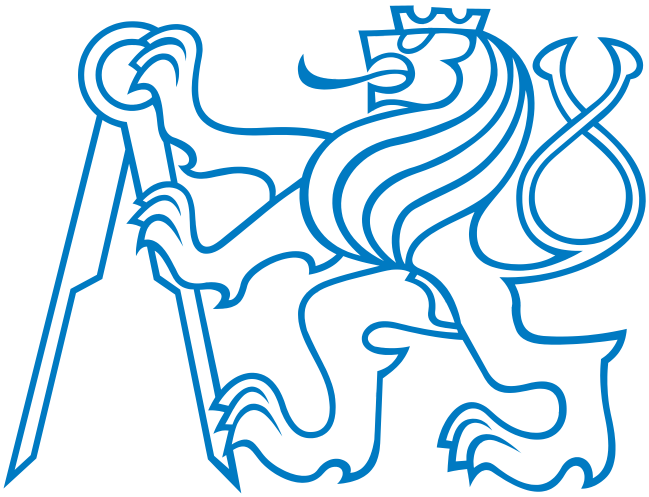
Czech Technical University in Prague
Faculty of Electrical Engineering

DIPLOMA THESIS



Petr Vaněk

Motion Planning for Formations of Mobile Robots and Unmanned Aerial Vehicles

Department of Cybernetics

Supervisor: **Ing. Vojtěch Vonásek**

Prague, 2012

Prohlášení autora práce

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

V Praze dne 11. 5. 2012 .....

.....
Podpis autora práce

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Petr V a n ě k

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Diploma Thesis: Motion Planning for Formations of Mobile Robots and Unmanned Aerial Vehicles

Guidelines:

1. Study state-of-the art sampling-based methods for motion planning (e.g. RRT, PRM..). Select a suitable method and design its extension for motion planning of formations of mobile robots and unmanned aerial vehicles.
2. Study Model Predictive Control (MPC) optimization technique. Desing algorithm for inicialization of the solution using selected sampling-based method.
3. Implement the selected algorithms and extensions into IMR simulation framework.
4. Select another fast method for trajectory initialization.
5. Compare inicialization techniques and their influence to quality of the resulting trajectory.

Bibliography/Sources:

[1] Hess, M. - Saska, M. - Schilling, K.: Application of Coordinated Multi-Vehicle Formations for Snow Shoveling on Airports. Intelligent Service Robotics. 2009, vol. 2, no. 4, p. 205-217. ISSN 1861-2776.
[2] Saska, M. - Vonásek, V. - Přeučil, L.: Control of ad-hoc formations for autonomous airport snow shoveling. In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010 [CD-ROM]. Taipei: IEEE Industrial Electronics Society, 2010, vol. 2, p. 4995-5000. ISBN 978-1-4244-6676-4.
[3] Saska, M. - Vonásek, V. - Přeučil, L.: Roads Sweeping by Unmanned Multi-vehicle Formations. In ICRA2011: Proceedings of 2011 IEEE International Conference on Robotics and Automation. Madison: Omnipress, 2011, p. 631-636. ISBN 978-1-61284-386-5.
[4] LaValle, S. M.: Motion planning, Cambridge University Press, 2006. (též online: http://planning.cs.uiuc.edu/)

Diploma Thesis Supervisor: Ing. Vojtěch Vonásek

Valid until: the end of the summer semester of academic year 2012/2013

prof. Ing. Vladimír Mařík, DrSc.
Head of Department

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S.

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 9, 2012

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Petr V a n ě k

Studijní program: Kybernetika a robotika (magisterský)

Obor: Robotika

Název tématu: Plánování pohybu formací mobilních robotů a helikoptér

Pokyny pro vypracování:

1. Seznamte se s problematikou sampling-based plánovacích metod (RRT, PRM..) a navrhněte jejich rozšíření pro plánování pohybu formací mobilních robotů a helikoptér.
2. Seznamte se s optimalizační technikou MPC (Model predictive control) a jejím použitím pro řízení pohybu formací mobilních robotů a helikoptér. Navrhněte propojení MPC a RRT tak, aby MPC optimalizace byla inicializována řešením nalezeným metodou RRT.
3. Vybrané algoritmy integrujte do simulačního prostředí vyvíjeného v laboratoři IMR.
4. Vyberte další metodu pro nalezení počáteční trajektorie formace.
5. Porovnejte vliv inicializačních metod na kvalitu výsledného řešení v simulátoru.

Seznam odborné literatury:

[1] Hess, M. - Saska, M. - Schilling, K.: Application of Coordinated Multi-Vehicle Formations for Snow Shoveling on Airports. Intelligent Service Robotics. 2009, vol. 2, no. 4, p. 205-217. ISSN 1861-2776.
[2] Saska, M. - Vonásek, V. - Přeučil, L.: Control of ad-hoc formations for autonomous airport snow shoveling. In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010 [CD-ROM]. Taipei: IEEE Industrial Electronics Society, 2010, vol. 2, p. 4995-5000. ISBN 978-1-4244-6676-4.
[3] Saska, M. - Vonásek, V. - Přeučil, L.: Roads Sweeping by Unmanned Multi-vehicle Formations. In ICRA2011: Proceedings of 2011 IEEE International Conference on Robotics and Automation. Madison: Omnipress, 2011, p. 631-636. ISBN 978-1-61284-386-5.
[4] LaValle, S. M.: Motion planning, Cambridge University Press, 2006. (též online: http://planning.cs.uiuc.edu/)

Vedoucí diplomové práce: Ing. Vojtěch Vonásek

Platnost zadání: do konce letního semestru 2012/2013

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 1. 2012

Abstract

This thesis deals with the motion planning for formations of robots and helicopters, where the task is to find a feasible (and possibly optimal) trajectory for all the entities of the formations. A Model Predictive Control (MPC) approach is used to find the optimal trajectories considering the kinematics constraints of the formations. The optimization is defined as a problem of Sequential Quadratic Programming (SQP), which need considerable amount of time to solve. To speed up the optimization, it is useful to provide an feasible initial solution for the SQP solver. We propose to use the sampling-based motion planning techniques like Rapidly Exploring Random trees (RRT), for this task. The advantage of the RRT approach is, that the kinematics motion constraints are considered during construction of the initial trajectory. This helps the SQP solver to find the trajectory in a less amount of time.

Abstrakt

Diplomová práce se zabývá plánováním pohybu pro formace robotů a helikoptér, jehož účelem je nalézt uskutečnitelnou (a pokud možno optimální) trajektorii pro veškeré entity formace. Model Prediktivního řízení "*Modele Predictive Control*" (MPC) je použit k nalezení optimální trajektorie uvažující kinematická omezení formace. Optimalizace je definována jako problém Sekvenčního kvadratického Programování "Sequential Quadratic Programming" (SQP), které vyžaduje značné množství času k nalezení řešení. Aby byla tato optimalizace urychlena je vhodné inicializovat SQP proveditelným řešením. Pro tuto úlohu zde předkládáme vzorkovací techniky plánování pohybu jako jsou Rychle náhodně rostoucí stromy "Rapidly Exploring Random Trees" (RRT). Výhodou RRT přístupu je, že kinematická omezení jsou uvažována při konstrukci inicializační trajektorie. To může pomoci SQP k nalezení trajektorie v kratším čase.

Acknowledgements

Firstly I would like to thank my supervisor, Ing. Vojtěch Vonásek, who devoted lots of his time, effort and patience to help me with the thesis. Further, I would like to thank my family who support me all the time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Formations of mobile robots are becoming more interesting in these days. The formations are groups of mobile robots, which try to establish a predefined geometric shape while solving a given task. A contribution of robot formations is studied in these days, especially how to make diverse activities easier. The applications of mobile robot formations can be found in various areas like search & rescue missions [33], airport snow shoveling [9, 29] or load transportation [36]. The Figure 1.1 shows an example of the snow shoveling task in an experimental laboratory environment.

Figure 1.1: Example of autonomous snow shoveling in a laboratory environment.

One of the crucial tasks in the area of the mobile robot formations is the motion planning. The motion planning deals with finding a feasible trajectory for the individual robots in the formation from a given initial location to a desired goal region. As the robots have to keep the predefined distances in the formation, these constraints have to be considered in the motion planning. To decrease the

energy consumption of the robots, it is required to a find optimal trajectories for the robots, rather than non-optimal ones. Finding the optimal trajectories however requires considerable amount of time, which is not preferable for real time applications. One of the approach to speedup finding the optimal trajectories is using a suitable and feasible initial solution for the optimization technique. This is studied in the presented thesis.

To find the initial solution, we employ sampling-based motion planning methods, which are widely used in robotics. After the initial trajectory is obtained from the sampling-based planner, it is passed to the optimization process. As the Model Predictive Control (MPC) is used for the optimization, the trajectories can be found in both static and dynamic environments.

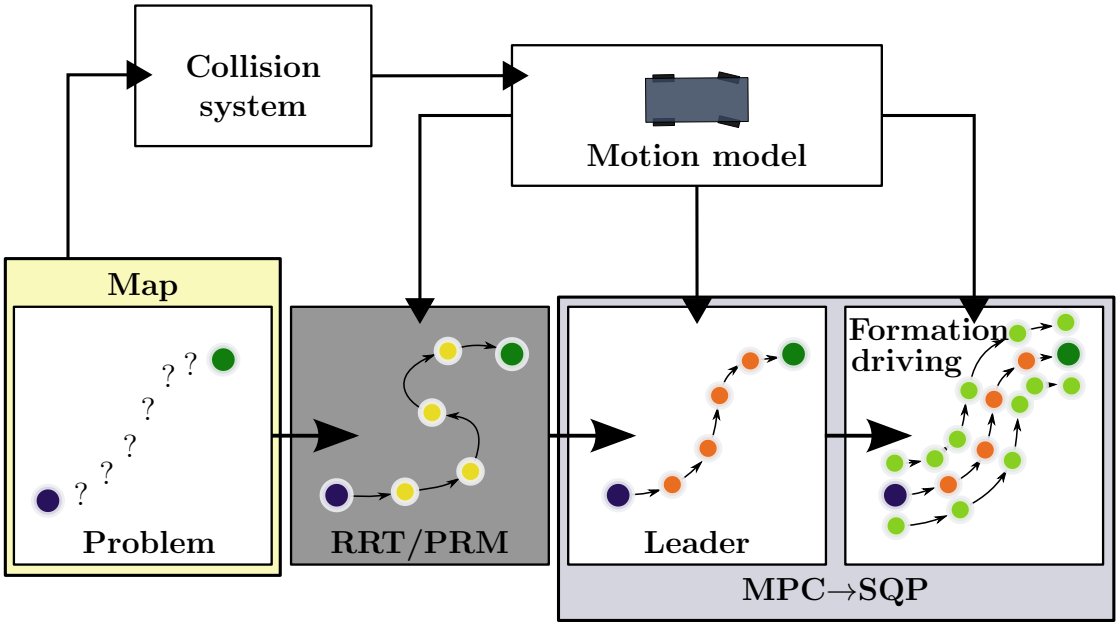


Figure 1.2: The hierarchy of the motion planning task for formation of mobile robots.

This thesis is extension of our previous work [38], where the sampling-based methods (Rapidly exploring Random Tree) for motion planning of a single robot were studied. On the base of these knowledge the direction of interest was aimed to the motion planning of mobile robots formations. The Figure 1.2 shows the combination of used "modules" as a system suitable for the task of motion planning of mobile robots formations. Particular parts of the system were designed and implemented into a framework.

First window **Problem** shows input parameters of the motion planning process, where an initial position, a goal position and a map is known. The task of the

system is to find motions for the individual robots in the formation from the initial position to the goal position. The map is used in the **collision system** for a **Motion model** to prevent a crash with the obstacles in the environment. All of those basic blocks are used to create an initial trajectory for a single robot by the **RRT/PRM** module. Next, the trajectory is used for **Leader** motion optimization, and this optimized trajectory is used for **Formation** driving. For the formation driving, a combination of Leader-Follower [6] approach and MPC is employed. Hence, only a one trajectory for the leader robot needs to be provided by the motion planning module (RRT/PRM).

1.1 Organization of the Thesis

The thesis is organized in the following way. In the next section, the used notation and specification of the used robots is described. The methods for finding the initial trajectories for a single robot are described in the Chapter 3. The main part of this chapter deals with sampling-based methods, which are suitable to find the trajectories for robots considering their motion constraints. To use these method for finding a trajectories for the whole formation, the Leader-Follower approach is employed, which is described in the Chapter 4. The problem of finding optimal trajectories for the whole formation under the Leader-Follower approach is described in the chapter 5.

Chapter 2

Motion preliminaries

The thesis deals with motion planning of formations of mobile robots. For this purpose, we need to define the models of the used robots, as well as the notation of configuration space, which is used in Chapter 3 to explain sampling-based motion planning methods.

