PLANNING AND NAVIGATION IN DYNAMIC ENVIRONMENTS FOR MOBILE ROBOTS AND MICRO AERIAL VEHICLES

DISSERTATION

zur Erlangung des Doktorgrades (Dr. rer. nat.)

der Mathematisch-Naturwissenschaftlichen Fakultät

der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

MATTHIAS NIEUWENHUISEN

aus Münster

Bonn, November 2018

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

Erster Gutachter:	Prof. Dr. Sven Behnke
Zweiter Gutachter:	Dr. Martin Saska
Tag der Promotion:	02.10.2019
Erscheinungsjahr:	2019

ABSTRACT

Reliable and robust navigation planning and obstacle avoidance is key for the autonomous operation of mobile robots. In contrast to stationary industrial robots that often operate in controlled spaces, planning for mobile robots has to take changing environments and uncertainties into account during plan execution. In this thesis, planning and obstacle avoidance techniques are proposed for a variety of ground and aerial robots. Common to most of the presented approaches is the exploitation of the nature of the underlying problem to achieve short planning times by using multiresolution or hierarchical approaches. Short planning times allow for continuous and fast replanning to take the uncertainty in the environment and robot motion execution into account. The proposed approaches are evaluated in simulation and real-world experiments.

The first part of this thesis addresses planning for mobile ground robots. One contribution is an approach to grasp and object removal planning to pick objects from a transport box with a mobile manipulation robot. In a multistage process, infeasible grasps are pruned in offline and online processing steps. Collision-free endeffector trajectories are planned to the remaining grasps until a valid removal trajectory can be found. An object-centric local multiresolution representation accelerates trajectory planning. The mobile manipulation components are evaluated in an integrated mobile bin-picking system.

Local multiresolution planning is employed for path planning for humanoid soccer robots as well. The used Nao robot is equipped with only relatively low computing power. A resource-efficient path planner including the anticipated movements of opponents on the field is developed as part of this thesis. In soccer games an important subproblem is to reach a position behind the ball to dribble or kick it towards the goal. By the assumption that the opponents have the same intention, an explicit representation of their movements is possible. This leads to paths that facilitate the robot to reach its target position with a higher probability without being disturbed by the other robot. The evaluation for the planner is performed in a physics-based soccer simulation.

The second part of this thesis covers planning and obstacle avoidance for micro aerial vehicles (MAVs), in particular multirotors. To reduce the planning complexity, the planning problem is split into a hierarchy of planners running on different levels of abstraction, i.e., from abstract to detailed environment descriptions and from coarse to fine plans. A complete planning hierarchy for MAVs is presented,

from mission planners for multiple application domains to low-level obstacle avoidance. Missions planned on the top layer are executed by means of coupled allocentric and egocentric path planning. Planning is accelerated by global and local multiresolution representations. The planners can take multiple objectives into account in addition to obstacle costs and path length, e.g., sensor constraints.

The path planners are supplemented by trajectory optimization to achieve dynamically feasible trajectories that can be executed by the underlying controller at higher velocities. With the initialization techniques presented in this thesis, the convergence of the optimization problem is expedited. Furthermore, frequent reoptimization of the initial trajectory allows for the reaction to changes in the environment without planning and optimizing a complete new trajectory.

Fast, reactive obstacle avoidance based on artificial potential fields acts as a safety layer in the presented hierarchy. The obstacle avoidance layer employs egocentric sensor data and can operate at the data acquisition frequency of up to 40 Hz. It can slow-down and stop the MAV in front of obstacles as well as avoid approaching dynamic obstacles.

We evaluate our planning and navigation hierarchy in simulation and with a variety of MAVs in real-world applications, especially outdoor mapping missions, chimney and building inspection, and automated stocktaking.

ZUSAMMENFASSUNG

Zuverlässige und sichere Navigationsplanung und Hindernisvermeidung ist ein wichtiger Baustein für den autonomen Einsatz mobiler Roboter. Im Gegensatz zu klassischen Industrierobotern, die in der Regel in abgetrennten, kontrollierten Bereichen betrieben werden, ist es in der mobilen Robotik unerlässlich, Änderungen in der Umgebung und die Unsicherheit bei der Aktionsausführung zu berücksichtigen. Im Rahmen dieser Dissertation werden Verfahren zur Planung und Hindernisvermeidung für eine Reihe unterschiedlicher Boden- und Flugroboter entwickelt und vorgestellt. Den meisten beschriebenen Ansätzen ist gemein, dass die Struktur der zu lösenden Probleme ausgenutzt wird, um Planungsprozesse zu beschleunigen. Häufig ist es möglich, mit abnehmender Genauigkeit zu planen desto weiter eine Aktion in der Zeit oder im Ort entfernt ist. Dieser Ansatz wird lokale Multiresolution genannt. In anderen Fällen ist eine Zerlegung des Problems in Schichten unterschiedlicher Genauigkeit möglich. Die damit zu erreichende Beschleunigung der Planung ermöglicht ein häufiges Neuplanen und somit die Reaktion auf Änderungen in der Umgebung und Abweichungen bei den ausgeführten Aktionen. Zur Evaluation der vorgestellten Ansätze werden Experimente sowohl in der Simulation als auch mit Robotern durchgeführt.

Der erste Teil dieser Dissertation behandelt Planungsmethoden für mobile Bodenroboter. Um Objekte mit einem mobilen Roboter aus einer Transportkiste zu greifen und zur Weiterverarbeitung zu einem Arbeitsplatz zu liefern, wurde ein System zur Planung möglicher Greifposen und hindernisfreier Endeffektorbahnen entwickelt. In einem mehrstufigen Prozess werden mögliche Griffe an bekannten Objekten erst in mehreren Vorverarbeitungsschritten (offline) und anschließend, passend zu den erfassten Objekten, online identifiziert. Zu den verbleibenden möglichen Griffen werden Endeffektorbahnen geplant und, bei Erfolg, ausgeführt. Die Greif- und Bahnplanung wird durch eine objektzentrische lokale Multiresolutionskarte beschleunigt. Die Einzelkomponenten werden in einem prototypischen Gesamtsystem evaluiert.

Eine weitere Anwendung für die lokale Multiresolutionsplanung ist die Pfadplanung für humanoide Fußballroboter. Zum Einsatz kommen Nao-Roboter, die nur über eine sehr eingeschränkte Rechenleistung verfügen. Durch die Reduktion der Planungskomplexität mit Hilfe der lokalen Multiresolution, wurde die Entwicklung eines Planers ermöglicht, der zusätzlich zur aktuellen Hindernisfreiheit die Bewegung der Gegenspieler auf dem Feld berücksichtigt. Hierbei liegt der Fokus auf einem wichtigen Teilproblem, dem Erreichen einer gu-

ten Schussposition hinter dem Ball. Die Tatsache, dass die Gegenspieler vergleichbare Ziele verfolgen, ermöglicht es, Annahmen über mögliche Laufwege zu treffen. Dadurch ist die Planung von Pfaden möglich, die das Risiko, durch einen Gegenspieler passiv geblockt zu werden, reduzieren, so dass die Schussposition schneller erreicht wird. Dieser Teil der Arbeit wird in einer physikalischen Fußballsimulation evaluiert.

Im zweiten Teil dieser Dissertation werden Methoden zur Planung und Hindernisvermeidung von Multikoptern behandelt. Um die Planungskomplexität zu reduzieren, wird das zu lösenden Planungsproblem hierarchisch zerlegt und durch verschiedene Planungsebenen verarbeitet. Dabei haben höhere Planungsebenen eine abstraktere Weltsicht und werden mit niedriger Frequenz ausgeführt, zum Beispiel die Missionsplanung. Niedrigere Ebenen haben eine Weltsicht, die mehr den Sensordaten entspricht und werden mit höherer Frequenz ausgeführt. Die Granularität der resultierenden Pläne verfeinert sich hierbei auf niedrigeren Ebenen. Im Rahmen dieser Dissertation wurde eine komplette Planungshierarchie für Multikopter entwickelt, von Missionsplanern für verschiedene Anwendungsgebiete bis zu schneller Hindernisvermeidung. Pfade zur Ausführung geplanter Missionen werden durch zwei gekoppelte Planungsebenen erstellt, erst allozentrisch, und dann egozentrisch verfeinert. Hierbei werden ebenfalls globale und lokale Multiresolutionsrepräsentationen zur Beschleunigung der Planung eingesetzt. Zusätzlich zur Hindernisfreiheit und Länge der Pfade können auf diesen Planungsebenen weitere Zielfunktionen berücksichtigt werden, wie zum Beispiel die Berücksichtigung von Sensorcharakteristika.

Ergänzt werden die Planungsebenen durch die Optimierung von Flugbahnen. Diese Flugbahnen berücksichtigen eine angenäherte Flugdynamik und erlauben damit ein schnelleres Verfolgen der optimierten Pfade. Um eine schnelle Konvergenz des Optimierungsproblems zu erreichen, wurde in dieser Arbeit ein Verfahren zur Initialisierung entwickelt. Des Weiteren kommen Methoden zur schnellen Verfeinerung des Optimierungsergebnisses bei Änderungen im Weltzustand zum Einsatz, diese ermöglichen die Reaktion auf neue Hindernisse oder Abweichungen von der Flugbahn, ohne eine komplette Flugbahn neu zu planen und zu optimieren.

Die Sicherheit des durch die Planungs- und Optimierungsebenen erstellten Pfades wird durch eine schnelle, reaktive Hindernisvermeidung gewährleistet. Das Hindernisvermeidungsmodul basiert auf der Methode der künstlichen Potentialfelder. Durch die Verwendung dieser schnellen Methode kombiniert mit der Verwendung von nicht oder nur über kurze Zeiträume aggregierte Sensordaten, ermöglicht die Reaktion auf unbekannte Hindernisse, kurz nachdem diese von den Sensoren wahrgenommen wurden. Dabei kann der Multikopter

abgebremst oder gestoppt werden, und sich von nähernden Hindernissen entfernen.

Die Komponenten der Planungs- und Hindernisvermeidungshierarchie werden sowohl in der Simulation evaluiert, als auch in integrierten Gesamtsystemen mit verschiedenen Multikoptern in realen Anwendungen. Dies sind insbesondere die Kartierung von Innen- und Außenbereichen, die Inspektion von Gebäuden und Schornsteinen sowie die automatisierte Inventur von Lägern.

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor Prof. Dr. Sven Behnke for the opportunity to write my thesis in the Autonomous Intelligent Systems group. I enjoyed the opportunity to work in the interesting and emerging field of robotics, especially aerial robotics.

I'd like to thank my colleagues David Droeschel, Dirk Holz, Jan Quenzel, Marius Beul, and Sebastian Houben for the close collaboration in all aerial robotics related projects that facilitated a major part of the research conducted in this thesis. The fruitful discussions about challenges and solutions within the projects strongly facilitated the development of working robotic systems upon which the work in this thesis could build. Additional thanks goes to Marius Beul for assisting in all flight related experiments as a safety pilot, even if the experiments were not part of his research interests.

For the quick assistance whenever technical problems occurred, or support during experiments and test campaigns was necessary, I'd like to thank Michael Schreiber. He quickly solved virtually any problem and offered help from building components needed for testing, over transport of equipment, to organizing test space.

I'd like to thank Hannes Schulz and Max Schwarz for many interesting discussions, in particular about software tools and writing.

I am grateful for the opportunity to experience the highly motivating and challenging atmosphere of many international robotics competitions as part of my research. For that I would like to thank the numerous members of the team NimbRo for working together as a team at several RoboCups in the @Home and soccer leagues, the European Robotics Challenges (EuRoC), and the Mohamed Bin Zayed International Robotics Challenge (MBZIRC).

Special thanks go to my family which always supported me along my way towards finalizing this thesis.

CONTENTS

1 INTRODUCTION

Within the last decade robots have started to become more and more present in everyday environments. Starting with relatively simple vacuum cleaning robots (Jones, 2006) and guide robots in museums (NMA, 2016) or shops (Gross et al., 2009), robotic systems left the controlled environments of industrial manufacturing cells. Current developments include self-driving cars (Ulrich, 2016)—with driver assistance systems that keep cars in a lane and control their velocity relative to obstacles already on the road (P. E. Ross, 2015)—and flying robots for measurements (Hößler and Landgraf, 2014) and delivery of small objects (Beer, 2016).

In contrast to the controlled environments in manufacturing cells, where robots are usually stopped when a human enters their workspace, the new application scenarios impose new challenges for robot navigation. When deploying autonomous robots in environments not solely controlled by the robot itself—e.g., spaces shared with humans or other agents not connected with the robot—accounting for the dynamics is key for safe operation and effective mission completion.

Even in structured manufacturing environments the demand for more flexible robot use in shared workspaces is rising (Szulewski, 2017). Furthermore, parts processed by humans or other machinery are nor necessarily delivered to a robot work cell in a defined pose neither isolated from each other. Hence, even in these so far controlled environments dealing with uncertainty and dynamics becomes necessary for efficient processes.

In other applications the environment itself might not be known in advance or just in a very coarse manner. Maps of the environment might just be built at the time the robot perceives a part of the environment for the first time. Instead of starting with empty maps, they can be based on initial models that are refined when observed. Examples for such models are coarse building outlines or bounding shapes of objects.

To address these challenges robots require abilities to perceive the state of their environment, plan a sequence of actions to achieve their goal or even fulfill multiple objectives, and to react on changes in the world or deviations from the planned action sequence. In this thesis, we address the planning and action generation parts in this sense-plan-act loop.

We categorize aforementioned changes in the environment model and deviations into four major classes with some typical attributes:

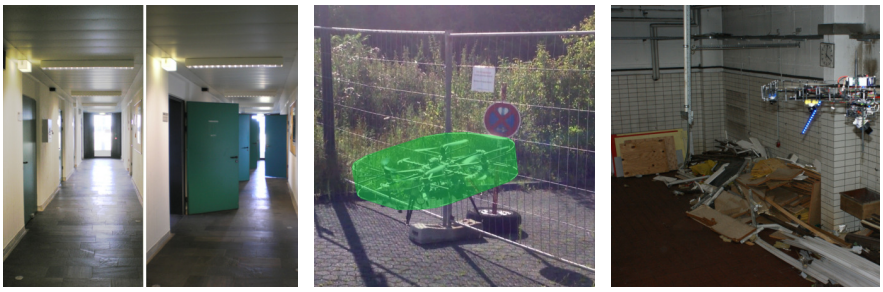


Figure 1.1: Static and semi-static objects can be collision hazards even in known environments: doors can change the environment significantly, new obstacles can appear between visits of an area, and small objects might not have been perceived during initial mapping of an environment.

LARGE SCALE ENVIRONMENT CHANGES Major parts of the environment model change, e.g., due to new perceptions of so far unknown areas or changed topology caused by blocked or newly opened passages. These changes often require complete new global planning. Typically, such large changes appear over long time scales or can be perceived well in advance. Hence, only a relatively slow reaction is required. Finding new global solutions might require a significant amount of time.

LOCAL ENVIRONMENT AND ROBOT STATE CHANGES Locally perceived changes of the environment that can often be incorporated into the current action sequence by local deviations from the global plan. These include smaller obstacles that cannot be perceived from far away or low-frequency changes of the environment. The changes happen typically on a medium time scale and reactions—and consequently planning times—have to be quicker than for large scale changes. Due to the locality the search-space for a solution is typically smaller.

DYNAMIC CHANGES OF ENVIRONMENT OR ROBOT STATE This class subsumes quick dynamic changes in the world model—including obstacles that are static but can only be perceived short before an immanent collision—and deviations from the planned action sequence by external influences. These changes typically require an immediate reaction to recover to a safe state. Quick action selection can typically be achieved by leaving restoration of a consistent action sequence from the safe state to achieving a goal to higher-level components.

SMALL DEVIATIONS FROM PLANNED ACTION SEQUENCE This class of changes occurs continuously and has to be perceived and corrected immediately. Usually this is performed by low-level control layers with only very limited world knowledge.

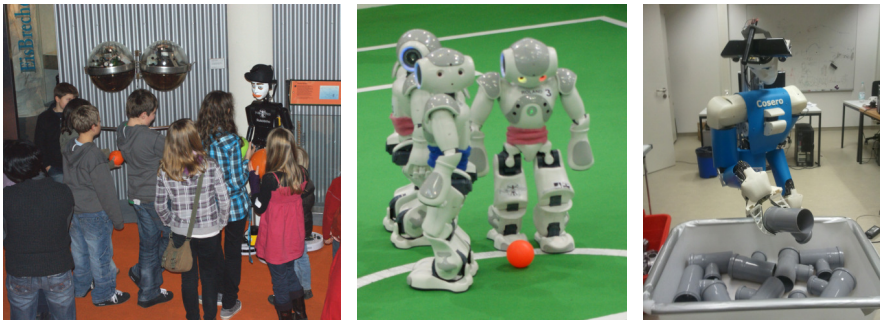


Figure 1.2: The ability to navigate in dynamic environments is key for robots in many applications. Persons or other robots can move unpredictably or block ways. Also, objects might move, e.g., when clearing a pile of unordered objects from a box.

This class of changes is only addressed indirectly by corrections of larger deviations in the other classes in this thesis.

Figure 1.1 shows some examples of possible changes in relatively static environments. Examples for these changes are doors and gates that can be opened or closed, new obstacles can appear as fencing or moved furniture, vegetation changes over time, and environment models can be incomplete. Common to these changes is that they are typically slow and predictable and often do not change much during the duration of a single task or mission.

Dynamic changes in the environment are caused by moving persons or animals in the workspace of the robot, other autonomous agents, or (unintended) modifications caused by the robot actions, e.g., by manipulating objects in a pile. These changes are typically so fast that they have to be taken into account during the robot task execution. To some extent these changes can be predicted by communication or dynamic world models, but in general the prediction is only valid for a short period of time, if at all. Figure 1.2 depicts cases of dynamic changes of the environment. In addition, the execution of planned actions is often not perfect such that a robot deviates from an initial plan with increasing task duration. This might render executing the initial plan infeasible after a while.

We address the uncertainties in the environment and robot state by reducing the complexity of the planning problems to facilitate short planning times and allow for frequent replanning based on updated environment models and robot state. Furthermore, if we can reason about the nature of the dynamics, this knowledge can aid robot navigation.

In the following chapters, we will present planning methods for ground and flying robots that implicitly or explicitly address the dynamic nature of the environment.

1.1 LIST OF CONTRIBUTIONS

In this thesis, we present integrated ground and aerial robotic systems for a variety of application domains. The focus of this thesis lies on the navigation parts of these systems. The key contributions are:

- Planning with an explicit representation of other robot movements for the Nao. To achieve this with the very limited processing power of the Nao V3, we employ local multiresolution techniques to represent space and time dimensions.
- Employing local multiresolution obstacle representations and multi-stage planning to speed up mobile bin-picking. Grasp and trajectory planning for an anthropomorphic mobile robot are split into offline and online planning stages. We use an object-centric local multiresolution representation to detect collisions.
- A complete 3D planning and navigation hierarchy for MAVs. The hierarchy starts with slower, abstract layers. The plans are refined in layers with increasing frequency and less abstraction to generate motion commands on the lowest layer.
- Tailored mission planners for new MAV applications. The applications are 3D mapping, chimney inspection, and autonomous stocktaking in a warehouse. We evaluate integrated MAV systems in these application domains.
- Fast trajectory optimization for MAVs that facilitates frequent reoptimization to react on perceptions and disturbances. A good initialization of the optimization problem accelerates the initial convergence to a feasible allocentric trajectory that can be followed open-loop by a low-level controller. By continuous reoptimization the trajectories are improved based on updated models. Local multiresolution in the time-dimension facilitates a reoptimization frequency comparable to the control frequency.
- Planning and optimization for additional objectives. These objectives include restricting MAV movements to the sensor field of view (FoV) and keeping sufficient structure required for localization in sight.
- Fast reactive obstacle avoidance for MAVs that operates at the obstacle sensor frequency. A low-level obstacle avoidance layer quickly reacting on sensor inputs ensures safe flight independent of good localization and correct high-level plans.

1.2 THESIS OUTLINE

This thesis is divided into two major parts. After a general overview over the topic of the thesis, the first part covers planning for ground

robots, and the second part covers planning and navigation for MAVs. The chapters in Part I are self-contained, including related work and discussion of the results. Part II follows a different structure with a joint discussion of related work and results.

In detail this thesis is structured as follows:

After this introduction chapter, in the next chapter we give an overview over some planning representations and introduce the concept of multiresolution. Furthermore, we describe basics of the employed planning algorithms. These basics are employed throughout this thesis at multiple occurrences.

In Part I, we focus on ground robots. There, we give an overview over planning for anthropomorphic ground robots.

As an example application for dynamics-aware path planning, we developed intention projection-based planning in a soccer domain in Chapter 3. Following obstacle-free paths towards the ball and avoiding opponents while dribbling are key skills to win soccer games. These tasks are challenging as the environment in soccer games is highly dynamic and the opponents will often be very close or deliberately blocking the path. Thus, exact plans will likely become invalid in short time and continuous replanning is necessary. Consequently, path planning algorithms have to be fast. We employ local multiresolution planning representations incorporating a time dimension, and we predict the opponent's movement by projecting the planning robot's intentions to the opponents.

In Chapter 4, we describe our approach to grasping individual objects from an unordered pile in a box with an anthropomorphic mobile robot. We present a new framework to grasp objects composed of shape primitives like cylinders and spheres. We implement object grasping based on the shape primitives in an efficient multi-stage process that successively prunes infeasible grasps in tests of increasing complexity. Grasps and arm motions are planned in an efficient local multiresolution height map. With our approach, our service robot can grasp object compounds from piles of objects, e.g., in transport boxes. All components are integrated and evaluated in a bin picking and part delivery task.

In Part II, we extend these techniques to a hierarchical planning and obstacle avoidance system for MAV navigation in 3D space.

We address multiple application scenarios for autonomous MAV flight. We motivate these applications in Chapter 5. The chapter contains the discussion of related work to our approaches to autonomous MAV operation. We give an overview of our complete hierarchy for planning and navigation from mission planning to obstacle avoidance. All employed MAV systems are described there as well.

In Chapter 6, we detail our application-specific mission planners. We present planners for outdoor mapping missions, automated stocktaking, and the inspection of chimneys and buildings. The allocentric

and egocentric path planners for executing the missions are presented in Chapter 7. This includes multiresolution planning and planning in the sensor FoV for safe navigation.

For the generation of dynamically smooth trajectories, we developed a trajectory optimization layer built upon our path planner. This layer is detailed in Chapter 8. Here, we present initialization techniques for faster convergence as well as frequent multiresolution reoptimization of the trajectory during the flight.

Our fast, reactive obstacle avoidance—acting as a low-level safety layer—is described in Chapter 9. On this safety layer, we employ artificial potential fields to quickly react on newly perceived and dynamic obstacles to slow-down the MAV or push it away from obstacles.

We evaluate the individual components and the integrated systems in Chapter 10, followed by a discussion of Part II in Chapter 11.

In Part III, we discuss the work presented in both major parts and give directions for future research.

A list with links to all referenced videos in this thesis can be found online at `www.nieuwenhuisen.de/thesis`.

1.3 PUBLICATIONS

Major parts of this thesis have been published before as peer-reviewed conference papers or journal articles. The relevant publications per part are listed here.

Part I Grasp and motion planning for ground robots

- Matthias Nieuwenhuisen, Ricarda Steffens, and Sven Behnke (2012a). "Local multiresolution path planning in soccer games based on projected intentions." In: *RoboCup 2011: Robot Soccer World Cup XV*. Ed. by Thomas Röfer, N. Michael Mayer, Jesus Savage, and Uluç Saranlı. Vol. 7416. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 495–506

- Matthias Nieuwenhuisen, Jörg Stückler, Alexander Berner, Reinhard Klein, and Sven Behnke (2012b). "Shape-primitive based object recognition and grasping." In: *Proceedings of the German Conference on Robotics (ROBOTIK)*

- Matthias Nieuwenhuisen, David Droeschel, Dirk Holz, Jörg Stückler, Alexander Berner, Jun Li, Reinhard Klein, and Sven Behnke (2013a). "Mobile bin picking with an anthropomorphic service robot." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*

Chapter 3 extends the work of a supervised Bachelor thesis on multiresolution path planning in dynamic environments for the Standard Platform League (SPL) by Steffens (2010).

Part II Multi-layered navigation for micro aerial vehicles

- Matthias Nieuwenhuisen, David Droeschel, Johannes Schneider, Dirk Holz, Thomas Läbe, and Sven Behnke (2013b). "Multimodal obstacle detection and collision avoidance for micro aerial vehicles." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*
- Matthias Nieuwenhuisen, Mark Schadler, and Sven Behnke (2013c). "Predictive potential field-based collision avoidance for multicopters." In: *International Arch. Photogramm. Remote Sens. Spatial Inf. Sci. (ISPRS)*. Vol. XL-1/W2, pp. 293–298
- Matthias Nieuwenhuisen and Sven Behnke (2014a). "Hierarchical planning with 3D local multiresolution obstacle avoidance for micro aerial vehicles." In: *Proceedings of the Joint International Symposium on Robotics (ISR) and the German Conference on Robotics (ROBOTIK)*
- Matthias Nieuwenhuisen and Sven Behnke (2014b). "Layered mission and path planning for MAV navigation with partial environment knowledge." In: *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*
- Matthias Nieuwenhuisen and Sven Behnke (2015). "3D planning and trajectory optimization for real-time generation of smooth MAV trajectories." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*
- Matthias Nieuwenhuisen, David Droeschel, Marius Beul, and Sven Behnke (2016). "Autonomous navigation for micro aerial vehicles in complex GNSS-denied environments." In: *Journal of Intelligent & Robotic Systems* 84.1, pp. 199–216
- Matthias Nieuwenhuisen and Sven Behnke (2016). "Local multiresolution trajectory optimization for micro aerial vehicles employing continuous curvature transitions." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*
- David Droeschel, Matthias Nieuwenhuisen, Marius Beul, Dirk Holz, Jörg Stückler, and Sven Behnke (2016). "Multi-Layered Mapping and Navigation for Autonomous Micro Aerial Vehicles." In: *Journal of Field Robotics* 33.4, pp. 451–475
- Matthias Nieuwenhuisen, Jan Quenzel, Marius Beul, David Droeschel, Sebastian Houben, and Sven Behnke (2017). "ChimneySpector: Autonomous MAV-based indoor chimney inspection employing 3D laser localization and textured surface reconstruction." In: *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*

- Jan Quenzel, Matthias Nieuwenhuisen, David Droeschel, Marius Beul, Sebastian Houben, and Sven Behnke (2018). "Autonomous MAV-based indoor chimney inspection with 3D laser localization and textured surface reconstruction." In: *Journal of Intelligent & Robotic Systems*. Available online

- Marius Beul, David Droeschel, Matthias Nieuwenhuisen, Jan Quenzel, Sebastian Houben, and Sven Behnke (2018). "Fast autonomous flight in warehouses for inventory applications." In: *IEEE Robotics and Automation Letters* 3 (4), pp. 3121–3128

- Matthias Nieuwenhuisen and Sven Behnke (2019). "Search-based 3D planning and trajectory optimization for safe micro aerial vehicle flight under sensor visibility constraints." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*

2

PATH PLANNING AND PLANNING REPRESENTATIONS

In this chapter, we give an overview over commonly employed planning representations and the A* path planner. Variants of the A* path planner are employed at different parts of this thesis.

