Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Master-Thesis

Path Planning for Dynamic Maneuvers with Micro Aerial Vehicles

Autumn Term 2014

Supervised by:
Markus Achtelik
Michael Burri

Author:
Fabian Neuschäfer

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Path Planning for Dynamic Maneuvers with Micro Aerial Vehicles

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

First name	Last name

Supervising lecturer

First name	Last name

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`http://www.ethz.ch/students/exams/plagiarism_s_en.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date	Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Abstract

The goal of this Master-Thesis is to develop a numerical robust trajectory-planning algorithm for aggressive multi-copter flights in dense environments. The trajectory generated by this algorithm is represented by polynomials which are jointly optimized. The cost function of the optimization consists of the total trajectory-time as well as the total quadratic snap (second derivation of the acceleration). Including the snap into the cost function guaranties a trajectory without abrupt or expensive control inputs.

Furthermore the process of exploring the state space using the Rapidly-Exploring Random Tree (RRT) algorithm is embedded into the numerical robust algorithm. The sampling points oft the RRT (or RRT*) algorithm are then used as the nodes in the polynomial optimization.

Symbols

Symbols

ϕ, θ, ψ	roll, pitch and yaw angle

Indices

x	x axis
y	y axis

Terms and Definitions

jerk	Derivation of acceleration
snap	Derivation of jerk
vertex	Fixed sampling point of a polynomial trajectory

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
UAV	Unmanned Aerial Vehicle
RRT	Rapidly-Exploring Random Tree
QP	Quadratic Programming

Chapter 1

Introduction

1.1 State of the Art

A lot of research has been done in the field of Unmanned Aerial Vehicles (UAV) in the last years leading to a strong improvement in planning [1] as well as in control [[2], [3]]. Another research field is machine learning [4] which is suitable to enhance the performance of aerobatic maneuvers but seams to have a downside regarding motion planning and trajectory generation in dense environments.
Speaking of trajectory planning, there are two different strategies which are pursued. On the one hand, the geometric and the temporal planning are decoupled [5] on the other hand, geometric and temporal information are coupled and the trajectory is the result of a minimization problem. For the couplet problem one can make use of the differential flatness of a quadrocopter to derive constraint on the trajectory. Then formulate a cost-function which could be the trajectory-time [3] or the total snap [6] (second derivation of acceleration).

Another aspect of planning is exploring the state space in the first place. A strong tool to do so are incremental search techniques as for instance the A* [7] or the RRT* algorithm [8]. The sampling points of the solution of the incremental search can then be used as the nodes for the polynomial optimization.

1.2 Quadratic Programming

1.2.1 Constrained Quadratic Programming

Quadratic Programming (QP) is a special case of an optimization problem in which a quadratic function $f(x)$ is optimized with respect to its optimizations variables (which are represented with the vector x in Equation 1.1)

$$f(x) = \frac{1}{2} \cdot x^T Q x + c^T x \tag{1.1}$$

The optimization can be performed under linear constraints on the optimizations variables. Whereas a distinction between equality($E\mathbf{x} = \mathbf{d}$) and inequality constraints ($A\mathbf{x} \leq \mathbf{b}$) has to be made. In case there are only equality constrains, the solution to the QP is given by the linear system in Equation 1.2 :

$$\begin{bmatrix} Q & E^T \\ E & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\mathbf{c} \\ \mathbf{d} \end{bmatrix} \tag{1.2}$$

where λ is a set of Lagrange multipliers.

The constrained QP gets ill-conditioned for a large number of optimization variables which lead to large matrices. The performance of the constraint QP deteriorate even more if the matrices are sparse. This particular case often appears in polynomial optimization for high order polynomials where some polynomial coefficients are close to zero.

To reduce the number of optimization variables, and therefore the size of the matrices, the constrained QP can be converted into a numerical robust unconstrained QP.

