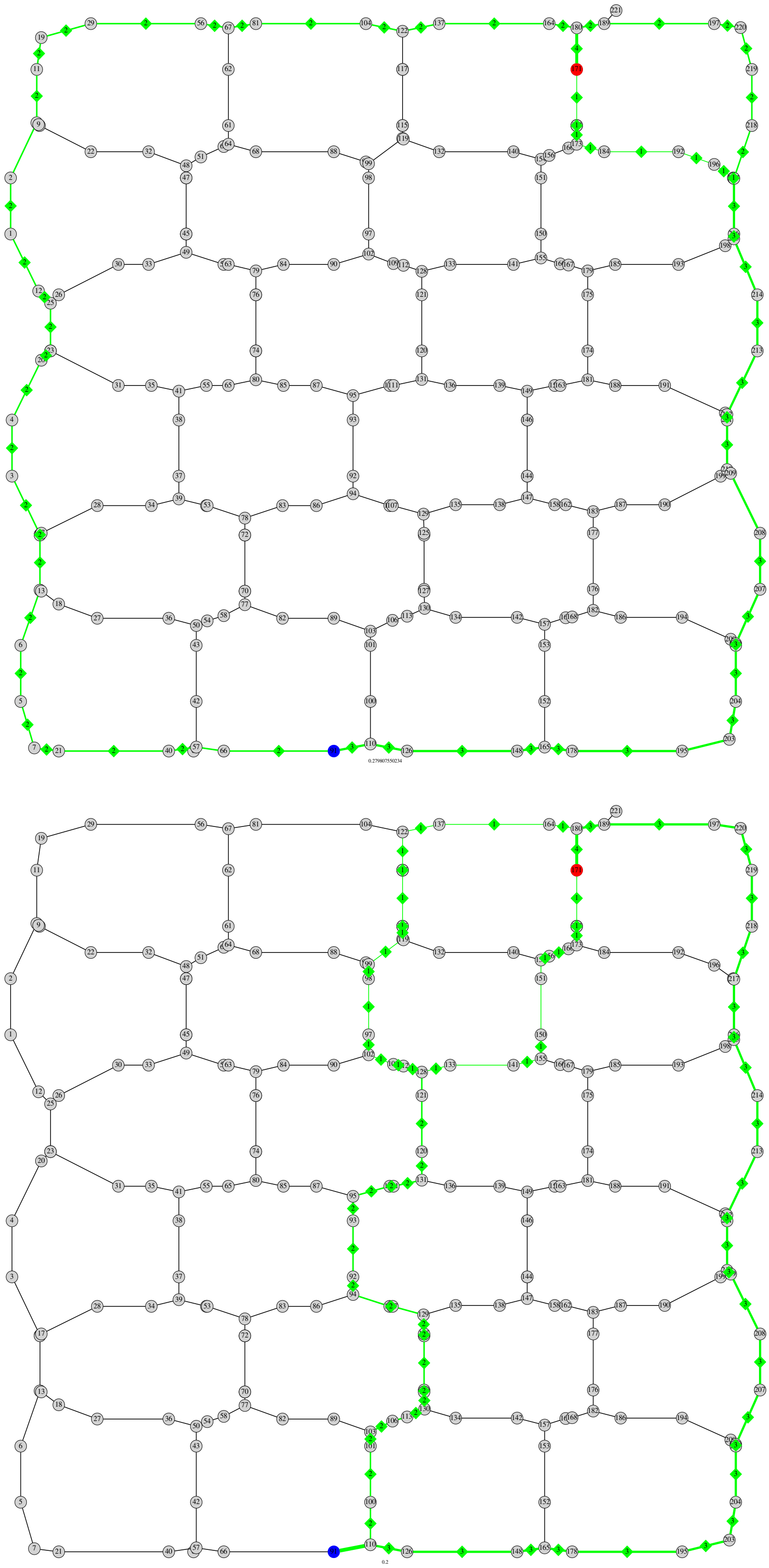


Figure E.4.: Swarm splitting algorithm result (dense environment, heuristic evaluation (top) / evaluation by simulation (bottom), path from node 91 to node 171)

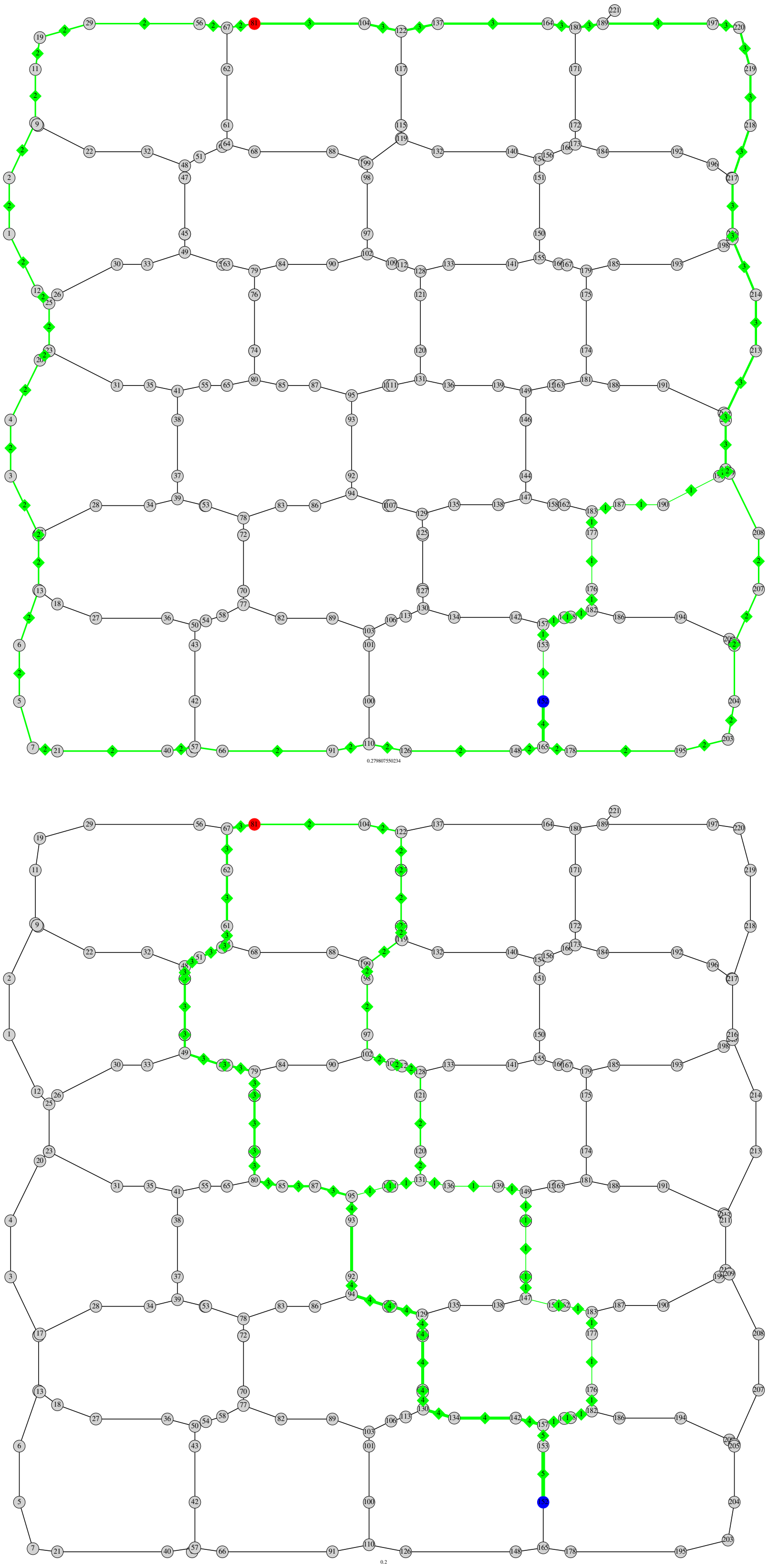


Figure E.5.: Swarm splitting algorithm result (dense environment, heuristic evaluation (top) / evaluation by simulation (bottom), path from node 152 to node 81)

