Skolkovo Institute of Science and Technology
Moscow, Spring 2019

Skoltech
Skolkovo Institute of Science and Technology

MASTER'S THESIS

Adaptive Control of Swarm of Drones for Obstacle Avoidance

Master's Educational Program: Space and Engineering Systems

Student: Agishev Ruslan Timurovich

Supervisor: Tsetserukou Dzmitry Olegovich

Moscow 2019

Acknowledgements

The author expresses sincere gratitude to his supervisor Tsetserukou Dzmitry Olegovich for their invaluable help and provided equipment in carrying out this work.

Skolkovo Institute of Science and Technology

Moscow 2019

Abstract

This paper presents a layered path planner algorithm to solve multiple agents navigation problem in a cluttered environment. The general path planning problem is divided into approximate global trajectory construction, which is further smoothed by a local path planning method.

The proposed approach provides a solution based on a leader-followers architecture with a prescribed formation geometry that adapts dynamically to the environment and avoids collisions. The path generated by the global planner based on rapidly-exploring random tree algorithm is corrected with the artificial potential fields method that ensures robots trajectories to be collision-free, reshaping the geometry of the formation when required by environmental conditions.

The implemented algorithm has been tested on nano-quadrotors. Flight experiments demonstrated the possibility to accurately navigate the formation of drones in a cluttered environment with static and moving obstacles. This navigation strategy for multiple autonomous robots could potentially have an impact on vision inspection and payload transportation tasks.

Due to its mobility and spatial distribution, the swarm of quadrotors could be the first responder for the different kind of emergencies, such as fire, earthquake or flood. To gather the initial information about a suffering area is the crucial point for any rescue team. Monitoring of the progress of disaster recovery is also an important point as far as an emergency is a dynamically changing environment. Navigation of swarm in the city environment, for example with multi-story buildings or even with skyscrapers, could be a challenging task. Maintaining the default geometry of the formation is reasonable in terms of real-life applications when it is important to gather special data evenly or provide communication through the formation.

Keywords – Robot formation path planning, rapidly exploring random trees, artificial potential fields, quadrotors

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background and problem statement

Multirotor aircrafts, particularly quadrotors, have been the most capable vehicles in terms of accessibility, maneuverability, capacity for onboard sensors, and applicability to a breadth of applications. Great research progress has been made across multiple areas, including the control of agile maneuvers; planning and perception in unknown, unstructured environments; and collaboration in multi-agent teams, sparking a surge of industry investment. This type of robots is now widely used in such applications as infrastructure inspection, rapid package delivery, precision agriculture, disaster response, and choreographed performances. The successes in single-robot autonomy have led to growing interest in the development of cooperative multi-robot systems.

1.2 Purpose and motivation

Robotic and autonomous systems (RAS) have received the increasing interests in a variety of applications where harsh environment (e.g. confined space to deploy and access) has been a challenging issue for numerous reasons. Among them, health, safety, and environmental concerns are the key drivers for the deployment of RAS technology. The inspection and monitoring in a harsh environment are critical both for safety and business reasons. With the current need to deploy an engineer into these environments, safety is of utmost importance and as a result, much preparatory work and additional safety assessments must be performed prior to human entry. In addition, RASs have enabled machines with greater levels of flexibility and adaptability, allowing them to perform various tasks more efficiently than the human counterpart. Multiple Autonomous Robots (MARs) (e.g., Unmanned aerial vehicles, UAVs) have emerged as highly agile systems that can be deployed in swarms to perform lightweight tasks quickly and efficiently. With the rising safety, time and cost concerns relating to the inspection of important industrial equipment and infrastructures, the use of small, lightweight UAVs that can be deployed quickly to assess the internal and external conditions are highly desirable.

The main goal of the thesis is the development of the algorithm for drone swarm navigation in complex environments including obstacles. Several existing robot navigation algorithms are explored, tested and compared during the work on the thesis in the lab. Their advantages and limitations are revealed in order to develop robust drone swarm navigation algorithm in a dynamic cluttered environment. The proposed algorithm for the robotic formation navigation should satisfy the set of criterion defining its performance. The first important parameter is safety in terms of robots' trajectories feasibility, in other words, drones dynamics should be taken under consideration during the path planning. In addition, vehicles' routes must prevent collisions between each pair of the drones and drones with static and moving obstacles, while being reasonably short in time to reach final goals. The navigation algorithm was tested in simulation and with an individual robot before switching to the multi-agent case. In order to prevent collisions between drones and obstacles, several concepts could be used, like Save Flight Corridor construction or dangerous regions defined as vicinity with influence radius near the obstacles.

1.3 Literature review

Robot motion planning in dynamic environments is one of the areas of research in computer science and computational geometry. The fundamental problem of motion planning is obtaining a collision-free path from start to goal for a robot that moves in a static and totally known environment that consists of one or many obstacles Latombe (2012). Robot motion planning in dynamic environments (RMPDE) has been studied extensively (Lozano-Perez (1990), Fiorini and Shiller (1995), Canny (1988), Latombe (2012), LaValle (2006)). Motion planning in dynamic environments with moving obstacles and moving targets is another thrust area (Fraichard and Asama (2004), Shiller et al. (2001), Shiller et al. (2010), Petti and Fraichard (2005), van den Berg (2007)). Approaches based on artificial potential fields, velocity, artificial intelligence, and probability appear to be the most active areas of research. Motion planning in dynamic environments is not studied only in robotics but also in Agent systems and Computer Geometry. An important point (Mohanan and Salgoankar (2018)) is that in spite of major advances in the area over the past three decades, not much work has been carried out on multi-robot systems.

The literature on multi-robot planning allows trajectories with instantaneous velocity

changes. Many methods further discretize the workspace into graph-based representations and return paths with arbitrarily sharp turns between vertices. Such trajectories are not dynamically feasible for quadrotors, and extending existing multi-robot planning algorithms to high-order, high-dimensional, underactuated systems is nontrivial.

In some applications, robots must travel in a formation to selected goal positions. Early works incorporated inter-robot distance constraints into a mixed-integer quadratic program for trajectory generation, (Mellinger et al., 2012). This approach allows for the simultaneous optimization of the team's formation and individual robot trajectories. Unfortunately, because this method concatenates all robots' trajectory coefficients into a single decision vector, it is impractical for large teams (more than 20 robots). This problem can be made more computationally tractable by requiring the team to select from a set of predefined formations, (Alonso-Mora et al., 2016).

In other instances, each vehicle must independently navigate to the desired goal position. In unlabeled problems, robots are considered identical; each goal in a set must be reached by some robot, but it does not matter which one. This abstraction can be used in surveillance tasks, where robots must distribute themselves among given locations from which to gather information, (Mohta et al., 2016), (Scaramuzza et al., 2014). Here, the task-allocation and trajectory-generation problems must be simultaneously solved.

In labeled problems, by contrast, robots must visit predefined, noninterchangeable goal positions. This collision-avoidance problem can be solved with reactive approaches, (Alonso-Mora et al., 2015) or optimization based methods, (Tang et al., 2016).

Many open questions remain in this area. Many planning works assume the existence of a central base station with global information, limiting the teams' range of operation and presenting a single point of failure. It is much more beneficial to adopt a decentralized architecture, where robots require only communication with (or sensing of) their most immediate neighbors. Decentralized methods have been successful for small teams of robots operating at low speeds, (Zhou et al., 2017). However, a safe, complete, scalable method for high-speed flight has not yet been found.

2 Quadrotor Trajectory Smoothness Estimation

Prior to dealing with the drone swarm, it is important to take into account individual robot kinematics. In this work, a quadrotor UAV is considered as an agent of the robotic formation. Due to maneuverability and the ability to hover in the air, quadrocopters are used in an urban densely built environment. An important request for the drone's path passing in the space with obstacles is a small deviation of the UAV from the intended trajectory. Computer model of the quadrotor was utilized in order to investigate the accuracy of the commanded trajectories following by the UAV. Evaluated trajectories are generated by polynomials of the time variable, and differ in degrees of smoothness.

From the equations of motion of the quadcopter, it can be obtained that the control signals (thrust forces and generated moments) depend on the snap, Mellinger and Kumar (2011), the 4-th time derivative of the position, $\vec{r}^{(4)}(t)$). Therefore, to construct smooth curves passing through a set of given points, a variational method is used to minimize the fourth time derivative of the coordinate. Thus, in order to find a curve connecting two adjacent control points of the trajectory, a mathematical problem is solved:

$$\vec{p}^{*}(t) = \arg\min_{\vec{p}(t)} \int_0^T \left\| \vec{p}^{(4)} \right\|^2 dt \tag{2.1}$$

Here $\vec{p}^{*}(t)$ is the desired trajectory between two subsequent way-points, T is the time spent on its passage to the next control point. Solution of the obtained variational problem, Eq. 2.1, is equivalent to dealing with the Euler-Lagrange differential equation:

$$\frac{d^{(4)}}{dt} \frac{\partial L}{\partial \vec{p}^{(4)}} = 0, \quad \text{where} \quad L = \vec{p}^{(4)}, \tag{2.2}$$

$$\text{or:} \quad \vec{p}^{(8)} = 0. \tag{2.3}$$

Therefore, the desired trajectory must be specified as a polynomial of the 7-th power of

time, for example:

$$p_i(t) = \alpha_{i0} + \alpha_{i1}\frac{t - S_{i-1}}{T_i} + ... + \alpha_{i7}(\frac{t - S_{i-1}}{T_i})^7. \tag{2.4}$$

Here the following notations are introduced: $S_0 = 0$, $S_i = \sum_{k=1}^{i} T_k$ is the time to reach the way-point number i moving from the initial location, $p = x, y, z$. All the polynomials, $p_i(t)$, $i = 1..n$ (n - the amount of way-points, w_i, to pass through), described above must satisfy to $8n$ conditions in order to find all the parameters α_{ij}, $i = 1..n$, $j = 0..7$, as follows:

$$p_i(S_{i-1}) = w_{i-1}, \quad p_i(S_i) = w_i, \quad i = 1..n \tag{2.5}$$

$$p_i^{(k)}(S_0) = p_n^{(k)}(S_n) = 0, \quad k = 1..3 \tag{2.6}$$

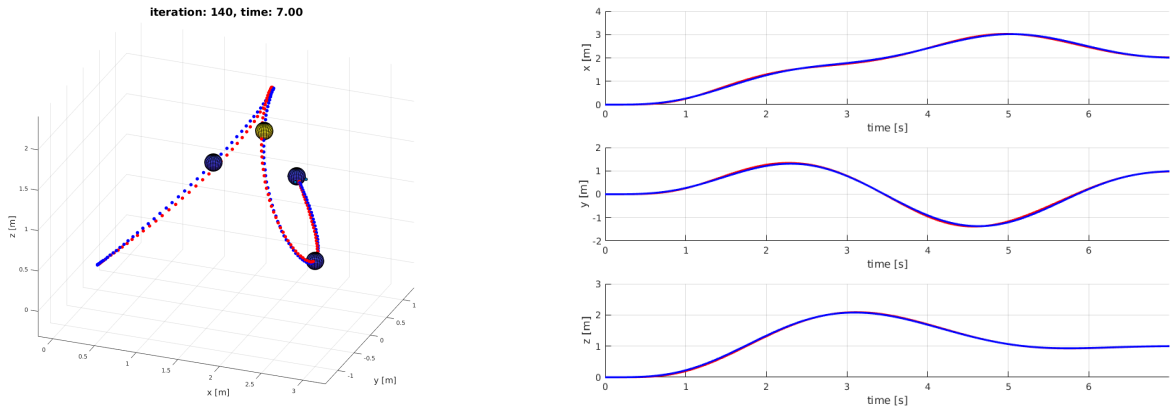
$$p_i^{(k)}(S_i) = p_{i+1}^{(k)}(S_i), \quad k = 1..6 \tag{2.7}$$

Eq. 2.5 ensures that the polynomials are moving through the way-points. The second condition, Eq. 2.6, states that in initial and final locations the values of velocity, acceleration and jerk of the quadrotor are zero, while the third equation is introduce to zip the adjacent polynomials in smooth way.

The system of equations (2.5, 2.6, 2.7) for unknowns α_{ij} is written further in matrix form:

$$A\alpha = b \tag{2.8}$$

In this equation, A is a matrix of dimensions $8n \times 8n$, α is a column of desired parameters, b is an $8n \times 3$-matrix. Solving the system of equations separately for each of the columns of the matrix b, we obtain the coefficients defining the trajectories $x(t), y(t), z(t)$, respectively. In such a way, a "minimum snap" trajectory moving through a set of way-points, $w_i, i = 1..n$, is obtained, Figure 2.1. In this example, the drone average velocity was set to 1 m/s, Figure 2.2.



(a) "Minimum snap" trajectory. Red dots - real, blue - commanded paths.

(b) Components of the resultant trajectory along spacial axes.

Figure 2.1: "Minimum snap" trajectory of the quadrotor, generated by a set of way-points, $[0, 0, 0], [1, 1, 1], [2, 0, 2], [3, -1, 1], [2, 1, 1]$.

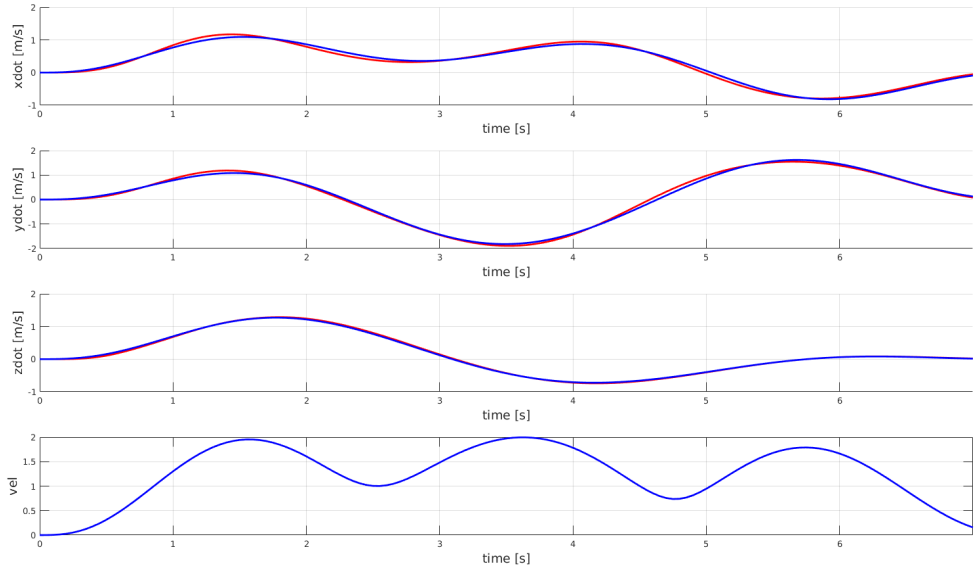


Figure 2.2: Velocity of the drone, following "minimum snap" trajectory through the set of way-points, $[0, 0, 0], [1, 1, 1], [2, 0, 2], [3, -1, 1], [2, 1, 1]$.