2.1 Configuration space

The *Configuration space* is an abstraction space, also know as a *C-space* or simply $\mathcal{C}$ [22], which is frequently used in path planning of complex robots with many degrees of freedom (DOF). The dimension of $\mathcal{C}$ is equal to the number of DOFs of the robot. In this thesis, we use car-like robots moving in the plane, which can be described by a configuration $q \in \mathcal{C}$, where $q = (x, y, \phi)^T$. Then $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1$ which means that the configuration space is composed from a translations $x, y \in \mathbb{R}$ and a rotation $\phi \in \langle -\pi, \pi) = \mathbb{S}$.

The obstacles define the *obstacle region*, where the robots cannot move

$$\mathcal{C}_{obst} \subseteq \mathcal{C}. \tag{2.1}$$

For its definition is necessary to know a *world* $\mathcal{W}$. In this world an *obstacle* $\mathcal{O} \subset \mathcal{W}$ and a rigid body *robot* $\mathcal{A} \subset \mathcal{W}$ are contained. The robot occupy the world in configuration $q \in \mathcal{C}$ which implies that $\mathcal{A}(q) \subset \mathcal{W}$. Then the obstacle region can be expressed as

$$\mathcal{C}_{obst} = \{q \in \mathcal{C} | \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}. \tag{2.2}$$

On the other side a *free space* is another part of the configuration space in which the robot is not in a collision with any obstacle and it is denoted as

$$\mathcal{C}_{free} = \mathcal{C} \backslash \mathcal{C}_{obst}. \tag{2.3}$$

If the $\mathcal{C}$ is a topological space and $\mathcal{C}_{obst}$ is a closed set, this is subsequent of that the $\mathcal{O}$ and $\mathcal{A}$ is closed set, then the $\mathcal{C}_{free}$ is open space. This definition implies every configuration of robot arbitrarily close to an obstacle until it is in $\mathcal{C}_{free}$. If the robot is close enough to an obstacle, it means $\mathcal{A}$ touches $\mathcal{O}$

$$\text{int}(\mathcal{O}) \cap \text{int}(\mathcal{A}(q)) = \emptyset \Leftrightarrow \mathcal{O} \cap \mathcal{A}(q) \neq \emptyset, \tag{2.4}$$

where $\text{int}(\mathcal{O})$ means the interior of the set $\mathcal{O}$, then $q \in \mathcal{C}_{obst}$.

A motion of a robot in the world $\mathcal{W}$ can be computed as a path in the corresponding C-space [21]. The method which find the trajectories directly using the configuration space are more general, and usually they can solve motion planning for various robotic systems. However, the configuration space in not known in advance — it needs to be constructed in order to find a path in it. While the configuration space can be built for simple robots with low DOFs by a brute force (checking all possible states of robots and classifying them as free or non-free), this approach cannot be used for more complex systems. As the dimension of the configuration space grows exponentially with the number of DOFs, a different technique has to be used to find a path in that space. This is solved in sampling-based motion planning methods, which try to cover the $\mathcal{C}_{free}$ by random samples. These methods are described in the Chapter 3.

2.2 Motion Model

Motion model describes how the robot moves according to a control input $u \in \mathcal{U}$, where $\mathcal{U}$ is the set of all possible inputs. For example, the control inputs u for differential-drive robots can be a velocity of left and right wheels. Motion of a robot can be described by the differential equation

$$\dot{q} = f(q, u) \tag{2.5}$$

as a two parameters function, where the first parameter is a position and the second parameter is an input $u \in \mathcal{U}$. By integration of equation (2.5), over the time interval T, the final position of the robot is computed:

$$q(T) = q_{init} + \int_0^T f(q, u)dt. \tag{2.6}$$

The motion models are used in motion planning methods to find motions, which are feasible for the robots. Moreover, planning methods considering the motion models can provide control inputs to drive the robot along a planned path.

2.2.1 Car-like robot motion model

The car-like robots are widely used in robotics. The robots are controlled by forward speed v and steering angle ψ. The car-like robots move along trajectories with curvature $K = \frac{\tan\psi}{L}$, where L is the distance between front and rear wheels. In this thesis, we assume, that the robots are controlled directly by velocity and curvature, hence $u = [v, K]$. Configuration q of a robot is described by its position in the configuration space $q = \begin{bmatrix} x & y & \phi \end{bmatrix}^T \in \mathcal{C}$, where x and y denotes its position in $\mathbb{R}^2$ and ϕ is heading angle from axis x. Motion of car-like mobile robots is

$$\begin{aligned} \dot{x}(t) &= v(t)\cos\phi(t), \\ \dot{y}(t) &= v(t)\sin\phi(t), \\ \dot{\phi}(t) &= v(t)K(t). \end{aligned} \tag{2.7}$$

All parameters in the previous equations are time depended. The robots are controlled in discrete interval of length Δt. During that timestep, the velocity and the curvature are constant which allows to use analytically precomputed integration to obtain the next state $q_n = [x, y, \phi]^T$ from initial state $q_0 = [x_0, y_0, \phi_0]^T$, where

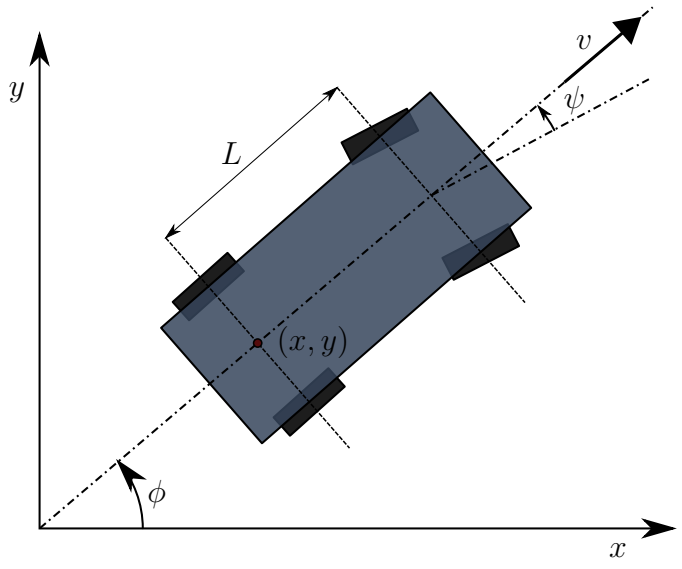


Figure 2.1: Car-like model

$$
\begin{aligned}
\phi &= vT\cos\phi + \phi_0 \\
x &= \begin{cases} vT\cos\phi + x_0 & \text{for} \quad K = 0 \\ \dfrac{1}{K}\left[\sin(vKT + \phi_0) - \sin\phi_0\right] + x_0 & \text{for} \quad K \neq 0 \end{cases} \\
y &= \begin{cases} vT\sin\phi + y_0 & \text{for} \quad K = 0 \\ \dfrac{1}{K}\left[\cos(vKT + \phi_0) - \cos\phi_0\right] - y_0 & \text{for} \quad K \neq 0 \end{cases}
\end{aligned} \tag{2.8}
$$

2.3 Collision detection

Important part of motion planning algorithms is a system for collision detection, where the task is to report whether a given trajectory (or its part) collide with an obstacle. Two types of collision detection can be realized: a) discrete and b) continuous. In the discrete collision detection, the trajectory is discretized and collisions are computed only in the discretized positions. The advantage of this approach is, that is can be implemented for arbitrary types of robots and obstacles. However, some collisions can be missed, if the discretization is too rough. In the continuous collision detection, the collisions are computed analytically for the whole trajectory. This approach is usually slower than the discrete collision detection and often it requires to use only a predefined types of trajectories (e.g. a circular ones) and/or limited types of obstacles (e.g. only boxes or circles). However, the continuous detection can prevent cases visualized on Figure 2.2, where discrete detection can fail.

To speed up the collision detection, we use box representation of the obstacles and the robot is represented as a single circle with radius r. This allows to compute analytically the collision detection, moreover, it allows to compute the distance between the robot and the individual obstacles. This distance is used in the optimization technique as one part of the objective function.

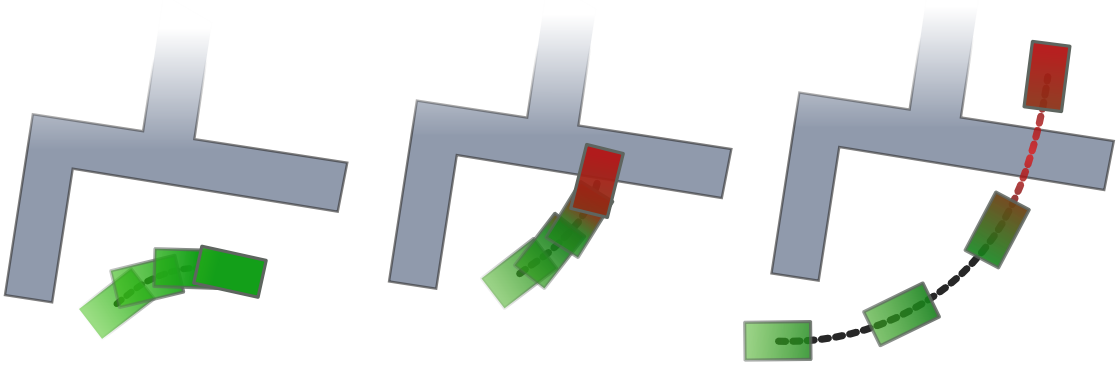


Figure 2.2: Successful and unsuccessful collision detection. Collision free maneuver (*left*). Discrete collision detection with low discretization step (*middle*). Discrete collision detection with higher discretization step. Although the discrete states on the trajectory are collision free, the trajectory goes through the wall (*right*).

Chapter 3

Motion planning

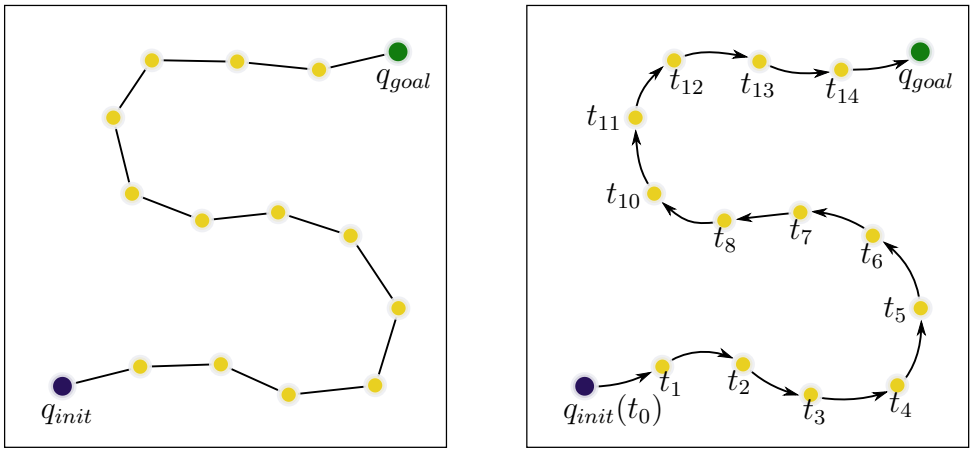
In this chapter, motion planning algorithms in static environments will be presented. The task of the motion planning is to find a feasible trajectory or path for a mobile robot from an initial position to a desired target region. A trajectory is feasible, if a robot moving along it does not collide with the obstacles. The target region can be defined as a point $g \in \mathcal{C}$ with a tolerance radius r_g. Every point in this region is a good solution, but better solution is closest to the point g.

In the motion planning problem, two terms describing the *motion* can be used: *path* or *trajectory*. In this thesis both terms path and trajectory are used with different meaning which can make difference between path planning and trajectory planning. Differences between those two terms are the following.

Path is a sequence of points $p = (q_1, \ldots, q_n)$ where $q_i \in \mathcal{C}_{free}$ defining *where* the robot should move. However, the path does not describe *how* it should move. The path can be continuous or discrete. In this thesis, we use the discrete path. The second important propery of the path is, that it can be found without considering the motion model of the robot. Let consider Fig. 3.1(a). Such a path is feasible for a point robot, however, it may not be traversable for a car-like robot with a limited curvature.

Trajectory is a sequence $t = (q_1, u_1, t_1, \ldots, q_n, u_n, t_n)$, which contains free configurations $q_i \in \mathcal{C}_{free}$ and control inputs u_i, which define *how* to move between the confiugrations. Moreover, the trajectory is parametrized by a time t_i to enable controlling the robot.

The difference between the path and trajectory is visualized on Fig. 3.1(a) and 3.1(b).



(a) *path* (b) *trajectory*

Figure 3.1: Difference between path and trajectory. In thesis the path will be marked as a straight line connecting two points together and the trajectory as a curve with a direction marker.

