CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER'S THESIS

Alexander Dubeň

Motion planning with Hermite splines for Unmanned Aerial Vehicle in information gathering task

Department of Cybernetics

Thesis supervisor: **Ing. Robert Pěnička**

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Dubeň Alexander**	Personal ID number:	**406505**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		
Branch of study:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

Motion planning with Hermite splines for Unmanned Aerial Vehicle in information gathering task

Master’s thesis title in Czech:

Plánování pohybu bezpilotního prostředku s využitím Hermitových křivek v úloze autonomního sběru dat

Guidelines:

1. Get familiar with motion planning for robots over multiple target locations with budget constraint (i.e. Orienteering Problem) [1,2] for information gathering tasks.
2. Design multi-goal motion planner in 3D environment [3] with Hermite splines motion primitives for Unmanned Aerial Vehicle.
3. For the designed Hermite spline motion planner, implement method for generating velocity profile with respect to the vehicle velocity and acceleration constraints in order to minimize the flight time over selected target locations.
4. Design method for Orienteering Problem (OP) with Hermite splines motion primitives [4] for the information gathering task using Unmanned Aerial Vehicle.

Bibliography / sources:

[1] Pieter Vansteenwegen, Wouter Souffriau, Dirk Van Oudheusden, 'The orienteering problem: A survey,' in European Journal of Operational Research, vol. 209, pp. 1-10, 2011.
[2] Aldy Gunawan, Hoong Chuin Lau, Pieter Vansteenwegen, Orienteering Problem: A survey of recent variants, solution approaches and applications, European Journal of Operational Research, Volume 255, Issue 2, 2016, Pages 315-332
[3] P. Váňa, J. Faigl, J. Sláma and R. Pěnička, 'Data collection planning with Dubins airplane model and limited travel budget,' in European Conference on Mobile Robots (ECMR), pp. 1-6, Paris, 2017.
[4] R. Pěnička, J. Faigl, P. Váňa and M. Saska, 'Dubins Orienteering Problem,' in IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 1210-1217, 2017.

Name and workplace of master’s thesis supervisor:

Ing. Robert Pěnička, Multi-robot Systems, FEE

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

Date of master’s thesis assignment: **16.01.2018** Deadline for master's thesis submission: **25.05.2018**

Assignment valid until: **30.09.2019**

Ing. Robert Pěnička	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
Supervisor’s signature	Head of department’s signature	Dean’s signature

Prohlášení autora práce

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

V Praze dne............................ ....

Acknowledgements

I would like to thank my supervisor for a lot of on point advices that really helped me on several occasions. Further more I would like to thank my family for psychological and material support during my whole studies with this work being the final step in the process of becoming an independent adult. Finally I would like to thank my girlfriend for her patience during my last year of studies.

Abstract

This thesis focuses on motion planning for Unmanned Aerial Vehicle (UAV). The goal is to find a feasible trajectory in 3-D or 2-D space through a subset of target locations with known rewards such that the sum of collected rewards is maximized. The starting and ending locations are known. A limiting factor for number of targets that can be visited is maximal time of flight. This task is called Orienteering Problem (OP). To fully utilize the motion range of the UAV, a Hermit splines are used to generate smooth trajectories. The minimal time of flight estimate for a given Hermite spline is calculated using known motion model of the UAV limited by maximum velocity and acceleration. The proposed solution to the Orienteering Problem using Hermite splines introduced as Hermite Orienteering Problem (HOP) is based on Random Variable Neighborhood Search algorithm (RVNS), combining random combinatorial state space exploration and local continuous optimization. This approach was compared with other currently known solutions to the OP motivated by UAV applications and showed to be superior as the resulting trajectories reached better final rewards in all testing cases. The trajectories have been also successfully tested on a real UAV and their feasibility was verified.

Abstrakt

Diplomová práce se zabývá plánováním trajektorií pro bezpilotní drony tzv. Unmanned Aerial Vehicle (UAV). Cílem je nalezení trajektorie ve 2-D či 3-D prostoru skrz určitý počet předem definovaných cílů se známým ohodnocením, tak aby součet ohodnocení navštívených cílů byl co největší. Počátek a cíl trajektorie je předem známý. Maximální doba letu je omezující podmínkou určující počet navštívených cílů. Takto definovaná úloha se nazývá Orienteering Problem (OP). Pro nalezení hladkých a spojitých trajektorií byly použity Hermitovy křivky, umožnující využít plný pohybový potenciál moderních UAV. Čas letu pro danou křivku je odhadován ze znalosti pohybového modelu UAV, který je omezen maximální rychlostí a zrychlením. Navrhované řešení OP s využítím Hermitových křivek definován jako Hermite Orienteering Problem (HOP) je založeno na algoritmu Random Variable Neighborhood Search (RVNS), který kombinuje náhodné kombinatorické prohledávaní stavového prostoru s lokalní kontinuální optimalizací. Toto řešení bylo srovnáno s jinými současně známými přístupy k řešení OP, přičemž ve všech provedených testech byla kvalita řešení s použitím Hermitových křivek nejlepší. Použitelnost a vhodnost získaných trajektorií byla ověřena na reálném UAV.

Contents

List of Figures

List of Algorithms

1 Introduction

The number of real world applications of Unmanned Aerial Vehicles (UAV) or so called drones, such as pictured in Figure 1, grows dramatically. Adoption of the UAV technology across industries leaped from the fad stage to the mega-trend stage fairly quickly as more businesses started to realize its potential, scope, and scale of global reach. The use of UAVs is already well anchored in following industrial and socio-economic fields [27]:

- Agriculture - Crop surveys, wild life counting [38]
- Topography - Aerial mapping, environment monitoring, land measurement [16]
- Safety - Forest fire detection, plume tracking, search and rescue, medical supplies delivery, flash flood detection [52][28][26][41]
- Military - Reconnaissance, demining, attack [34]
- Cinematography - Film-making, aerial photography [44]
- Industrial servicing - Inspection of power-lines, pipes, buildings, power plants [11][24][33]
- Other [44]

Whether UAVs are controlled by a remote controller or accessed via a smart-phone app, they possess the capability of reaching the most remote areas with little to no manpower needed and require much less effort, time, and energy compared to conventional means. This is one of the biggest reasons why they are being adopted worldwide.

Imagine a transmission tower or a solar power-plant. Both consist of parts that need to be periodically maintained. The insulation might degrade with time or the solar panels need to be checked for faults or accumulated dust. UAVs are very suitable for this kind of monitoring task since typical UAV is equipped with a camera. This kind of UAV application is already being used in industry. For example transmission line servicing company [20] uses manhandled drones to check the power lines and towers for possible cases of corrosion and wear. Another company [40] uses drones for solar panel surveillance. By using a drone with infra red camera, it is possible to detect individual hot spots on the cells, diode failures, shattered or dirty modules, coating and fogging issues or junction box heating. The crucial part of these applications is the need of a trained human operator for the UAV handling. An autonomous UAV capable of such task would dramatically lower the cost of maintenance and allow the automation of the whole process which is a current global trend in industry. Hence the motivation behind this thesis is enabling the autonomous surveillance using UAVs.

Unfortunately the challenge of fully autonomous movement in a generic environment is a very complex and hard task even with the use of recent AI breakthroughs. The whole

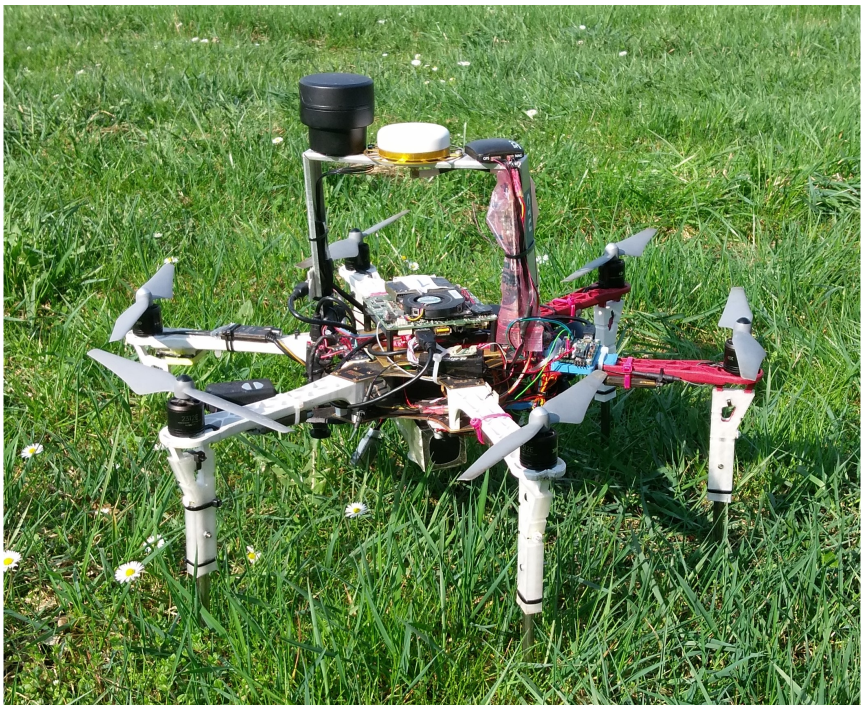


Figure 1: Hexarotor UAV created by MRS research group and used for testing proposed path planning method

problematic can be divided into several topics such as path planning, obstacle avoidance, on-line path corrections, dynamic re-planning, mapping, localization etc. The main goal of this thesis is enabling a surveillance over given target locations, which basically means finding a path through given target locations in optimal manner. This challenge belongs to the domain of path planning tasks and all the other subjects such as the collision avoidance and mapping can be considered auxiliary, hence only the path planning part of the whole problematic is addressed in this thesis.

The need for finding an optimal path through a set of target locations is very generic problem with possible applications not only in UAV related topics and has been extensively researched in recent years [4]. This led to a definition of Orienteering Problem (OP), which is a formalization directly applicable to the problem of UAV surveillance. The goal of the OP is to find a trajectory through a set of target locations with known rewards (importance) and known start and end locations, so the sum of rewards of visited targets is maximized. The number of visited targets is limited by maximal time of flight. This constraint arises from limited capacity of batteries in current UAVs. This effectively means that the UAV will not always be able to visit every target but rather those targets that are more important.

The transmission tower maintenance was mentioned as one of the motivating scenarios and possible application of this thesis. The target locations in envisaged transmission tower maintenance application would be the critical parts prone to wear and damage. The rewards determining the importance of each target location could be defined by the life expectancy of each inspected part. As the inspection is usually done visually using a camera each part could be also inspected from several viewing angles with some being more informative than other which further influences the reward for visit.

1.1 Problem Overview

The problem solved in this thesis can be logically divided into several sub-problems. First of all, the Orienteering Problem (OP) is defined, which is the core problematic. The precondition of this problem is having a set of points which might be either defined in plane or in full three dimensional space. Each point has a reward assigned that indicates the importance of the target and there is a start and end point given. The goal is to visit a subset of these points such that the collected reward from all visited locations is maximized while the time of flight is kept smaller or at most equal to a given time budget. A problem defined in this way is called Orienteering Problem which is a generalization of well known Traveling Salesman Problem (TSP). The rigorous definition of OP is given in Chapter 4.

The OP in its original form utilizes Euclidean paths between each target location to construct the solution trajectory. Even though the real UAV is capable of flying along Euclidean trajectories, since it is a holonomic vehicle (vehicle with zero turning radius), it requires flying with very small velocities as the turns composed of Euclidean trajectories are very sharp. This leads to an introduction of parametric polynomial curve called Hermite spline. The Hermite spline enables to generate fairly complex trajectories that are smooth, which enables to follow the trajectory with considerably higher velocities. In addition it is very simple to use the Hermite splines in 2D as well as 3D space. The mathematical properties and definition of Hermite splines are described in Chapter 2.

Since the OP is limited by a given maximal time of flight, it is desirable to fly as fast as possible. This requires a generation of such trajectories that are in accordance with physical limitations of the used UAV. Hence an the velocity with which the UAV is capable of flying along the spline when evaluating the suitability of a given curve needs to be estimated. This so called velocity profile have to be calculated separately for planar and spatial case. The solution to velocity profile estimation is proposed and described in Chapter 3.

The new variant of the Orienteering Problem motivated by high velocity Hermite splines is defined as Hermite Orienteering Problem (HOP). The difference from the generic Orienteering Problem is a bigger complexity because apart from the target locations that should be visited and the order of visit, also the heading angles and actual velocities need to be determined at each location. The definition of the HOP is given in Chapter 4 (Section 4.3).

The OP is NP-hard problem as it combines two NP-hard problems which is the

Traveling Salesman Problem and the Knapsack problem. This makes finding the optimal solution very computationally demanding even for small data sets. This sort of problem is typically solved using heuristic methods which find the optimal solution or a solution very close to the optimum with less requirements on computational time. A heuristic based on Random Variable Neighborhood Search (RVNS) algorithm was used to solve the proposed HOP. This approach was already successfully used for solving OP and similar class of problems on several occasions. The created HOP solution based on RVNS combines random combinatorial exploration of the state space with continuous optimization which enables the to find high quality solution in reasonable time. The created HOP solution extends the RVNS algorithm, used to solve the combinatorial part of the problem, by continuous optimization which determines the optimal heading angles and velocities at each visited location. The combinatorial optimization works with a set of samples determined by an initial heading angle and velocity sampling rates which is further extended by the continuous optimization that introduces more promising samples. The proposed solution based on RVNS algorithm is described in Chapter 5.

1.2 State of the art

All the sub-tasks mentioned in Section 1.1 are very generic with large number of possible applications, so there is a lot of research groups actively working on solving them or enhance existing solutions. A brief review of current research performed on this topic and topics closely related is listed in following section.

The OP is very similar to one of the most classical routing problems with a long history which is Traveling Salesman Problem and can be considered its generalization [25]. There exist a large number of heuristics solving the TSP [36]. Probably the most used is Lin-Kernighan heuristic [22][19]. The OP is NP-hard optimization problem combining two other NP-hard problems which are TSP and the Knapsack problem [39]. This makes finding the optimal solution very computationally demanding. The Branch and Bound and Branch and Cut algorithms are examples of efficient optimal solutions to OP, but these algorithms still require significant computational resources even for small data sets. This led to a creation of several heuristic methods for OP solution.

The Orienteering Problem has been introduced in 1984 by Tsiligirides [46] who defined the Euclidean OP (EOP) and proposed two heuristics for solving the problem. First method uses Monte Carlo based strategy for picking the best solution among large number of randomly generated paths. The second approach utilizes vehicle-scheduling algorithm with one depot. Tsiligirides also designed several benchmarks in a form of datasets used to compare the quality of different solutions.

Another approach to EOP was introduced by Ramesh and Brown [35], who defined a four-phase heuristic which uses insertion, improvement and deletion phases to iteratively improve the path. A heuristic called Fast and Effective was proposed by Chao et al. [10]. In this solution only the targets locations are used, that lie inside an ellipse around the start and end with axes defined by travel budget. The initial set of generated paths is searched and the path with largest reward is modified by a set of simple operations such as two-point exchange, one-point movement and other optimization operations. Chao et al. also proposed several datasets for benchmarking.

The solution of the HOP used in this thesis is based on the Variable Neighborhood Search heuristic by Hansen and Mladenovi [18] which was already successfully applied to OP by Sevkli et al. in [43]. The basic Variable Neighborhood Search (VNS) introduced by Hansen and Mladenovi is very generic algorithm which can be used for wide range of optimization problems. The VNS algorithm works with neighborhood structures around incumbent solution that define a sub-space in the whole state space, that can be searched for better solutions. The scope of these neighborhoods is changed with each step, typically extending the scope to find possible better solutions further away from current solution. Each newly defined neighborhood is thoroughly searched for a local optima. This process continues until a given stopping condition is met such as number of iterations. The VNS algorithm was used for solving the OP by Sevkli et al. who defined the neighborhoods as shake and local search operations which are used to modify the paths through the OP target locations and find the best solution. The Path Insert and Path Exchange were used as shake operators in order to randomly modify the current solution and escape from possible local optimum. The local search operators which aim to systematically search the imminent neighborhood of current solution were Insert and Exchange operators. Insert operator changes a position of one target location in current solution and Exchange operator switches positions of two target locations in the solution. Sevkli et al. also introduced a variant of VNS algorithm which is Randomized Variable Neighborhood Search where the local search operators are randomized instead of systematic approach.

The RVNS heuristic was also used to solve the proposed HOP solution, but it had to be extended, as HOP also introduces the need for continuous optimization of heading angles and velocities. The RVNS heuristic for HOP uses a predefined set of heading angle and velocity samples in each target location in order to find initial feasible solution using greedy algorithm. The method further iteratively tries to improve the current solution using a set of random shaking and local search operators for selecting most rewarded set of targets and their order of visit. This is considered to be the combinatorial part of the problem. Furthermore, a continuous optimization operators are used to introduce new samples at target locations in order to minimize the time of flight and further improve the collected reward. This combination of random walk-like exploration of continuous state space and exhaustive local optimization enables to find solutions of good quality while keeping the sampling rate of continuous space and thus the computational time reasonably low.

Euclidean trajectories used in EOP are typically not suitable for real world UAVs as it creates very sharp and even unfeasible trajectories which can’t be traversed in fluent motion. This led to the usage of other methods in trajectory generation for autonomous vehicle movement. An important breakthrough in this field was published in year 1957 by L. E. Dubins who introduced a new type of planar curve called Dubins curve [12]. This curve can be solved analytically and yields the shortest path between two points with known tangents and maximal curvature. Dubins curves are used very extensively in robotics and control theory. It is suitable for navigation of wheeled vehicles [8] (cars are a typical example of curvature constrained vehicle), airplanes and underwater vehicles [9] as well. Dubins curves were also used in previous work upon which this thesis is based which combines the RVNS solution of Orienteering Problem with usage of Dubins curves as a motion primitive [31]. This variant of the OP was named Dubins Orienteering Problem (DOP). The DOP solution was further extended by the use of neighborhoods around the targets, since usually in information gathering, it is enough to be close enough to the target to gather required information [32]. The neighborhood is a disk with defined radius in space around the target, inside which any point can be visited in order to successfully perform the surveillance of target location in the center of the disk. A solution to DOP using self-organizing maps was proposed in [14].

However this thesis aims to solve the OP for UAVs that are typically not constrained by topology of the trajectory if the movement is slow enough. A competing approach to trajectory generation is the usage of parametric polynomial splines. There exist a wide range of parametric curve types with each having different features. In the motion control domain, two types are used very frequently thanks to their natural suitability - Bézier and Hermite curves. The Bézier and Hermite curves are almost equivalent apart from the difference in the mathematical definition, hence both curves are used in similar problem solutions. A lot of the research on this topic is done in planar space for autonomous ground vehicles, for example here [51, 50]. Hermite curves have been used in similar tasks such as [48, 7, 45]. The Hermite curve is parametrized by starting and ending location and by heading vectors originating in these points. This parametrization is very suitable for UAV motion model and was chosen as the motion primitive used in our solution to OP. In addition Hermite curves can be easily connected to create a smooth and complex trajectory through several locations. This is known as Hermite spline.

Even though Hermite curves offer neat way to generate trajectories in space, it is not ensured that the trajectory will be feasibly traversed by the given vehicle since each vehicle is limited by maximal speed, acceleration and other constraints. This means that in order to check feasibility and estimate the time of travel, the maximal allowable velocity along the whole curve and the motion dynamics need to be determined from known motion model of the vehicle. This approach is usually called a calculation of velocity profile.