Discrete representations of the state space are often used as base for planning, particularly for search-based approaches. Common discretizations include the spatial discretization of the environment geometry (occupancy grid maps), discretization of time, and discrete robot actions. While coarse representations—low spatial resolution, large time steps and small number of actions—are generally easy to compute, finer representations quickly hit memory limits and render search-based approaches intractable. To make planning algorithms tractable, multiresolution approaches have been developed that reduce the complexity and size of problems. The general idea of multiresolution is to represent a state or configuration as coarse as possible and as fine as necessary. Multiresolution approaches can be classified into global and local approaches.

2.1 UNIFORM REPRESENTATIONS

Uniform occupancy grid maps are a commonly used spatial environment representation. The environment is discretized into equally-sized cells representing either a binary state—occupied or free—or an occupancy likelihood. Grid cells can be accessed computationally efficient by a simple linear mapping between spatial coordinates and grid indices in constant time.

Graph search-based planning algorithms, e.g., A* search (Hart et al., 1968), are particularly straightforward applied to grid maps as grid cell centers can be interpreted as nodes of a search graph with edges to all eight adjacent cells of the Moore neighborhood in 2D grids or to all $3^d - 1$ cells of its generalization to d dimensions. Due to the simple structure of the grid, there is no need to explicitly model the cell connectivity. Furthermore, the number of different edge lengths in the search graph is strictly limited to the number of dimensions of the grid, e.g., straight along one grid axis, diagonal in the plane, and diagonal in three dimensions in a 3D grid. This allows for the (partial) precomputation of traversal costs. Uniform grids can be easily implemented because of their simplicity, which allows arbitrary environments to be represented equally accurate, up to a freely chosen resolution. This makes uniform grids a good choice

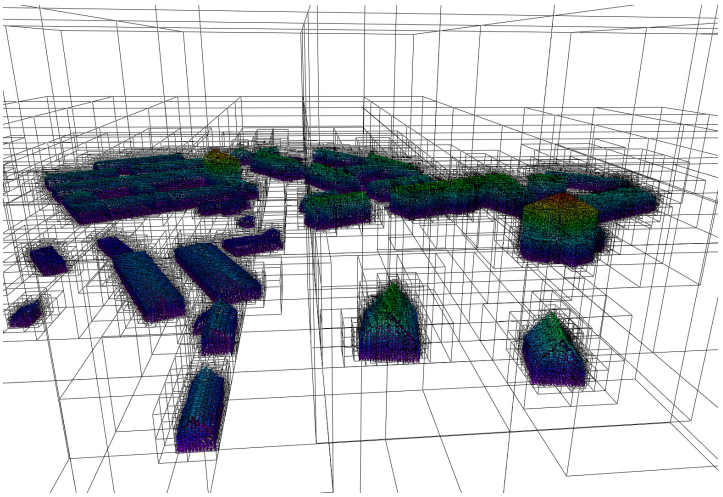


Figure 2.1: OctoMap of a village. Global multiresolution allows for efficient representation of large areas with high resolution—25 cm in this map. The tree structure is depicted by black cell outlines.

for many applications. A major disadvantage is their computational complexity, which does not scale well with increasing grid size and dimensionality.

Uniform discretizations of the time dimension can be modeled equally with directed implicit edges between the cells.

2.2 GLOBAL MULTIRESOLUTION

Global multiresolution representations adapt their resolution to the complexity of the represented entity. One well-known example of a spatial global multiresolution representation is the quadtree in 2D or octree in 3D (Meagher, 1980). Octrees partition the represented space into eight octants recursively as long as the information in an octant is consistent—e.g., fully occupied or fully free space—or the maximum resolution is reached. Thus, large areas of similar space can be represented efficiently without loosing much accuracy. In contrast to uniform representations, the structure of the grid needs to be modeled explicitly introducing some overhead. Nevertheless, virtually all environments are dominated by larger volumes of similar occupancy, e.g., free space and solid objects. The OctoMap proposed by Hornung et al. (2013) is a 3D occupancy grid map based on an octree, depicted in Figure 2.1. We use OctoMap representations as an allocentric environment representation in several parts of this thesis. Planning algorithms based on global multiresolution include Multiresolution Field D* by Ferguson and Stentz (2006) and kinodynamic motion planning by interior-exterior cell exploration (KPIECE) by Şucan and Kavraki (2008).

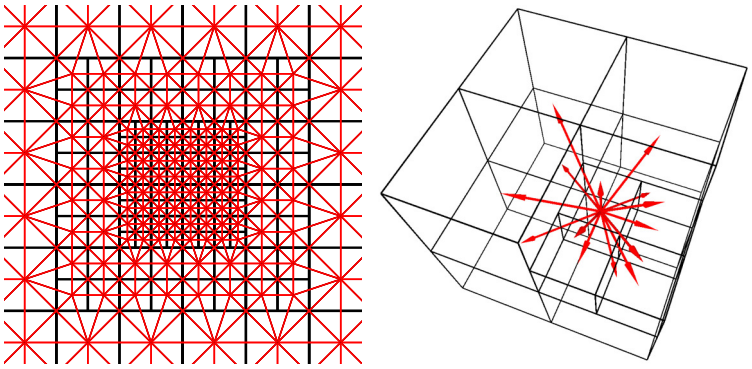


Figure 2.2: Connectivity in local multiresolution grid. Centered grids with different resolutions are embedded into each other. The edge connectivity between cells from different levels differs from the connectivity in a uniform grid. Red lines and arrows depict the edges between neighboring cells. Left: Excerpt of 2D grid. The robot is in the center of the grid. Right: Cut through 3D grid. In the 3D case only the connectivity of one example cell is depicted by red arrows.

2.3 LOCAL MULTIRESOLUTION

Local multiresolution refers to approaches where the resolution of the represented entity is higher the more local it is. For spatial representations this translates to a high resolution in the center of a grid-based map and decreasing resolution for outer grid cells. In the time domain the resolution of the near future is high and decreases with increasing time.

Our local multiresolution representations are inspired by the work of Behnke (2004), a method for resource-efficient path planning. The spatial environment representation consists of multiple origin-centered grids with different resolutions embedded into each other. With increasing distance to the origin, the grid resolution decreases.

We embed grids with M cells in each dimension into each other. A grid at multiresolution level l has a cell size s_l of $2^l s_{min}$, where s_{min} is the cell size of the innermost grid. Thus, the resolution of the embedded grids is higher than the resolution of the containing outer grids. The embedded grid with dimensionality d replaces the cells with the indices $\left[\frac{M}{4} : \frac{3M}{4}\right]^d$ of the outer grid. Consequently, it covers a quarter of the surface of the outer grid in 2D or an eighth of the volume in 3D, We repeat this until the minimal cell size is reached. The resulting multiresolution representation containing LM^d cells covers the same surface or volume as a uniform d-dimensional grid with N grid cells per dimension. Here, $L = \log_2(N/M) + 1$ is the number of necessary resolution levels given desired uniform and multiresolution discretizations. Figure 2.2 depicts 2D and 3D local multiresolution grids.

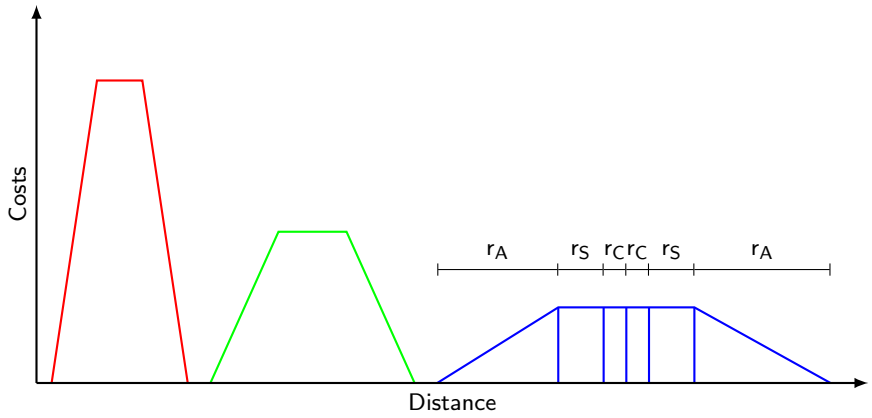


Figure 2.3: We model obstacles in the local multiresolution grid as a fixed core r_C, a safety area with maximum costs r_S, and an avoidance zone with linear decreasing costs r_A. With increasing distance to the grid origin the radii of these areas increase and their maximum cost decreases to account for the uncertainty in measurements (red: close, green: medium, blue: far away obstacle).

The neighborhood of an inner grid cell consists of $3^d - 1$ neighbors, similar to the neighborhood of the uniform grid. However, the cells at the border to a coarser grid have fewer neighbors, whereas the grid cells at the border to a finer grid have more neighboring cells. In the 2D case, for example, cells at the border to coarser grids have seven neighbors at the edges and six neighbors at the corner and the grid cells at the border to finer grids have nine neighbors and eight neighbors, respectively. Higher dimensional grids show even more special cases.

Thus, cell adjacencies cannot be handled implicitly as in the uniform grid, but have to be modeled explicitly. In addition to these special cases, edge lengths differ between resolution levels causing more overhead for precomputations of traversal costs. Both overheads are compensated by a significantly reduced number of overall grid cells.

Figure 2.3 shows our obstacle model composed of: a core radius r_C of the perceived obstacle enlarged by the approximate robot radius, a distance-dependent minimum safety distance r_S with maximum costs, and an area $r_A = 2(r_C + r_S)$ with decreasing costs that can be traded off against other costs, e.g., path length. For a distance d between a grid cell center c and the obstacle center o the obstacle costs h_o are given by

$$h_o(d) = \begin{cases} h_{max} & \text{if } d \leqslant (r_C + r_S) \\ h_{max}\left(1 - \frac{d-(r_C+r_S)}{2(r_C+r_S)}\right) & \text{if } (r_C + r_S) < d < 3(r_C + r_S) \\ 0 & \text{otherwise} \end{cases}. \tag{2.1}$$

To match the multiresolution structure of the underlying grid and to account for the uncertainties of farther away perceptions, we increase the radius of obstacles with increasing distance to the grid center. To achieve this, the distance-dependent part

$$r_S = r_C + 0.1 \cdot \|o\|$$

grows linearly with the distance of the obstacle o to the map center. The maximum obstacle costs h_{max} are defined as

$$h_{max} = \frac{r_C^2}{(r_C + r_S)^2}.$$

Thus, the integral of the obstacle cost stays constant by reducing its maximum costs with increasing radius, illustrated by the three differently colored cost functions in Figure 2.3. A constant cost integral is required to represent that the obstacle costs when covering the whole area of the uncertain obstacle position is the same as covering a smaller area with a certain obstacle position.

In contrast to global approaches, the structure of the representation is solely defined by the locality of grid cells and not by the structure of the represented environment. Instead of resembling the complexity of the represented environment, local multiresolution representations model, e.g., the uncertainties caused by local sensing with only relative precision and by the prediction of the near future. Local multiresolution planning exploits the fact that the world changes continuously while the plan is processed. Another use case of local multiresolution planning is precise planning in the vicinity of a goal and approximate planning while approaching it.

Analogous to uniform representations, multiresolution representations can be employed for time discretizations.

2.4 PATH PLANNING

For motion and path planning a variety of planning algorithms exist in the literature. Many state-of-the-art algorithms are either sampling-based or search-based planners. Sampling-based planners, e.g., rapidly-exploring random trees (RRTs) (LaValle, 1998) and KPIECE (Şucan and Kavraki, 2008), are primarily used for high-dimensional planning problems where modeling and/or searching the state space is intractable. The advantages regarding computation time and memory footprint come at the cost of only stochastic completeness—existing solutions are not necessarily found in the given planning time and found solutions are not optimal. Search-based planners are complete and optimal, but suffer from the curse-of-dimensionality, i.e., the problem complexity increases exponentially with the number of state space dimensions.

Algorithm 1 A* algorithm with closed list.

```
1:  procedure A* SEARCH(planning_grid, start_node, target_node)
2:      Initialize open_list                              ▷ Priority queue
3:      start_node.costs ← 0
4:      add start_node to open_list
5:      while open_list not empty do
6:          current_node ← first element from open_list
7:          remove first element from open_list
8:          if current_node = goal_node then
9:              return best solution
10:         end if
11:         if current_node not in closed list then
12:             EXPAND_NODE(current_node)
13:             add current_node to closed list
14:         end if
15:     end while                                         ▷ No solution found
16: end procedure
```

A widely used search-based planning algorithm is A* search by Hart et al. (1968)—an informed extension of Dijkstra's algorithm (Dijkstra, 1959). We distinguish between the concepts of a path and a trajectory. A path is an ordered set of segments connecting a start and a goal configuration, e.g., connecting 3D points in Euclidean space. A trajectory is a path with timing information. Thus, a trajectory contains the derivatives of the path, e.g., its velocities and accelerations. Given an undirected graph with edge weights, A* finds a cost-optimal path from a start to a goal node in the graph.

The start node is initialized with zero costs, other nodes have infinite costs. Dijkstra's algorithm always expands the node with the minimal costs that has not been expanded before until a path to the goal node has been found. To achieve this a priority queue is maintained with the costs to reach a node as key. By contrast, in the A* algorithm, the graph search is directed towards a goal node by a heuristic that estimates the remaining costs to reach the goal from a given node. The key for the priority queue is in this case the sum of the costs to reach a node plus the heuristics value. Thus, nodes are sorted according to the estimated costs a minimal path would have from start to goal via this node. If the heuristic is admissible, i.e., it never overestimates the remaining costs, the planner remains complete. For the special case of a heuristic that is always zero the A* search is equal to the Dijkstra algorithm.

In general, the A* search works with arbitrary undirected, weighted graphs. For path planning, grid graphs embedded into the environment representation are often employed. The edge weights then rep-

Algorithm 2 Node expansion procedure of the A* algorithm.

```
1:  procedure EXPAND_NODE(open_list, node)
2:      successors ← all neighbors of node not in closed list
3:      for all successors as successor do
4:          if successor is in obstacle then
5:              add successor to closed list
6:              continue with next successor node
7:          end if
8:          g ← node.value+ edge costs          ▷ edge length × costs
9:          if successor is in open_list ∧ g ⩾ successor.value then
10:             continue with next successor node
11:                 ▷ node is already in open_list with a better g value
12:         end if
13:         successor.value ← g
14:         successor.score ← g+ heuristic from successor to goal
15:         insert successor into open_list
16:     end for
17: end procedure
```

resent, e.g., distances between nodes in the state space and proximity to obstacles.

Algorithm 1 shows a variant of the A* algorithm with closed list. The cost to reach a node is called the node value, the sum of costs and heuristic is called the node score. Nodes that have been expanded once are added to the closed list as by definition no shorter path to these can be found. Thus, nodes on the closed list are not considered for further expansions in the remaining iterations. The search terminates when the goal node has to be expanded in the next step. In this case no shorter path to the goal can be found. By backtracking from goal node to start node the best solution is determined and returned. This solution can either be determined by explicit backtracking from goal to start by pointers to the parents of the nodes or implicitly by following the least cost path. While the first method is always possible, the later method is only possible if the expansion step can be inverted uniquely.

Algorithm 2 details the node expansion step. The cost values and scores of all neighbor nodes that are not in the closed list are calculated and the nodes are added to the open list priority queue if the costs are not prohibitively high, e.g., if the nodes are too close to obstacles. Costs to traverse an edge are often calculated by multiplying the edge length in a grid voxel with the grid value, e.g., a value depending on the distance to the next obstacle.

Many search-based planners build upon the basic A* algorithm. Examples are, e.g., D* Lite (S. Koenig and Likhachev, 2005) which inverts the search from goal to start and can efficiently repair paths

when obstacles in the vicinity of the start node are perceived, and ARA* (Likhachev et al., 2004) which starts with a suboptimal path and improves the path as long as a time budget is not exceeded.

Modified versions of the A* path planner are widely used throughout this thesis. Other planners employed for specific problems are introduced in the corresponding chapters.

Part I

GRASP AND MOTION PLANNING FOR GROUND ROBOTS

PLANNING AND NAVIGATION FOR GROUND ROBOTS

In this part of the thesis, we investigate multiresolution techniques for planning and collision avoidance in the context of ground robots. The navigation and motion generation for ground robots—while being a research topic since the field of robotics has been established decades ago—is still an important aspect of current robotic developments.

Early approaches to navigation of mobile robots focused on shortest paths in completely mapped or modeled, static environments (Lozano-Pérez and Wesley, 1979). For robots to be able to operate in coexistence with humans outside of controlled lab environments where, e.g., furniture or other objects are moved and people are present, navigation approaches have been extended to navigate in (partially) unknown (Stentz, 1994) and even dynamic environments (Fox et al., 1997). In the next chapter, we focus on the domain of humanoid soccer robots as an example for 2D path planning. The properties of this particular application are that virtually no static environment is present, at least not within the bounds of the soccer field, but a number of dynamic obstacles in form of teammates and opponents that move on the field. Furthermore, the opponents—in contrast to most other agents in robotic workspaces that can share the effort of avoiding each other (Silva and Fraichard, 2017)—are uncooperative and even deliberately obstructive.

Another important ability for ground robots is picking and placing objects or tools. In the first applications of stationary industrial robots, arm trajectories were programmed as a list of trajectory points to pass through by an operator (Pieper, 1968). Later, trajectories were generated online with consideration of self-collisions and collisions with the environment (Maciejewski and Klein, 1985; Pieper, 1968). For more flexibility, manipulator arms have been attached to mobile robot platforms (Khatib, 1999) to enlarge their workspace significantly. In Chapter 4, we describe an integrated, autonomous mobile manipulation system for bin-picking, i.e., picking objects from an unordered pile of objects from a transport box. This requires flexible adaption of manipulator trajectories to many different configurations of the environment. To speed up the planning of final grasps and object removal, we employ local multiresolution techniques.

Due to the different nature of the applications robot soccer and mobile bin-picking, the two chapters of this thesis part are self-contained. The related work is accordingly discussed in the corresponding chapters.

3

RESOURCE EFFICIENT CONTINUOUS MOTION PLANNING BASED ON PROJECTED INTENTIONS

In the domain of humanoid robot soccer competitions, an important skill for the robots is the ability to reach a position behind the ball facing the opponent's goal. Collision-free movements are not only important because of possible penalties for pushing the opponent, but also because collisions can easily lead to a fall and significantly slow down the robot. This is particularly challenging, as the environment is very dynamic and multiple agents compete to achieve similar objectives. Figure 3.1 shows a typical game situation, where three robots can see the ball and are trying to reach positions close to it.

One approach to navigation on a soccer field is reactive action generation. Perceptions are directly mapped to actions, i.e., direction and velocity of the robot motion, by incorporating the target position and the position of obstacles. The mapping may depend on additional factors, like the role assigned to the robot or the game state, but it does not consider the (foreseeable) future. This can lead to inefficient obstacle avoidance, e.g., the robot passes an obstacle on the side closer to its target just to be blocked by the next obstacle in the same direction.

On small mobile robot platforms—like the Nao, used in this work—the computational resources are often limited due to weight and power constraints. Accordingly, exact planning—even for relatively small problem instances—is not possible in real-time. Moreover, performing time-consuming planning, and committing to a long-term plan, is not an option in highly dynamic domains like soccer. Due to the limited capabilities of the robot sensors, it is furthermore not possible to estimate precise obstacle positions. As a consequence, not only the dynamic environment has to be taken into account, but path planning has to deal with uncertainties too.

We address these challenges by using approximate path planning with replanning every time a new state of the environment is perceived. With increasing time since the last perception, the prediction of the world state becomes more uncertain. Consequently, planning steps in the far future should be more approximate than planning steps that have to be executed immediately, as the former will likely be invalid at the time they are executed. In order to reduce the complexity of the plan representations, we employ local multiresolution methods, namely, a local multiresolution grid and a log-polar grid. Both representations have in common that the resolution decreases with increasing distance from the robot.

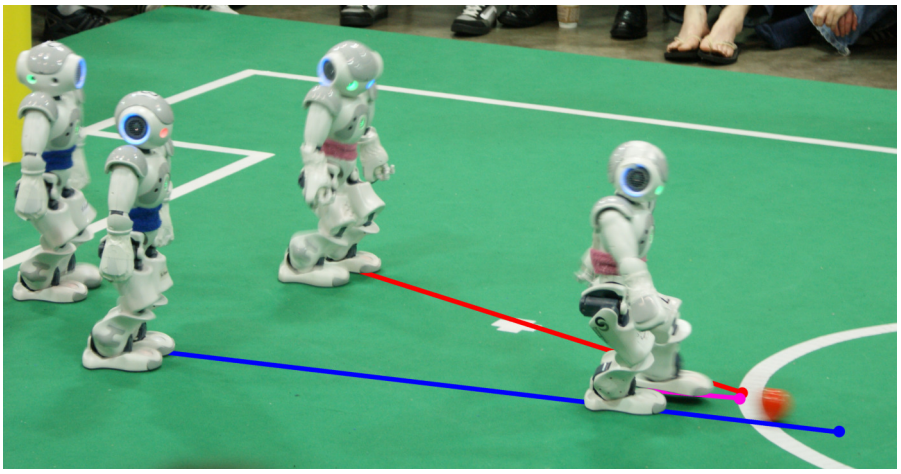


Figure 3.1: Typical situation in a humanoid robot soccer game. Several robots see the ball and try to reach poses behind the ball, facing towards the opponent's goal. The direct paths to the intended target positions are depicted by colored lines.

In arbitrary situations, it is hard to predict the movements of dynamic obstacles without tracking them. Thus, these movements are often modeled as random noise. In contrast, the movements of the opponent in soccer games are aimed at scoring goals. We employ this fact to gain a better estimation of probable obstacle trajectories by projecting the intentions of the planning robot to other field players.

3.1 RELATED WORK

In soccer games, we historically employed a hierarchical reactive approach to control the robot motion (Behnke and Stückler, 2008). The approach presented in this thesis is based on planning however. It considers the foreseeable future to determine obstacle-free robot paths. Continuous replanning allows for constant incorporation of the most recent sensory information, and for quick reactions to changes in the environment.

Many planning-based systems exist in the literature for 2D robot navigation. The key challenge is the computational complexity of real-time planning and execution on a computationally limited robot. Kaelbling and Lozano-Pérez (2011) reduce the complexity of task planning by top-down hierarchical planning. In their approach, an agent commits to a high-level plan. The refinement of abstract actions is performed at the moment an agent reaches them during plan execution. We follow the same assumption that there is likely a valid refinement at the time an abstract action has to be concretized, and that every plan can be reversed without huge costs in case there is no such refinement.

Karkowski and Bennewitz (2016) plan footsteps for the Nao robot in cluttered environments using an RGB-D sensor mounted on the robot head. Whereas the plans are mostly computed by external hardware, the authors report that the algorithm can be ported to a Nao onboard computer more recent than the one employed in this thesis and runs at 10 Hz there. We make the assumption that in our domain the robot can be abstracted to an omnidirectionally moving platform without planning individual footsteps as the environment is not cluttered.

Strategic positioning of soccer robots has been proposed by Kaden et al. (2013). They weight Voronoi cells in a map to find good positions for the different roles of field players. This map is also used to plan paths that can, e.g., avoid a blockage of the line-of-sight between a striker and the opponent's goal. We, in addition, predict the movement of an opponent to employ this knowledge in planning.

A method for resource-saving path planning is the local multiresolution Cartesian grid described in Section 2.3. This representation resembles well the uncertainty in future actions by as the planning resolution decreases with increasing distance to the robot and, thus, become more approximate. In addition, it was designed for soccer robots (Behnke, 2004) and, consequently, for the problem domain considered here.

Apart from Cartesian occupancy grid maps (Thrun et al., 2006), polar coordinate-based grids can be found for egocentric robot motion planning in literature (Borenstein and Koren, 1991; Lagoudakis and Maida, 1999). In these grids, the environment close to the robot has a high Cartesian resolution that decreases with the distance, due to the fixed angular resolution. Polar grids with hyperbolic distance functions are used to represent infinite radii within a finite number of grid cells. This property is used to plan long-distance paths in outdoor environments (Sermanet et al., 2008).

Longega et al. (2003) extend polar grids to log-polar grids by using a logarithmic discretization for the grid radii. Like the local multiresolution grid, this approach emphasizes a more precise path planning in the vicinity of the robot. Furthermore, polar grids have the advantage of easily allowing the integration of obstacles that are perceived by sensors with an angular characteristic, such as ultrasonic sensors and cameras.

3.2 ROBOT PLATFORM

In the RoboCup Standard Platform League, Nao robots from Aldebaran Robotics are used (RoboCup Technical Commitee, 2010). The Nao V3+ edition, used in the 2010 and 2011 RoboCup competitions, where our team NimbRo participated, has 21 degrees of freedom. The environment is perceived by two cameras—from which only one can

be used at a time—and two ultrasonic sensors. In our system, we continuously switch between the two cameras, which results in a frame rate of approximately 14 Hz. The ultrasonic sensors are located at the robot chest, covering an angle of approximately 110° in front of the robot. Their detection range is 300–700 mm[1].

The ultrasonic sensors measure the distance towards an obstacle, but not its precise angular position. In contrast, both cameras provide the exact direction towards an obstacle, but no precise distance. This leads to uncertainties which have to be taken into account.

The Nao V3+ robot is equipped with an x86 AMD Geode LX 800 CPU running at 500 MHz. It has 256 MB of RAM and 2 GB of persistent flash memory[2]. The built-in processor has the advantage of low power consumption, with the tradeoff of low computational power. Compared to state-of-the-art computer systems, the resources are rather limited. Hence, a low system load is an important requirement for the development of new software components.

Our software is based on the framework of the German SPL team B-Human (Röfer et al., 2010). The framework consists of several modules executing different tasks, e.g., perception of ball, lines, robots, locomotion, and behavior control. In addition to these modules the 3D simulator SimRobot is part of the framework. For our tests, we extended the framework by a new path planning module.

3.3 PATH PLANNING REPRESENTATIONS

At RoboCup 2010, we used reactive target selection and obstacle avoidance behaviors. Thus, a gait target vector (v_x, v_y, v_θ), which determines the walking speed in the forward, lateral, and rotational directions, respectively, is determined merely by direct perceptions and active behaviors. Typical behaviors that influence the direction of the gait target vector include approaching the ball and avoiding obstacles. The approach behavior generates a direct gait target vector moving the robot to a pose behind the ball facing towards the opponent's goal. Obstacles induce repulsive forces affecting the direction and magnitude of the gait target vector. This lets the robot bypass an obstacle or leads to a stop in front of it.

Our approach to planning a path to a target introduces a planning layer between the abstract behaviors and motion control. Waypoints provided by this layer are used to determine the target velocities for the locomotion layer. The abstract behaviors configure the planner, and perceptions are integrated into the world representation of the planner.

