České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Martin H o b z a
Studijní program:	Kybernetika a robotika (bakalářský)
Obor:	Robotika
Název tématu:	Vyhlazování trajektorií vzájemně kooperujících bezpilotních helikoptér

Pokyny pro vypracování:

Cílem práce je navrhnout, implementovat a experimentálně ověřit systém umožňující plánovat hladké trajektorie pro skupinu vzájemně lokalizovaných helikoptér.

1. Student navrhne a implementuje jednoduchou metodu pro plánování pohybu bezpilotních helikoptér v konkrétní multi-robotické aplikaci (např. [5]). Získané trajektorie nemusí být optimální, ale musí splňovat omezení daná systémem relativní vizuální lokalizace a modelem pohybu helikoptér.
2. Student se seznámí s algoritmy pro vyhlazování a optimalizaci trajektorií mobilních robotů (např. [1-4]).
3. Student navrhne a implementuje metodu pro zkrácení a vyhlazení získaných trajektorií, při zachování výše zmíněných omezení.
4. Implementovaný systém bude experimentálně ověřen sérií numerických experimentů. V případě dostupnosti robotické platformy bude následně systém otestován s reálnými helikoptérami. V opačném případě bude provedena statistická analýza spolehlivosti systému v simulátoru.

Seznam odborné literatury:

[1] Hauser, K. - Ng Thow Hing, V., Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts, IEEE International Conference on Robotics and Automation (ICRA), 2010.
[2] Hejin Yuan, A Novel Trajectory Smoothing Algorithm Based on Empirical Mode Decomposition, Fifth International Conference on Image and Graphics, 2009.
[3] Bottasso, C.L. - Leonello, D. - Savini, B., Path Planning for Autonomous Vehicles by Trajectory Smoothing Using Motion Primitives, Control Systems IEEE Transactions on Technology, vol.16, no.6, pp.1152-1168, 2008.
[4] Anderson, E.P. - Beard, R.W. - McLain, T.W., Real-time dynamic trajectory smoothing for unmanned air vehicles, IEEE Transactions on Control Systems Technology, vol.13, no.3, pp.471-477, 2005.
[5] Saska, M. - Chudoba, J. - Precil, L. - Thomas, J. - Loianno, G. - Tresnak, A - Vonasek, V. - Kumar, V., Autonomous deployment of swarms of micro-aerial vehicles in cooperative surveillance, International Conference on Unmanned Aircraft Systems (ICUAS), 2014.

Vedoucí bakalářské práce: Ing. Martin Saska, Dr. rer. nat.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

Bachelor’s Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Katedra kybernetiky

Bakalářská práce

Martin Hobza
Studijní program: Kybernetika a robotika.
Obor: Robotika.

Duben 2015
Supervisor: Ing. Martin Saska, Dr. rer. nat.

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student:	Martin H o b z a
Study programme:	Cybernetics and Robotics
Specialisation:	Robotics
Title of Bachelor Project:	Multiple Trajectory Smoothing for Teams of Closely Cooperating Micro Aerial Vehicles

Guidelines:

The main purpose of the thesis is to design, implement, and experimentally verify system that enables planning of smooth and feasible trajectories for groups of mutually localized Micro Aerial Vehicles (MAVs).

1. Student designs and implements a simple method for trajectory planning of MAVs in a specific multi-robot application (e.g. [5]). Obtained trajectories do not have to be optimal, but they must satisfy requirements given by visual relative localization and MAV motion constraints.
2. Student will learn approaches for trajectory smoothing and optimization in mobile robotics (e.g. [1-4]).
3. Student designs and implements an algorithm for smoothing and optimization of obtained trajectories, while still keeping the localization and motion constraints.
4. The implemented system will be verified in various simulations. An experiment with real MAVs will be realized or statistical analyses of algorithm reliability will be done in simulator. The decision will be taken by thesis advisor based on availability of robotic platform.

Bibliography/Sources:

[1] Hauser, K. - Ng Thow Hing, V., Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts, IEEE International Conference on Robotics and Automation (ICRA), 2010.

[2] Hejin Yuan, A Novel Trajectory Smoothing Algorithm Based on Empirical Mode Decomposition, Fifth International Conference on Image and Graphics, 2009.

[3] Bottasso, C.L. - Leonello, D. - Savini, B., Path Planning for Autonomous Vehicles by Trajectory Smoothing Using Motion Primitives, Control Systems IEEE Transactions on Technology, vol.16, no.6, pp.1152-1168, 2008.

[4] Anderson, E.P. - Beard, R.W. - McLain, T.W., Real-time dynamic trajectory smoothing for unmanned air vehicles, IEEE Transactions on Control Systems Technology, vol.13, no.3, pp.471-477, 2005.

[5] Saska, M. - Chudoba, J. - Precil, L. - Thomas, J. - Loianno, G. - Tresnak, A - Vonasek, V. - Kumar, V., Autonomous deployment of swarms of micro-aerial vehicles in cooperative surveillance, International Conference on Unmanned Aircraft Systems (ICUAS), 2014.

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

Valid until: the end of the summer semester of academic year 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic	prof. Ing. Pavel Ripka, CSc.
Head of Department	**Dean**

Prague, February 9, 2015

/ Declaration

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

V Praze dne

....

Abstrakt / Abstract

Tato práce je zaměřena na plánovací a zkracovací algoritmus produkující hladké trajektorie pro roje vzájemně kooperujících bezpilotních helikoptér podléhajících omezením na vzájemné vzdálenosti a kolize s překážkami.

Jejich úkolem je nalézt cestu prostředím s překážkami a monitorovat cílovou oblast pomocí zabudovaných kamer.

Následující postup byl inspirován několika vyhlazujícími a optimalizačními algoritmy používanými v mobilní robotice.

Algoritmus generuje trajektorii v prostředí s překážkami použitím variace rapidně rostoucího náhodného stromu.

Pro vyhlazení trajektorie na ní algoritmus opakovaně vybírá dva body a snaží se nahradit díl mezi nimi kratší, splnitelnou trajektorií.

Body trajektorie jsou následně posouvány v jejich nejbližším okolí pro nalezení lokální nejkratší splnitelné trajektorie.

Výsledky ukazují účinnost optimalizační části. Vzhledem k nedostatku funkčního vybavení bylo experimentální testování provedeno jen v robotickém simulátoru.

Klíčová slova: mikroletoun; bezpilotní letou; bakalářská práce; vzájemná lokalizace; plánování trasy; zkracování; optimalizece; rapidní náhodný strom; mobilní robotika.

This thesis considers a planning and shortcutting algorithm producing smooth trajectories for swarm of closely cooperating micro aerial vehicles subjected to proximity constraints and collision constraints.

Their task is to find a way through environment with obstacles and use their onboard cameras for surveillance of targeted zone.

The following approach takes inspiration in number of various trajectory smoothing and optimization algorithms used in mobile robotics.

The algorithm generates a trajectory in environment with obstacles using a variation of rapidly exploring random tree.

To smooth the trajectory it repeatedly picks two points on it. Then it attempts to replace the segment between these points with a shorter, feasible trajectory.

The points of the trajectory are then shifted in their proximity to find the local shortest feasible trajectory.

The results show efficiency of the optimization part. Due to lack of functioning equipment, experimental testing has been done only in robotic simulator.

Keywords: micro aerial vehicle; unmanned aerial vehicle; bachelor thesis; mutual localization; path planning; shortcutting; optimization; rapid random tree; mobile robotics.

/ Contents

Tables / Figures

Chapter 1
Introduction

Autonomous robots today are faced with many tasks such as generating safe collision free trajectories, cooperating with other autonomous robots, finding a path in unknown environment, and keeping a natural looking motion. The latter is often omited at the expense of functionality or optimality of the first three.