A very convenient way to generate the velocity profile is to bind the maximal velocity with the actual curvature of the curve as was done in [21]. However, this was used only in planar case for a ground vehicle. This idea can be used also in 3D if we assume decoupling of vertical and horizontal movement as was proposed in [15]. This approach is also used in this thesis as it is fast and does not require a complicated motion model of the UAV. Another solution to velocity profile calculation is method used in [23, 42]. The idea is again based on curvature constrained maximal velocity in each sample but then the minima are found along the curve using gradient descent search algorithm and the velocity profile is determined using numerical integration applying a bang-bang control policy on the reference dynamic model. If the feasible profile is not found the velocity minima are iteratively lowered until one is found.

1.3 Contributions

The main advantage of solution described in this thesis is a fact that it can be applied in 2D as well as 3D space topology which is enabled by the use of Hermite splines. Majority of the current solutions is designed only in planar scenarios which greatly limits the applicability in real world situations.

The OP solution using RVNS algorithm is based on work done in [31], where the Dubins curves are used as a motion primitives and the UAVs is presumed to use constant height and velocity when following the trajectory. In the solution proposed in this thesis, the velocity profile respecting the dynamic movement constraints such as maximal acceleration and velocity is generated over the whole trajectory based on Hermite spline. This ensures smoothness and feasibility of generated trajectories as well as fast and dynamic movement which is major advantage compared to the constant velocity movement of DOP. The fast movement of the UAV enables to find solutions with high quality. This was verified in a series of comparison tests where the new Hermite curve based solution to the OP yields superior results over the Dubins and Euclidean based approaches as the average gathered reward was better in all tested scenarios.

The use of continuous optimization in the RVNS based solution also enables to find HOP solution in reasonably short time while using only sparse initial sampling of the heading angle and velocity at the target locations.

2 Hermit splines

Parametric curves in general are very powerful tool with wide range of applications in computer graphics and robotics. They are effective whenever there is a need for data interpolation using smooth continuous function. The main advantage is the small number of defined parameters upon which the curve is generated. One of the most used curve types are the cubic polynomial splines. A spline is a parametric curve defined piecewise by polynomials. The term spline comes from the flexible spline devices used by shipbuilders and draftsmen to draw smooth shapes.

The cubic polynomial splines are composed of one or more third degree polynomial curve segments connecting the interpolated data points. Third degree curves usually offer sufficient flexibility for creation of considerably complex shapes while being simple to calculate and work with. There exists a wide variety of spline types such as Hermit spline, Bézier spline, B-Spline or Catmull-Rom spline. Each type is a bit different in its parametrization and implementation and is suitable for different application [49].

Since this thesis aims to generate paths for flying vehicle, where it is possible to determine the heading, velocity and acceleration, there is one spline type which seems to be naturally suitable for this kind of application - the Hermite curve.

2.1 Hermite curve

A generic third degree polynomial parametric curve in 3D space is defined as follows:

$$\begin{aligned} x(t) &= a_3t^3 + a_2t^2 + a_1t + a_0 \\ y(t) &= b_3t^3 + b_2t^2 + b_1t + b_0 \\ z(t) &= c_3t^3 + c_2t^2 + c_1t + c_0 \end{aligned} \tag{1}$$

Unknowns a_i, b_i, c_i are constant coefficients of the polynomials describing the characteristic of the curve in each dimension. The curve parameter t is used for samples generation along the curve, with coordinates in each axis given by corresponding polynomial and the value of parameter t (hence the name parametric curves). This parameter is usually set in interval $[0; 1]$ where $t = 0$ yields the starting point of the curve and $t = 1$ the end point. This means that once the coefficients of all three polynomials are calculated it is possible to sample the curve with arbitrary accuracy by defining the size of the step of t.

The Hermite curve is defined by two endpoints and corresponding derivatives in the endpoints (tangent vectors of the curve) which determine the shape or rather the curvature of the curve. The tangent vector can be defined by angle relative to x axis and vector norm. In the case of the UAV, these parameters can be related to heading angle and actual speed. In relation to orienteering problem the triplet of target location, heading angle and actual speed defines a state. The example of such curve in 2D can be seen in Figure 2 [47].

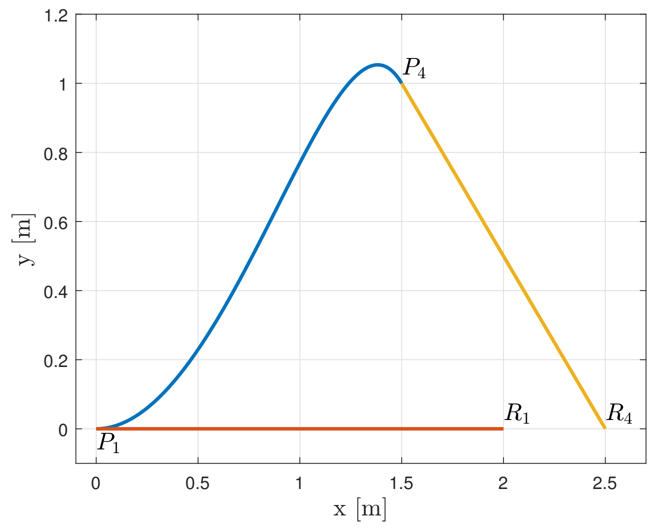


Figure 2: An example of Hermit curve in 2D through target locations P_1 and P_4 with heading vectors determined by points R_1 and R_4

With the knowledge of Hermite curve constraints, the Equation (1) can be rewritten into vector form where matrix A is the unknown constant matrix and T a vector of powers of curve parameter. Following equations will be derived only for x coordinate as the solution for other coordinates is identical which also means it is possible to easily create curves in any number of dimensions:

$$\begin{aligned} x(t) &= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}, \\ x(t) &= T \cdot A. \end{aligned} \tag{2}$$

Since we know the coordinate of the start location, end location and also their respective derivatives we can write a system of linear equations:

$$\begin{aligned} x(0) &= \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \cdot A, \\ x(1) &= \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \cdot A, \\ x'(0) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix}_{t=0} \cdot A &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \cdot A, \\ x'(1) &= \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix} \cdot A. \end{aligned} \tag{3}$$

Writing these equations in a matrix form yields:

$$G = B \cdot A. \tag{4}$$

The solution of A is then:

$$
\begin{aligned}
&A = B^{-1} \cdot G = M \cdot G, \\
&A = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{bmatrix},
\end{aligned} \tag{5}
$$

and the whole curve equation now can be expressed as:

$$x(t) = T \cdot M \cdot G, \tag{6}$$

$$x(t) = \begin{bmatrix} 2t^3 - 3t^2 + 1 & 2t^3 + 3t^2 & t^3 - 2t^2 + t & t^3 - t^2 \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{bmatrix}. \tag{7}$$

Matrix G are the given geometric constraints of the curve. Matrix M is called basis matrix and its product with vector T is called a basis or a set of blending functions. Equation (8) shows elements of basis - four blending functions. The solution for x can be rewritten as (9) which clearly shows how the blending functions act as a weighting factor for each of the geometrical constraint [5].

$$
\begin{aligned}
f_1(t) &= 2t^3 - 3t^2 + 1 \\
f_2(t) &= -2t^3 + 3t^2 \\
f_3(t) &= t^3 - 2t^2 + t \\
f_4(t) &= t^3 - t^2
\end{aligned} \tag{8}
$$

$$x(t) = f_1 \cdot x(0) + f_2 \cdot x(1) + f_3 \cdot x'(0) + f_4 \cdot x'(1) \tag{9}$$

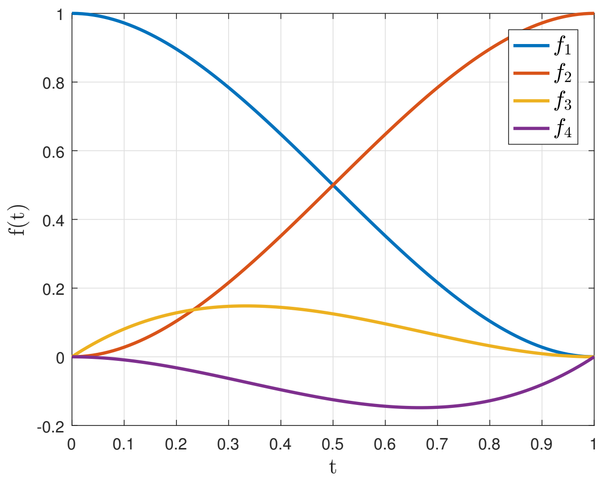


Figure 3: Hermit basis functions

Figure 3 shows the basis functions. It is noticeable that blending function corresponding with beginning/ending point tend to overwhelm the blending functions for derivatives and diminish/empower with parameter t changing. This result is expected as the location of beginning point clearly defines the curve at $t = 0$ and similarly the ending location is most important at $t = 1$. The influence of each limit point is on par exactly at $t = 0.5$.

2.2 Hermite spline

The goal of the orienteering problem is to find a multi target trajectory, however the Hermite curve was so far defined only between two points. As noted earlier, the spline consists of several separate parametric curve segments. Hence the Hermite spline can be constructed by connecting several curves at limit points. This means if we construct each consecutive curve segment starting in the ending point of the previous one we get a spline. The only problem is that different heading vectors might cause singularities violating the smoothness of the whole spline. This can be avoided if the ending heading vector of previous segment is always enforced to be equal to the starting heading vector of the following segment. The HOP state is also defines only by actual orientation and speed at target location which relates to both incoming and outgoing curve, so the heading vector equality is not violated.

2.3 Bézier curves

It is worth mentioning that well know and widely used Bézier curves are equivalent to Hermit curves. The difference is in the constraints defining the curve. Figure 4 shows how the Bézier curve is defined. The coordinates of start and end point stays the same but instead of derivatives in the limit points, two additional control points are used. All four control points define a convex hull which won't be crossed by the curve. This parametrization makes the behavior of the curve more predictable and that is the reason it is used more often than the Hermit curve, although in our case the Hermit curve is more suitable. This fact can be for example leveraged when generating trajectories through corridors with turns. The resulting curves are equivalent nevertheless, since the geometric constraints for Bézier curve can be easily rewritten into a form defining a Hermit curve [47]:

$$\begin{aligned}
P_{1_H} &= P_{1_B} \\
P_{4_H} &= P_{4_B} \\
R_{1_H} &= 3(P_{2_B} - P_{1_B}) \\
R_{4_H} &= 3(P_{4_B} - P_{3_B})
\end{aligned} \tag{10}$$

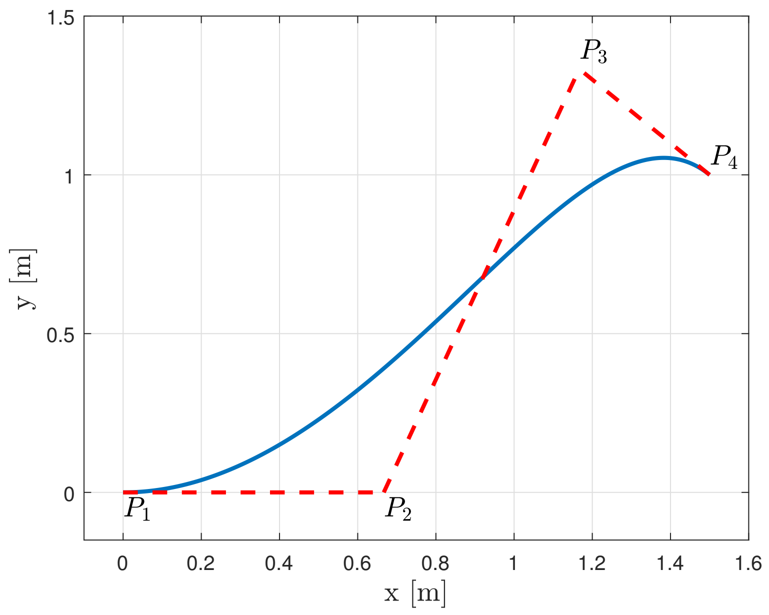


Figure 4: An example of Bézier curve in 2D through target locations P_1 and P_4 with control points P_2 and P_3

2.4 Implementation

In order to generate and manipulate the Hermite curves easily a hspline c++ library was created. Since the theory is suitably described by matrix equations an external open-source library Eigen3 [30] was used in order to work with matrices with ease, eliminating the need for low level matrix operation implementation which is unnecessary.

The hspline library enables to create HSpline object by passing necessary information for the curve creation which is:

- A list of states containing locations through which the curve should traverse and heading angles along with initial velocity.
- Resolution defining the step of the curve parameter in order to generate samples. This parameter needs to be set with care as large amount of samples dramatically influence the performance when working with multiple curves which is the case of HOP solution algorithm. In the performed experiments, the 15-20 samples per curve offered satisfactory details while keeping the performance demands relatively acceptable.
- Maximal vertical and horizontal acceleration and velocity. This is used to determine the time of flight (TOF) by calculating the velocity profile.
- Heading multiplier which is used to determine the heading vector of the curve. This

enables to separate the UAV speed from actual tangent vector of the curve to generate curves with different curvature characteristic while keeping the UAV velocity intact.

- Flag indicating the spatial resolution of the curve - to generate either 2D or 3D curve.

The samples of required curve are then calculated by direct application of the theory in Section 2.1. The constraint matrix G is constructed and multiplied by the basis matrix which yields the coefficients of curve polynomial. This polynomial is then used to generate the curve samples by stepping through the parameter t from 0 to 1. The 2D and 3D curves are created in almost the same way. The only difference is in the dimensions of used matrices and structures used to save the samples. The library is also indifferent to the number of states the spline should traverse, although mostly only the basic Hermite curves given by two limit points are used in the HOP RVNS algorithm. The example of generated Hermite spline along with the heading vectors in each node is shown in Figure 5.

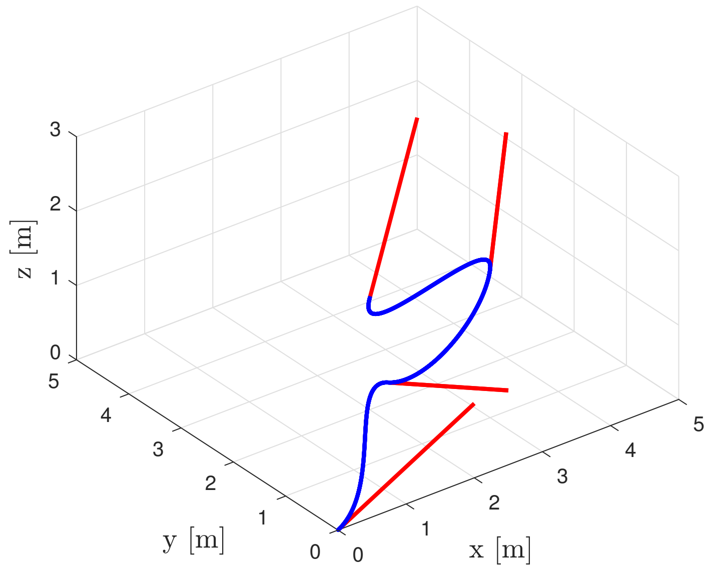


Figure 5: An example of generated Hermite spline in three dimensional space traversing four target locations. The heading vectors at each target are plotted in red color.

3 Velocity profile

The introduction of Hermite splines in Chapter 2 yields smooth trajectories in space, connecting arbitrary points with arbitrary resolution. Since the goal is to find smooth curves to use in HOP solution, it is not possible to measure the curve quality by length as the shortest path is always the Euclidean. The use of Euclidean paths in UAV trajectory generation is not desirable as the trajectories are hard to follow by the UAV especially at the target locations where the trajectory changes direction. A better measure to determine the fitness of given curve is the required time the UAV needs to traverse it. This leads to the definition of so called velocity and acceleration profile based on the known motion dynamics model of the UAV, which yields the Time of Flight (TOF) estimation. Naturally the most preferable solution would be the shortest time of flight along the curve. This is determined by the UAV's limited flight capabilities in terms of maximal acceleration and velocity. The calculation of the velocity profile is different in a 2D curve scenario from a 3D scenario.

A very frequent approach to the velocity profile generation along parametric curves uses the curvature of the curve to determine the maximal allowable velocity along the whole curve. This defines profile with low velocities in sharp turns and high velocities in straight segments which is desirable. This approach was successfully used in planar scenarios in [21, 48, 7, 45]. An extension of this approach to be used in full three dimensional space was introduced in [15].

3.1 2D velocity profile

In the case of the planar curve generation, the method based on curvature constrained velocity used in [21] can be directly applied. The vertical velocity and acceleration is considered zero and constant during whole flight. This method defines the possible velocity and acceleration at each sampled point along the curve using a simplifying assumption that the trajectory between two samples of the curve is Euclidean and that acceleration and velocity vectors always keep the direction towards next sample. This enables to use simple dynamical equations for constantly accelerated motion in straight line when modeling the UAV's movement along the curve. The drawback is a deformation of the curve shape and shortening actual travel distance, although if sufficiently small step between curve samples is chosen, this error can be neglected. The whole process of determination of velocity and acceleration at each curve sample, in order to fulfill the motion constraints of the UAV, is described in following text.

The horizontal acceleration of the UAV is composed of tangential a_r and radial a_t acceleration which are bounded together:

$$a = \sqrt{a_r^2 + a_t^2}. \tag{11}$$

To make the acceleration directly dependent on the curvature of the flight path is very convenient, since it is possible to generate the velocity profile where the velocity is lowered during sharp turns (high curvature areas) and vice versa which is expectable behavior when dealing with the feasible flight trajectories. The curvature κ along the whole Hermit spline in the horizontal plane can be calculated as:

$$\kappa = \frac{|x'y'' - y'x''|}{(x'^2 + y'^2)^{\frac{3}{2}}}. \tag{12}$$

The derivatives of the curve in x and y coordinates can be easily obtained by using derivative of vector T in (6). The radial acceleration a_r along the curve can be calculated using the actual velocity v as:

$$a_r = \kappa v^2. \tag{13}$$

The acceleration a of the UAV along the curve is composed of radial acceleration a_r and tangential acceleration a_t which are bounded together by $a_{h_{max}}$, which is the maximal possible acceleration the UAV can achieve. Maximal possible velocity along the curve $v_{h_{lim}}$ can be defined as:

$$v_{h_{lim}} = \frac{\sqrt{a_{h_{max}}}}{\kappa}. \tag{14}$$

Equation (14) is a solution for v from (13), assuming the worst case scenario when $a_r = a_{h_{max}}$. In this case, the only acceleration component of the a is the radial acceleration with tangential acceleration being zero (very sharp turn). This ensures that the maximal velocity is dependent on the curvature. However this velocity needs to be also limited by actual maximal horizontal velocity of the UAV $v_{h_{max}}$, hence the full equation for the maximal velocity along the curve is

$$v_{h_{lim}} = max(v_{h_{max}}, \frac{\sqrt{a_{h_{max}}}}{\kappa}). \tag{15}$$

Using (11), the maximal possible tangent horizontal acceleration can be calculated as:

$$a_{t_{lim}} = \sqrt{a_{h_{max}}^2 - a_r^2}. \tag{16}$$

The velocity profile calculation algorithm iterates over the points of sampled curve. The iteration needs to be performed in forward and also backward manner in order for the UAV to be able to accelerate and decelerate accordingly. In each iteration only two samples (current and next) are used to calculate the actual velocity in the next sample. The performed algorithm is summed up in Algorithm 1.