1.2.2 Unconstrained Quadratic Programming

For the unconstrained QP the equality constraints $E\mathbf{x} = \mathbf{d}$ resp. $\mathbf{x} = E^{-1}\mathbf{d}$ are embedded into the quadratic cost-function from Equation 1.1 resulting in Equation 1.3:

$$f(d) = \frac{1}{2} \cdot d^T E^{-T} Q E^{-1} d + c^T E^{-1} d \tag{1.3}$$

Referring to polynomial trajectory optimization, the vector x containing the polynomial coefficients is now replaced by the vector d containing the endpoint derivatives and the mapping matrix E. In other words, the polynomial coefficients are no longer the optimization variables but the free endpoint derivatives are optimized. Furthermore the polynomial trajectory optimization does not have a linear term $c^T\mathbf{x}$. Hence Equation 1.3 can be simplified to:

$$f(d) = \frac{1}{2} \cdot d^T E^{-T} Q E^{-1} d \tag{1.4}$$

Since we are interested in the optimal endpoint derivatives d^* and not in the cost itself, the constant multiplier $1/2$ can be dropped:

$$f(d) = d^T E^{-T} Q E^{-1} d \tag{1.5}$$

Equation 1.5 can be compared to to multidimensional cost function in Equation 2.5 in the Section 2.2.3 "Polynomial Optimization as a Unconstrained QP".

Chapter 2

Polynomial Trajectory Optimization

2.1 Polynomial Trajectory

Regarding the differentiability of polynomials, they are a profound choice to represent a trajectory. Especially for the use in a differentially flat representation of the UAV dynamics. (Flatness in the proper sense of system theory means that all the states and inputs can be expressed in terms of the flat output and a finite number of its derivative). Furthermore, the differentiability of polynomials enables the possibility to check the derivatives of the trajectory for bounding violations to avoid input saturation. This saturation-check can be perform during trajectory optimization and therefore guarantees the feasibility of the resulting trajectory.

2.2 Optimization

The goal is to optimize a trajectory which passes through way-points (also called vertices or nodes) which are defined in advance. This way-points can be chosen manually or by a path-finding algorithm such as RRT* which will be discussed in Chapter 3. Furthermore, not only the way-points (therefore the position) can be fixed in advance but also its derivatives (such as speed, acceleration etc.). The position and its derivatives are then utilized as the equality constrains for a QP (explained in Section 1.2).

2.2.1 Cost Function

Optimization for the purpose of trajectory planning means to minimize a cost function. The cost function in this case is a combination of temporal and geometric cost. The geometric cost penalizes the (square) of the derivatives of the trajectory. In this Master Thesis the geometric cost is represented by the squared snap which guarantees a trajectory without abrupt control inputs.
The temporal cost is simply the total trajectory-time multiplied by a user chosen factor k_T which determines the aggressiveness of the resulting trajectory. The usage of k_T can be seen in Equation 2.10 which represents the combined geometric and temporal cost.

To express the geometric cost in a compact way on can make use of the Hessian matrix Q. The Hessian matrix is defined as a squared matrix of second-order partial derivatives which follows from differentiation a function with respect to each of its coefficients (in this instance the polynomial coefficients). The geometric cost function $J(T)$ for a fixed time for one segment can now be written as

$$J(T) = p^T \cdot Q(T) \cdot p \tag{2.1}$$

where $Q(T)$ is the Hessian matrix for a fixed segment-time T and p is the vector containing the coefficients of the polynomial.

If the trajectory consists of more than one segment the Hessian matrix has to extended to a block-diagonal matrix and the geometric cost function for multiple segments with fixed bud individual segment-times can be written as

$$J = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}^T \cdot \begin{bmatrix} Q_1(T_1) & & \\ & \ddots & \\ & & Q_n(T_n) \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix} \tag{2.2}$$

2.2.2 Polynomial Optimization as a Constrained QP

In a first, intuitive approach the equality constraints on the endpoint derivatives (mentioned in Section 2.2) are utilized in a constrained QP. Therefore a mapping matrix E between endpoint derivatives and polynomial coefficients is needed. The resulting formula for the i^{th} segment can be written as

$$E_i \cdot p_i = d_i \tag{2.3}$$

where p is the vector containing the polynomial coefficients and d is the vector containing the endpoint derivatives. Regarding the total number of segments of the trajectory, Formula 2.3 can be written in matrix form:

$$\begin{bmatrix} E_1 & & \\ & \ddots & \\ & & E_n \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix} \tag{2.4}$$

The constrained QP is suitable for a small amount of segments but gets ill-conditioned for a large amount of segments and therefore large matrices. Especially if there are matrices which are close to singularity and have coefficients which are close to zero, the constrained QP can get numerical unstable.