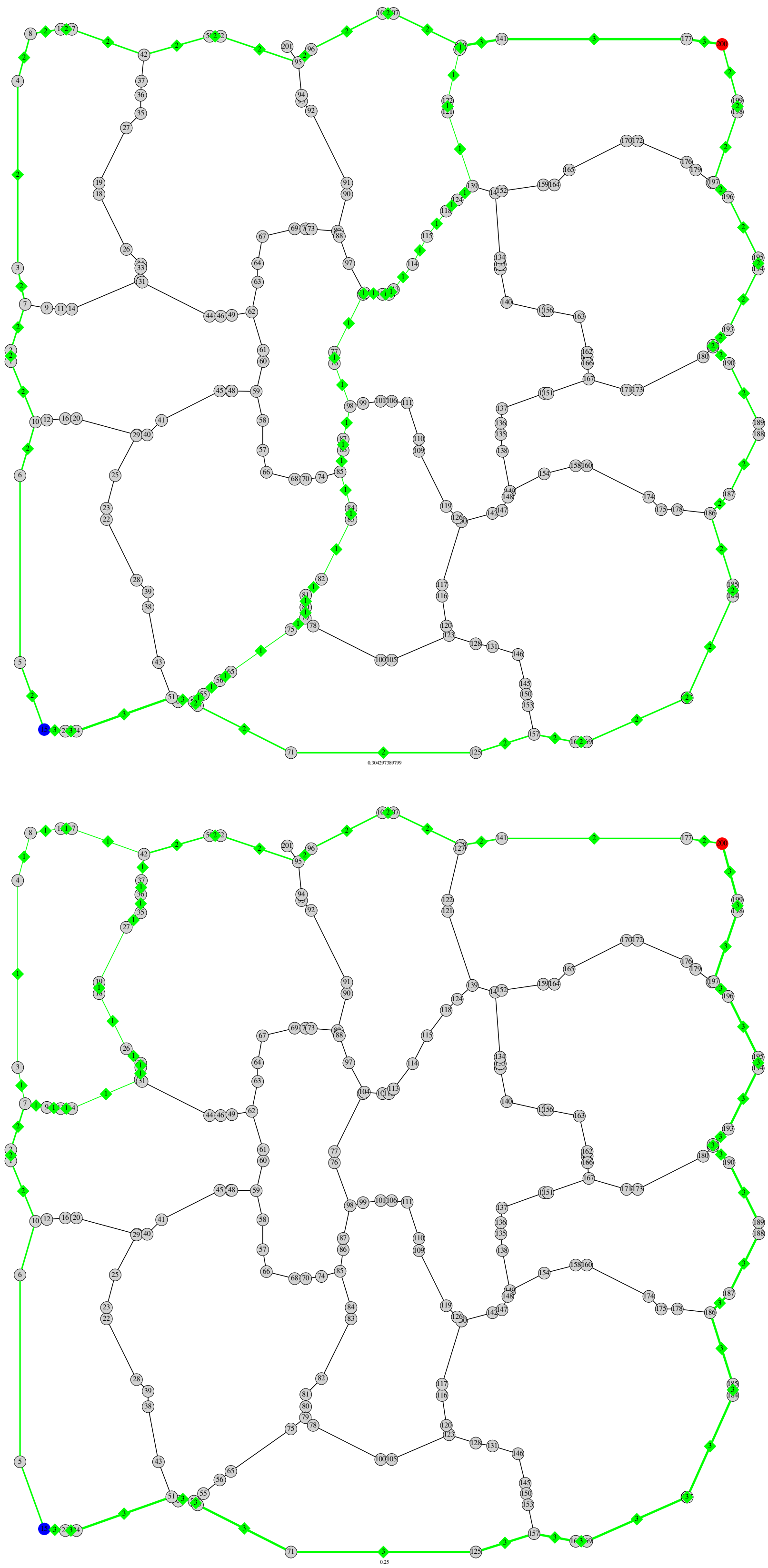


Figure E.6.: Swarm splitting algorithm result (maze environment, heuristic evaluation (top) / evaluation by simulation (bottom), path from node 15 to node 200)

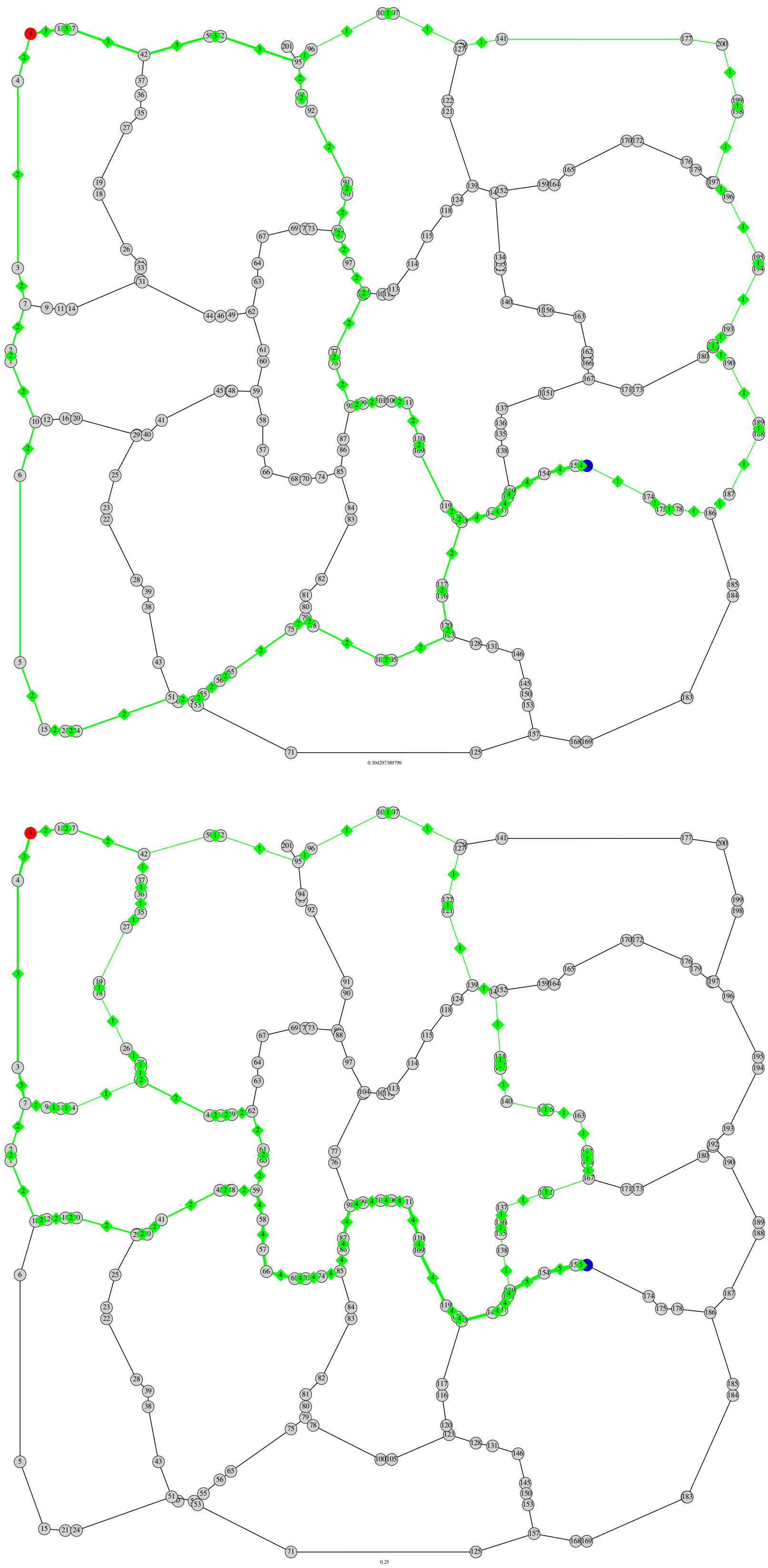
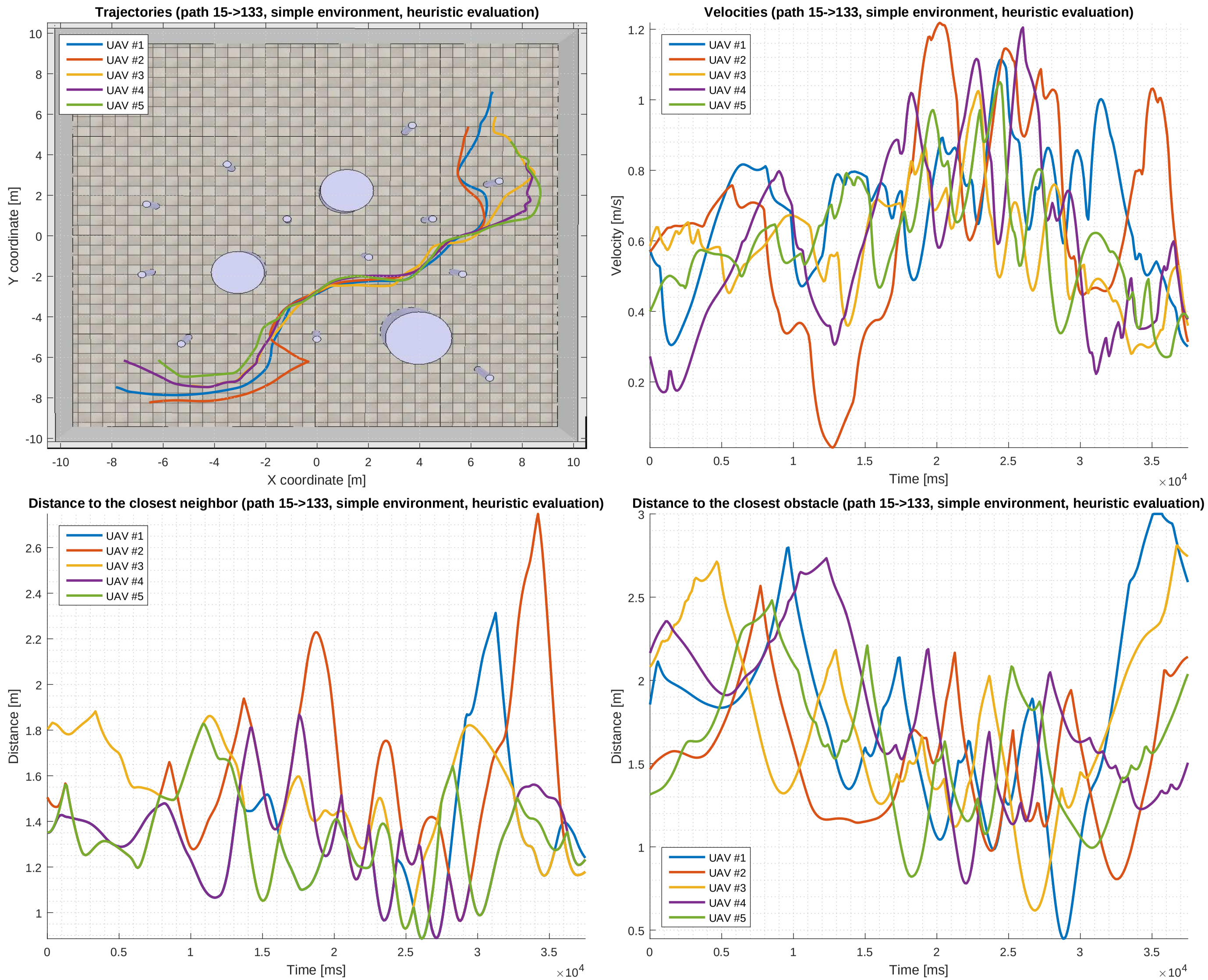