The chapter is focused especially to the family of methods called *randomized planners* or *sampling based methods*, which use random samples to searching the configuration space. The main advantage of these methods is, that they are efficient to solve motion planning in high dimensional space. Moreover, the motion models of the robots can be considered in the methods. As a price for these ability is that the methods cannot specify an absence of nonexistent solution. Another behavior of these algorithms is that they do not find optimal solution and a searched solution needs to be further optimized. Despite these facts, the sampling based methods have been successfully used in many application, as a motion planner for autonomous cars, unmanned aerial vehicles (UAV) [7, 40] or planetary robots [18].

The motion planning methods can be also used outside robotics. For example in Computer Aided Design (CAD), for testing as capability of assembling parts of designed device as dis-assembling. Another example is virtual motion planning for computer games and animation where the planning methods are used as for actor animation or for camera motion. Last but not least interesting place is computational biology, where the motion planning is used for protein folding and/or protein docking, process important for drug design. Here, the problem of planning is how to find a path for docking a small protein to a large one.

3.1 Geometric methods

Firstly invented methods for the motion planning were purely geometric. These methods was used for path planning in known polygonal environments. Usually, the methods build a graph of free points in the environment called RoadMap, and find a path in the roadmap using a graph-search algorithm. Altough the methods can be easily implemented, they do not consider the motion model of the robots. Thus, the provide path for the robots, not the trajectories. Despite these fact, the geometric methods are widely used in robotics. For example, thay can be used to find path in a virtual environment.

Visibility graph First candidate of these methods is designed on principle of building a graph with shortest distance of planed path. The graph is built by connecting all visible corners of obstacles together. Building a graph in polygonal environment can be obtained in $O(n^2)$ [39]. Further path planning from an initial position to the goal position can by found by Dijkstra's algorithm. The example of a visibility graph is depicted on Fig. 3.2.

A disadvantage of this method is, that the planned path is too close to the obstacles. To find paths for a circular robots, the Visibility graph can be computed on a enlarged map. The enlarged map can be obtained by computing Minkowski sum of the map and the circle representing the robot.

Voronoi diagram Opposite to the *Visibility graph* the *Voronoi diagram* is built in such way, that the planned path is in maximal distance to the obstacles. Once, the diagram is built, the path planning can be provided also by Dijkstra's algorithm. The example of the Voronoi diagram is depicted on Fig. 3.2.

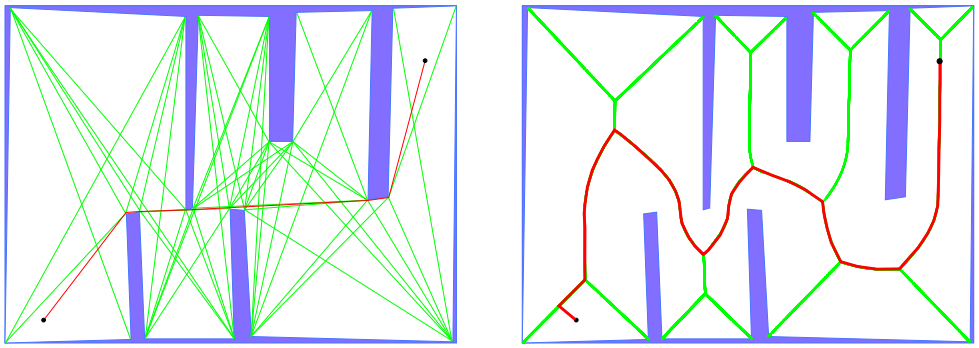


Figure 3.2: Example of Visibility graph (left) and Voronoi diagram in a map. The resulting path between two states is depicted in red.

3.2 Probabilistic Road-Map

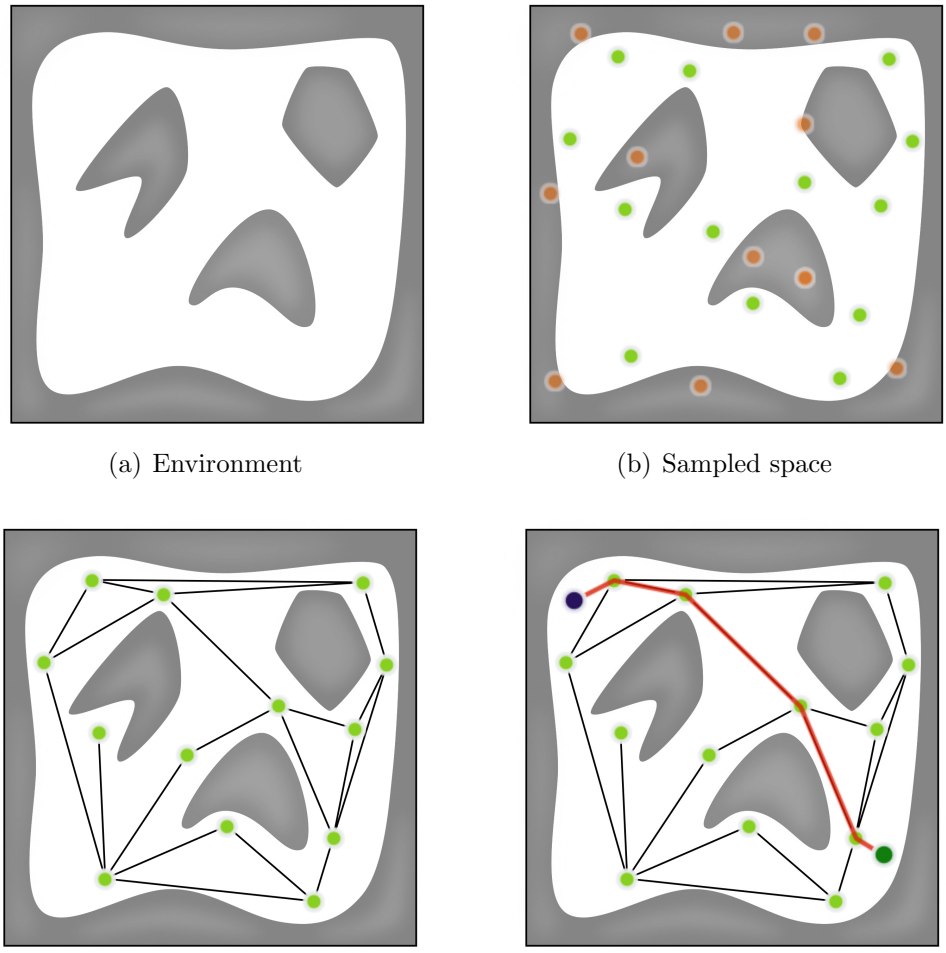
Sampling-based method called PRM (Probabilistic Road-Map) was introduced by Kavraki in [16]. The method belongs to the family called Randomized Path Planners (RPP) which use random samples to create a path from an initial to a goal position. Principle of the PRM method is that the free configuration space $\mathcal{C}_{free}$ is covered by the random samples which create a set $Q_{rand} \subset \mathcal{C}_{free}$. Samples in this set Q_{rand} are connected to a graph (Road-Map). Connection that passes through the obstacle space $\mathcal{C}_{obst}$ are excluded using a discrete collision detection algorithm. For the collision detection, libraries like Rapid [8] and Solid [37] are often used. This step can be computationaly intensive, which can slow down the performance of the method.

The graph (Road-Map) obtained by this method is prepared for the path planning by a graph algorithm. As an example the Dijkstra's graph search algorithm can be used for such a path finding.

This kind of motion planning requires a static environment. Hence, the Road-Map is built only in the beginning of the planning. Another planning is provided so that the nearest suitable samples are searched in the graph, this means that the connection between the initial position and its nearest samples in graph is provided and all the same for the goal position. Necessity of rebuilding the graph is needed only in the environment with changes. Thats why this method is not useful in dynamically changing environments.

The performance of the PRM method can be descreased in environment containing narrow passages. A narrow passage is a such part of the environment, which is significantly smaller than the other parts. The example of a narrow passage is depicted on Fig. 3.5. To prepare a roadmap which can provide path through the passage, several samples has to be placed into the passage. However, as the samples are generated randomly in the whole configuration space, the probability of placing samples in the small passage is low. Hence, the PRM method ofter fail on such types of environment.

Gaussian Sampler To increase the performance of the PRM algorithm on maps with narrow passages, several methods have been proposed. An improvement of the standard PRM method is based on choosing samples for building the Road-Map with density of Gaussian probabilistic distribution around the obstacles [5]. It means that large open regions in the configuration space are not widely sampled and samples are focused to the space around obstacles, which also increases the chance to put samples into a narrow passage. Therefore the number of samples for finding a path is smaller in comparison with standard PRM which use uniform sampling method. This method can improve the path planning especially in environments with narrow passages. On the other hand, its performance can be



(a) Environment (b) Sampled space

(c) Built Road-Map (d) Planned path

Figure 3.3: The process of building of PRM is figured here in a know environment (a). Firstly the environment is sampled (b) and collision states (*orange points*) are excluded. Next the Road-Map is built by connecting free samples (*green points*) (c). Finally the path planning can be provided by a graph algorithm (d). The initial q_{init} and the goal position are connected to its nearest position in Road-Map.

worse on environments without obstacles (e.g. large planar rooms). The number of wasted samples can by payed by higher computational cost. For this reason a combination of the original PRM sampling and the Gaussian sampling [5] was proposed.

Randomized Bridge Builder Another improvement of the PRM method using hybrid sampling strategies was introduced by Sun in article [34]. This method use so called Bridge-Test and then, method called Randomized Bridge Builder (RBB) is run. The Bridge-Test is a method where the collisional sample $x \in \mathcal{C}$ is searched first. The next step is to search another collisional sample $x' \in \mathcal{C}$ with distance d to x, where d is computed from Gaussian distribution function. Finally the middle sample $q \in \mathcal{C}$ is searched in half between x and x'. If the middle sample q is collision free, it can be added to the Road-Map. This is why the most of the samples are added to narrow passages and also into corners of the configuration space $\mathcal{C}$. The best behavior of this method is in combination with uniform sampling which searches solution with minimum samples in short time.

The presented PRM methods provide good results in high dimensional configuration space. However, the PRM-based methods are primarily designed for a path planning, and they needs another boosting if the trajectory planning is required. This is typicaly solved a by two-phase approach: first, a path is found using the PRM method and then, the trajectory is computed from the path. While this approach is suitable for simple differential-drive robots, it cannot be used for robots withing a formation. The reason is, that motion of the robots in the formation is constrained by the shape of the formation as well as by the kinematics of the individual robots. As the PRM does not considred these constraints, it can produce such a path, which cannot be physically followed by the robots.

Different sampling-based approach to find trajectories for the robots in a formation is described in the next section.

3.3 Rapidly-exploring Random Trees

The rapidly Exploring Random Trees (RRT) method was created by Steven M. LaValle in 1998 [20]. The method is widely used under the shortcut RRT or RDT with meaning *Rapidly-exploring Dense Trees*. Main idea of this method is to build a tree of feasible configurations $q \in \mathcal{C}_{free}$ from an initial position $q_{init} \in \mathcal{C}_{free}$ through the free configuration space until the goal region is reached. The process of building the tree is simple and it is listed in Algorithm 1. For a better imagination the interesting points of building the tree can be seen on Figure 3.4. At the beginning of algorithm, the root of the tree is created in the initial position q_{init}. Then, a random sample q_{rand} is generated which is used for tree growing. If the tree is grown by state q_{best}, the distance δ between q_{best} and q_{goal} is measured as shown a line 5 of Algorithm 1. Whenever the measured distance is smaller than chosen ϵ, the searching is complete and solution can be returned.

The way how to grow a tree is mentioned in Algorithm 2. Firstly, the nearest

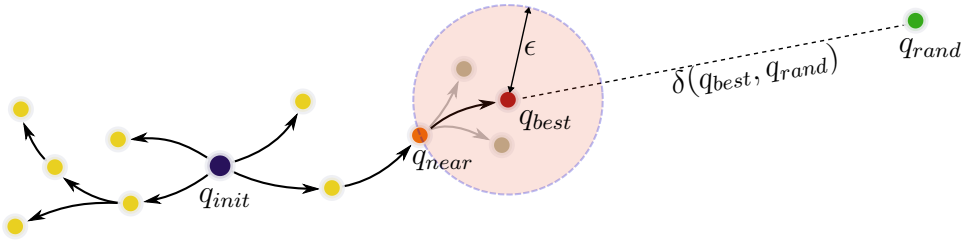


Figure 3.4: Interesting points within a process of building the RRT. A gray colored points represent another expansion from the nearest state q_{near}, but its distance to the random state q_{rand} is grater than distance to q_{best}.