In our implementation, we employ A* search with closed list, introduced in Section 2.4. We implemented our planner with three differ-

1 Nao User's Guide Ver 1.6.0, Aldebaran Robotics
2 Nao Academics Datasheet, Aldebaran Robotics

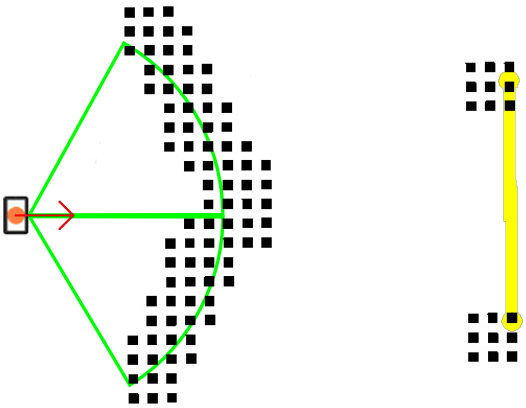
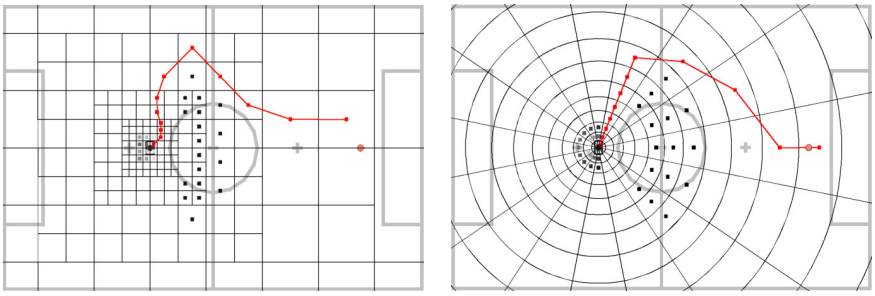


Figure 3.2: Obstacle model for soccer robots. The black squares depict occupied cells in a uniform obstacle grid. Left: Ultrasonic measurements (green) are represented as arcs discretized to the map resolution. Right: Visually perceived obstacles (e.g., goal posts, yellow) are discretized circles of known radius.

ent types of representations: a uniform, a local multiresolution, and a log-polar grid. All representations have in common that their coordinate system is fully egocentric, i.e., they are translated and rotated according to the robot pose.

Obstacle Representation

Obstacles are represented by their core radius and obstacle costs, as depicted in Figure 2.3 in Section 2.3. For visually perceived obstacles, we use the known radius of goalposts and other robots as core radii. Ultrasonic measurements are treated as obstacle arcs enlarged by measurement uncertainty and then discretized into single small occupancies. Figure 3.2 shows these obstacle representations. In addition to the static core radius of the obstacles, we add a component that increases linearly with the distance to the grid center. This component represents the uncertainty in distant measurements and the obstacle movements. To keep the cost integral of an obstacle constant, the maximum costs are reduced accordingly. Furthermore, obstacles are enlarged by the approximate robot radius to allow for planning with a point-like robot. Finally, a safety margin with linearly decreasing costs is added to the obstacle, before it is inserted into the map. This allows to plan between robots standing close to each other, but avoids the vicinity of obstacles if this is possible without larger detours.



(a) Local multiresolution grid (b) Log-polar grid

Figure 3.3: Multiresolution grid representations. Non-uniform grids allow a given area to be covered with multiple orders of magnitude less cells than uniform ones. Depicted are planned paths on a soccer field (red dots and lines) starting from the robot position in the center of the grid to the ball 3 m in front of it. The robot is facing the positive x-axis. Occupied cells are marked by black dots. The local multiresolution grid covers the area with five levels and 8×8 cells on each level, and the log-polar grid with 16 discrete steps for angle and distance, respectively.

Uniform Grid

As a baseline we employ a simple uniform robot-centric 2D grid map as described in Section 2.1. Obstacle positions, i.e., goalposts and other robots perceived by the vision system, and filtered distance measurements from the ultrasonic sensors, are maintained in external data structures and inserted into the map before each planning step. This avoids costly moving and rotating of cell occupancy probabilities while the robot is moving.

To allow for planning close to the ball and opponent players, a fine resolution of the planning grid is required. Especially in environments with open space characteristics, like a soccer field, more efficient representations are possible.

Local Multiresolution Grid

The local multiresolution grid—described in Section 2.3—allows for exact planning if the robot is close to the ball and takes into account the uncertainty of local sensing and the own and opponent's movements, implicitly. Figure 3.3a shows an 8×8 local multiresolution grid with a minimum cell size of 100 mm—the approximate footprint of the Nao robot—that covers an area of 163.84 m^2 with 256 cells using five grid levels. By contrast, a uniform grid covering the same area consists of 16 384 cells. Incorporating obstacles into the grid is analogous to the case of a uniform grid.

For path planning with the A* search algorithm, the costs of each step are calculated by means of the Euclidean distance of the centers

of both cells and the added costs of the target cell. The employed heuristic is the distance from the current grid cell to the target.

The advantages of this representation are the low requirements on the robot memory and on computational power. Moreover, uncertainties occurring in dynamic environments are considered by the increasing cell size with increasing distance from the robot.

Log-Polar Grid

Although our soccer robots are able to walk omnidirectionally, the highest speed can be achieved in the forwards direction. Furthermore, the robot sensors are designed for perceiving the environment in front of the robot. Accordingly, walking in the forwards direction towards the target, i.e., to the next waypoint of the plan, is preferred for long distances. On a soccer field with only smaller obstacles, it is likely that relatively long straight segments are often part of the plans. As our grid representations are egocentric, a target in front of the robot is on the positive x-axis. As this axis is a cell boundary in every resolution, this case is not well supported by the local multiresolution grid. Due to imprecise measurements of the target and inaccurate motion execution, distant targets in a uniform grid presumably change their respective cells, too.

This leads us to a grid representation that fits the motion and sensing characteristics of the robot in a more efficient way. In contrast to the Cartesian grid representations, a polar grid representation provides a straight path towards targets in front of the robot if there are no obstacles. In addition, sensor measurements relevant to path planning are best represented in polar coordinates on our robots. Ultrasonic measurements are represented as a distance and an apex angle, and visual perceptions are estimated distances and directions to obstacles. Both can be easily incorporated in a grid representation in polar coordinates.

We represent the environment in polar coordinates with an angle θ and a distance ρ with regard to the robot pose, written as tuple (θ, ρ). The robot faces in the direction of the positive x-axis. We discretize the angular component θ into T equally sized sectors. The first sector is defined such that the positive x-axis becomes the angle bisector of it. Hence, small angular inaccuracies in the perception or gait execution will not change the grid cell of a waypoint or the target that is on the x-axis straight ahead.

To reach the implicit consideration of uncertainties and computational advantages of the local multiresolution grid, the cells of our polar grid grow exponentially with the distance. In order to achieve this, the logarithm of the distance to the robot is partitioned. To define a minimal cell size and still achieve a reasonable growth until the maximum radius, we use a slightly shifted logarithm to avoid the ini-

tial strong slope of the logarithm. The calculations to determine the polar coordinates (θ, ρ) of a point (x, y) in Cartesian coordinates, and the corresponding discretized grid cell (r, t) are

$$\begin{aligned}
\rho &= \log_b \left(\left(\frac{(b-1)\sqrt{x^2+y^2}}{l} \right) + 1 \right), \\
\theta &= \arctan\left(\frac{y}{x}\right), \\
(r, t) &= \left(\lfloor \rho \rfloor, \left\lfloor \left(\frac{T}{2\pi} \right) \theta + 0.5 \right\rfloor \right),
\end{aligned}$$

where b is the base of the logarithm, l is the minimal cell size and T is the number of angular partitions.

Hence, the inverse operation is described by

$$(x, y) = \left(\frac{l(b^{r+0.5}-1)}{b-1} \right) \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}.$$

In our implementation, we use a base b of 1.1789 and a minimal cell size l of 100 mm. This base has been selected such that we reach a maximum radius of 7211 mm with 16 cell rings, which is sufficient to plan paths for any two points within the SPL field boundaries. We use 16 steps for the angular component as well, leading to 256 cells in total, the same number of grid cells used in the local multiresolution grid. The resulting grid is depicted in Figure 3.3b. Obstacles are maintained in the same external representations used for the other grid representations.

The costs of each step are calculated by means of the Euclidean distance of the centers of both cells, as in the local multiresolution grid. We calculate the heuristic likewise. The cell distances are precalculated to decrease the computational complexity of the logarithmic and trigonometric operations. Consequently, single node expansions of a planner are not more costly in this representation than in uniform grids.

Implementation Details

In our 2D planning implementation, we neglect the orientation and velocities of the robot for efficiency reasons. Consequently, due to the frequent replanning, sudden changes of the gait target vector are possible. To avoid this, we introduce a virtual obstacle behind the gait target vector, which represents its starting speed, depicted in Figure 3.4, following ideas from Behnke (2004). The polar grid representation employs a semicircle with cost interpolation between a minimum at the edges and a maximum at the midway of the circle segment. In contrast, the Cartesian grid representations use a rectangle having

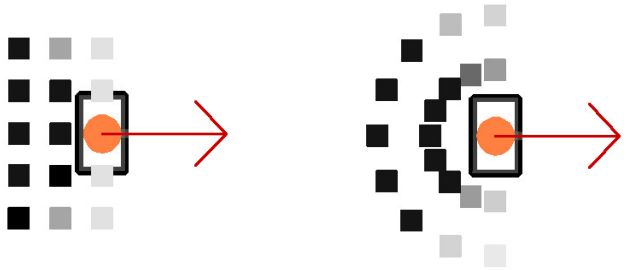


Figure 3.4: An artificial obstacle that leads to a preference of paths towards the current walking direction to avoid sudden directional changes. Cells with higher costs are drawn darker. The red arrow depicts the gait target vector. Left: Cartesian grid. Right: polar grid.

the same characteristic. When the robot is moving, this obstacle is opposed to the gait target vector with costs corresponding to the scalar magnitude of the velocity.

To generate motion commands for a planned path, the planning module sends the next waypoint on the path to the motion control in every execution cycle. Replanning is performed if the robot movement along the path exceeds a threshold. Between consecutive planning calls, the plan is adjusted using odometry data. The resulting gait target vector is the weighted vector of position and angle of the waypoint relative to the robot. If the angle between robot and waypoint is larger than 30° the robot only rotates towards the waypoint. If the angle is below 20° the robot simultaneously rotates towards the waypoint and walks omnidirectionally into its direction. In the hysteresis interval 20–30° the robot stays in the previous state.

3.4 DYNAMIC PLANNING WITH INTENTION PROJECTION

Most of the obstacles apparent on the soccer field are not static. As described before, we take this into account by smoothing distant obstacles. If the movement of the other robots can be estimated in advance, it is possible to avoid future positions of these. Naturally, it is not possible to reason about the exact behaviors and targets of the opponent team. Thus, we have to apply situation specific assumptions for probable movements of the opponent. For example, if our robot intends to reach a position where it can get ball possession, this intent is projected to the opponent. Thus, we assume it will move to a position where it can control the ball, too. This assumption is reasonable, as opponents with the same intent as the planning player will more likely intersect the planned path than robots with different intents.

Our dynamic obstacle model extends the static obstacle model by introducing an assumed target relative to the obstacle and a velocity vector v. To be consistent with the multiresolution representation of

space, the obstacle radius is increased and the costs are decreased with increasing distance to the robot. We assume that the opponent may either keep standing still or move to the target with any speed up to a maximum v_{max}. This results in a uniform probability distribution of the opponent's position along a vector $v_{max}t$. We model this by growing the obstacle representation into the direction of its movement up to the assumed target with advancing time t. The overall costs of the obstacle are kept constant. Hence, the costs per grid cell of the obstacle decrease.

To represent the resulting different shapes of obstacles at different times, we extend the grid representation to the time dimension. The resulting three-dimensional grid is discrete in time and this discretization may not necessarily equal the space discretization for arbitrary robot speeds. To take this into account, we explicitly calculate the average time the robot needs to follow the planned path up to the current grid cell and discretize it afterwards to determine whether edges connecting nodes within the same time level can be followed.

Figure 3.5 shows an example where the planned paths with and without dynamic planning are qualitatively different. Our robot aims to reach a position next to the ball to dribble it towards the opponent goal. Therefore, it has to walk around the ball. Another player is approaching the ball from the left bottom side of the map. Utilizing a planning algorithm not taking the directed movement of this robot into account, the robot would try to pass the ball at the side the opponent is coming from, possibly having to avoid it later on. The dynamic approach plans the same shortest path, if the opponent is still far away. Otherwise, the ball is passed on the other side, where an intersection with the opponent's path is unlikely.

Obviously, adding the time dimension makes the planning computationally more involved. For the uniform time discretization we use a 16 *s* lookahead, discretized into 1 *s* steps. This results in a local-multiresolution grid with 4096 cells. Thus, we reduce the complexity, following the same ideas of a multiresolution representation as in the space dimensions. The minimal time an opponent robot needs to reach a cell corresponds to the distance of that robot to the cell. Here, we have three qualitatively different cases to consider:

- The robots are close to each other and to the cell. Thus, the cell is represented at a fine spatial resolution and an intersection of the robot paths may occur soon. A finer temporal resolution is necessary.

- The cell is further away. Hence, it is represented at a coarse spatial resolution and coarse temporal resolution is sufficient, regardless of the position of the opponent.

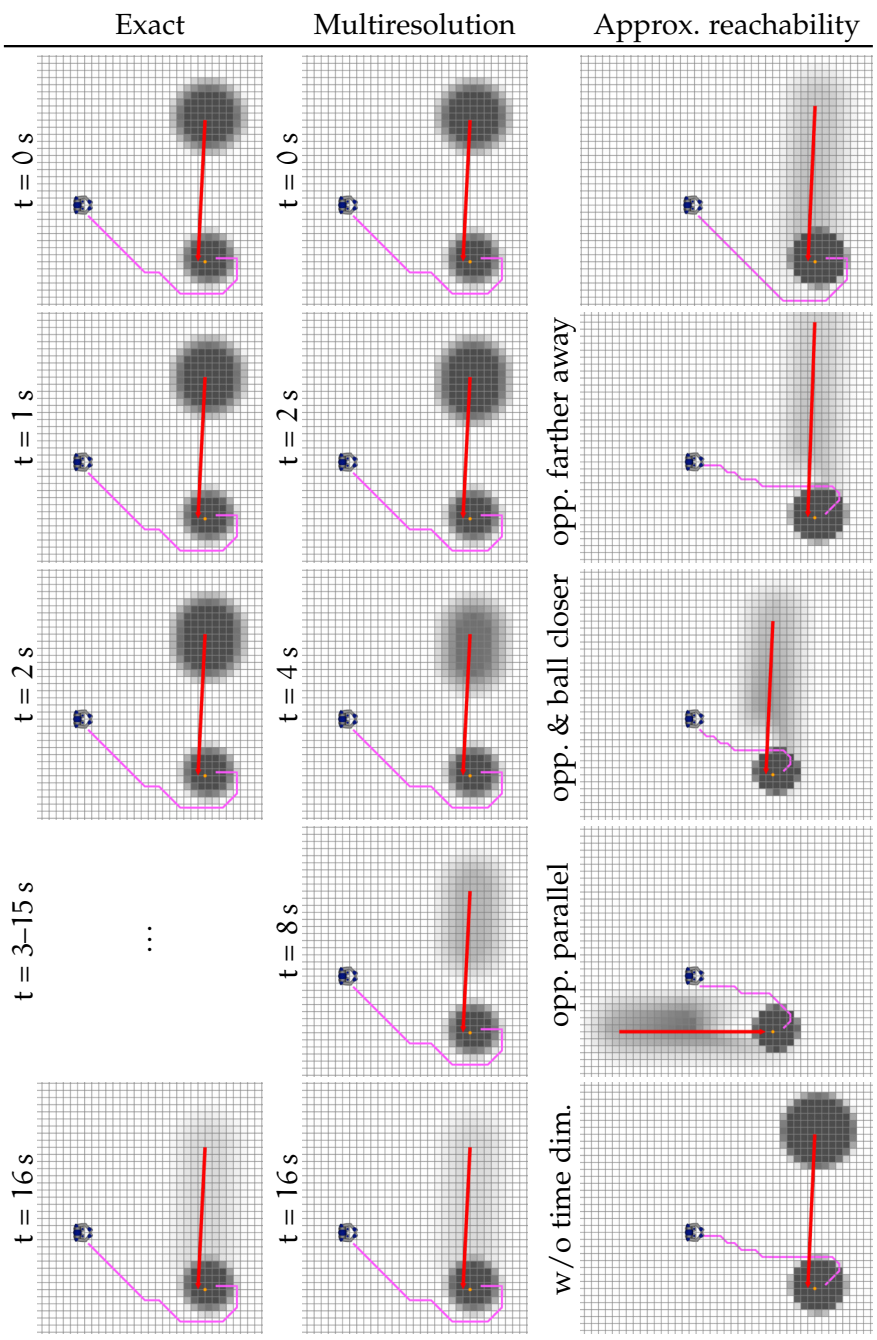


Figure 3.5: The planning robot and an opponent aim to reach opposing positions behind the ball towards their respective goals. The opponent's position is propagated over time by elongating the corresponding obstacle shape until it reaches the target. The time increases from the top (t = 0) to the bottom images (t = 16 s). The left image series shows the uniform time discretization and the middle series the multiresolution time discretization. The first four figures in the right column depict the implicit consideration of time for different configurations of ball and opponent. The bottom-right figure shows the baseline without the time dimension.

- The cell is close to the planning robot, but the opponent is further away. In this case a coarse resolution is sufficient to determine, if an intersection of paths in this cell is likely.

Figure 3.5 (center) depicts snapshots of the ball approach example using a multiresolution time representation. The discrete time steps t_m are defined as

$$t_m(i) = \begin{cases} 0 & \text{if } i = 0 \\ 2^i & \text{if } i > 0 \end{cases}.$$

The resulting grid contains 1280 cells and covers the same time and space as the uniform grid.

As the speed of the planning robot is bound, the time necessary to reach cells increases linearly with distance to the robot, defining a lower envelope of the reachable cells. Thus, major parts of the three-dimensional grid cannot be reached. Especially, the fine-grained temporal resolution steps are mostly below this envelope. For the planner it is of primary importance if a path through a cell may possibly cause a collision in the future. Shortest paths often start with long straight segments towards the target. In the case of an apparent obstacle, the segment is split into two segments with an intermediate point that modifies the path to be collision-free. The time to follow the first segment can be approximated by a linear function of the distance between robot and endpoint of that segment. This leads us to an implicit incorporation of the time dimension into our obstacle model. We determine the minimal distance to the line segment between the opponent and its assumed target. With the linear time approximation function, we get the approximate time of the probable intersection of the two paths. This approximation is used to estimate the distribution over the possible positions of the obstacle on the field. As time is only implicitly taken into account for planning, the planning problem stays two-dimensional.

3.5 EVALUATION

We evaluate the efficiency of our planning representations and the intention projection quantitatively and qualitatively in simulation and on the robot.

Planning Representations

In the first experiment, we measure the planning time required with the three planning representations in the simulator and on a Nao robot with regard to four different test cases. These test cases represent all possible qualitative constellations found on the soccer field. The four cases are

Table 3.1: Path planning time with different grid representations on the Nao.

Test case	Planning time for grid type (in ms)		
	uniform	local multiresolution	log-polar
no obstacles	2.9	0.4	0.3
ultrasonic	12.2	0.8	1.2
camera	18	2.3	3.0
ultrasonic & camera	23	3.0	3.5

I) no obstacle is detected,

II) obstacles are detected either in the ultrasonic sensor measurements or

III) through cameras, and

IV) obstacles are measured by both, ultrasonic sensors and cameras.

The used sensors influence the planning time, because sonar sensors can only perceive close obstacles and have a low angular resolution. This results in only few, but large obstacles in the vicinity of the robot. In every test case, the target is 4000 mm in front of the robot and five other robots are positioned on the field. In order to avoid an influence of noisy sensor data and for a better reproducibility, we set the egocentric obstacle positions manually when testing on the real robot and use ground truth data in simulation. The measured execution times are averaged over 1000 planner runs.

The results of the tests on the Nao robot are shown in Table 3.1. Overall, the multiresolution approaches clearly outperform the uniform grid planning representation. Furthermore, there are differences between the local multiresolution and the log-polar grid, as the first has a lower computational complexity.

Planning with Intention Projection

We compare the planning time on the Nao robot for the ball approach example. To plan a path around the ball without considering the opponent's movement takes on average 1.6 ms, and with 16 uniform time discretization steps 55.4 ms. The multiresolution discretization with 5 steps takes 7.9 ms and the implicit approach takes 3.3 ms. With every representation of the time component it is possible to use the shortest path if the opponent is sufficiently far away and to avoid the opponent's path if it is closer. Figure 3.6 shows resulting trajectories for all three representations.

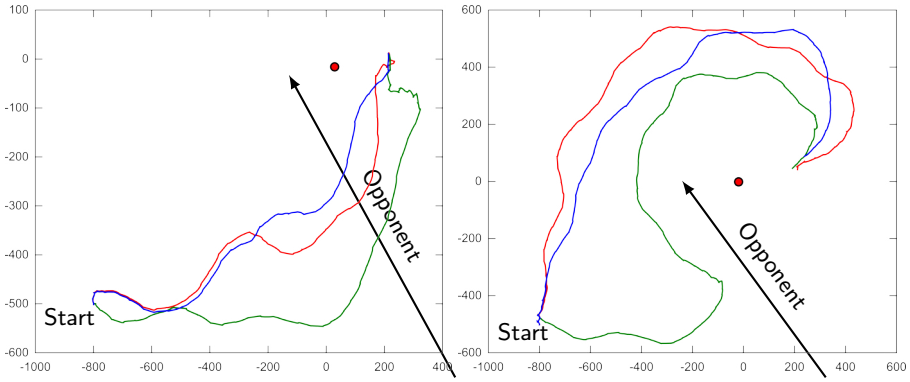


Figure 3.6: Comparison of the robot trajectories while using the planner without time consideration (green), with uniform time discretization (blue), and with implicit time incorporation (red). The target is a point behind the ball (circle). An opponent is approaching from the bottom right side. In the example on the left side the opponent is farther away than in the example depicted on the right side. Both planners that propagate the opponent's position plan qualitatively similar path. While following the plan without representation of the time, the robot had to replan to avoid a collision with the opponent in the case depicted in the right figure. Field coordinates are in mm.

3.6 CONCLUSION

In this chapter, we evaluated two approaches to path planning which are applicable to soccer robots with relatively low computational resources. Both approaches make use of properties found in the soccer domain. An important property is the lack of static obstacles in the environment of the soccer field. For this reason, it allows planning at a coarse resolution for regions which are far from the robot. Furthermore, one can expect to find a valid plan refinement in order to avoid dynamic obstacles while approaching them. Consequently, our grid representations employ a decreasing resolution for distant parts of the environment.

As virtually all obstacles are dynamic, it is likely that the situation in distant cells will have changed at the time a plan refinement will be necessary. Therefore, we are convinced that approximate planning with continuous and fast replanning is superior to slower exact planning. Furthermore, the estimation of possible future obstacle positions is likely to improve plans with regard to the need of necessary replanning. Thus, we use assumptions about the behavior of soccer playing robots to estimate their trajectories. The real-robot experiments reveal that the speedup facilitates real-time planning on the Nao.

4

GRASP AND TRAJECTORY PLANNING FOR MOBILE BIN-PICKING

Object manipulation is a key capability in many industrial and service robotics applications. Removing individual objects from an unordered pile of parts in a carrier or box—bin picking—is one of the classical problems of robotics research. It has been investigated by numerous research groups in the last decades (Ikeuchi et al., 1983; Liu et al., 2012; Rahardja and Kosaka, 1996).

In general, grasp planning addresses the problem of selecting feasible grasps given the specifications of the object to grasp, the robot kinematics, and the surrounding scene. A variety of approaches determine contact points for each finger for precision fingertip grasps (Borst et al., 1999; Heester et al., 1999). In contrast, power grasps that exploit contacts with finger surfaces and the palm may yield much more stable grasps. To generate a rich set of fingertip and power grasps, pre-grasp shape approaches have been proposed (Miller et al., 2003; Xue et al., 2009).

Typical bin picking solutions consist of a 3D sensor mounted above the box, a compute unit to detect the objects, estimate their pose and plan grasping motions, and an industrial robot arm that is equipped with a gripper. In order to extend the workspace of the robot and to make bin picking available for environments that are designed for humans, we implement bin picking using an autonomous anthropomorphic mobile robot. Due to the inaccuracies in perception and navigation of the mobile base to the bin, this poses an additional challenge of varying positions of the manipulator relative to the box. Stückler et al. (2012) developed an approach to grasping objects from a table top that is based on sampling feasible grasps on segmented point clouds and fast, but conservative, collision check heuristics for our service robot.

Here, our scenario is motivated by industrial applications. We consider the task of grasping objects of known geometry from an unordered pile of objects in a transport box, depicted in Figure 4.1. The grasped object is to be transported to a processing station where it is placed. Due to the operability of the robot in environments designed for humans, this scenario is easily transferable to a household scenario, e.g., clearing a shopping box.

Solving this mobile manipulation task requires the integration of techniques from mobile robotics, like localization and path planning, and manipulation, like object perception and grasp planning. In this chapter, we focus on the grasp and retrieval planning.

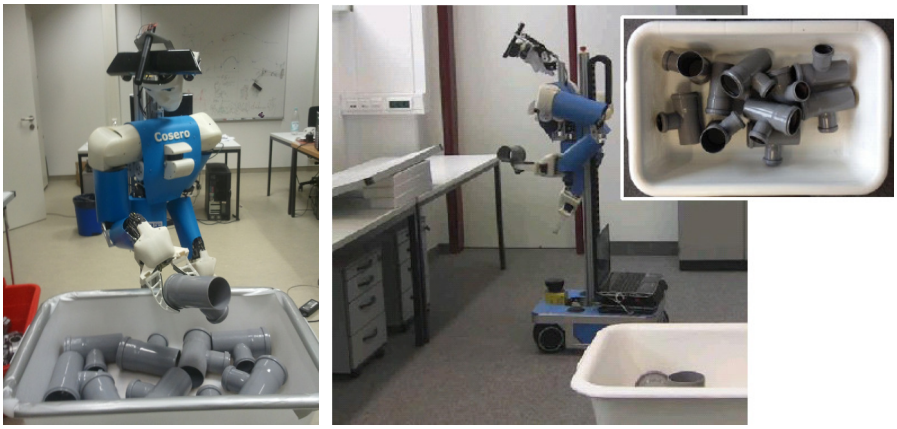


Figure 4.1: Mobile bin picking scenario. Cosero grasps objects from a transport box. It navigates to a processing station and places the objects there.

We present an efficient pre-grasp shape approach to plan grasps for objects that are composed of shape primitives like cylinders and spheres. These shape primitives are used for both object recognition and grasp planning in an integrated and efficient way. With the proposed approach, our service robot can grasp object compounds from piles of objects. For the perceived objects, we plan feasible grasping motions for a 7-DoF (degrees of freedom) manipulator. In order to gain efficiency, we plan grasps in a multi-stage process in which we successively prune infeasible grasps in tests with ascending complexity. We plan reaching motions with an efficient multi-resolution sampling-based method. We integrated all components to perform the complete task and evaluated the performance of the integrated system.

4.1 RELATED WORK

Despite its long history, static bin picking is still an active research area. One implementation of Papazov et al. (2012) utilizes a Microsoft Kinect sensor mounted above a table to acquire depth images of the scene. Object models are matched to the measured point cloud by means of a normal-based RANSAC (Fischler and Bolles, 1981) procedure. Papazov et al. consider table-top scenes where multiple objects are arranged nearby, including the stacking of some objects. They select the object to be grasped based on the center-of-mass height (high objects are preferred). Each object is associated with a list of predetermined grasps, which are selected according to the orientation of the gripper (grasps from above are preferred) and checked for collisions. The grasping is performed by a compliant lightweight robot arm with parallel gripper. Bley et al. (2006) propose another approach of grasp

selection by fitting learned generic object models to point cloud data. In contrast to our approach they manipulate separated objects.

E. Klingbeil et al. (2011), utilized a Willow Garage PR2 robot to grasp unknown objects from a pile on a table and read their bar-codes to demonstrate a cashier checkout application. Because the dense packing of objects in a pile poses considerable challenges for perception and grasping, Chang et al. (2012) proposed pushing strategies for the interactive singulation of objects. Gupta and Sukhatme (2012) estimate how cluttered an area is and employ motion primitives to separate LEGO bricks on a pile.