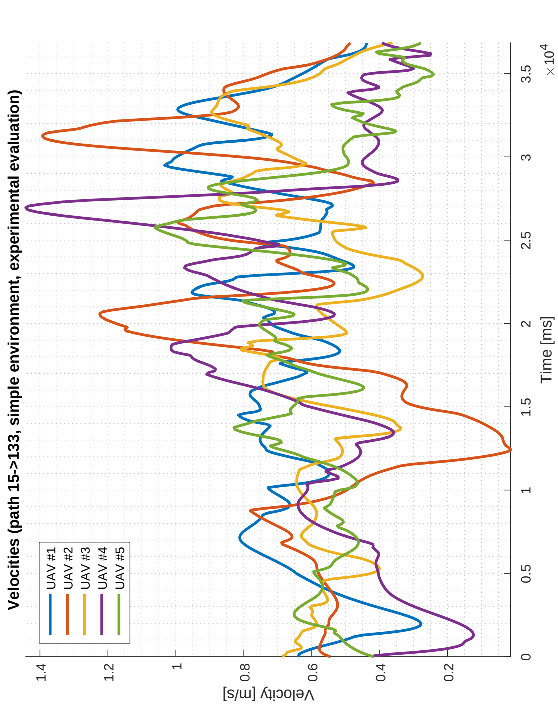
Figure E.7.: Swarm splitting algorithm result (maze environment, heuristic evaluation (top) / evaluation by simulation (bottom), path from node 160 to node 8)

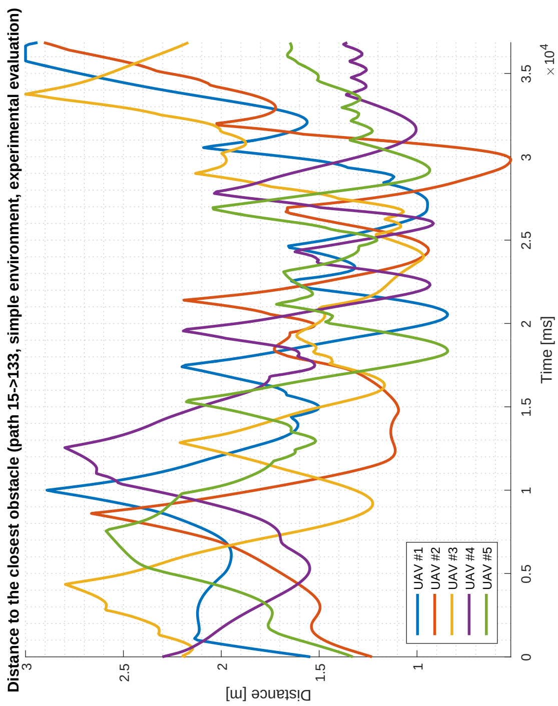
F

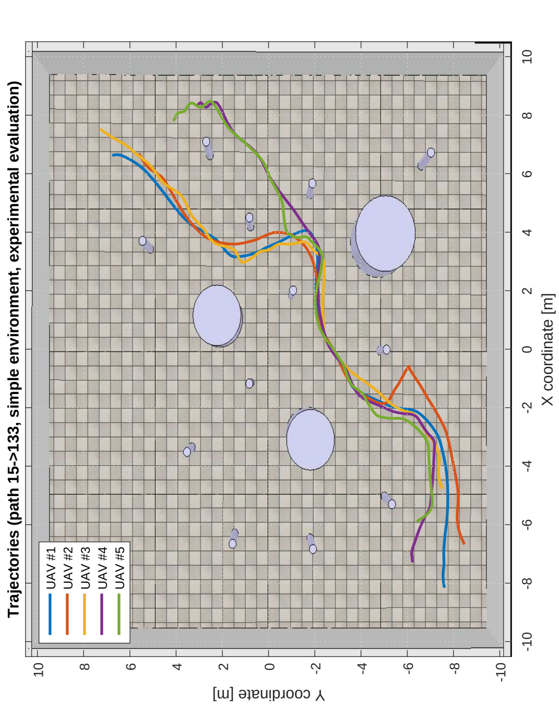
Appendix F.

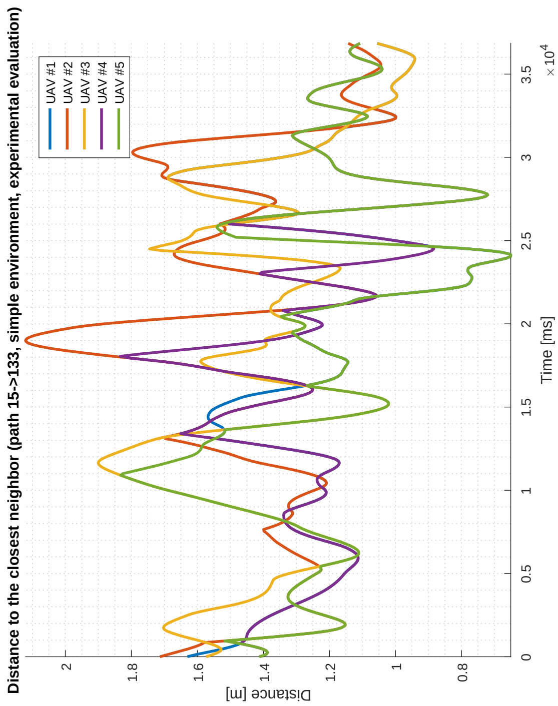
Graphs for swarm splitting

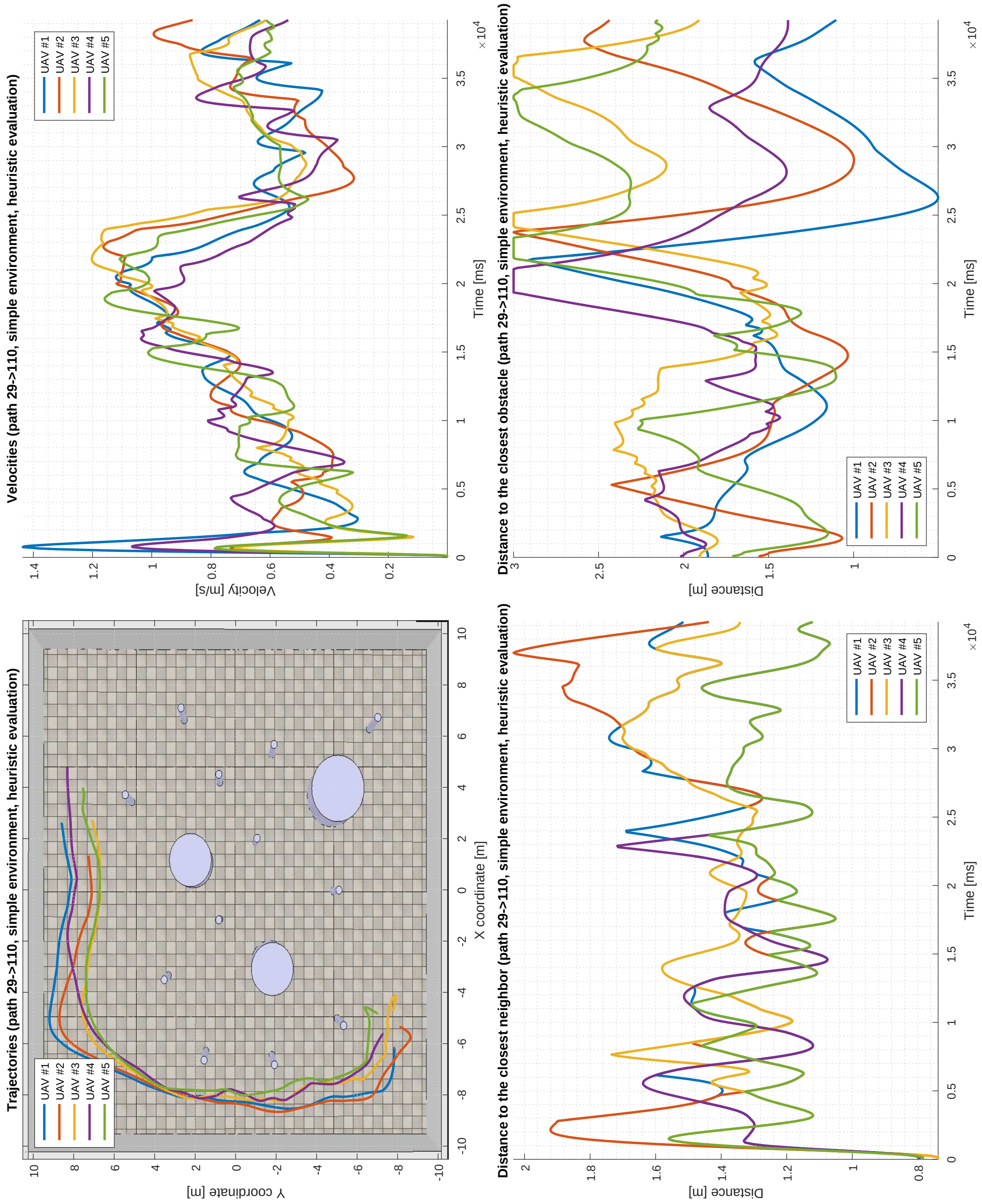

Trajectories (path 15->133, simple environment, heuristic evaluation)
Y coordinate [m]
X coordinate [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
Velocities (path 15->133, simple environment, heuristic evaluation)
Velocity [m/s]
Time [ms]
$\times 10^4$
Distance to the closest neighbor (path 15->133, simple environment, heuristic evaluation)
Distance [m]
Time [ms]
Distance to the closest obstacle (path 15->133, simple environment, heuristic evaluation)
Distance [m]
Time [ms]