Algorithm 1: RRT

Input: Initial position q_{init}, Goal Position q_{goal}.
Output: Output trajectory

T = Initialize the tree T with root q_{init}
for $i = 0$ **to** max **do**
    q_{rand} = create random state
    q_{best} = grow the tree T toward a q_{rand}
    if $q_{best} \wedge \delta(q_{best}, q_{goal}) < \varepsilon$ **then**
        return extract an solution from the tree T
return $\emptyset$

Algorithm 2: Grow Tree

Input: Tree T, random state q_{rand}
Output: Best state to random state q_{rand}

q_{near} = find the nearest neighbor of a q_{rand} in the tree T
q_{best} = pick control (q_{near}, q_{rand})
if q_{best} **then**
    $T \rightarrow$ add point q_{best} to the tree T
    return q_{best}
return $\emptyset$

neighbor q_{near} of the random state q_{rand} is searched. The nearest state q_{near} is used for further expansion, which gives the best state q_{best}. When the expansion succeed, the best state q_{best} is added to the tree. For picking the control that expand the nearest state q_{near} to the best state q_{best} is used in Algorithm 3. For the expansion of the tree, all control inputs $\mathcal{U}$ are used: each control input $u_i \in \mathcal{U}$,

$i = 1, 2, \ldots, n$, is tried for expansion of nearest state q_{near}. This process creates the new set of reachable states. From this set, only the one with the shortest distance to q_{rand} is selected and marked as a q_{best}.

Algorithm 3: Pick control

Input: Near state q_{near}, random state q_{rand}
Output: Best state q_{best} or $\emptyset$

```
1  d_min = δ(q_near, q_rand)
2  foreach u ∈ U do
3      q = expand q_near by u
4      if q is in collision then
5          continue
6      d = δ(q, q_rand)
7      if d < d_min then
8          d_min = d
9          q_best = q
10 if q_best was found then
11     return q_best
12 return ∅
```

For the expansion it is needed to know how to expand the state using an input u. This is solved by using motion model of the robot. The motion model can be implemented as mathematical expressions or in a numerical way like a physical simulator. Each method has their advantages and disadvantages. For example, the motion of a modular robot on a rough surface can be so complex, that is impossible to get a sufficient solution of the mathematical expression; in such a case, the motion can be simulated. On the other side, mathematical expression could be more precise and fast for simple robots, like for differential drives or car-likes. During the expansion, we use the discrete motion model (2.8).

The collision detection in the newly reached states has to be performed to discard the unfeasible motions. Similarly to the PRM, this is done using the dicrete collision detection.

The basic RRT algorithm is easy to implement and simple for using with different types of robots, which could be simple or complex to control, like mobile robots, robotic manipulators, drones or 3D objects. The main advantage of the RRT method is that a returned solution is sequence of control inputs, which could be applied to the robot directly in specific time. Also the motion planning can be provided in complex environments. On the other hand, the method can be slow in complex environments, especially in the environments containing narrow pas-

sages. The slowness in these environments is caused by frequent collision checking, and also by wasting lots of samples in useless parts of environment. For this reason, different improvements were proposed by adding a heuristic to the standard method.

Goal Bias One of improvements that should speedup the standard method is the technique using for boosting growth of the tree toward the goal q_{goal}. The principle of the technique is to switch the random sample q_{rand} to the goal position q_{goal} in each k-iteration of the RRT method. Instead of switching every k-iteration, it is better to do the substitution with a certain probability which maintain a principle of randomness [25]. When the environment between an initial position q_{init} and the goal position q_{goal} contains few obstacles, the output trajectory should be searched in short time.

The modification can achieve better solution in many cases. However, its performance can be worse in environments containing complex obstacles. The example can be environment known as a *bug trap* which is shown on Figure 3.5. The environment contains a narrow passage, but goal bias tries to grow the tree toward a goal position q_{goal} which is opposite to the narrow passage. This decreases a chance of finding a solution, because the goal bias attracts the tree over the obstacle.

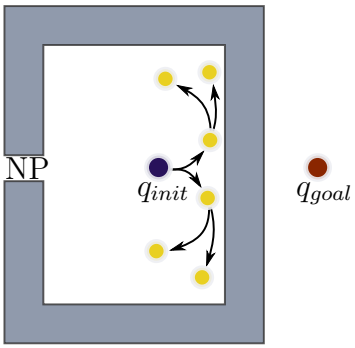


Figure 3.5: Example of a Bug Trap. The NP denote the narrow passage.

Goal Zooming Better results can return a method which is an extension of the Goal Bias. A sticking behind obstacles is tried to remove by the following heuristic. The principle of the technique is to select a random sample q_{rand} from a region around goal position q_{goal} with radius which changes over the time [25]. The region size is controlled by nearest sample in the tree to the goal q_{goal} in every iteration, which has an effect that the tree is focused to the goal q_{goal} as the RRT grow near to it.

By adding this region, the modification should achieve better solution either in environments with lots of obstacles. Similarly to Goal bias, the Goal zooming can fail to solve the Bug Trap problem.

RRT-Bidirectional Another modification of the standard RRT technique is called RRT-Bidirectional, which uses two growing trees. One is rooted in the initial configuration q_{init} and the second one has root at the goal position q_{goal}. Both trees are growing until the connection between them can be provided. It occurs when one state in the first tree has a neighbor in the second tree near than a predefined tolerance ϵ. The predefined tolerance ϵ should be small, near to zero, which increases the searching time. Motivation for the bidirectional method was environments with narrow passages where the previously described methods fails.

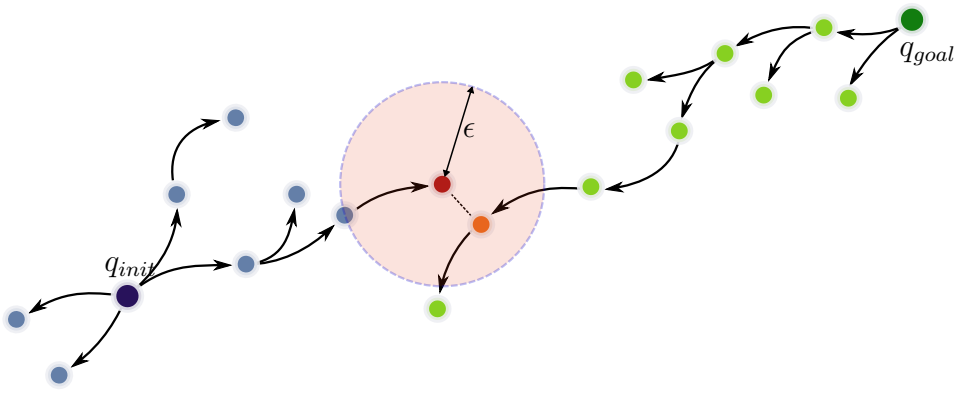


Figure 3.6: The principle of the RRT-Bidirectional. One tree grows from an initial position q_{init} (*blue points*) and the second tree grows from a goal position q_{goal} (*green points*). If the expanded state (*red point*) has a nearest neighbor (*orange point*) from second tree with shortest distance than ϵ, the both trees can be connected.

The technique was firstly mentioned by LaValle and Kuffner in article [24]. The principle of method proposed in this article is following. First, the random state q_{rand} is generated and the first tree attempts to grown towards it with a state q_i. If the growing is successful, the nearest state is searched in a second tree toward to the state q_i with distance smaller than chosen ϵ. If such a state exists in the second tree, the output trajectory is searched. If not, a second tree with the root at q_{goal} is attempted to grow by state q_g. On success, the nearest state in the first tree is searched with the distance smaller than ϵ. If the nearest state exists, the output path is found. Otherwise, the technique continues again by generating new

random sample q_{rand}.

A little different principle of the bidirectional trees growing was proposed by Kalisiak in thesis [14]. The random state q_{rand} is generated and toward it the first tree T_A is attempted to grow. If a state q_a extending the tree T_A exists, the second tree T_B is attempted to grow toward q_a. If the extending state q_b exists, the distance δ between q_a and q_b is measured. If the distance is smaller than a predefined $\epsilon < \delta$ the resulting trajectory is provided. Otherwise, the trees T_A and T_B are switched and the searching continues by generating new random sample q_{rand}.

The disadvantage of this method is in the process of connecting the trees. Whether the motion planning is provided for a nonholonomic kinodynamic robot, a link of the trees needn't satisfying a constraint of the robot. This disadvantage can be removed by an approximation function applied to the link.

RRT-Connect The next modification was proposed by Kuffner and LaValle in article [17]. Principle of the method is in generating the random sample q_{rand} and toward to it is grown until the target region is reached or the random sample q_{rand} is reached or the tree cannot be more expanded. If the target region is not reached, the method continues by generating the new random sample q_{rand}. The method quickly searches the space, therefore reaching a solution in short time is supposed. Better behavior of the method is assumed in environment with a smaller number of obstacles or without obstacles.

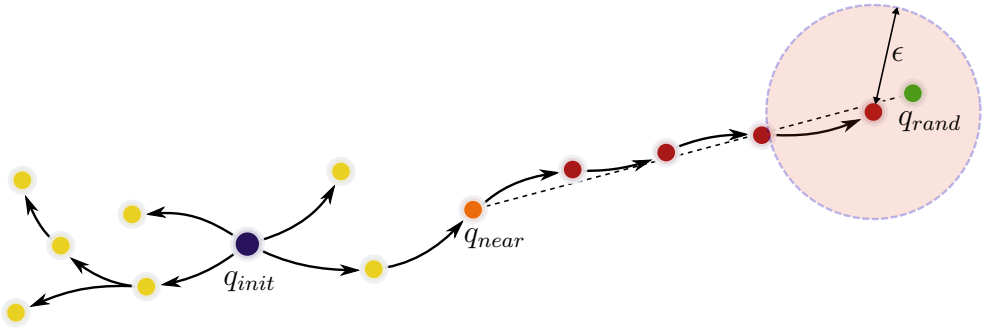


Figure 3.7: Principle of method RRT Connect. If the nearest point q_{near} (*orange point*) is searched, the following expansion (*red points*) is provided until the random point (*green point*) is searched or the collision with obstacles has occurred.

This modification can be used in conjunction with the RRT-Bidirectional method. Kalisiak in his thesis [14] specifies three variants of this conjunction, RRTConExt, RRTConCon and RRTExtCon, where the Con denotes the connection presented RRT-Connect and Ext denote the extension presented in Algorithm 1. Previously

presented the RRT-Bidirectional method should be named as RRTExtEXt under this notation

RRT-Blossom The previously presented method extends the tree only by one state at each iteration. However, situations where other expanded states can contribute to better covering the space also exists. Such an improvement was presented by Kalisiak in [15]. The tree is not grown only by the best state q_{best}, but other states are also added. The name of the method is derived from from a flood-filling behavior, that the blossom of flower can remind. The tree growing is controlled by regressive condition which is defined by following equation

$$\exists q \in T \mid \delta(q, q_{leaf}) < \delta(q_{parent}, q_{leaf}), \tag{3.1}$$

where T is the searched tree, and δ is the distance metrics, Euclidean for example. Thus only the states which do not contribute to searching the space are excluded. A problem can occur, if the planing for the nonholonomic robot is provided, because of dynamic constraints of robot. Distance between expanded states can be so short that only the first expanded state can be added to the tree. Therefore the method can serve only as assumption for more general method presented in following paragraph.

RRT-BlossomVF Strategies that improve RRT-Blossom method by adding the viability attributes to the states and edges are called RRT-BlossomVF, which means the RRT-Blossom with Viability Focus, and it was presented by Kalisiak in [13, 14]. The mentioned attributes of viability are {`untried`, `dormant`, `live`, `dead`} and the transition between them can be seen on Figure 3.8. At the beginning the states and edges are labeled as `untried`. After the expansion, the collision states are labeled as `dead` just as edges that make a connection with their parents. Collision free states are signed by the condition of regression (3.1), where suitable states are signed as `live` and unsatisfactory states are not excluded, but labeled as `dormants`. For further expansion are used only this states, which are labeled as `live`. For completeness of method is necessary to say, that the propagation of states is needed. After labeling the state, the propagation is done from the leaf to the root through the tree. It means, that the edge that is connected to `dormant` sample is labeled as a `dormant` too. If all edges going from state are labeled as `dormant` or `dead` respectively, then the state is also labeled as `dormant` or `dead` respectively.

During this propagation two types of *deadlock* can occur. First one, if all states are `dead`, means that planner cannot find solution between initial position q_{init} and goal position q_{goal}. The seconds case means that all nodes are labeled as `dormant`. If the tree is in this type of *deadlock*, for further expansion the `dormant` states

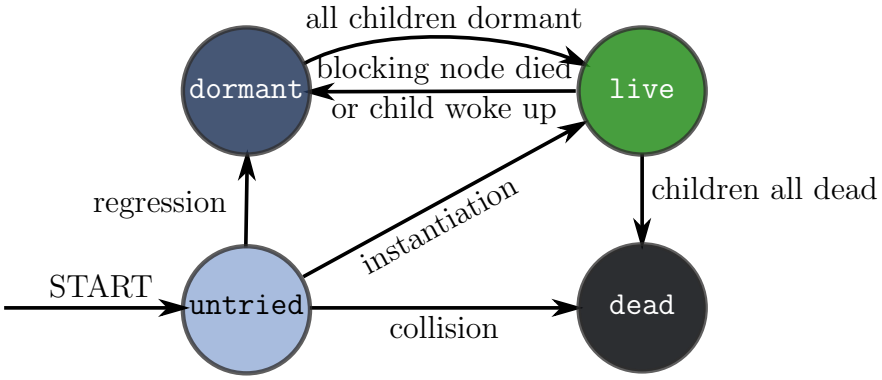


Figure 3.8: Transition diagram between attributes of viability. *Picture was adapted from [14].*

are used either. This allows to the tree expand the states that was excluded by regression.