2.2.3 Polynomial Optimization as a Unconstrained QP

To avoid the numerical instability of a constrained QP the optimization problem is converted into a unconstrained QP. Therefore the polynomial coefficients p_i from Formula 2.2 have to be substituted by the endpoint derivatives d_i which are now the new optimizations variables. The cost function of the unconstrained QP can now be written as

$$J = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix}^T \cdot \begin{bmatrix} E_1 & & \\ & \ddots & \\ & & E_n \end{bmatrix}^{-T} \cdot \begin{bmatrix} Q_1 & & \\ & \ddots & \\ & & Q_n \end{bmatrix} \cdot \begin{bmatrix} E_1 & & \\ & \ddots & \\ & & E_n \end{bmatrix}^{-1} \cdot \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix} \tag{2.5}$$

where Q_i is the Hessian matrix according to the i^{th} segment-time.

As mentioned above, the endpoint derivatives are the new optimization variables. Do to the equality constrains some of the endpoint derivatives are already specified consequently reducing the number of optimizations variables. Expediently, the endpoint derivatives are divided in fixed derivatives d_f and unspecified derivatives d_p and then reordered using the matrix C which consists of zeros and ones. After reordering the endpoint derivatives Formula 2.5 can be rewritten as

$$J = \begin{bmatrix} d_f \\ d_p \end{bmatrix}^T \underbrace{C^T E^{-T} Q E^{-1} C}_{\text{R}} \begin{bmatrix} d_f \\ d_p \end{bmatrix} \tag{2.6}$$

where the product of the reordering matrix C, the mapping matrix E and the Hessian matrix Q han be expressed as a single Matrix R. The matrix R for his part can be defided into four submatrices according to the fixed and unspecified endpoint derivatives which modifies Formula 2.6 as follows:

$$J = \begin{bmatrix} d_f \\ d_p \end{bmatrix}^T \begin{bmatrix} R_{ff} & R_{fp} \\ R_{pf} & R_{pp} \end{bmatrix} \begin{bmatrix} d_f \\ d_p \end{bmatrix} \tag{2.7}$$

Partially differenting Formula 2.7 with respect to the unspecified derivatives d_p and equate it to zero yields the optimized/minimized unspecified derivatives d_p^*

$$d_p^* = -R_{pp}^{-1} \cdot R_{fp}^T \cdot d_f \tag{2.8}$$

as a function of the fixed derivatives d_f and two of the submatrixes (R_{pp}, R_{fp}) of R.

2.2.4 Initial Solution

Equation 2.8 can now be used to compute the initial solution. As can be seen in Equation 2.5 the Hessian matrix for the i^{th} segment Q_i depends on the segment-time i. Thus, all the segment-times has to be defined. For the initial solution the segment-times are estimated based on the 2-norm distance d_{norm} and on the user specified maximal speed (v_{max}) and maximal acceleration (a_{max}).
Basically the segment-time is determined by the term $d_{norm}/v_{max} \cdot 2$ which is twice the time the UAV would need for a segment by flying at maximal speed the whole distance. Although this is a good estimation for long segments, for shorter ones the time needed to accelerate gets significant. Therefore a multiplier, which is zero for long segments and unequal to zero for short ones, is added. The segment-time t_i for the i^{th} segment can be computed according to

$$t_i = \frac{d_{norm_i}}{v_{max}} \cdot 2 \cdot \left(1 + 6.5 \cdot \frac{v_{max}}{a_{max}} \cdot \frac{1}{e^{\frac{d_{norm_i}}{v_{max}} \cdot 2}}\right) \tag{2.9}$$

where d_{norm_i} is the 2-norm distance of the i^{th} segment, v_{max} the user specified maximal velocity and a_{max} the user specified maximal acceleration. The fraction v_{max}/a_{max} gives an idea how much time is needed to accelerate to maximum velocity whereas 6.5 is a empirical correction factor.

The result from Equation 2.9 is depicted in Figure 2.1 whereat the x-axis represents the 2-norm distance d_{norm} and the y-axis represents the segment time t. For this plot the user specified limitation on speed and acceleration has been set to $v_{max} = 3\frac{m}{s}$ and $a_{max} = 5\frac{m}{s^2}$. The green line represent the term $d_{norm}/v_{max} \cdot 2$, the blue graph takes the time needed for acceleration into account and is therefore the exact representation of Equation 2.9.