This paper focuses on trajectories generated with cooperative behaviour of unmaned aerial vehicles (UAVs) in mind. More specificaly multirotor helicopters, also known as quadcopter, hexacopters, octacopter, depending on their amount of rotors, or simply multicopters.

The multicopter trajectory planning problem is inherently different from the trajectory planning problem for other kinds of unmaned aerial vehicles incapable of hovering state. As those non-helicopter aerial vehicles depend on constant propulsion and wing generated upward lift to stay in the air, they have severe constraints on trajectory curvature due to their heading rate constrains as well as minimum velocity constraints.

Althought the task considered by this paper has less problems with dynamics constraints of vehicles involved, it encounters problems with their mutual localization. The helicopters use their on-board cameras to determine their position within the swarm, and thus are able to travel in a group without collision.

The goal of this thesis is to design and implement trajectory planning algorithm for a multi-robot application described at the end of chapter 2.1. Next step is to learn various trajectory smoothing algorithms. Then, with their inspiration, design and implement a method that will smooth and optimize trajectories generated by the previous algorithm.

This paper is organized as follows. First part 2.1 of this thesis takes a closer look at UAVs and their problems related to their usage in a multi-robot application described at the end of said chapter.

Following chapter 3 briefly describes a number of various trajectory smoothing algorithms. Then it presents an opinion on their relevance to the specific application from second chapter, or which parts of them might be used or adjusted for that application.

After briefly explaining these various approaches, the solution itself is presented 4. The solution was divided into three seperate parts. First part starts with explanation of the original path planning algorithm and its modifications making it usable for multi-robot application. Second part describes smoothing process inspired by preivously stated smoothing approaches. And last part explains how the obtained trajectory is optimized and what sort of demands does it put on the previous part.

The functionality of proposed algorithm has been tested on series of runs. It has also been tested in virtual simulator, which is shortly presented. The results are presented and discussed in chapter 5

Chapter 2
Micro Aerial Vehicles

Military aircraft design has recently focused on the development of UAVs[1]). This class of aircraft has been successfully adapted to perform many of the same roles as manned aircraft but at lower cost. The absence of a pilot on board eliminates many of the safety and life support requirements as well as physical limitations, and also changes the control objectives. For example, rapid acceleration or aggressive maneuvering can hinder a pilot's ability to control the aircraft but do not affect the control system of a UAV. Therefore UAVs can be utilized in many more applications than manned aircraft.

Micro aerial vehicles (MAVs) are a class of unmaned aerial vehicles limited by size, often small enough to be man-portable. MAVs are far more common in civil use, as it is much simplier for ordinary hobbist to become owner of a multicopter [2]) than to become owner of a military drone[3]).

2.1 Multicopters

A sub-class of MAVs are multi-rotor helicopters which have recently become very popular. Their use has spread from military and research purposes to civil and hobby purposes. Example of a civil use is Amazon Prime Air project [4]) currently in development, which hopes to achieve package delivery via multirotor helicopters.

The reason unmaned multicopters became the most popular of all the variety of MAVs is their simplicity and safety. Compared to winged, jet propelled drones, multicopters are much more suitable for urban environment due to their ability to hover in place and taking very sharp turns. Furthermore multicopters can also fit their blades with protective frames, which makes them suitable for inside use with minimal risk of damaging the blades or obstacles in case of collision.

However, multicopter are far less stable than their single rotor counterparts. They are so unstable, they require on-board flight controller to stay in the air. Without one, they are unflyable by a human operator. Althought this seems like a huge disadvantage, the mechanical simplicity greatly outweights all advantages that conventional helicopter has on multicopters. As nearly any home-made design is flyable with sufficient amount of rotors and properly tuned controller[5]).

[1]) PopularMechanics.com, The Future For UAVs in the U.S. Air Force `http://www.popularmechanics.com/military/a5383/4347306/`

[2]) Ebay.com, search result for `quadcopter` `http://www.ebay.com/sch/i.html?_from=R40&_trksid=p2050601.m570.l1313.TR0.TRC0.H0.Xquadcopter.TRS0&_nkw=quadcopter&_sacat=0`

[3]) telegraph.co.uk, How to buy an American military drone `http://www.telegraph.co.uk/news/worldnews/northamerica/usa/11419566/How-to-buy-an-American-military-drone.html`

[4]) Amazon.com, Amazon Prime Air `http://www.amazon.com/b?node=8037720011`

[5]) Forbes.com, Various news and articles `http://www.forbes.com/sites/quora/2013/12/23/what-makes-the-quadcopter-design-so-great-for-small-drones/`

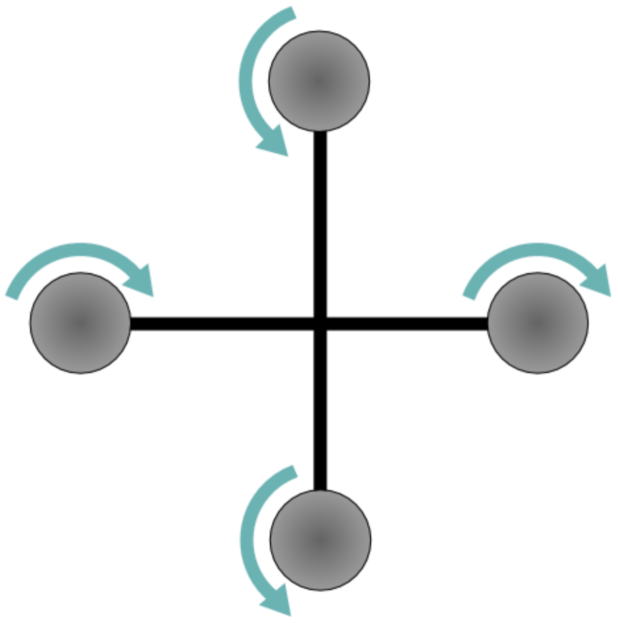

Figure 2.1. multicopter blades rotations [1])

As mentioned above, multicopters have no mechanical parts, except for the motors, and all control is done by adjusting the speed of individual blades. Each blade spins in opposite direction from the blades next to it. It turns by speeding up blades spinning in one direction and slowing down blades spinning in the opposite direction. It rolls by speeding up blades on one side and slowing down blades on the other side. Horizontal motion is accomplished by speeding up all blades while the multicopter is leaning in the direction of travel. Slowing down or stopping is done in the same manner, by leaning in the opposite direction of travel and speeding up all the blades.

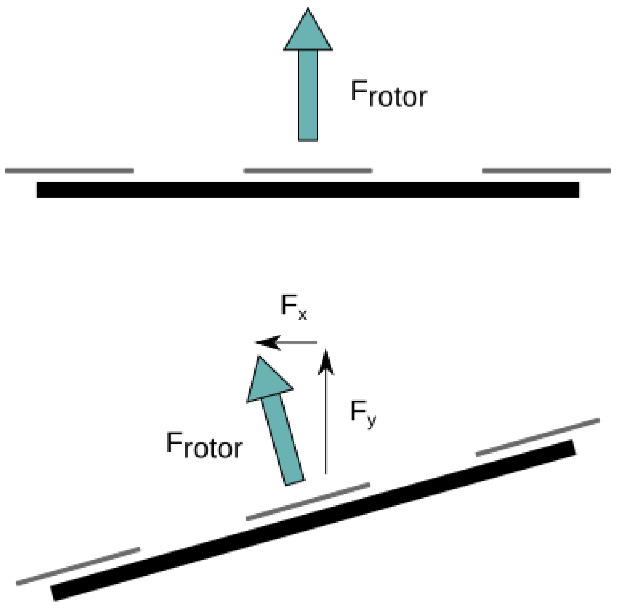

Figure 2.2. multicopter horizontal motion [1])

[1]) ThomasTeisberg.com, Autonomously-Stabilized Quadcopters `www.thomasteisberg.com/quadcopters`

2.2 Multi-robot application

The goal of the application [1] used in this paper is to guide a swarm of mutually localized MAVs through an environment with obstacles to a goal area, which needs to be monitored using their on-board cameras.