From previously presented methods only this one can say that the solution of motion planning don't exists. Main advantage of this method is that can reuse states that was firstly signed as unneeded which could be a contribution for finding a solution in environments with narrow passages. Method could be used as an improvement of the original RRT method or for improvement of the Bidirectional RRT method. A disadvantage could be the required amount of memory used for storing information about the states and edges, which could be problem for motion planning of robots with lot of DOF.

RRT-Path The crucial property of the RRT method is the way of exploration the configuration space. This exploration is caused by the nearest-neighbor rule, where the randomly generated states attract the tree toward unexplored regions. However, in certain situation, the exploration is not prefered. To suppres the exploration and to boost the tree gowing along a predefined path, a method called RRT-Path was proposed in [39]. This method use a precomputed guideline and the tree is grown near this path. As a initialization guideline for the tree growing can be used algorithm introduced in Sections 3.1 or 3.2 [39]. Guideline does not accomplish with dynamic constraints of robot, only must be collision free. Holonomicity of the robot is accomplished by the tree growing around this path.

Guideline is provided to the method as a sequence of points $p_i \in P$, where $i = 1, 2, \ldots, n$. Methods works as a standard RRT technique with the goal bias improvement, but instead of switching goal position q_{goal} as a random sample q_{rand}, the points p_i from path are used. Points are selected sequentially as they are reached. Figure 3.9 shows a possible situation of motion planning by this method.

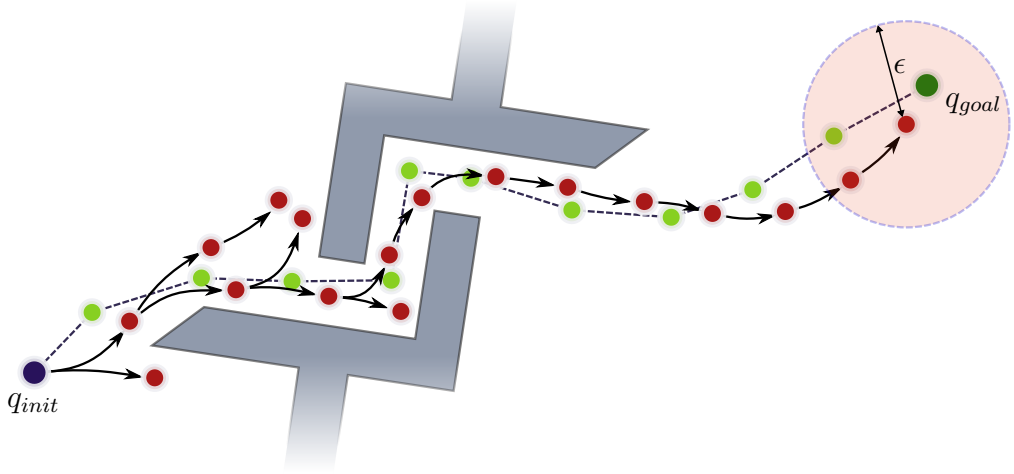


Figure 3.9: Principle scheme of method RRT Path. The tree (*red points*) is grown around guide path (*green points*).

3.4 Conclusion

In this chapter, we have presented several methods for find both path and trajectories for a mobile robot. For purpose of motion planning of mobile robot formations, where the motions of the individual robots are constrained, it seems suitable to use the RRT method.

The RRT method considers the motion model of the robots, moreover, it provides the trajectory including the control inputs. Such a trajectory can be used to directly control the robots. However, as the RRT provides path only for one robot, another approach has to be used to find trajectories also for the other robots in the formation. This is described in the next chapter.

Chapter 4

Formations of mobile robots

Formations of mobile robots are one of the hot topics in robotics. A wide variety of robotics applications, where the networks of mobile robots should be used, is studied by scientists for theirs high potential of facilitation works commonly preformed by human as well as dangerous applications too risk for human. Account of such domains can imply search and rescue missions, 3D cooperative mapping, agricultural coverage task, surveillance, security patrols or team of robotic vehicles intended to by fielded as a scout unit by the army [1]. Another applications where formations should be used are traffic control, satellite clustering or harvesting. Furthermore the dangerous applications, where robots reduce the need for human presence, such as the cleanup of toxic waste, nuclear power plant decommissioning, extra-planetary exploration [28].

Certain complex environments of missions may require a different robotic capabilities. In this applications, a mixture of heterogeneous robots can actually be easier or cheaper. An article [12] describes how the cybernetics science and robots are used for this form of amusement. Using of formations in this area doesn't require special capabilities of robots, therefore their deployment should be easy.

The shape of formation often plays its role during process of investigation. For this purpose researches frequently consider four shapes of formation, namely *line*, *column*, *diamond*, and *wedge*, where the robots travel in shape of latter "V". These shapes are used especially for their strategical appearance in the previously mentioned applications. The shape also plays role during investigations of different methods for controlling and stabilization of the formations. For this purpose, several approaches with different principle were invented. A division can be provided into two groups according to relation between the individual robots. The first set of methods can be called centralized, which means that one robot is denoted as leader used for driving whole formation to the target region. Other robots only follow the leader according to the relative position to the leader. Methods that use this scenario are *Behavior-based* method, *Neural Networks* and *Leader-*

Follower concept which is used for formation control in this work. The second set of methods are decentralized, where the robots work individually as pursuing to hold a predefined position in formation. Approaches using this behavior are *Virtual Structure* and *Potential field* method. The advantage of this approach is their independence on robot leader that can fails and thus the operation success of whole formation is aborted.

Firstly mentioned **Behavior-based** method is inspired from formations behavior in the nature, like flocking and schooling of animals [1]. Animals living in groups have better chance to forage a food and to maximize the chance of detecting predators by combination of their sensors. This inspiration could similarly benefit in robotic research. Simple behaviors like collision avoidance, aggregation and dispersion can be combined to create a flocking behavior for group of mobile robots. Position of the robots in the formation could be maintained relative to a leader or their neighbors. Therefore, the method is on the edge between decentralized and centralized behavior of the system.

Potential field is well known motion planning method for mobile robots. An enhancement for motion planning of mobile robots formations was presented in [2]. The approach is inspired by the way molecules "snap" into place as they form crystals. This is provided by attachment site of each robot which can attract other robots in the formation. The collision avoidance is kept as a repulsion energy around the obstacles. For motion planning of whole formation is used *social potential*, a potential function that respects other robots in the formation.

Another decentralized method is the **Virtual structure**, where the robots are forming in a virtual rigid structure. Robot's position in the structure is relative to the body of the virtual structure and the points representing robot position in structure are fixed [35]. Therefore the method does not require leader selection. If the virtual body is moving, points keep their position respect to the reference frame. In [35], the Virtual Structure (VS) is defined as a collection of elements (robots) which maintain a (semi) rigid geometric relationship to each other and to a frame of reference. Robots formed in VS with their position in space still have some DOFs in varying their orientations. For motion of a whole formation, first the virtual body is created — the positions of the robots withing the structure are defined. This step is followed by moving with virtual structure to a new desired position. After that the fitting each robot to a desired point in VS is computed and finally the motion with forward velocity and angular velocity computed from desired trajectories is realized. These steps are provided in loop. The main advantage of this approach is that robots precisely keep the formation.

A method that needs the selection of a leader robot is based on the **Neural Networks** principle [11]. The leader is a key robot and only one which knows the path to the goal. Other robots track the leader and for maintaining the formation

a neural network is used. In the formation, each robot is distinguished by different color and equipped with a camera for tracking the other robots. Every robot has its color unique and thus the method is inappropriately for formations with large amount of entities. The method is neither proper for miscellaneous shapes of formation.

Finally, a leader robot is used in the **Leader-Follower** concept. Here, the leader moves along its trajectory toward the goal position. The leader is tracked by the followers, which try to maintain their position in formation relatively to the leader [6]. The advantage of this approach is the capability of forming miscellaneous shapes of formation. Moreover, it can be mathematically proved, that the formations controlled by the Leader-Follower approach will converge to the target region [31].

This is why we chose this method for controlling the motion of the formation. This concept will used for a static formation in the following Section 4.1. Last Section 4.3 extends the concept of robotic formation with drones.

4.1 Static formations

The Leader-follower concept was chosen as an appropriate method for formation driving. The previously presented sampling based motion planning methods (Chapter 3) can be used with an advantage for the trajectory initialization. This concept is also suitable for optimization of motion which will be used with advantage in next Chapter 5.

The presented description of the formation is based on works [3, 4, 29, 31]. All of these works use the same model of motion like the one described in Section 2.2.1.

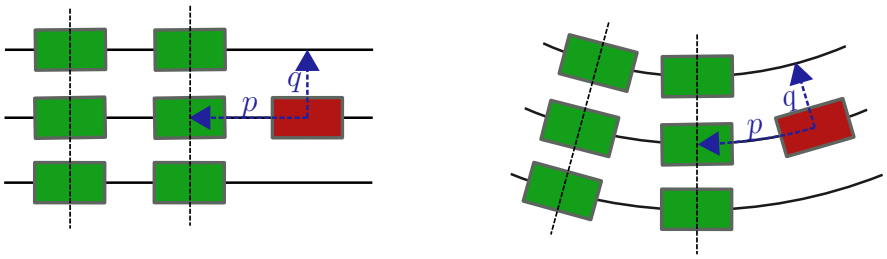
The formation consists of several robots $\mathcal{A}_i$, where $i \in \{1, \ldots, n_r, L\}$ (the letter L signs the leader robot). Here, it is assumed that a trajectory $T = [x_L(t), y_L(t), \phi_L(t), v_L(t), K_L(t)]$ of the leader robot is known, where x_L, y_L, ϕ_L denote the position of the leader and $v_L(t), K_L(t)$ are the control inputs. The trajectory is parametrized by time t, where the control inputs changes. But instead of using the time is more convenient to use the control as a function of distance $d_L(t)$. This can be simply provided by velocity integration over the time:

$$d_L(t) = \int_0^t v_L(\tau) d\tau. \tag{4.1}$$

Consequently, the geometry of the trajectory can be simply computed independently on the velocity.

A shape of the formation is described by vector $\begin{bmatrix} p_i & q_i \end{bmatrix}^T$ in curvilinear system and their meaning can be seen in Figure 4.1. Reason for using the curvilinear

coordinate system rather than rectilinear coordinate system is based on accommodation the nonholonomic constrain of the robots tracking the leaders trajectory [3].



(a) $K_L = 0$ (b) $K_L \neq 0$

Figure 4.1: Parameters of formation (a). Formation while turning (b). The position of followers (*green*) is defined relative to the leader (*red*).

For a static formations, the parameters p_i and q_i are constants. A leaders trajectory is defined as sequence of control inputs, therefore control inputs for the individual followers can be easily computed as

$$v_i(s_i) = v_L(d_L)\,(1 - q_i K_L(s_i)) \tag{4.2}$$

$$K_i(s_i) = \frac{K_L(s_i)}{1 - q_i K_L(s_i)}, \tag{4.3}$$

where $s_i = d_L + p_i$. This, together with the equation (2.7) defines a motion model for the followers.

The motion of the robots is constrained, hence the control inputs are valid under conditions:

$$\begin{aligned} v_{i,min} \leq v_i \leq v_{i,max} \\ K_{i,min} \leq K_i \leq K_{i,max} \end{aligned} \tag{4.4}$$

Whenever the formation is composed from heterogeneous robots with different motion constraints, the leader must consider all of the constraints. The set of constraints of the leader's trajectory is:

$$
\begin{aligned}
K_{L,min} &= \max_{i=1,\dots,n_r,L} \left(\frac{K_{i,min}}{1 + q_i K_{i,min}} \right) \\
K_{L,max} &= \min_{i=1,\dots,n_r,L} \left(\frac{K_{i,max}}{1 + q_i K_{i,max}} \right) \\
v_{L,min} &= \max_{i=1,\dots,n_r,L} \left(\frac{v_{i,min}}{1 + q_i K_L} \right) \\
v_{L,max} &= \min_{i=1,\dots,n_r,L} \left(\frac{v_{i,max}}{1 + q_i K_L} \right)
\end{aligned}
\tag{4.5}
$$

To compute a position of a follower in the rectangular coordinate system (global position), the following equation can be used:

$$
\begin{aligned}
x_i &= x_L(s_i) - q_i(s_i)\sin(\phi_L(s_i)), \\
y_i &= y_L(s_i) + q_i(s_i)\cos(\phi_L(s_i)), \\
\phi_i &= \phi_L(s_i).
\end{aligned}
\tag{4.6}
$$

4.2 Dynamic formations

An environment often contains narrow passages too strait for a whole formation. Therefore, a dynamic behavior of the formation is required. Its means, that the formation can change the shape during a process of going through the narrow passage.