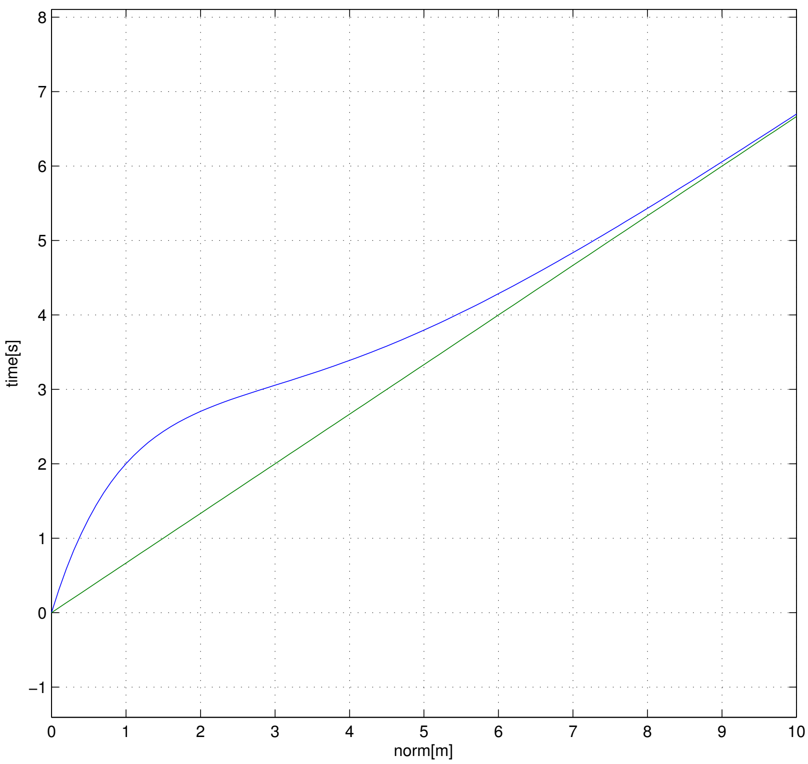

Figure 2.1: The segment-time t depends on the distance d_{norm} of a segment and on the maximal velocity v_{max}. The blue graph additionally considers the time needed for acceleration and is therefore a more elaborate approach to estimate the segment-time.

Once the segment-times are estimated the initial, snap minimized solution can be computed according to Equation 2.8. The initial solution for a 3 dimensional problem with 4 segments is depict in Figure 2.2. The first of this 3 subplots shows the position, the second the velocity and the third the acceleration. The x-axis for all the 3 subplots is the time. The 3 graphs in the first subplot represent the 3 dimension where the colors are different for each segment. In the second and third subplot, there are also 3 graphs representing a dimension but also a fourth, thicker graph which represents the 2-norm of the velocity respectively the acceleration. Furthermore the limitation ($v_{max} = 3\frac{m}{s}$ and $a_{max} = 2\frac{m}{s^2}$ for this problem) are depicted.

2.2.5 Penalty on Time

So fare, only the geometric cost (i. e. the squared snap) was discussed. Minimization of the geometric cost ensures a smooth trajectory without abrupt input signal but has no effect on the aggressiveness of a trajectory. Therefore Formula 2.7 has to be extended by the temporal cost which results in the total cost J_{total}:

$$J_{total} = \begin{bmatrix} d_f \\ d_p \end{bmatrix}^T \begin{bmatrix} R_{ff} & R_{fp} \\ R_{pf} & R_{pp} \end{bmatrix} \begin{bmatrix} d_f \\ d_p \end{bmatrix} + k_T \cdot \sum_{i=1}^{N} T_i \tag{2.10}$$

where k_T is a user specified penalty on time and T_i is the segment-time of the i^{th} segment.