Algorithm 1 2D Velocity profile calculation algorithm (s_i - curve sample, l - sample location, c - actual curvature, v - actual velocity, $v_{h_{lim}}$ - maximal possible actual velocity from (15))

```
$s_i : \{l, c, v, v_{h_{lim}}\}$
$s_c \leftarrow get\ first\ curve\ sample$
$s_n \leftarrow get\ second\ curve\ sample$
$s_l \leftarrow get\ last\ curve\ sample$
$d \leftarrow getDistance(l_c, l_n)$
while $s_n \neq s_l$ do
    $a_r \leftarrow v_{s_c}^2 \cdot c_{s_c}$
    $a_{t_{lim}} \leftarrow \sqrt{a_{h_{max}}^2 - a_r^2}$
    $v_{lim} \leftarrow \sqrt{v_{s_c}^2 + 2 \cdot a_{t_{lim}} \cdot d}$
    $v_{s_n} \leftarrow min(v_{s_n}, v_{lim}, v_{h_{lim}}^{s_n})$
    $s_c \leftarrow s_n$
    $s_n \leftarrow s_{n+1}$
end while
```

The equation used to find v_{lim}, which is a candidate for velocity in the next sample, was derived as described in the (17). The variable s is an Euclidean distance between current sample s_c and next sample s_n, t is time of flight between these samples.

$$\begin{gathered} s = \frac{1}{2} a_{t_{lim}} t^2 + v_{s_c} t \\ t = \frac{-v_{s_c} + \sqrt{v_{s_c}^2 + 2a_{t_{lim}} s}}{a_{t_{lim}}} \\ v_n = a_{t_{lim}} t + v_{s_c} \\ v_n = \sqrt{v_{s_c}^2 + 2a_{t_{lim}} s} \end{gathered} \tag{17}$$

With the velocity profile calculated this way the UAV is trying to accelerate to the maximal possible velocity given by the actual curvature while respecting the maximal acceleration limit. The process described in the Algorithm 1 is performed twice, with second run using reverse order of samples. This ensures that both acceleration and deceleration phases do not violate the UAV constraints.

3.2 3D velocity profile

The calculation of the velocity profile over 3D curve is also based on curvature constrained velocity but the whole process is extended in order to include also the vertical components of the movement. The motion model of the UAV can be described by maximal velocity and acceleration in horizontal plane and vertical axis separately. In order to bound both vertical and horizontal limitations together, the simplification of using Euclidean paths with velocity and acceleration vectors heading towards next sample is used again, although in this case its in three dimensional space. The main advantage of this approach in three dimensional case is that the horizontal and vertical components of velocity and acceleration can be described using triangle similarity based on geometry shown in Figure 6.

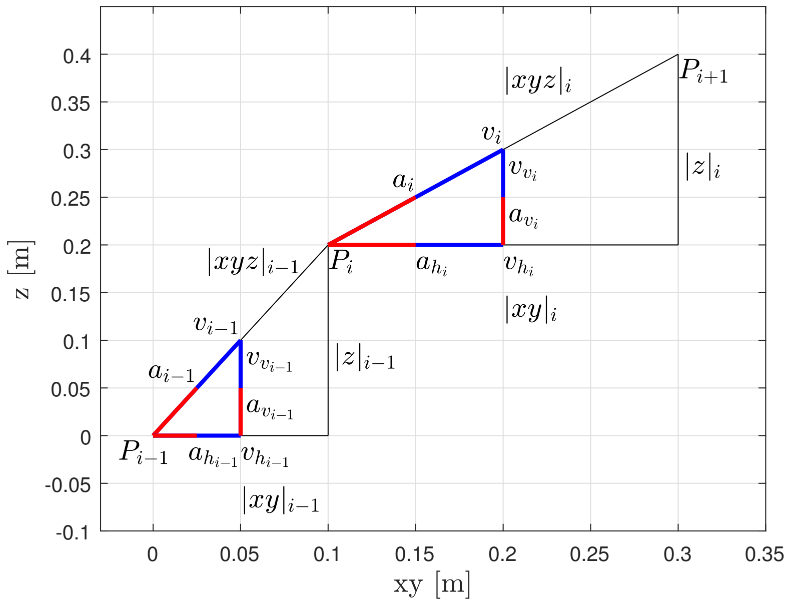


Figure 6: Velocity, acceleration and its components between two samples of the curve. xy axis corresponds with projection of the samples into the horizontal plane

Following equations then hold true:

$$a_i = \frac{|xyz|_i}{|z|_i} a_{v_i}, \tag{18}$$

$$a_i = \frac{|xyz|_i}{|xy|_i} a_{h_i}, \tag{19}$$

$$v_i = \frac{|xyz|_i}{|z|_i} v_{v_i}, \tag{20}$$

$$v_i = \frac{|xyz|_i}{|xy|_i} v_{h_i}. \tag{21}$$

The velocity profile calculation is similar to the 2D case with the horizontal and vertical limitations being applied separately. The horizontal components are again limited based on actual curvature along with maximal values for horizontal acceleration and velocity. The vertical dynamics are only limited by maximal vertical acceleration and velocity. It is assumed that the acceleration is always positive hence samples have to be again firstly iterated in forward direction, which will limit the velocity profile for acceleration, and than backwards which will limit the deceleration. On each step, the current radial acceleration in horizontal plane a_r is calculated, which yields the maximal horizontal tangent acceleration $a_{h_{lim}}$ using (16). The maximal vertical acceleration $a_{v_{lim}}$ is determined by UAV's maximal achievable vertical acceleration $a_{v_{max}}$. Components $a_{v_{lim}}$ and $a_{h_{lim}}$ each define different acceleration in the direction of Euclidean path towards next sample (tangential acceleration) using the triangle similarity equations (Equation (18) for vertical component and Equation (19) for horizontal component). The smaller acceleration of these two is chosen for further calculations. The selected acceleration is then used to calculate the candidate velocity in the next sample using again equations in (17). This candidate velocity v_{lim} is then limited by known horizontal and vertical maximal velocities, transformed into the direction toward next sample by triangular similarity equations (21) and (20). The whole 3D velocity profile calculation is laid out in Algorithm 2.

Algorithm 2 3D Velocity profile calculation algorithm (s_i - curve sample, l - sample location, c - actual curvature, v - actual velocity, $v_{h_{lim}}$ - maximal possible actual velocity from Equation 15), $v_{v_{lim}}$ - maximal vertical velocity of the UAV, d_{xyz} - distance between samples, d_{xy} - distance between samples projected into horizontal plane, d_z - vertical distance between samples

```
$s_i : \{l, c, v, v_{h_{lim}}, v_{v_{lim}}\}$
$s_c \leftarrow get\ first\ curve\ sample$
$s_n \leftarrow get\ second\ curve\ sample$
$s_l \leftarrow get\ last\ curve\ sample$
$d_{xyz} \leftarrow getDistanceXYZ(s_c, s_n)$
$d_{xy} \leftarrow getDistanceXY(s_c, s_n)$
$d_z \leftarrow getDistanceZ(s_c, s_n)$
while $s_n \neq s_l$ do
    $v_{h_{s_c}} \leftarrow v_{s_c} \frac{d_{xy}}{d_{xyz}}$
    $a_r \leftarrow v_{h_{s_c}}^2 \cdot c_{s_c}$
    $a_{h_{lim}} \leftarrow \sqrt{a_{h_{max}}^2 - a_r^2} \cdot \frac{d_{xyz}}{d_{xy}}$
    $a_{v_{lim}} \leftarrow a_{v_{max}} \cdot \frac{d_{xyz}}{d_z}$
    $a_{lim} \leftarrow min(a_{h_{lim}}, a_{v_{lim}})$
    $v_{lim} \leftarrow \sqrt{v_{s_c}^2 + 2 \cdot a_{lim} \cdot d_{xyz}}$
    $v_{max} \leftarrow min(v_{h_{lim}}^{s_n} \cdot \frac{d_{xyz}}{d_{xy}}, v_{v_{lim}}^{s_n} \cdot \frac{d_{xyz}}{d_z})$
    $v_{s_n} \leftarrow min(v_{s_n}, v_{lim}, v_{max})$
    $s_c \leftarrow s_n$
    $s_n \leftarrow s_{n+1}$
end while
```

3.3 Time of flight

With the velocity profile generated either in 2D or 3D, the TOF estimate can be calculated by iterating through samples with the calculated velocities, summing the time intervals t between samples determined as

$$t = \frac{2s}{v_c + v_n}. \tag{22}$$

In addition, the velocity profile can be used to generate equidistant samples of the curve in time. This is very useful as the MPC trajectory following controller [6] used in UAV this thesis was tested on, accepts samples of the trajectory in this format. This enables to combine spatial and dynamical characteristics of the trajectory in one set of data. The trajectory tracking controller in the UAV accepts series of points in the space and tries to traverse them with a constant time gap. Providing the controller a set of trajectory samples with constant distance differences would mean constant speed of the UAV along the curve. The samples with varying distances need to be selected to modify the velocity (samples with larger distance will be traversed in the same time as close samples which effectively means larger velocity of the UAV). Equidistant in samples in time are simply acquired by iterating over current curve samples, while summing the actual time of flight until required time step is reached. Only the last sample in this procedure is used with the rest being filtered out. Unfortunately this used sample usually does not fulfill the condition of being equidistant in time to previous sample, hence the sample is moved proportionally to the left over time difference. This is repeated until the whole spline is covered which filters out a large number of samples. The density of sampling in areas of the curve where the velocities should be low, such as sharp turns, is very high and the spatial gaps between samples are larger on narrow high velocity sections. This is shown in Figure 7.

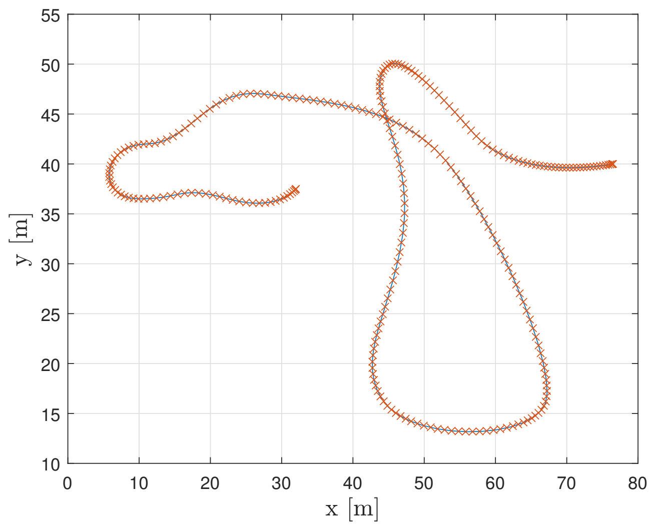


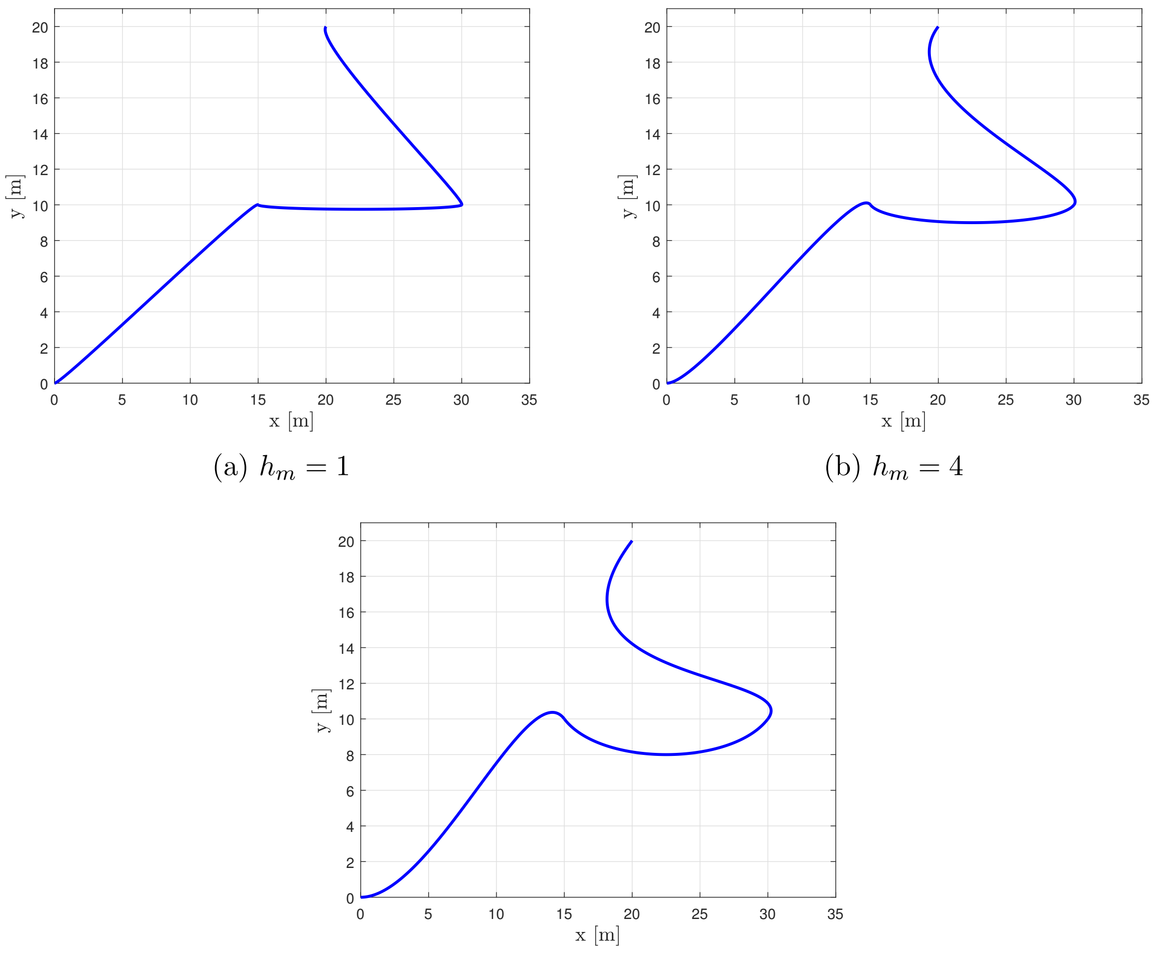
Figure 7: Filtered curve samples based on the calculated velocity profile created for trajectory following controller

3.4 Implementation

The velocity profile calculation is incorporated into hspline library. The whole algorithm described in this chapter is performed right after the Hermite spline samples are generated. Velocity profile calculation can be time consuming if the curve consists of large number of samples since it is needed to iterate over all samples at least two times (forward and backward). In the performed experiments where each curve was approximately 5 meters long on average, a number of 15 samples per curve has been shown to be sufficient enough to show decent details of sampled curve, while keeping the calculation time reasonable.

It is important to note that since the maximal velocity is curvature constrained, it might be not possible to achieve required velocities at target locations defining the curve. The OP solution algorithm marks these curves as unfeasible and does not further work with combinations of states that caused it. This also enables to calculate the velocity profile over spline traversing multiple target locations in piece-by-piece manner as it is ensured that it will be always possible to achieve the velocity at the target location.

Furthermore, the curvature near the limit points is largely defined by the norm of the heading vector which is dependent on the velocity of the UAV at corresponding locations. This dependence was adopted in order to lower the complexity of the optimization performed in OP solution. The heading vector norm was set to be proportional to the actual velocity at the target location. The chosen multiplier h_m determining the ratio between velocity and heading vector norm can be chosen arbitrarily and has big influence on the resulting shape of the spline. Spline generated over the same set of states but using different heading vector multiplier are pictured in Figure 23.



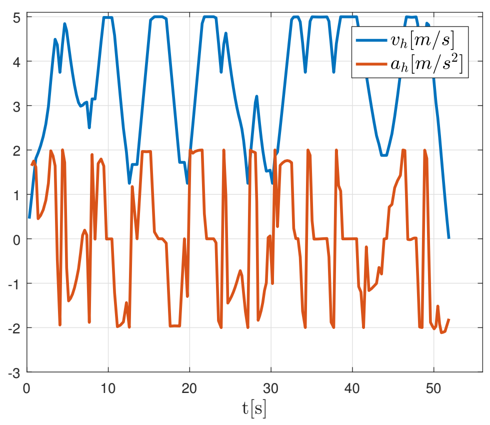
(a) $h_m = 1$ (b) $h_m = 4$

(c) $h_m = 8$

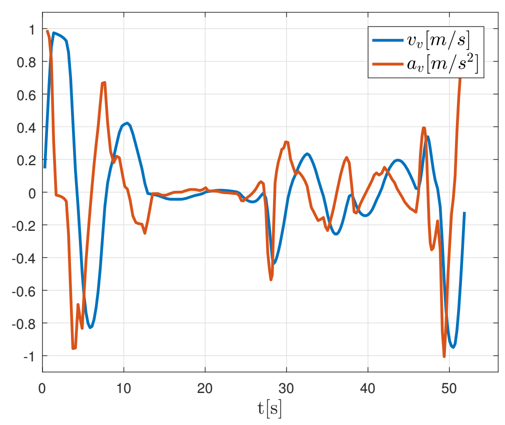
Figure 8: Hermite spline generated using different values of heading multiplier (h_m)

It is clear that the larger the heading multiplier is the more the smoother the curve is, however, with the multiplier being too large a significant overshooting starts to show, which is also not ideal. In addition, if the curve is smooth, there are less instances of unfeasible velocity profiles because of small heading norms for curves spanning large distances result in curves very similar to Euclidean paths which are considered not suitable for UAV trajectories. This hints there should be an optimal value of heading multiplier for given set of target locations, however the analytical formula for this dependence was not found during work on this thesis. Hence the suitability of a given multiplier was determined experimentally for each scenario as the distances between points in given scenarios can be largely different which requires differently curved trajectories.

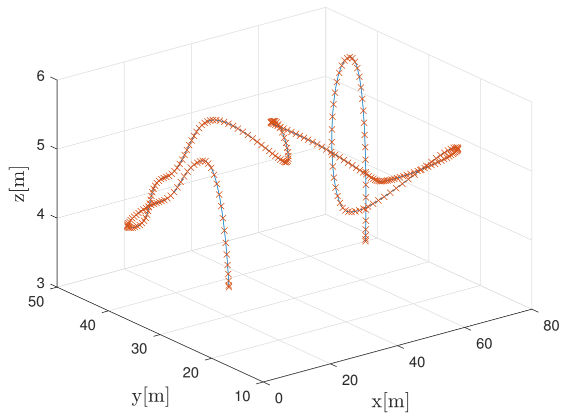
An example of generated velocity and acceleration profile using methods described in this chapter are shown in Figure 9. The limitations used in this figure are determined by the maximal horizontal $v_{h_{max}}$ and vertical $v_{v_{max}}$ velocities and accelerations $a_{h_{max}}$, $a_{v_{max}}$ of the UAV used in subsequent testing of this thesis and are summarized in Table 1.



(a) Horizontal velocity v_h and acceleration a_h profile



(b) Vertical velocity v_v and acceleration a_v profile



(c) Generated Hermite spline with samples equidistant in time for the trajectory following controller

Figure 9: Example of calculated velocity profile over Hermite spline in three dimensions

Table 1: UAV motion constraints

$v_{h_{max}}$	$a_{h_{max}}$	$v_{v_{max}}$	$a_{v_{max}}$
$5m/s$	$2m/s^2$	$1m/s$	$1m/s^2$

4 Orienteering Problem

When dealing with the Orienteering problem (OP) it is important to mention well known Traveling Salesman Problem (TSP). The origins of this problem can be traced as far back as 18th century, although back than it served more as a mathematical puzzle without further scientific background. First general mathematical formalization of the problem has emerged in 1930s with the name Traveling Salesman Problem being introduced by Hassler Whitney at Princeton University [13]. Since then it became one of the most intensively studied problems in optimization because it became clear that there is enormous number of possible applications for this problem such as vehicle path planning, logistics, networking or even DNA sequencing.

The goal of the TSP is finding the shortest Hamiltonian cycle in a given graph with known distance between vertices. This means finding a path that traverses each vertex while being the shortest possible. Unfortunately it was shown that the TSP problem is NP-complete, thus the worst-case running time for any systematic algorithm searching for TSP solution increases super-polynomially with the number of vertices in the graph. This means that finding the optimal solution for even relatively small number of vertices might be out of reach of current processing power. Luckily, since the problem was studied so extensively, a large number of heuristics to find solutions very close to the optimum or even the optimal one, while using limited computational complexity was found and successfully applied.