Let us further carry out a comparative analysis of the obtained "minimum snap" trajectory with a number of other spatial curves describing the motion of the quadcopter. The motion of the UAV was also modeled through a series of control points connected by polynomials of 3-rd ("minimum acceleration") and 5-th ("minimum jerk" trajectory) degrees. We will consider also a trajectory which consists of the straight lines between the way-points, "Line" in Table 2.1. The main metric of the experiment is the deviation

of the quadrotor trajectory from the commanded (desired) path:

$$\|\vec{p}_{des} - \vec{p}\|_{L_2} = \sqrt{\sum_i (x_{des,i} - x_i)^2 + (y_{des,i} - y_i)^2 + (z_{des,i} - z_i)^2}. \tag{2.9}$$

The desired trajectory in this experiment consists of 4, 5, 6 or 7 subsequent way-points from a set (starting from $[0, 0, 0]$): $[0, 0, 0], [1, 1, 1], [2, 0, 2], [3, -1, 1], [4, -2, 2], [5, -1, 3], [6, 0, 0]$.

Table 2.1: Deviation errors for 4 types of the trajectories moving through the same set of N way-points

	Number of way-points, N			
Trajectories	4	5	6	7
Min acc	1.76 m	2.08 m	2.05 m	2.14 m
Min jerk	1.39 m	1.70 m	1.73 m	1.83 m
Min snap	1.26 m	1.52 m	1.58 m	1.73 m
Line	7.48 m	8.96 m	9.97 m	10.59 m

Based on the data presented in Table 2.1, it is obvious, that smooth trajectories are much more preferable than a path composed of straight lines for the task of following the trajectory given by control points. The "minimum snap" trajectory demonstrated the highest performance in terms of the smallest trajectory deviation error. The conducted experiment proves, that it is important to construct quadrotor trajectories with low acceleration, jerk and snap values. In this way, the actually executed drone trajectory has low deviation error from the commanded path.

3 Global Trajectory Planning

The aim of this section is to give an overview of the path planning problem and to describe the initial path construction for a single robot. Due to its ability to quickly explore the space and deal with obstacles of complex shape, the first approximation of the path through the static map is provided by the Rapidly-exploring Random Tree (RRT) algorithm, Lavalle (1998). This method is widely used in robotics and has numerous extensions. The grown tree of configurations is able not only to cover the entire state space but also to take into account nonholonomic constraints of the robots (Palmieri and Arras (2014)), planning smooth trajectories. Moreover, the RRT-based algorithms successfully deal with environments of different structure (Brunner et al. (2013)) and complexity (Noreen et al. (2016)). Multiple trees are grown to connect a set of way-points that a robot must visit in a certain order in order to complete a task-planning problem, Wong et al. (2018). It is also important to mention, that due to the random nature of the RRT-based methods, the retrieved trajectories oftentimes have lots of turns, and, as a result, they are not the shortest possible solution to reach the goal. That is why path shortening (Yang (2011)) algorithms are utilized further to process the resultant trajectories.

The multi-agent case is the main focus of this thesis. In order to guide several robots, leader-follower architecture is considered in the present paper. In this case, it is important to provide a collision-free path for a leader-robot, while the followers' trajectories are defined relative to the main agent position. In such a way, the robotic formation is considered as a united object, which parameters are defined by the composing robots states. In this section, we consider further the trajectory construction process for the formation leader.

A path planning problem is generally viewed as a search in a metric space X, for a continuous path from an initial state, s_{init} to a goal region $X_{goal} \subset X$ or goal state s_{goal}. The term state space is used in general path planning problems. For a standard problem $X = C$, which is the configuration space of a rigid body or system of bodies in a 2D- or 3D- world.

It is assumed that a fixed obstacle region, $X_{obs} \subset X$ must be avoided, and that an explicit

representation of X_{obs} is not available. One can only check whether a given state lies in X_{obs}. States in X_{obs} could correspond to velocity bounds, configurations at which a robot is in collision with an obstacle in the world, or several other interpretations, depending on the application. A Rapidly-exploring Random Tree (RRT) is constructed so that all of its vertices are states in X_{free}, the complement of X_{obs}. Furthermore, each edge of the RRT will correspond to a path that lies entirely in X_{free}, Lavalle (1998). The main idea of the method is to grow a tree of possible robot's configurations from its initial state, expanding in all directions, as described in the algorithm below, Algorithm 1.

while s_{goal} *is not reached* **do**

1. Initialize tree with first node s_{init}.
2. Pick a random target location (every 100^{th} iteration, choose s_{goal}).
3. Find the closest vertex in the roadmap.
4. Extend this vertex towards the target location.

end

Algorithm 1: RRT algorithm

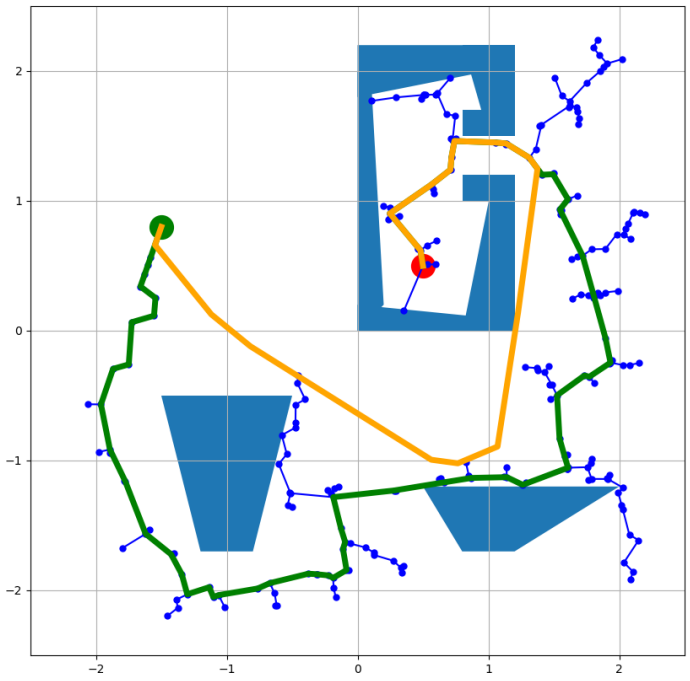


Figure 3.1: RRT construction on a 2D configuration space from the initial state (red circle) to the goal (green circle). The green curve represents a path obtained from the built tree. The shortened path is depicted as an orange line.

Figure 3.1 represents an example of the RRT construction in 2-dimensional configuration space, where the state of the robot is defined as its position, $s = [x, y]^T$. Random choice of the branches to be extended allows the tree to find a path even if the obstacles on the map have a complex non-convex shape, like a bug trap in this case.

Ones the tree is constructed (the goal is added to the tree), the next step is to retrieve a path out of it. This could be done with the help of the hierarchical structure of the RRT, where each vertex knows its parent. First of all, the path is initialized with a goal position. After this, on every iteration, the parent of the last vertex in the constructed trajectory is appended to the end of the path. The process continues until the start position is reached. The implementation of the algorithm could be found in section A1.

In such a way we obtained the green path. Out of it, we can obtain further a shortened orange path. That is done iteratively, checking if it is possible to connect intermediate points of the green trajectory without collisions with obstacles, 3.2. There are different

more intelligent path shortening and smoothing methods exist, for example, it is important to construct a path at a reasonable distance from obstacles, Wong et al. (2018). However, as it will be discussed further, the obtained global path serves only as an initial approximation, which will be corrected afterwards.

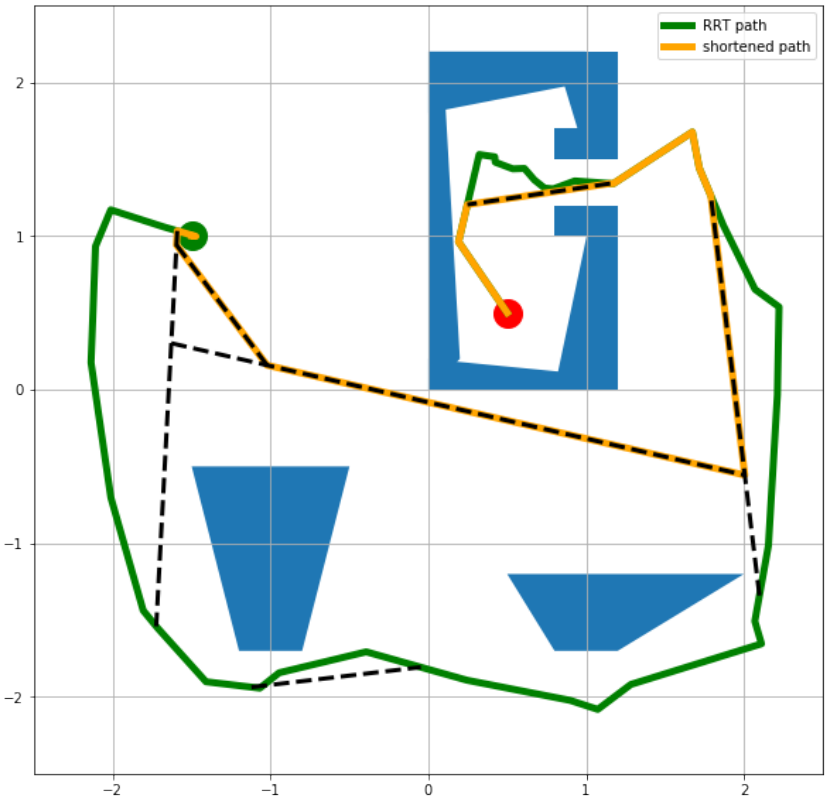
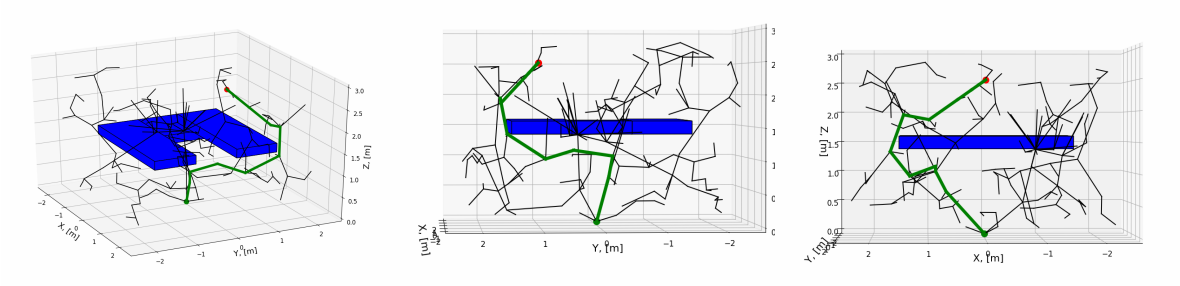


Figure 3.2: Path shortening. The green curve represents trajectory, extracted from the RRT, while the orange line depicts its shortened version. Black dashed line corresponds to one successful iteration of the path shortening algorithm.

Obtained orange trajectory is utilized further as a global planner path. RRTs are efficient in many practical problems, due to they do not require to store the whole map of the environment. In order to construct a collision-free trajectory with the help of the RRTs, it is enough to answer a question if a particular robot's configuration is in collision or not. The algorithm and its modifications are often utilized for the robots with many degrees of freedom when the state of the robot is described with a long vector (7 DOF robot arm), Lee et al. (2015).

In this work, the main goal is to construct a 3-dimensional collision-free trajectory. However, the orientation of the robot is not taken into account. That is why, the configuration space

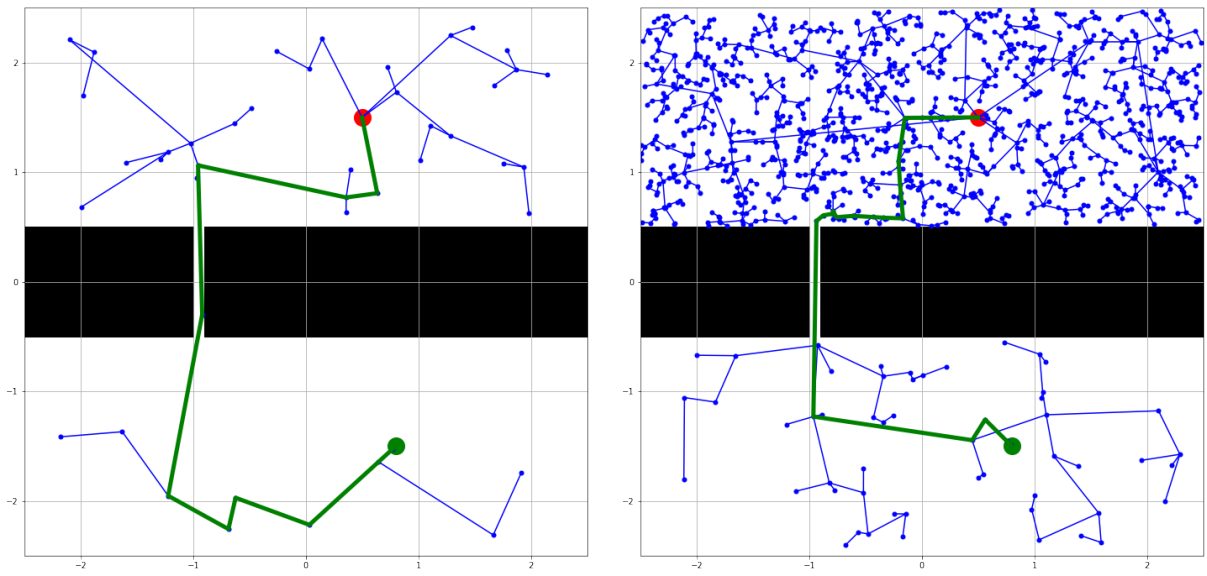
is considered to have 2 or 3 dimensions (if the robot avoids obstacles only in XY-plane or XYZ-space respectively). An example of RRT construction in 3D is given in Figure 3.3.



(a) RRT in 3D-configuration space **(b)** Side view (from X-axis). **(c)** Side view (from Y-axis).

Figure 3.3: Path construction in 3D-configuration space with the help of RRT. Blue boxes represent obstacles, the green line is a path obtained after the RRT (black lines) is expanded from start (red circle) to goal (green circle).

The algorithm also possesses a strong probabilistically completeness property. This means, given enough time it finds a solution if one exists. However, there is no guarantee on the exact time the method would take to construct a path, due to the random nature of the algorithm. For example, the algorithm performance may differ dramatically in terms of time to connect start and goal positions even on two trials with the same conditions (obstacles map), like in this example, Figure 3.4.



(a) Time to construct a path: 0.36 sec **(b)** Time to construct a path: 7.5 sec

Figure 3.4: RRT algorithm performance at the same obstacle map (narrow passage) for 2 different trials. Experiments were running on an ordinary PC with 2.4 GHz Intel Core i5 and 8 GB of RAM.

4 Local Trajectory Planning

Ones global path is constructed, another task is to avoid collisions between robots and other moving obstacles in the environment. This problem is solved by a local trajectory planner. An accurate local planner accounts for the constraints and generates feasible local trajectories in real-time. The robots with the help of local planner should not only execute smooth feasible trajectories but also be able to avoid moving obstacles.