This application has two major problems distinguishing it from most path planning tasks.

First problem is, there is no set ending location or position. Usually there is a given point or location that must be reached. A set end position allows for algorithms, that search a path from both the start and the end. A set end position also allows for greedier search in its direction. In this case the goal is to monitor as much predefined area as possible, but the swarm has to find a viable position on its own.

Second problem is the mutual localization. In most path planning tasks, the only limitations are obstacles in the way, and robots dynamic properties, such as its turning radius. In a cooperating group application, the MAVs have to stay within each others field of vision, which is also limited in maximum distance given by the resolution of their on-board cameras.

It is apparent, that to generate a trajectory in this application, a path planning algorithm is needed as well as some sort of heuristic for finding a viable distribution of MAVs above the goal area. Rapidly exploring random tree [8] (RRT) was chosen for this task. RRT is relatively easy to implement, which makes it ideal candidate for being experimented on. It will have to be modified in order to plan for a group of robots, instead of a single robot. It will have to respect new conditions given by mutual localization. And lastly, a modification for optimized search will be attempted.

The problem could also be simplified into 2 dimensions. As the multicopters need to be all at approximately the same altitude, which will be given by their onboard surveillance camera. That is altitude at which they see the most area, while still having sufficient picture quality needed for surveillance.

Chapter 3
Smoothing approaches

A number of different approaches for trajectory smoothing is presented in this chapter. Each one of them will be briefly described along with explanation why it is, or isn't applied to the final solution offered by this paper. Their options of collision checking are also important, as those will have to be expanded on mutual localization checking.

3.1 Fast smoothing of manipulator trajectories using optimal bounded- acceleration shortcuts

This paper [2] presents a shortcutting heuristic. This heuristic repeatedly picks two points on a given trajectory of a manipulator and attempts to interpolate a new collision-free segment between the two endpoints with specified velocity. These segments consist of parabolic and straight-line curves.

This heuristic uses a recursive bisecting technique [7] to check for collision with obstacles. It rejects the simplier method of discretizing the curve to resolution ϵ and checking for collision in each point due to demands on algorithm speed. If ϵ is too small, the checker would be too slow, but if ϵ is not small enough, the chance of missing a collision rises.

To put it simply, the bisecting techniqu covers the segment with a neighborhood which is checked for collision. In case of collision, the segment is bisected and both segments are covered with new neighborhoods. The bisecting point is also covered with a robot-environment distance neighborhood. If the points neighborhood detects collision, the segment has collision. If one of the bisected neighborhoods detects collision, it is recursively bisected.

This method was picked for its major effect on resulting trajectory length. It may need to be simplified to straight lines instead of parabolic curves, before being modified for multi-robot application.

3.2 Trajectory Smoothing Algorithm Based on Empirical Mode Decomposition

This paper [4] puts forward a novel trajectory smoothing algorithm based on empirical mode decomposition which is mainly used for decomposition of signals. In this method, the x and y coordinates series of the trajectories are thought of as individual signals and they are respectively decomposed into different frequency parts through empirical mode decomposition. Then the high frequency parts of the coordinates are adaptively discarded. Finally the residual and low frequency intrinsic mode functions are retained as the smoothed result.

This approach provides convenient results in terms of trajectory smoothing for any robot. However it isn't easily appliable to the multi-robot application considered in this paper, as there wasn't any uncomplicated way to account for the restrictions given by mutual localization of the swarm during the decomposition.

3.3 Trajectory Smoothing Using Motion Primitives

The approach proposed in this paper [5] takes as input a sequence of waypoints connected by straight flight trim conditions, and "smooths" it in an optimal way with the goal of making it compatible with the vehicle dynamics. The smoothing step is achieved by selecting appropriate sequences of alternating trims and maneuvers from within a precomputed library of motion primitives. The idea here is to make the resulting trajectory compatible with the vehicle and therefore trackable with small errors.

This approach would make an excellent last step in the multi-robot application as it makes the trajectory more easily trackable. However, it doesn't ensure that the vehicles will reach their waypoints at the same time, which is a major condition given by the way, the original trajectory is generated.

3.4 Real-time dynamic trajectory smoothing for unmanned air vehicles

This approach [6] is similar to the previous one in a sense of creating transitions between straight line segments on a waypoint defined trajectory. It gives the options of interpolating neighbouring line segments with a curve that either has the same length as the original pieces of segments it is replacing, or goes through the waypoint connecting the two segments, or minimizes the transition time. It does that by knowing the vehicles turning radius and calculating the maneuver as the vehicle approaches its waypoint.

Much like the previous approach, this one would also make excellent last step. The fact, that this approach offers curve of the same length as the segment it's replacing, cancels out the issue presented with previous approach. However this approach is aimed more at winged aircraft, that cannot stop at the waypoint or make a sharp turn. Also this method doesn't consider any obstacles.

Chapter 4
Path planning

4.1 Trajectory generation

To generate a trajectory a modification of RRT [8] algorithm is used. In this approach RRT algorithm grows a tree rooted at the starting configuration by expanding it towards random point P from the search space.

The tree is represented by a matrix where each column represents a configuration of MAVs.

$$\begin{bmatrix} i \\ j \\ x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \\ x_n \\ y_n \end{bmatrix} \tag{1}$$

Where i is number of current column, j is number of column representing previous configuration, n is number of MAVs, and x_l, y_l are coordinates of l-th MAV.

As each sample is drawn, an expansion is attempted between it and the nearest configuration in the tree. As each state can expand in virtually infinite amount of configurations, algorithm simplifies the expansion process by discretizing the expansion options into k possibilities based on current and previous configuration.

The vector representing current configuration is reshaped into matrix C_{ur} for simplier use in the following operations

$$C_{ur} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \end{bmatrix} \tag{2}$$

The vector representing preceding configuration is reshaped into matrix P_{rev} in the same way as C_{ur}. This allows us to calculate the vector of four-quadrant inverse tangents for each MAV more easily.

$$\overrightarrow{angle} = atan2\big(C_{ur}(2,:) - P_{rev}(2,:), C_{ur}(1,:) - P_{rev}(1,:)\big) \tag{3}$$

Then matrix N_{ewp} is sequentially put together by MAV closest to the P point picking one of its k possible new positions closest to the point P. Then the rest of MAVs are one by one picking only those new positions which won't collide with the choices of MAVs that already picked a new position.

$$N_{ewp}(:,l) = C_{ur}(:,l) + \begin{bmatrix} d * cos\big(\overrightarrow{angle}(l) + \overrightarrow{offset}(h)\big) \\ d * sin\big(\overrightarrow{angle}(l) + \overrightarrow{offset}(h)\big) \end{bmatrix} \tag{4}$$

Where $\overrightarrow{offset}$ is vector of k values ranging from $-\pi/4$ to $\pi/4$. l is ranging from 1 to n. And h is ranging from 1 to k. If there are feasible versions of N_{ewp}, it is reshaped back into a column vector and added to the tree.