Manipulation in restricted spaces like boxes and shelves leads to difficult high-dimensional motion planning problems. To this end, Cohen et al. (2011) proposed a search-based motion planning algorithm that combines a set of adaptive motion primitives with motions generated by two analytical solvers.

All the above approaches are demonstrated with a stationary robot. By contrast, Chitta et al. (2012) proposed an approach to mobile pick-and-place tasks, which integrates 3D perception of the scene with grasp and motion planning. The approach has been used for applications like table-top object manipulation, fetching of beverages, and the transport of objects. In these applications, objects stand well-separated on horizontal surfaces or are ordered in feeders.

The Armar robots (Vahrenkamp et al., 2012) demonstrated tasks in a kitchen scenario that require integrated grasp and motion planning. Beetz et al. (2011) let a PR2 and the robot Rosie cooperatively prepare pancakes, which involves mobile manipulation and the use of a tool.

Common to most of these mobile manipulation demonstrations is that the handled objects are well-separated. Since the work described in this chapter was first presented—the system described here was to the best of our knowledge one of the first mobile bin-picking system—new approaches to mobile bin-picking and bin-picking in general have been developed. A direct successor of our system is a mobile manipulator for an automated kitting task (Holz et al., 2015). The system consists of a FANUC arm mounted on a mobile base. Schwarz et al. (2017) developed a fast, stationary system for object retrieval and storage in an automated warehouse. Another stationary bin picking system for the Amazon Picking Challenge was developed by Hernandez et al. (2017). Both systems employ an overhead camera system to perceive objects in a tote at a fixed position and a 6-DoF industrial manipulator arm.

Domae et al. (2014) find feasible grasps by calculating graspability maps from depth images. The method allows fast grasp selection, but considers a simplified problem where all grasps are top-grasps. We follow a more flexible approach to full 6-DoF grasp planning.

Another approach inspired by the Amazon robotics challenge is affordance-based grasp evaluation (Zeng et al., 2018). They grasp any

reachable object in a bin by applying one of four grasping primitives depending on the perceived object shape and recognize it after the grasp.

An example for learning grasps of objects in clutter is (Laskey et al., 2016). A manipulator learns to push unwanted objects out of its way to reach a target object. The employed gripper is restricted to 2-DoF—rotation around one joint and linear extension of the arm.

4.2 SYSTEM OVERVIEW

We consider a scenario where unordered parts need to be grasped from a transport box, as shown in Figure 4.1. The objects are transported to a processing station and placed there.

For the experiments, we use the cognitive service robot Cosero (Stückler et al., 2011), which navigates on an eight-wheeled omnidirectional base and has an anthropomorphic upper body with two 7-DoF arms that end in grippers with two Festo FinGripper fingers. Due to the Fin Ray effect, the finger tips passively bend inwards, creating a closure around a grasped object. To extend the workspace, the upper body of the robot can be twisted around the vertical axis and lifted to different heights. With only 32 kg, Cosero has a low weight, compared to other service robots. Nevertheless, its arms can lift a payload of maximum 1.5 kg each. The robot senses its environment in 3D with a Microsoft Kinect RGB-D camera in the pan-tilt head. For obstacle avoidance and tracking in farther ranges and larger field of views, it is equipped with multiple laser rangefinders (LRFs), of which one in the chest can be pitched and one in the hip can be rolled. The main computer is a quadcore notebook with an Intel Core i7-Q720 processor.

We divide the mobile bin picking task into the cognition phase where the robot explores the transport box and recognizes the top-most objects, the pick-up phase where the robot grasps a top-most object out of the transport box, and the place phase where the robot places the object on the processing station.

The autonomous robot behavior is generated in a modular control architecture, using the interprocess communication infrastructure of the Robot Operating System (ROS) (Quigley et al., 2009). We implemented the mobile bin picking task as a finite-state machine. It monitors the state of task fulfillment and triggers individual behaviors in the appropriate order. The task starts with the robot navigating to the transport box. When the robot is in front of the transport box, it switches to a local navigation mode that accurately aligns it to the box. The next step is the acquisition of a 3D point cloud of the entire transport box, which is then processed by the object recognition module. The detected objects are fed to the grasp planner, which selects an appropriate grasp and plans trajectories for approaching the

object and for removing it from the box. After the planned grasping motion is executed, Cosero navigates to the processing station using the environment map and local alignment with the processing station. Finally, our robot releases the object into the processing station. This process continues until the transport box is empty.

To cope with the challenges of mobile bin picking, we use a global-to-local strategy for approaching the transport box and fetching the work piece. We use a global navigation approach based on Adaptive Monte Carlo localization (AMCL) (Fox et al., 1999) and A*-based path planning (see Section 2.4) in a given occupancy grid map using an LRF to roughly approach a pose in the map. A local navigation approach accurately aligns the robot with the transport box and the processing station.

4.3 SHAPE PRIMITIVE DETECTION AND OBJECT RECOGNITION

To grasp objects from an unordered pile in a transport box it is necessary to recognize these objects robustly in point cloud data. In our system, we recognize the objects and determine their pose by employing the shape primitive-based object recognition from Berner et al. (2013). Shape primitives, e.g., cylinders, spheres, and planes, are searched by a RANSAC-based approach in the point cloud. Objects are modeled as a graph with nodes representing shape primitives and edges representing their relative poses. We use a similar representation in our grasp planner to determine generally feasible grasps, depicted in Figure 4.2 The sensor transmits measurements of the scene as 3D point cloud messages to the object perception module. After the detection and recognition of objects in the scene, the object perception module provides a list of detected objects and their 3D poses to grasp and motion planning. Besides the parameterized object model, the 3D points filtered from the points on the detected objects are transmitted for the collision map. The points and object representations are stored in a multiresolution height map, detailed in the next sections.

4.4 GRASPING OF SHAPE PRIMITIVE COMPOUNDS

We investigate efficient solutions to grasp and motion planning in order to achieve fast planning and short delays between object perception and motion execution. In our approach, we first determine feasible, collision-free grasps at the object. We rank these grasps and find the best grasp that is achievable by a collision-free reaching motion.

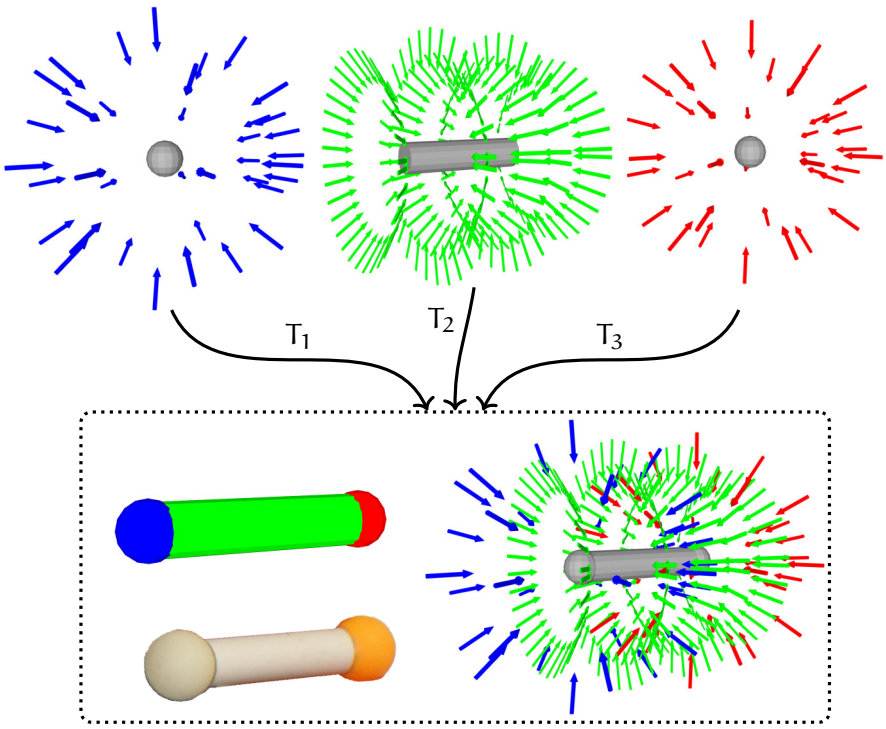


Figure 4.2: Primitive compound object. We model objects for grasp planning as a set of geometric primitives with possible grasps and corresponding transformation matrices T. Top: Sphere and cylinder primitives, possible grasps are depicted by arrows. Bottom: Compound object (model and photo) and all possible grasps before further filtering.

Grasp Planning

For grasp planning, we find feasible, collision-free grasp postures on the object to grasp. We define a grasp as a tuple containing

- the final pose of the gripper when grasping the object (the grasp pose),
- the closure of the gripper (according to object width),
- an initial pose of the gripper (the pre-grasp pose) for approaching the grasp pose, i.e., in a distance of 10 cm to the object,
- a score encoding preferences for certain grasps, e.g., grasping cylinders instead of spheres, and preferring, as the grasp pose, the central part of the cylinder.

We plan grasps in an efficient multistage process that successively prunes infeasible grasps in tests with increasing complexity: In the first stages, we find collision-free grasp poses on the object, irrespective of the pose of the object and not considering its scene context.

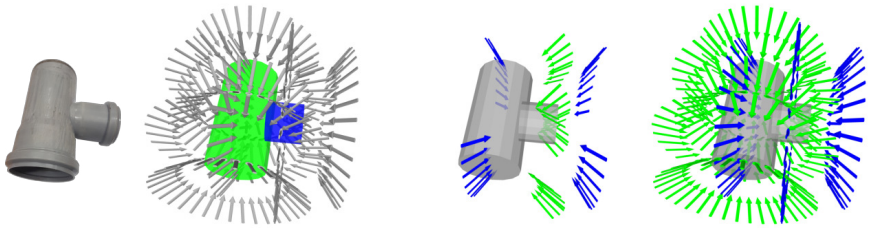


Figure 4.3: Analytical offline grasp pruning. In a first step, we prune grasps where the line of sight between the grasp pose and the projection to the corresponding shape primitive is obstructed by other primitives of the compound object. From left to right: Photo of the object, union of all possible grasps of the individual primitives, pruned grasps, and remaining grasps of the object compound. The color of the arrows depict the corresponding shape primitive.

These poses can be precalculated efficiently in an offline planning phase. We sample grasp poses on the shape primitives. From these poses, we extract grasps that are collision-free from pre-grasp pose to grasp pose according to fast collision-check heuristics.

Sampling of Grasps

We sample equidistant grasp poses at the surface of an object model. Since the objects are described by compositions of parametric shape primitives, we sample grasps efficiently according to the primitives (see Figure 4.2 for an example). The pre-grasp pose is then selected to lie along the local surface normal at the sampled grasp position and in a distance corresponding to the length of the gripper. Accordingly, the orientation of the grasp pose is restricted to point along the normal. From one sampled grasp position, we then generate multiple grasp poses by sampling different rotation angles around the normal. By collecting the grasp poses sampled on all primitives of an object model, we obtain an initial set of grasp candidates.

Constraining Grasps by In-Object Feasibility

We evaluate a local primitive-dependent score of the candidates and remove those grasps that are in collision within the object itself. We first remove all candidates where the line segment between the grasp pose and the pre-grasp pose intersects another primitive, depicted in Figure 4.3. For the remaining set of candidates, we simulate a grasp by transforming the gripper model to the grasp pose and checking collisions of the gripper shape with the object model (see Figure 4.4). By this, a considerable amount of infeasible grasps can already be rejected. For each remaining grasp, we compute a quality score. This local primitive-dependent score prefers stable grasps, e.g., at the center of mass of the shape primitives.

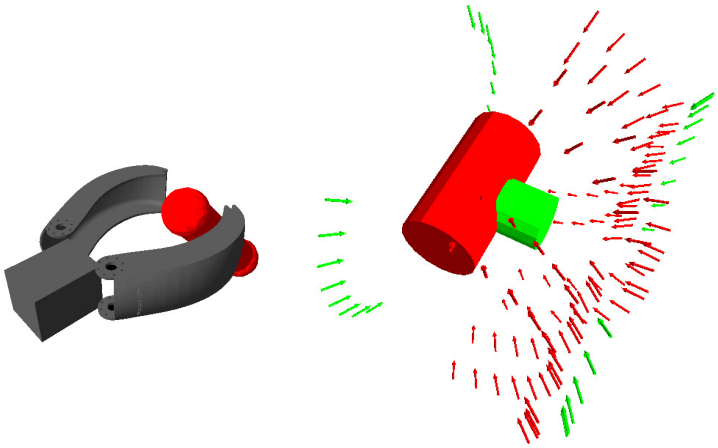


Figure 4.4: Offline collision checking with gripper model. In a second offline preprocessing step, we prune more generally infeasible grasps by collision checking with a gripper model and an object. Left: We check for collisions of the open gripper with the object at the grasp pose. By this, we take the actual gripper geometry into account in order to prune grasps that have been sampled on a specific shape primitive in the object but result in collisions of the gripper at further shapes. Right: Grasps pruned by collision checking. The color of grasp poses depict the corresponding shape primitive.

This set of grasps can be computed offline for a particular object compound, since this set is determined without taking the surrounding and the reachability in specific object poses into account. That is, for every object compound, we only have to determine (and evaluate) this set of grasps once.

Considering In-Environment Feasibility

During online planning, we examine the remaining grasp poses in the actual poses of the objects to find those grasps for which a collision-free solution of the inverse kinematics in the current situation exists. Grasps from below the object are considered as infeasible and filtered first. Next, we test grasp and pre-grasp positions against our height-map. The remaining grasps are ranked first by the object ranking—the quality score computed offline—and second by the grasp ranking.

The grasp ranking includes

- height of the grasp pose above the ground plane to prefer grasps at high positions,
- angle of the grasp w.r.t. the robot, preferring *top grasps* from above the object,
- and deviation of the arm posture from convenient joint angles.

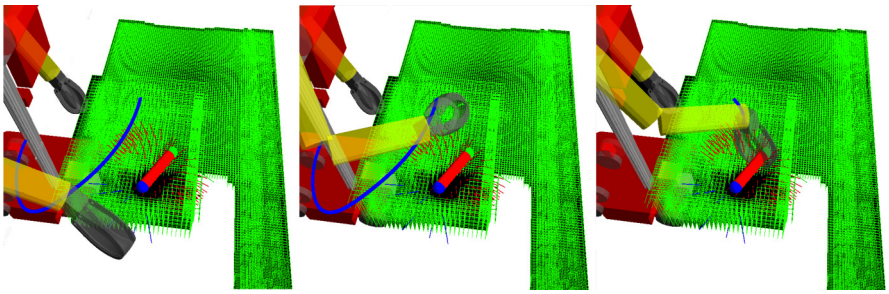


Figure 4.5: Planned endeffector trajectories for grasping a compound object from a transport box. The trajectories for the approach (left and middle) and grasp (right) phases are shown as blue lines.

Collision-free inverse kinematics solutions are searched for the grasps in descending order. We allow collisions of the fingers with other parts in the transport box in the final stage of the grasp, i.e., in the direct vicinity of the object to grasp. The shape of the fingers allows for pushing them into narrow gaps between objects. If a valid solution is found, we employ motion planning to find a trajectory. If none is found, we continue with the grasp evaluation.

Motion Planning

Our grasp planning module finds feasible, collision-free grasps at the object. The grasps are ranked according to a score which incorporates efficiency and stability criteria. The final step in our grasp and motion planning pipeline is now to identify the best-ranked grasp that is reachable from the current posture of the robot arm. We solve this by successively planning reaching motions for the found grasps (see Figure 4.5). We test the grasps in descending order of their score. For motion planning, we employ a bidirectional version of KPIECE (Şucan and Kavraki, 2008) with lazy evaluation (LBKPIECE).

We split the reaching motion into multiple segments:

- moving the endeffector over the transport box,
- reaching the pre-grasp pose,
- and finally grasp.

This allows for a quick evaluation if a valid reaching motion can be found by planning in the descending order of the probability that planning for a segment will fail.

Multiresolution Height Map

To speed up the process of evaluating collision-free grasp postures and planning trajectories, we employ a local multiresolution height map based on the 2D multiresolution grid described in Section 2.3.

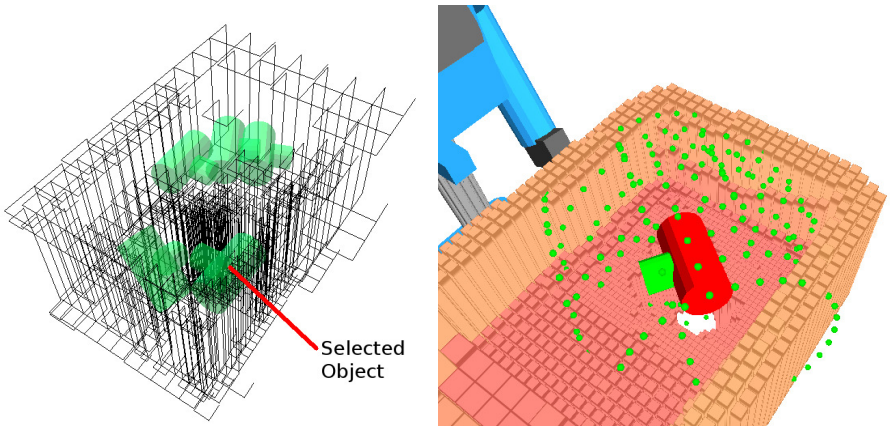


Figure 4.6: Local multiresolution height map. Left: The height map is object-centric, i.e., it has a high resolution close to the object the robot attempts to grasp. Right: We filter out grasp candidates that lie below the height value in the corresponding grid cells. A 3D representation of the height map is used for arm trajectory planning.

Our height map is aligned with the ground plane and each cell contains the maximum height of an object projected into the grid. We make the assumption that the objects in the transport box can only be accessed from above and that objects positioned below other objects cannot be directly picked. Consequently, we reduce the full 3D environment representation to this simpler 2.5D representation to simplify the collision checking process to a query if the height coordinate of a point is smaller than the value in the corresponding grid cell.

Planning in the vicinity of the object needs a more exact environment representation as planning farther away from it. This is accomplished by centering the collision map at the object. This approach also leads to implicitly larger safety margins with increasing distance to the object as the grid resolution decreases (see Figure 4.6).

Removal Planning

After the execution of the reaching motion, we check if the grasp was successful based on the gripper servo feedback. If the object is within the gripper, a removal motion is planned with the object model attached to the endeffector using the detected object pose. We allow minor collisions of the object and the endeffector with the collision map in a cylindrical volume above the grasp pose. Finally, the work piece is deposited at the processing station. To reach it, global navigation and local alignment are used in the same way as for the box approach.

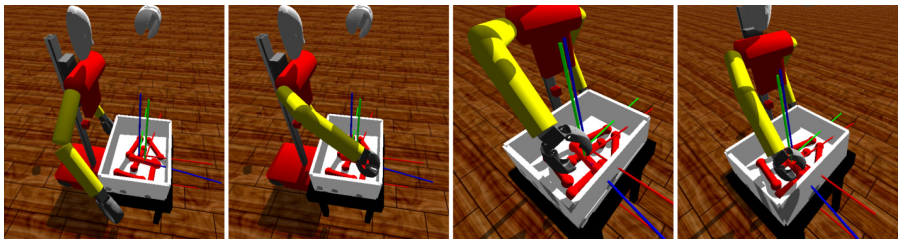


Figure 4.7: Grasp planning experiments in simulation. We employed the physics-based simulator Gazebo for testing and evaluating our approach before transferring it to the real robot.

Implementation Details

To speed up our grasp and motion planning pipeline, we implemented it in a way that directly after a feasible grasp is found planning for a reaching motion is performed. Only if no reaching motion can be found, further grasps are evaluated. We implemented the handling of different failure conditions during grasp execution to increase the robustness of our system. The planning and execution pipeline has been adapted to allow for choosing between both arms to execute a grasp. The preference for an endeffector is based on object position—objects in the left half of the transport box are preferably grasped with the right arm and vice versa. If a grasp fails, i.e., the reaching motion is aborted during execution, or no object is measured in the gripper after grasping, one more attempt of execution another grasp is performed. After two failed grasps, we switch the used arm and try to reach the objects with the other gripper. If that fails after another two attempts, we assume that our model of the transport box is no longer valid and rescanning is necessary.

4.5 EVALUATION

We evaluated our grasp and motion planning pipeline in simulation as well as in two scenarios—an industrial application and in the RoboCup@Home competition—with the real robot.

Simulation Experiments

The evaluation in a physical simulation of the robot allows for reproducible experiments. Our simulation environment is based on the Gazebo simulator (N. Koenig and Howard, 2004) and shown in Figure 4.7. We placed a single dumbbell-shaped object (see Figure 4.2) randomly in a transport box and measured the time for planning and the success rate. In ten runs, our approach requires on average 14.9 s to choose a non-colliding grasp that is within the workspace of the robot. We then plan reaching motions to choose a reachable

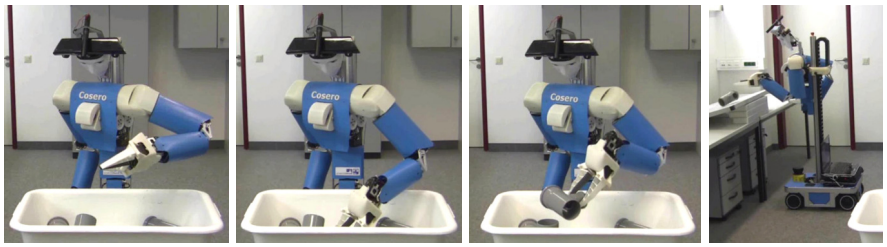


Figure 4.8: Example of a mobile bin picking and delivery run. From left to right: the robot grasps an object out of the transport box and puts it on the processing station.

grasp within 2.45 s on average. Note, that multiple reaching motions may have been evaluated. A successful reaching motion is planned in 0.45 s on average. In all test runs, the robot succeeded to grasp the object.

Lab Experiments

Our experiments in the lab were inspired by an industrial scenario. We let our robot Cosero clear a transport box filled with up to ten pipe connectors as depicted in Figure 4.1. We split the complete task into single runs where the robot picks up one pipe connector and delivers it. Figure 4.8 shows a series of images from a successful run. The complete execution of the grasp and delivery of the object is shown in Video 4.1[1]. In Table 4.1 we report timings, the number of detected objects per run, and the number of trials needed to successfully pick objects for clearing one complete transport box. On average our approach required 5 s to detect objects, 19.9 s to choose a grasp, and 3.8 s to plan a valid reaching motion. This includes one run, where the robot aborted the execution of a planned motion and replanned for the other arm (Run 5 in Table 4.1). In Run 2 in Table 4.1 noisy measurements of the used Kinect sensor lead to wrongly connected primitives in the object detection. This is not a large problem for our approach, as grasps are calculated on a per-primitive-base, but it slows down the grasp evaluation as an additional phantom object is detected and grasps for that object are evaluated.

In total, we have recorded 32 runs for this scenario, including the ten runs discussed in the previous paragraph. In 28 runs, the robot could successfully grasp a pipe connector and deliver it to the processing station. In nine of these successful runs, the robot first failed to grasp an object, detected its failure, and performed another grasp. This was the case, for instance, when the object slipped out of the gripper after grasping. In four runs, the object was not successfully delivered to the processing station. In three out of the four failed runs the last object could not be detected. In one instance, the object slipped out of the gripper after lifting. This is caused by the fact that

1 Video 4.1: www.nieuwenhuisen.de/thesis/bin-picking-industrial.mp4

Table 4.1: Timings for grasp and motion planning and number of found objects while clearing a transport box. In Run 5, reaching the initially chosen grasp pose was aborted and replanning was necessary, leading to higher evaluation and planning times. In Run 2 the detection result includes two false positives.

Run	Detections	Duration (in s)		Trials
		Selection	Planning	
1	4 / 10	4.0	1.2	1
2	10 / 9	16.6	2.9	1
3	8 / 8	33.7	3.0	1
4	5 / 7	28.1	6.8	1
5	4 / 6	48.2	9.3	3
6	4 / 5	9.9	4.9	1
7	3 / 4	30.2	3.0	1
8	3 / 3	10.0	1.2	1
9	2 / 2	9.0	3.7	1
10	1 / 1	9.5	1.6	1
Avg.	76%	19.9	3.8	1.2

Table 4.2: Time needed for phases of the bin picking demonstration. The values for the subphases *grasp selection* and *motion planning* are based on a subset of ten runs, the complete clearing of one box filled with ten objects.

Phase	Duration (in s)		
	Mean	Std. dev.	
Navigation (transport box)	20	8	
Approaching (transport box)	16	11	
Cognition phase	83	41	
- Grasp selection	19.9	14.4	Grasp
- Motion planning	3.8	2.6	
Grasping	36	7	
Navigation (processing station)	26	9	
Approaching (processing station)	22	9	
Putting the object on the processing station	18	2	

we have to allow collisions between the gripper and other objects during the grasp. Occasionally, these minor collisions can cause changes in the object pose that can make the chosen grasp impossible or un-

Figure 4.9: Public demonstration of mobile bin picking at RoboCup 2012.

stable. We report mean and standard deviation of the required time for the individual phases of the individual 32 runs in Table 4.2. The cognition phase includes object perception and detection as well as the grasp and motion planning. Please note, that the timings for the grasp selection and motion planning within the cognition phase are averaged over the ten runs it took to clear one completely filled box. One can see that the longest phase is the cognition phase where objects are detected and the grasping motion is planned. This phase also includes the transmission of the sensor data to the object recognition module on a physically distinct computer.

In addition, we tested the applicability to grasp other shape primitive-based objects in earlier stages of the development as shown in Video 4.2[2].

RoboCup@Home Competition

To demonstrate the robustness of our motion planning, we used the system in the finals of the RoboCup@Home competition 2012 in Mexico City. In contrast to the industrial scenario, we picked a package of tea from a transport box to store it in a shelf as an example of a typical household task. For object detection, we used the table-top segmentation from (Stückler et al., 2011) and integrated it with our grasp and motion planning pipeline. Instead of the model-based offline processing of grasps, we performed the sampling of grasps after perception and started directly with the online grasp evaluation phase. The task was successfully fulfilled and contributed—in addition to other subtasks performed by our two robots—to winning the competition. In Video 4.3[3], we show footage from the finals.

4.6 CONCLUSION

In this chapter, we presented an integrated system for a mobile bin picking application. This requires a combination of navigation, manipulation, and perception skills.

2 Video 4.2: www.nieuwenhuisen.de/thesis/bin-picking-dumbbell.mp4
3 Video 4.3: www.nieuwenhuisen.de/thesis/bin-picking-robocup.mp4

We employ methods to flexibly grasp objects composed of shape primitives. Grasping objects is realized as a multistage process from coarse, i.e., global navigation in the environment, to fine, i.e., planning a collision-free endeffector trajectory within a multiresolution collision map.

We solve grasp and motion planning in a multi-stage process with tests of increasing complexity. We divide grasp planning into an offline and an online planning stage: In the offline phase, we examine the feasibility of grasps irrespective of the actual situation of the object in the scene. In the actual scene, these grasps are further evaluated for collisions with the environment and reachability by the robot.

We showed the applicability of our approaches in a mobile bin picking and part delivery task in our lab, where the service robot Cosero cleared a transport box with pipe connectors. Among other skills, we demonstrated mobile bin picking in the RoboCup@Home final.