We assume that, within the swarm, every agent makes a decision on where to go next using both the local information about surroundings and the global goal (direction and velocity of motion). In such a scenario, each quadrotor could plan its obstacle avoidance considering the position of nearest obstacles and neighbor agents. The location of drones and obstacles is defined by the motion capture system. Each quadrotor is aware of the position of local obstacles. The two local planners are considered in this work. The first one is based on the inclusion of springs and dampers to create virtual forces between robots and obstacles. The main idea of the other one is the usage of artificial potential fields (APF) to influence the location of each robot during movement in the formation. We consider the two algorithms in the following subsections, starting from the impedance control-based method, which utilizes mass-spring-damper interlinks between the interacting agents on the map (robots and obstacles). The final part of this section is devoted to the application of APF-based algorithm for a single- and multi-robot cases.

4.1 Obstacle Avoidance with Impedance Control

In this scenario, obstacles affect drones default locations inside the formation with the position-based impedance control, (Tsetserukou et al.). During the formation of drones navigation in space, impedance model updates the goal positions for each flying robot, which changes default drone-to-drone distances. In order to calculate the impedance correction term for the robot's goal positions, we have to solve a second-order differential equation (4.1) that represents the impedance model. In order to find this term, the following differential equation should be solved for each dimensional axis, along which the

correction is going to be performed:

$$M_d\Delta\ddot{x} + B_d\Delta\dot{x} + K_d\Delta x = F_{ext}(t) \tag{4.1}$$

where M_d is the desired mass of a virtual body, D_d is the desired damping, and K_d is the desired stiffness, Δx is the difference between the current x^c_{imp} and desired x^d_{imp} position, and $F_{ext}(t)$ is an external force, applied to the mass. It is well known that by selecting the desired dynamics parameters for the impedance model, we can get various behavior of the oscillator, described by (4.1), undamped, underdamped, critically damped, and overdamped. State space representation of the (4.1) has the form:

$$\begin{bmatrix}\Delta\dot{x}\\ \Delta\ddot{x}\end{bmatrix} = A\begin{bmatrix}\Delta x\\ \Delta\dot{x}\end{bmatrix} + BF_{ext}(t), \tag{4.2}$$

where $A = \begin{bmatrix}0 & 1\\ -\frac{K_d}{M_d} & -\frac{D_d}{M_D}\end{bmatrix}$, $B = \begin{bmatrix}0\\ \frac{1}{M_d}\end{bmatrix}$. In discrete time-space, after integration, we could write the impedance equation in the following way:

$$\begin{bmatrix}\Delta x_{k+1}\\ \Delta\dot{x}_{k+1}\end{bmatrix} = A_d\begin{bmatrix}\Delta x_k\\ \Delta\dot{x}_k\end{bmatrix} + B_dF^k_{ext} \tag{4.3}$$

where $A_d = e^{AT}$, $B = (e^{AT} - I)A^{-1}B$, T is the sampling time, I is the identity matrix, and e^{AT} is the state transition matrix. Impedance model, as the second order differential equation, could be classified by the shape of the step response. The most challenging part in here is to compute the term e^{AT}. Matrix exponential is refined form Cayley-Hamilton theorem, according to which every matrix satisfies its characteristic polynomial. Using this statement, we can find:

$$A_d = e^{\lambda T}\begin{bmatrix}(1-\lambda T) & T\\ -bT & (1-\lambda T - aT)\end{bmatrix}, \tag{4.4}$$

$$B_d = -\frac{c}{b}\begin{bmatrix}e^{\lambda T}(1-\lambda T) - 1\\ -bTe^{\lambda T}\end{bmatrix}, \tag{4.5}$$

where λ is eigenvalue variable of the matrix A, $a = -\frac{D_d}{M_d}$, $b = -\frac{K_d}{M_d}$, $c = \frac{1}{M_d}$. A_d and B_d matrices could be used for calculating the current x^c_{imp} position of the impedance model using equation (4.3).

The method described above is used to calculate the impedance correction vector $\left[x_{imp}, y_{imp}, z_{imp}\right]^T$ or the current position of the virtual body of each impedance model. In order to demonstrate the performance under the assumption on the boundedness of the external inputs, the impedance terms are limited with the maximum values:

$$\begin{bmatrix} x_{imp} \\ y_{imp} \\ z_{imp} \end{bmatrix} \leq \begin{bmatrix} x_{imp\_limit} \\ y_{imp\_limit} \\ z_{imp\_limit} \end{bmatrix}, \tag{4.6}$$

where the right part represents the safety threshold that prevents an overrun of the impedance model, Tsykunov et al. (2018).

In this work, a local trajectory planner which changes a robot position in the XY-plane is considered. This limitation might be reasonable in case if the flying robots do not change their altitude above the ground. The interaction forces between drones and obstacles are described further. Each obstacle has a safety zone around its center, which is defined as a cylinder (a circle for planar motion) with radius R, (Figure 4.1), $R > R_{obstacle} + R_{drone}$ (here $R_{obstacle}$ and R_{drone} define the maximum radius of circumcircles around considered obstacle and drone). For example, we assume that a quadrotor default trajectory goes from point A to point E as a straight line, see Figure 4.1 for reference, crossing the safety region of the obstacle. In that case, the goal trajectory of the quadrotor is corrected, relatively to the obstacle pose. The new trajectory will go through the points $A - B - C - D - E$, following the circumference, defining the obstacle's safety zone boundary.

However, the defined obstacle avoidance method requires drones to move through the arc faster in comparison with a straight line trajectory. In addition, the quadrotors should perform sharp maneuvers being in the points B and D.

In order to mitigate these challenges, two impedance models are proposed in this work. The goal of the one of them is to create a virtual viscosity force, which should produce a torque braking the robot movement along the arc of the obstacle safety zone. In this way drones movement along the boundary of the obstacles will not be much faster, than

along a straight line going through the safety area. The other impedance model is aimed at smoothing the trajectories in the points B and D (Figure 4.1).

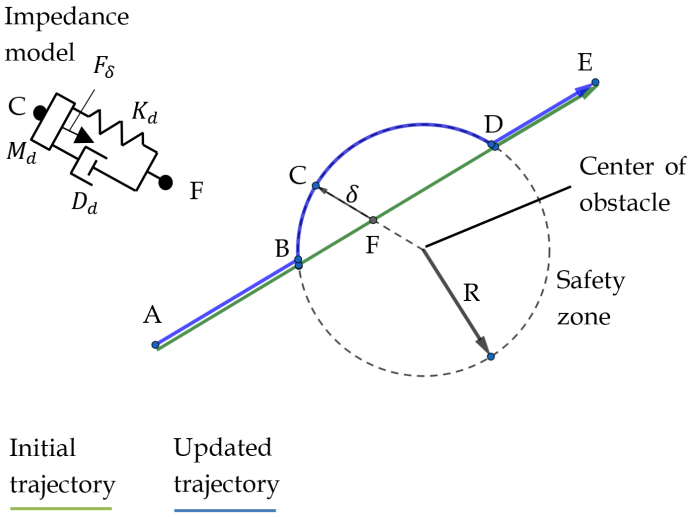


Figure 4.1: Obstacle avoidance by a quadrotor.

4.1.1 Angular Impedance Control

The angular impedance model presented (see Figure 4.2) in this chapter, is introduced to slow down the drone movement along the circumference, defining an obstacle safety zone. Here we consider a robot position in polar coordinates. The origin of the system is defined at the obstacle center, point O in Figure 4.2. It is also considered that the robot position (point P) belongs to the safety zone circumference. In order to control drones angular velocities near obstacles circumferences, the following angular impedance model is implemented.

$$J\ddot{\theta} + D\dot{\theta} = M(\Delta\theta) \tag{4.7}$$

where $M(\Delta\theta) = M(\theta - \theta_{sp})$ is a torque acting on the robot near the obstacle, J is the desired inertia moment of the model, and D is desired damping coefficient. The impedance model choice is explained by damped convergence of the set-point of the drone to desired position (R, θ_{imp}), expressed in polar coordinates relatively to the obstacle.

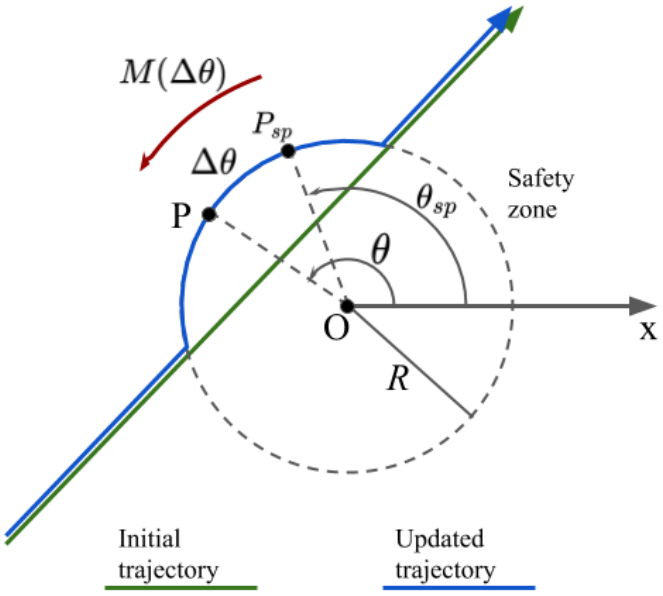


Figure 4.2: Angular impedance control-based obstacle avoidance scheme. P - current drone position, P_{sp} - commanded drone position without angular impedance model correction. $M(\Delta\theta)$ - torque produced by the impedance control strategy.

The torque was developed to be a function of the difference between the current angular position of the drone, θ, (affected by the impedance model) and θ_{sp} - commanded angular position if there is no any braking torque. It is designed as follows: $M(\Delta\theta) = -K_\theta \sin(\theta - \theta_{sp})$. Here $K_\theta > 0$ is a scaling coefficient, controlling the magnitude of the applied virtual torque. The choice of the sin-function is motivated by the smoothness and 2π - periodicity requirements. The last property is important due to the torque $M(\Delta\theta)$ value should be equal for $\Delta\theta = 0$ and $\Delta\theta = 2\pi$. The output of the differential equation, $\theta = \theta_{imp}$, defines resultant angular position of the robot on the safety zone circumference with radius R.

4.1.2 Radial Impedance Control

In order to make trajectories near obstacles more feasible (especially in points B and D in Figure 4.1) and smoother, we propose to use position-based impedance control. We introduce a mass-spring-damper model between points C (current position of the drone) and F (Figure 4.1), which is defined as:

$$M_d \Delta\ddot{x} + D_d \Delta\dot{x} + K_d \Delta x = F_\delta(t). \tag{4.8}$$

Where M_d, D_d, and K_d are desired dynamic coefficients. $F_\delta(t)$ is the virtual force defined as $F_\delta = K_\delta \delta$. K_δ is the scaling coefficient and $\delta = |CF|$ is the distance between points C and F, i.e. the desired jump into the safety zone.

In order to solve (4.8), the same numerical method (4.3), could be used. Virtual force $F_\delta(t)$ affects a drone position only in the obstacle's safety zone. Input parameters of the impedance model are defined using the state of the drone, located in this area. For example, when the drone crosses the safety zone for the first time (it is located in the point B), $\Delta x_0 = 0, \Delta \dot{x}_0 = v_{CF}$, initial impedance velocity is defined as a projection of the drone velocity to the CF line. It is also important to prevent a drone to fly too close to an obstacle, that is why the desired jump to the safety zone, defined by radius R, should be limited:

$$\delta \leq R_{drone} \tag{4.9}$$

Obtained impedance correction term, δ, is used further to update the drone position, while avoiding the obstacle (Figure 4.3). As a result, the arc curvature is decreasing due to the dynamics of the impedance model. Therefore, the trajectory of the drone in the vicinity of the obstacle becomes smoother.

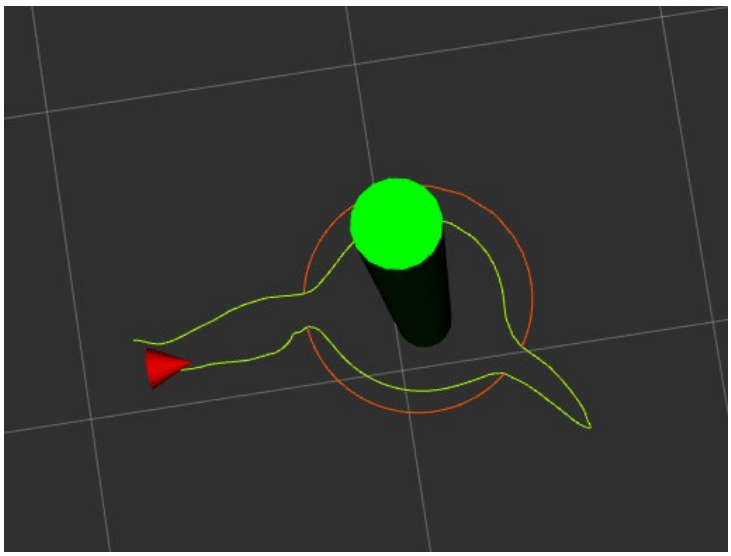


Figure 4.3: Obstacle avoidance based on impedance control. Red circumference defines the safety zone of the obstacle (green cylinder). Drone's position is depicted as a red triangle. Its trajectory (green path) is affected in the obstacle vicinity by virtual impedance force, acting along the safety zone radius.

Both described impedance control-based model could be utilized simultaneously, due to the

virtual forces that they produce are perpendicular to each other (the angular impedance model force is anti-collinear to the robot velocity during the movement along the arc and has tangent direction, while F_δ is directed to the obstacle center). These models complement each other solving two different tasks. They allow a drone not only to move along obstacle's safety zone without big acceleration (with respect to the straight line propagation), but also avoid sharp maneuvers entering and exiting the obstacle vicinity.

However, the presented method based on impedance control doesn't scale well for the multi-robot case and for the dense placement of obstacles. The main problem of this algorithm arises in situations when the safety zones of obstacles intersect. This is because the algorithm assumes that the robot is affected by the influence of one safety zone at any particular time, and it is not clear, how to avoid another obstacle (or neighbouring drone) located in this safety flight zone.

The other method considered in the following subsection does not have these drawbacks, due to it allows calculating the resultant force from all neighbouring obstacles for each robot in the formation. It is important, that this resultant force should not only repel the robots from obstacles but also to attract them to the goal as well.

4.2 Artificial Potential Fields

Another local path planner implemented and tested in this work utilizes artificial potential fields method to correct drones positions. Every controlled robot in the swarm should not only be aware of static obstacles on the map but also take into account moving obstacles, which could be humans and other agents in the formation. This collision avoidance method ensures safe real-time robots swarm navigation in a dynamic environment. We first consider the obstacle avoidance problem of a single robot in a plane. Figure 4.4 shows a 2-dimensional Cartesian space with obstacles. The environment is represented as a grid, each cell of which could be free or occupied (it is assigned with 0 or 1 values respectively).