The usability of TSP in UAV path planning is very suitable since every navigation problem can be transformed to a set of points that have to be visited during the UAV travel with solution being basically the solution to TSP problem. However current UAV technology suffers by one big limiting factor and that is battery life. The electronics on board the UAV are typically high performance computing machines using wide array of sensors and interconnected submodules with high energy demands, but the motion itself using propellers driven by electro motors consumes the largest amount of energy. In the case of electrically driven ground vehicles, such as cars with electro engines, this problem is addressed by carrying very large battery with enough capacity for a longer operational times. Unfortunately the larger battery used in UAV means more weight so the motion requires more energy and the advantage of bigger capacity is largely negated. In addition, enlarging the dimensions of UAV contradicts with the aim of UAV applications which is having a very small and agile remote vehicle enabling to reach remote locations which might be very hard or impossible to get to by conventional methods. The typical operational time of current commercial solutions is in the range of tens minutes of flight. Hence it would be suitable to include an constraint on the maximal length/time of flight for the path. If the length of the path is limited there need to be some motivation for the selection of points that are to be included in the solution. This can be achieved by assigning the reward or priority to each target location. The variant of TSP modified in this way is called Orienteering Problem and it is the main focus of this thesis.

4.1 Euclidean Orienteering Problem

The goal of the Orienteering Problem is to find a subset of target locations with given rewards to be visited in order to maximize the sum of collected rewards while keeping the length of the path less or equal to a specific budget. To define the problem mathematically we introduce a set of target locations $S = \{s_1, ..., s_n\}$ with each location $s_i = (t_i, r_i)$ consisting of position in plane $t_i \in R^2$ and specific positive reward $r_i \in R$. The location can be also expanded to R^3. Two targets are chosen to act as a start and an end of the solution and are typically assigned a reward equal to 0 since it would only act as a bias for the final reward as these two would be always included. In further text we will note σ_1 as the starting location and σ_n as the ending location. The path length is constrained by T_{max} which is the travel budget. The goal is to find a subset of k target locations $S_k \subseteq S$ in order to maximize the sum of collected rewards $R = \sum r_i$ while keeping the length of the solution within the budget T_{max}. The path is defined as a sequence of visited locations in order given by permutation Σ of indexes of locations σ_i in the subset S_k:

$$\begin{gathered} S_k = \{s_{\sigma_i}\} \ \forall i \in (1, ..., k), \\ \Sigma = (\sigma_1, ..., \sigma_k), \\ 0 \leq \sigma_i \leq n, \\ \sigma_i \neq \sigma_j \ for \ i \neq j, \\ \sigma_1 = 1, \sigma_k = n. \end{gathered} \tag{23}$$

The variables being optimized in the OP are k, S_k and Σ meaning number of visited targets, actual locations of the targets and the order of visit. The basic form of the OP uses Euclidean distances as a travel cost between locations [17]. An operator $\mathcal{L}_E(t_{\sigma_i}, t_{\sigma_j})$ defines Euclidean distance between locations t_{σ_i} and t_{σ_j}. The optimization problem can be formalized as [32]:

$$\begin{gathered} \underset{k, S_K, \Sigma}{maximize} \ R = \sum_{i=1}^{k} r_{\sigma_i} \\ subject \ to \ \sum_{i=2}^{k} \mathcal{L}_E(t_{\sigma_{i-1}}, t_{\sigma_i}) \leq T_{max}, \\ \sigma_1 = 1, \sigma_k = n. \end{gathered} \tag{24}$$

The Orienteering Problem is a generalization of TSP, since if we ensure the budget T_{max} is large enough to visit all given locations and instead of maximizing the sum of rewards, the path length is minimized, the problem becomes the TSP. This fact can be also used in the heuristic for OP solution as often we might find different paths with the same reward and it is natural to accept the one with shorter distance or TOF, since shorter solution could lead to a possibility of addition of a new target.

4.2 Dubins Orienteering Problem

Even though the Euclidean Orienteering Problem is very generic and can have a lot of applications, in the case of UAV it is not as useful because the Euclidean trajectories are not typically feasible for motion planning of vehicles. Even though the UAVs are holonomic vehicles, the traversal of Euclidean path would enforce to use very small velocities at turning points since these would be typically very sharp. This fact led to the introduction of different motion primitives used in OP solution. The extension of OP was proposed in [31], which utilizes Dubins curves as motion primitives instead of Euclidean paths. In [12] L. E. Dubins introduced a new kind of planar curves suitable for curvature constrained vehicles, i.e., vehicles with limited turning radius. The state of the so called Dubins vehicle is defined as $q = (t, \theta)$ where $t \in R^2$ is the position in plane and $\theta \in \langle 0, 2\pi)$ is the heading angle. Dubins proved that the shortest path between two states consists only of straight segments (S-segment) and arcs with given curvature (either left turn or right turn - L-segment and R-segment). The optimal path is one of six possible maneuvers (LSL, LSR, RSL, RSR, LRL, RLR). These Dubins maneuvers can be determined analytically.

In the Euclidean OP, the heading angle of the path at each target location is uniquely determined by the relative positions of the subsequent target points. This heading, however, might not be the most suitable when dealing with the Dubins curves. This means, apart from the positions that will be traversed by the generated trajectory in the OP, that also the heading angles have to be determined since different heading angles might produce shorter trajectories. This variant of the OP was introduced as Dubins Orienteering Problem (DOP). The mathematical definition of the OP (24) is extended by a set of heading angles corresponding with each visited location $\Theta = (\theta_{\sigma_1}, ..., \theta_{\sigma_k})$. The Dubins vehicle state in the visited target locations is denoted $q_{\sigma_i} = (t_{\sigma_i}, \theta_{\sigma_i})$. The length of Dubins curve between q_i and q_j is defined by an operator $\mathcal{L}_D(q_i, q_j)$. The DOP is defined as:

$$\begin{gathered}
\underset{k, S_K, \Sigma, \Theta}{maximize}\ R = \sum_{i=1}^{k} r_{\sigma_i} \\
subject\ to\ \sum_{i=2}^{k} \mathcal{L}_D(q_{\sigma_{i-1}}, q_{\sigma_i}) \leq T_{max}, \\
q_{\sigma_i} = (t_{\sigma_i}, \theta_{\sigma_i}),\ t_{\sigma_i} \in S_k,\ \theta_{\sigma_i} \in \Theta, \\
\sigma_1 = 1, \sigma_k = n.
\end{gathered} \tag{25}$$

The DOP adds another dimension to the state space of the OP with optimization over heading angles which makes finding the solution even harder. Even though the introduction of the Dubins motion primitives into the OP enables finding usable trajectories for UAVs, the motion capabilities are not fully utilized as in [31] only a motion with constant velocity along the curve is allowed. Even though it might be enough for most touring problems it raises the time the UAV needs to traverse the trajectory. If the UAV is able to fly as fast as possible, it can be capable of visiting more targets in a given time. Another drawback is that the DOP can be used only for planar trajectories which is presumably even stronger limitation as one of the biggest advantages of UAV is the full motion in six degrees of freedom.

4.3 Hermite Orienteering Problem

One of the aims of this thesis is to remove the limitations of the current DOP solution, which are only planar trajectories, and insufficient utilization of reachable velocities by the UAV. As was noted in previous sections, the limitations are caused by the used motion primitive used in the DOP solution. Hence a new type of motion primitive was chosen which is the Hermite spline. The properties of this curve type are very suitable for use with motion model of UAV, namely the smoothness of the curve since the fluent flight trajectory is the main motivation for not using the Euclidean paths. The ease with which the curve can be parametrized in 2D as well as 3D space is the biggest advantage. The complete set of properties of the Hermite curves were described in detail in Chapter 2.

When introducing the Dubins curves to the OP the complexity of the searched state space had to be expanded by heading angle variable which made the already demanding problem even more complex. Even though the Hermite curves offer a lot of advantages, the drawback is that in order to generate a Hermite curve the target locations and heading vectors, which can be described as norm and heading angle, are needed. This means the OP have to be extended by another two dimensions which is the norm of the heading vector and the heading angle. This can be also considered as a limiting factor for the order of the parametric polynomial curve used which is three, since curves of the higher orders require even more parameters which would make the OP too complex. Luckily the third-order polynomial curves offer sufficient flexibility for creation of complex shapes.

As was noted earlier, one of the goals is to use wider repertoire of motion capabilities of the UAV. To achieve this goal a velocity profile is introduced in order to fly as fast as possible along the curve. This velocity profile is determined by maximal vertical and horizontal velocities and accelerations as was described in Chapter 3. Even though the norm of the heading vector of the used Hermite curve and actual speed of the UAV can be separated, it would mean another degree of freedom in the already wide state space. Hence these two variables are dependent through a constant multiplier. The consequences of this dependence are shown in Section 3.4. This also changes the evaluation factor of the trajectories to be the time of flight rather than the length.

This new variant of the OP was called Hermite Orienteering Problem (HOP) since the Hermite splines are used as the motion primitive. The HOP extends the DOP by another variable vector $\mathcal{V} = (v_{\sigma_1}, ..., v_{\sigma_k})$ which is the vector of actual velocities at given target locations. The needed information about the heading angle, velocity and location is defined as a state $q_{\sigma_i} = (t_{\sigma_i}, v_{\sigma_i}, \theta_{\sigma_i})$. The HOP utilizes an operator $\mathcal{T}_H(q_{\sigma_i}, q_{\sigma_j})$ which is the time of flight between two states q_i and q_j using Hermite curve as a motion primitive. The visiting order of the states is given by Σ, Θ is a set of used heading angles and $\mathcal{V}$ is a set of actual velocities at target locations. The Hermite Orienteering Problem can be described as following optimization problem:

$$\begin{gathered}\underset{k,S_K,\Sigma,\Theta,\mathcal{V}}{maximize}\; R=\sum_{i=1}^{k} r_{\sigma_i}\\ subject\ to\ \sum_{i=2}^{k}\mathcal{T}_H(q_{\sigma_{i-1}},q_{\sigma_i})\le T_{max},\\ q_{\sigma_i}=(t_{\sigma_i},v_{\sigma_i},\theta_{\sigma_1}),t_{\sigma_i}\in S_K,\ v_{\sigma_i}\in\mathcal{V},\ \theta_{\sigma_i}\in\Theta,\\ v_1=0,v_k=0,\\ \sigma_1=1,\sigma_k=n.\end{gathered}\tag{26}$$

5 Proposed Solution to HOP

The proposed solution to HOP is based on the Variable Neighborhood Search algorithm introduced by Mladenovi et al. in [18]. This algorithm can be used in wide range of combinatorial optimization problems and was successfully used to solve OP in [43]. The VNS algorithm operates on a given neighborhood structures N_l, where $l = \{l_1, ..., l_{max}\}$ is the maximal distance between solutions inside the neighborhood, which is a number of different targets visited in the OP solution. $N_l(x)$ is a neighborhood of a solution x with range l. The range of the neighborhood structure is determined by an operation that is used to modify the current solution x to a different solution x' inside the neighborhood. The VNS algorithm uses two types of procedures to search for new solutions, it is *shake* and *local search* procedure.

The *shake* procedure serves as a tool for random-walk like exploration of a given neighborhood. The current solution x is randomly moved to another x' inside the neighborhood. This operation is essential to escape from local maximum as the admissible distance towards other solutions is much greater than in the case of *local search*. The *local search* on the other hand serves as a mean to explore immediate neighboring solutions.

The VNS algorithm starts with a given initial solution and periodically applies the *shake* procedure to escape from possible local maximum and subsequently uses the *local search* operator to find the optima in vicinity. This is repeated until given terminating condition is met, such as runtime or number of iterations without improvement. The limiting factor is that algorithm implements only the combinatorial optimization, which means selecting the targets that will be visited and in what order. Hence if it is needed to optimize continuous variables such as the heading angle at the targets, the VNS algorithm needs to be extended.

The problem of continuous optimization while using the VNS algorithm was addressed in [31], where the VNS algorithm was successfully applied to DOP problem described in Section 4.2. The continuous optimization part of the DOP, lies in determination of the optimal heading angles at each of the targets in order to construct Dubins curves of the shortest length. This was solved simply by using only discrete samples of possible angles at each target location. Drawback of this approach is that in order to find a solution of good quality, it is better to use high sampling rates which dramatically enlarges the complexity of the state space. This problem was addressed by introduction of several simplifying heuristics such as using only targets located in ellipse around start and end location with axis defined by the maximal budget for the length of the solution. This filters out targets which violate the budget constraint and would not be used in any possible solution. Another simplification that was used is the randomization of the *local search*. This means that during the local exploration of the neighborhood is not explored systematically, which is very demanding with large amount of samples, but rather randomly using given number of iterations. This variant of VNS is called Randomized Variable Neighborhood Search

(RVNS) and it was shown that it is able to find solutions of the same rewards while being significantly faster than normal VNS.[43].

The RVNS algorithm introduces several operators which can be used in the context of the OP. The operators are:

- *shake* operators:
  - **Path Move**
  - **Path Exchange**
- *local search* operators:
  - **Point Move**
  - **Point Exchange**.

All of these operators modify the subset S_k of visited targets as well as order of visit Σ. The difference is the scope of the change.

Even though the sampling based RVNS is usable in the case of DOP, the HOP is more complex as there are two continuous variables that need to be optimized - the heading angle and the actual velocity. This means that the total amount of samples is given by Kartezian product of both types of samples. Such total number of samples can grow quite fast for even small sampling resolutions which greatly limits the capability of finding a good quality solution in reasonable time as the combinatorial complexity is too large.

This leads to the introduction of new operators **Angle Speed Shake** and **Improve Angle Speed**. These operators solve the continuous optimization part of the problem and enable to explore neighboring solutions even outside of the scope defined by initially sampled velocity and heading angle. The **Angle Speed Shake** introduces new samples of heading angle and velocity by random shaking, while the **Improve Angle Speed** systematically improves the solution by implementing a simple hill climbing algorithm when introducing new samples to the state space. This enables to accept even shake solutions that slightly violate the budget as these can be further improved by continuous optimization to the point where the budget constraint is no longer violated. The biggest advantage of continuous optimization is a fact that the initial sampling rates can be fairly low as new promising samples are added during the optimization process. The whole RVNS algorithm used to solve HOP is described in detail in following sections.

5.1 RVNS algorithm

The Random Neighborhood Variable Search algorithm works with Neighborhood structures N_l with defined maximal l distance between solutions inside the neighborhood.

The distance l is defined by the operations used to search through the neighborhood structures by modifying the current solution. The solution of HOP is internally represented in RVNS by a solution vector $v = (s_{\sigma_1}, ..., s_{\sigma_k}, s_{\sigma_{k+1}}, ...s_{\sigma_n})$, which holds all the possible targets s_i. The order of the targets in the vector corresponds with the order of visit during flight with first k being included in the solution with order given by Σ. The other $n-k$ states are not used (above budget), but since they are still in the solution vector the neighborhood operators can include these unused targets in possible new solutions. Simply said, all the operators just modify the locations of the elements inside the vector.

The Chapter 3 describes how the velocity profile is calculated in order to estimate the time of flight needed to traverse given Hermite curve. This process, especially for larger amount of samples per curve, can be quite time demanding if working with hundreds of thousand possible curves, which is not uncommon even with fairly low initial sampling rates in angle and velocity. This leads to a definition of a structure which holds the pre-calculated time of flight estimates between all possible states in order to significantly improve the performance of neighborhood operators. Let's call this structure *allDistances*. The drawback is that this structure can be quite large, so it preferable to generated this structure before the actual algorithm starts. The *allDistances* structure also needs to be appropriately maintained as new samples are added during continuous optimization operations. Lastly, as was noted in the Chapter 3, it is possible that the curve given by two states is not feasible as it would violate the motion limitations of the UAV. Such combinations are marked unfeasible and are not used in further optimization. It is important to mention that the number of unfeasible combinations can be significant.

As the neighborhood operators modify the order and number of visited target location and their order in the solution vector, the combination of velocity and heading angle samples which generates the shortest trajectory needs to be found. This requires a search through the *allDistances* structure, which can be thought of as searching through a subgraph with vertices being the heading angle and velocity samples. In addition these vertices are grouped together by the location to which they belong to. The edges of the graph can run only between vertices of neighboring groups (target locations) in the solution. An example of such search graph is shown in Figure 10. The shortest path needs to be found using brute force methods in order to check all possible combinations which is very time demanding. The resulting path consists of best combinations of samples at each target location which yield the shortest trajectory. In order to make the process of searching through the possible combinations of samples faster, the structures *shortestToStart* and *shortestToEnd* are introduced. These two structures hold a list of currently used targets in the solution and their respective samples with information about the current distance toward starting or ending location along with the id of neighboring target sample which is the best option for getting closer to the start or end. This effectively means that the exhaustive search through *allDistances* structure can be performed only once at the beginning of the algorithm when the initial solution if provided and the best combination of samples is saved into these structures. When the solution is changed by the neighborhood operators, the *allDistances* structure does not have to be searched again as it is enough to correspondingly update the *shortestToStart* and *shortestToEnd* structures which is much faster.

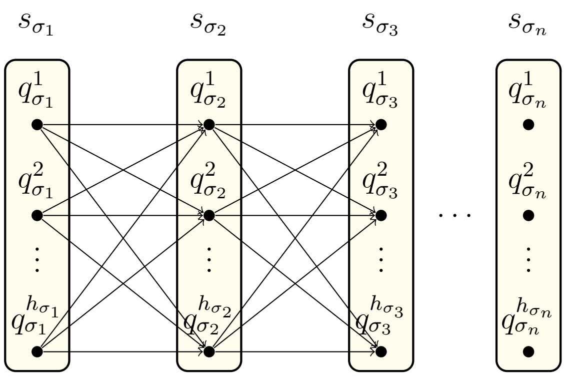


Figure 10: Search graph used to find the best combination of samples in the current solution vector $v = (s_{\sigma_1}, s_{\sigma_2}, s_{\sigma_3}, ..., s_{\sigma_n})$. The sample q^j_i is defined by given value of heading angle and velocity

To start the whole RVNS algorithm an initial solution is needed. Before this solution is generated the unreachable targets are filtered out. The target is qualified as unreachable if it's so far away from the starting target that even if the UAV would fly with maximal possible velocity the whole time, it would not be able to reach this target without violating the maximal budget condition. This simple operation can have large impact on performance especially for scenarios with very short budget limit as most of the samples will be filtered out. This lowers the combinatorial complexity significantly. The initial solution is then created using simple greedy heuristic. The algorithm searches through the elements of the solution vector $v = (s_{\sigma_1}, ..., s_{\sigma_k}, s_{\sigma_{k+1}}, ...s_{\sigma_n})$ which are not currently used in the solution (indexes k+1...n). Each element is experimentally added to each position in current solution and the situation with the minimal added time of flight per reward is accepted and solution is extended by this target. This is repeated until the budget is hit or all targets are included. The additional time of flight per reward measure (TPR) is calculated using the reward of added target R_{add} and TOFs of old and new solution $T_{new},\ T_{old}$ as:

$$TPR = \frac{T_{new} - T_{old}}{R_{add}}$$

This solution can be optionally improved further by the continuous optimization operators described in Section 5.3. With the initial solution found, the RVNS algorithm can start. The whole process is described in Algorithm 3. The operator that is used for given value of l can be found in Table 2..