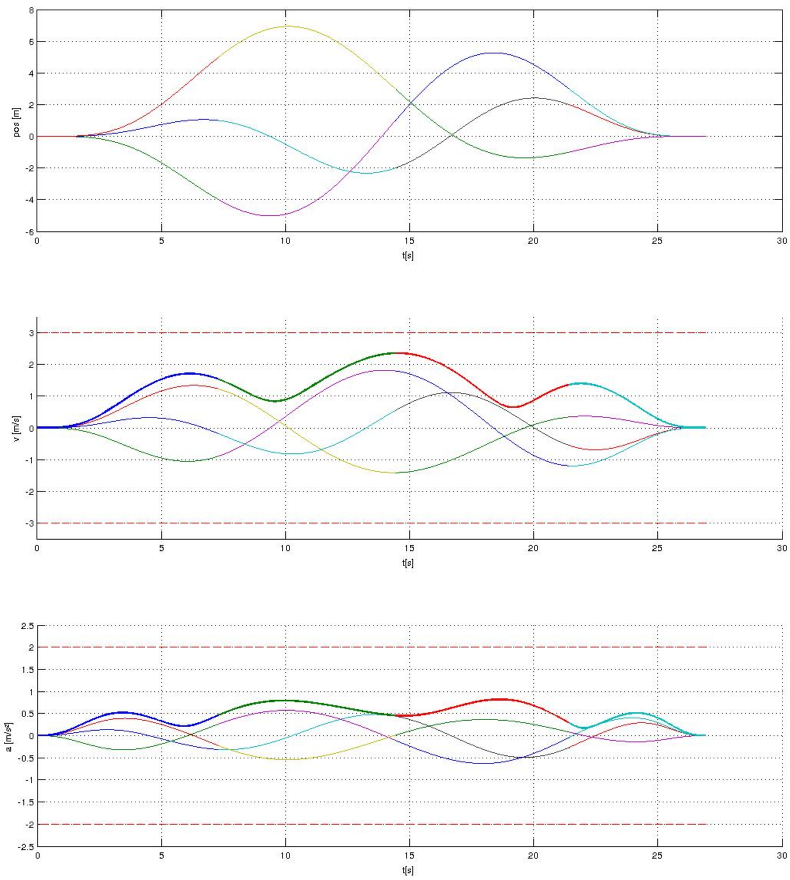

Figure 2.2: Initial solution: The first subplot shows the position, the second the velocity and the third the acceleration of the initial solution. A thicker graph represents the 2-norm of the velocity respectively the acceleration.

In contrast to Equation 2.7, Equation 2.10 cannot be solved analytically. Due to that a nonlinear solver is used. In this Master Thesis NLopt, a open-source library for nonlinear optimization, is applied. The optimizations variables of the nonlinear optimization are the estimated segment-times and the unspecified endpoint derivatives dp. The computational cost of the nonlinear optimization is highly depending on the quality of the initial solution.
The termination condition of the optimization can be specified on the optimization variables as well as on the total cost. Generally, the termination conditions are formulated relative to the current value(s). For instance, if the relative termination condition for the total cost J_{rel} is set to 0.01 the optimization ends if the total cost changes less than a percent during an iteration. The relative termination condition for the optimization variables x_{rel} is only fulfilled if all of the optimization variables change less then the threshold. Additionally, a absolute termination condition x_{abs} is applied to the optimization variables but is only called into action if one or several

optimization variables are close to zero and the relative criteria therefore don't work properly. During the optimization the constraints on velocity and acceleration are checked every iteration.

The result of the nonlinear optimization is depicted in Figure 2.3. As can be seen, the trajectory passing the same way-points as in Figure 2.2 only needs around 18 seconds where as the initial solution required around 27 second.

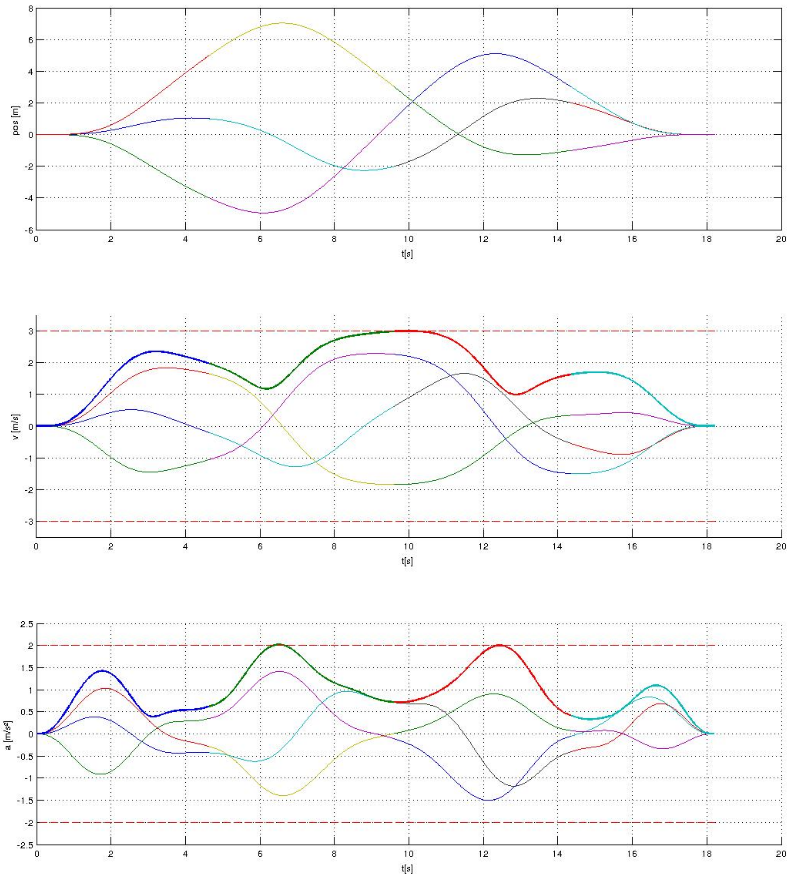

Figure 2.3: Optimized solution: The first subplot shows the position, the second the velocity and the third the acceleration of the optimized solution. A thicker graph represents the 2-norm of the velocity respectively the acceleration.