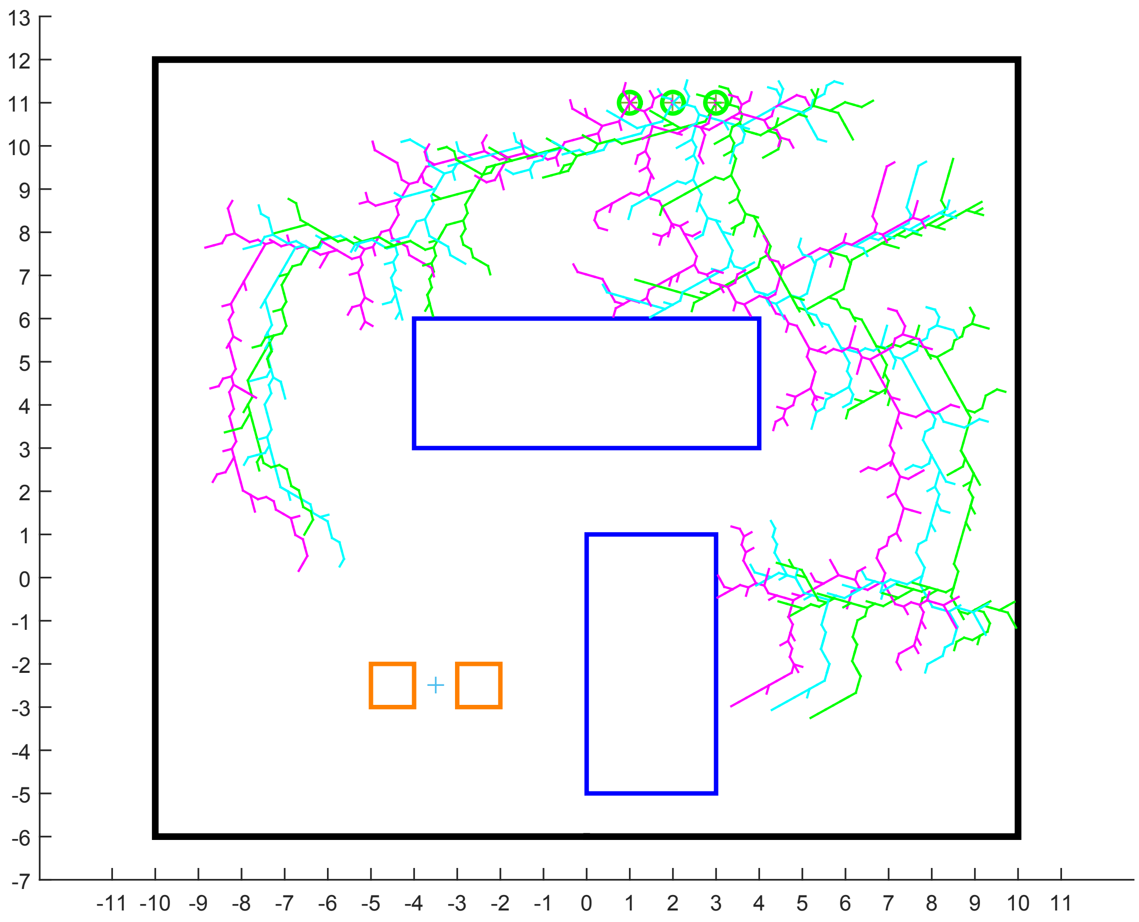

Figure 4.1. tree growth process

After a set amount of iterations, the algorithm checks if any of the MAVs in the tree overlap the goal area. If so, the MAVs don't expand based on distance towards a random point, but based on a cost function calculating coverage cost of the goal area.

The cost function works as follows. The goal area, or areas, is represented by a single matrix of ones and zeros, overlapping all the areas. Its dimensions are given by goal area size and set resolution. Each MAV has its own polygon of ground vision. With the use of Matlab function *poly2mask* which turns region of interest into a matrix, where the region of interest is the goal area, cost function adds up all MAVs contributions to the goal area matrix. Then it applies *log2* to all elements and piecewise multiply the new matrix with the original goal area matrix. That will ensure that cost function will not count MAVs vision that adds to part of matrix representing space between multiple goal areas 4.4 . The output of the cost function is a sum of all elements greater than 0.

This way MAVs profit much more from monitoring new area than from monitoring already monitored area, since $log_2(a) + log_2(b) > log_2(a + b)$ where $a, b > 1$.

```
valuefield=fin.mask;
for j=1:MAVs
 vision{j}=poly2mask(res*([rob.p(1,j)-0.5, rob.p(1,j)-0.5,...
 rob.p(1,j)+0.5,rob.p(1,j)+0.5]-fin.minX),...
 res*([rob.p(2,j)-0.5,rob.p(2,j)+0.5,...
 rob.p(2,j)+0.5,rob.p(2,j)-0.5]-fin.minY),fin.Ydim,fin.Xdim);
 valuefield=valuefield+vision{j};
end
valuefield=valuefield.*fin.mask;
valuefield=log2(valuefield);
cov=sum(sum(valuefield(valuefield>0)))
end
```

Figure 4.2. Part of matlab code calculating cost function

In this process, each MAV picks a new step that contributes the most to the cost function. However if none of the steps increase the value of the cost function, it picks a new step on random. This process is limited only by set number of iterations and all collision restrictions.

After the maximum number of iterations is reached, or after the MAVs come to a deadlock, algorithm picks a position with the highest cost function value as the end point. Then it traces the trajectory back to the root 4.3 by using the references on previous column vector as shown in (1).