Opposite the static formation in dynamic formation, the parameters of the formation function are function of time $\begin{bmatrix} p_i(t) & q_i(t) \end{bmatrix}^T$, which means that the shape of formation can be changed for each robot individually. As for the static formation can be time recomputed to distance by equation (4.1). Then the position of individual follower can be expressed as

$$
s_i(t) = d_L(t) + p_i(t) \tag{4.7}
$$

where $p_i(t)$ can change through the time now.

Before recomputing trajectory for every i-th follower is necessary to determine all of the following quantities

$$
s_i(t), \quad q_i(s_i), \quad \frac{dq_i}{ds_i}(s_i), \quad \frac{d^2 q_i}{ds_i^2}(s_i) \tag{4.8}
$$

Therefore the maneuver should be relatively simple, to easy determining these quantities. The control inputs for follower then can be computed by following

equations

$$
\begin{aligned}
v_i &= SQv_L(d_L) \\
K_i &= \frac{S}{Q}\left(K_L + \frac{(1-q_iK_L)\dfrac{d^2q_i}{ds_i^2} + K_L\left(\dfrac{dq_i}{ds_i}\right)^2}{Q^2}\right)
\end{aligned}
\tag{4.9}
$$

with

$$S = \text{sign}(1 - q_iK_L) \tag{4.10}$$

$$Q = \sqrt{\left(\frac{dq_i}{ds_i}\right)^2 + (1-q_iK_L)^2} \tag{4.11}$$

When the reference curvature is K_L is zero, the equation (4.9) should be simplify to a form

$$K_i = \frac{\dfrac{d^2q_i}{ds_i^2}}{\left(1 + \left(\dfrac{dq_i}{ds_i}\right)^2\right)^{\frac{3}{2}}} \tag{4.12}$$

These equation determines that the maneuver will be continuous, but also the limitation of control (4.2) inputs must be observe.

4.3 Drone extension

Extending the formations driven by the Leader-Follower concept about UAVs (Unmanned aerial vehicles) like quadrocopters, is relatively simple. First, the motion model of the drone must be defined, but for simplicity the model of motion can be same as for car-like robot (2.7), only the variable z should be added as

$$\dot{z}(t) = l(t) \tag{4.13}$$

where l means lift. Further, the boundaries value for optimization should be added

$$l_{min} \leq l(t) \leq l_{max} \tag{4.14}$$

Another parameter of this extension is relative position to leader described by a vector $\begin{bmatrix} p_i & q_i & h_i \end{bmatrix}^T$. The $h_i = 0$ for all ground robots; $h_i > 0$ for the drones. Detail description of parameters should be seen on Figure 4.2.

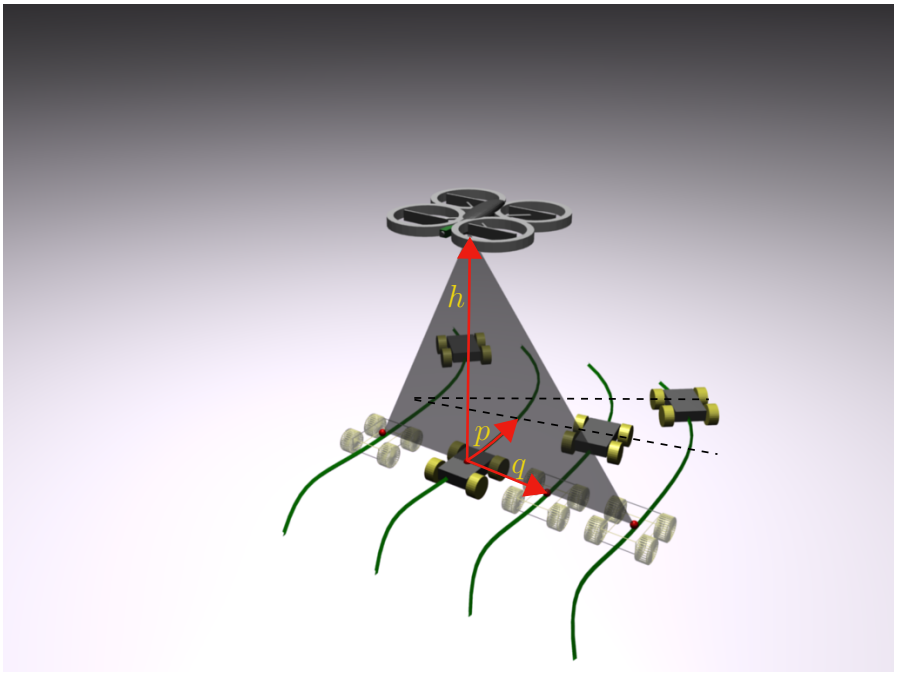


Figure 4.2: Formation with drone

Chapter 5

Optimization

Almost every motion planning process needs additional optimization to achieve a motion that is less energetically consuming than the originally planned. The optimization of the planned trajectory is provided from several aspects as a reduction of the time spent by a motion or assurance of the continual motion without unnecessary breaking. Another criterion of the optimization could be a length of the planned path or a distance to obstacles. A combination of more or less criteria together during the process of enhancing is commonly used and thus the optimized trajectory is smoother and preferable for a controller treatment.

When the motion planning is provided for nonholonomic robot, the optimized trajectory must accomplish particular constraints of motion (for example the car-like robot can not turn on the spot). But how the number of constraints and other smoothing criteria grow, the speed of optimization decreases which is counterproductive especially at applications with real robots. Suitable selection of the initialization method, which computes the initial solution, can speed up the optimization. Therefore, the classical path planning approaches, presented in 3.1 are not proper as they not consider the motion model of robots and they do not provide control inputs for the trajectory.

For this reason, the Rapidly Exploring Random Tree (RRT) method, presented in section 3.3 is used for the initialization. The RRT method can quickly find a feasible solution, either in difficult environments (containing narrow passages) or for robots with many DOFs. Trajectories returned from sampling-based planners are not optimal, however they returns feasible solution as a sequence of control inputs. Therefore the feasibility of initialized solution should significantly boost a speed-rate of optimization method.

A good choice of the optimization method and proper combination with the motion planning initialization method is consequently a key part of design robot motion process. Therefore the advantages of combination of RRT with Model Predictive Control (MPC) optimization method is presented in this thesis. As

the optimization part of the MPC, the Sequential Quadratic Programming SQP is used.

The Section 5.1 brings introduction to Model Predictive Control, a method used for trajectory optimization with receding horizon. This approach use for optimization a Sequential Quadratic Programming method, introduced in Section 5.2. But this method is time consuming and selecting a suitable library can significantly speedup optimization loop. Therefore the Subsection 5.2.1 introduce the CFSQP library.

5.1 Model Predictive Control

Model Predictive Control (MPC) is an optimization technique, which computes trajectory starting from actual position of robots over the time $(t_0, t_0 + N\Delta t)$, where N denotes a control horizon and Δt is a time between changing control inputs [19]. Sometimes, the method can be found under name Receding Horizon Control (RHC). The stability of the method using MPC is guaranteed by adding a Lyapunov function to the cost function as the terminal-state penalty [31]. Receding step is portion of computed control applied on the interval $(t_0, t_0 + n\Delta t)$, where the parameter n is the number of transition points applied in one receding step. Process is then repeated on the interval $(t_0 + n\Delta t, t_0 + N\Delta t + n\Delta t)$ as the finite horizon moves by time steps $n\Delta t$. As the optimization process is called in a loop, it allows to react to new situation, like the presence of new obstacles. Appropriate optimization technique is Sequential Quadratic Programming for nonholonomic robots presented in following section 5.2.

5.2 Sequential Quadratic Programming

One of the most effective numerical optimization method used for nonlinear constrained optimization is Sequential Quadratic Programming (SQP). This approach generate steps by solving quadratic subproblems [27]. SQP approach is in a way the generalization of Newton's method for solving Nonlinear Optimization Problem (NLP). One disability of SQP is absented ability to overcome local extremes in the cost function [31].

Planning loop needs to be quick as possible to respond for dynamic changes in environment. The SQP approach can be time consuming and selection of suitable implementation is important. In this thesis, we employ the CFSQP library [26].

5.2.1 CFSQP

For Sequential Quadratic Programming optimization the CFSQP (*C code for Feasible Sequential Quadratic Programming*) library [26] was used. Advantage in using this library is that the objective and constrain function can be defined by the user.

The solver provides minimization of a general functions:

$$\text{minimize} \max_{i \in I^f} \{f_i(x)\} \tag{5.1}$$

where $I^f = \{1, \ldots, n_f\}$ ($I^f = \emptyset$ if $n_f = 0$) and X is the set of points $x \in \mathbb{R}^n$ satisfying

$$\begin{aligned}
bl \leq x \leq bu & \\
g_j(x) \leq 0, & \qquad j = 1, \ldots, n_i \\
g_j(x) \equiv \langle c_{j-n_i}, x \rangle - d_{j-n_i} \leq 0, & \qquad j = n_i + 1, \ldots, t_i \\
h_j(x) = 0, & \qquad j = 1, \ldots, n_i \\
h_j(x) \equiv \langle a_{j-n_e}, x \rangle - b_{j-n_e} = 0, & \qquad j = n_e + 1, \ldots, t_e
\end{aligned}$$

where $bl \in \mathbb{R}^n$ and $bu \in \mathbb{R}^n$ define the lower and upper bounds for the variables, $f_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, n_f$ is a smooth function to be optimized, $g_j : \mathbb{R}^n \to \mathbb{R}, j = 1, \ldots, n_i$ are nonlinear and smooth inequality constraint functions, $c_j \in \mathbb{R}^n, d_j \in \mathbb{R}, j = 1, \ldots, t_i - n_i$; $h_j : \mathbb{R}_n \to \mathbb{R}, j = 1, \ldots, n_e$ are nonlinear and smooth equality constraint functions and $a_j \in \mathbb{R}^n, b_j \in \mathbb{R}, j = 1, \ldots, t_e - n_e$.

The CFSQP can cope with problems including sequentially related objectives and constraints, but this advantage is not needed for an optimization in these work.

5.3 Objective function and constraints

The optimization of trajectories for a formation of mobile robots is divided into two parts: a) the global trajectory is optimized for the leader in such way, that the whole formation can proceeding, b) followers' trajectories derived from leader's one are optimized within the short horizon. Both process of optimization need different cost functions under different constraints.

For the purpose of optimization, the leader's trajectory is described only by the control inputs. Let Ω_L denote the optimization vector, then $\Omega_L = (v_1, K_1, t_1, v_2, K_2, t_2, \ldots, v_N, K_N, t_N, \ldots, v_M, K_M, t_M)$, where M is the length of the trajectory and $(t_1, \ldots, t_N)$ has constant time interval Δt. Other parts of this long vector are variable.

5.3.1 Leader

The total time of the robot between actual position and target region is

$$t_f - t = N\Delta t + \sum_{k=N+1}^{M} \Delta t(k) \tag{5.2}$$

The cost function of the leader's trajectory is given by equation

$$J_L(\Omega_L, \mathcal{O}) = \sum_{k=N+1}^{M} \Delta t(k) + \alpha \sum_{j=1}^{n_0} \left(\min \left\{ 0, \frac{\delta_j(\Omega_L, \mathcal{O}) - r_{s,L}}{\delta_j(\Omega_L, \mathcal{O}) - r_{a,L}} \right\} \right)^2 \tag{5.3}$$

where $\delta_j(\Omega, \mathcal{O})$ is the shortest distance between the trajectory Ω and the obstacle j and n_0 is the number of obstacles. The main course of the optimization of the motion is to reach a desired goal region as soon as possible. This effort is expressed in the first part of cost function. How the final solution is influenced by the environment describe a second part of this function. The constant α estimates, how much the second sum will be influencing the value of the cost function. A higher α causes a longer trajectory with larger distance from obstacles. If the distance to obstacles is large enough, then the cost function is not affected by this sum. This behavior is produced by fraction, where the ration between safety radius r_s and collision avoidance radius r_a is computed.

If the precise final position is required, the another criterion to the cost function (5.3) can be added

$$\beta ||p_{d,L}(M) - p_L(M)||^2. \tag{5.4}$$

Again, the constant β says, how much this part influences the total value of the cost function. An increase of β signifies that the accurate position is required. On the other hand, if the achieving of target region is only needed the β constant should be near to zero.