Part II

MULTI-LAYERED NAVIGATION FOR MICRO AERIAL VEHICLES

5

HIERARCHICAL CONTINUOUS 3D PLANNING FOR MICRO AERIAL VEHICLES

In recent years, micro aerial vehicles (MAVs) became increasingly popular for inspection and surveillance tasks. In the future, the application of MAVs will extend to even more domains, e.g., delivery (Ackerman, 2014) or aerial manipulation tasks (Jiang and Voyles, 2013). With MAVs it is possible to reach otherwise inaccessible areas with low effort, e.g., it is possible to inspect structures in higher altitudes without the need for scaffolds or climbers. Nevertheless, at the moment most of the MAVs are remotely controlled or navigate to fixed global navigation satellite system (GNSS) waypoints that are reached in an obstacle-free altitude and hold the position. More complex operations have to be performed by a human operator. This restricts the applications to well observable obstacle-free areas in the line of sight of an operator. Flying in more challenging 3D environments, like low altitude flight in outdoor environments with vegetation or indoor environments, demands a higher level of autonomy.

Especially on larger sites, a constant connection to the MAV may not be maintainable. Also, passages may be narrow and surrounding environmental structures may be hard to perceive for a human operator. In order to safely navigate in such environments, an alternative is to have an autonomous MAV that can on its own—and without interaction with the operator—solve well-defined sub-tasks, i.e., autonomously approach multiple view-poses and collect (and/or transmit) sensor information. For the autonomous operation of MAVs, a key prerequisite is the planning of collision-free trajectories.

We aim for fully autonomous safe operation of MAVs in known and (partially) unknown environments. Obstacles can either be static, like houses and power poles, or dynamic, like humans or animals. In this thesis, we cover three application scenarios:

Creation of 3D Maps on Demand

Our MAV operates in an outdoor environment to acquire laser scans and camera images to build high-quality 3D maps of buildings and their surroundings. Special focus lies on the inspection of building facades (L. Klingbeil et al., 2014). Hence, our MAV has to operate in the vicinity of buildings and other structures, e.g., trees and power cables. Figure 5.1 shows the mapping of an old manor house close to trees and bushes. To achieve these objectives, we cannot solely rely on predefined GNSS waypoint following—especially as the accuracy

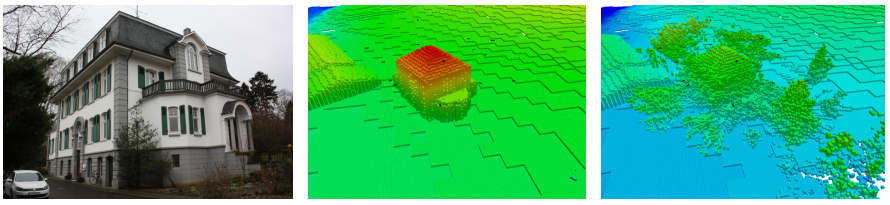


Figure 5.1: Autonomous outdoor mapping. We build mission-specific outdoor maps on demand employing an autonomous MAV. Left: Photo of the depicted area. Middle: We start with a coarse environment model—derived from city and elevation models. Right: We incorporate newly perceived obstacles, e.g., vegetation, into allocentric navigation obstacle maps during the flight.

drops in the vicinity of larger obstacles—but need more elaborated means of navigation.

In this application domain, we assume—in contrast to fully autonomous exploration—to have partial environment knowledge in advance. For initial mission and path planning, we employ digital elevation models (DEMs) and 3D city models acquired by land surveying authorities. These models—specified as Level-of-Detail 2 in CityGML (Gröger et al., 2008)—contain building footprints, heights, and roof shapes. These models do not include smaller structures, which constitute a collision hazard for the MAV. Thus, the initial mission plans need to be adjusted on the fly, whenever more information becomes available during a flight. Nevertheless, buildings are often the largest obstacles and might inhibit local path planners to find a feasible path towards the global goal. Other obstacles, e.g., power poles, vegetation, or building attachments, are likely to be small enough to be covered by our local obstacle map, built by means of efficient multiresolution scan registration. Hence, a globally consistent path enables a local planner to navigate towards a global goal.

Automated MAV-based Inventory Taking

Our MAV operates in the indoor environment of a warehouse (see Figure 5.2). In contrast to ground-based inventory, an MAV can easily reach shelf positions at higher levels and scan RFID tags or optical markers on wares and storage places. Nevertheless, remotely controlling an MAV in complex 3D environments is much more demanding than controlling a ground vehicle. Furthermore, narrow passages and fully closed rooms also prevent the use of GNSSs like global positioning system (GPS) or GLONASS such that GNSS-based hovering or waypoint following is not an option.

Initial acquisition of a warehouse model can be performed by guiding the MAV remotely. To ensure safe operation in the narrow corridors between shelves with only onboard sensors, reliable obstacle avoidance without allocentric localization is an important skill. Thus,

Figure 5.2: Flight in a warehouse. In indoor environments MAVs in general have to operate close to obstacles. Reliable obstacle avoidance is important not only for autonomous, but also for remote-controlled operation in these scenarios.

indoor operation of MAVs requires assisted teleoperation, i.e., adding an obstacle avoidance layer between the operator command input and the MAV controller, or fully autonomous operation.

Inspection of Chimneys

MAVs have a high potential to reduce the required time and costs for the inspection of industrial chimneys. This application requires a combination of some techniques from the aforementioned applications. We assume to have a coarse initial model, which consists of a geometric primitive such as a cylinder or a cone that can be easily parameterized from the chimney construction documents, and the MAV has to operate in narrow confined spaces. Figure 5.3 shows our MAV during autonomous flight in a chimney. To build detailed 3D models of the chimney surface with a 3D laser scanner and RGB-D cameras, the MAV has to approach the surface closely. Due to the height of chimneys, this cannot be accomplished by a human pilot in a safe way. Figure 5.3 shows an example of the pilot's perspective in a chimney. Thus, fully autonomous operation with obstacle avoidance is necessary. Specific to this application is the focus on navigation relative to the surface—which is a 2D manifold—and a strongly reduced complexity of allocentric planning by a nearly convex workspace.

Common to all of these approaches is a multi-layered approach to navigation. This enables the MAV to quickly react on obstacles and to allow for globally consistent planning. Our planning hierarchy goes from slower deliberative to fast reactive layers and includes mission planning, global and local path planning, and fast local obstacle avoidance. We generate motion commands for the different underlying motion controllers (Beul and Behnke, 2017; Beul et al., 2015; Kamel et al., 2017).

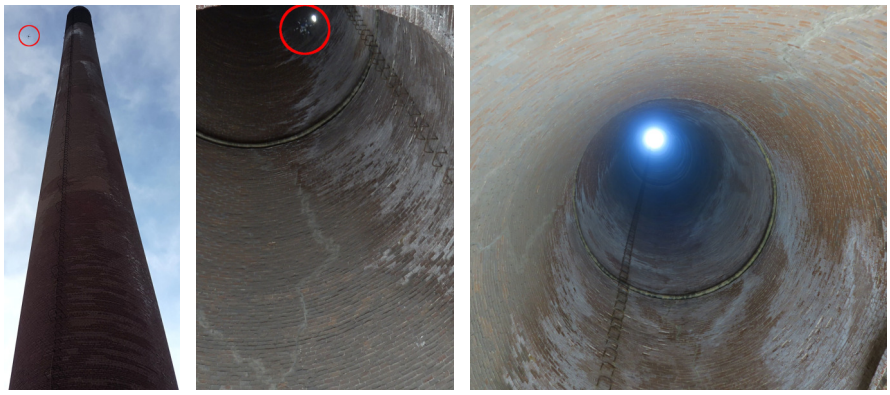


Figure 5.3: Industrial chimney inspection. Surface inspection in industrial chimneys with MAVs requires safe navigation in proximity to the surface in a small confined space. The perspective of a pilot on the bottom of a chimney makes flight maneuvers in larger heights hard to control without automation (MAV circled red). Left: Manual chimney inspection from the outside in larger distance from the wall is not suitable for 3D surface reconstruction. Center: Autonomous flight close to the walls. Right: Manual operation in larger heights inside the chimney is prohibitive, especially in daylight conditions.

5.1 RELATED WORK

In the last years MAVs have become popular in many areas. This raised the demand for more autonomy. First assistance functions for MAVs were hovering at a specified GNSS position (Bouabdallah et al., 2004). This was extended to processing a list of waypoints or follow a specified trajectory (Puls et al., 2009). Usually, this does not include obstacle avoidance and the MAV approaches waypoints on a direct line while flying at a height that is assumed to be obstacle-free. A special case is returning to a previously recorded position, e.g., the start position, where tracing back the trajectory to reach the current position (DJI, 2017b). Some of the first fully autonomous MAV systems were presented by Grzonka et al. (2012) and Shen et al. (2011).

Still, for most MAVs employed in real world applications a human pilot is necessary to teleoperate the vehicle to ensure safe operation in the vicinity of obstacles. It is an active research area to overcome this limitation.

Path Planning

An early example for autonomous indoor MAV flight is Grzonka et al. (2012). However, they simplify path planning and obstacle avoidance to a 2D manifold in a fixed distance to the ground due to the employed 2D laser rangefinder (LRF) and very limited compute power on the MAV. We allow omnidirectional 4D movements (3D transla-

tion + yaw rotation) of our MAV. Similar to their work, we reduce the planning dimensionality by deriving the MAV orientation from the planned 3D path in many applications.

Heng et al. (2014) use a multiresolution grid map to represent the surroundings of a quadrotor. A feasible plan is generated with a vector field histogram. Schmid et al. (2014) autonomously navigate to user specified waypoints in a mine. The map used for planning is created by an onboard stereo camera system. By using rapidly-exploring random belief trees (RRBTs), Achtelik et al. (2014) plan paths that do not only avoid obstacles, but also minimize the variability of the state estimation. Other search-based methods for obstacle-free navigation include work of MacAllister et al. (2013). They use A* search to find a feasible path in a four-dimensional grid map. They also incorporate the asymmetric shape of their MAV. SPARTAN from Nuske et al. (2015) generates an approximation of a 3D visibility graph on the fly. Here, also A* search is employed to find a shortest path in the visibility graph.

Due to the limited computational power onboard the MAV, in particular low computational costs are crucial for the applicability of these methods. To meet real-time demands, layered planning approaches are often used.

Andert et al. (2010) use a three-level hierarchical behavior control algorithm to fly a helicopter through a gate. Whalley et al. (2014) employ five navigation layers to fly 230 km/h with a helicopter. Obstacles are detected and avoided with an onboard laser scanner. While in their work sensing and consequently planning is limited to a narrow field of view (FoV) in flight direction, we employ full 3D planning, including flying sideways and backwards. The sampling-based BIT* planner is used by Lan et al. (2016) to plan a geometric obstacle-free path for an MAV. This path is refined to a dynamically feasible trajectory employing a second layer which connects path points with generated trajectory segments.

Many approaches address the problem of planning sensor poses (Englot and Hover, 2010; Stefas et al., 2018). In contrast to our work on planning paths that stay in the sensor FoV, they aim at covering allocentric areas of interest, not necessarily in the direction of flight. We aim at covering egocentric areas of interest that move together with the MAV. Costante et al. (2018) extend RRT* by photometric information. This yields paths that minimize the localization error during flight. Nevertheless, they do not consider sensor constraints for obstacle avoidance.

Trajectory Optimization

To plan high-dimensional trajectories, often sampling-based planners are employed, including KPIECE (Şucan and Kavraki, 2008) and ran-

domized kinodynamic planning (LaValle and Kuffner, 2001). Implementations for many sampling-based planners are provided in the Open Motion Planning Library (OMPL) (Şucan et al., 2012). In addition to those sampling-based motion planning algorithms, trajectory optimization allows for efficient generation of high-dimensional trajectories. Covariant Hamiltonian optimization and motion planning (CHOMP) is a gradient-based optimization algorithm proposed by Zucker et al. (2013). It uses trajectory samples, which initially can include collisions, and performs a covariant gradient descent by means of a differentiable cost function to find an already smooth and collision-free trajectory. A planning algorithm inspired by CHOMP is stochastic trajectory optimization for motion planning (STOMP) by Kalakrishnan et al. (2011). STOMP combines the advantages of CHOMP with a stochastic approach. In contrast to CHOMP, it is no longer required to use cost functions for which gradients are available, while the performance stays comparable. This allows to include costs with regard to, for instance, general constraints or motor torques. Pavlichenko and Behnke (2017) use a modified version of STOMP for multicriteria optimization. The optimized criteria include the trajectory duration and joint limits, in addition to obstacle costs, but no sensor visibility constraints. Another algorithm derived from CHOMP is ITOMP, an incremental trajectory optimization algorithm for real-time replanning in dynamic environments (Park et al., 2012). In order to consider dynamic obstacles, conservative bounds around them are computed by predicting their velocity and future position. Since fixed timings for the trajectory waypoints are employed and replanning is done within a time budget, generated trajectories may not always be collision-free.

The real-time generation of highly dynamic trajectories is mostly performed in free space inside of motion capture volumes, e.g., to quickly reach also dynamically changing goal states (Mueller et al., 2015). Other approaches plan collision-free trajectories in advance and execute those with high precision (Richter et al., 2013). Similar to our approach, they plan MAV trajectories in a low dimensional space (using RRT*) and optimize the trajectory with a dynamics model afterwards to achieve short planning times. Our approach does not have the constraint that the optimized path has to include the planned waypoints. Another approach using optimization by means of polynomial splines between waypoints focuses on time-optimal trajectories computed in real-time (Bipin et al., 2014). Collisions are avoided by intermediate waypoints from a high-level planner and are not explicitly considered in the optimization process. These approaches assume complete knowledge of the environment and very reliable control of the MAV. In most application scenarios outside of a controlled lab, the environment can change unpredictably or acquiring a model of the environment itself is the mission objective. Thus, closing the gap

between conservative flying in free space and dynamic trajectory following is key to expand the application domain of MAVs.

Andreasson et al. (2014) employ optimization to compute steerable trajectories for automated ground vehicles. Similar to our approach, they first plan a path and optimize it to a drivable trajectory in a second step. While they use a lattice-based 2D planner, we plan in the 3D state space and employ an intermediate step to get closer to a dynamically feasible trajectory. For flying an MAV inside of ships, Fang et al. (2017) also add a global planning layer to initialize trajectory optimization. Nevertheless, their planning layer is restricted to 2D. In contrast, we plan and optimize in 3D space.

Optimization of trajectories with continuous timings can be achieved by fitting polynomials to the trajectory points (Oleynikova et al., 2016). We aim at using a time discretization that matches the discretization of the underlying controller for the beginning of the trajectory.

Majumdar and Tedrake (2017) use compositions of preprocessed trajectories to generate flight paths that are safe under uncertainty in real-time. In contrast, we frequently modify a trajectory in real-time to react on changes in the environment and uncertain path execution. Zhang et al. (2018) generate a set of dynamically feasible paths prior to a flight and quickly select suitable ones based on sensor input during the flight.

Obstacle Avoidance

So far, most of the approaches to obstacle avoidance developed for MAVs are camera-based, due to the limited payload.

Early reactive collision avoidance algorithms for MAVs are based on optical flow (Green and Oh, 2008) or a combination of flow and stereo vision (Hrabar et al., 2005). However, solely optical flow-based solutions cannot cope well with frontal obstacles and these methods are not well suited for omnidirectional obstacle avoidance as needed for our scenarios. Mori and Scherer (2013) detect obstacles by visual features. This allows to react on frontal obstacles. A learning-based visual approach has been presented by S. Ross et al. (2013). The MAV predicts steering angles based on video data and is corrected by a supervisor after the flight if a prediction would lead to a collision. These approaches are mostly employed to steer an MAV through vertical obstacles, like trees. Obstacle avoidance in recent consumer MAVs is also based on visual cues (DJI, 2017a). Hence, collision avoidance is restricted to the FoV of the cameras.

Israelsen et al. (2014) present an approach to local collision avoidance that works without global localization and can aid a human operator to navigate safely in the vicinity of obstacles. Similar to our approach, this can be used for assisted teleoperation to prevent a pi-

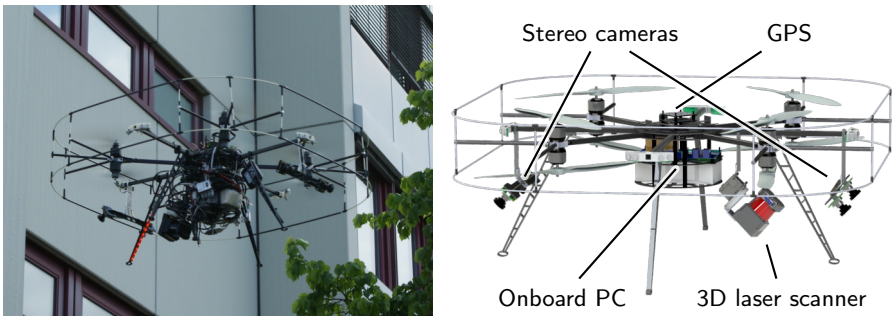


Figure 5.4: *MoDCopter* MAV. Our MAV is equipped with eight co-axial rotors. For localization and obstacle perception it employs a multimodal sensor setup, including a continuously rotating 3D laser scanner, two stereo cameras, and a precise GPS system.

lot to crash into obstacles. We extend this idea by actively avoiding approaching obstacles instead of only restricting the possible pilot commands.

A two-level approach to collision-free navigation using artificial potential fields on the lower layer is proposed by Ok et al. (2013). Similar to our work, completeness of the path planner is guaranteed by an allocentric layer on top of local collision avoidance. In contrast to this work, we consider the current motion state of the MAV and select motion commands accordingly. Johnson and Mooney (2014) use reactive obstacle avoidance on a small helicopter for velocities up to 12 m/s. This is achieved by a selection of motion primitives from a small subset of possible motions to locally avoid obstacles. They do not address hover conditions with approaching obstacles.

Florence et al. (2016) use a combination of vision and a 2D laser scanner to avoid obstacles at high velocities. Their system flies in cluttered unknown environments with large state uncertainties. For our application, we rely on high-level planning to surround larger obstacles.

Obstacle avoidance for teams of MAVs have been proposed by Baca et al. (2018). Similar to our work, they use obstacle avoidance as a safety layer below other planning layers. The MAVs share information about their future trajectories to avoid each other if the high level plans would yield a collision. As the focus here lies on the coordination of MAV teams in an open space, no external sensing is used and, thus, obstacle avoidance relies on the shared information.

5.2 SYSTEM SETUP

We developed and tested our MAV planning and navigation components on multiple flying platforms. This section gives a brief overview over the four main platforms used in this thesis.

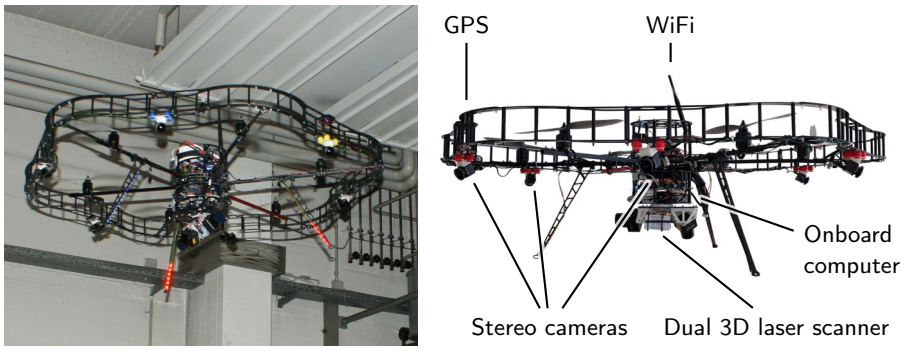


Figure 5.5: *AIRCopter* MAV. Our second flying platform is a hexarotor that was developed for autonomous warehouse inventory. It is equipped with three stereo camera pairs, a continuously rotating laser scanner aiding obstacle perception and indoor localization, and an RFID scanner.

MoDCopter

Our first MAV platform *MoDCopter* is an octorotor platform with a co-axial arrangement of rotors, depicted in Figure 5.4 (Holz et al., 2013). This yields a compact flying platform that is able to carry a plurality of sensors and an onboard computer with sufficient computing power (Intel Core i7-3820QM 2.7 GHz, 8 GB RAM) for sensor data processing and navigation planning. To allow for safe omnidirectional operation of the MAV in challenging environments, the MAV is equipped with a multimodal sensor setup. Our main sensor for obstacle perception is a continuously rotating 3D laser scanner based on a lightweight Hokuyo UTM-30LX-EW 2D LRF that measures distances of up to 30 m (Droeschel et al., 2013). The measurement density of the 3D laser scanner varies and has its maximum in a forward-facing cone. This laser covers the space around the MAV in almost all directions—only a small cone towards the upper rear of the MAV is shadowed—at a rate of 2 Hz. Two stereo camera pairs—pointing in forward and backward direction—are used for visual odometry and obstacle perception. Equipped with fish-eye lenses they cover a large area around the MAV. For outdoor localization the MAV is equipped with a u-blox GNSS module. This module can be replaced with a precise differential GPS (Eling et al., 2013) if required.

AIRCopter

The second MAV platform *AIRCopter*, depicted in Figure 5.5, is a hexarotor with a frame surrounding the rotor plane to enhance safety in indoor environments (Beul et al., 2015). While fragile equipment like computer and laser scanner lie in the core of the MAV, the frame protects the rotors and is used for mounting multiple sensors. In con-

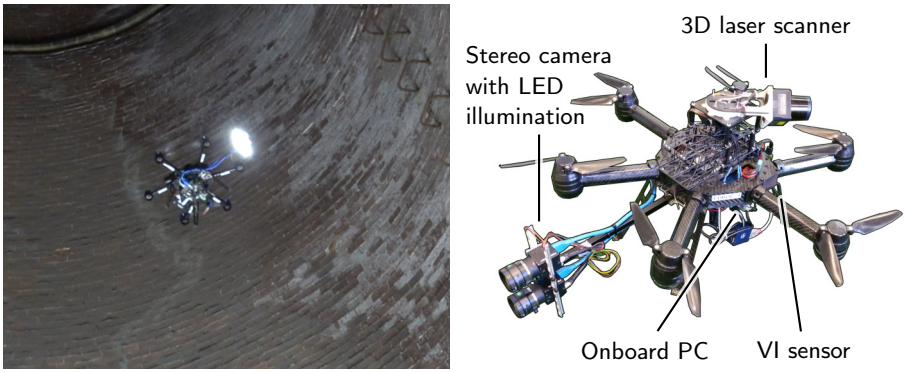


Figure 5.6: *ChimneySpector* MAV. Our MAV is a hexarotor equipped with a rotating 3D laser scanner for localization and obstacle avoidance, a VI-sensor for visual odometry estimation, and a stereo camera for surface reconstruction.

trast to the *MoDCopter* platform, the hexarotor design makes the MAV slightly larger while yielding more thrust per rotor. For sensor data processing and navigation planning, the MAV is equipped with an Intel Core i7-4770R quadcore CPU (3.2 GHz) and 16 GB RAM.

The *AIRCopter* platform is equipped with an evolved multimodal sensor setup for state estimation, obstacle detection, localization and mapping. It has three stereo camera pairs, yielding an omnidirectional FoV. Furthermore, it has a 3D laser scanner consisting of two rotating Hokuyo UST-20LX laser scanners, reducing the area occluded by the MAV and doubling the perception frequency for large parts of the environment to 4 Hz. Each laser scanner provides a scanning range of up to 20 m with 270° apex angle. For stocktaking applications it is equipped with an RFID reader.

ChimneySpector

Our chimney inspection robot *ChimneySpector* is based on the Ascending Technologies Neo hexarotor platform (Quenzel et al., 2018). With a diameter of about only 80 cm, the platform is well-suited for indoor flights. Figure 5.6 shows the MAV and the used sensor setup.

A rotating Hokuyo UST-20LX LRF on top of the MAV is employed for localization and obstacle avoidance. The sensor rotates at a frequency of 1 Hz yielding a spherical 3D FoV. Due to the 270° apex angle and its mounting pose, it covers the space above the MAV with 2 Hz—chimney inspection starts at the ground and thus unknown obstacles are more likely to be above the MAV—and the space below with 1 Hz.

The platform is equipped with a front-facing Skybotix VI-Sensor (Nikolic et al., 2014) used as stereo camera system for visual odometry.

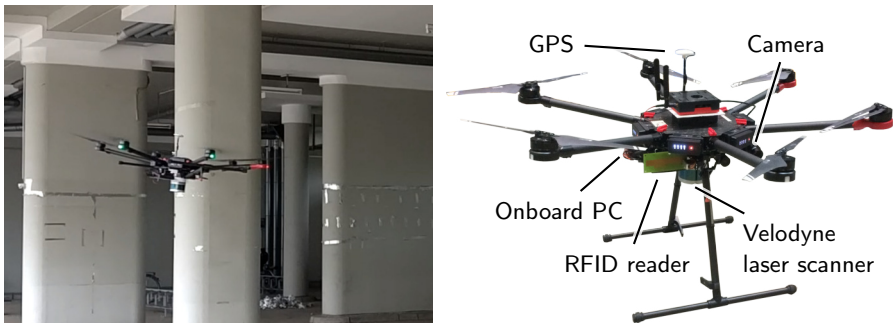


Figure 5.7: *MBZIRC2* MAV. *MBZIRC2* is based on the commercially available DJI Matrice 600 platform. It is equipped with a high-frequency Velodyne Puck LITE 3D laser scanner. For stocktaking applications it has two cameras pointing to both sides and an RFID reader.

For surface reconstruction and inspection, the MAV is equipped with a compact stereo camera rig at the rear-end of the MAV. The MAV is equipped with a small and lightweight Intel NUC PC with an Intel Core i7-5557U dual core CPU running at 3.1 GHz and 16 GB of RAM. The overall weight of the system with all sensors and batteries is about 3.4 kg.

MBZIRC2

The newest employed MAV is based on the commercially available DJI Matrice 600 hexarotor platform, depicted in Figure 5.7 (Beul et al., 2018). With a diameter of 1.7 m it is larger than the previously employed MAVs, this requires even more accurate navigation for flying in indoor environments. In contrast to the other MAVs, the Matrice 600 is equipped with a fixed Velodyne Puck LITE laser scanner that captures the surroundings of the MAV with up to 20 Hz—an order of magnitude faster than the rotating 3D laser scanners. This facilitates quick detection and propagation of dynamic and static obstacles, but at the price of a limited FoV compared to the near spherical FoV of the other MAVs. The employed scanner setup allows for fast navigation, but requires tailored navigation approaches to address the sensing limitations.

For stocktaking missions in a warehouse, it is equipped with an RFID reader, similar to the *AIRCopter* platform. Onboard computation is performed on a PC equipped with an Intel Core i7-6770HQ quadcore CPU running at up to 3.5 GHz and 32 GB of RAM. The MAV takeoff-weight is approximately 11 kg.

General Components

On all our MAVs, we employ the Robot Operating System (ROS) by Quigley et al. (2009) as middleware. For low-level velocity and attitude control, the *MoDCopter* and *AIRCopter* MAVs are equipped with Pixhawk flight control units (Meier et al., 2012). The *ChimneySpector* and *MBZIRC2* MAVs are equipped with proprietary flight control units, namely the AscTec Trinity and the DJI A3, respectively.

For allocentric localization, we use GNSS outdoors and laser-based 6D localization employing a previously acquired environment map (Droeschel et al., 2014a) in GNSS-denied environments. Low-level (egocentric) state estimation, e.g., attitude, velocity, and acceleration, is performed employing visual odometry and inertial measurement unit (IMU) measurements—allocentric localization information is incorporated indirectly through an extended Kalman filter (EKF). We control the MAVs egocentrically. Hence, allocentric localization is not required for basic operation. Control commands sent from the onboard computer to the Pixhawk flight control unit are egocentric 4D velocities (linear velocities and yaw rate).