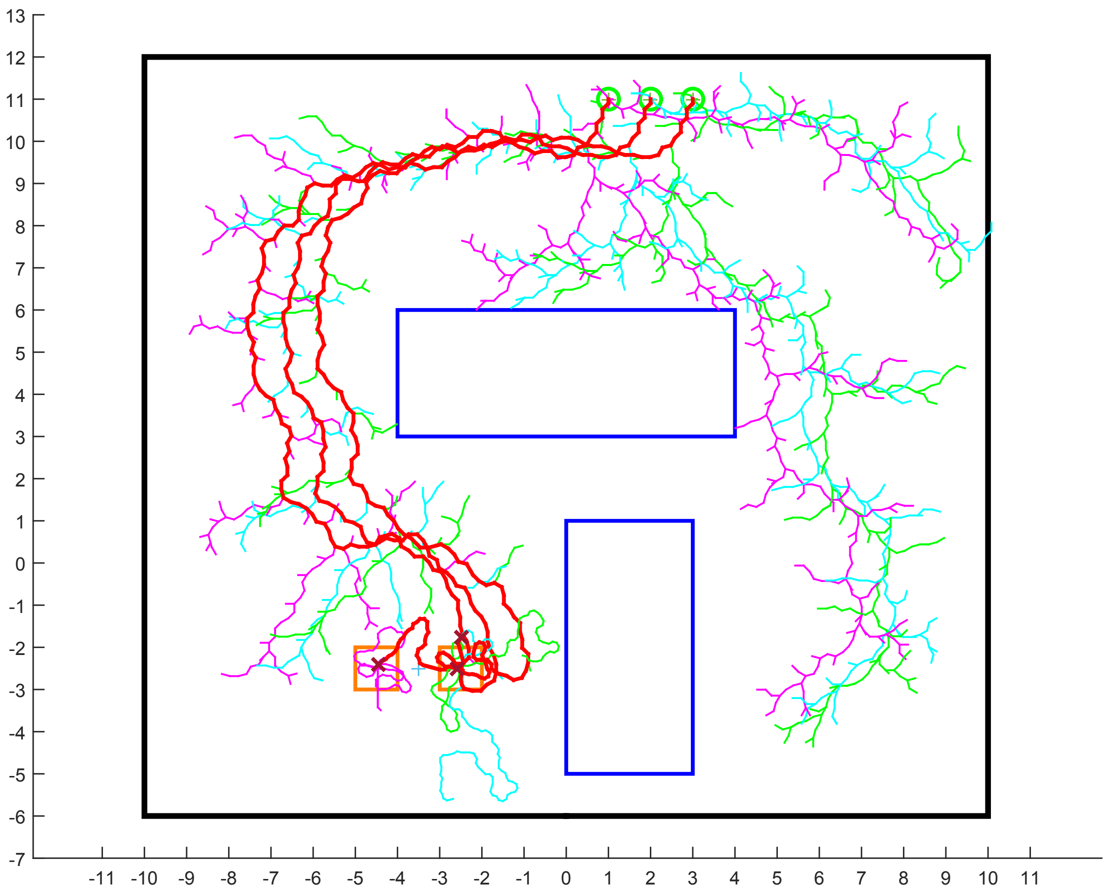

Figure 4.3. backtracing trajectory from finished tree

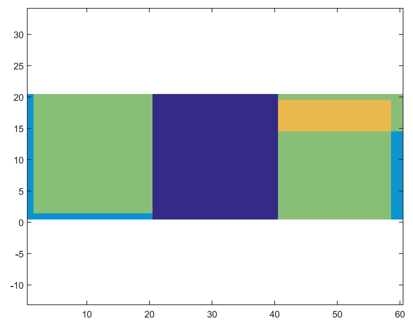

Figure 4.4. matrix representation of goal area in cost function

4.2 Smoothing

The product of the trajectory generator is a feasible yet lengthy and chaotic trajectory containing large amount of points in a form of matrix similar to the tree. This matrix is however missing first two rows, which are no longer necessary and has the following format.

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,r} \\ y_{1,1} & y_{1,2} & \cdots & y_{1,r} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,r} \\ y_{2,2} & y_{2,2} & \cdots & y_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,r} \\ y_{n,1} & y_{n,2} & \cdots & y_{n,r} \end{bmatrix} \tag{5}$$

Where r is number of configurations forming the trajectory.

From the nature of RRT algorithm, which tends to cover as much of the free space as possible, this trajectory tends to have plenty of unnecessary turns and detours that human operator would never make. To reduce the number of these turns and points, a variation of shortcutting heuristic [2] is used

Two random points on the path are selected and connected with a new segment. New segment is checked for collision with obstacles as well as MAV positioning constrains. Collision with obstacles is done by searching for an intersection between a trajectory segment and all lines defining obstacles.

MAVs positioning constrains checking is done by discretizing the the new segment into a number of formation that MAVs go through as they move over the segment. MAVs are expected to reach their waypoints at the same time. Thus the discretization si done as follows

$$P = P_1 + i(P_2 - P_1) \tag{6}$$

Where P_1 and P_2 are columns of (5) and endpoints of the segment, $i = \{0, \frac{1}{res}, \ldots, 1\}$ and res dictates the efficiency of discretization. Each formation P is then checked for minimum distance between all MAVs. If the new segment is feasible, it replaces the original trajectory.

If the segment is not feasible due to collision with obstacle, the smoothing continues into next iteration with no changes where new random two points are selected.

If the segment is not feasible due to MAV positioning constrains, the algorithm takes advantage of MAVs interchangeability and tries a different combination of MAVs starting position and ending position and checks for both obstacles collision and positioning constrains again. This resolves problems with segments that weren't originaly feasible due to two or more MAVs colliding during their flight over the segment 4.5. If no feasible solution is found after checking all $!n$ combinations, shortcutting continues into next iteration with no changes.

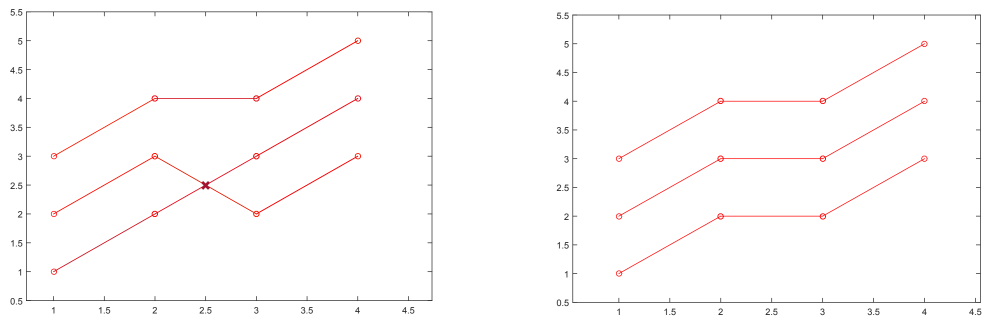

Figure 4.5. MAVs collision resolution

Waypoints P_1 and P_2 are already checked for maximum distance from previous trajectory generation. And since they are connected with straight lines, the MAVs cannot move apart, when moving along the segment, more than they already are in one of the two endpoints. This operation goes on until either the amount of points defining the trajectory meets set minimum amount, or until maximum set number of iteration is reached.

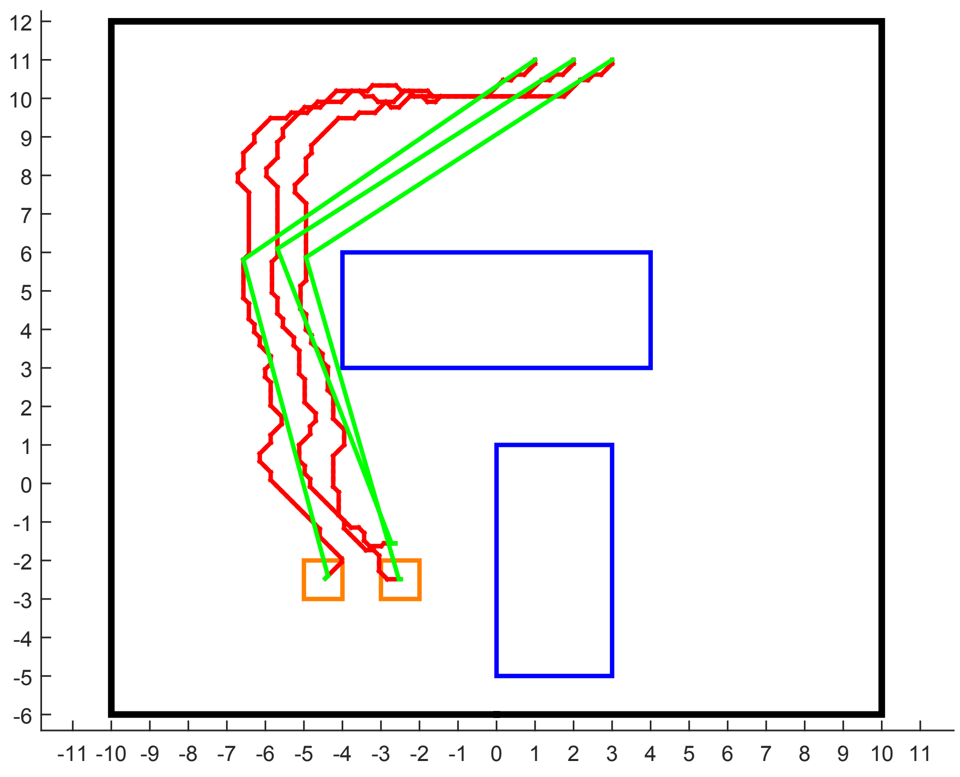

Figure 4.6. shortcutting reached minimum amount of points

4.3 Minimizing

After smoothing is finished, we're left with a path consisting of fewer states. It is apparent to the human eye that this smoothed path could use a few touches to be shorter.

Matlab offers *fminsearch* function, which uses Nelder-Mead simplex algorithm as described in Lagarias et al.[3] to find a local minimum of a scalar function starting at given initial estimate.