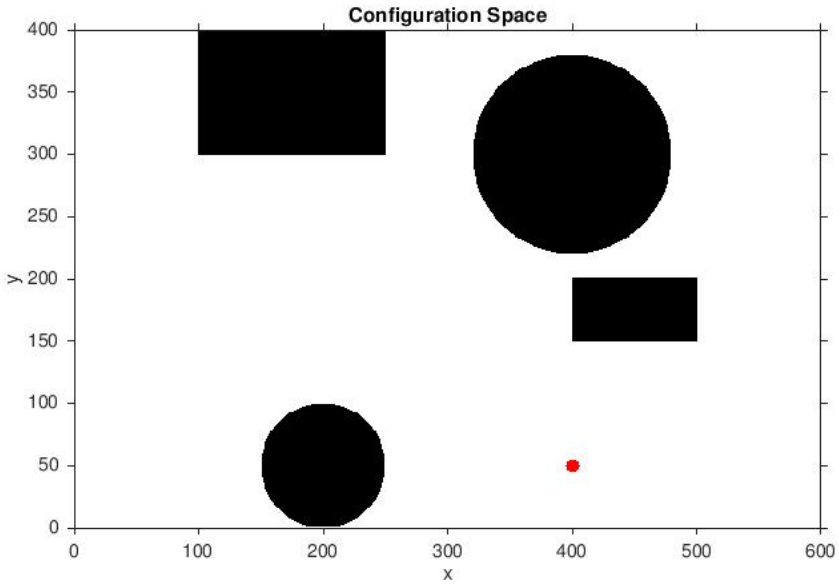


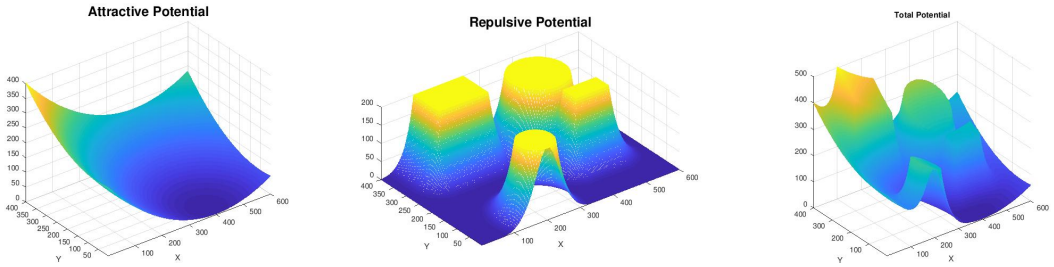
Figure 4.4: Operational environment 400 × 600 grid representation. Obstacles are depicted the in black, collision-free area is drawn in white. Robot's desired destination is a red dot on the grid.

The basic idea of potential fields-based obstacle avoidance algorithm is to construct a smooth function over the extent of robot's configuration space which has high values when the robot is near to an obstacle and lower values when it is further away. This function should have its the lowest value at the desired goal location and its value should increase while moving to configurations that are further away. If such a function is constructed, its gradient can be used to guide the robot to the desired configuration, Khatib (1986). Typically this function consists of two components, attractive and repelling.

An attractive potential function, $U_a(x, y)$, can be constructed by considering the distance between the current position of the robot, $\boldsymbol{p} = [x, y]^T$, and the desired goal location, $\boldsymbol{p}_g = [x_g, y_g]^T$, as follows:

$$U_a(x, y) = \xi ||\boldsymbol{p} - \boldsymbol{p}_g||^2 \tag{4.10}$$

Here ξ is a constant scaling parameter. Artificial potentials can affect a robot's motion in X- and Y- directions. For 2-dimensional map potential functions could be visualized as surfaces, Figure 4.5.



(a) $U_a(x, y)$ **(b)** $U_r(x, y)$ **(c)** $U(x, y) = U_a + U_r$

Figure 4.5: Artificial Potential Functions

A repulsive potential function in the plane, $U_r(x, y)$, can be constructed based on the distance, $\rho(x, y)$, to the closest obstacle from a given point, $[x, y]$, in configuration space.

$$U_r(x,y) = \begin{cases} \eta(\frac{1}{\rho(x,y)} - \frac{1}{d0})^2 & \text{if} \quad \rho(x,y) < d0 \\ 0 & \text{if} \quad \rho(x,y) \geq d0 \end{cases} \tag{4.11}$$

Here η is simply a constant scaling parameter and $d0$ is a parameter that controls the influence of the repulsive potential.

Once the combined potential, $U(x, y) = U_a(x, y) + U_r(x, y)$ is constructed, Figure 4.5(c), a robot's desired velocity can be estimated as $\boldsymbol{v} \propto -\nabla U(x, y)$. Figure 4.6 represents the gradients plot, where each arrow defines local movement direction and velocity magnitude. In such a way, obstacles on the known map correct trajectories of the approaching robots, Figure 4.7.

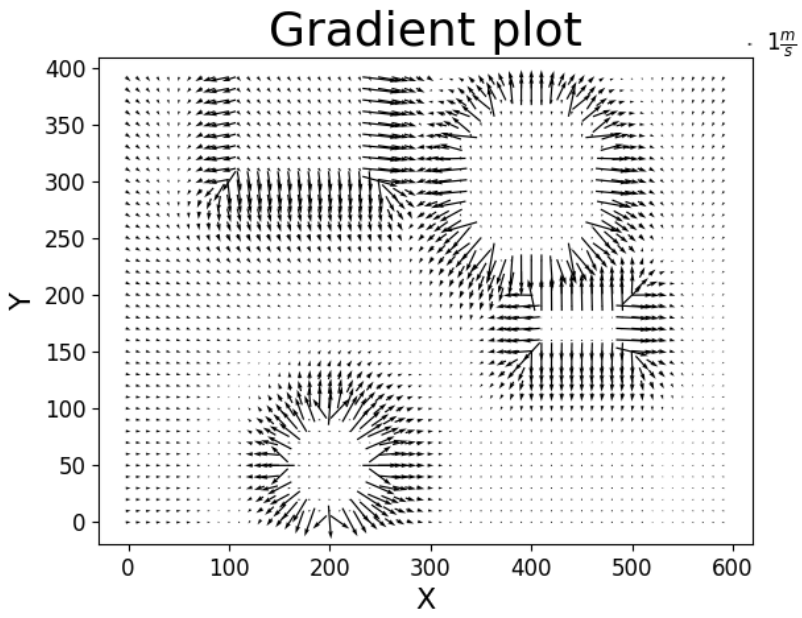


Figure 4.6: Gradient of the combined potential function.

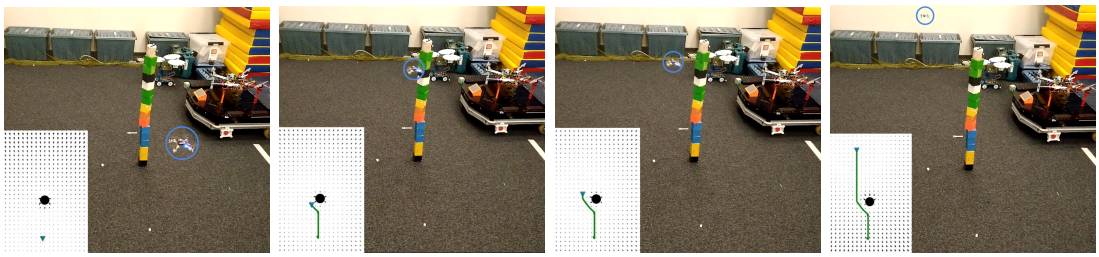


Figure 4.7: Collision avoidance with a single static obstacle.

This method could also be expanded for the swarm of drones navigation, defining interaction forces (which are proportional to velocities in the considered example) between all the agents. The algorithm tracks static as well as dynamic obstacles. Local motion planner based on artificial potential fields allows formation agents positions correction preventing collisions. In the multi-robot case, a point of attraction, x_g^d, (goal location) for every drone, d, is defined relative to the leader-drone position with prescribed geometrical formation shape.

Each robot and obstacle on the known map possesses its own local potential which contributes to the global field. These artificial potentials define interaction forces between neighboring robots and obstacles. Figure 4.8 represents these forces inside the formation of four drones, depicted as connected circles. This gradient plot is visualized for the robot depicted with the green circle. It recognizes other drones in the swarm as obstacles, while

its desired position (one of the vertices of the rhomboid formation) is an attractive point for itself.

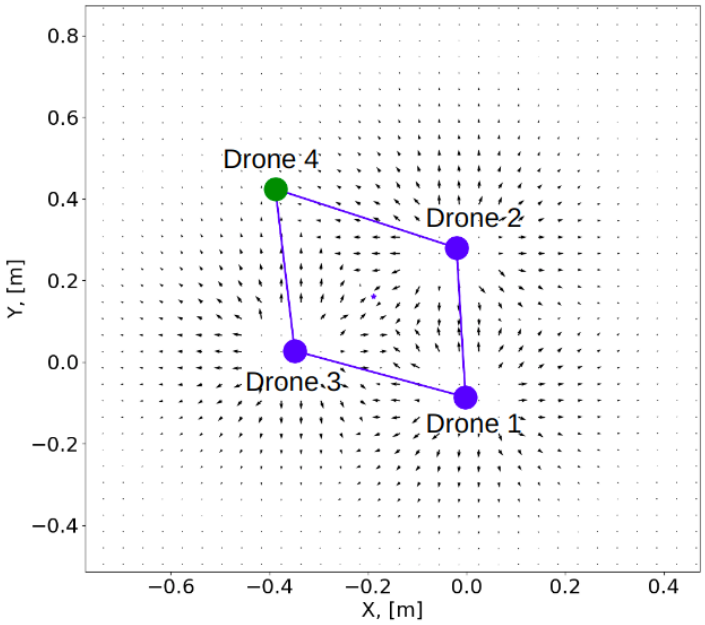


Figure 4.8: Inter-agent collision avoidance strategy, based on the potential-field method. Quiver plot represents, how the green robot recognizes the other 3 drones (depicted in blue color) in the gradient map.

In such a way drone swarm is capable to adapt its shape according to local obstacles positions.

Due to the interactive nature of the potential field obstacle avoidance method, it is possible to prevent collisions between drones and other moving objects in real time. A sequence of figures 4.9 demonstrates, how two drones avoid moving obstacles, at the same time not colliding with each other. In this experiment the drones were commanded to take off and hold their initial X, Y-positions, being aware of the obstacles' locations. After that the obstacles were pulled towards the hovering drones, disturbing robots locations. The drones at that time were repelling from these moving obstacles and each other when they were commanded to fly close, see Figure 4.9.

Local motion planner based on artificial potential fields (APF) allows correcting formation agents positions preventing collisions. In addition, this algorithm can be applied to any kind of robot formation, with real or virtual leaders. Several experiments will also be considered throughout the paper, featuring formations with different shapes. In Figure

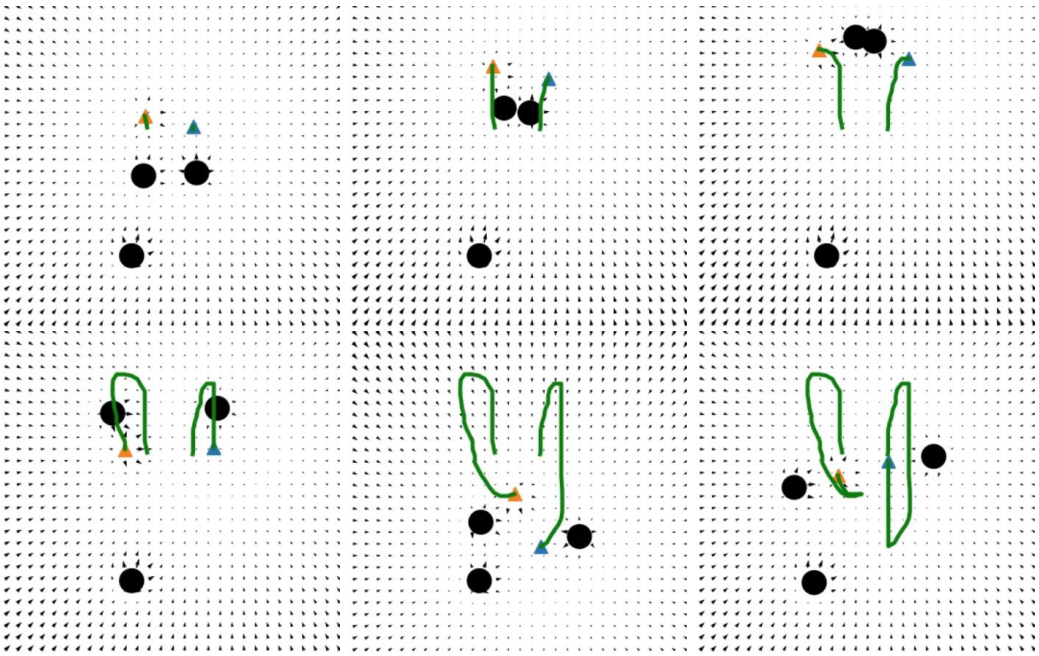


Figure 4.9: Two drones affected by the potential field of moving obstacles. Drones are represented with small triangles with green trajectories. Obstacles are black circles. The potential field is visualized for the drone, depicted as a blue triangle.

4.10 the complete sequence of movement for the swarm of 8 agents is shown.

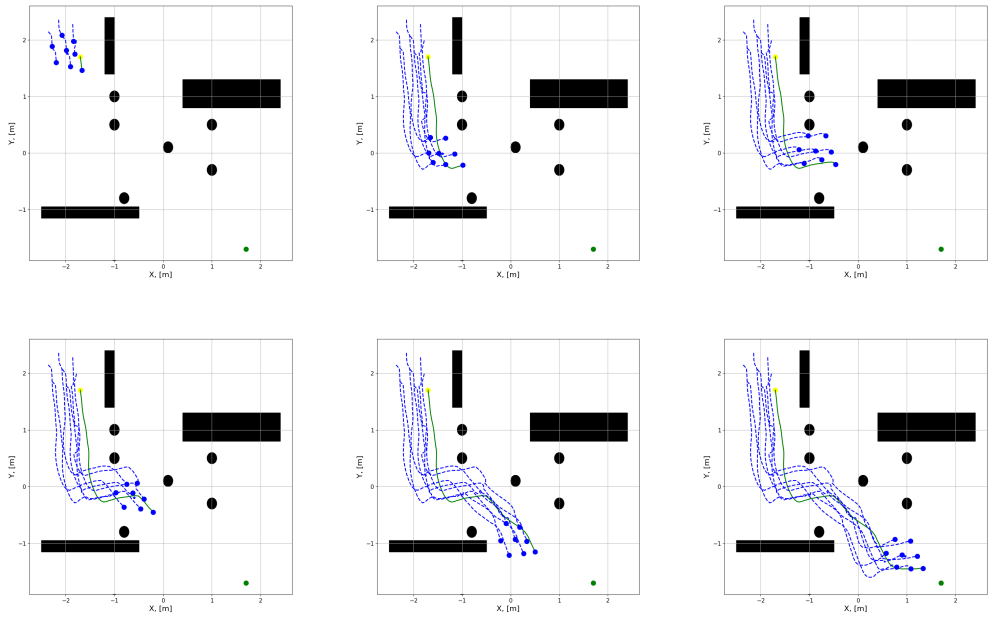


Figure 4.10: Sequence of movement of 8 robots formation with the APF-based path planning algorithm, simulated in a map of static obstacles (black figures).