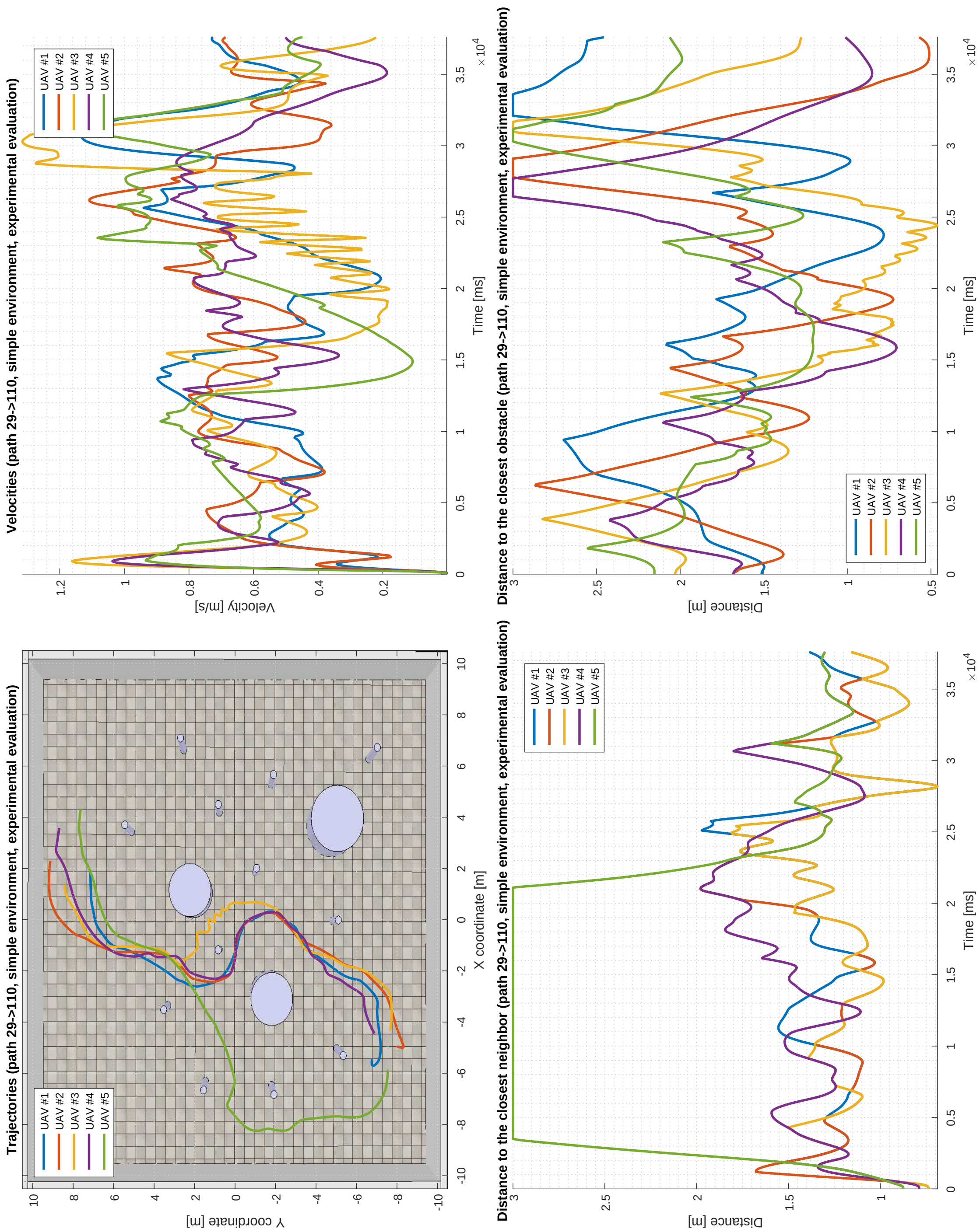

Velocities (path 15->133, simple environment, experimental evaluation)
Velocity [m/s]
Time [ms]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5


Distance to the closest obstacle (path 15->133, simple environment, experimental evaluation)
Distance [m]
Time [ms]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

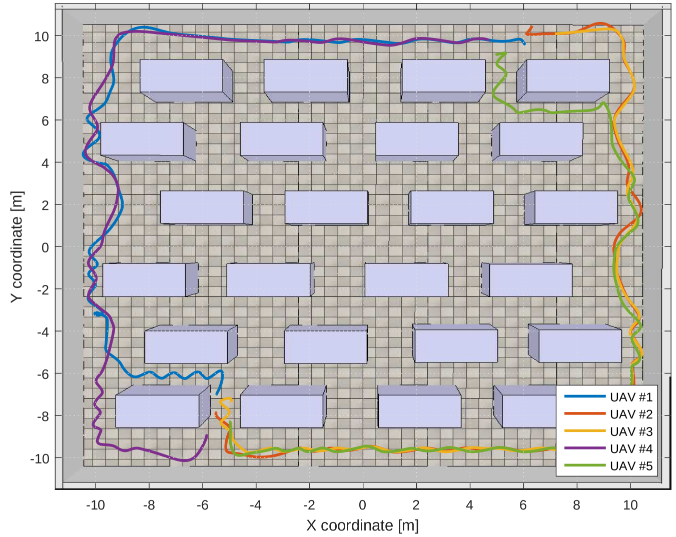

Trajectories (path 15->133, simple environment, experimental evaluation)
Y coordinate [m]
X coordinate [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5


Distance to the closest neighbor (path 15->133, simple environment, experimental evaluation)
Distance [m]
Time [ms]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

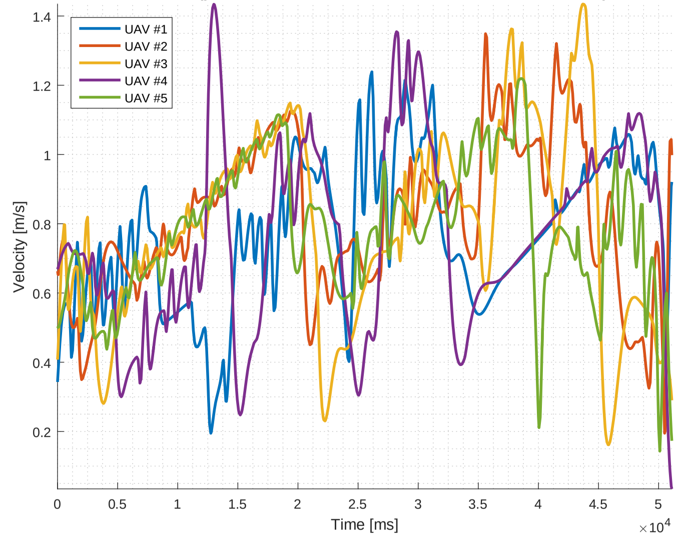

Trajectories (path 29->110, simple environment, heuristic evaluation)
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
X coordinate [m]
Y coordinate [m]
Velocities (path 29->110, simple environment, heuristic evaluation)
Velocity [m/s]
Time [ms]
$\times 10^4$
Distance to the closest neighbor (path 29->110, simple environment, heuristic evaluation)
Distance [m]
Time [ms]
Distance to the closest obstacle (path 29->110, simple environment, heuristic evaluation)
Distance [m]
Time [ms]


Trajectories (path 29->110, simple environment, experimental evaluation)
X coordinate [m]
Y coordinate [m]
Velocities (path 29->110, simple environment, experimental evaluation)
Velocity [m/s]
Time [ms]
Distance to the closest neighbor (path 29->110, simple environment, experimental evaluation)
Distance to the closest obstacle (path 29->110, simple environment, experimental evaluation)
Distance [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

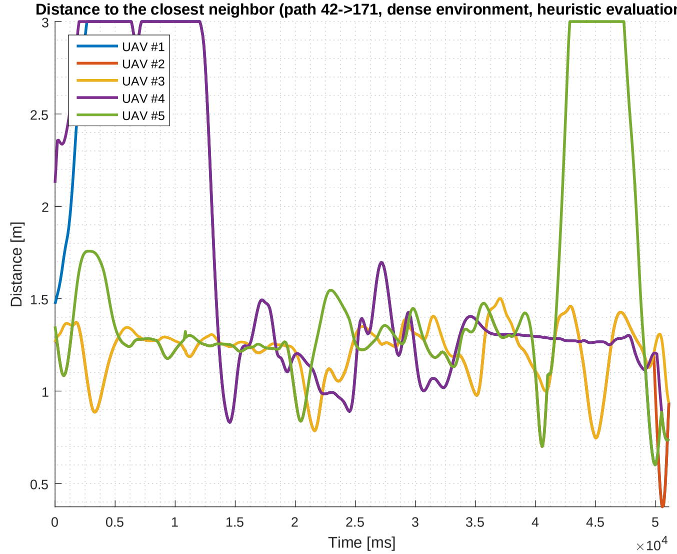
Trajectories (path 42->171, dense environment, heuristic evaluation)



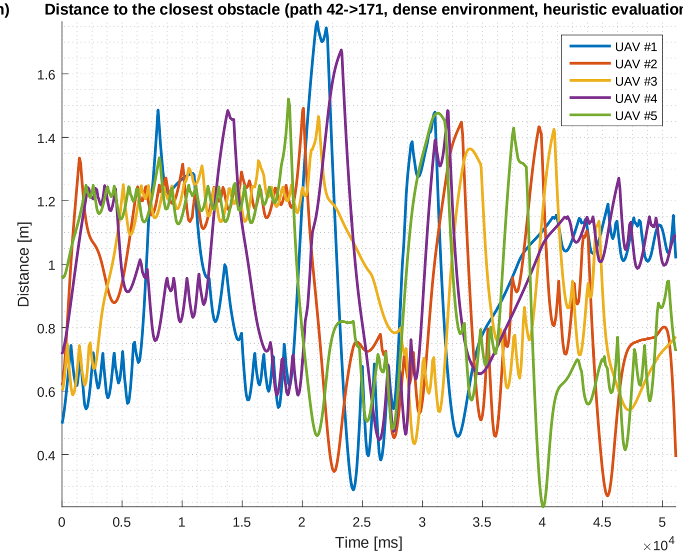
Velocities (path 42->171, dense environment, heuristic evaluation)