One input of this function is the product of smoothing as the initial estimate.

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,s} \\ y_{1,1} & y_{1,2} & \cdots & y_{1,s} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,s} \\ y_{2,2} & y_{2,2} & \cdots & y_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,s} \\ y_{n,1} & y_{n,2} & \cdots & y_{n,s} \end{bmatrix} \tag{7}$$

Where s is number of configurations forming the trajectory after smoothing.

Another input to this function is a scalar function that needs to be minimized. The scalar function calculates the total sum of lengths of all trajectory segments. Therefore the only thing being optimized is travel distance.

To be usable by *fminsearch*, it needs to have only one input which is the matrix representing the trajectory of MAVs. However the start location and end location should stay unchanched. It also needs to consider all restrictions and return *NaN* in case of collision with obstacle or violation of proximity limits on any pair of MAVs.

Time consumption of this operation depends on previous smoothing process as every column of the trajectory adds $2n$ extra dimension of search space to the *fminsearch* function.

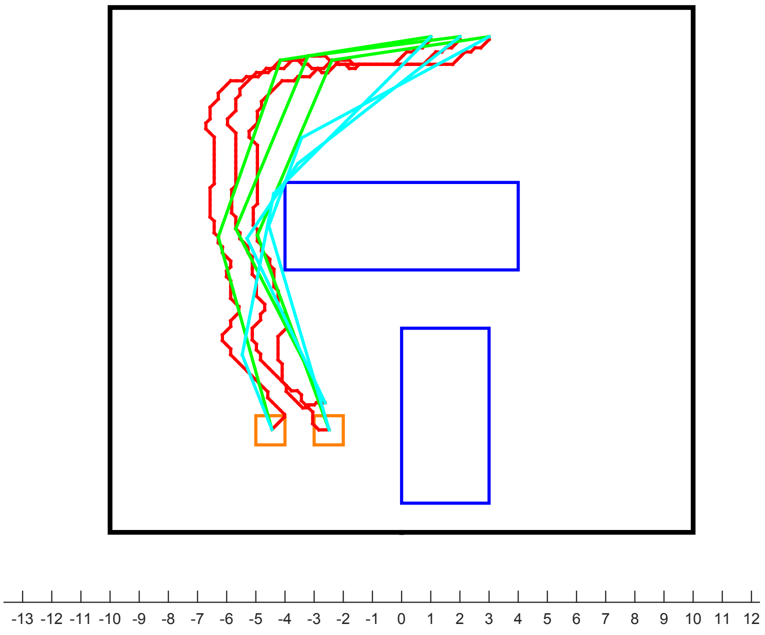

Figure 4.7. result of length minimizing process

Chapter 5
Numerical results

5.1 Length minimizing results

A series of experiments was conducted to demonstrate the procedure described above. While an experiment on real equipment would be more demonstrative, the analysis in virtual environment is important for quantifications of its performance.

The analysis consists of repeatedly smoothing and minimizing generated trajectory, for a series of different generated trajectories, for a number of different maps.

As such the smoothing and minimizing proces ran 10 times for 20 different generated trajectories on 3 different maps.

Recorded values are length of original trajectory, number of points reached by smoothing, as smoothing does not necessarily reach set amount of waypoints, time spent in *fminsearch*, and length of minimized trajectory.

Table 5.1 shows example of statistics for one series of 10 smoothing runs for 1 generated trajectory. Fig. 5.1 shows length statistics for all series.

total length [m]	amount of points	minimizing time [s]
56	283	0
33.37	4	10.45
33.38	3	11.22
33.61	4	10.21
32.57	4	10.59
33.32	3	5.14
33.20	3	4.99
32.80	4	10.60
33.42	3	7.42
33.57	4	10.35
33.54	4	10.47

Table 5.1. 10 data entries for 1 trajectory

The minimum amount of points was set to 3. The amount of points, that smoothing converged to, ranged from 3 to 5. However the amount of points doesn't have a significant effect on minimized length as all cases converged to $\sim 70\%$ of original length. On the other hand it has significant effect on time spent minimizing, as it grows proportional to number of MAVs. Growing from ~ 6 seconds at 3 points to ~ 100 seconds at 10 points, with only 3 MAVs.

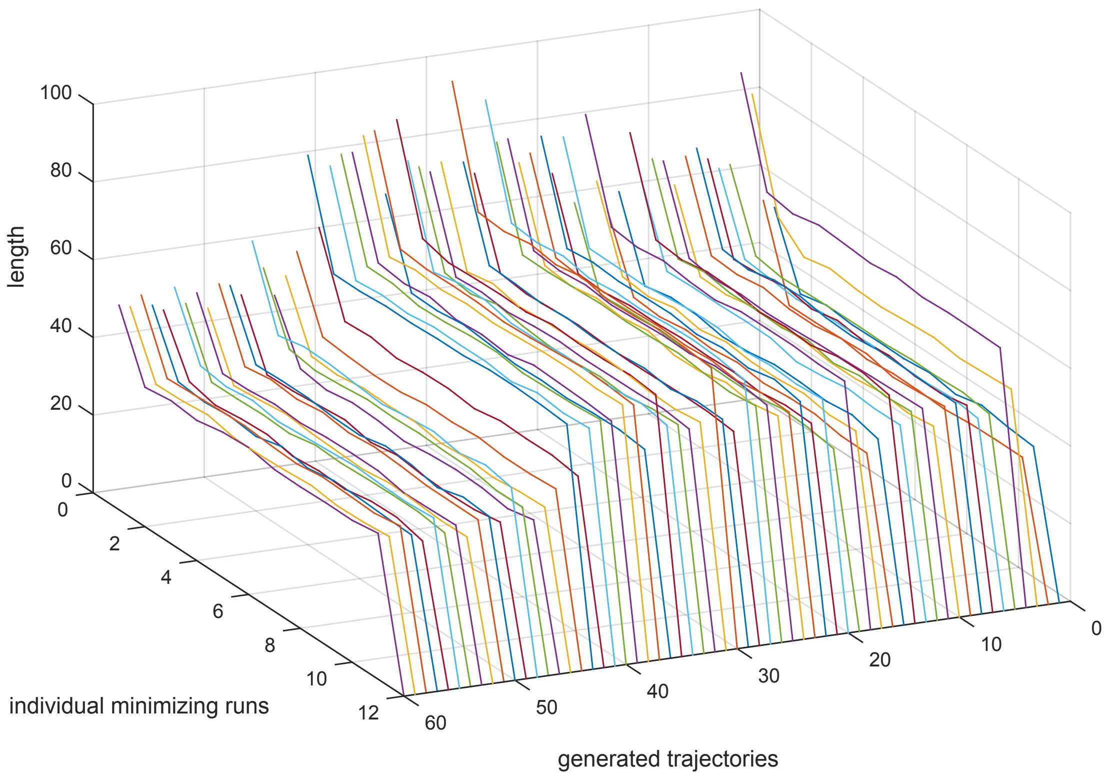

Figure 5.1. Trajectory length statistics

Fig.5.1 shows all recorded runs. The first value of each line represents the original trajectory length. Following values are trajectory lengths after smoothing and minimizing the original trajectory.

5.2 V-REP simulation

With no working equipment available at the time when this algorithm was completed comes no practical experiment. However V-REP [1]) experimentation platform offers sufficiently sophisticated simulation environment.