In most robotic applications, there will be two types of obstacles: static, such as walls; and dynamic, like people walking around, doors, etc. In a real application, a robot formation

must be able to change its path according to the dynamic obstacles in the scenario. Since the leader of the formation is recalculating its complete path in each iteration, the path will always be collision-free for the leader. The followers compute their path to the partial goals, so mobile obstacles do not represent a problem for the followers until they are close to them. The obstacles can be detected in many different ways: cameras, robot sensors, motion capture systems, etc. The next example, Figure 4.11, demonstrates the ability to apply APF-based planner for navigation of the fleet of 4 drones on a map with moving convex obstacles.

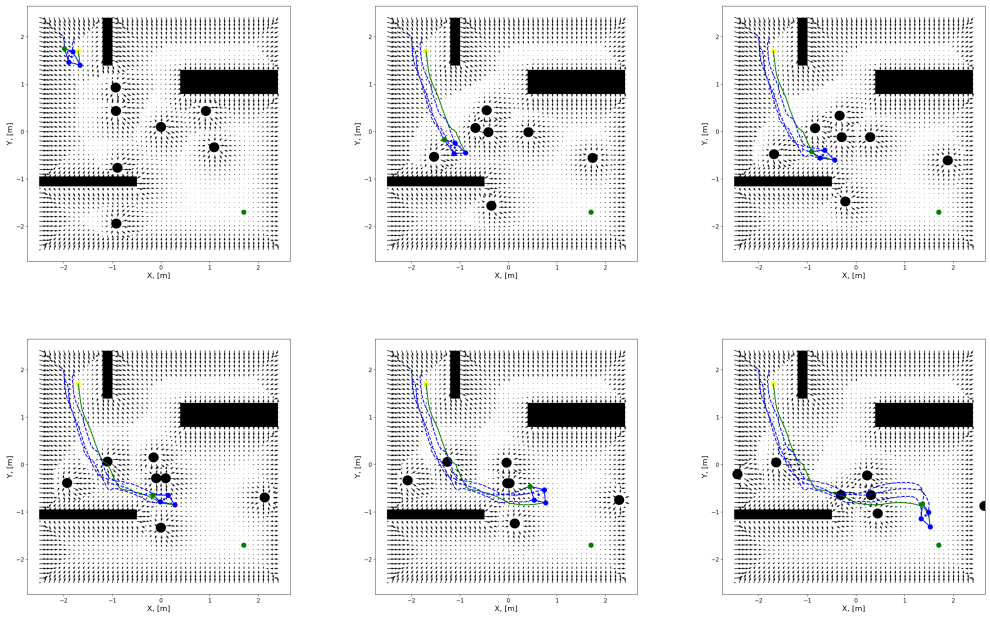


Figure 4.11: Sequence of movement of 4 robots formation with the APF-based path planning algorithm, simulated in a map of moving obstacles (black circles).

Due to the presence of obstacles, the formation of drones changes its shape to avoid collisions. Figure 4.12 represents the deformation of the geometrical configuration of 4 drones, following the paths depicted in Figure 4.11.

There are no rigid links between robots, and inter-agents distances could be changed due to interaction forces in the environment. In such a way, the movement of followers does not replicate the trajectory of the leader-drone. This fact is proved by the velocities graphs in Figure 4.13.

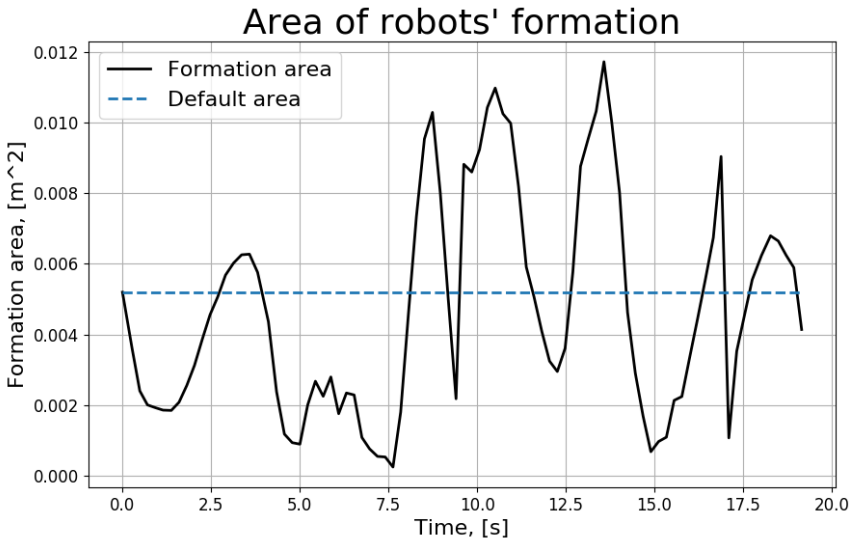


Figure 4.12: Formation of four drones area change. The dashed line represents an initial area when the leader-drone is located at the starting point.

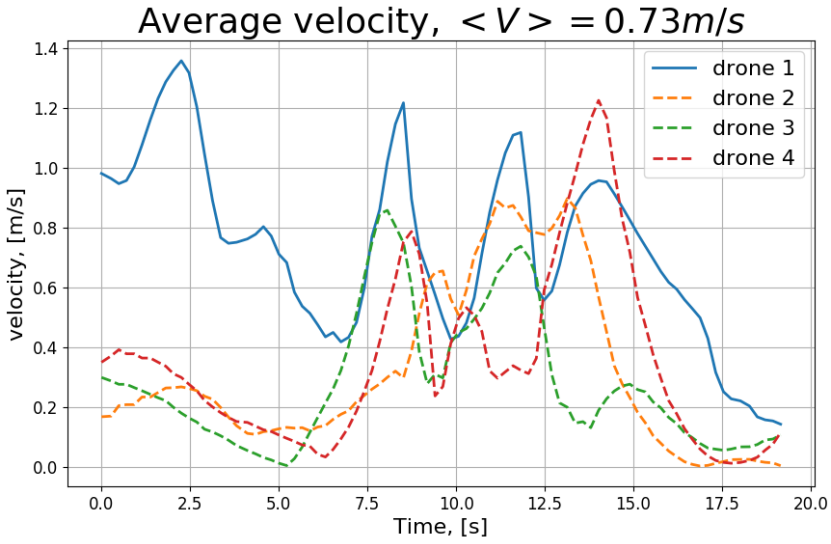


Figure 4.13: Reference velocities for each robot of the formation.

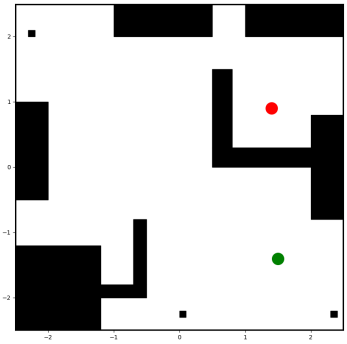
5 Layered Path Planner

At this point, it was described in the previous two sections, how to construct an initial path for the leader-drone (Global Trajectory Planner) as well as the Local Trajectory Planner, which allows to correct positions of each robot in the whole swarm during the leader's location following in prescribed formation. The current part of the thesis is devoted to the two-layered planner description. This is an algorithm that combines the two presented planners (RRT and APF).

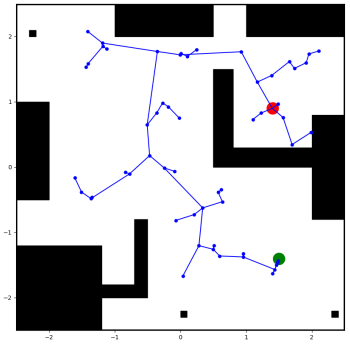
Decoupled approaches become appealing because they divide the big problem into modules that are each easier to solve individually. The general path planning problem is oftentimes divided into approximate initial (or global) trajectory construction, which is further smoothed by a local path planning method. An approximate global planner computes paths ignoring the kinematic and dynamic vehicle constraints. It designs motions for the robots, ignoring robot-robot interactions. Once these interactions are considered, the choices available to each robot are already constrained by the designed motions.

In this work, the layered planner, that combines RRT and APF planners, is proposed. First of all, Figure 5.1, a road map of the environment is constructed with the help of the Rapidly-exploring Random Tree algorithm. After this, an initial approximation of the path is extracted from the tree. It could be noticed, that the obtained green path has a lot of sharp turns. That is why this path is shortened further. The resultant orange path is used as a global planner, while the local path planning algorithm (based on APF method) allows robots not only not to go too close to obstacles and also to avoid moving objects in real-time.

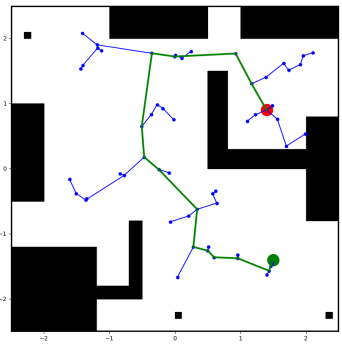
As already was discussed, APF method suffers from the local minimum problem. That is why it is not practical to use this method to plan the whole path from start to the desired location when the environment includes non-convex obstacles. However, the APF algorithm could be used to correct robots' positions locally, i.e. plan smooth short trajectories between the robot's via-points. Local path planner has also a practically important property to allow a robot to avoid moving obstacles on its way, as long as this trajectory planner could be run in real-time. An example of the layered planner trajectory for a single robot is shown in Figure 5.2. Here orange line represents the path generated



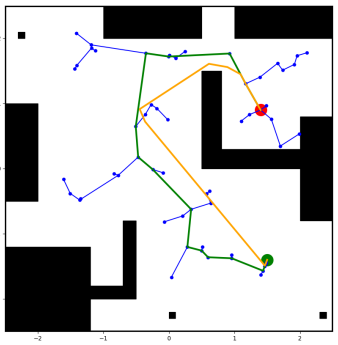
(a) Obstacles map with initial and desired locations.



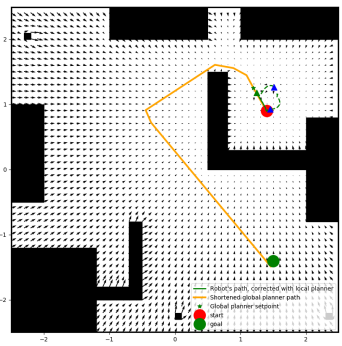
(b) Road map construction with the RRT.



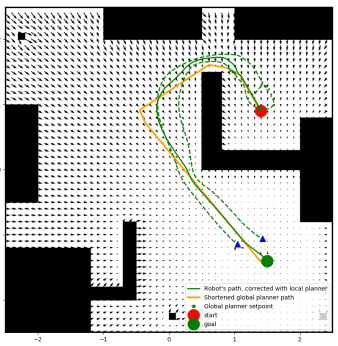
(c) Retrieving the path from the RRT.



(d) Path shortening.



(e) Formation of 3 robots starts to move with the help of the APF.



(f) The swarm trajectories from start to goal. The solid line represents leader-robot movement.

Figure 5.1: General algorithm of the RRT+APF layered planner.

by the global planner (from RRT), while the green curve is the actual drone's trajectory, affected by APF-based local planner. The blue dot on the orange path from the global planner defines a point of attraction for APF-based local planner moving towards the goal. It could be noticed in this example, that the robot not only successfully reached the goal but also escaped from the non-convex bug trap obstacle, not moving too close to the borders.

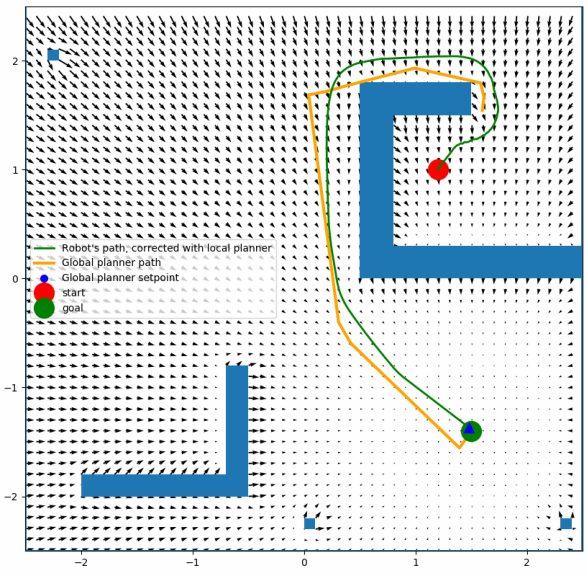


Figure 5.2: Navigation of a single robot in a 2-dimensional environment with the help of the RRT+APF layered planner algorithm. The orange curve here represents the global planner path (from RRT), while the green trajectory is a smoothed with a local planner (APF) version. Quiver plot represents here the gradient of the potential field, defined by the obstacles in the environment.

This method also allows for dealing with moving obstacles. The global trajectory was again constructed apriori not taking into account any possible changes on the map. However, the local planner handles environment deformations successfully as could be noticed from Figure 5.3. Here small moving obstacles were simulated to propagate along straight lines with constant speed.

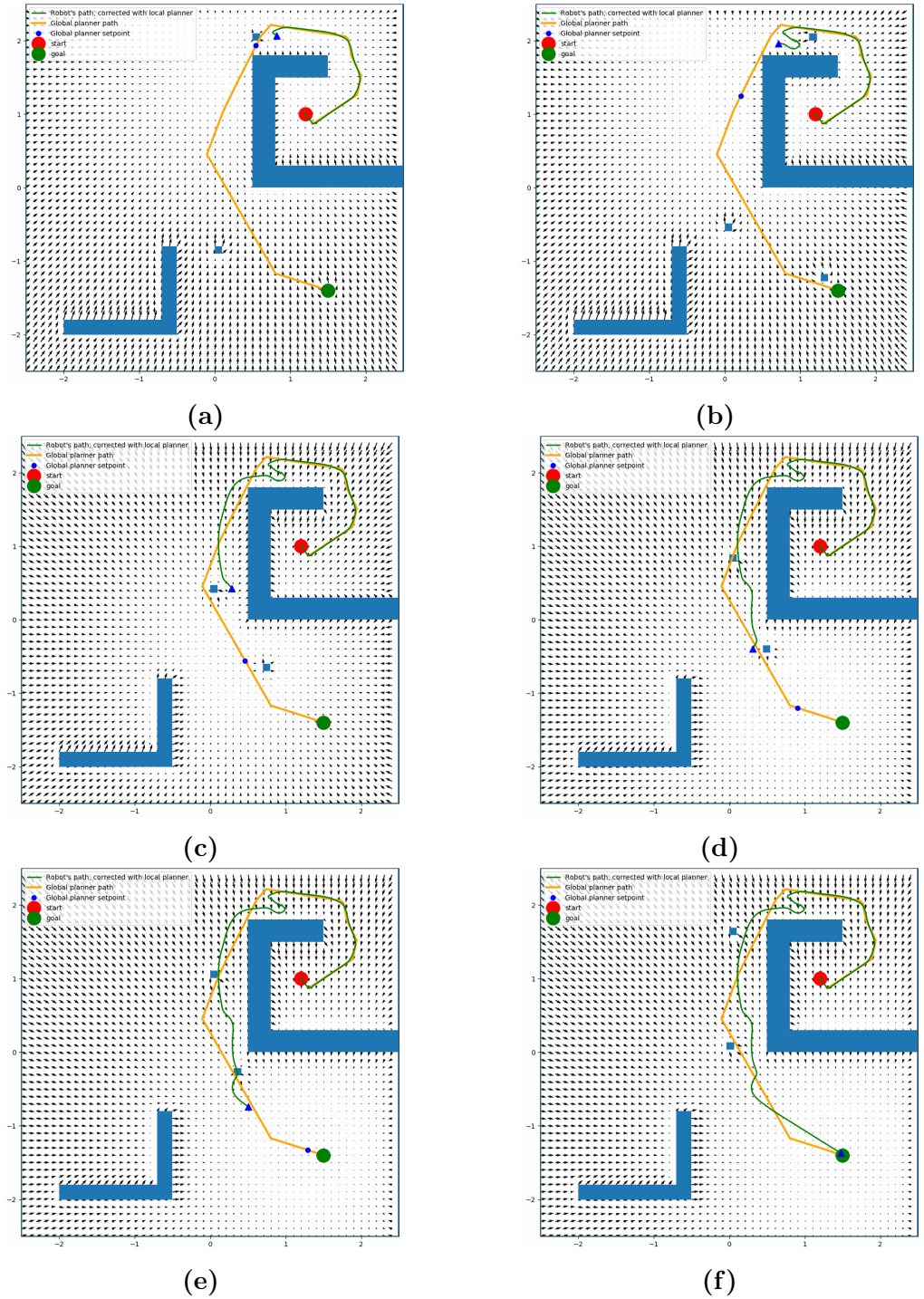


Figure 5.3: Navigation of a single robot (blue triangle) avoiding moving obstacles with the help of the layered planner algorithm. Dynamic obstacles are designed as small squares.