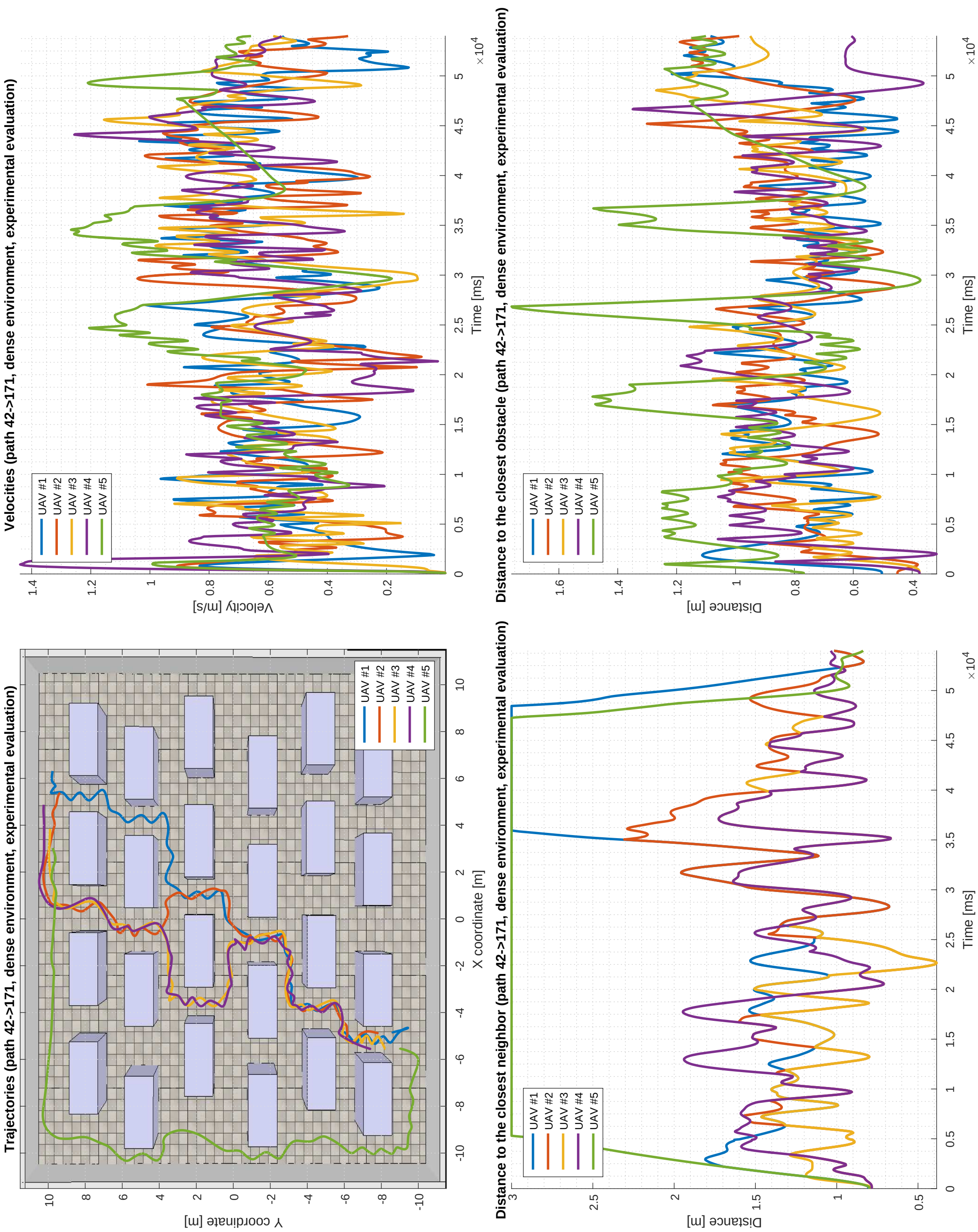


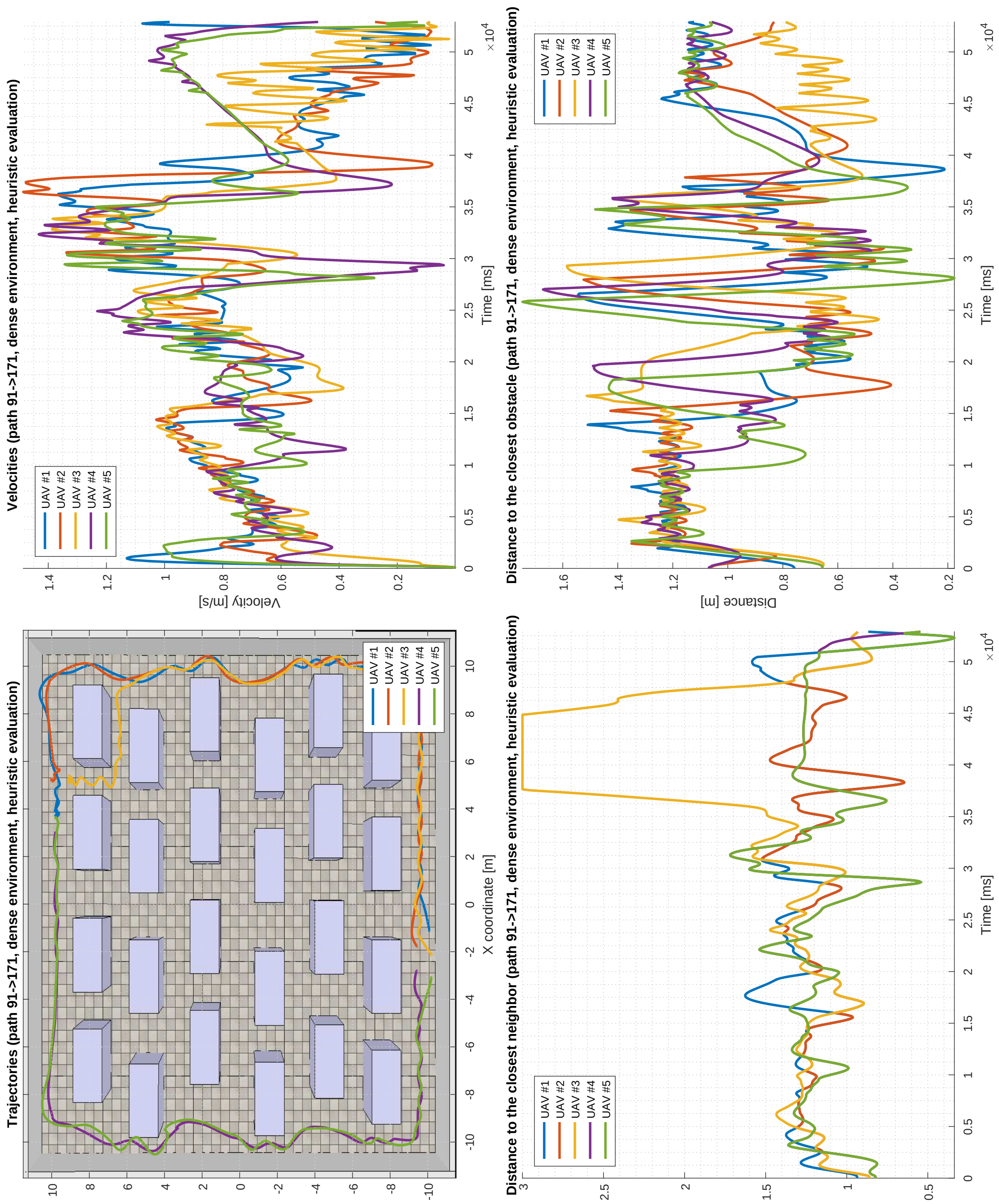
Distance to the closest neighbor (path 42->171, dense environment, heuristic evaluation)