To distinguish between feasible and infeasible trajectories, additional constrains have to be considered. The first one is the avoidance inequality constraint characterizing collision avoidance criterion in environment.

$$g_{r_{a,L}}(\Omega_L, \mathcal{O}) = r_{a,L}^2 - \delta_j^2(\Omega_L, \mathcal{O}), \quad j \in \{1, \ldots, n_0\} \tag{5.5}$$

This constraint work with static obstacles as well as with dynamic obstacles. Second constraint ensures that the leader will enter the circular target region with center C_{S_F} and radius r_{S_F}.

$$g_{S_F}(\psi_L(M)) = ||p_L(M) - C_{S_F}|| - r_{S_F} \tag{5.6}$$

Into the process of optimization the boundaries as maximal and minimal curvature, maximal and minimal velocity and also minimal time step between states must be considered.

$$b_l \equiv \begin{bmatrix} v_{min} \\ K_{min} \\ \Delta t_{min} \end{bmatrix} \leq \begin{bmatrix} v(k) \\ K(k) \\ \Delta t(k) \end{bmatrix} \leq \begin{bmatrix} v_{max} \\ K_{max} \\ \Delta t_{max} \end{bmatrix} \equiv b_u \tag{5.7}$$

Boundary values are derived from kinematic constraints of formation as it is shown in Section 4.1.

The optimization problem of finding trajectory Ω_L for the leader is then defined using objective function (5.3) and constraints (5.5) and (5.6).

5.3.2 Follower

After the leader's trajectory is optimized, the trajectories for the individual followers are optimized to. The cost function of the follower has a little different composition as the whole trajectory is not considered.

$$\begin{aligned} J_i(\Omega_i) = \sum_{k=1}^{N} ||(p_{d,i}(k) - p_i(k))||^2 + \alpha \sum_{j=0}^{n_0} \left(\min \left\{ 0, \frac{\delta_j(\Omega_i, \mathcal{O}) - r_s}{\delta_j(\Omega_i, \mathcal{O}) - r_a} \right\} \right)^2 \\ + \beta \sum_{j \in n_n} \left(\min \left\{ 0, \frac{d_{i,j}(\Omega_i, \Omega_j^\circ) - rs, i}{d_{i,j}(\Omega_i, \Omega_j^\circ) - r_{a,i}} \right\} \right)^2 \end{aligned} \tag{5.8}$$

where i is the number of the follower, $\delta_j(\Omega, \mathcal{O})$ is the shortest distance between the follower's trajectory and j-th obstacle, $\delta_{i,j}(\Omega_i, \Omega_j)$ is the shorest distance between the i-th follower and j-th follower moving on their trajectories. The first sum describes a distance from desired position, the second sum describes distance to the obstacles and the third sum is for collision avoidance between individual robots.

The first inequality constraint is identical with the leader's one:

$$g_{r_a}(\Omega_i, \mathcal{O}) = r_a^2 - \delta(\Omega_i, \mathcal{O})^2, \quad j \in \{1, \ldots, n_0\} \tag{5.9}$$

The second constraint is used for the collision avoidance between the individual robots and obstacles, the static one as well as the dynamic one.

$$g_{r_{a,i}}(\Omega_i, \Omega_j^\circ) = r_{a,i}^2 - d_{i,j}(\Omega_i, \Omega_j^\circ)^2, \quad j \in n_n \tag{5.10}$$

The optimization problem for the followers is then defined: find the trajectory Ω_i maximizing the objective function (5.8) subject to constraints (5.9) and (5.10).

Chapter 6

Examples

The system for motion planning of the formations of mobile robots presented in the previous chapters has been implemented into a framework. The implemented library allows to define new motion models of robots, as well as various optimization criteria. The library is provided on the attached CD.

In this section, we will show screenshots from the planning process. The initial position in the map is always $q_{init} = (0.5, 1, 0)$ and a goal region is represented by a red circle. The obstacles are highlighted by gray color. The motion planning was provided in several tasks with different composition of obstacles. Every Figure 6.1–6.7 show single independent run.

The Figure 6.1 shows how the tree with 0.8 s long edge looks like. The tree is grown quickly and output trajectory is composed from several edges, therefore the optimization could provide satisfying solution in short time.

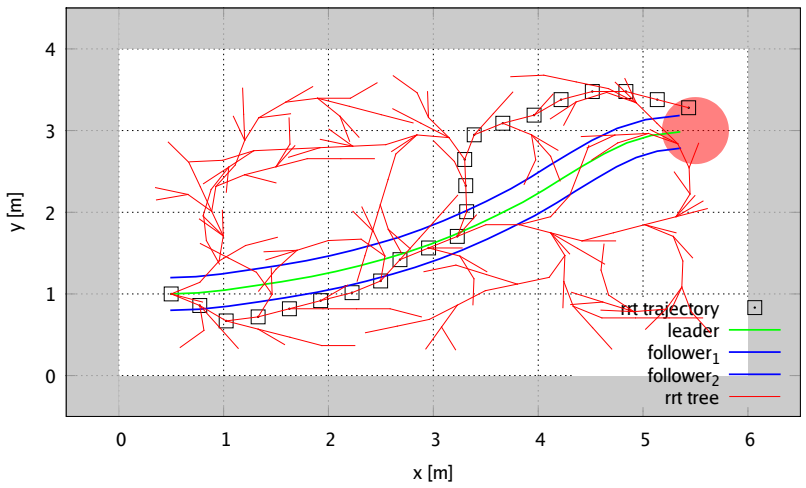


Figure 6.1: Example of RRT tree with 0.8 s long edge.

Another example of the tree with 0.4 s long edge could be seen on Figure 6.2. The searching of the environment is provided more precisely and output trajectory contains more segments. Therefore the optimizer could spend amount of time by smoothing. In comparison with the Figure 6.1, the trajectory is more complicated.

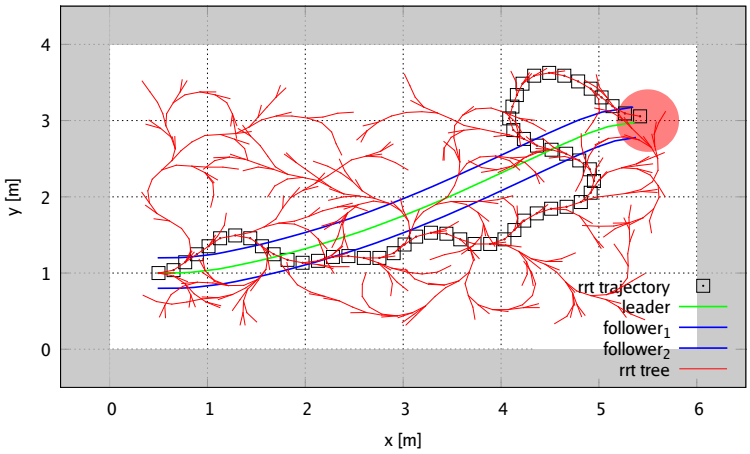


Figure 6.2: Example of RRT tree with 0.4 s long edge.

The example where the motion planning is provided in environment containing only border obstacle could be seen on Figure 6.3. The illustration shows that even non-optimal RRT trajectory could be optimized to a suitable form.

A more complex environments are figured on 6.5 and 6.6. The formation needs to perform a maneuver around obstacles in the middle of the space. This is provided with minimal energetic cost. The cost function of optimization method ensure that obstacles will be avoided with enough security distance.

The Figure 6.6 shows solution in a map containing a narrow passage. Here, the RRT is the most proper solution for initialization, because initialized trajectory is suitable even in narrow passage.

Finally, the example of motion planning for a heterogeneous formation consisting of three ground vehicles and one UAV is shown on Figure 6.7. The drone is directly up to leader robot, and from top view their trajectories are identical. Therefore, it can be seen that the drone needs to execute maneuver where the obstacle is flown under.

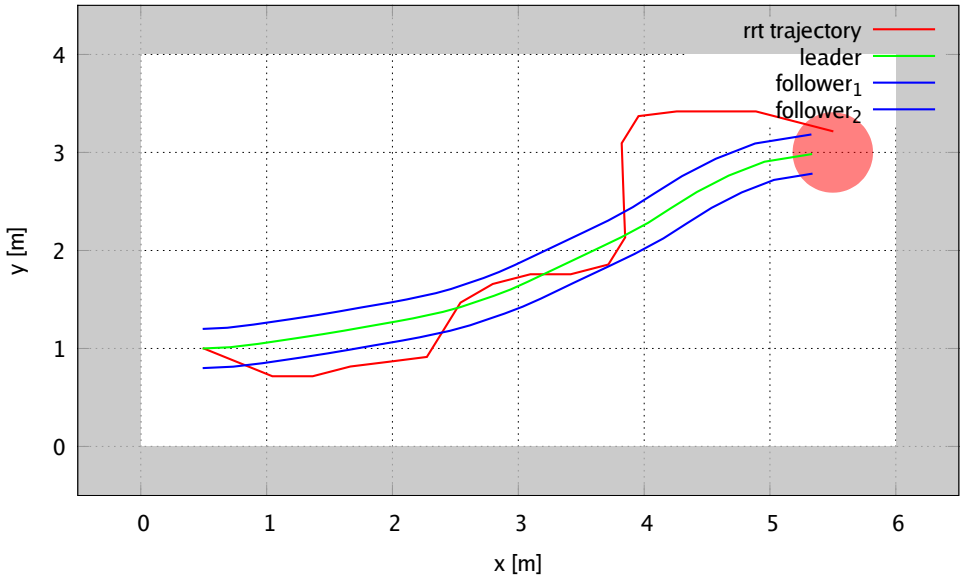


Figure 6.3: Environment without obstacles.