The layered planner could also be applied for robots formation navigation, Figure 5.4. In such a case a global planner is used to construct an initial path for the leader-drone. Followers in this situation adapt their positions relative to the leader with prescribed geometrical formation. In this example, the default formation shape of 4 robots is

rhomboid, each vertex of which defines attractive points for the navigating robots.

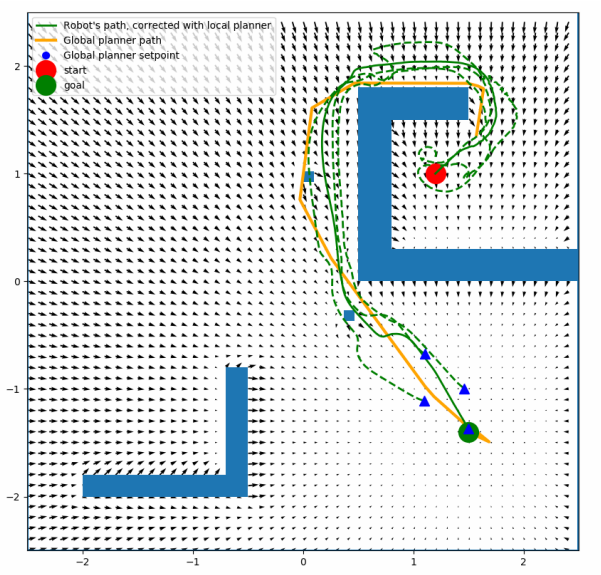


Figure 5.4: Navigation of a formation of 4 robots in a 2-dimensional environment with the help of the RRT+APF layered planner algorithm. The orange curve here represents the global planner path (from RRT) for the leader-robot, while the solid green trajectory is a smoothed with a local planner (APF) path of the leader. Followers trajectories are depicted as dashed green curves. Quiver plot represents here the gradient of the potential field, defined by the obstacles in the environment.

In the same manner, trajectories of all the robots are corrected with the local potential fields-based planner. For each robot in the formation, other agents are considered as moving obstacles in order to prevent inter-drones collisions in the swarm. Figure 5.5 represents robotic formation trajectories, affected by the obstacles. The surface plot here is a repulsive potential of the environment, perceived by one of the robots in the swarm.

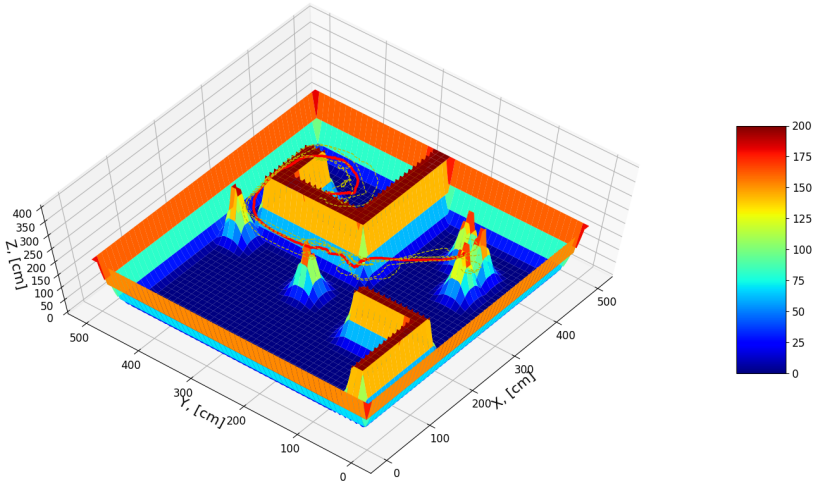


Figure 5.5: Navigation of the 4 drones formation in the field of the repulsive potential, generated by obstacles in the environment. The solid red curve defines the swarm centroid trajectory, while dashed lines correspond to the trajectories of the robots. The surface of the artificial potential includes also 3 picks corresponding to the drones, considered as moving obstacles for the 4-th robot in the group.

6 Experiments

This section is dedicated to the experimental part of the paper. Different simulation tests, as well as real quadrotors flights, are described here. The main goal of the conducted experiments is to reveal the opportunities and limitations of the developed drone swarm navigation algorithm.

6.1 Experimental Setup

In the implementation phase, the decentralized control approach (Mas and Kitts (2010)) was used. In this case, one main computer receives all the information through sensors and communicates the decisions (position set-points) directly to the robots. The Vicon motion capture system was used to track the drones forming a swarm and obstacles using spherical passive infrared markers. Vicon gets the high-quality positioning of the quadrotors with the help of 12 cameras (Vantage V5) covering $5m \times 5m \times 5m$ in the lab. It is also possible to add virtual obstacles, defining their coordinates in software implementation. In such a way, robots have prior knowledge of the environment. In addition, the motion capture system helps in the data collection procedure. For example, it is used to record drones' trajectories and to calculate their velocities. While the motion-capture system creates a single point of failure, it is often chosen over alternatives due to its high performance: typical position errors are less than one millimeter, Merriaux et al. (2017). In comparison, a state-of-the-art decentralized localization system using ultra-wideband radio triangulation showed position errors of over 10 centimeters, too large for dense formations. While vision-based methods are both accurate and decentralized, the required cameras and computers necessitate much larger vehicles, which require much wider flight space. It is also worth mentioning, that lightning conditions do not affect the localization quality provided by a motion capture system.

Although the Vicon provides position information only in a bounded space defined by the cameras fields of view intersection, the covered volume is enough to fly the whole fleet of nano-quadrotors. The hardware platform Bitcraze Crazyflie 2.0 is well suited for swarm robotics due to its small size and weight, Preiss et al. (2017). The *crazyflie ros* stack helps

to deal with an individual robot in such actions like hovering and way-point following and also in more complex multi-UAV cases. A formation of three Crazyflie 2.0 quadrotors was involved to perform the algorithm verification flight tests.

The algorithm proposed in the thesis widely utilizes the Robot Operating System (ROS). ROS provides useful tools and packages in order to fly small quadcopters both individually and as a group. It is possible to write software code using such programming languages like *C++*, *Python*, and *MatLab* using ROS. The Robot Operating System (ROS) Kinetic was chosen as a framework to run the development software and ROS stack for Crazyflie 2.0, Honig et al. (2015). Position and attitude update rate is 100 Hz for all drones. Before conducting any type of experiment, it is important to make sure that the drones are able to perform a stable and smooth flight, following the desired trajectory. In order to do that, all PID coefficients for position controller were set to default values for Crazyflie 2.0, according to Honig et al. (2015).

6.2 Multiple Robots Flights

One of the experiments, that clearly demonstrates the ability of the navigated robotics formation to change its shape in order to avoid collisions, is moving the swarm through the narrow passage. In this conditions, when the width of the passage is considerably smaller, than the size of the robotic formation, the swarm adapts its geometrical configuration in accordance to the effect of the repulsive forces from the passage borders. Although the RRT algorithm could face difficulties to compute a path through a narrow (in comparison with the robotic formation dimensions) passage in a short time, as it was discussed in the Sec. 3, it eventually finds a route from the initial position to the goal, Figure 6.1.

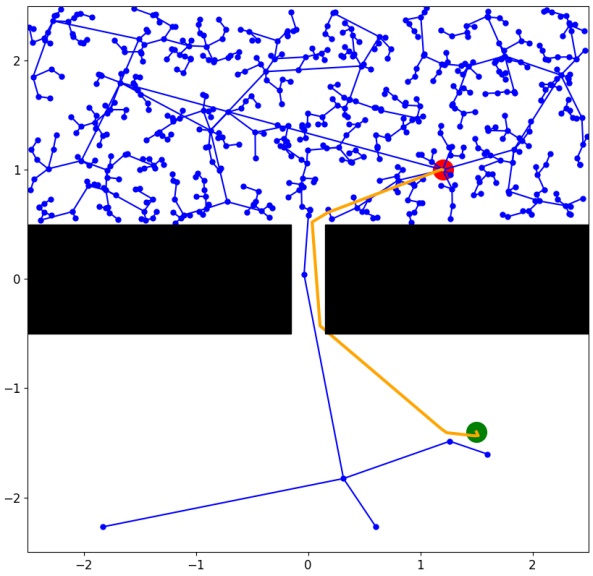


Figure 6.1: Construction of the RRT on the obstacles map, separating start and goal positions by the border with a narrow passage. The orange curve represents a shortened path obtained from the generated tree.

Ones the global trajectory is constructed, it is also important, when dealing with the map, which includes narrow gaps, to choose the influence radius parameter ($d0$ in the Eq. **??**, defining U_r APF) of the repulsive potential to be not larger than the passage width, Figure 6.2. Otherwise, robots could not pass through it, getting stuck in the local minimum not entering the gap between obstacles.

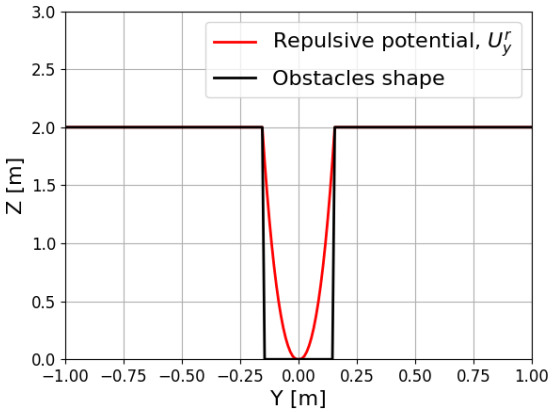


Figure 6.2: Repulsive potential (red curve) and obstacles (black borders of the gap) representation in YZ-plane.

For example, in the simulated experiment of a group of 4 robots navigation through the passage of width 0.3 m, the obstacles influence radius was set to 0.15 m. Ones this condition is satisfied, the robotic formation successfully moves between the two walls, Figure 6.3. The initial rhomboidal geometrical shape of the polygon of the drones is transformed into 4 points approximately lying on the line during movement through the passage. After the swarm escapes the gap, robotic formation returns to its original rhomboidal configuration.

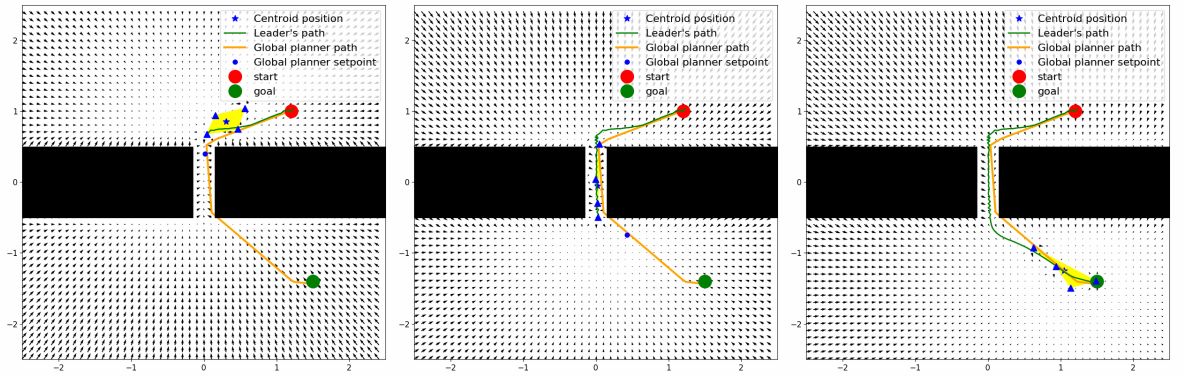


Figure 6.3: Formation of 4 simulated robots navigated through the narrow passage.

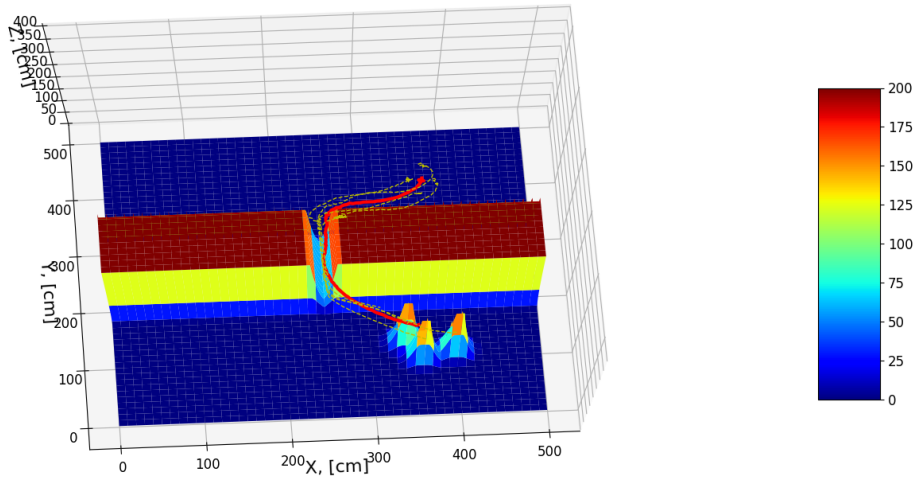


Figure 6.4: Repulsive artificial potential surface plot with drones trajectories (dashed curves). The solid red line represents the swarm centroid route.

The same experiment was conducted with real drones as well. In this case, the quadrotors were commanded to fly at a constant height and were affected by the virtual attractive (to moving goals) and repulsive (from obstacles) forces in the horizontal plane. Figure 6.4

represents the repulsive potential, generated by the passage and free neighbouring drones in the formation. It could be noticed, that the obstacles borders have hyperbolic shape, defined by the function $U_r(x, y)$ in the Local Trajectory Planner section, describing the APF algorithm.