Chapter 3

RRT

3.1 General

The goal of this Thesis was not only to generate a numerical stable, snap optimized polynomial trajectory but also to explore a dense (indoor) environment and plan an aggressive trajectory in between the obstacles. Hence, the Rapidly-Exploring Random Tree (RRT) algorithm is used to find a collision free straight-line solution through dense environments. The sampling points oft the RRT (or RRT*) algorithm are then used as the vertices in the polynomial optimization.

3.2 Algorithm

3.2.1 RRT

RRT is a computational efficient algorithm to find a path in a high dimensional space by randomly building a space-filling tree. The sampling points are drawn randomly from the sample space and the tree grows incrementally. For each new sample the algorithm attempts to build a collision-free connection to the nearest state in the tree. If a collision-free connection is possible the sample and the connection are added to the tree.

The RRT algorithm can depicted schematically:

1. Sample
2. Find nearest state in tree
3. Try to build a collision-free connection to the nearest state
4. If feasible, add the sampled state and the connection to the tree

3.2.2 RRT*

In contrast to the RRT algorithm the RRT* (or RRT Star) algorithm not only tries to connect to the nearest state in the tree but to several states near the sampled state.

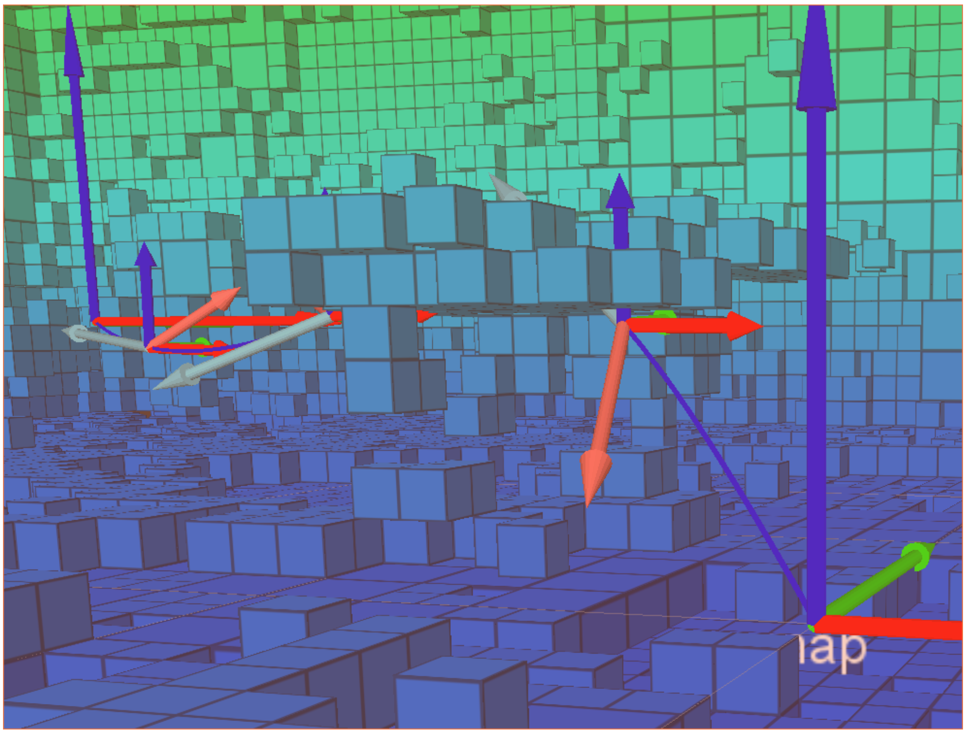

Figure 3.1: Ein Bild.