On the *ChimneySpector*—in contrast to *MoDCopter* and *AIRCopter*—we rely on a proprietary flight control unit, and we employ control components provided to the participants of the European robotics challenges (EuRoC) by the challenge hosts. Consequently, the low-level interfaces differ from the Pixhawk-based systems. To keep the differences between the MAV systems small, we estimate the MAV odometry with a port of the modified Pixhawk state estimation filter used on our previous MAVs to the onboard PC. The AscTec Trinity is controlled by attitude and thrust commands, calculated onboard the PC with the linear model predictive controller (MPC) from Kamel et al. (2017). We modified the MPC to be controllable by either velocity setpoints—similar to the two previous systems—or trajectory segments including positions and derivatives generated by higher layers.

Also, *MBZIRC2* is equipped with a proprietary flight controller. It provides different control modes, e.g., position control, velocity control, or attitude control. We employ the low-level attitude control mode, the attitude commands are generated by the MPC from Beul and Behnke (2017). Input to the controller are either pose setpoints or intercept points that include the current pose of a target point and its velocity. In the first case the MAV reaches the position with a given velocity, in the second case the MAV intercepts the moving target and follows its movement. Similar to the *ChimneySpector*, we employ the PC port of the Pixhawk state estimation filter.

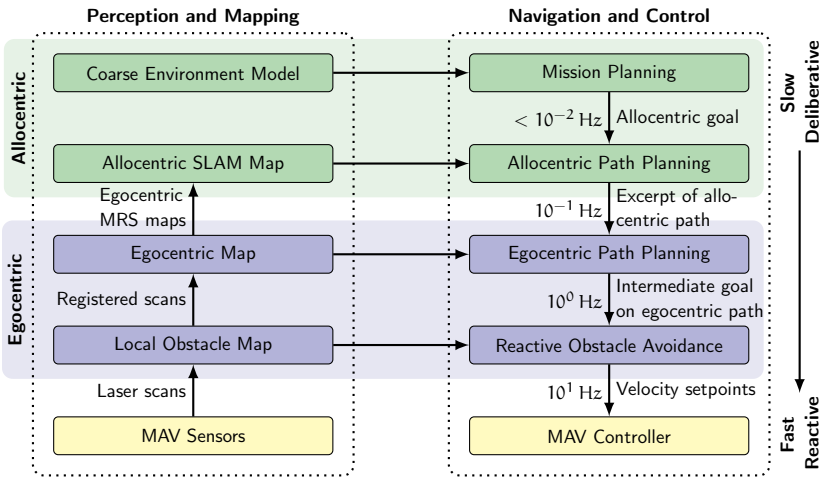


Figure 5.8: Planning hierarchy for a mapping mission. From top to bottom the execution frequency increases while the environment representations become more local and the planned action sequences cover shorter durations and become more specific.

5.3 PLANNING AND NAVIGATION HIERARCHY

To operate MAVs in complex environments for inspection or mapping missions, safe navigation in the vicinity of obstacles is key. Especially when operating indoors, the free space is restricted and keeping a large safety margin to obstacles is not an option. Hence, only quick reactions given the observed vehicle and environment state ensure the successful and safe mission completion. To allow for direct reactions on obstacle perceptions on one end and consistent mapping and complex planning on the other end, our system architecture is layer-based (see Figure 5.8) with slower global layers on the top (deliberative planning, allocentric mapping) and faster local layers on the bottom (reactive obstacle avoidance, egocentric obstacle maps). From top to bottom the abstraction level of planning and mapping is reduced and the processing frequency approaches the sensor measurement frequency. In the following, we will focus on the planning part of our hierarchy.

To plan an initial mission, we need a coarse (semantic) model of the environment; to plan collision-free paths, we need a finer and up-to-date consistent geometric model; and to avoid collisions, we need a non-aggregated local representation of the close vicinity of the MAV. The planned actions also have different granularity, which is represented by the planning frequency, from once per mission to multiple times per second. The higher-layer planners set goals for the lower-level planners which produce more specific action sequences based on more local and up-to date environment representations. In detail, the navigation layers are:

- *Mission planning:* The topmost layer is a mission planner. Input to the mission planner is a set of view poses defined by the user. A coarse static environment model is employed to determine a cost-optimal sequence of view poses. The result of mission planning is a flight plan composed of an ordered list of 4D waypoints (x, y, z, yaw). It is executed once per mission.

- *Global path planning:* The global path planner operates either on a static or updatable geometric allocentric environment model to plan a 3D path to the next active view pose in the flight plan. Replanning based on the current MAV state is performed every 5–10 s to account for deviations from the flight path and to incorporate updated allocentric maps. Output is an allocentric 4D path to either trajectory optimization or local path planning, depending on the application.

- *Trajectory optimization:* We employ trajectory optimization as a way to generate dynamically feasible allocentric trajectories. A globally planned 4D path is refined given dynamic constraints of the MAV. The resulting trajectory is smooth and can be followed with low-level controllers. Positions, velocities, and accelerations are defined for every point of the trajectory.

- *Local path planning:* The local path planning layer employs egocentric multiresolution map and plan representations. This layer plans based on the allocentric path within the range of the onboard sensors. It refines the global path according to the actual situation at 1–2 Hz, the duration of perceiving the environment in every direction with the 3D laser scanner. Output to the next layer is the next intermediate 4D waypoint on the local plan relative to the MAV.

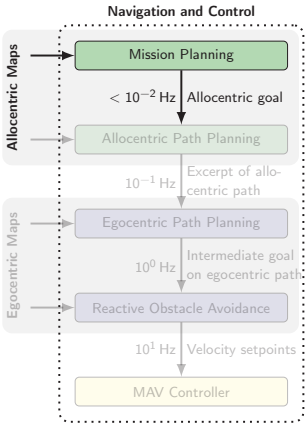
- *Obstacle avoidance:* Due to the complex flight dynamics of the MAV and dynamic or previously unknown obstacles, it is necessary for the MAV to quickly react on deviations from the plan. Between the control and the planning layers, we employ a fast reactive collision avoidance module based on artificial potential fields as a safety measure. It reacts directly on the available sensor information at a higher frequency than used for planning. This enables the MAV to immediately react to perceived obstacles in its vicinity. Also in manual operation, obstacle avoidance assists a human pilot to operate the MAV safely in challenging situations, e.g., flying through a narrow passageway. The output of this layer are egocentric 4D velocity commands.

- *MAV controller:* Within the lower layers, high-frequency controllers stabilize the attitude of the multicopter. A control layer provides an interface to the higher layers, allowing control of

linear and angular velocities instead of the multicopter attitude and thrust. This layer is out of the scope of this thesis.

In the following chapters, we will detail the layers of the navigation hierarchy from the deliberative layers on the top to the reactive layers on the bottom.

6 MISSION PLANNING



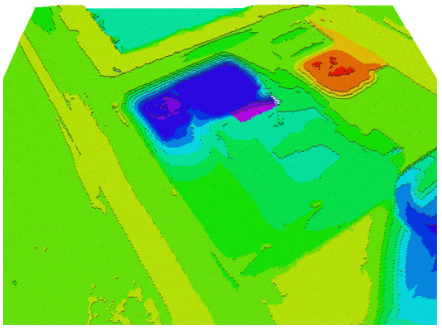
On top of our planning hierarchy is the mission planning layer. This layer allows for high-level interaction with the user and is, naturally, tailored to the specific application domain we address. We consider two qualitatively different mission types. Our first type are missions where the micro aerial vehicle (MAV) executes a series of discrete waypoints. A set of mission-relevant 4D view and auxiliary poses for the MAV is specified by a human operator or an external view pose planning module. The mission planner determines the best order of the mission poses in terms of total flight path costs. We employ this type of mission planning for, e.g., mapping or inspection applications. The second mission type is the coverage of areas or objects with the MAV onboard sensors. Here, a user defines an area of interest and the mission planner generates coverage tours. We employ mission planners of this type for, e.g., chimney inspection, automated inventory taking, and mapping the interior of buildings from the outside.

Common to all these approaches is our assumption that partial environment knowledge—either from models or maps acquired before the mission—is available. Thus, our MAV does not explore the environment fully autonomously. Nevertheless, newly acquired information can be incorporated into lower layers during flight, such that the MAV can deviate from the initially planned mission to account for obstacles.

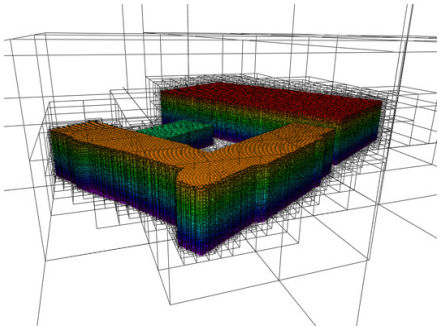
6.1 PLANNING FOR OUTDOOR MAPPING MISSIONS

For the mapping of outdoor building complexes and the surrounding environment MAVs are superior to ground vehicles. They are able to overfly obstacles and capture buildings from the ground up to the roof. To build and refine outdoor maps on demand, we start with already available information. For the initial mission and path planning, we employ digital elevation models (DEMs) and 3D city models acquired by land surveying authorities (Figure 6.1b) in outdoor applications. These models—specified as Level-of-Detail 2 in CityGML (Gröger et al., 2008)—contain building footprints, heights, and roof shapes. Figure 6.1 shows the input data that is stored efficiently in an OctoMap (Hornung et al., 2013), detailed in Section 2.2. Other structures are added to the map during flight as they are perceived.

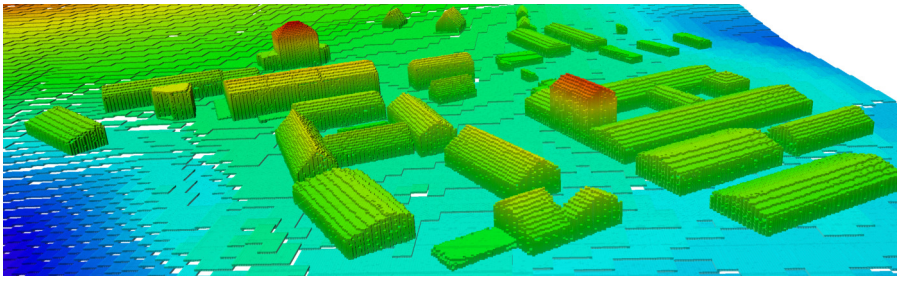
An operator defines observation poses for the mapping mission in the initial map by either setting the poses in an operator front end or



(a) Digital Elevation Model (DEM)



(b) LoD 2 model of an office building



(c) Combined DEM and LoD 2 models of a farm area

Figure 6.1: The static environment representation a priori known, consists of (a) a digital elevation model and (b) a 3D city model. The color corresponds to height. For planning, geometric representations of both are stored in an OctoMap that allows efficient storage of larger areas (c).

teach-in the poses during manual flight. The estimated flight costs between every pair of observation poses is determined by means of the global path planner described in the next section, operating on the static environment model. Figure 6.2 shows an example specification to map an old manor house and all optimal flight paths between the observation poses. Our path planner is undirected. Thus, the mission planner has to evaluate $n^2/2$ allocentric plans in total. For efficiency, we reuse information in the grid-based path planning representation in consecutive planning cycles. As the map is static, the obstacle induced costs can be reused for every plan. Furthermore, we reuse the costs to reach each grid cell if the start node remains the same, which is the case for an average of $n/2$ plans per node.

The evaluation of the minimal-cost order of mission poses is an instance of the traveling salesman problem (TSP). As our problem instance usually is comparably small (tens of poses), it is feasible to solve the problem exactly. In this case, we employ the freely available Concorde TSP solver (Applegate et al., 2006) with our previously calculated distance matrix. For larger problem instances, we employ the approximation by Lin and Kernighan (1973). Together with the costs of pairs of mission poses, the corresponding planned paths are stored. These paths can serve as initial guess to speed up the global path plan-

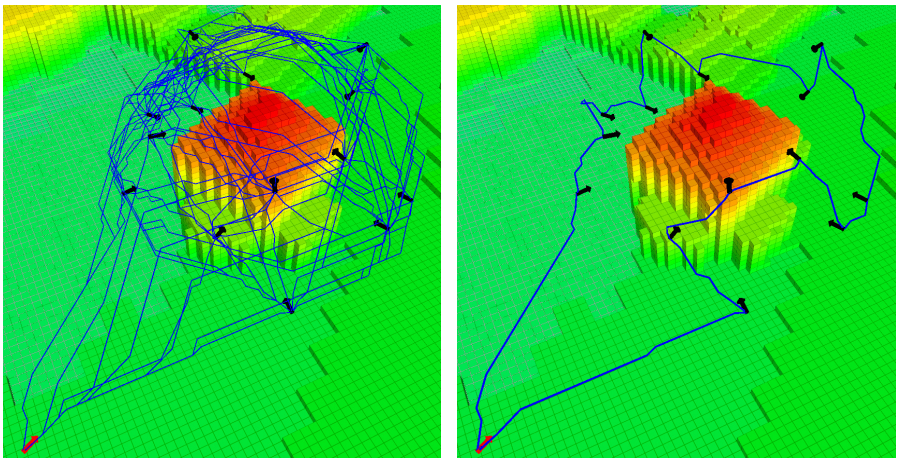


Figure 6.2: Mission planning for outdoor mapping. A mission planner evaluates the best execution order of mission poses (black arrows). Left: All cost-optimal trajectories between each pair from the set of mission waypoints, including the current robot pose (red arrow). Right: The optimal flight plan to reach all waypoints and return to the start pose. The cost function allows for positions close to the building but penalizes these more than paths farther away.

ning on the next lower layer. The result of mission planning is a flight plan composed of an ordered list of 4D waypoints (x, y, z, yaw).

An interface to modules for more high-level interaction with the user was established to the software from Loch-Dehbi et al. (2013). By employing this interface a user can select parts of buildings in a semantic map to be inspected, e.g., the rain gutter of a building. The external software generates poses along the building part that a camera on the MAV shall observe, the mission planner then generates inspection poses for the MAV in a safe distance and includes these into the mission. Another extension to our approach includes the use of simultaneous localization and mapping (SLAM)-based maps to operate in combined indoor and outdoor environments. Figure 6.3 shows an example solution for a mission to inspect parts of a hall and a garage, based on a laser map.

6.2 PLANNING FOR WAREHOUSE INVENTORY MISSIONS

Another application domain we cover is autonomous warehouse inventory. Here, we employ semantic descriptions of the warehouse structure to derive an initial map (Figure 6.4).

The layout of large warehouses yields in general a very structured basic environment. Large halls are filled with shelves containing standardized storage units, e.g., capable to store exactly one EUR-palette of size 800 mm × 1200 mm. Thus, on the topmost layer, we describe equal parts of a warehouse by numbers of shelves, unit height, and

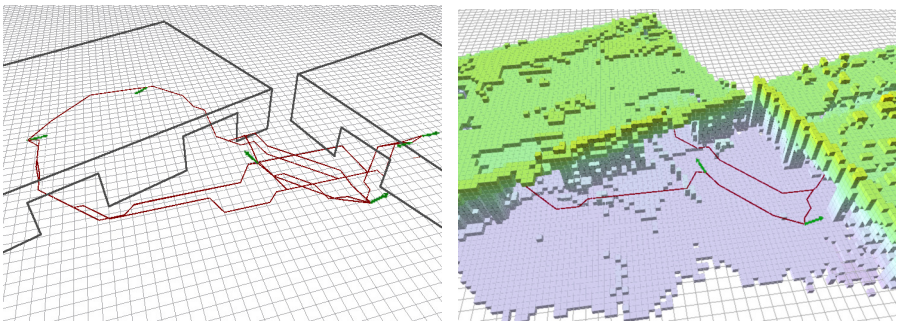


Figure 6.3: Mission planning in combined indoor/outdoor laser map. Mission poses are depicted by green arrows. Red lines depict planned cost-optimal paths between mission poses. The number of possible flight paths is reduced by the constraints that the MAV has to fly through the doors. Left: Planned paths between each pair of mission relevant poses. Right: Cost optimal flight path that yields a cost-optimal order of mission poses from the current MAV pose. Color encodes height.

the numbers of units in horizontal and vertical direction. If the storage unit IDs are assigned in a systematic way, we can derive a mapping between storage unit coordinates, scan positions, and IDs automatically. In our evaluated scenario this is the case—storage unit IDs contain shelf number, vertical and horizontal position and sub-unit. Otherwise, a simple mapping table can still be easily created without exactly measuring unit positions in the warehouse. Figure 6.4 depicts a resulting model of a warehouse. We derive an initial OctoMap from the model for navigation planning. For development and debugging, flight plans containing flights to individual storage units and coverage paths for whole shelves can be assembled employing an RViz-based interface. In real-world applications, IDs of shelves or units to inspect will be provided by a warehouse management system (WMS).

Coverage paths to inventorize shelves are generated employing a user-defined distance to the shelf and the sensor apex angles. A 10 % overlap between scans allows to detect visual tags that could be cropped otherwise and mitigates the effects of small deviations from the flight altitude. To mitigate effects of motion blur, a constant velocity without stopping in front of individual storage units can be advantageous. Thus, the mission planner allows for continuous flight by removing colinear waypoints on an inventory mission if specified in the mission profile. Missions can include an arbitrary set of individual storage units and coverage patterns which are then ordered and connected to a complete trajectory by formulating the mission as a TSP, as before. For simplicity, we omit the horizontally flipped version of coverage patterns—i.e., with endpoints on the top-left and bottom-right instead of bottom-left and top-right—as this would result in an ambiguity of the pattern endpoints and add additional complexity to the optimization problem without much gain.

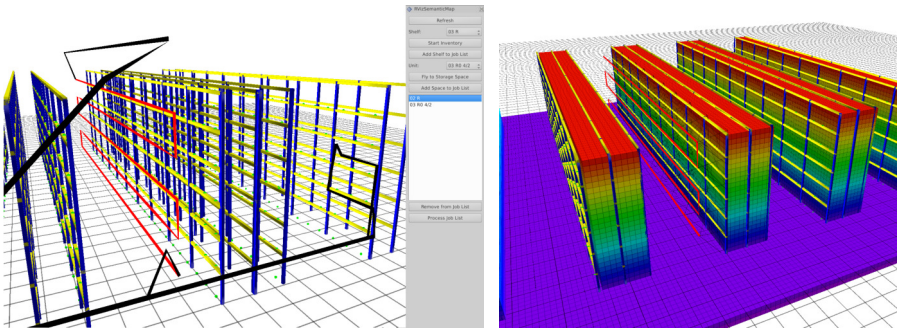


Figure 6.4: Coverage planning in semantic map. Based on warehouse parameters a semantic map of the shelves is generated. Green dots depict storage units. Left: An operator can command coverage tours to inventorize complete shelves (red path in left aisle), flights to specific storage units (black path to right aisle), or a combination as part of a more complex flight plan. Right: To aid initial mission and path planning, we derive an OctoMap from the semantic map (color depicts height).

6.3 COVERAGE PLANNING FOR CHIMNEY INSPECTION MISSIONS

Furthermore, we developed similar methods for the inspection of industrial chimneys based on simple geometric parameters that can be easily measured or derived from blueprints or aerial photography. We model chimneys either as a regular prism—with the parameters width and height of one wall segment and the number of segments—or as a conic section—with bottom and top radii and height.

Capturing the surface of a chimney requires a steady flight path with a fixed distance between sensor and walls. Furthermore, the images need sufficient overlap in every direction to build a consistent 3D model for the whole flight. These demands are hard to fulfill in manual operation, especially given the turbulent air movement close to the walls pushing the MAV away and requiring constant control actions. Thus, we operate the MAV fully autonomously except for start and landing. To cover the whole surface, we plan an upwards spiraling inspection path starting at 1 m above the ground. The parameters of the spiral are determined by the sensor characteristics, i.e., horizontal and vertical apex angles and best scanning distance, and the part of the chimney to cover.

In a first setup, we captured the surface of a chimney with an RGB-D camera. This demanded a large overlap between the images in all four directions. Given an image overlap of 50 % as a constraint, our coverage planner generates an upwards spiraling base motion overlaid with a circling motion parallel to the wall to ensure a good image overlap and facilitate loop closures. This pattern has been found to be advantageous over a simple spiraling motion. Figure 6.5 shows the resulting inspection path in our octagonal chimney mock-up.

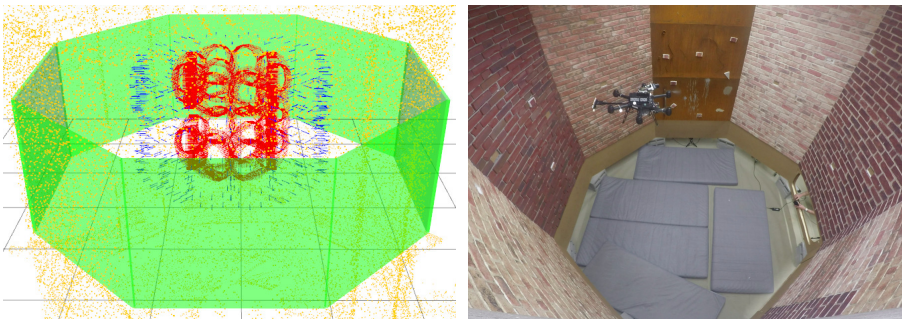


Figure 6.5: Inspection of chimney mock-up. For building a 3D chimney model with an RGB-D sensor requires sufficient overlap between images. Thus, our mission planner generates a double spiral coverage pattern.

For the coverage of larger surfaces the vision setup was improved. With the improved setup, the coverage pattern can be simplified to a single spiral. The generation of this spiral is analogue to the base spiral in the aforementioned scenario. Figure 6.6 shows the resulting path for an actual industrial chimney.

After a first complete inspection, the user can specify poses for a targeted second inspection, e.g., to take close-up images of potential defects in the chimney. The MAV processes a set of inspection poses and determines an optimal processing order to achieve a short inspection flight employing a TSP solver. Figure 6.7 shows the solution for an example targeted inspection mission with nine inspection poses. Selection of poses for targeted inspection is assisted by a graphical tool on the ground station that shows the recorded video streams from the MAV to an operator. The operator can select points in time when potential defects are visible in the stream. These times are then matched to MAV poses and stored for a second, more detailed inspection mission.

6.4 MAPPING OF BUILDING INTERIORS FROM THE OUTSIDE

In the context of mapping buildings, detailed scans of the facade are of special interest. Not only to facilitate high resolution surface modeling, but also to map the basic structure of the interior of buildings from the outside, e.g., to guide rescue personnel. For interior mapping the windows are of particular interest. Many building facades have a regular structure. Thus, we follow an approach to model the facade similar to modeling warehouses: First, we define window templates by width, height, and margins between windows. Then, a facade segment is modeled as a matrix of dimension floors × window positions with indices to the matching window templates and a transformation to the corresponding facade in world coordinates. Figure 6.8 shows a facade model with two types of windows templates.

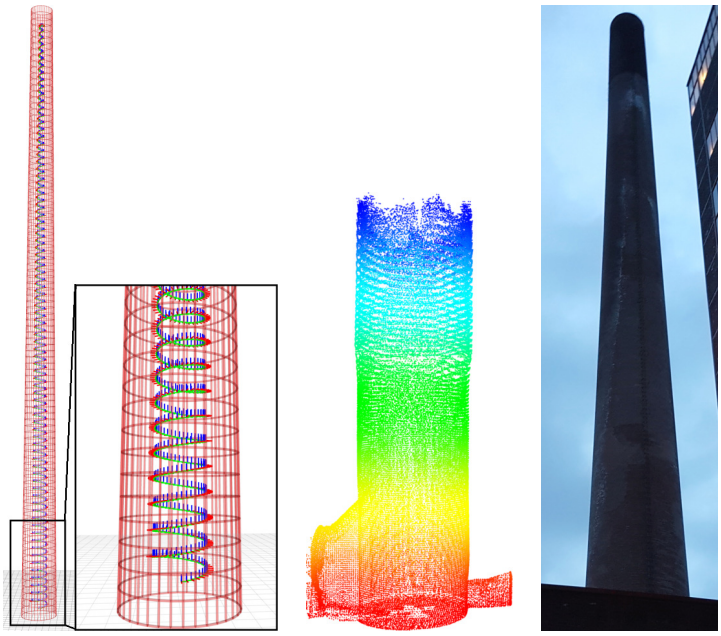


Figure 6.6: Coverage tour in the chimney. Based on a few chimney parameters, we plan sensor coverage tours in a coarse geometric chimney model (left, red). The MAV moves from the bottom to the top in a spiraling motion (sensor poses depicted by coordinate axes). In contrast to the initial laser map captured from the starting position (center), the geometric model covers the complete chimney. Right: Photo of the chimney for comparison.

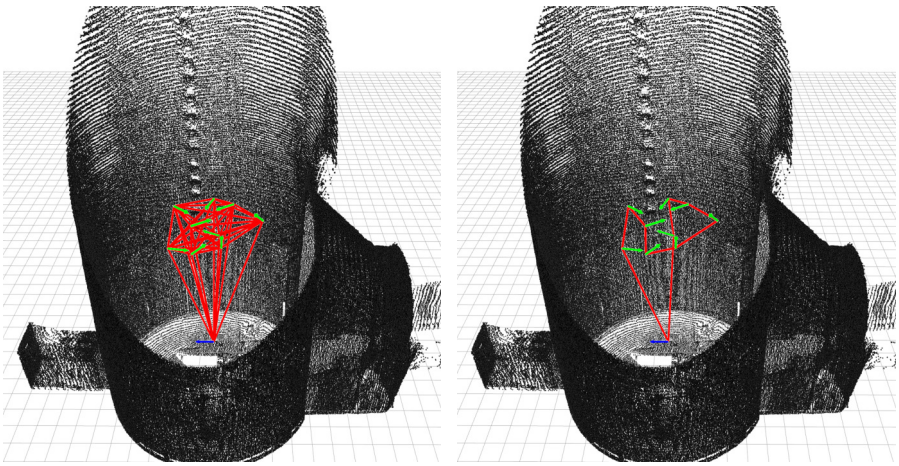


Figure 6.7: Targeted inspection. After an operator selected targets to reinspect, the observation poses (green arrows) are sent to the MAV where a mission planner finds an optimal processing order. The left figure depicts all possible paths between view poses and the start/return pose (blue arrow), the right figure shows the best mission path. The MAV navigates to these poses autonomously and hovers there for several seconds to acquire more detailed data of the surface. Black dots depict the chimney map.

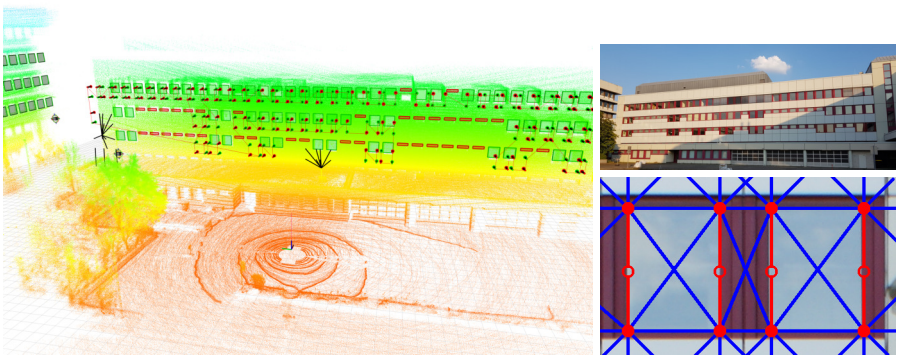


Figure 6.8: Facade model and graph structure. Left: Regular building facades are modeled by a small set of parameters. An operator can select which windows are part of a mapping mission. Windows depicted in green are part of the mission. The arrows depict endpoints of the sweeping motion, black lines connect endpoints to obstacles blocking the observation pose. Top right: Photo of the modeled facade. Bottom right: We model our problem as fully connected graph (blue) of sweep line endpoints (solid red) and sweep nodes (red cirles). Sweep nodes facilitate a sweeping motion along the vertical window edges.