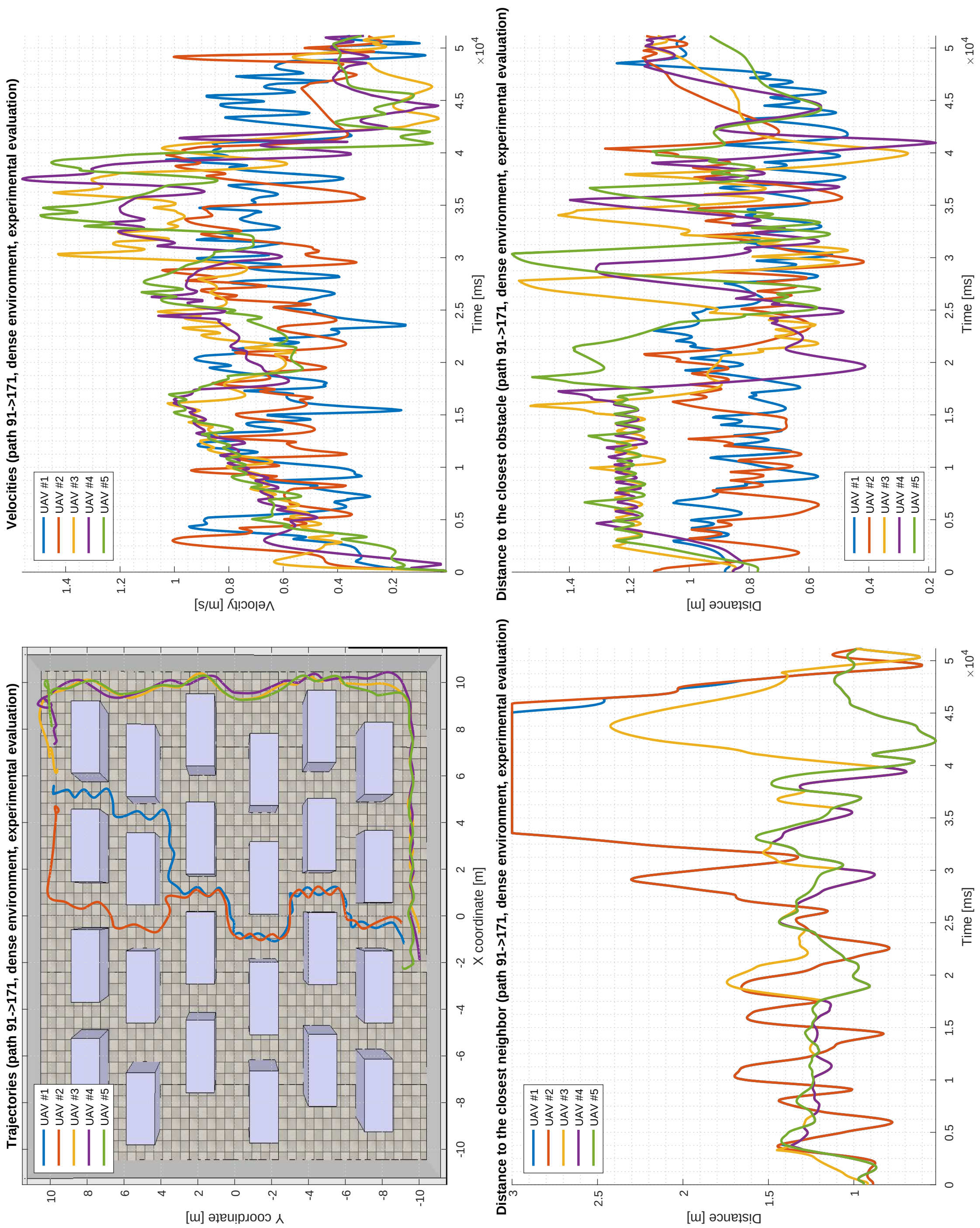


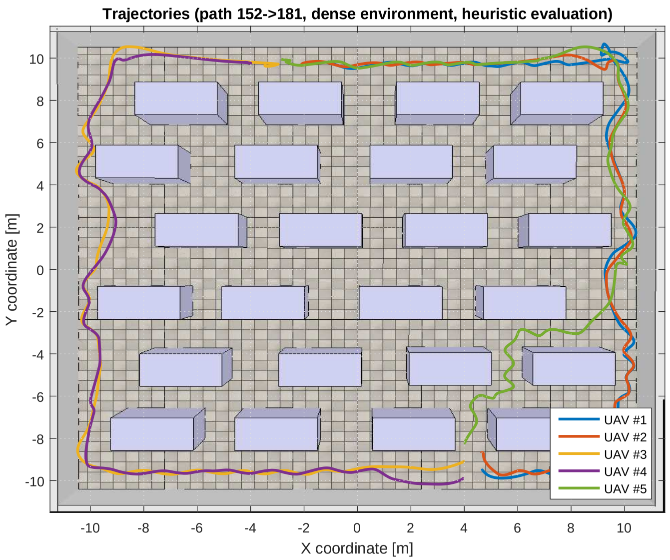
Distance to the closest obstacle (path 42->171, dense environment, heuristic evaluation)

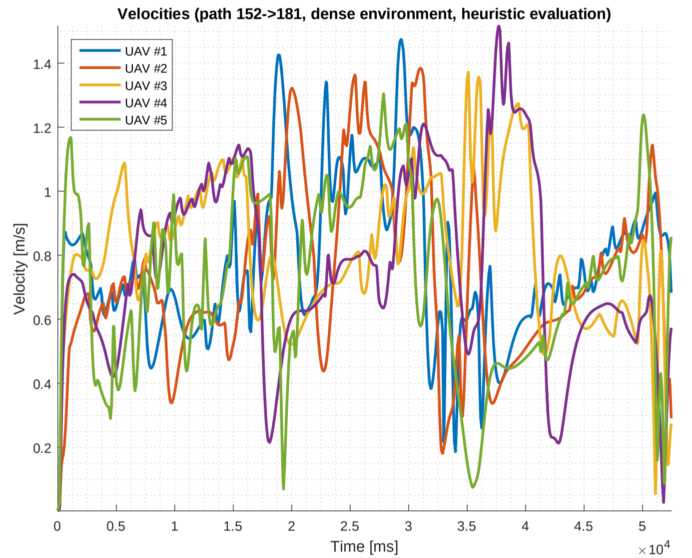


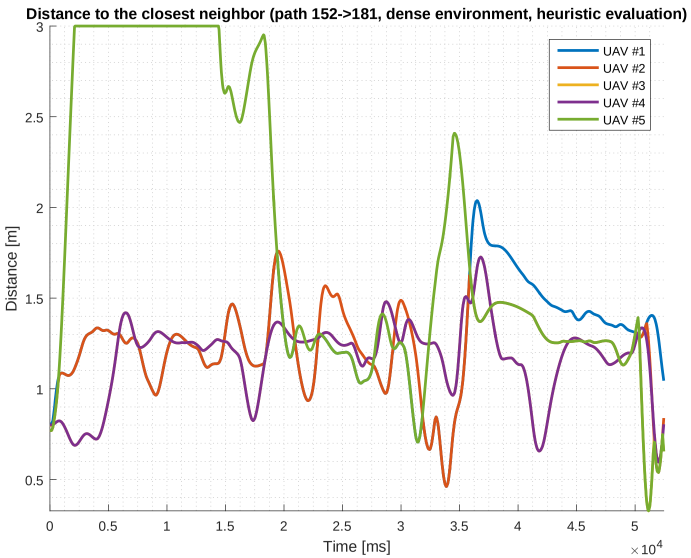

Trajectories (path 42->171, dense environment, experimental evaluation)
X coordinate [m]
Y coordinate [m]
Velocities (path 42->171, dense environment, experimental evaluation)
Time [ms]
Velocity [m/s]
Distance to the closest neighbor (path 42->171, dense environment, experimental evaluation)
Distance to the closest obstacle (path 42->171, dense environment, experimental evaluation)
Distance [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

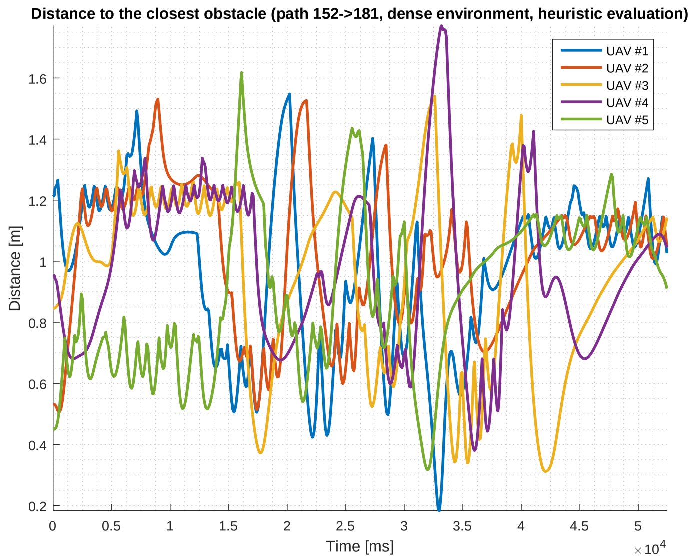

Trajectories (path 91->171, dense environment, heuristic evaluation)
X coordinate [m]
Y coordinate [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
Velocities (path 91->171, dense environment, heuristic evaluation)
Time [ms]
Velocity [m/s]
Distance to the closest neighbor (path 91->171, dense environment, heuristic evaluation)
Time [ms]
Distance [m]
Distance to the closest obstacle (path 91->171, dense environment, heuristic evaluation)
Time [ms]
Distance [m]