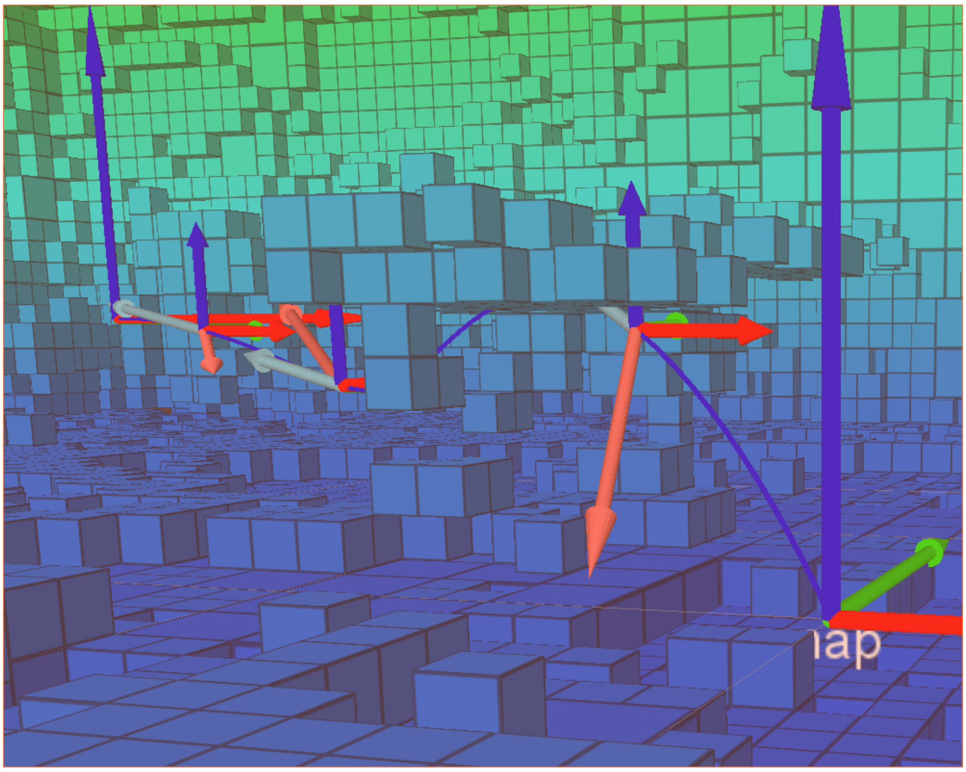

Figure 3.2: Ein Bild.

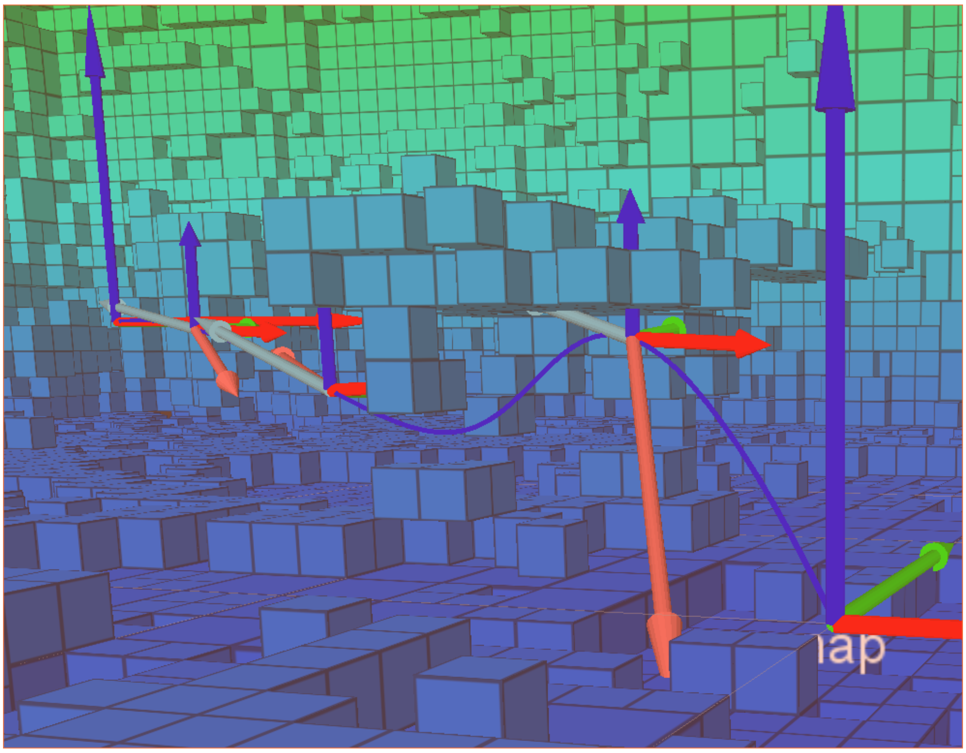

Figure 3.3: Ein Bild.