[1]) CoppeliaRobotics.com, V-REP virtual robot experimantation platform `http://www.coppeliarobotics.com`

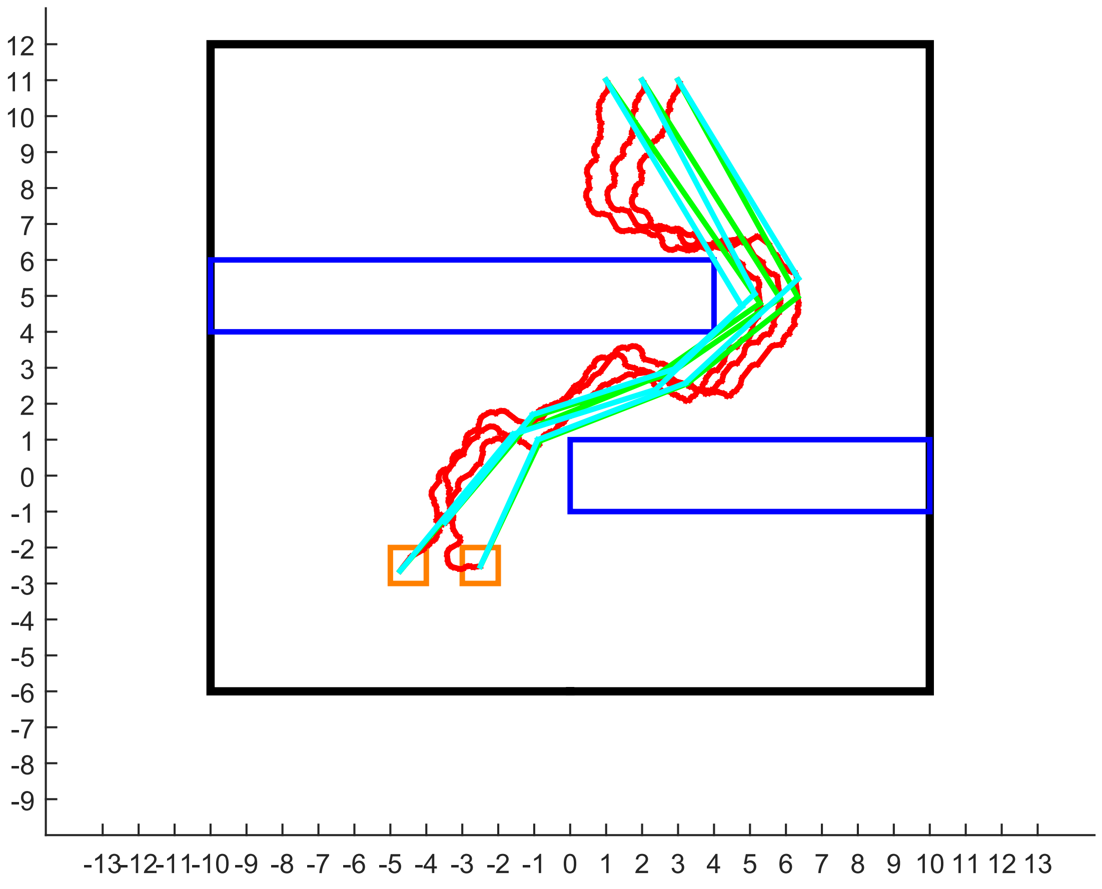

Figure 5.2. Matlab outcome

A trajectory generated by the algorithm was picked on random. This trajectory was generated with the following parameters

```
MAVs=3;
BF=5;
NumPoints=3;
\vdots
rob.minradius = 0.5;
rob.maxradius = 4;
```

Figure 5.3. Part of matlab code declaring variables

Where *NumPoints* sets minimum amount of points the algorithm is allowed to reach. And *rob.minradius* and *rob.maxradius* sets the lower and upper limit on distance between MAVs.

Apparently algorithm couldn't remove any more points after reaching 5 waypoints. It also seems that passing through a corridor, combined with mutual localization, doesn't allow the minimizing function to stray too far from initial estimate.

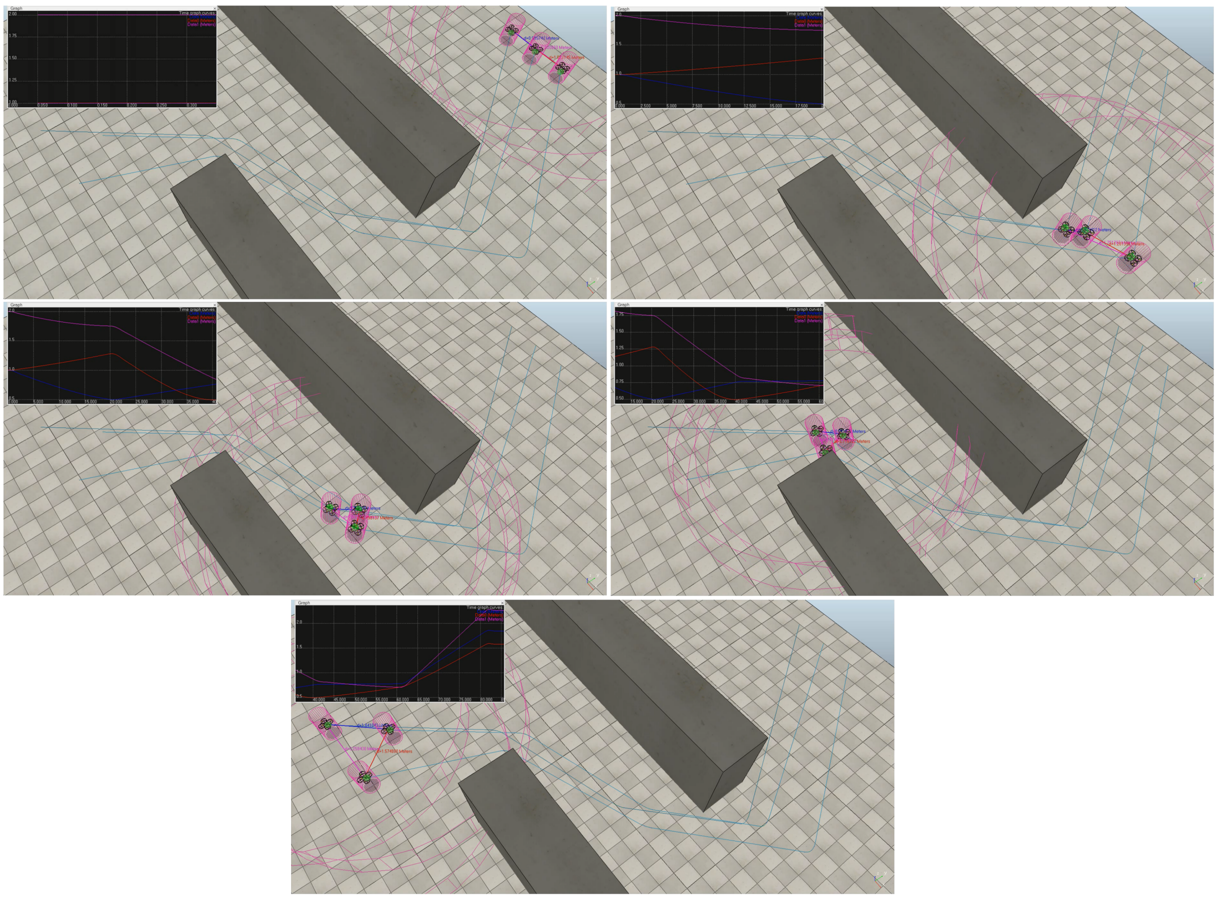

Figure 5.4. V-Rep simulation

MAVs in V-REP must behave closely to the way the algorithm expects them to. This means mainly passing through waypoints at the same time. The way V-REP handles motion control of majority of the robots is by having a controller already implemented in them, and then making them follow a `dummy` [1]. Dummy is yours to control. After creating paths from the waypoints, dummies will follow them exactly the way MAVs are thought of in the algorithm.