The current best solution is modified by applying the *shake* procedure which enables to escape from possible local maximum and then the *localSearch* procedure to finely explore the imminent neighborhood of the solution. If the reward of the new solution is

Algorithm 3 RVNS algorithm for HOP solution

```
$S$: set of target locations
$T_{max}$: maximal time budget
$l$: neighborhood scope defining the type of neighborhood operator, $l \in \langle 1, 3 \rangle$
$l_{max}$: maximal value of $l = 3$
$P$: the best solution path found
R(P): collected reward of path P
$\mathcal{T}_H(P)$: time of flight of path P
Output: The best found path P

$S_r \leftarrow getReachableLocations(S, T_{max})$
$P \leftarrow createInitialSolution(S_r, T_{max})$
while $Stopping\ condition\ not\ met$ do
  $l \leftarrow 1$
  while $l \leq l_{max}$ do
    $P' \leftarrow shake(P, l)$
    $P'' \leftarrow localSearch(P', l)$
    if $[\mathcal{T}_H(P'') \leq T_{max}$ and$R(P'') > R(P)]$ or $[R(P'') == R(P)$ and $\mathcal{T}_H(P'') < \mathcal{T}_H(P)]$
    then
      $P \leftarrow P''$
      $l \leftarrow 1$
    else
      $l \leftarrow l + 1$
    end if
  end while
end while
return P
```

Table 2: All operators used in the RVNS algorithm for HOP

l	*shake*	*local_search*
1	Angle Speed Shake	Improve Angle Speed
2	Path Move	Point Move
3	Path Exchange	Point Exhange

higher than reward of the currently best solution, it is accepted as the new best solution and used in further optimization. If the reward of the new solution is the same as the current best reward, the time of flight becomes the deciding measure. In the case of the time budget being large enough to visit all targets, this condition enables to perform the optimization in the sense of Traveling Salesman Problem. If the newly found solution is worse than currently best solution the process is repeated using *shake* and *localSearch* operations with larger scope of neighborhood structures that can be explored (defined by l variable in the algorithm). The continuous optimization part of the problem is embodied within this algorithm as both *shake* and *localSearch* operations each containing one operator used for continuous optimization. These are described in Section 5.2.2.

Whole process continues until one of the stopping conditions is met. This can be defined arbitrarily. In experiments performed in this thesis a maximal run time along with maximal number of iterations and a number of iterations without improvement were used as a stopping conditions. The operators used to find new solutions are described in following sections.

5.2 Combinatorial operators

The combinatorial operators are used to improve the current solution by adding and removing targets in the solution and changing the order of visit. After each manipulation, the shortest trajectory through all targets is found using currently known samples of heading angle and velocity using *shortestToStart* and *shortestToEnd* updated structures. The combinatorial operators work with solution vector defined in Section 5.1. Only first k members are considered as a part of the current solution. This enables to introduce new targets into the current solution as the operator do not distinguish between used and omitted targets. Both *shake* and *local search* operators are based on random exploration.

5.2.1 *shake* operators

shake operators are used to explore the neighborhood structures with large possible distance between solutions. This enables to escape from local maxima and can change the current solution dramatically. These operators are:

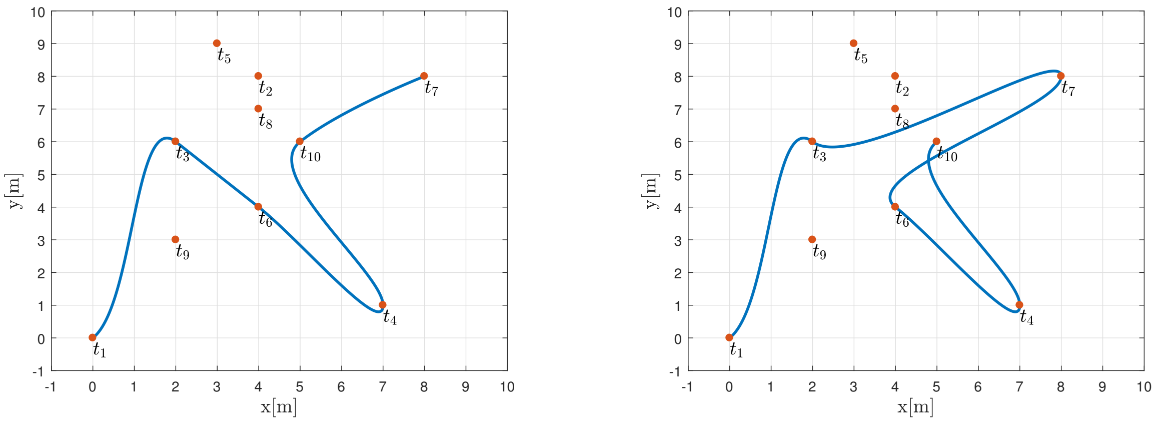
- **Path Move** ($l = 2$): This operator selects a random path inside the solution vector v and changes its position inside the vector. Even though the distance l was defined as a number of different targets between two solutions and change introduced by this operator typically changes more than two targets, it was still assigned this value for the sake of simplicity as the meaning of l can be viewed as symbolical measure (the reason is apparent when looking at the Algorithm 3 since if the values of l are unified for *shake* and *local search* operators, it is easy to apply both operations of a given type and similar scope using smaller number of different values of l). The operator itself performs the changes in the current solution by selecting three random indexes inside the vector $v = (s_{\sigma_1}, ..., s_{\sigma_n})$ such as:

$$\begin{aligned} &i_1 \in \langle 2, n-1 \rangle, \\ &i_2 \in \langle i_1 + 1, n-1 \rangle, \\ &i_3 \in \langle 2, i_1) \bigcup (i_2, n-1 \rangle. \end{aligned} \tag{27}$$

The initial start and end locations are not affected. Index i_1 and i_2 define the path that is going to be moved and index i_3 defines its new position inside the vector v. The example of modified solution vector for the case $i_3 > i_2$ then equals to:

$$v = (s_{\sigma_1}, ..., s_{\sigma_{i_1 - 1}}, s_{\sigma_{i_2+1}}, ..., s_{\sigma_{i_3}}, s_{\sigma_{i_1}}, ..., s_{\sigma_{i_2}}, s_{\sigma_{i_3+1}}, ..., s_{\sigma_n}) \tag{28}$$

An example of this operation is pictured in Figure 11.



(a) Path before operation:
$v = (s_1, s_3, s_6, s_4, s_{10}, s_7, s_9, s_5, s_2, s_8)$
Move path $\{s_6, s_4, s_{10}\}$ to index 7

(b) Pathe after operation:
$v = (s_1, s_3, s_7, s_6, s_4, s_{10}, s_9, s_5, s_2, s_8)$

Figure 11: An example of **Path Move** operation

As was noted in the Section 5.1, there are two structures *shortestToStart* and *shortestToEnd* that make finding the shortest path from all possible samples of currently used target locations much faster. These structures need to be maintained when the current solution is changed. In the case of **Path Move** operator the modification is similar to the changes performed on the solution vector. This means in practice that only samples between states that changed their neighbors inside the solution vector need to be updated by search through all possible angle and velocity combinations with the new neighboring locations.

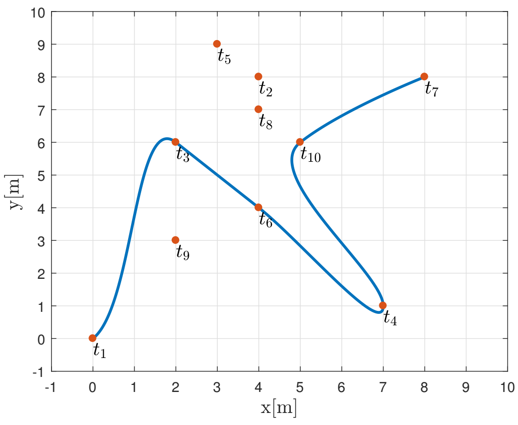
- **Path Exchange** ($l = 3$): This operator tries to find solution even further away than **Path Move** in terms of solution distance. As the name suggest, in this case there are two paths selected in the solution vector and their positions is switched. This can be performed by randomly selecting four indexes, but these need to be selected in feasible manner, which means the selected paths can't be overlapping:

$$\begin{aligned} i_1 &\in \langle 2, n-1 \rangle, \\ i_2 &\in \langle i_1 + 1, n-1 \rangle, \\ i_3 &\in \langle i_2 + 1, n-1 \rangle, \\ i_4 &\in \langle i_3 + 1, n-1 \rangle. \end{aligned} \tag{29}$$

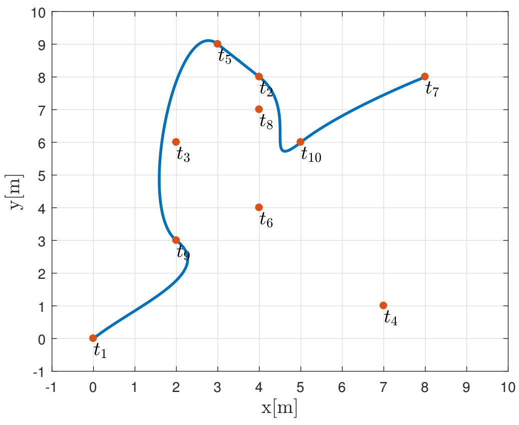
The modified solution vector v than equals to:

$$v = (s_{\sigma_1}, ..., s_{\sigma_{i_1-1}}, s_{\sigma_{i_3}}, ..., s_{\sigma_{i_4}}, s_{\sigma_{i_2+1}}, ..., s_{\sigma_{i_3-1}}, s_{\sigma_{i_1}}, ..., s_{\sigma_{i_2}}, s_{\sigma_{i_4+1}}, ..., s_{\sigma_n}). \qquad (30)$$

An example of this operation is shown in Figure 12.



(a) Path before operation:
$v = (s_1, s_3, s_6, s_4, s_{10}, s_7, s_9, s_5, s_2, s_8)$
Exchange path $\{s_3, s_6, s_4\}$ with $\{s_9, s_5, s_2\}$



(b) Path after operation:
$v = (s_1, s_9, s_5, s_2, s_4, s_{10}, s_3, s_6, s_4, s_8)$

Figure 12: An example of **Path Exchange** operation

5.2.2 *local search* operators

Local search operators aim to explore the imminent neighborhood of the current solution searching for local optima. This is done by modifications that change the solution in smaller scope than the shake operators (the difference l between solutions will be smaller). In addition, since the used algorithm is not pure VNS but rather RVNS, the local search is also randomized. The operators apply only very simple changes to the solution vector so it is possible to run multiple iterations in one optimization step. This process practically emulates VNS where this local search would be some systematic procedure determining the local optimum, although typically more resource demanding than randomized hill climbing. The number of iterations during which the same local search operator is repeatedly applied is equal to the square root of reachable targets. Each time the operation creates a better solution, the change is accepted and further improvements continue with this new solution - this is typically called a stochastic hill climbing algorithm [37]. The local search operators are:

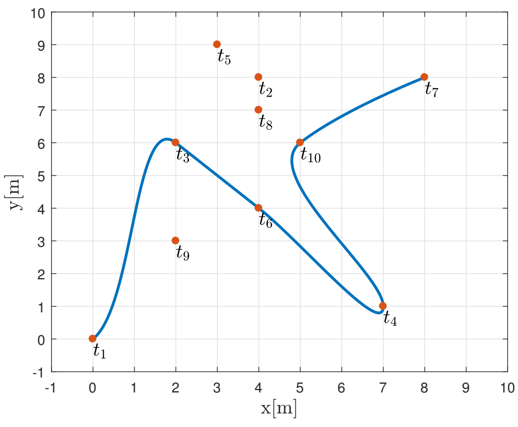
- **Point Move** ($l = 2$): This operator simply randomly selects one of the elements in the solution vector and moves it to another position. This is performed by selecting only two random indexes such as:

$$i_1 \in \langle 2, n-1 \rangle, \\ i_2 \in \langle 2, i_1) \bigcup (i_1, n-1 \rangle. \tag{31}$$

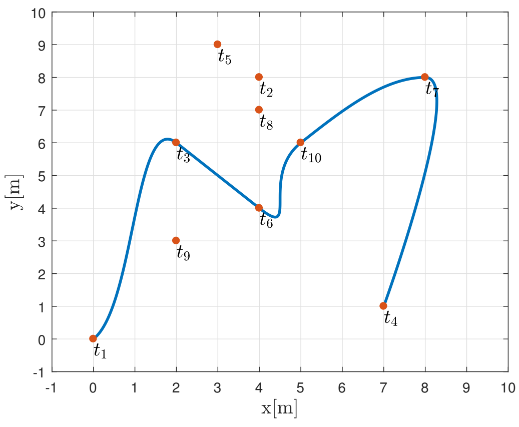
The newly acquired solution vector for the case where $i_1 < i_2$ is then:

$$v = (q_{\sigma_{i_1}}, ..., q_{\sigma_{i_1-1}}, q_{\sigma_{i_1+1}}, ..., q_{\sigma_{i_2-1}}, q_{\sigma_{i_1}}, q_{\sigma_{i_2+1}}, ..., q_{\sigma_n})$$

. An example of this operation is shown in Figure 13.



(a) Path before operation:
$v = (t_1, t_3, t_6, t_4, t_{10}, t_7, t_9, t_5, t_2, t_8)$
Move point t_4 to index 7



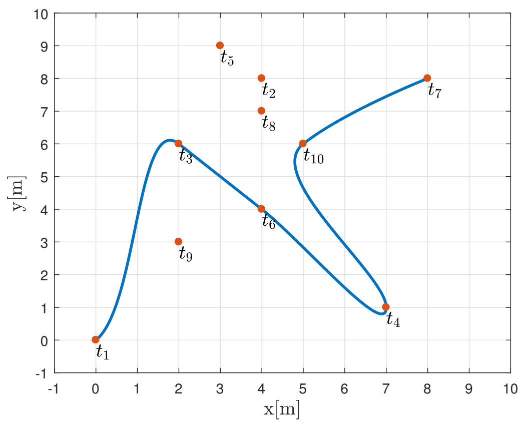
(b) Path after operation:
$v = (t_1, t_3, t_6, t_{10}, t_7, t_4, t_9, t_5, t_2, t_8)$

Figure 13: An example of **Point Move** operation

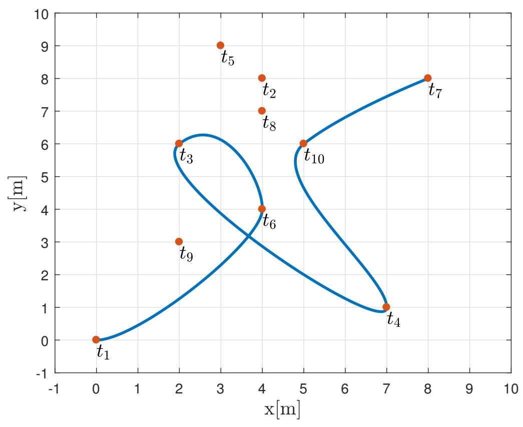
- **Point Exchange** ($l = 3$): This operator exchanges positions of two randomly selected elements in the solution vector. This is again implemented as random selection of two indexes but the solution vector is modified in different manner:

$$v = (q_{\sigma_{i_1}}, ..., q_{\sigma_{i_1-1}}, q_{\sigma_{i_2}}, q_{\sigma_{i_1+1}}, ..., q_{\sigma_{i_2-1}}, q_{\sigma_{i_1}}, q_{\sigma_{i_2+1}}, ..., q_{\sigma_n})$$

. An example of this operation is shown in Figure 14.



(a) Path before operation:
$v = (t_1, t_3, t_6, t_4, t_{10}, t_7, t_9, t_5, t_2, t_8)$
Exchange point t_3 with point t_6



(b) Path after operation:
$v = (t_1, t_6, t_3, t_4, t_{10}, t_7, t_9, t_5, t_2, t_8)$

Figure 14: An example of **Point Exchange** operation

In addition, when performing the local search and a new target is introduced in the solution, it is possible to apply the continuous optimization methods to improve heading angles and velocities. This can further raise the quality of found solution during local search and also introduce new samples for angle and velocity, opening more options in subsequent iterations. However the process of continuous optimization is quite time demanding when working with Hermite curves as motion primitives. The number of iterations inside the local search can be large so the whole algorithm performance is negatively impacted. Because of this reason the continuous optimization was not used in solution proposed in this thesis during the local search operations Path Move and Path Exchange.

5.3 Continuous optimization operators

The continuous optimization part of the HOP problem, which is related to the selection of suitable heading angle and velocity, is tackled by introduction of two special operators **Angle Speed Shake** and **Improve Angle Speed**. Both operators can introduce new samples for angle and velocity into the full set of all samples which is used in combinatorial optimization and effectively decrease the needed time of flight for given combination of target locations. This enables to start the algorithm with relatively low density sampling which makes the initialization of the algorithm, construction of initial solution and all combinatorial operations much faster. Used operators again follow the shake and local search design.

- **Angle Speed Shake** ($l = 1$): This operator performs the shake operation upon velocities and heading angles. That means randomly changing values for these two variables which might be different from already used samples. To further lower the complexity of the problem, the sampling of heading angle is performed only on planar disk around each target even in 3D variant of the problem, even though in 3D all possible heading angles are given by samples on sphere.This defines the heading angle interval as $\langle 0, 2\pi))$. This reduces the complexity while still offering reasonably good extent of possible states. A disadvantage of this technique is a fact that the heading vector of appropriate Hermite curve determined by heading angle will always have the z component (vertical axis) equal to zero, which effectively means the vertical velocity in each target location should be zero. This seemed to raise the number of unfeasible Hermite trajectories and is a candidate for further improvement. The velocity samples are limited by interval $\langle 0, v_{h_{max}} \rangle$, as the only contributing component is the horizontal velocity at each target location. The solution modified by this shake operation is then improved by following local search operator and if the new solution has better reward all the newly introduced samples are added to the set of all available samples and can be used in the combinatorial optimization part of the problem.

- **Improve Angle Speed** ($l = 1$): This operator performs the local search in terms of continuous optimization. This again involves stochastic hill climbing technique. The admissible interval for velocity and heading angle stays the same as in the case of Angle Speed Shake. The procedure that is used for both variables at each target location starts with given step which is a fraction of maximal used sampling rate over the whole interval. This step is added to the current value of a given variable and this newly created solution is checked for improvement in time of flight and feasibility. If the new solution is shorter, the step size is doubled and the process is repeated. If the solution is worse, the step size is halved and negated which changes the direction of exploration and makes the resolution finer. This is repeated until maximal or minimal values of possible resolution are reached by the step size. If the step operation violates the limits given by feasible intervals the new value is normalized to lie in the admissible interval (by overflowing and underflowing). In addition this whole process is repeated for each target location in current solution until given number of iterations without improvement is reached. This process has a lot of nested cycles incorporated hence it is quite time demanding for large number of targets but the elementary changes performed affect only the target itself and its imminent neighbors in the solution. This means only two Hermite curves need to be constructed at most in order to evaluate the new time of flight, which is not that challenging.

  Even though the Improve Angle Speed procedure is defined as a neighborhood operator it can be used anywhere in the algorithm to optimize crude solutions. In the implementation of this thesis this operator is used after the creation of initial solution and then after each combinatorial operation which finds a better solution. It can be even used in the combinatorial operations in order to admit solution that slightly violates the maximal budget which might be optimized by Improve Angle Speed so that this solution becomes feasible.

6 Results

The HOP is considered to be NP-hard problem, hence it is very time and performance demanding to find a quality solution even with the use of heuristic methods. This limits the usability of the implementation in real time planning scenarios. The HOP solution implementation can be run directly on board of the UAV, but since the output is the trajectory generated in a suitable format for the MPC tracker to follow, the solution can be as well generated off board with more processing power with only the resulting trajectories being sent to the UAV. This leads to definition of two types of experimental setups.

First set of experiments aims to show the computational capabilities and performance of the HOP solution algorithm. Since the solution is generated in a stochastic way, a large number of tests is needed in order to get somewhat meaningful statistic sample. To achieve this goal, the computational grid services provided by the organization Metacentrum were utilized [1]. This service enables to run the implementation on fast Intel Xenon processors. First test shows the average and maximal collected rewards for three different scenarios depending on the travel budget along with the calculation time needed. This might serve as a comparison benchmark for other approaches to the OP. The second test aims to show how the initial velocity and heading angle sampling rates influence the quality of found solutions along with computational complexity. Another parameter explored in this set of tests is the heading multiplier h_m defined in the Section 3.4. The value of this parameter significantly modifies the shape of generated Hermite spline, so also its impact on the collected reward was tested. During these tests an optimal value of heading multiplier was found for each used scenario. In the last set of tests the HOP solution is compared to EOP and DOP. Both these solution use the maximal length as the budget constraint, hence both approaches have been slightly modified so it is possible to compare them with the time based HOP solution somewhat fairly. All computational results are described in the Section 6.1.