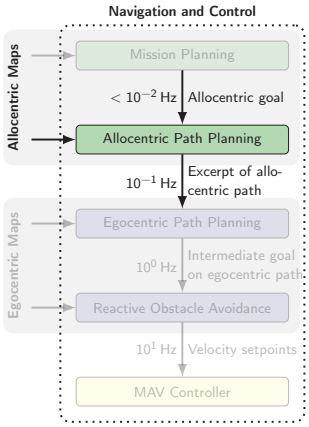
To gain good coverage of the interior with the relatively sparse Velodyne laser scans, the MAV sweeps in front of both vertical edges of a window. For planning the optimal coverage tour for one or multiple facades, we model the facades as a graph and employ the aforementioned TSP solvers. We modify the fully connected graph containing all nodes that represent start and endpoints of individual sweep edges by adding nodes with a dimensionality of two to graph edges coincident with sweep edges, depicted in Figure 6.8. The edge weights of sweep lines are zero—the weight would only add a constant cost overhead as all sweep line edges will be traversed. All other edge weights are determined by global path planning.

7

ALLOCENTRIC AND EGOCENTRIC PATH PLANNING

Whereas the mission planning layer generates an ordered list of view poses—including entry and exit poses to coverage patterns—we employ allocentric and egocentric deliberative path planning to determine shortest paths from the current micro aerial vehicle (MAV) pose to the next mission pose. The environments in which MAVs operate become more challenging with new applications, e.g., indoor and disaster response operations. These scenarios prohibit the optimistic assumption that direct flight paths are obstacle-free at a certain altitude that can be reached. Furthermore, the assumption that the environment is static cannot be made in the presence of human or machine activities, or when the structural integrity of a building cannot be assured. Hence, continuous monitoring of the environment that is traversed and quick reaction to unforeseen obstacles is key to safe and collision-free flights. In this chapter, we detail both the allocentric planning layer and the egocentric layer. Both layers have much in common while operating on different environment representations and time scales.

7.1 GLOBAL PATH PLANNING



The next layer in our planning hierarchy below the mission planner is a global path planner. This layer plans globally consistent plans, based on

I) the (updated) OctoMap environment model, discretized to grid cells with 0.25–1 m edge length, depending on the mission requirements,

II) the current pose estimate of the MAV, and

III) the next mission waypoint, including 3D position, yaw orientation, and required accuracies.

The planning frequency is in the order of 0.1 Hz and we use the A* algorithm to find cost-optimal paths.

We assume that for most MAV application domains most obstacles which are not known in advance and, thus, not represented in our allocentric map can be surrounded locally without the need for global replanning. Hence, it is sufficient to replan globally on a more long-term time scale to keep the local planner detours synchronized to the global plan and to avoid the MAV to get stuck in a local mini-

mum that the local planner cannot solve due to its restricted planning horizon.

In general, we employ A* graph-search on a representation based on a modified grid-based graph, detailed in Chapter 2.1. We represent nodes by indices to underlying grids. Thus, all properties of a node can be accessed in constant time. These properties include, e.g., the costs to reach a node from the start. The planning graph is complemented by additional helper data structures for efficiency reasons. An obstacle cost grid and a heuristic grid store these calculated values after the first node expansion. We maintain another grid as closed list to allow for efficient checks whether a node has already been expanded before. Another grid stores whether a node is already in the open list. In our implementation, we usually do not require this distinction, as we will detail later. As priority queue for our open list, we employ a heap.

The general case of finding obstacle-free shortest paths is straightforward with a cost function modeling the distance from graph nodes to the nearest obstacles. Instead of a simple Euclidean distance function, we use a partial linear distance function which has I) prohibitive high costs inside and in the vicinity of obstacles up to the minimum safe distance D_s, II) linearly decreasing costs between the safe distance D_s and a distance D_o that should be avoided if possible, III) and zero costs for larger distances. The weight w of an edge in the graph is then

$$w_e = l_i(c_i + 1) + l_{i+1}(c_{i+1} + 1),$$

where l_i denotes the edge length in grid cell i and c_i the cost function for that cell. With this edge weight function paths close to obstacles are avoided if that would not lead to long detours. Edges with prohibitive costs are not expanded in our A* implementation and the corresponding nodes are directly added to the closed list.

As a modification to the standard A* algorithm, we do not search and remove duplicate nodes in the open list, when we reduce the value of a node. In general, the algorithm would check if a node is already in the open list and compare both values. If the node is not in the open list, it is added to the priority queue. If the new node value is higher, it is discarded. If the new value is smaller, the value of the node in the open list would be updated and its position in the open list would be updated accordingly.

By representing the nodes by their indices in the open list, we can maintain a common lookup table that contains the current node values of all instances of a node. So, the algorithm can look up the current value of the node in the open list without searching this instance. Instead of reducing the value of the node in the open list, we update the lookup table and add another instance of the node to the open list, regardless of if it is already in the list. This leads to a longer

and partially unsorted open list at the advantage of saving the time to search for the duplicates and restoring the heap property of the whole open list every time a node value is decreased. Still, the heap property that the node with the lowest value is always in front of the open list is maintained: If the value of a node is reduced it is added at the corresponding position in front of the duplicate that was added before. Thus, by also maintaining a closed list, the duplicates that are at wrong positions in the open list are already closed if they arrive at the front of the open list and can simply be discarded without much overhead.

7.1.1 *Obstacle Cost Models*

For efficient distance queries for the calculation of obstacle costs, we have investigated different obstacle representations. As the initial OctoMap representation is well suited to efficiently represent occupancy but does not allow for efficient distance queries, we derive distance representations from it. The representations for our planner are

- a uniform resolution k-d tree,
- a multilevel k-d tree,
- and a multilevel k-d tree with different levels of abstraction for the MAV.

The k-d tree-based representations (Bentley, 1975) are built with the center points of the occupied cells. Thus, half of the cell size of one cell has to be subtracted from a distance query result to get the approximate distance to an occupied cell. The uniform resolution k-d tree representation is derived from a fully expanded OctoMap, i.e., all occupied leaves exist, resulting in a single tree with as many leaves as the OctoMap has. Our multiresolution k-d tree is built from a pruned OctoMap, i.e., inner nodes will replace all child nodes if they have the same occupancy. We build one k-d tree per OctoMap depth containing the center points of all occupied nodes that have no children. The result of a distance query is now the minimum of distance queries to all k-d trees. For a typical OctoMap, we now have 16 distance queries instead of a single one, but on potentially much smaller individual k-d trees, depending on the structure of the data. For all three representations, we employ the k-d tree implementation from Blanco and Rai (2014).

Common to all these representations is that the MAV is approximated by a bounding sphere. This simplifies the distance calculation as the distance to the next obstacle is the distance of the MAV pose reduced by the bounding sphere radius. Nevertheless, this approximation has the disadvantage that a bounding sphere might be a too conservative approximation if the MAV is not equally sized in all dimensions. This constrains the set of reachable positions. For example, the employed Matrice 600 platform has a diameter of 1.7 m, but a

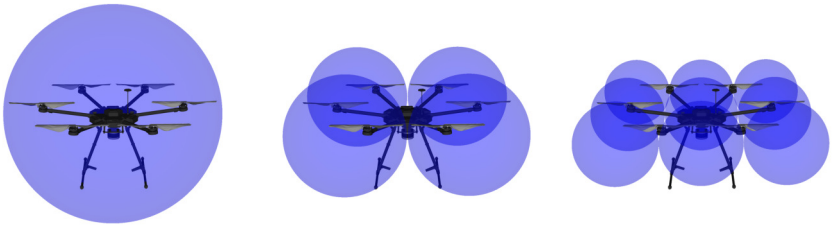


Figure 7.1: MAV representation abstraction levels. If a single bounding sphere is too conservative for planning, we employ iteratively finer MAV models with decreasing distance to obstacles.

height of only 0.5 m. Thus, we model the MAV with multiple levels of abstraction, from a single bounding sphere to a set of spheres covering the MAV shape, depicted in Figure 7.1. The coarsest model of the MAV is employed in the initial distance query. Only if the distance is below a threshold, the planner iteratively refines the model until the finest resolution is reached or the distance is below the threshold for further refinement. Figure 7.2 shows the reachable cells with and without multiple abstraction levels. Our refinements converge to a cuboid with an ground plane edge length of the MAV diameter instead of a cylinder. This results in corner cells that become unreachable with increasing MAV model resolution (red cells in Figure 7.1). In our implementation, the effect is mitigated by taking the maximum distance to obstacles from multiple abstraction levels.

7.1.2 *Specific Cost Models*

In addition to finding a shortest, obstacle-free path between two poses other objectives might be of interest depending on the mission profile. When planning w.r.t. multiple objectives c_j, the modified edge weight w_e^* employed in our planner is

$$w_e^* = l_i(\sum_j a_j c_{j,i} + 1) + l_{i+1}(\sum_j a_j c_{j,i+1} + 1),$$

where a_j are weighting factors of the individual objective costs. The objectives that we address in this section are to support laser-based localization, avoiding wind, and to stay within the range of communication infrastructure.

Supporting Laser-based Localization

For application-specific maps, it is often not necessary to cover the whole reachable environment, but only the parts that are relevant for mission execution. In particular, they cannot cover the complete space outside of buildings. The laser-based localization that we employ (Droeschel et al., 2014a) needs to perceive sufficient structure

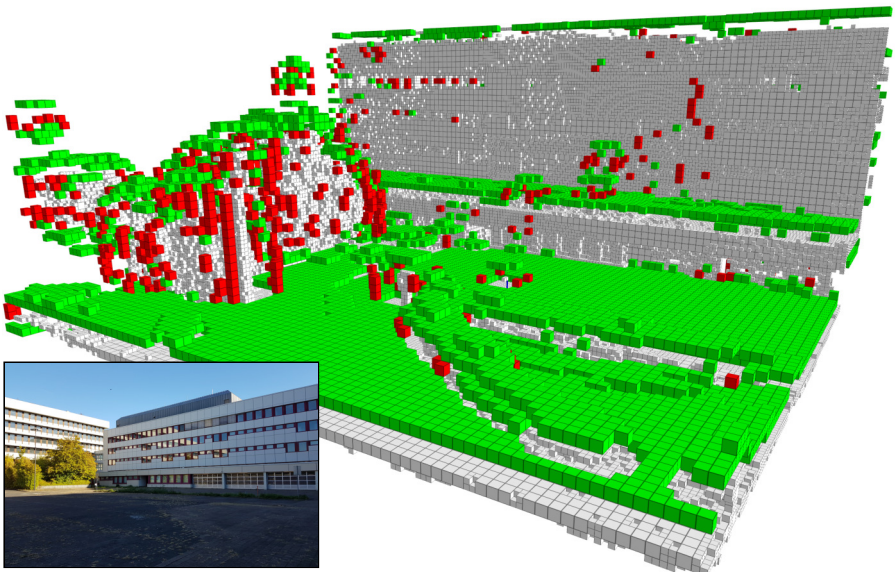


Figure 7.2: Reachable cells with three levels of MAV model abstraction. Green depicts cells reachable with the fine Matrice 600 model but not reachable with the coarse model. Especially cells above or below horizontal surfaces become reachable. Some cells close to corners become unreachable due to the more rectangular approximation of the MAV, depicted in red. This effect can be mitigated by using the maximum distance of all abstraction levels. Occupied grid cells are depicted in light gray. The photo shows the mapped environment.

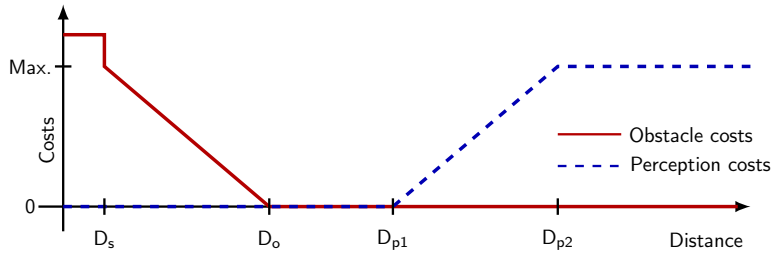


Figure 7.3: Cost function for allocentric planning. We model the traversal costs for a grid cell in our global planners as a piece-wise linear function of the distance to the next obstacle (red). With increasing distance, we have fixed maximum costs in the direct vicinity of obstacles, linear decreasing costs to fly farther away from obstacles where possible, and a range with zero obstacle costs. As our laser-based localization needs structure, the perception costs increase linear, when flying too far away from obstacles (blue). The resulting traversal costs are a linear combination of both functions.

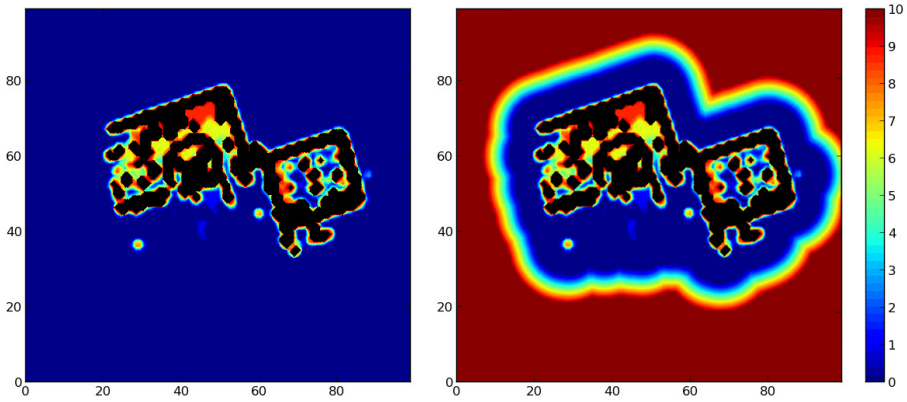


Figure 7.4: For robust LRF-based localization, laser scans have to contain sufficient structure. We modify the traversal costs to not only avoid obstacles, but also keep them within sensor range. We compare the resulting traversal costs for a cut through a map some meters above the ground plane. Left: Obstacle avoidance only. Right: Obstacle avoidance and robust localization.

to work robustly. Hence, the MAV should not fly in completely unmapped or free space, e.g., at a height where the ground is no longer observed by the laser rangefinder (LRF). To ensure robust localization even in partial maps and unbound environments, we employ an approach inspired by coastal navigation (Roy et al., 1999).

The cell traversal costs for the path planner are calculated according to the function depicted in Figure 7.3. Similar to our general approach, our obstacle cost function $h_c(d)$ decreases with increasing distance to obstacles until distance D_o. Starting at a distance D_{p1}, the perception cost function h_p increases up to a maximum at D_{p2} to keep the obstacles in the observable range of the MAV. Our cost model $h(d)$ is:

$$h_c(d) = \begin{cases} \infty & \text{if } d \leqslant D_s, \\ h_{max} \frac{1-d+D_s}{D_o-D_s} & \text{if } D_s < d < D_o, \\ 0 & \text{otherwise;} \end{cases}$$

$$h_p(d) = \begin{cases} 0 & \text{if } d \leqslant D_{p1}, \\ h_{max} \frac{d-D_{p1}}{D_{p2}-D_{p1}} & \text{if } D_{p1} < d < D_{p2}, \\ h_{max} & \text{otherwise;} \end{cases}$$

$$h(d) = w_1 \cdot h_c(d) + w_2 \cdot h_p(d),$$

with equal weights $w_1 = w_2$. D_s is the safety distance around obstacles the robot should never enter (at least the robot radius). Figure 7.4 illustrates the resulting traversal costs with and without our approach.

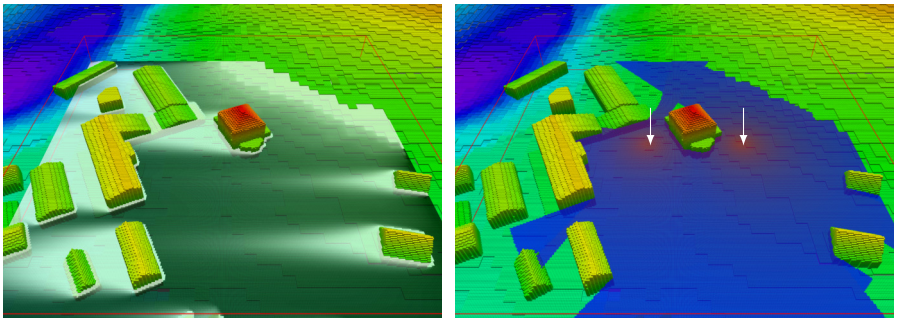


Figure 7.5: Costmaps for wind and signal strength. Left: We employ a simplified wind model to estimate the magnitude of wind. Shown is a slice of the 3D magnitude model overlaid on our obstacle map. Darker areas depict stronger magnitude. Right: Estimated signal strength from two WiFi access points. Close to the access points (arrows) the signal is strong (red) and becomes weaker (blue) until a threshold. Obstacles block the signal.

Wind Model

We model the strength of constant wind in the vicinity of structures. For this, we use a very simplified model neglecting the turbulent behavior of wind very close to structures, and we make the assumption that the wind is constant in direction and magnitude for the time of at least one planning and execution cycle. Whereas the simplifying assumptions do not allow to accurately estimate the required control effort, it still is possible to employ less wind affected areas behind larger structures. Furthermore, safety margins in more windy areas could be increased.

We discretize the wind field into uniform voxels. The grid is oriented such that the wind direction coincides with the grid x-axis. All voxels with the lowest x index not inside of obstacles are initialized with the wind magnitude in free space, e.g., measured by a wind estimator on the control layer. While sweeping a filter plane along the x-axis, the wind magnitude of voxels inside of obstacles is set to zero. The magnitude of wind in other voxels is determined by applying a 2D filter kernel averaging wind magnitudes from the previous voxel plane. Figure 7.5 illustrates a slice through a 3D wind map.

Communication Quality Model

For some missions the communication with a control station is important. We address this by modeling the signal strength from, e.g., WiFi access points, in a 3D map. Figure 7.5 shows an example with two WiFi access points. For each voxel in the cost grid, a line-of-sight check to each signal source is performed. This is achieved by raycast-

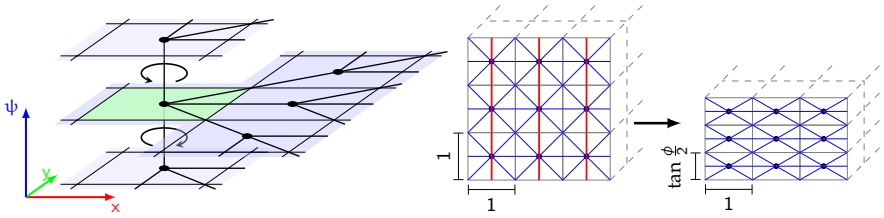


Figure 7.6: Modified grid for FoV-aware planning. Left: Outgoing edges from a grid cell (green) when planning with horizontal visibility constraints. Depicted are the planar position and the orientation ψ of an MAV. The z-dimension is omitted for clarity. Right: To allow only transitions within the vertical FoV ϕ of a sensor, we remove the vertical edges (red) and use anisotropic grid cells.

ing in the obstacle OctoMap. If no line-of-sight to any signal source exists, the signal strength s is zero. Otherwise, it is the maximum of

$$s = \underset{i}{\operatorname{argmax}} \left((s_{max} - s_{min}) - 20 \log_{10} \frac{r_i}{r_{min}} \right),$$

for all signal sources i with maximum signal strength s_{max} ($-30\,\mathrm{dBm}$), threshold s_{min} under which the signal strength is considered too low for communication ($-70\,\mathrm{dBm}$), the distance r_i to the signal source and radius r_{min} where the signal has the assumed maximum strength.

7.1.3 *Path Planning in Sensor Field-of-View*

To increase the safe flight speed, a faster perception of the environment for localization and obstacle avoidance is inevitable. Modern lightweight 3D laser scanners as the Velodyne Puck LITE acquire 3D point clouds of the environment at a high frequency. Nevertheless, this comes at a price: The new laser scanner setup employed on the Matrice 600 MAV does only cover a vertical FoV of 30° in contrast to the 180° of the slower omnidirectional laser scanner employed on the other MAV platforms. This raises the demand for paths that let the MAV only move within the FoV of the scanner to reliably detect obstacles that impose a collision hazard. Similar considerations have to be taken into account when using camera or radar sensors that are typically restricted not only in the vertical FoV but also in the horizontal FoV. Consequently, these sensors furthermore require that the MAV is only flying into forward direction of the sensor.

One option would be to define motion patterns for ascent and descent that ensure the perception of the flight path and to plan at fixed altitudes in-between. Nevertheless, this yields far from optimal flight paths. Thus, our approach extends the planning specification by adding these additional visibility constraints. Paths are safe, if they do not only stay in a safe distance to known obstacles but also if

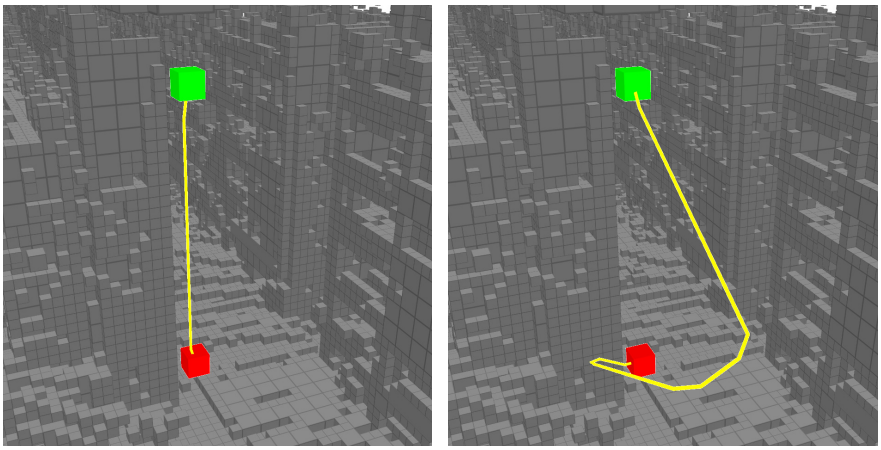


Figure 7.7: Planning under visibility constraints. Left: Without visibility constraints the shortest path (yellow) from a start (green) to a target position (red) below solely descents in place. Right: With visibility constraints the MAV has to move within the field of view of the LRF and consequently follows a longer descent path with a vertical angle of 15°.

unknown obstacles can be reliably perceived by obstacle avoidance sensors. The ascent and descent angles of planned paths should stay within the vertical FoV of our obstacle sensor.

FoV-aware Planning Representation

To plan an obstacle-free path incorporating the visibility constraints by graph-search—which is complete and optimal w.r.t. the planning specification—requires a modification of the underlying grid-based graph structure. From the 26 edges connecting nodes centered in the voxels of the grid with their Moore neighborhood, we remove the two edges connecting voxels directly above or below the current voxel to prohibit ascent and descent movements in place. For the case of a vertical apex angle ϕ of the obstacle sensors smaller than 90° the angular resolution of the remaining uniform voxel grid of 45° is still too coarse to represent the allowed maximum ascent and descent angles of $\frac{\phi}{2}$. In our case the apex angle of the LRF is 30° requiring an angular resolution of 15°. To increase the angular resolution, we employ an anisotropic voxel grid with horizontal edge lengths of v_{xy} and a voxel height of $v_z = \tan(\frac{\phi}{2})v_{xy}$. A schematic of the anisotropic grid is depicted in Figure 7.6. The resulting graph structure enforces restricted ascents/descents within the FoV of the sensors. Figure 7.7 illustrates the resulting plans with and without consideration of visibility constraints.

To penalize frequent changes in the flight direction and to facilitate forward flight for sensors with this requirement, we introduce the direction of flight as an additional planning dimension. Without this modification, a zigzag motion to ascent or descent would be equal to

larger straight glide paths in path costs, but would significantly slow down the MAV due to numerous stops to change direction. The direction of flight is discretized to the eight possible transitions in the plane, angles of up to 45° are not penalized. We remove edges yielding larger changes in direction, thus, these transitions are still possible, but at the cost of multiple intermediate transitions. Figure 7.6 depicts the modified planning grid.

The MAV orientation in the planned path depends on the actual sensor constraints. If paths for MAVs with front-facing sensors, e.g., cameras, are planned, the MAV orientation is coupled to the flight direction dimension. Thus, the MAV yaw angle is linearly interpolated along plan edges such that the angle between the front of the MAV and the current flight direction is at most 45° and arrives at a difference of 0° when the next waypoint is reached. For sensors with a horizontal FoV of at least 90° the full path segment between waypoints is guaranteed to be visible. Sensors with a smaller FoV require rotating the MAV in place until the path segment is in the sensor FoV before starting with the position interpolation. In omnidirectional mode, e.g., for laser sensors, the yaw orientation of the MAV can be freely set to mission requirements and the flight direction dimension is solely restricting sudden direction changes.

To speed up the node expansions without the requirement to process and store the whole graph structure in advance, our planner employs look up tables (LUTs) for edge costs and possible angular transitions. Furthermore, obstacle costs per vertex are cached in a lower dimension grid until they are invalidated by map updates.

FoV-aware Heuristic

As the Euclidean distance heuristic strongly underestimates altitude changes, we employ a modified heuristic better suited for our planning structure. Figure 7.8 illustrates the idea. For a node position p_n and a target position p_t, we define the heuristic $h(p_n, p_t) = h(d)$ on the position difference d as

$$h(d) = \sqrt{d_x^2 + d_y^2 + z_e^2} + z_z, \tag{7.1}$$

$$z_e = \min\left(|d_z|, \tan\frac{\phi}{2}\sqrt{d_x^2 + d_y^2}\right), \tag{7.2}$$

$$z_z = \frac{\max\left(0, |d_z| - z_e\right)}{v_z}\sqrt{v_{xy}^2 + v_z^2}, \tag{7.3}$$

where z_e is the slope-restricted Euclidean altitude change that is possible over a distance $\sqrt{d_x^2 + d_y^2}$ with maximum angle $\phi/2$; z_z is the shortest possible detour to overcome the remaining altitude difference.

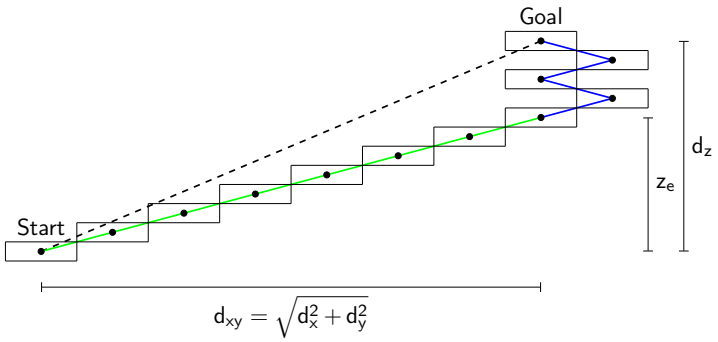


Figure 7.8: FoV-aware heuristic. The Euclidean distance (dashed line) underestimates the path length in our planning representation. We split the heuristic into two parts. First, a Euclidean part (green) to the closest point to the goal the MAV can reach on a straight line given the sensor constraints. And Second, an estimate for the shortest possible path for the remaining vertical distance to the goal $|d_z| - z_z$ (blue).

2

Corollary 1. *The heuristic* $h(.)$ *is an admissible heuristic for A* search in the visibility constrained graph structure.*

Proof. In the first case

$$|d_z| \leqslant \tan\left(\frac{\phi}{2}\right)\sqrt{d_x^2 + d_y^2}$$

follows that $z_e = |d_z|$ and Equation (7.3) vanishes. The remaining heuristic term

$$h(d) = \sqrt{d_x^2 + d_y^2 + d_z^2} = \|d\|_2$$

is then the Euclidean distance which is an admissible heuristic.