The algorithm also expects obstacles to be bigger then they are in reality, since it thinks of MAV as a point, than as a volumetric object. The opposite is done in this case, obstacles will shrink to compensate for MAVs volume.

V-REP allows tracking of nearly any value you can think of and implement. In this case, the distances between the MAVs during the flight will be tracked. Fig. 5.4 shows the simulation process in V-REP.

[1]) CoppeliaRobotics.com, V-REP User Manual `http://www.coppeliarobotics.com/helpFiles/`

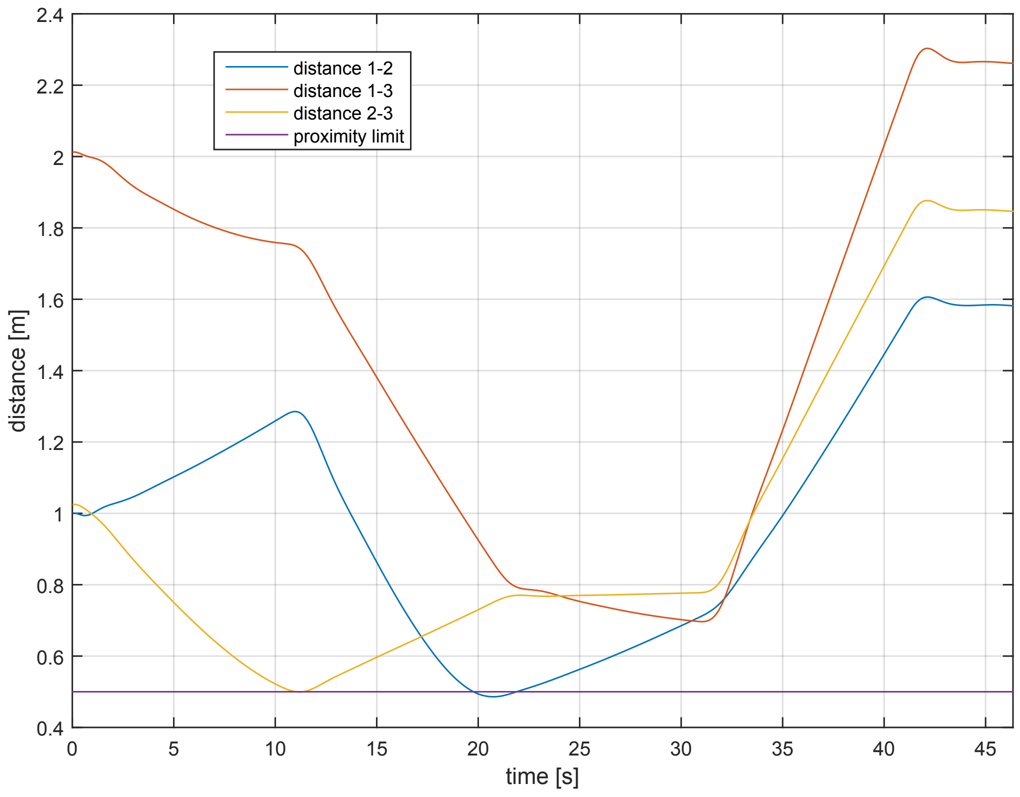

Figure 5.5. mutual distances during the simulation

The graph in fig.5.5 shows the development of distances between each pair of MAVs throughout the flight. It is apparent that the MAVs reached a waypoint every 11 seconds and also reached the end without crashing. However there is a slight overshoot at 20th second. This is due to a dummy moving too quickly through a turn and tracking error taking its toll.

Most preferable way to move the dummies along the path would be, if they didn't have constant speed profile, but instead sped up when moving across the line segment, and slowed down when approaching waypoints.

Chapter 6
Conclusion

The goal of this paper was to design, implement, and experimentally verify algorithm for planning feasible trajectories for a group of mutually localized MAVs in a specific application [1]. Particle Swarm Optimization /cite[PSO] (PSO) technique was used in the original approach. PSO is primarily an optimization technique and the paper was experimenting with its application on path planning task.

This paper uses a modified version of RRT. RRT is primarily a path planning technique and as such has much less issues with finding a path than PSO. But as a path planning technique, it isn't very fond of finding an optimal solution. Although, after all the modifications done in order to respect mutual localization, and further modifications for maximizing MAVs vision of goal area, the algorithm can barely still be associated with RRT.

Next task was to design and implement an algorithm for smoothing and optimization of obtained trajectories while keeping all the previous constraints. The solution for this task was inspired by studying existing approaches for trajectory smoothing [2] [4] [5] [6].

In the end, smoothing was achieved by utilizing the shortcutting heuristic and connecting it with numerical length optimization. The outcome of the proposed method is a series of waypoints optimized for shortest trajectory for all MAVs while respecting their proximity constraints.

Originally the MAVs only carried a pair of cameras on each side. Furthermore these cameras didn't have very wide field of view (FOV), and even narrower field of vision where they could safely recognize other helicopter. The resolution of the cameras put more constraints on minimum and maximum detection range.

As a result, these constraints on mutual positioning were so strict, that the whole group moved as a very slightly changing block of MAVs. As such, it could have been replaced with a single robot representing the whole group.

Later on, the MAVs were supposed to be fitted with four cameras with overlapping FOV, allowing for a full 360° vision. Thus reducing the constraints only to minimum and maximum recognition distance between each two closest helicopters.

However the experimental MAVs ended up being unoperational. Thus the proposed method has been demonstrated using only numerical data and a simulated model.

References

[1] Saska, M. - Chudoba, J. - Precil, L. - Thomas, J. - Loianno, G. - Tresnak, A - Vonasek, V. - Kumar, V., Autonomous deployment of swarms of micro-aerial vehicles in cooperative surveillance, International Conference on Unmanned Aircraft Systems (ICUAS), 2014.

[2] Hauser, K. - Ng Thow Hing, V., Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts, IEEE International Conference on Robotics and Automation (ICRA), 2010.

[3] Lagarias, J.C., J. A. Reeds, M. H. Wright, and P. E. Wright, `Convergence Properties of the Nelde` SIAM Journal of Optimization, Vol. 9 Number 1, pp. 112-147, 1998.

[4] Hejin Yuan, A Novel Trajectory Smoothing Algorithm Based on Empirical Mode Decomposition, Fifth International Conference on Image and Graphics, 2009.

[5] Bottasso, C.L. - Leonello, D. - Savini, B., Path Planning for Autonomous Vehicles by Trajectory Smoothing Using Motion Primitives, Control Systems IEEE Transactions on Technology, vol.16, no.6, pp.1152-1168, 2008.

[6] Anderson, E.P. - Beard, R.W. - McLain, T.W., Real-time dynamic trajectory smoothing for unmanned air vehicles, IEEE Transactions on Control Systems Technology, vol.13, no.3, pp.471-477, 2005.

[7] F.Schwarzer, M.Saha, and J.-C. Latombe, Exact collision checking of robot paths, in WAFR, Nice, France, Dec 2002

[8] LaValle, M. Steven, Rapidly-exploring random trees: A new tool for path planning, Computer Science Department, Iowa State University, Oct 1998

[9] J. Kennedy, R. Eberhart, `Particle Swarm Optimization`, IEEE International Conference on Neural Networks, vol.4, pp. 1942-1948, Nov/Dec 1995

Appendix A
DVD

Included DVD contains the following:

- PDF version of this thesis `BP_2015_hobzamar.pdf`.
- Matlab source codes for generating trajectory and for trajectory smoothing and minimizing + all necessary functions.
- video of V-Rep simulation from chapter 5.
- V-Rep safe file containing the scene in the video.