The second experimental setup verifies the suitability and feasibility of the resulting trajectories for the UAV. These are created with characteristics defined by the motion model of a real UAV, which is used to fly through the generated trajectory and verify the correctness of the design. This prototype was built by Multi Robotics Systems research group [2] as an experimental platform for wide range of UAV themed research projects. It is based on DJI hexacopter F550 commercially sold UAV frame with E310 DJI motors along with PixHawk Autpilot low level controller. A lot of other support systems were added to this platform, with Intel NUC-i7 mini PC being at the hearth of the system serving as a fully operational on board computer. This processor offers enough computational resources even for performance demanding procedures and along with mobile graphical chips such as NVIDIA Jetson TX2 can be used for tasks such as real time computer vision and SLAM [29]. In addition a lot of different sensory equipment is used such as TeraRanger One laser rangefinder for the distance measurements, Intel RealSense IR camera, high-resolution FOV Mobius ActionCam or Scanse Sweep LIDAR. A precise differential

GPS PRECIS-BX305 is used to enable localization accuracy in magnitude of centimeters. The functionality of all the subsystems are fused together using Robotic Operating System [3]. This collection of software frameworks for robot related software development offers tools for hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes and package management. Several HOP scenarios were created showing the capabilities of the solution tested on the UAV in real conditions. Results are described in the Section 6.2.

6.1 Computational results

Most of the results were performed on a scenario created for several UAV applications by the MRS research group. The map showing the locations and rewards of the target locations in this scenario is shown in Figure 15. This scenario will be called MBZIRC as it was used in the international UAV competition with the same name. The main advantage of the MBZIRC scenario is that it can be easily built on the premises of testing range used for the real UAV testing. Furthermore all of the tests are performed in plane so it is possible to compare the results with existing solutions which are all in 2D.

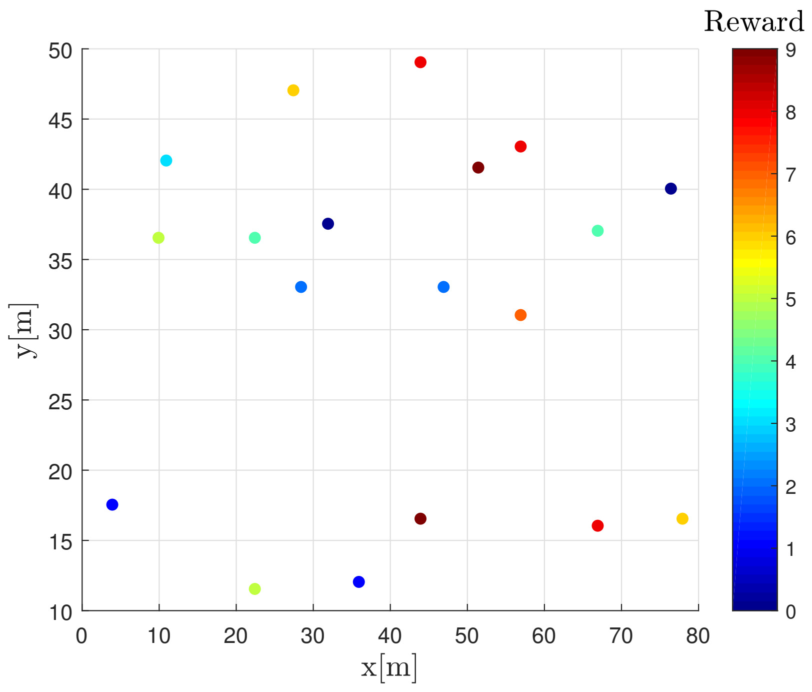
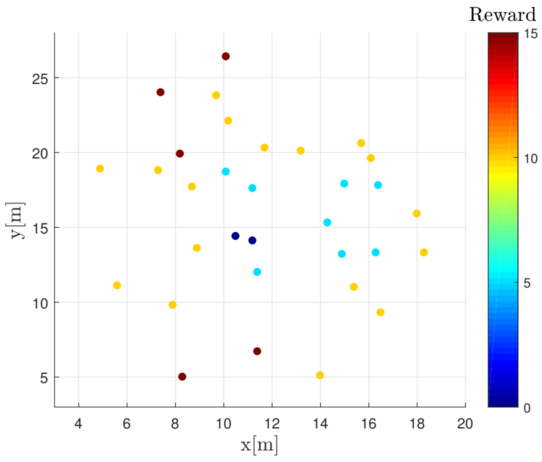
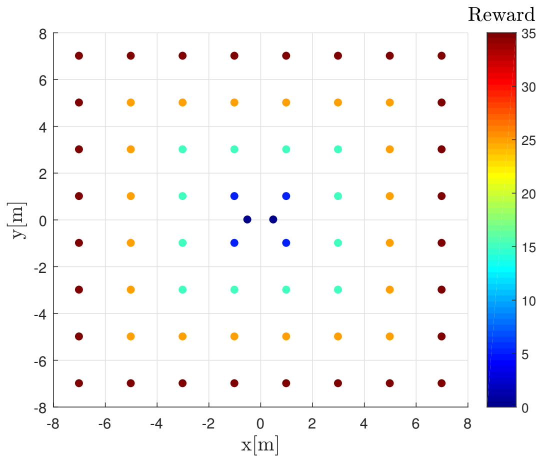


Figure 15: MBZIRC scenario map showing the topology of target locations with color coded reward for visiting each target with maximal collectible reward of R=88

The first computational test performed, can serve as a benchmark for further projects on similar topics. The HOP solution implementation was run for several different budgets with collected reward being the examined result. Since the results are stochastic, each test with given maximal budget was run ten times and the average and maximal collected reward was determined. This test was performed using three different scenario maps. First tested scenario is the given by the MBZIRC data set shown in Figure 15. The second used scenario was introduced by Tsiligirides in [46]. He defined three scenarios containing up to 32 target locations each, naming them Set 1, Set 2 and Set 3. The scenario named Set 1 is used in further testing and will be referred to as Tsiligirides scenario. The third used scenario was defined by Chao in [10] and contains 66 target locations. This scenario will be called Set66. The topologies of target locations along with respective rewards for Tsiligirides and Set66 scenarios are shown in Figure 16.



(a) Tsiligirides with maximal collectible reward R=285



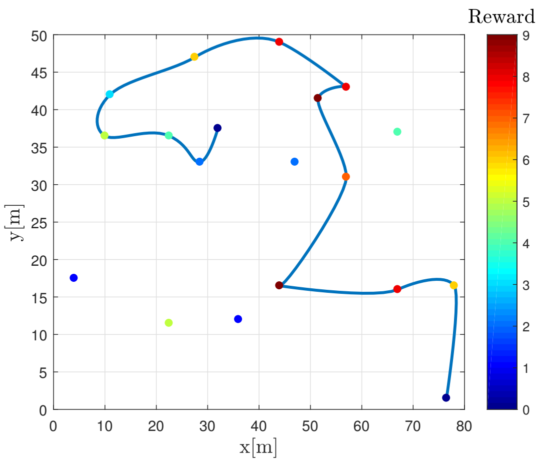
(b) Set66 with maximal collectible reward R=1680

Figure 16: Tsligirides and Set66 scenario maps showing the topology of target locations with color coded reward for visiting each target

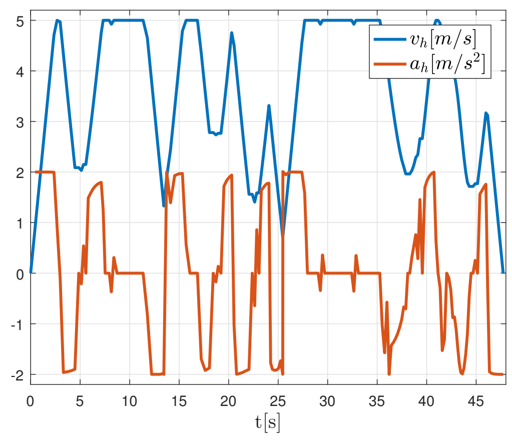
The resulting collected average and maximal rewards with respect to the maximal travel budget for each scenario are shown in Table 3. Examples of solutions for each scenario are shown in Figure 17 and 18.

Table 3: The collected average and maximal collected reward based on the maximal budget for different HOP scenarios, T_{max} - maximal traveling time budget, R_{Avg} - average collected reward in ten runs, R_{Max} - maximal collected reward in ten runs, t_{Avg} - average time of reaching the maximal achieved reward in ten runs. All tests were performed using UAV limitations $v_{h_{max}} = 5m/s$ and $a_{h_{max}} = 2m/s^2$.

$T_{max}[s]$	MBZIRC			Tsiligirides			Set66		
	R_{Avg}	R_{Max}	$t_{Avg}[s]$	R_{Avg}	R_{Max}	$t_{Avg}[s]$	R_{Avg}	R_{Max}	$t_{Avg}[s]$
25	42.0	42	17.3	166.5	185	628.9	668.5	745	3210.2
30	52.0	52	156.2	211.5	215	1297.7	748.5	855	1635.0
35	58.0	58	17.9	234.0	240	3011.6	868.5	935	2936.0
40	65.0	65	2262.8	254.4	265	3205.6	1059.5	1145	2449.0
45	72.6	73	170.3	273.0	280	1734.6	1191.5	1255	3845.8
50	77.0	77	41.0	284.0	285	958.9	1350.0	1415	5916.5
55	81.0	81	21.3	285.0	285	104.0	1430.0	1485	4423.8
60	85.8	87	280.4	285.0	285	66.2	1465.5	1565	4143.8
65	87.7	88	1413.2	285.0	285	56.3	1576.0	1650	3182.2
70	88.0	88	18.3	285.0	285	68.1	1613.0	1650	2979.3
75	88.0	88	521.4	285.0	285	324.3	1669.0	1680	3032.2
80	88.0	88	147.7	285.0	285	124.7	1679.0	1680	2599.6
85	88.0	88	57.2	285.0	285	789.1	1680.0	1680	3750.9

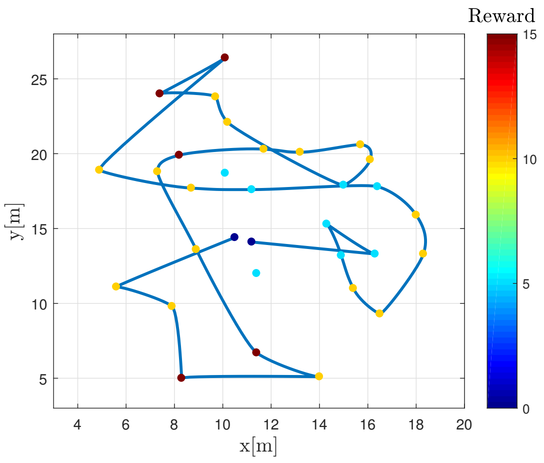


(a) MBZIRC scenario HOP solution

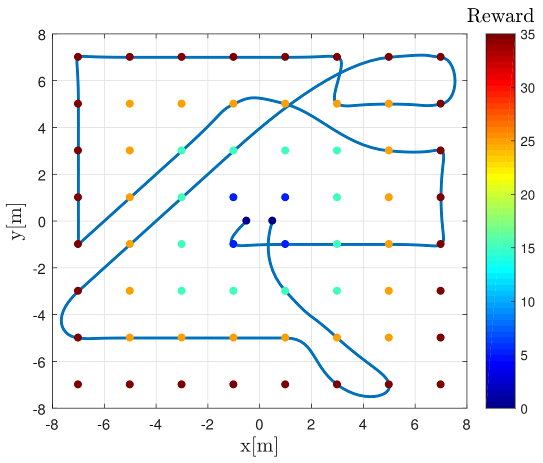


(b) Velocity and acceleration profile

Figure 17: An example solution of MBZIRC scenario with time budget $T_{max} = 50s$, maximal horizontal acceleration $a_{h_{max}} = 2m/s^2$, maximal horizontal velocity $v_{h_{max}} = 5m/s$ along with calculated velocity and acceleration profile



(a) Tsiligirides



(b) Set66

Figure 18: Example HOP solutions of Tsiligirides and Set66 scenarios with time budget $T_{max} = 50s$, maximal horizontal acceleration $a_{h_{max}} = 2m/s^2$, maximal horizontal velocity $v_{h_{max}} = 5m/s$

Apart from the maximal budget, the parameters influencing the algorithm are the initial sampling rates of the heading angle and actual velocity at target locations. The denser the sampling is, the better initial solution can be found, but the calculation time is larger. Two tests were designed to explore the influence of initial sampling rates on the performance. First test aims to find the average gathered reward based on used sampling rates. All rewards were again averaged over ten runs with the same settings and all test were performed using MBZIRC scenario. The results are pictured in Figure 19. The graph shows that the quality of the solution is almost indifferent to the initial sampling rate that was used. Only the very low sampling rate cases (1-2 samples per location) show slightly worse performance. This is very positive results, as it shows that it is really sufficient to use lower density initial sampling rates such as three to four samples per target location per variable, since the quality of the final solution will be unchanged compared to the solution obtained using higher sampling rates. This is the main motivation for introduction of continuous optimization operators in Section 5.3 and this result shows that the approach is successful.

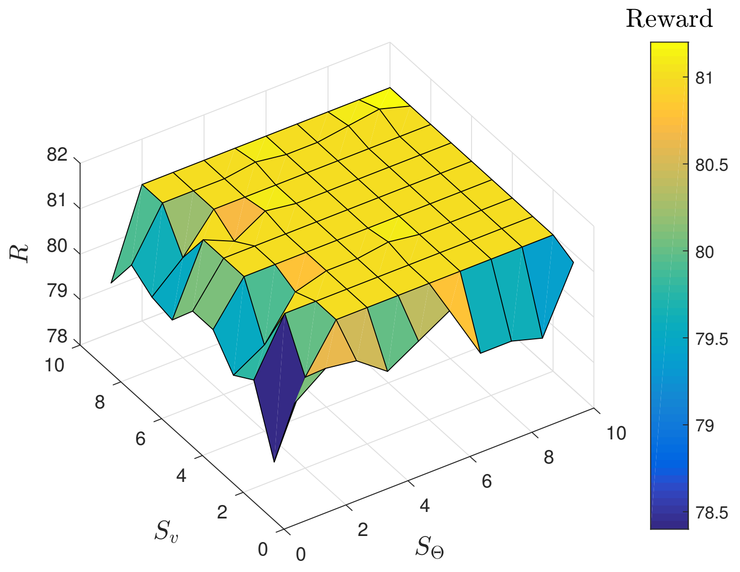


Figure 19: Average collected reward in MBZIRC scenario based on used initial sampling rates, R - average gathered reward, S_v - number of velocity samples per target, S_Θ - number of heading angle samples per target

Since it was shown it doesn't really matter which initial sampling is chosen in order to find good quality solution, the other factor that should offer sufficient insight on the suitable strategy for sampling rate determination is the time performance. The average time of reaching the value of the maximal collected reward during the run for given sampling rates is shown in Figure 20. Even though there are some outliers throughout the graph, there is overall tendency for the calculation times to be higher in sections of either very low sampling rates or very high sampling rate. In the case of low sampling rates, the higher calculation time is clearly caused by low number of possible combinations that can be used by the algorithm. It takes significant number of iterations until new samples are introduced. This also causes the lower average collected reward. In the case of high sampling rates, the longer calculation times are caused by too many samples to iterate through. As was noted in the Section 5.1 one of the biggest performance hits is the calculation of the *allDistance* structure before the actual algorithm starts. For high sampling rates this initialization takes much more time compared to low sampling rates. The Figure 21 shows this initialization time for given sampling rates. To find just the pure optimization time performance, the average initialization time is subtracted from the average time of reaching the maximal achieved reward. This situation is shown in the Figure 22. It can be noticed, that the maximas, especially in high sampling areas, are bit lowered but the overall tendency stays the same. This is another evidence that it is beneficial to use fairly small initial sampling rates in order to find solution of good quality as fast as possible, thanks to the introduction of the continuous optimization.

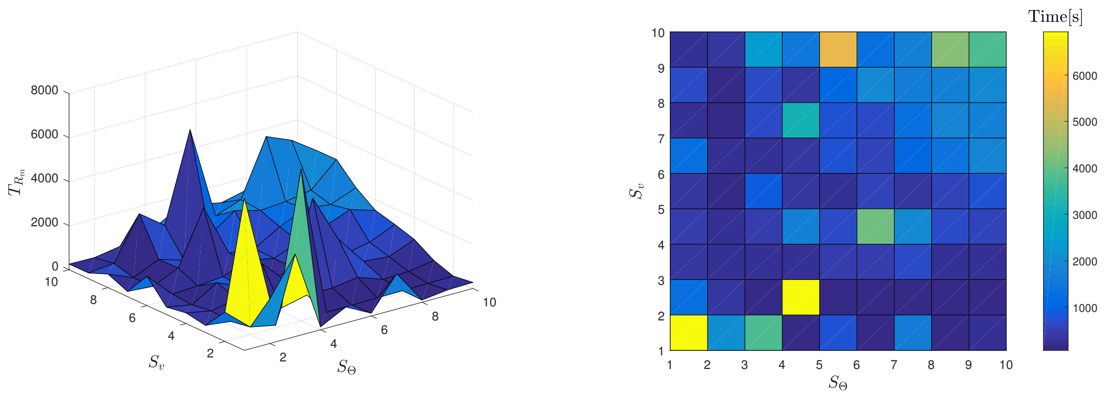


Figure 20: Average time of finding the first solution of maximal reached reward based on chosen initial sampling rate