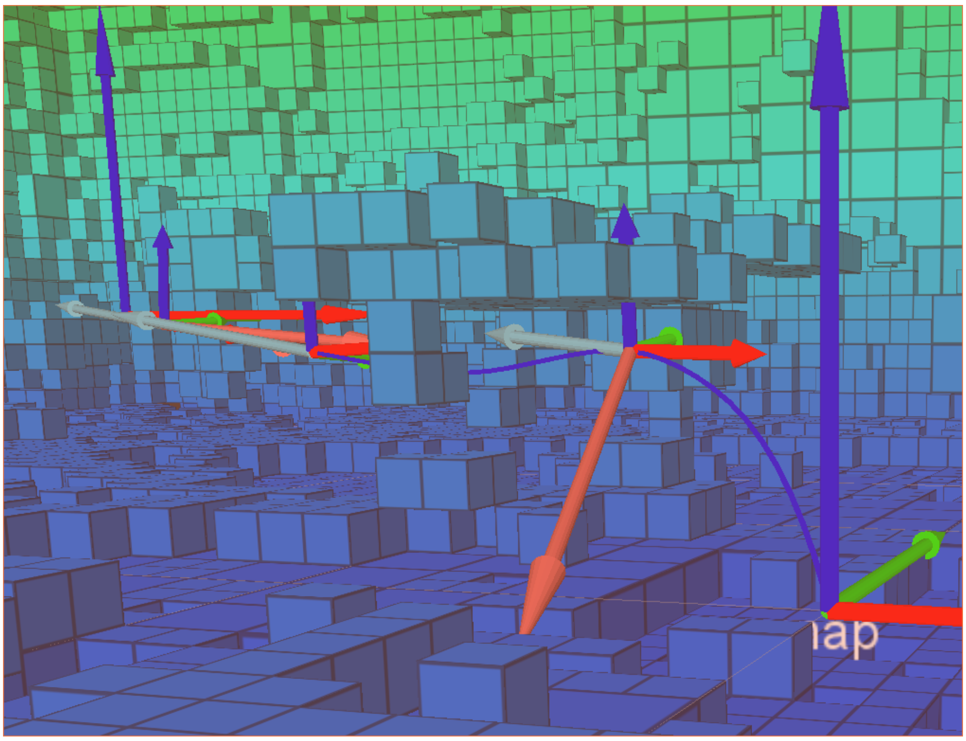

Figure 3.4: Ein Bild.

Appendix A

Irgendwas

Bla bla ...

Bibliography

[1] R. He, A. Bachrach, M. Achtelik, A. Geramifard, D. Gurdan, S. Prentice, J. Stumpf, and N. Roy: *On the design and use of a micro air vehicle to track and avoid adversaries*. The Int. Journal of Robotics Research, vol. 29, pp. 529-546, 2010.

[2] D. Colling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke: *A prototype of an autonomous controller for a quadrotor UAV*. In Proceedings of the European Control Conference, Kos, Greece, 2007, pp. 1-8.

[3] M. Hehn and R. D'Andrea : *Quadrocopter trajectory generation and control*. In International Federation of Automatic Control (IFAC), World Congress 2011, 2011.

[4] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea: *A simple learning strategy for high-speed quadrocopter multi-flips*. In Proc. of the IEEE Int. Conf. on Robotics and Automation, Anchorage, AK, May 2010, pp. 1642-1648.

[5] Y. Bouktir, M. Haddad, and T. Chettibi: *Trajectory Planning for a Quadrotor Helicopter*. In Mediterranean Conference on Control and Automation, Jun. 2008, pp. 1258-1263

[6] D. Mellinger and V. Kumar: *Minimum snap trajectory generation and control for quadrotors*. In International Conference on Robotics and Automation, 2011, pp. 2520-2525.

[7] M. Likhachev, G. Gordon and S. Thrun: *ARA*: Anytime A* with Provable Bounds on Sub-Optimality* . Advances in Neural Information Processing Systems, vol. 16, 2003

[8] C. Richter, A. Bry, and N, Roy: *Polynomial Trajectory Planning for Quadrotor Flight*. In International Conference on Robotics and Automation, 2013.