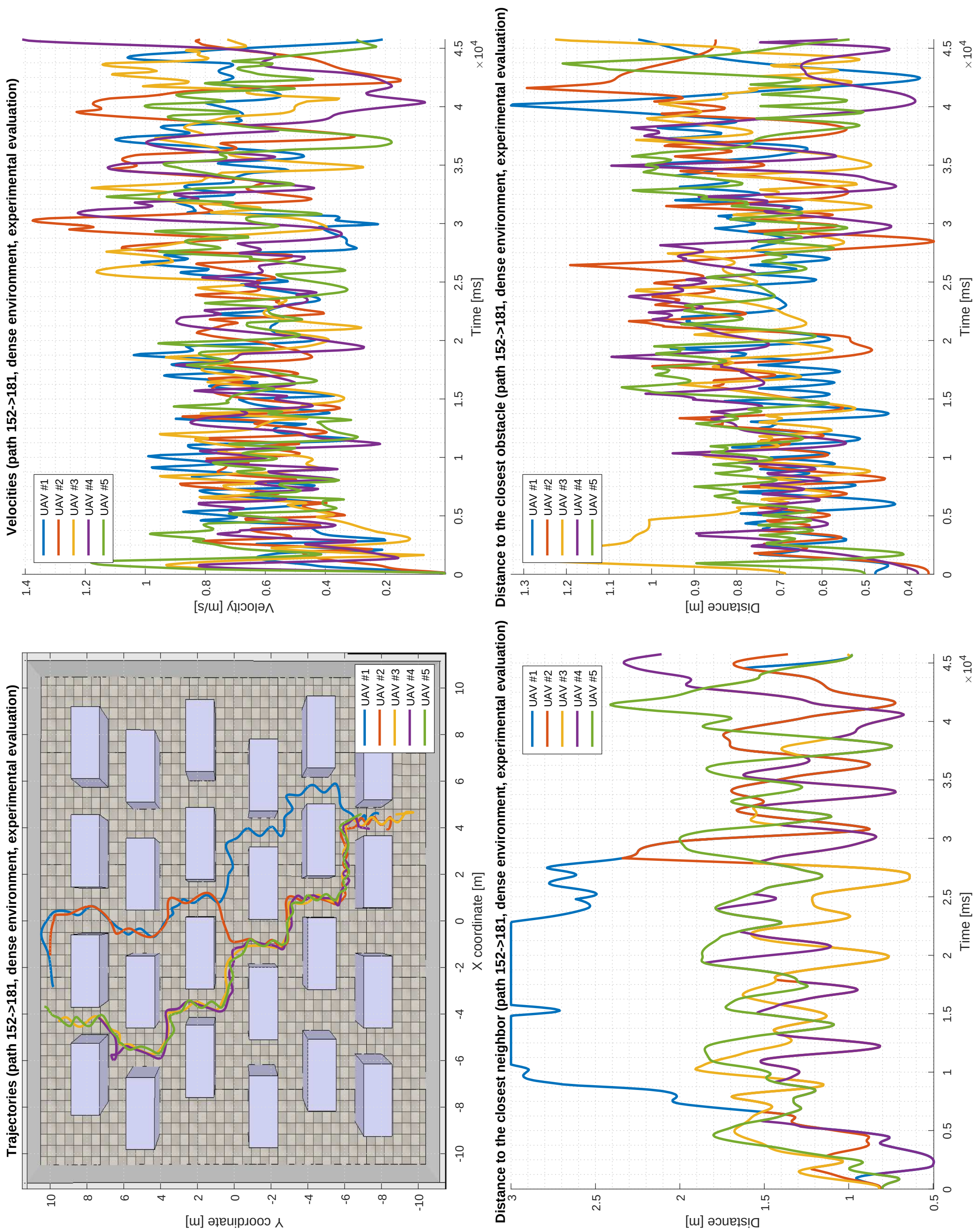

Trajectories (path 91->171, dense environment, experimental evaluation)
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
X coordinate [m]
Y coordinate [m]
Velocities (path 91->171, dense environment, experimental evaluation)
Velocity [m/s]
Time [ms]
Distance to the closest neighbor (path 91->171, dense environment, experimental evaluation)
Distance to the closest obstacle (path 91->171, dense environment, experimental evaluation)
Distance [m]
$\times 10^4$


Trajectories (path 152->181, dense environment, heuristic evaluation)
Y coordinate [m]
X coordinate [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

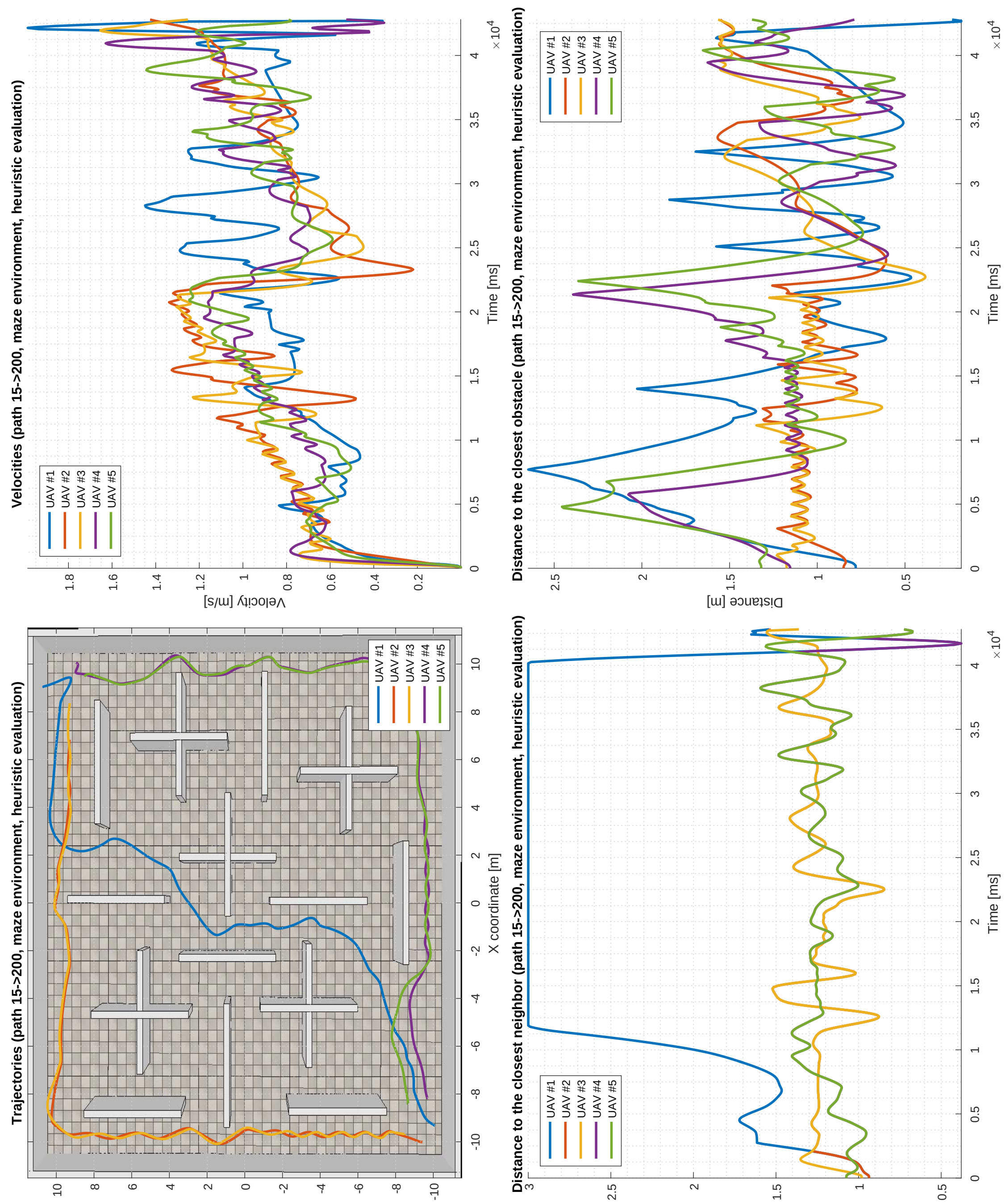

Velocities (path 152->181, dense environment, heuristic evaluation)
Velocity [m/s]
Time [ms]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

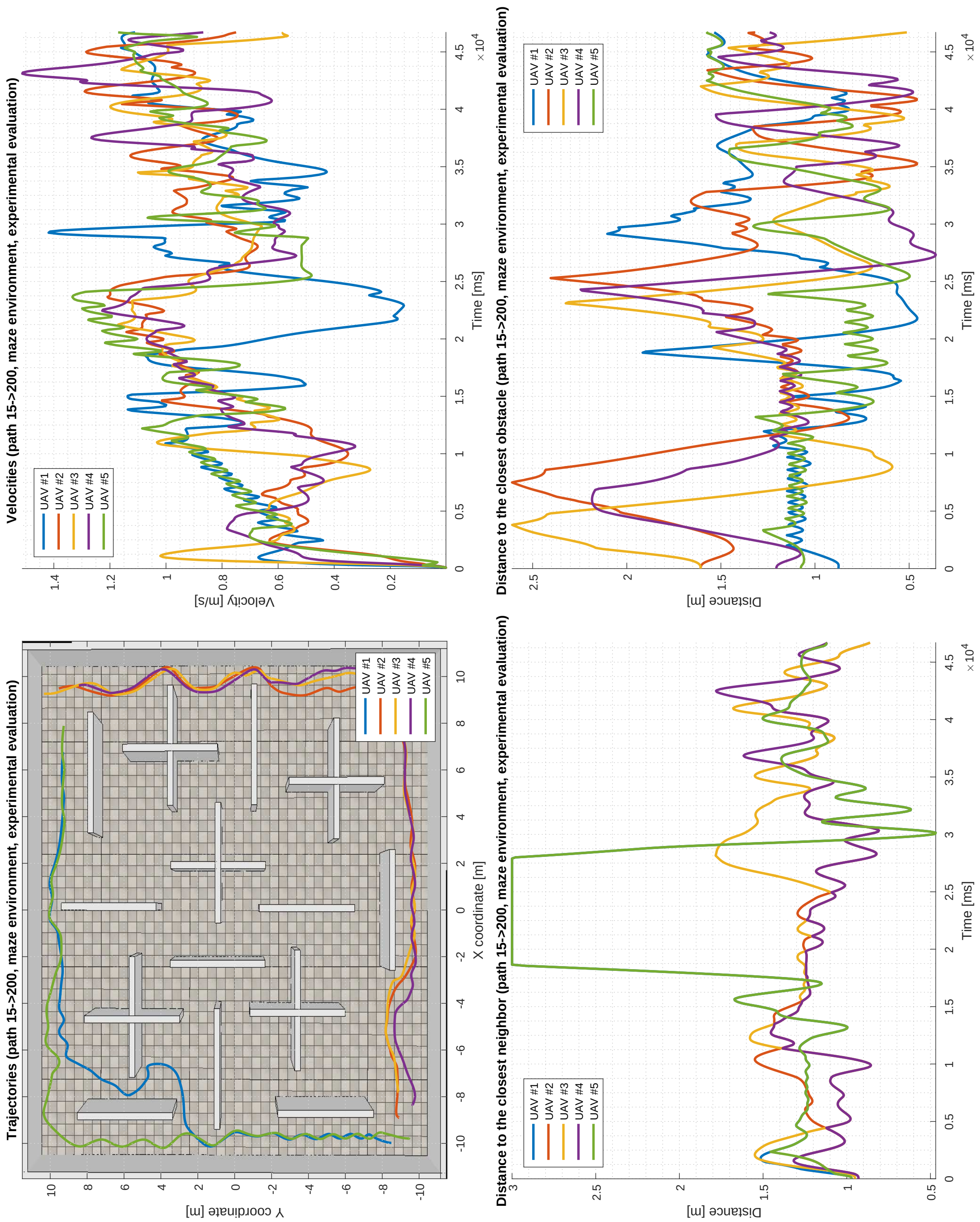

Distance to the closest neighbor (path 152->181, dense environment, heuristic evaluation)
Distance [m]
Time [ms]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