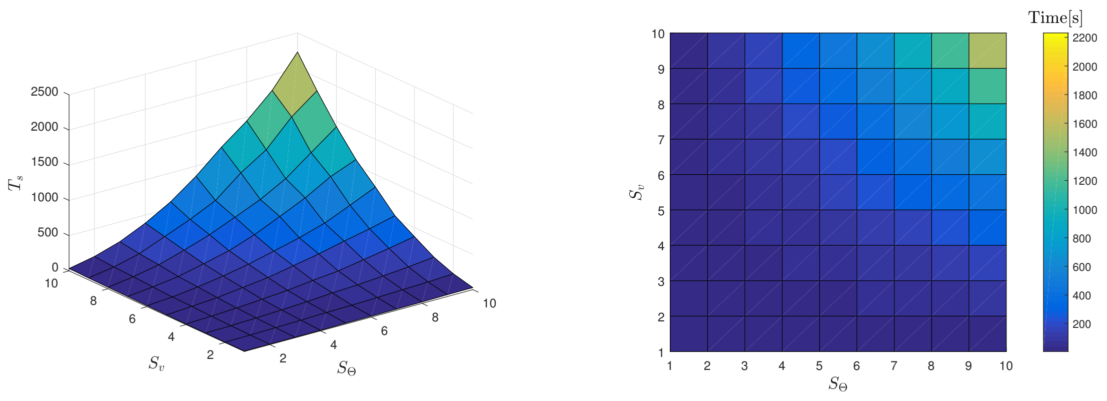


Figure 21: Average initialization time in seconds based on used sampling rate

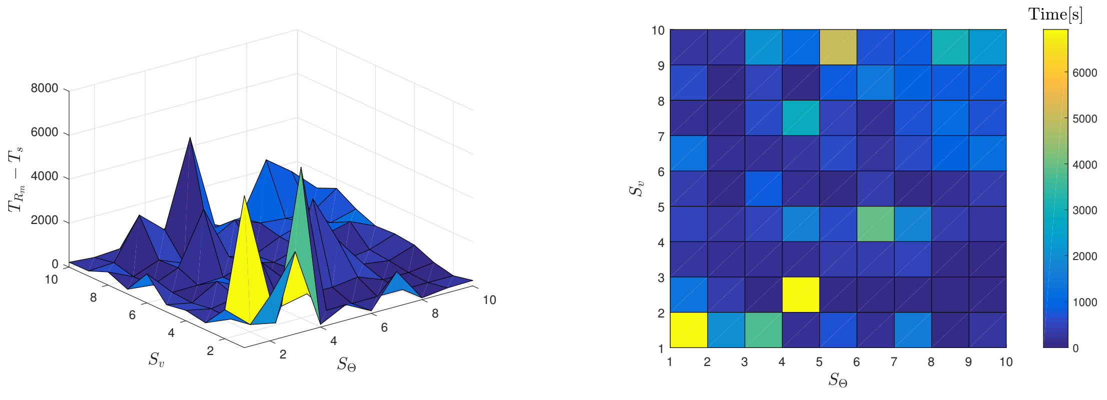
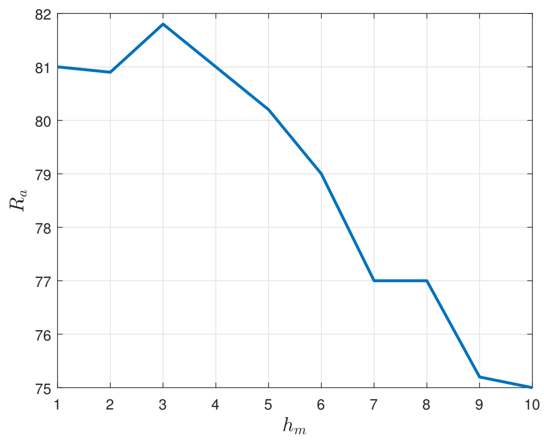
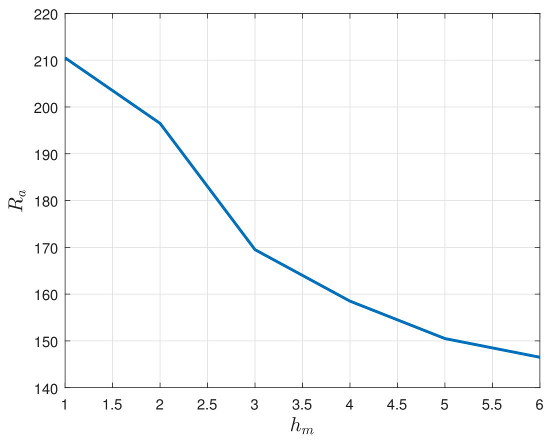


Figure 22: Average time of finding the first solution of maximal reached reward based on chosen initial sampling rate reduced by time needed to initialize the whole algorithm

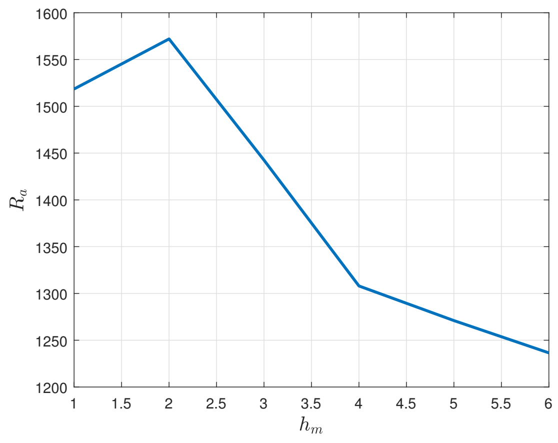
The last parameter that will be explored is the Hermite curve heading vector multiplier h_m described in the Section 3.4. This parameter has a big impact on the resulting shape of the trajectory, but the optimal value differs for each scenario as it is dependent on the topology of the target locations. Unfortunately no analytical formula for choosing the optimal heading multiplier was found, hence an iterative process for finding the best heading multiplier for given scenario was used. The optimal heading multiplier is determined by series of tests using different values of the heading multiplier, with average collected reward being recorded. The best performing heading multiplier is then used in other tests. This was performed for MBZIRC, Tsiligirides and Set66 scenarios. The results are summed up in the Figure 23. The values of the best heading multiplier h_m found for each scenario are given in Table 4.



(a) MBZIRC scenario



(b) Tsiligirides scenario



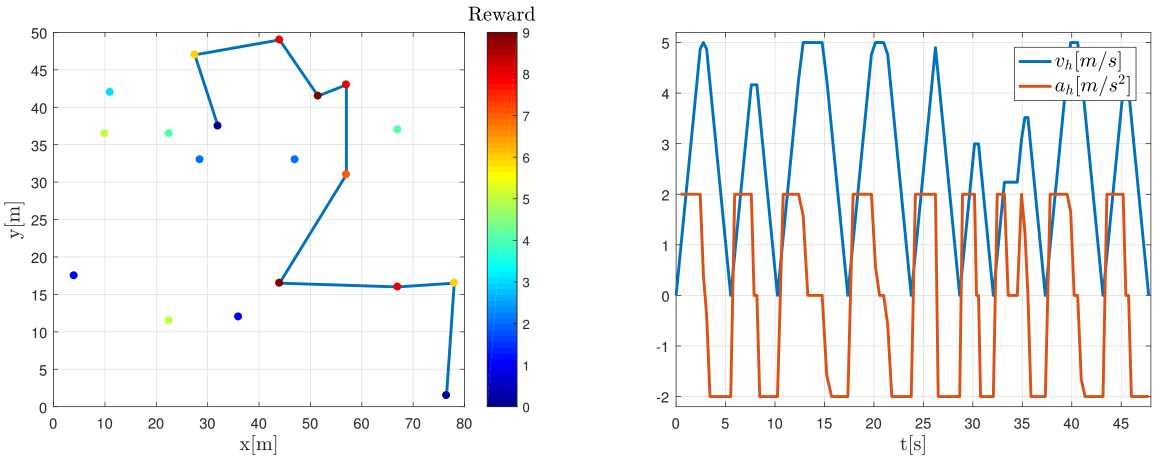
(c) Set66 scenario

Figure 23: Average gathered budget R_a based on used heading vector multiplier h_m for given scenarios

Table 4: The best heading multiplier h_m found for given scenarios

MBZIRC	Tsiligirides	Set66
3	1	2

The last set of experiments in this section compares the EOP and DOP solutions with the HOP solution. In order to simulate the natural disadvantage of Euclidean paths for UAV trajectories, a slightly modified version of HOP implementation was used. The continuous optimization was disabled and the only samples available are samples with zero velocity. This generates Euclidean paths and the UAV is forced to stop at each target location since the turning angles are very sharp. The velocity profile generation and the combinatorial optimization stays the same. An example of such Euclidean solution along with the velocity profile is shown in Figure 24.



(a) MBZIRC scenario EOP solution (b) Velocity and acceleration profile

Figure 24: An example solution of MBZIRC scenario using EOP approach with time budget $T_{max} = 50s$, maximal horizontal acceleration $a_{h_{max}} = 2m/s^2$, maximal horizontal velocity $v_{h_{max}} = 5m/s$ along with calculated velocity and acceleration profile

The implementation of the DOP solver provided by the authors of [31] was used for comparison with the HOP approach. However, the DOP solver uses the length of the trajectory instead of time of flight as the budget constraint. The DOP approach assumes that the UAV moves with constant velocity which also determines the turning radius of the Dubins curves. In order to fairly compare the DOP solution with HOP, the average velocity with which the UAV moves along the trajectory generated in HOP solution is determined from the velocity profile. This average velocity of flight is then used in the DOP as the constant velocity. From the known constant velocity v_{DOP} and known maximal time budget T_{max} in HOP solution, it is possible to determined the length budget L_{max} for DOP which is equivalent to the time budget in HOP. The Equation (32) is used to determine the DOP length budget.

$$L_{max} = v_{DOP} \cdot T_{max} \tag{32}$$

An example of the DOP solution is shown in Figure 25. The velocity profile is not included as it is considered constant.

The test for comparison of all three solutions of the EOP, DOP and HOP determines the average collected reward based on value of time budget or equivalent length budget. The average reward is again calculated using results from ten runs for each budget value. The MBZIRC scenario was used for all three solutions. The results are shown in Figure 26. It is clear that the HOP solution is superior. This shows that by taking advantage of Hermite curves and the introduction of the velocity profile it is really possible to fly the generated trajectories faster even if the actual distance might be longer.

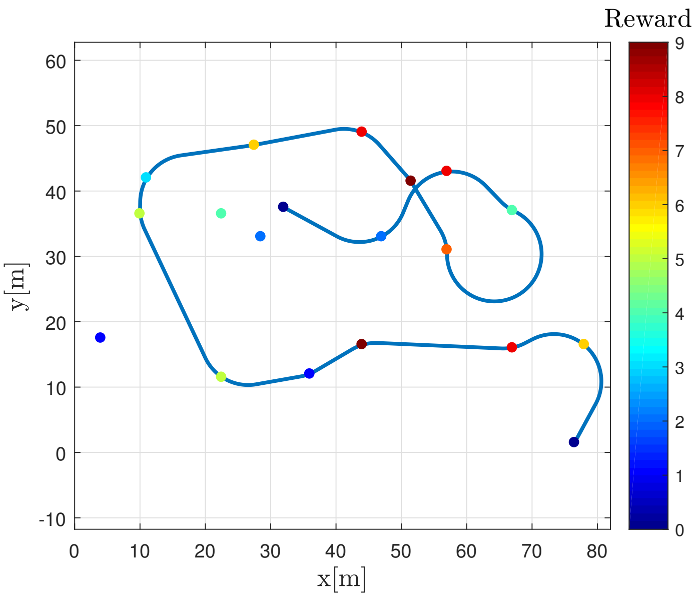


Figure 25: An example solution of MBZIRC scenario using DOP approach with length budget $L_{max} = 247.31\ m$ and constant horizontal velocity $v_h = 3.81\ m/s$ which corresponds to minimal turning radius of $7.23m$. This corresponds to a time budget of 65 seconds

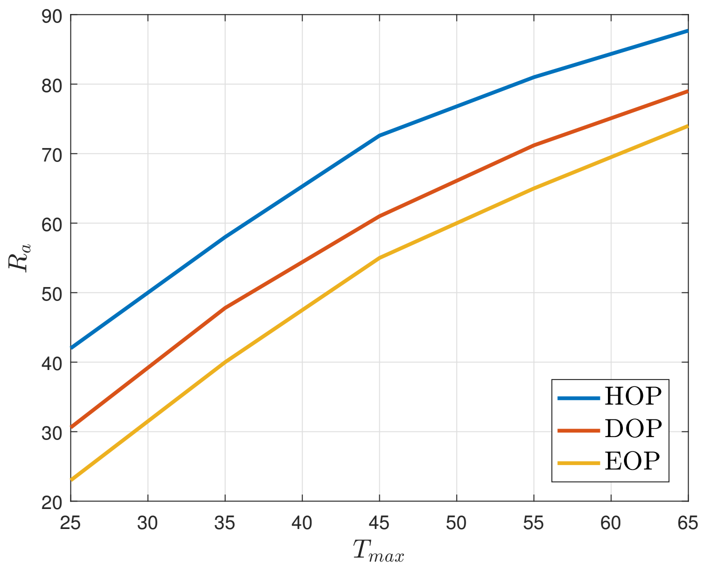


Figure 26: Comparison of average rewards R_a for given maximal budget T_{max} for HOP, DOP and EOP solutions

6.2 Experimental verification using real UAV

The experiments with the UAV prototype aim to show the feasibility of the generated trajectories and the correctness of the time of flight estimate. Three HOP solutions using different sets of parameters were selected in order to test the full range of capabilities of this solution. The MBZIRC scenario was used in all cases. The target locations were manually marked in open field by objects with numbers corresponding to rewards. The locations of these targets were determined using differential GPS. The duration of the experiments was recorded by on-board Mobius camera with wide field of view used mainly to record the target markers on the ground. Additionally the flight of the UAV was recorded externally by commercially sold UAV with 4k camera from the height of roughly hundred meters.

Table 5: Experiments setup

Code name	2D_HOP	3D_EOP	3D_HOP
Scenario	MBZIRC	MBZIRC (modified)	MBZIRC (modified)
Dimension	2D	3D	3D
Heading mult.	5	0 (Euclid)	5
Budget	45	55	55
Reward	70	51	62

Table 6: UAV motion constraints

$v_{h_{max}}$	$a_{h_{max}}$	$v_{v_{max}}$	$a_{v_{max}}$
$5m/s$	$2m/s^2$	$1m/s$	$1m/s^2$

The set of used parameters for each of three experiments is summarized in Table 5. In the 3D cases 3D_EOP and 3D_EOP, the topology of the target locations remained the same in the plane with each target being randomly assigned the theoretical height ranging from 4-20 meters. An example of vertical profile in time for 3D_HOP experiment is shown in Figure 27. Also for 3D scenarios the starting location was moved. The maximal horizontal and vertical velocities of used UAV are shown in the Table 6.

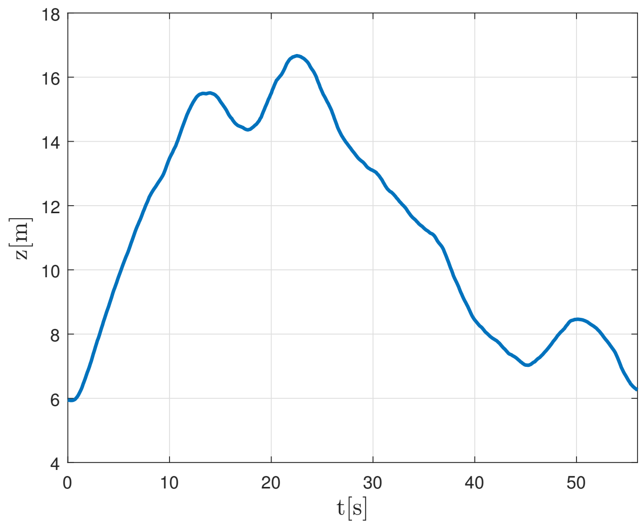
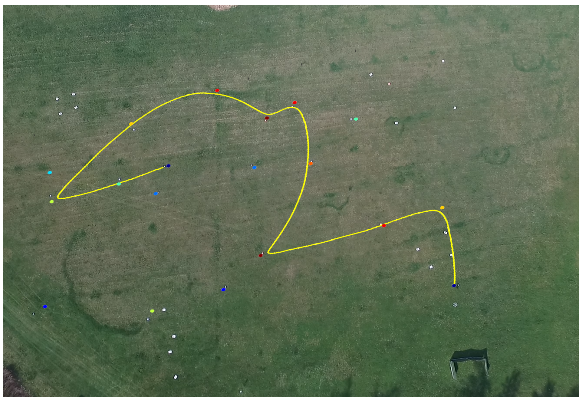


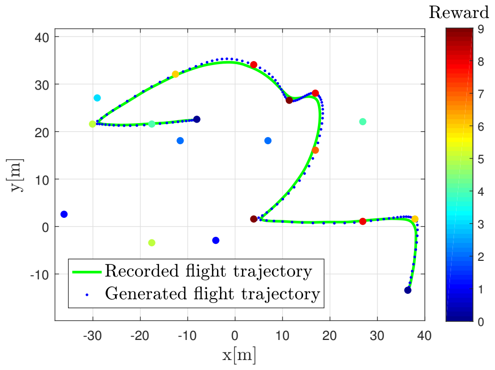
Figure 27: Vertical profile of the 3D_HOP experiment in time

Recorder flight trajectories plotted over the image of the testing range are shown in Figure 28 along with the comparison of recorded trajectories with the theoretical splines which were supplied to the UAV model predictive controller. The actual locations of the targets in the image are distorted by the camera of the UAV so the plotted trajectories are not exactly on spot. It is clear that the UAV follows the trajectory with reasonable accuracy. Minor deviations can be noticed, but these do not hold much importance for the actual goal of surveillance, since all of the target locations are visited. This is also supported by a video [1] from the on-board camera which shows all of the target locations in the filed of view. The UAV might not be exactly at the target location but the deviations do not prevent successful surveillance since the field of view of used sensor, which is usually some form of image acquisition device, is rarely very restricted.

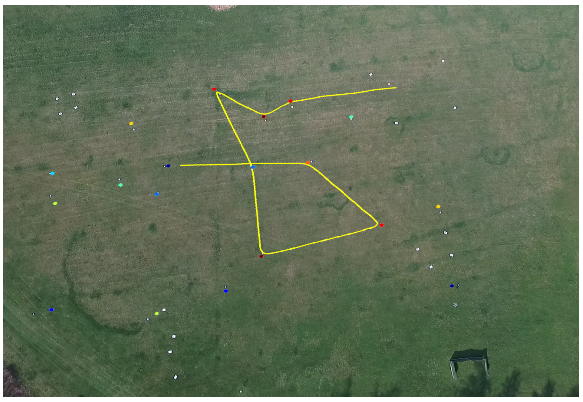
[1]A link to a video footage from all three experiments: https://www.youtube.com/watch?v=gagYFLpGVC4



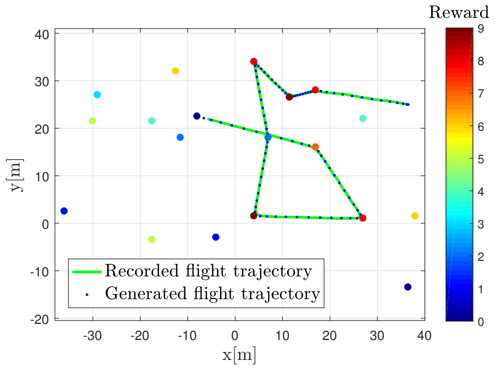
(a) 2D_HOP aerial view of measured trajectory



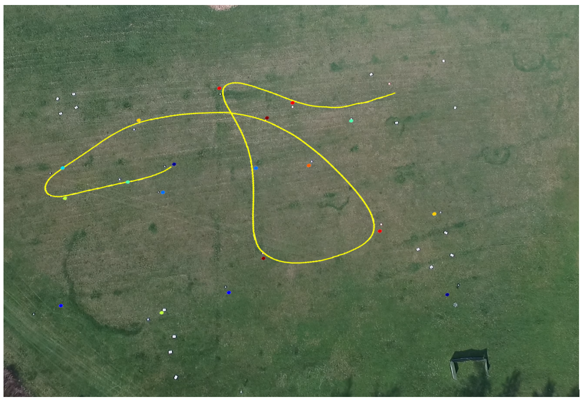
(b) 2D_HOP measured and theoretical trajectory comparison



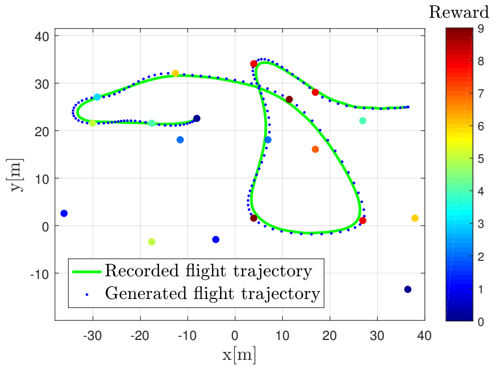
(c) 3D_EOP aerial view of measured trajectory



(d) 3D_EOP measured and theoretical trajectory comparison



(e) 3D_HOP aerial view of measured trajectory



(f) 3D_HOP measured and theoretical trajectory comparison

Figure 28: Recorder flight trajectories by the UAV video platform along with comparison of measured and theoretical generated trajectories

An interesting phenomenon is shown in the detail of the EOP scenario in Figure 29. It shows that the MPC is not able to follow the Euclidean path accurately and smooths the flight path in very sharp turns. This supports the motivation behind not using the Euclidean paths as motion primitives in UAV motivated OP solutions.