The narrow passage experiment was tested on the real Crazyflie 2.0 drones as well, Figure 6.5. The flight was conducted indoors, using a motion capture system for the drones localization. The location of the narrow passage in space was provided to the group of 3 robots. The gap in this scenario was formed by the space between the two mats placed vertically. The volume on the left from the left mat was considered as occupied, as well as the space on the right from the right mat. This placement of obstacles was forming the same configuration from the top view, as shown in Figure 6.3. The drones were given a task to navigate themselves from their initial point, located on the one side of the room from the gap, to the other part. In such a way, the route to the final location could only go through the passage, constructed with mats. One leader-drone and two followers formed a swarm. Layered planner algorithm was running on the ground station computer, providing position commands to the robots. The robots were also given the set-points of constant height, $h = 0.8$ m, while the vertical dimension of the passage amounted 1.2 m. The gap width at the drones' height was 0.4 m, and the obstacles influence radius was set to 0.2 m. Figure 6.5 consists of a sequence of pictures, representing the navigation of the robotic group. Default triangular geometrical formation of the swarm modifies to a straight line during the narrow passage traversal.

(a) **(b)** **(c)**

Figure 6.5: Formation of Crazyflie 2.0 drones adapts its shape in order to move through the passage. Black arrows represent the direction of the resultant forces (attraction from the goal and repulsive from the neighbouring obstacles) applied to the robots.

Formation of the drones moving together, capable to avoid obstacles on its way and keep inter-robots distances, has a potential to be used for many cooperative tasks, for example, payload transportation, Fink et al. (2011). The same algorithm of the navigation of a swarm through the narrow passage was successfully applied for the group of three drones and a small cargo (5 grams in this example), carried by the robots with the help of ropes, Figure 6.6. However, it is worth mentioning, that the presented path planner method has not yet tuned for the specific task of the parcel delivery.

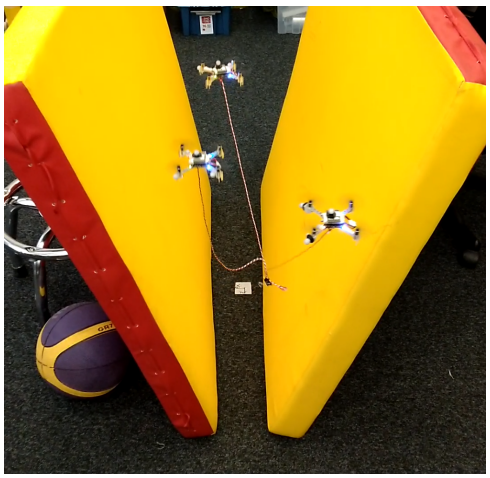


Figure 6.6: Payload transportation with a group of three Crazyflie 2.0 drones, navigated through a passage.

It could be noticed from this example, Figure 6.6, that only 2 ropes out of 3 are stretched at the moment of capturing. This means, that not all of the robots in the swarm pull the cargo at any time period (only 2 drones are involved in the transportation). The mass of the payload to carry is distributed between the robots in the best way when the drones form an equilateral triangle. Designing a path planning algorithm for a group of drones carrying a suspended mass with cables, it is important to make sure that there is always enough thrust produced by the drones involved in the transportation process to hold the payload in the air, Lee (2014). Moreover, if the payload weight is not distributed in a uniform way between all the agents in the swarm (the ropes have a significant difference in tension force), the batteries of the drones, involved in holding process during the most of the time of the navigation task, discharge quicker.

7 Results and Discussions

The conducted experiments data was utilized in order to estimate the performance of the multi-robot path planner method. The main metrics for the algorithm evaluation were chosen as follows:

(a) robots trajectories smoothness (velocity, jerk, snap);

(b) robots trajectories length;

(c) robotic formation centroid trajectory length;

(d) area occupied by the formation in space;

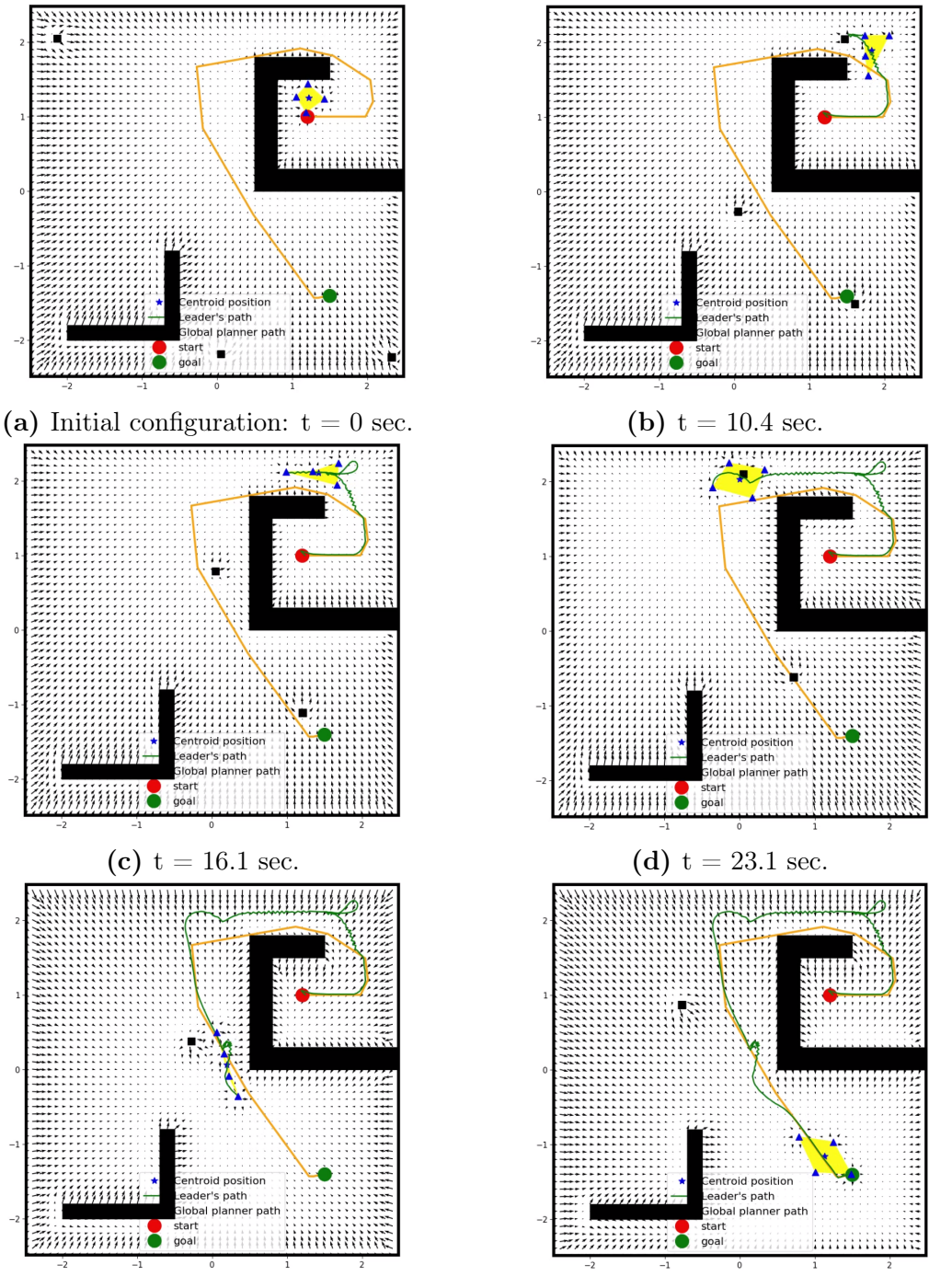
(e) inter-robot distances;

(f) time to reach the goal.

The choice of the parameters to investigate the layered planner efficiency aims not only to obtain the information about individual drones trajectories feasibility but also to evaluate the state of the robotic formation as a united object itself. The idea behind parameters (a), (b), (c) evaluation is motivated in the "Quadrotor Trajectory Smoothness Estimation" chapter. In this paper, the magnitude of the formation area (d), and how it evolves in time during the navigation process, as well as distances between different individual robots (e) were studied as the attributes that belong to the formation state. The inter-robot distances serve an important metric in such swarm robotics applications as payload transportation (as it was discussed in the previous chapter), visual inspection (Schilling et al. (2018)), and communication (Cui et al. (2017)). The dependence of the proposed parameters on the number of robots was studied under different environmental conditions, for example, the geometrical shape of the obstacles on the map (convex or non-convex), their configuration (relative placement) and presence of moving obstacles.

The first simulated experiment discussed already in the layered planner description section, Sec. 5, is aimed to investigate the ability of the algorithm to guide a formation of robots through the map that contains non-convex obstacles (like a "bug-trap" on the upper-right part of the images presented in Figure 7.1). In addition to this, several dynamic obstacles are present in the simulated environment that could influence drones trajectories in

real-time (during the navigation), deforming the formation geometrical shape. These are small (10×10 cm) black squares, travelling along the straight lines. The size of the entire map is defined by 5×5 m area. This choice is motivated by the dimensions of the experimental room in the lab in order to test the algorithm in similar environmental conditions further with real drones.



(a) Initial configuration: t = 0 sec. **(b)** t = 10.4 sec.

(c) t = 16.1 sec. **(d)** t = 23.1 sec.

(e) t = 37.3 sec. **(f)** Final configuration: t = 44.3 sec.

Figure 7.1: Snapshot of the formation of 4 drones (blue triangles) moving. The leader's path is the green curve. The yellow area is the current formation shape. Gradient plot represents the current potential field for one of the followers. The orange line is the trajectory from the leader's initial position (red circle) to the goal (green circle) generated by the global planner (RRT).

It could be noticed, from Figure 7.1, that the robotic formation changes its shape due to interaction with the environment. The robots attract to the prescribed geometrical

points defined by the leader. However, they are also affected by repulsive forces from the obstacles and neighboring robots. That is why the shape of the resulting formation adapts to the map. The next graph, Figure 7.3, represents how the area evolved during the navigation of the 4 drones. The prescribed geometrical formation for the agents has rhomboidal shape, Figure 7.2.

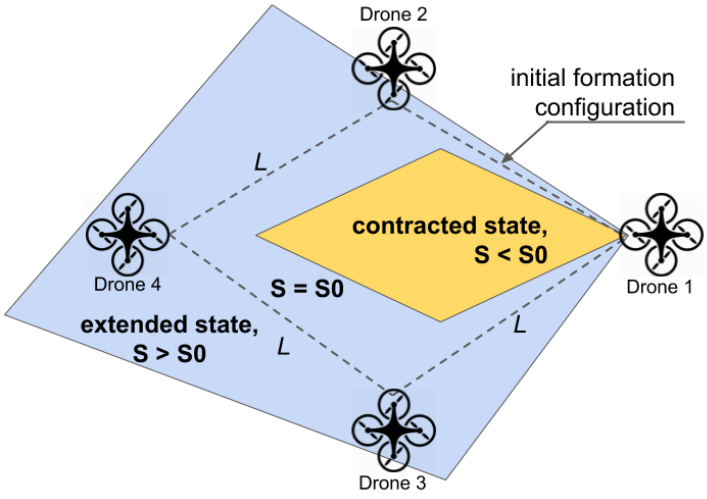


Figure 7.2: Possible formation of 4 drones geometrical configurations during navigation. Default area of the rhomboidal shape is denoted by S_0.

It is worth mentioning, that at the period from 21 sec till 23 sec, and from 41 sec to 44 sec the area of the robotic formation became more than two times wider than in initial configuration, denoted as "default area" on the graph. This was caused in this example by the dynamic obstacle moving inside the formation, disturbing the movement of the drones by generated repulsive forces. Another interesting part of the graph corresponds to the time period from 36 sec to 38 sec. At this moment the area of the robotic formation is close to 0 m^2. However, due to the presence of repulsive forces between neighboring robots, this fact does not mean that the formation is contracted in such a way, that the distances between the drones are too small (it could lead to inter-robots collisions). In fact, the 4 drones are lined up during this period, while moving through the passage, therefore the formed rhomboid is very narrow, and its area is small.

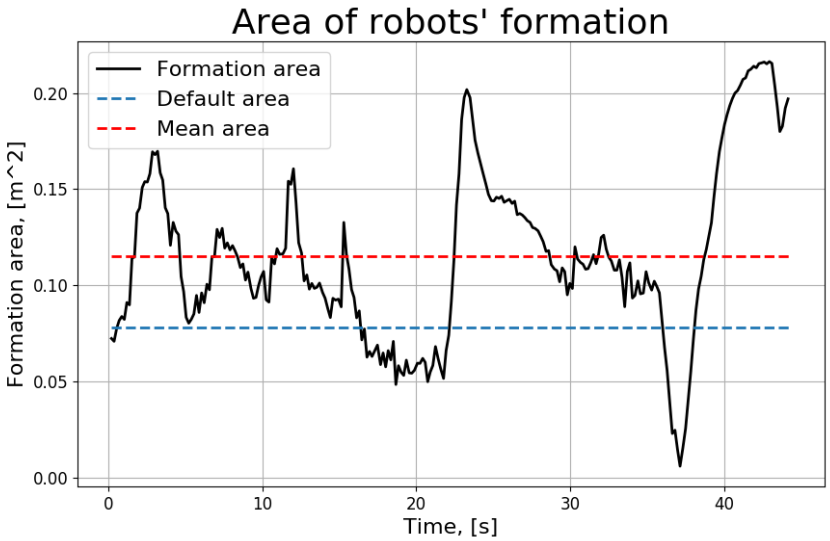


Figure 7.3: Formation of 4 robots area change during navigation. The black curve defines its evolution in time, while the dashed blue line denotes the area of the default rhomboid formation.

Although the trajectories of the robots crossing the map in one formation are very similar, their movements frequently do not have lots of similarities. For example, velocities values of the followers are usually different from the leader's one, Figure 7.4, and Table 7.1.