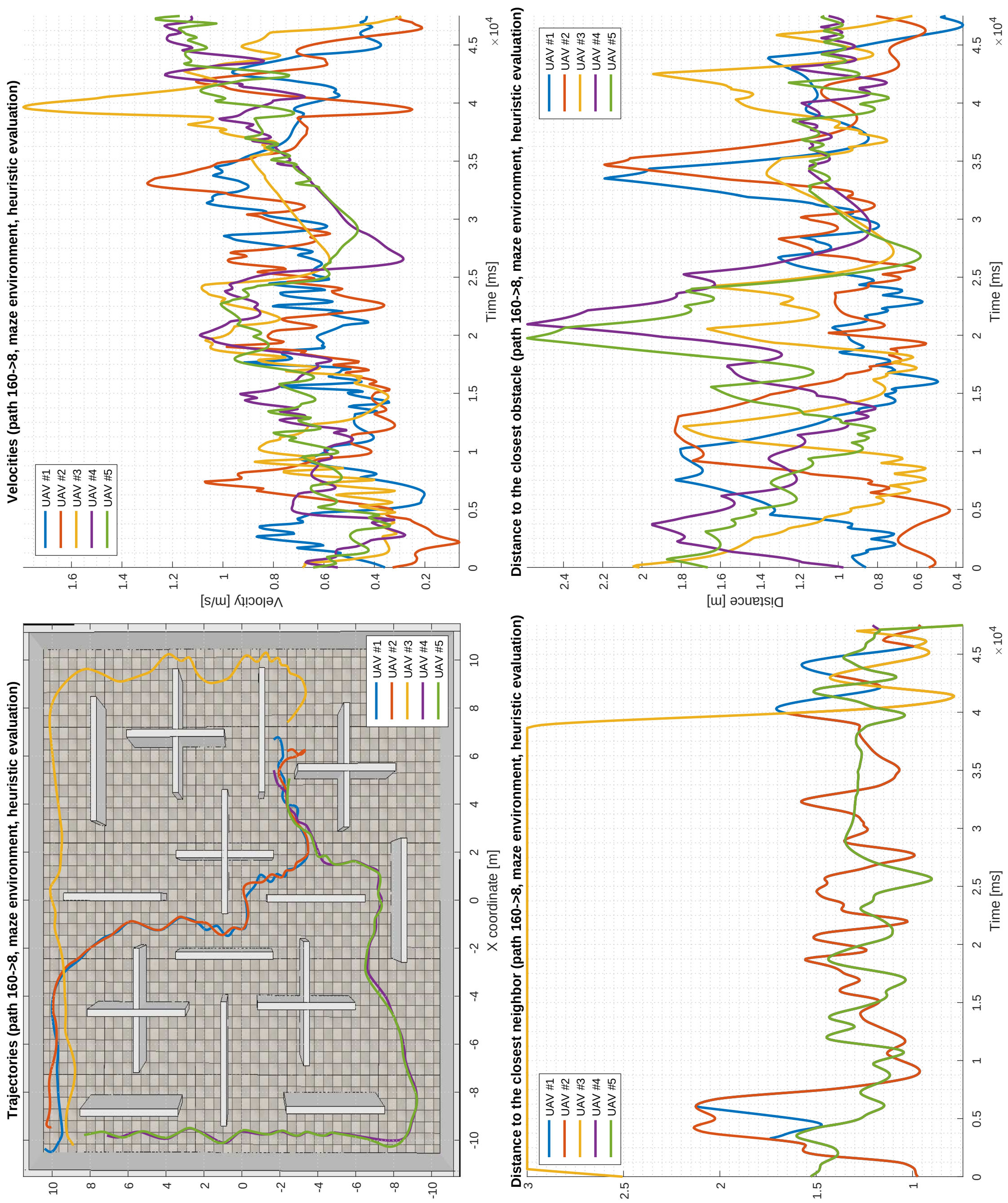

Distance to the closest obstacle (path 152->181, dense environment, heuristic evaluation)
Distance [m]
Time [ms]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5

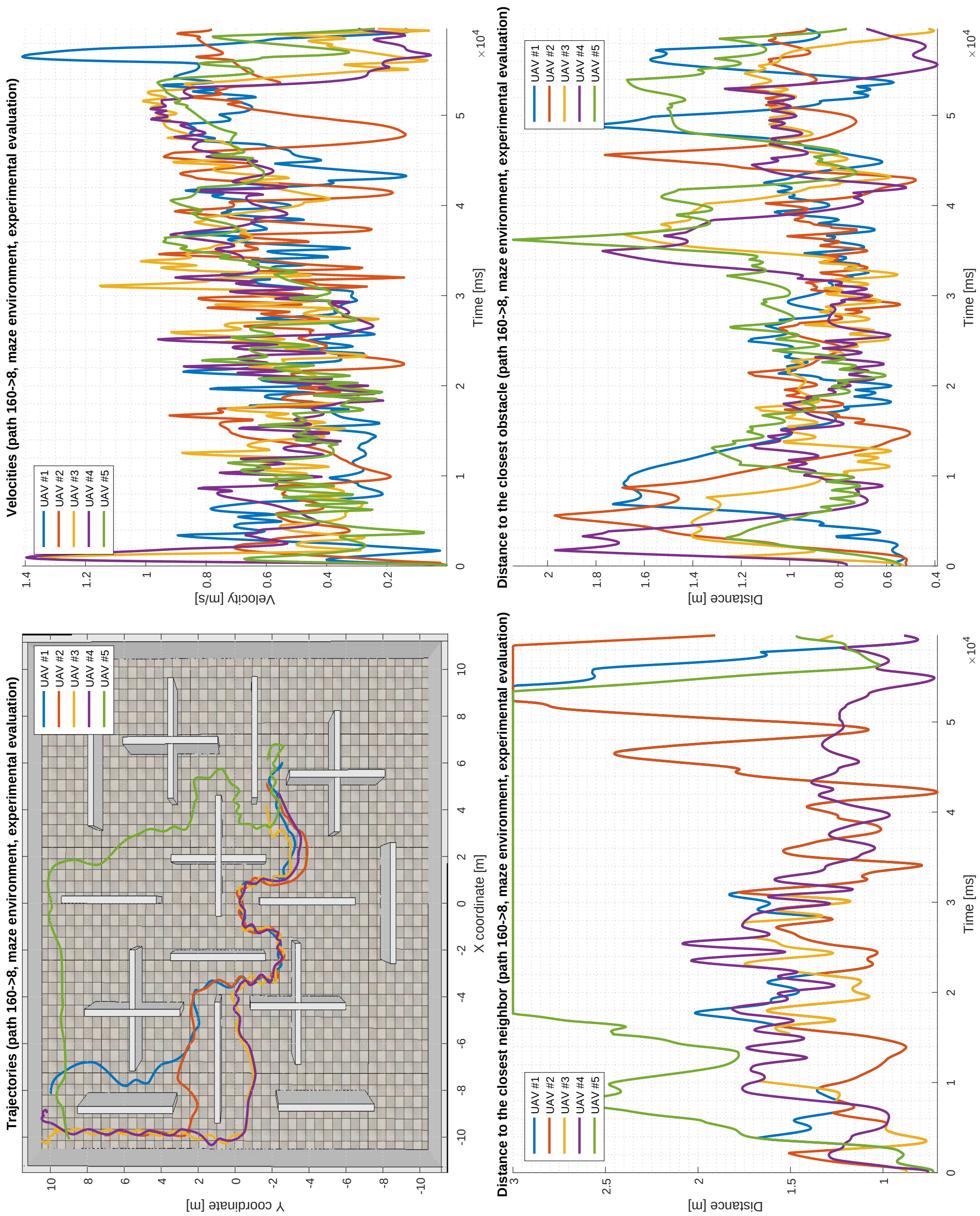


Trajectories (path 152->181, dense environment, experimental evaluation)
X coordinate [m]
Y coordinate [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
Velocities (path 152->181, dense environment, experimental evaluation)
Time [ms]
Velocity [m/s]
Distance to the closest neighbor (path 152->181, dense environment, experimental evaluation)
Distance [m]
Distance to the closest obstacle (path 152->181, dense environment, experimental evaluation)


Trajectories (path 15->200, maze environment, heuristic evaluation)
Y coordinate [m]
X coordinate [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
Velocities (path 15->200, maze environment, heuristic evaluation)
Velocity [m/s]
Time [ms]
×10^4
Distance to the closest neighbor (path 15->200, maze environment, heuristic evaluation)
Distance [m]
Distance to the closest obstacle (path 15->200, maze environment, heuristic evaluation)


Trajectories (path 15->200, maze environment, experimental evaluation)
X coordinate [m]
Y coordinate [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
Velocities (path 15->200, maze environment, experimental evaluation)
Time [ms]
Velocity [m/s]
Distance to the closest neighbor (path 15->200, maze environment, experimental evaluation)
Distance to the closest obstacle (path 15->200, maze environment, experimental evaluation)
Distance [m]


Trajectories (path 160->8, maze environment, heuristic evaluation)
Y coordinate [m]
X coordinate [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
Velocities (path 160->8, maze environment, heuristic evaluation)
Velocity [m/s]
Time [ms]
Distance to the closest neighbor (path 160->8, maze environment, heuristic evaluation)
Distance to the closest obstacle (path 160->8, maze environment, heuristic evaluation)
Distance [m]


Trajectories (path 160->8, maze environment, experimental evaluation)
X coordinate [m]
Y coordinate [m]
UAV #1
UAV #2
UAV #3
UAV #4
UAV #5
Velocities (path 160->8, maze environment, experimental evaluation)
Time [ms]
Velocity [m/s]
Distance to the closest neighbor (path 160->8, maze environment, experimental evaluation)
Time [ms]
Distance [m]
Distance to the closest obstacle (path 160->8, maze environment, experimental evaluation)
Time [ms]
Distance [m]