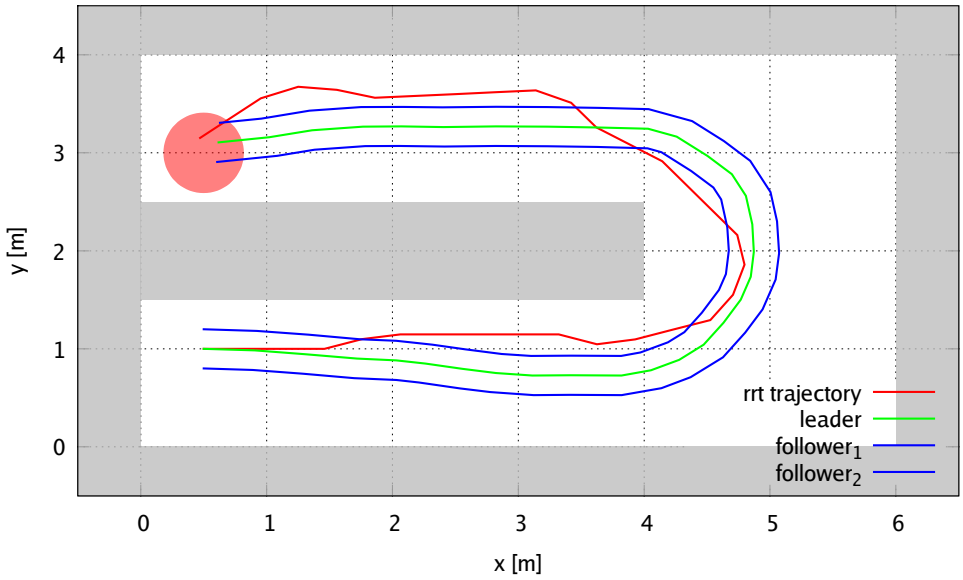


Figure 6.4: Maneuver in corridor

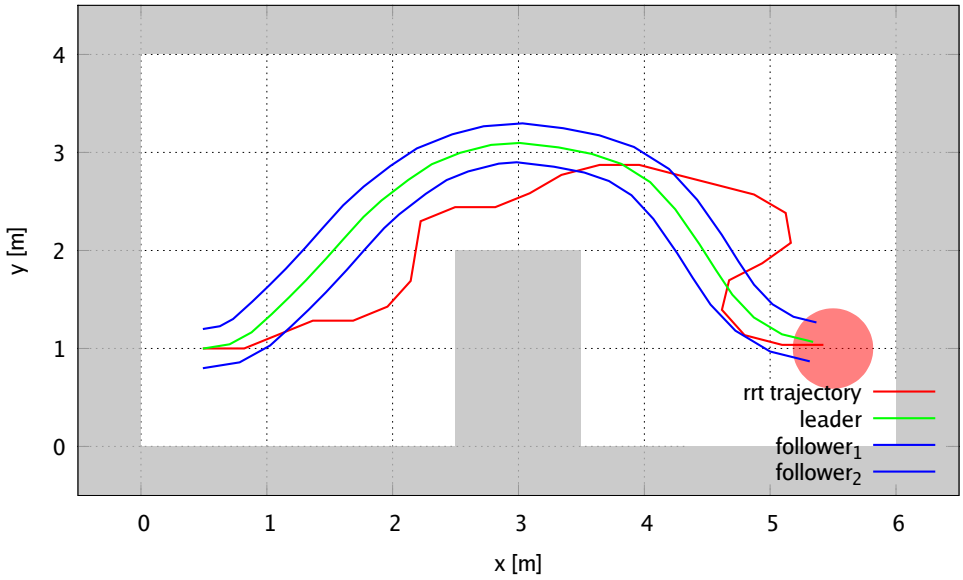


Figure 6.5: Obstacle avoidance

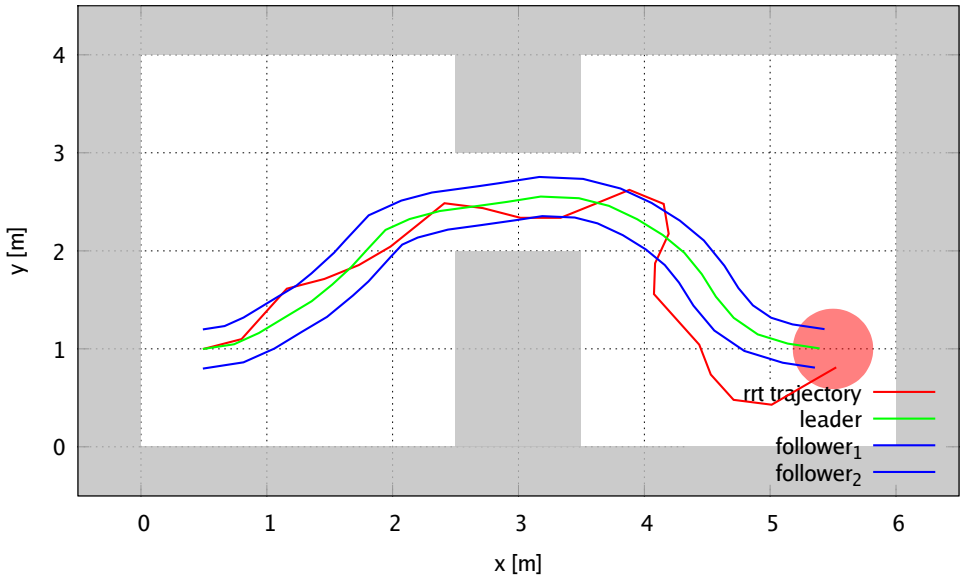


Figure 6.6: The narrow passage problem.

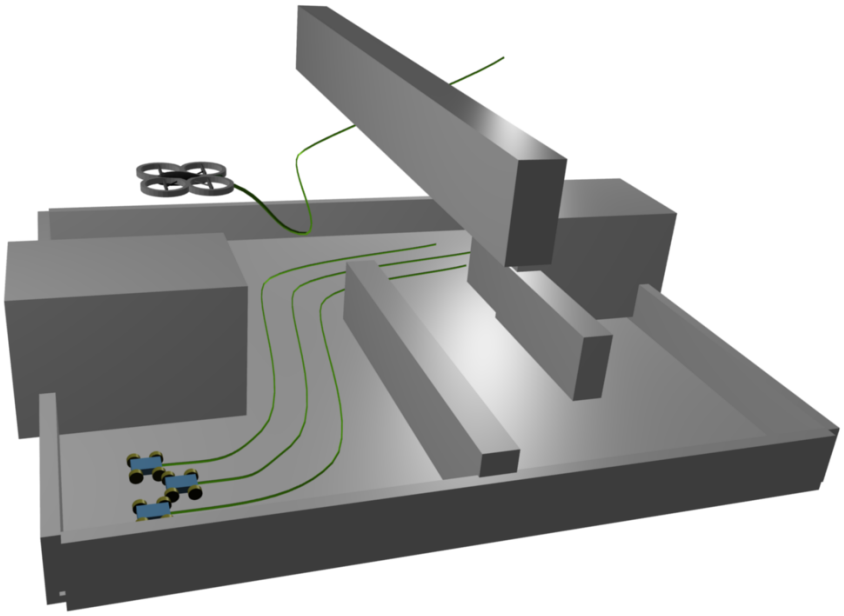


Figure 6.7: 3D formation including drone.

Chapter 7

Conclusion

The presented thesis deals with motion planning of formations of mobile robots. To find a motion for the formations, a Leader-Follower approach has been used. In this approach, the trajectory of the formations is found only based on know trajectory of the leader. It is thus suitable to plan the motion only for the leader robot, which allows to employ wide variety of methods for solving the problem.

To find optimal trajectories for the robots in the formation, the Model Predictive Control Approach has been employed. In this approach, the initial trajectory is optimized and the robots move along it for a predefine time. After that, the method computes the new optimization, possibly considering new situations, like newly detected obstacles.

However, due to high computational burden of the used optimization method, it is suitable to provide the MPC feasible trajectories. For this task, we use the Rapidly Exploring Random Tree method. The method provides both positions and control inputs of the trajectory. Moreover, it considers the motion model of the robot, the provided trajectory is thus always feasible.

The main results of this thesis is the software library providing MPC control of the formations. For this purpose of optimization the SQP library was need to choose. As a suitable, the CFSQP library was selected, because the user can define objective and constrain functions. The library further includes implementation of the RRT algorithm, which is used for initialization of MPC. The library is designed to solve both motion planning for homogeneous formations (all the robots are same) and for heterogeneous formations (e.g., mobile robots with drones). The library has been used during last two semesters in course A3M99PTO and for several bachelor theses.

Future Work

The framework is prepared for adding another motion planning techniques and motion models. This allows to provide lots of different experiments where the initialization techniques for different types of robot can bring interest conclusions. In the future we would like expand this framework to real targets, to ensure the suitability of selected approaches in real environment.

Bibliography

[1] Tucker Balch and Ronald C. Arkin. Behavior-based formation control for multirobot teams. *Robotics and Automation, IEEE Transactions on*, 14(6):926–939, December 1998.

[2] Tucker Balch and Maria Hybinette. Social potentials for scalable multi-robot formations. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 73–80, April 2000.

[3] Timothy D. Barfoot and Christopher M. Clark. Motion planning for formations of mobile robots. *Robotics and Autonomous System*, 46(2):65–78, February 2004.

[4] Timothy D. Barfoot, Christopher M. Clark, Stephen M. Rock, and Gabriel M. T. D'Eleuterio. Kinematic path-planning for formations of mobile robots with a nonholonomic constraint. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2819–2824, October 2002.

[5] Valérie Boor, Mark H. Overmars, and Frank van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1018–1023, 1999.

[6] Rafael Fierro, Aveek K. Das, Vijay Kumar, and James P. Ostrowski. Hybrid control of formations of robots. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 157–162, May 2001.

[7] Emilio Frazzoli, Munther A. Dahleh, and Eric Feron. Real-time motion planning for agile autonomous vehicles. *AIAA JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS*, 25:116–129, August 2000.

[8] Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. OBBTree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*,

pages 171–180, New York, NY, USA, 1996. ACM. Also available at: http://gamma.cs.unc.edu/OBB/.

[9] Martin Hess, Martin Saska, and Klaus Schilling. Autonomous multi-vehicle formations for cooperative airfield snow shoveling. In *Proceedings of the 3rd European Conference on Mobile Robots (ECMR 07), Freiburg, Germany*, 2007.

[10] Martin Hess, Martin Saska, and Klaus Schilling. Application of coordination multi-vehicle formations for snow shoveling on airports. *Intelligent Service Robotics*, 2(4):205–217, 2009.

[11] Kaoru Hirota, Tsuyoshi Kuwabara, Kenichi Ishida, Akihiko Miyanohara, Hiroaki Ohdachi, Toshihiro Ohsawa, Wataru Takeuchi, Naoyoshi Yubazaki, and Masayuki Ohtani. Robots moving in formation by using neural network and radial basis functions. In *Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Proceedings of 1995 IEEE International Conference on*, volume 5, pages 91–94, March 1995.

[12] Jana Horáková. Robotické performance – Divadlo toužících strojů. *disk – Časopis pro studim dramatického umění*, 12:107–115, June 2005. in Czech.

[13] Maciej Kalisiak. Thesis proposal: Kinodynamic motion planning with viability models. Technical report, University of Toronto, 2006.

[14] Maciej Kalisiak. *Toward More Efficient Motion Planning with Differential Constraints*. PhD thesis, University of Toronto, 2007.

[15] Maciej Kalisiak and Michiel van de Panne. Rrt-blossom: Rrt with a local flood-fill behavior. In *ICRA*, pages 1237–1242. IEEE, 2006.

[16] Lydia E. Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, August 1996.

[17] James J. Kuffner and Steven M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 995–1001, 2000.

[18] Yoshiaki Kuwata, Alberto Elfs, Mark Maimone, Andrew Howard, Mihail Pivtorai, Thomas M. Howard, and Adrian Stoica. Path planning challenges for planetary robots. 2008.

[19] Yoshiaki. Kuwata and Jonathan P. How. Stable trajectory design for highly constrained environments using receding horizon control. In *American Control Conference, 2004. Proceedings of the 2004*, volume 1, pages 902–907. IEEE, Juny 2004.

[20] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Dept., Iowa State University, oct 1998.

[21] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[22] Steven M. LaValle. Motion planning. *Robotics Automation Magazine, IEEE*, 18(1):79–89, March 2011.

[23] Steven M. LaValle. Motion planning. *Robotics Automation Magazine, IEEE*, 18(2):108–118, June 2011.

[24] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 473–479, 1999.

[25] Steven M. LaValle and James J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000.

[26] Craig Lawrence, Jian L Zhou, and Andre L Tits. *User's Guide for CFSQP Version 2.5: A C Code for Solving Large Scale Constrained Nonlinear Minimax Optimization Problems , Generating Iterates Satisfying All Inequality Constraints*, volume 20742. 1997.

[27] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*, chapter Sequential Quadratic Programming, pages 526–573. 3. Springer, Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY 10010, USA, 1999.

[28] Lynne E. Parker. *Heterogeneous Multi-Robot Cooperation*. PhD thesis, Massachusetts Institute of Technology, February 1994.

[29] Martin Saska. *Trajectory planning and optimal control for formations of autonomous robots*. PhD thesis, Julius-Miximilian-Universität Würzburg, Am Hubland, 97074 Würzburg, November 2009.

[30] Martin Saska, Juan Mejía, Dušan Stipanović, and Klaus Schilling. Control and navigation of formations of car-like robots on a receding horizon. In *3rd*

IEEE Multi-conference on Systems and Control (MSC 2009). Saint Petersburg,Russia, pages 1761–1766, July 2009.

[31] Martin Saska, Vojtěch Vonásek, and Libor Přeučil. Control of ad-hoc formations for autonomous airport snow shoveling. In *IROS*, pages 4995–5000. IEEE, 2010.

[32] Martin Saska, Vojtěch Vonásek, and Libor Přeučil. Roads sweeping by unmanned multi-vehicle and automation. In *ICRA*, pages 631–636. IEEE, 2011.

[33] K. Schilling and F. Driewer. Remote control of mobile robots for emergencies. In *16th IFAC World Congress, Prague, Czech Republic*, 2005.

[34] Zheng Sun, David Hsu, Tingting Jiang, Hanna Kurniawati, and John H. Reif. Narrow passage sampling for probabilistic roadmap planning. *Robotics, IEEE Transactions on*, 21(6):1105–1115, December 2005.

[35] Kar-Han Tan and M. Anthony Lewis. Virtual structures for high-precision cooperative mobile robotic control. In *Proceedings of the 1996 IEEE/RSH Conference on Intelligent Robots and Systems*, volume 1, pages 132–139, Osaka, 1996. IEEE.

[36] Herbert G. Tanner, Sawas G. Loizou, and Kostas J. Kyriakopoulos. Nonholonomic navigation and control of cooperating mobile manipulators. *Robotics and Automation, IEEE Transactions on*, 19(1):53–64, February 2003.

[37] Gino van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal Graphics Tools*, 2(4):1–13, 1997.

[38] Petr Vaněk. The RRT motion planning techniques. Bachelor's thesis, Czech Technical University in Prague, Czech Republic, 2010. in Czech.

[39] Vojtěch Vonásek, Jan Faigl, Tomáš Krajník, and Libor Přeučil. RRT-Path: a guided Rapidly exploring Random Tree. In *Robot Motion and Control 2009*, pages 307–316, Heidelberg, 2009. Springer.

[40] Kwangjin Yang and Salah Sukkarieh. 3D-smooth path planning for a UAV in cluttered natural environments. In *IROS*, pages 794–800. IEEE, September 2008.

Appendix A

CD content

The CD is attached to the printed version of this work containing the text of the Thesis in a PDF format, source codes of thesis in LaTeX format and source codes of simulator. In following table the directory structure on the CD is described.

Directory / File	Description
`thesis.pdf`	the Diploma Thesis report in PDF format
`doc`	Diploma Thesis source codes in LaTeX format
`framework`	The source code of an implemented framework without CFSQP library

Table A.1: Directory structure on the CD