	Drone 1	Drone 2	Drone 3	Drone 4
Average velocity, $V_{mean}, m/s$	0.35	0.51	**0.56**	0.44
Maximum velocity, $V_{max}, m/s$	1.78	1.38	1.24	**2.31**
Path length, L, m	**10.82**	10.69	10.69	10.70

Table 7.1: Parameters of the Agents in the Formation

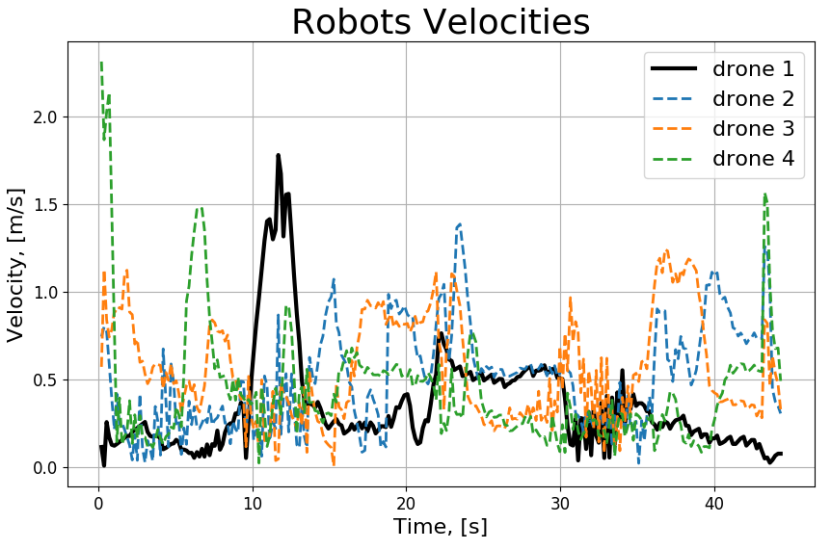


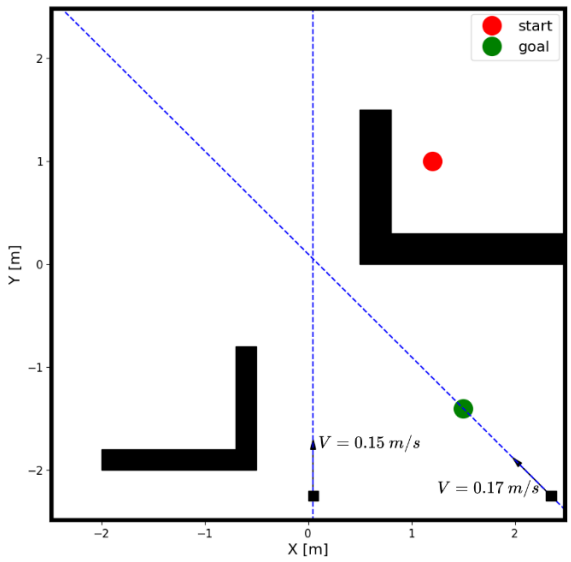
Figure 7.4: Reference velocities for each robot of the formation. The solid black curve defines the motion of the leader, while dashed lines correspond to followers-robots.

The next step is an investigation of the algorithm ability to be used for the navigation of the swarm consisting of a different number of agents. In the next simulated experiment, different formations of drones (3, 4, 5, 6, 7 and 8 robots) were moving through the map of the same configuration of obstacles (2 static and 2 dynamic, Figure 7.5 (a)). Dynamic obstacles were moving with constant velocities along the straight dashed blue lines. Each considered robotic formation has prescribed geometrical shape, presented in Figure 7.5 (b). Yellow circles here defines drones desired locations, where label 1 corresponds to the leader-drone position.

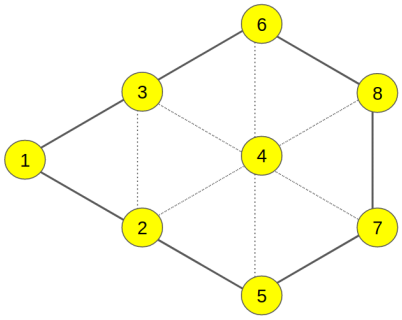
During the navigation, this labeled set-points serve as attraction points for the drones respectively (1 labeled point per drone). Distances between each pair of neighbouring positions (connected with one dashed line) are the same and equal to $l = 0.3m$. The goal of this setup is to evaluate individual robots trajectory smoothness, as well as the whole formation parameters during navigation.

The results of the experiment are summarized in Table 7.2. For each of the formation of drones configuration a batch of 12 simulated trials were conducted (each column of the table provides results for one batch of the trials). Every individual flight provided different results, because of the random nature of the RRT. That is why the trajectory provided by the global planner was not exactly the same in every trial. Therefore it is

important to consider several experiments for each kind of formation shape and take an average of the obtained results.



(a) Configuration of obstacles in the environment. Small squares start their movement with constant velocities.



(b) Default drones configuration.

Figure 7.5

	Number of robots					
Parameters	3	4	5	6	7	8
Time to reach goal, sec	**25.58**	32.72	42.75	43.99	52.09	65.78
Robots path length, m, sec	**6.96**	8.16	9.55	8.93	10.14	11.93
Centroid path length, m, sec	**5.72**	5.88	6.97	6.34	7.23	8.37
Average velocity, $V_{mean}, m/s$	0.39	0.38	0.36	0.40	0.45	**0.48**
Maximum velocity, $V_{max}, m/s$	1.32	2.66	1.71	2.85	2.70	**2.74**
Average acceleration, m/s^2	**0.23**	0.22	0.18	0.20	0.22	0.21
Average jerk, m/s^3	**0.74**	0.67	0.535	0.57	0.62	0.56
Average snap, m/s^4	**3.62**	3.05	2.29	2.24	2.29	1.94
Default formation area, S_0, m^2	0.039	0.078	0.126	0.161	0.196	0.275
Average formation area, S_{mean}, m^2	0.076	0.165	0.231	0.351	0.384	0.504
Maximum formation area, $S_{max}, m/s$	0.152	0.385	0.487	0.875	0.647	0.849
Formation area deviation, $\frac{S_{mean}-S_0}{S_0}$	0.95	1.12	0.83	**1.18**	0.96	0.83
Average formation radius, R, m	0.33	0.51	0.55	0.64	0.62	0.66

Table 7.2: Navigation parameters estimation for the formations of different number of agents.

The drones set-points were moving with a constant speed. For example, for the leader-drone its commanded location was travelling along the global planner trajectory. Points of attraction for the followers were defined by the prescribed formation shape, Figure

7.5 (b). In this experiment, the robots' set-points were moving with the speed equal to $0.33m/s$. However, it could be noticed from the table of results, Table 7.2, that drones average velocity for any number of drones in robotic formation is greater than $0.33m/s$.

That is because the global planner trajectory usually lies too close to obstacles, sometimes it's part even consists of an obstacle border. Due to the correction of the trajectories by the local planner, the lengths of the resulting robots' paths become longer than a route passed by the global planner set-points by the same amount of time.

It is also worth mentioning, that velocities of robots in larger formations are higher. That is because if the number of neighbouring robots increases, every individual drone has more agents on the map to interact with. Thus, the trajectory is getting corrected by more amount of the repulsive forces, which also leads to the increase of the trajectories length.

However, it is interesting to notice, that higher time derivatives of the velocity (acceleration, jerk, and snap) are larger for the formations of the small number of agents. This means that the trajectories of the drones in 8-robots formation are smoother than the routes of the agents in 3-drones group. Therefore (according to the "Quadrotor Trajectory Smoothness Estimation" section) the drones in larger formations should follow the generated trajectories with less deviation.

This experiment also provides information about, what type of formations deviates more from the default geometrical shape. The formation of robots is described by the volume, the drones fleet occupy in space. In a plane, it is an area of the polygon, which vertices are defined by drones positions, Figure (formation of 8 drones). For the case of 8 robots in the formation the drone 4 is not taken into account, because its set-point lies inside the polygon, produced by the other 7 connected points.

Finally, the last parameter in the table, formation radius, R, is defined as the length of the straight line from the swarm centroid to the most distant drone. It is important to have this characteristic of the robotic formation, due to the area does not always provide enough information about the formation shape. For example, if the majority of the drones in swarm aligns along the straight line, the area of the resultant 2-dimensional figure would be close to zero, despite the fact, that the inter-robots distances could be large in comparison of the size of the drones.

8 Conclusion

The layered path planner for a robotic formation navigation was developed and tested on nano-quadrotors. The algorithm allows a group of robots to plan their trajectories on the map with obstacles and does not suffer from the local minimum problem. The experiments with moving obstacles and narrow passages prove that the algorithm behaves well with complex, non-regular, cluttered environments. It is also shown that many different formation shapes can be implemented, depending on the requirements of the specific application. In addition, this approach allows to include any number of robots in the formations, by only setting the desired position with respect to the leader or the other robots. The formation is flexible enough to autonomously decide how to move through the map which consists of obstacles of non-convex shape.

Several experiments have been conducted in order to estimate the algorithm performance under different environmental conditions. The planner evaluation was investigated with carefully developed metrics, which allow to better understand the algorithm capabilities for specific application field. In addition, the navigation method was tested with formations of different number of robots and geometrical shapes. Crazyflie-quadrotors flights performed with the help of the algorithm support the planner applicability to real industrial problems.

Future work is related to expanding the proposed algorithm to outdoor environments, where all the robots of the formation will not be on the same plane (as occurs in 2D); and also to study how to create formations in 3D cases, which is of great importance for unmanned air vehicles (UAVs) flight control. More complex fields can be studied, such as cooperative SLAM with formations, where the uncertainty is present when sensing the environment.

Development of fully autonomous, intelligent quadrotor systems capable autonomously navigate in the environment with obstacles will affect the breadth of industries, including manufacturing and construction, entertainment, inspection and surveillance, and precision agriculture, and will undoubtedly be transformable for technological future.

References

Alonso-Mora, J., Montijano, E., Schwager, M., and Rus, D. (2016). Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 5356–5363. IEEE.

Alonso-Mora, J., Naegeli, T., Siegwart, R., and Beardsley, P. (2015). Collision avoidance for aerial vehicles in multi-agent scenarios. *Autonomous Robots*, 39(1):101–121.

Brunner, M., Brüggemann, B., and Schulz, D. (2013). Hierarchical rough terrain motion planning using an optimal sampling-based method. In *2013 IEEE International Conference on Robotics and Automation*, pages 5539–5544. IEEE.

Canny, J. (1988). *The complexity of robot motion planning*. MIT press.

Cui, Q., Liu, P., Wang, J., and Yu, J. (2017). Brief analysis of drone swarms communication. In *2017 IEEE International Conference on Unmanned Systems (ICUS)*, pages 463–466. IEEE.

Fink, J., Michael, N., Kim, S., and Kumar, V. (2011). Planning and control for cooperative manipulation and transportation with aerial robots. *The International Journal of Robotics Research*, 30(3):324–334.

Fiorini, P. and Shiller, Z. (1995). Robot motion planning among moving obstacles.

Fraichard, T. and Asama, H. (2004). Inevitable collision states—a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024.

Honig, W., Milanes, C., Scaria, L., Phan, T., Bolas, M., and Ayanian, N. (2015). Mixed reality for robotics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Khatib, O. (1986). Real-time obstacle avoidance avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1).

Latombe, J.-C. (2012). *Robot motion planning*, volume 124. Springer Science & Business Media.

LaValle, S. (2006). *Planning Algorithms*. Cambridge University Press.

Lavalle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report.

Lee, S.-J., Baek, S.-H., and Kim, J.-H. (2015). Arm trajectory generation based on rrt* for humanoid robot. In Kim, J.-H., Yang, W., Jo, J., Sincak, P., and Myung, H., editors, *Robot Intelligence Technology and Applications 3*, pages 373–383, Cham. Springer International Publishing.

Lee, T. (2014). Dynamics and control of quadrotor uavs transporting a rigid body connected via flexible cables. In *Proceedings of 53rd IEEE Conference on Decision and Control*, pages 6155–6160.

Lozano-Perez, T. (1990). Spatial planning: A configuration space approach. In *Autonomous robot vehicles*, pages 259–271. Springer.

Mas, I. and Kitts, C. (2010). Centralized and decentralized multi-robot control methods using the cluster space control framework. In *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 115–122. IEEE.

Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525. IEEE.

Mellinger, D., Kushleyev, A., and Kumar, V. (2012). Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *2012 IEEE international conference on robotics and automation*, pages 477–483. IEEE.

Merriaux, P., Dupuis, Y., Boutteau, R., Vasseur, P., and Savatier, X. (2017). A study of vicon system positioning performance. *Sensors*, 17(7):1591.

Mohanan, M. and Salgoankar, A. (2018). A survey of robotic motion planning in dynamic environments. *Robotics and Autonomous Systems*, 100:171–185.

Mohta, K., Turpin, M., Kushleyev, A., Mellinger, D., Michael, N., and Kumar, V. (2016). Quadcloud: a rapid response force with quadrotor teams. In *Experimental Robotics*, pages 577–590. Springer.

Noreen, I., Khan, A., and Habib, Z. (2016). Optimal path planning using rrt* based approaches: a survey and future directions. *Int. J. Adv. Comput. Sci. Appl*, 7(11):97–107.

Palmieri, L. and Arras, K. O. (2014). A novel rrt extend function for efficient and smooth mobile robot motion planning. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 205–211. IEEE.

Petti, S. and Fraichard, T. (2005). Safe motion planning in dynamic environments. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2210–2215. IEEE.

Preiss, J. A., Honig, W., Sukhatme, G. S., and Ayanian, N. (2017). Crazyswarm: A large nano-quadcopter swarm. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Scaramuzza, D., Achtelik, M. C., Doitsidis, L., Friedrich, F., Kosmatopoulos, E., Martinelli, A., Achtelik, M. W., Chli, M., Chatzichristofis, S., Kneip, L., et al. (2014). Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments. *IEEE Robotics & Automation Magazine*, 21(3):26–40.

Schilling, F., Lecoeur, J., Schiano, F., and Floreano, D. (2018). Learning vision-based cohesive flight in drone swarms. *arXiv preprint arXiv:1809.00543*.

Shiller, Z., Gal, O., Fraichard, T., et al. (2010). The nonlinear velocity obstacle revisited: The optimal time horizon. In *Workshop on guaranteeing safe navigation in dynamic environments. In IEEE international conference on robotics and automation*.

Shiller, Z., Large, F., and Sekhavat, S. (2001). Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 4, pages 3716–3721. IEEE.

Tang, S., Thomas, J., and Kumar, V. (2016). Safe navigation of quadrotor teams to labeled goals in limited workspaces. In *International Symposium on Experimental Robotics*, pages 586–598. Springer.

Tsetserukou, D., Tadakuma, R., Kajimoto, H., and Tachi, S. Optical torque sensors for implementation of local impedance control of the arm of humanoid robot. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE.

Tsykunov, E., Labazanova, L., Tleugazy, A., and Tsetserukou, D. (2018). Swarmtouch: tactile interaction of human with impedance controlled swarm of nano-quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4204–4209. IEEE.

van den Berg, J. P. (2007). *Path planning in dynamic environments.* PhD thesis, Utrecht University.

Wong, C., Yang, E., Yan, X.-T., and Gu, D. (2018). Optimal path planning based on a multi-tree t-rrt* approach for robotic task planning in continuous cost spaces. In *2018 12th France-Japan and 10th Europe-Asia Congress on Mechatronics*, pages 242–247. IEEE.

Yang, K. (2011). Anytime synchronized-biased-greedy rapidly-exploring random tree path planning in two dimensional complex environments. *International Journal of Control, Automation and Systems*, 9(4):750.

Zhou, D., Wang, Z., Bandyopadhyay, S., and Schwager, M. (2017). Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. *IEEE Robotics and Automation Letters*, 2(2):1047–1054.

Appendix

A1 Main Algorithms Implementation

In this chapter the Github links to the main algorithms software implementation are provided. The author (https://github.com/RuslanAgishev) decided to make the code open-source for further collaborations on the project development.

- Layered Planner Algorithm: link
- RRT Algorithm: link
- APF Algorithm: link
- Quadrotor Trajectories Generation: link
- Quadrotor PID Controller: link