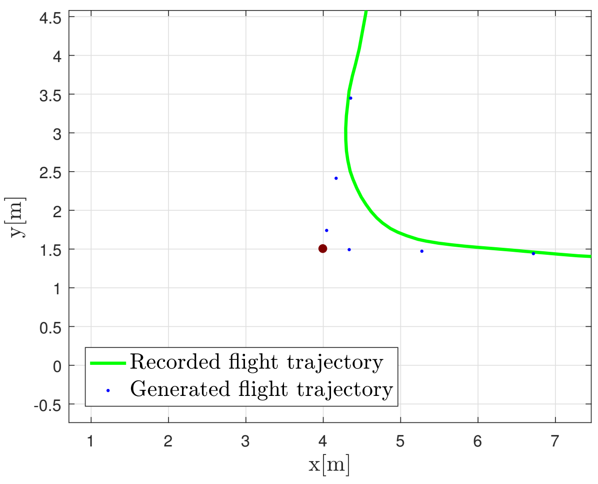
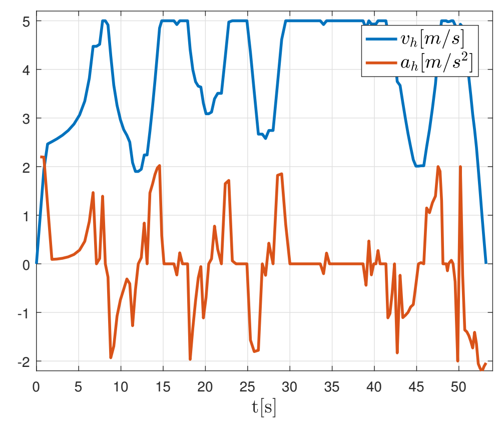
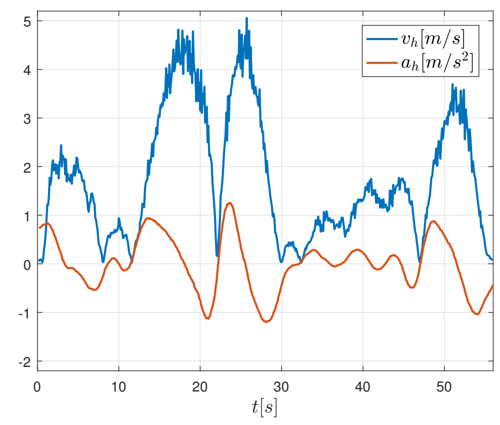


Figure 29: A detail of generated Euclidean trajectory and the actual recorded UAV trajectory

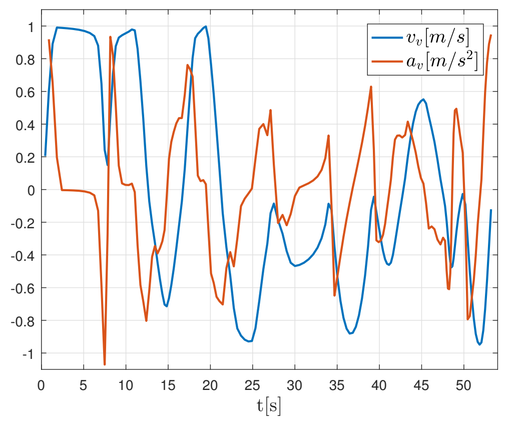
It was shown that the trajectory following is reasonably accurate from geometrical point of view, but the resulting trajectories are defined with respect to the maximal theoretical velocity profile, so it is important to test the accuracy of this estimate. The horizontal and vertical theoretical velocity and acceleration profile along with the real measured profiles from the 3D_HOP experiment are shown in Figure 30. It is noticeable that the profiles differ quite significantly. The measured horizontal acceleration is much smoother and does not change as dramatically. This difference was determined to be caused by the UAV motion regulator, since it is not only limited in velocity and acceleration but also in jerk. Both profiles along with the jerk profile are shown in Figure 31. It is clear that the jerk in the MPC controller is limited, which results in different velocity and acceleration profiles. It means that the velocity profile is theoretically correct but in order to comply with the chosen UAV it would have to be modified to take the jerk limitation in account or the UAV controller itself needs to be changed to be more benevolent towards the jerk limitations. This contradiction also resulted in differences in predicted and measured time of flight for each scenario, which was, however, a difference of approximately 1.8%, so the estimation is still reasonably accurate. Either way, this problem is a definite candidate for improvement in further development, but since the change affects only the velocity profile calculation it is fairly isolated problem, which can be enhanced without dramatically affecting the over all approach and performance of the HOP solver.



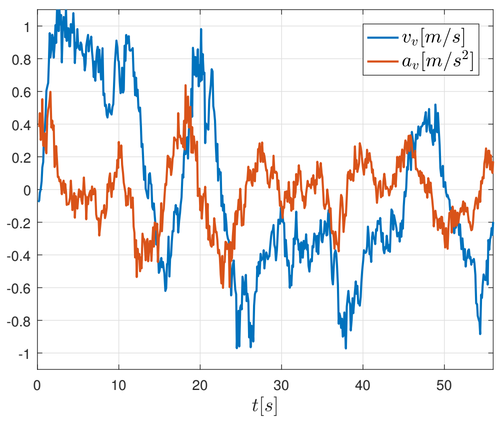
(a) Theoretical horizontal velocity and acceleration profile



(b) Measured horizontal velocity and acceleration profile

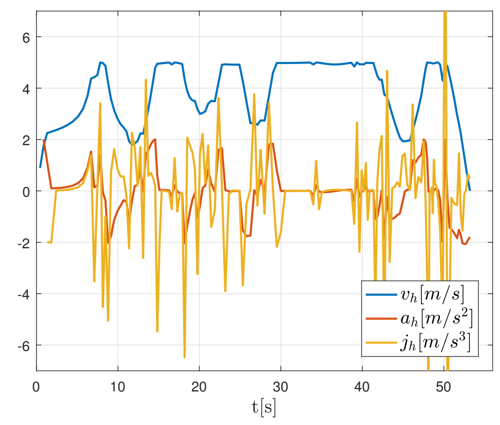


(c) Theoretical vertical velocity and acceleration profile

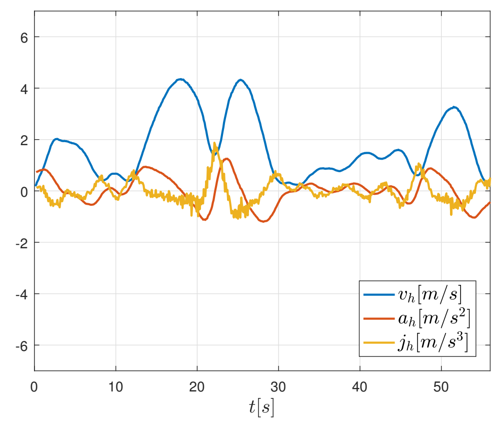


(d) Measured vertical velocity and acceleration profile)

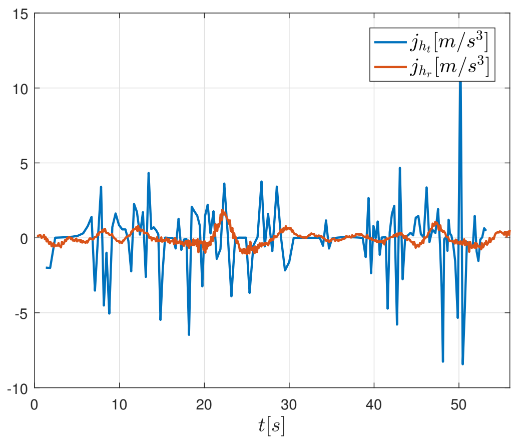
Figure 30: Theoretical and measured velocity and acceleration profiles comparison for the 3D_HOP scenario solution. v_h - horizontal velocity, a_h - horizontal acceleration, v_v - vertical velocity, a_v - vertical acceleration



(a) Theoretical horizontal velocity, acceleration and jerk profile



(b) Measured horizontal velocity, acceleration and jerk profile



(c) Comparison of theoretical and measured jerk profile

Figure 31: Theoretical and measured velocity, acceleration and jerk profiles comparison for the 3D_HOP scenario solution.v_h - horizontal velocity, a_h - horizontal acceleration, j_h - horizontal jerk, j_{h_r} - measured horizontal jerk,j_{h_t} - theoretical horizontal jerk

7 Conclusion

A novel approach to UAV trajectory planning based on Orienteering Problem was introduced and implemented in this thesis. The proposed solver uses Hermite cubic splines as a motion primitives to find fast and feasible solutions of the OP. This variant of the OP was called Hermite Orienteering Problem.

First part of the solution consists of creation of method to easily parametrize and generate Hermite splines. Along the generated curve a velocity and acceleration profile is calculated respecting the motion limitations of the UAV, which determines the estimated time of flight needed to traverse the curve by the UAV. Trajectories can be generated in plane or in full 3D space. This is a big advantage over currently existing solutions as majority is solved only in plane.

Hermite curves are then used to find solutions to the OP. The solution of HOP is based on the Random Variable Neighborhood Search algorithm which is a suitable heuristic for this kind of problem and was already successfully used in similar tasks such as the DOP. Unfortunately, the RVNS algorithm performs only the combinatorial part of the optimization task, which consists of combinatorial and continuous optimization problem. The continuous optimization part of the problem is introduced by the use of Hermite curves, that are defined by target locations and heading vectors, so in order to find solutions of good quality, the continuous heading angle and heading vector norm need to be optimized as well. This is done by introduction of continuous optimization operators into the RVNS algorithm, which enable to use small initial sampling resolution of heading angle and velocity determining the heading vector norm and still find solutions of good quality, as new samples are added continually by combination of random shake and systematic exploration operations.

Two types of tests were performed to verify the performance and correctness of implemented solution. The computational tests and also the experiments using real UAV. The computational tests were performed by solving large number of testing scenarios on a distributed grid of high performance processors which enabled to get sufficient statistical sample. These tests showed that the continuous optimization works successfully as the quality of found solutions did not change with higher sampling rates, which is much more performance demanding. The current solution was compared to other approaches to OP which were the Euclidean OP and the Dubins OP. The HOP solution proposed in this thesis showed to be superior to both, as the average collected reward was higher in all test cases which means the UAV is able to visit more targets compared to the EOP and DOP solutions using the same travel budget.

Another set of tests were performed using the real UAV. It was shown that the UAV is able to successfully track the generated Hermite trajectories and can use its full kinematic potential, since it can reach high velocities and move in full three dimensional space. A maximal jerk limitation in the real UAV motion controller was found to be a source of error in velocity profile and time of flight estimation, as the velocity profile introduced in this thesis does not address the jerk limitation in the solution. This can be improved in future work.

Overall, the solution to the UAV motivated Orienteering Problem proposed in this thesis has been shown to be applicable in real UAV applications while demonstrating significant advantages over existing solutions.

References

[1] Metacentrum by cesnet. https://metavo.metacentrum.cz. Cited on: 20.5.2018.

[2] Multirobotic system group. https://mrs.felk.cz. Cited on: 20.5.2018.

[3] Robotic operating system. http://www.ros.org/. Cited on: 20.5.2018.

[4] Otto Alena, Agatz Niels, Campbell James, Golden Bruce, and Pesch Erwin. Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. *Networks*, 0(0), 2018.

[5] Jim Armstrong. Hermite curves. http://algorithmist.net/docs/hermite.pdf, 2015. Cited on: 20.5.2018.

[6] T. Baca, G. Loianno, and M. Saska. Embedded model predictive control of unmanned micro aerial vehicles. *2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 992–997, August 2016.

[7] Pornporm Boonporm. Online geometric path planning algorithm of autonomous mobile robot in partially-known environment. *The Romanian Review Precision Mechanics, Optics and Mechatronics*, pages 185–188, January 2011.

[8] Xuan-Nam Bui, J. D. Boissonnat, P. Soueres, and J. P. Laumond. Shortest path synthesis for dubins non-holonomic robot. pages 2–7 vol.1, May 1994.

[9] C. V. Caldwell, D. D. Dunlap, and E. G. Collins. Motion planning for an autonomous underwater vehicle via sampling based model predictive control. pages 1–6, Sept 2010.

[10] I. M. Chao, B. L. Golden, and E. A. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475–489, 1996.

[11] C. Deng, S.Wang, Z. Tan Z. Huang, and J. Liu. Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications. *J. Commun*, 9(9):687–692, 2014.

[12] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[13] Wikipedia entry. Travelling salesman problem. https://en.wikipedia.org/wiki/Travelling_salesman_problem. Cited on: 20.5.2018.

[14] J. Faigl, R. Penicka, and G. Best. Self-organizing map-based solution for the orienteering problem with neighborhoods. *IEEE International Conference on Systems, Man, and Cybernetics*, pages 1315–1321, October 2016.

[15] Jan Faigl and Petr Váňa. Surveillance planning with bézier curves. *IEEE Robotics and Automation Letters*, 3(2):750–757, April 2018.

[16] J. A. Goncalves and R. Henriques. Uav photogrammetry for topographic monitoring of coastal areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104:101–111, June 2015.

[17] A. Gunawan, H. C. Lau, and P. Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.

[18] P. Hansen and N. Mladenovi. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, May 2001.

[19] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[20] Hydro-Québec International. Uav transmission line surveillance. http://www.hydroquebec.com/international/en/technology/robotics.html. Cited on: 20.5.2018.

[21] Marko Lepetič, Gregor Klančar, Igor Škrjanc, Drago Matko, and Boštjan Potočnik. Path planning and path tracking for nonholonomic robots. *Mobile Robots: New Research*, pages 345–368, 2005.

[22] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[23] Sven Lorenz and Florian M. Adolf. A decoupled approach for trajectory generation for an unmanned rotorcraft. *Advances in Aerospace Guidance, Navigation and Control*, pages 3–14, 2011.

[24] G. Morgenthal and N. Hallermann. Quality assessment of unmanned aerial vehicle (uav) based visual inspection of structures. *Advances in Structural Engineering*, 17(3):289–302, 2014.

[25] P. Oberlin, S. Rathinam, and S. Darbha. Today's traveling salesman problem. *Robotics Automation Magazine*, 17(4):70–77, 2010.

[26] Doherty P. and Rudol P. A uav search and rescue scenario with human body detection and geolocalization. *AI 2007: Advances in Artificial Intelligence*, 4830, 2007.

[27] G. Pajares. Overview and current status of remote sensing applications based on unmanned aerial vehicles (uavs). *Photogrammetric Engineering & Remote Sensing*, 81(4):281–329, 2015.

[28] M. T. Perks, A. J. Russell, , and A. R. Large. Advances in flash flood monitoring using unmanned aerial vehicles (uavs). *Hydrology and Earth System Sciences*, 20(10):4005–4015, 2016.

[29] Matěj Petrlík. Onboard localization of an unmanned aerial vehicle in an unknown environment. Master's thesis, Czech technical university in Prague, Czech Republic, 2018.

[30] Eigen project. Eigen opensource library. http://eigen.tuxfamily.org. Cited on: 20.5.2018.

[31] R. Pěnička, J. Faigl, P. Váňa, and M. Saska. Dubins orienteering problem. *IEEE Robotics and Automation Letters*, 2(2):1210–1217, April 2017.

[32] R. Pěnička, J. Faigl, P. Váňa, and M. Saska. Dubins orienteering problem with neighborhoods. *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1555–1562, June 2017.

[33] P. B. Quater, F. Grimaccia, S. Leva, M. Mussetta, and M. Aghaei. Light unmanned aerial vehicles (uavs) for cooperative inspection of pv plants. *IEEE Journal of Photovoltaics*, 4(4):1107–1113, 2014.

[34] M. Quigley, M.A. Goodrich, S. Griffiths, A. Eldredge, and R.W. Beard. Target acquisition, localization, and surveillance using a fixed-wing mini-uav and gimbaled camera. *Robotics and Automation*, April 2005.

[35] R. Ramesh and K. M. Brown. An efficient four-phase heuristic for the generalized orienteering problem. *Computers & Operations Research*, 18(2):151–165, February 1991.

[36] C. Rego, D. Gamboa, F. Glover, and C. Osterman. Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research*, 211(3):427–411, 2011.

[37] S. Russell and P. Norvig. Artificial intelligence: A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25(27):79–80, 1995.

[38] Heikki Saari, Ismo Pellikka, Liisa Pesonen, Sakari Tuominen, Jan Heikkilä, Christer Holmlund, Jussi Mäkynen, Kai Ojala, and Tapani Antila. Unmanned aerial vehicle (uav) operated spectral camera system for forest and agriculture applications. *Remote Sensing for Agriculture, Ecosystems, and Hydrology*, 13, October 2011.

[39] H. M. Salkin and C. A. De Kluyver. The knapsack problem: A survey. *Naval Research Logistics Quarterly*, 22(1):127–144, 1975.

[40] AIRX3 Sarl. Uav solar panel surveillance. http://www.airx3.com. Cited on: 20.5.2018.

[41] M Saska, J Langr, and L Preucil. Plume tracking by a self-stabilized group of micro aerial vehicles. *Modelling and Simulation for Autonomous Systems*, pages 44–55, 2014.

[42] Simon Schopferer and Florian M. Adolf. Rapid trajectory time reduction for unmanned rotorcraft navigating in unknown terrain. *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 305–316, May 2014.

[43] Z. Sevkli and F. E. Sevilgen. Variable neighborhood search for the orienteering problem. *ISCIS*, pages 134–143, 2006.

[44] Ch. Siyuan, F. Laefer, and M. Eleni. State of technology review of civilian uavs. *Recent Patents on Engineering*, 10(3):160–174, December 2016.

[45] Christoph Sprunk. Planning motion trajectories for mobile robots using splines. Student project, University of Freiburg, 2008.

[46] T. Tsiligirides. Heuristic methods applied to orienteering. *The Journal of the Operational Research Society*, 35(9):797–809, 1984.

[47] Michiel van de Panne. Online lecture on parametric curves. https://www.cs.helsinki.fi/group/goa/mallinnus/curves/curves.html, 1996. Cited on: 20.5.2018.

[48] Petr Wagner, Jiri Kotzian, Jan Kordas, and Viktor Michna. Path planning and tracking for robots based on cubic hermite splines in real-time. *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, pages 1–8, September 2010.

[49] Edward J. Wegman and Ian W. Wright. Splines in statistics. *Journal of the American Statistical Association*, 78(382):351–365, 1983.

[50] Ji wung Choi, Renwick Curry, and Gabriel Elkaim. Path planning based on bézier curve for autonomous ground vehicles. *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*, pages 158–166, October 2008.

[51] G. J. Yang, Raimarius Delgado, and B. W. Choi. A practical joint-space trajectory generation method based on convolution in real-time control. *International Journal of Advanced Robotic Systems*, 13(2):56, March 2016.

[52] C. Yuan, Y. Zhang, and Z. Liu. A survey on technologies for automatic forest re monitoring, detection, and ghting using unmanned aerial vehicles and remote sensing techniques. *Canadian journal of forest research*, 45(7):783–792, 2015.

Appendix A CD Content

In Table 7 are listed names of all root directories on CD.

Directory name	**Description**
thesis	the thesis in pdf format
thesis_sources	latex source codes
vns_hop	HOP solver c++ source code

Table 7: CD Content

Appendix B   List of abbreviations

In Table 8 are listed abbreviations used in this thesis.

Abbreviation	**Meaning**
OP	Orienteering Problem
HOP	Hermite Orienteering Problem
DOP	Dubins Orienteering Problem
EOP	Euclidean Orienteering Problem
UAV	Unmanned Aerial Vehicle
RVNS	Random Variable Neighborhood Search
VNS	Variable Neighborhood Search
TSP	Traveling Salesman Problem
2D	Two-dimensional
3D	Three-dimensional

Table 8: Lists of abbreviations