In the other case, z_e is the maximum altitude change that is possible with the allowed ascent angle, i.e., $p_n + (d_x, d_y, z_e)$ is the closest point to p_t that can be reached on a straight line from p_n without violating the angular constraint. All points closer to the target p_t in z would increase the distance in the x-y-plane with factor $1/\tan(\phi/2)$ which is > 1 following the assumption that $\phi \leqslant 90°$ for this case and Equation (7.2). The remainder $z_r = |d_z| - z_e$ can only be eliminated by an ascent through $\frac{z_r}{v_z}$ voxels with edge length $\sqrt{v_{xy}^2 + v_z^2}$ each resulting in the value of z_z which has to be added to the distance to the closest point. Thus, no shorter path exists and the heuristic is admissible in both cases. □

Figure 7.9 shows the difference in node expansions with and without our modified heuristic for the path depicted in Figure 7.7.

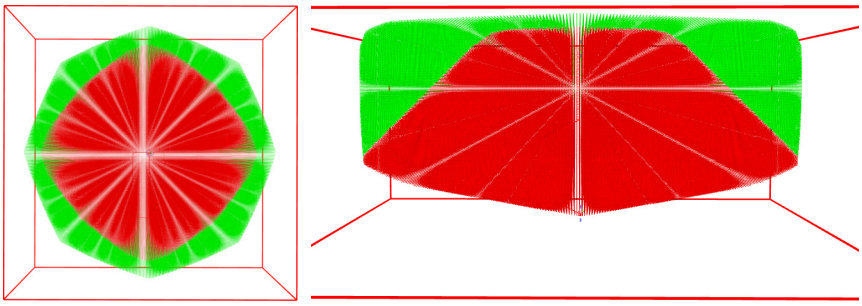
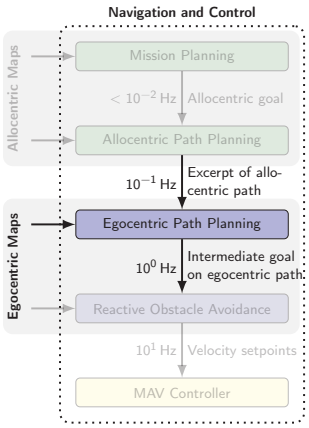


Figure 7.9: Visibility constraint planning heuristic. A standard Euclidean distance heuristic strongly underestimates the cost of altitude changes in the grid. This results in more unnecessary node expansions (green). Our modified heuristic expands fewer nodes (red). Left: Top-view. Right: Side-view. Red lines depict the planning volume.

7.2 LOCAL PATH PLANNING



Below the allocentric planning layer our hierarchy contains the egocentric or local path planner. The local planner is based on the same techniques as described in the global path planning section. Hence, we will only focus on the differences in this section.

In contrast to the global path planner, the local path planner operates on a restricted view on the environment based on local sensor perceptions. A path planned on this layer is also expressed in robot-centric coordinates with the MAV in the center of the planning representation. The purpose of this layer is mainly to quickly react on obstacles not represented in the allocentric representation of the environment.

We make the optimistic assumption that most obstacles not known in advance can be surrounded locally. Thus, at the time they are perceived a local modification of the globally planned path is often sufficient. While the local obstacle avoidance layer—detailed in Chapter 9—below the local path planner operates on a comparable environment representation, and can avoid small obstacles without the need for replanning, its abilities are limited.

To close the gap between slow allocentric planning and fast reactive obstacle avoidance, we add an intermediate local planning layer. This allows the global planner to replan on a more long-term time scale, while preventing the obstacle avoidance to get stuck in a local minimum. Whereas the assumption of local surroundability might not always hold—e.g., in maze-like environments—this layer still facilitates quick solutions in situations that can be resolved locally without waiting for another allocentric planning cycle. Figure 7.10 shows an example where reactive obstacle avoidance without replanning would fail to surround an obstacle, but the local planner can find a valid plan.

The local planner runs at a frequency in the order of 2 Hz. This is equal to the rate at which the rotating laser scanner of many of our

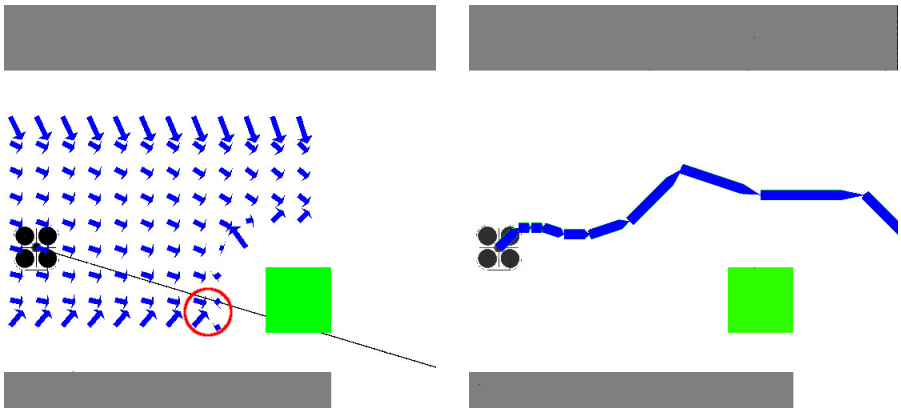


Figure 7.10: Avoidance of unknown obstacles. To surround a priori unknown obstacles (green rectangle in this example), we employ an egocentric path planner. Left: Artificial potential forces (arrows) push the MAV away from obstacles, but the MAV will get stuck in a local minimum (red circle) while approaching a waypoint in the direction of the black line. Right: Frequent egocentric replanning resolves this local minimum in the next planning cycle.

MAVs acquire a new 3D point cloud of the close environment. The spatial planning horizon is restricted to the measurement range of the laser sensor This is sufficient as no local planner deviations from the allocentric plan are possible beyond this horizon. The minimum possible planning horizon would be the distance the MAV can travel within one planning cycle. Scans are aggregated into a grid-based local multiresolution obstacle map (Droeschel et al., 2014b) with an edge length of the map of 40 m. To avoid obstacles not locally perceived but represented in the allocentric map, it is possible to add the occupied parts of the allocentric map inside of the local planning volume to the egocentric map.

Fast replanning with low latencies is performed by employing 3D local multiresolution path planning, detailed in Section 2.3. This efficient planning technique allows for frequent replanning, which makes 3D navigation in dynamic, unpredictable environments possible. Its planning representation—a grid-based robot-centric obstacle map with higher resolution in the center and decreasing resolution with increasing distance from the MAV position—resembles the onboard sensor characteristics with distance dependent precision and the uncertainty in the motion execution of the MAV by external influences, e.g., gusts of wind.

A typical approach to couple global and local planners is to use a waypoint on a, potentially simplified, allocentric plan as the goal for the local planner, e.g., the next point where the path makes a turn. As the global path is already cost-optimal with respect to the allocentric map, we want the local plan to follow the global path as close as possible. Furthermore, waypoints on the allocentric plan that are not

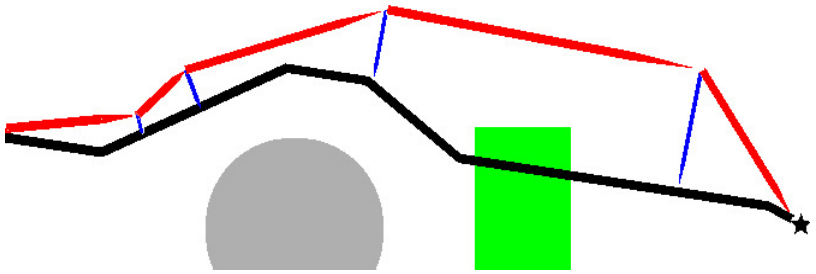


Figure 7.11: Coupling of local plan to allocentric plan. The local plan (red) is coupled with the allocentric plan (black) by a cost term that penalizes deviations from the allocentric plan. The blue lines depict the deviation vectors at example points, the star is the planner goal. The gray circular obstacle is in the allocentric map, the green rectangular obstacle has to be surrounded based on the local map.

mission critical can be blocked by locally perceived obstacles. Thus, enforcing that these waypoints have to be reached or, otherwise if that's not possible, wait for a corrected allocentric plan is disadvantageous. Consequently, it is not sufficient to send the next waypoint of the global path to the local planning layer as goal pose.

Instead, the input to the local planner is the complete global plan. Hence, the path costs of the global path are a lower bound to path costs for refined plans, based on newly acquired sensor information—mostly dynamic and static previously unknown obstacles—and a local path deviating from the global plan cannot be shorter in terms of path costs. Locally shorter plans on layers below the allocentric path planner with a local view on the map may yield globally suboptimal paths. Thus, we add the estimated path costs between waypoints of the global plan to its edges to facilitate efficient exploration of the search space on the local path planning layer. Also, mission goals are marked as the local planner has to reach these exactly. If this is not possible, the mission planning layer has to resolve this failure condition.

In every planning cycle, the local obstacle map obtained from the onboard sensors is merged with a local excerpt of the allocentric world model given the current MAV pose estimate. Distances to all obstacles are calculated for estimating their radius in later processing steps. We use lazy evaluation of the obstacle costs in the planning grid during the node expansions of the planner.

The local planner is initialized with a local goal on the global path. This goal is the intersection of the global plan with the local planning volume. If the global plan contains mission-relevant waypoints within the local planning volume, the closest of these is taken as the local goal.

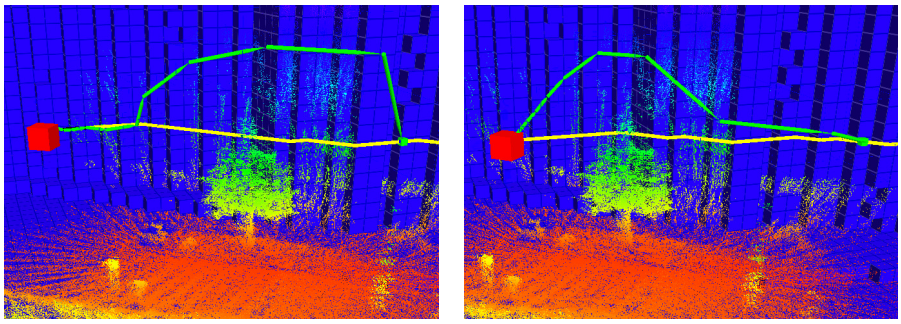


Figure 7.12: Surrounding obstacles by local planning. While following a planned allocentric path (yellow) based on the allocentric environment model (blue), a tree is perceived with local sensing (colored measurements). The local multiresolution path planner finds a local plan around the obstacle without global replanning (green). Left: When the MAV (red box) is still farther away, the local plan around the obstacle is coarse due to the multiresolution planning. Right: As the MAV approaches the obstacle, the plan gets finer. Behind the obstacle it returns to the global plan.

To bind the local plan to the globally consistent path we add a proximity term to the local planner cost function. The cost function for a cell with center c incorporates the sum of all obstacle costs h_o according to Equation (2.1) and the distance of the cell's center to the closest path segment s of the global plan G:

$$\text{cost}(c) = \sum_{o \in \text{Obst.}} h_o(\text{dist}(c, o)) + \gamma \operatorname*{argmin}_{s \in G} \text{dist}(c, s).$$

The parameter γ controls how tightly the local plan is bound to the global plan. Figure 7.11 illustrates the binding of the local to the global path.

Similar to the global planner, we embed an undirected graph into this grid and perform A* graph search (see Section 2.4) from the center of the MAV-centered grid to the goal. As the MAV is at the grid center between cells, we add additional edges from the center to the eight adjacent cells. The costs of expanding an edge e from a grid cell c_{from} to a cell c_{to} are the cell costs multiplied by the lengths of the edge within the respective cells:

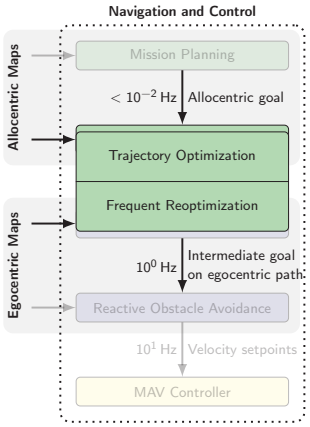
$$\text{cost}(e) = \|e_{\text{from}}\| \text{cost}(c_{\text{from}}) + \|e_{\text{to}}\| \text{cost}(c_{\text{to}}).$$

The edge length $\|e_c\|$ within a cell is the Euclidean distance from the cell center to the intersection point between the cells.

Figure 7.12 shows a local deviation from a global plan without global replanning. It can be seen that the local plan returns to the allocentric plan after locally surrounding the obstacle. The provided trajectories serve as input for the local collision avoidance layer, that we will detail in Chapter 9.

8

TRAJECTORY OPTIMIZATION



In contrast to fixed-wing micro aerial vehicles (MAVs), multirotors are capable of flying omnidirectionally and can stop or change direction within a short time horizon. Hence, when flying with low velocity, the dynamics of a multirotor MAV can be neglected on higher planning layers. When aiming at shorter mission execution times, higher average velocities are desirable, though. To achieve these, we extend our path planning by optimizing flight trajectories with a simple MAV dynamic model.

Continuous perception of the environment and tracking the MAV state is a prerequisite for safe operation. To react on the perceived changes timely, frequent adaptation of the planned flight trajectory is inevitable. When aiming at fast trajectory execution, the flight dynamics have to be taken into account in addition to spatial constraints. To reduce the complexity of this kinodynamic planning problem, we follow a multistage approach from a 3D spatial planner to dynamic 12D trajectory optimization. First, we plan a coarse obstacle-free 3D path, second, we process this plan to fill missing dimensions (i.e., yaw rotation, velocities) with an initial guess as close as possible to the expected final result, and third, we optimize this initialization trajectory w.r.t. control and obstacle costs to obtain a smooth trajectory that can be followed with low control effort. To obtain smooth collision-free trajectories, we use the gradient-based trajectory optimizer CHOMP (covariant Hamiltonian optimization and motion planning) from Zucker et al. (2013).

In this chapter, we employ a static environment model similar to our allocentric path planner, detailed in Section 7.1. Furthermore, we relax the static world assumption in Section 8.4 by reoptimizing the trajectory when the environment changes.

8.1 PROBLEM FORMULATION

The static state of an MAV is a 6-tuple of a 3D position $p = (x, y, z)$ and a 3D orientation $r = (\mathrm{roll}, \mathrm{pitch}, \mathrm{yaw})$. Although in general poses of the MAV are six dimensional, for multirotors only four dimensions can be controlled independently. The roll and pitch angles directly influence the horizontal acceleration of multirotors. Thus, our start and goal poses are 4D tuples (x, y, z, θ) with a 3D position and yaw-rotation θ. Here, we assume that velocity and acceleration at start and goal pose are zero, even though this assumption can be relaxed.

To generate smooth trajectories for our MAV, we need poses and velocities as input for the underlying model predictive controller (MPC) (Beul and Behnke, 2017). Input to the controller is an 8D trajectory containing 4D poses and the corresponding velocities. We achieve this by formulating trajectory planning as an optimization problem. For accurate trajectory following, we have to optimize the trajectory for low acceleration control costs. Consequently, outputs of our trajectory optimization step are time-discretized 12D trajectories with 4D poses, velocities, and accelerations without discontinuities. Accordingly, the goal is to find a trajectory, which minimizes the costs calculated by a predefined cost function.

As an input, the trajectory optimizer gets a start and a goal configuration $x_0 = (p_0^x, p_0^y, p_0^z, \theta_0)^\top, x_N = (p_N^x, p_N^y, p_N^z, \theta_N)^\top \in \mathbb{R}^4$. The output of the algorithm is a trajectory $\Theta \in \mathbb{R}^{4\times(N+1)}$ consisting of one trajectory vector $\Theta^d = (x_0^d, \dots, x_N^d)^\top \in \mathbb{R}^{N+1}$ per dimension d, discretized into $N+1$ waypoints. To be directly executable by the MAV controller, in general the trajectory points need to have a fixed duration $\Delta t = 10\,ms$. We relax this assumption in Section 8.4. Besides a cost function, the trajectory optimizer has to be initialized with an initial trajectory, e.g., an interpolation between start and goal configuration.

The optimization problem we solve iteratively is defined as

$$\min_{\Theta} \left[\sum_{i=0}^{N} q(\Theta_i) + \sum_{d} \frac{1}{2} {\Theta^d}^\top \mathbf{R} \Theta^d \right].$$

The state costs—obstacle costs, velocity, and visibility constraints—are calculated by a predefined cost function $q(\Theta_i)$ for each state in Θ. ${\Theta^d}^\top \mathbf{R} \Theta^d$ is the sum of control costs along the trajectory in dimension d with $\mathbf{R}$ being a matrix representing control costs. The trajectory optimizer now attempts to solve the defined optimization problem by means of the gradient-based optimization method CHOMP (Zucker et al., 2013).

If a gradient for the used cost function cannot be computed, an alternative is to optimize the trajectory w.r.t. to the cost function $q(\tilde{\Theta}_i)$ with $\tilde{\Theta} = \mathcal{N}(\Theta, \Sigma)$ being a noisy state parameter vector with mean Θ and covariance Σ by means of a stochastic optimizer (Kalakrishnan et al., 2011).

The cost function $q(\Theta_i)$ is a weighted sum of I) piecewise linear increasing costs c_o induced by the proximity to obstacles, II) squared costs c_a caused by acceleration limits, III) squared costs c_v caused by velocity constraints, and IV) costs from violating visibility constraints. The obstacle costs c_o increase linear with a slope o_{far} from a maximum to a minimum safety distance. From the minimum safety distance to the obstacle, the costs increase with a steeper slope o_{close} to allow for gradient computation in the vicinity of obstacles.

Velocities and accelerations as derivatives of the state are implicitly modeled by the duration between discretization steps.

To efficiently obtain obstacle costs during optimization, we calculate a distance field (Kalakrishnan and Anderson, 2009) with a 20 cm resolution from our static environment model. We propagate distances up to a maximum distance where obstacle costs are zero. Our distance-dependent obstacle costs are modeled as a piecewise linear function, similar to the basic cost model in Section 7.1, with decreasing costs up to a maximum distance from obstacles of 4 m.

8.2 INITIALIZATION

Initialization of the optimizer with a control cost-optimal interpolation between start and goal configuration leads to trajectories that converge to a non collision-free local minimum in general. Furthermore, the trajectory optimization converges faster when the initialization is close to the (locally) optimal trajectory. Hence, a good initialization of the trajectory is necessary to facilitate quick convergence to a valid, collision-free optimum.

Thus, to find valid paths between start and goal configuration, we plan collision-free paths employing an A* path planner (see Section 7.1) in a low-dimensional subspace of only the 3D translational part of the trajectory. We simplify the resulting plans to reduce discretization effects. Although, these coarse plans prevent an optimizer to get stuck in local minima that yield infeasible trajectories, they are far from smooth and lack velocity and acceleration dimensions. We mitigate the influence of the suboptimal initialization by approximating optimal velocities and accelerations with a simple MAV dynamics model and by smoothing the input trajectory to a trajectory requiring less control effort. Even though the optimal solution is naturally not known in advance, we can make some assumptions about the MAV dynamics that reduce the convergence time and avoid infeasible local minimum trajectories. With these fast preprocessing steps, we achieve a combined planning and optimization time of approximately 1 s for allocentric trajectory generation. Figure 8.1 shows resulting trajectories from each of the processing steps, i.e., a planned path, a spline interpolation of this path with motion model-based timings, and the resulting trajectory after 500 optimizer iterations.

As our MAV is approximately rotational symmetric, we omit planning for the heading and initialize the yaw trajectory with a control cost-optimal interpolation.

Transition Between Path Segments

The translational part of the planned flight paths is a sequence of straight line segments connecting start and goal position. By em-

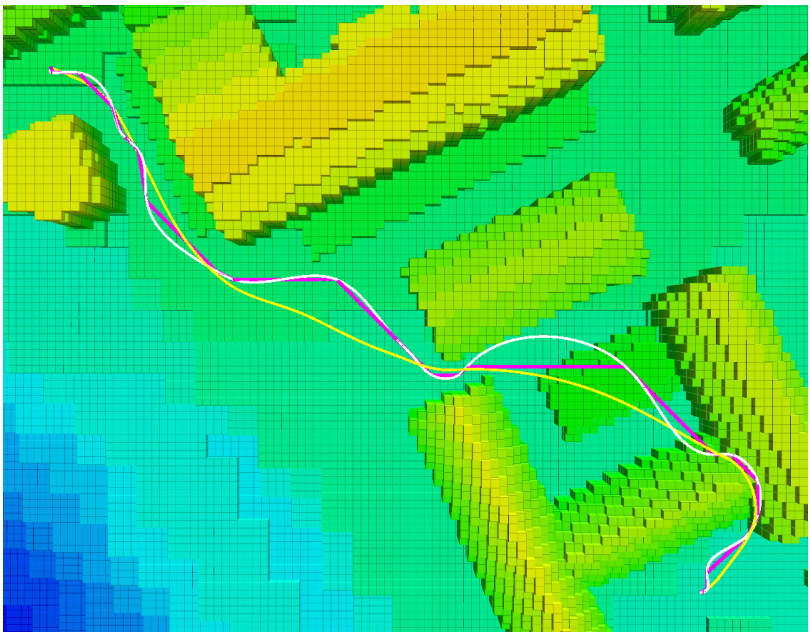


Figure 8.1: Exact planning of smooth trajectories with velocity and acceleration dimensions for MAVs is often prohibitively slow for frequent replanning. We plan coarse 3D trajectories (pink), incorporate simple assumptions about the MAV dynamics (white), and optimize this initial guess quickly to obtain the least cost kinodynamic trajectory (yellow) w.r.t. distance to obstacles and control costs.

ploying a grid-based planner, the transition between consecutive segments is restricted to a few discrete angles. After simplification of the initial plan, the discretization effects are mitigated, but still present, resulting in discontinuities in its derivatives—most importantly velocity and acceleration. These discontinuities cause gradients of large magnitude during optimization. Thus, a lot of optimization effort is spent on smoothing these transitions instead of optimizing the continuous parts of the trajectory.

A first solution is to employ cubic spline interpolation to acquire smoother trajectories. The sampling points are the endpoints of the segments from the simplified planned path at the transition times estimated by our simple motion model, detailed later in this chapter. Splines mitigate the discretization effects leading to lower accelerations. Figure 8.2 shows the necessary accelerations to follow the planned and spline-based trajectories. Nevertheless, splines can overshoot and cause collisions without further processing. Furthermore, the trajectories tend to oscillate. The derivatives at the sampling points are omitted. We calculate velocities and accelerations along the trajectory after the spline interpolation. Even though the spline-based trajectories are smooth, acceleration and velocity limits can still be violated without further optimization by non-optimal timings and large curvature necessary to pass the planned waypoints.

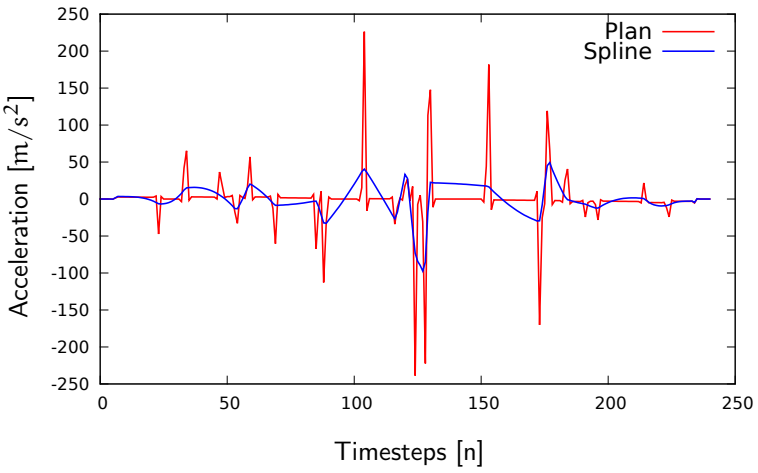


Figure 8.2: Accelerations of the trajectory before optimization. Initialization with the path from a grid-based planner yields high accelerations at the connection points between plan intervals. Spline interpolation of the path reduces these spikes yielding faster convergence.

Continuous Curvature Transition Segments

We address the aforementioned issues with spline interpolation by introducing connections with continuous curvature between straight segments. These continuous curvature transition segments (CCTSs) mitigate the influence of suboptimal initialization by modifying the planned path locally. To assure that the modified path remains collision-free without costly exact planning of the connecting segments, we calculate spheres containing solely free space around the segment transitions. The radius of a sphere around the transition between path segment s_{i-1} to s_i is calculated as

$$r_i = \min\left(\frac{1}{2}\|s_{i-1}\|, \frac{1}{2}\|s_i\|, d_o(i)\right),$$

where $d_o(i)$ is the distance to the nearest obstacle. The start and end points of the connecting segment are the intersection points of the planned path with the sphere. We construct the connecting segment in a planar subspace defined by start point, end point, and transition point. In the following, we will use 2D coordinates on this plane.

As connecting segments, we employ clothoids—also often referred to as Euler spirals. Clothoids are curves with linear increasing curvature $\kappa(l) = l\sigma$ along the curve. The Cartesian coordinates are given by

$$x(l) = \sqrt{\frac{\pi}{\sigma}} C\left(\sqrt{\frac{l^2\sigma}{\pi}}\right), \qquad y(l) = \sqrt{\frac{\pi}{\sigma}} S\left(\sqrt{\frac{l^2\sigma}{\pi}}\right),$$

with $C(x) = \int_0^x \cos(\frac{\pi}{2}t^2)dt$ and $S(x) = \int_0^x \sin(\frac{\pi}{2}t^2)dt$ and sharpness σ.

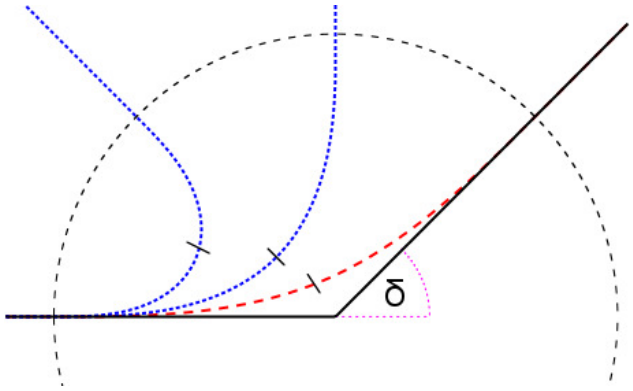


Figure 8.3: Construction of transition segments. Transition segments (red dashed line) connect two straight line path segments (black solid lines) with continuous curvature inside a bounding sphere restricted by the distance to obstacles and other waypoints. Their shape is defined by heading change δ (violet dotted lines). The initially normalized transition segments are scaled by the radius of the bounding sphere. The segments are constructed by two clothoids mirrored in the center (short black lines). Transitions for sharper heading changes of 90° and 135° are depicted by blue dotted lines.

To calculate the sharpness parameter for our clothoid segment, we follow ideas from Fraichard and Scheuer (2004). The parameter σ for a normalized clothoid depends on the change in heading between start and end of the clothoid segment. Due to the construction of the start and end points of our connecting segment, the curvature is zero and the tangents of the connecting curve and the circle are orthogonal. Without loss of generality, we assume that the heading and the position at the beginning of the segment are zero. For heading δ after the transition follows

$$\sigma(\delta) = \frac{\pi \left[\cos\frac{\delta}{2} C\left(\sqrt{\frac{\delta}{\pi}}\right) + \sin\frac{\delta}{2} S\left(\sqrt{\frac{\delta}{\pi}}\right)\right]^2}{\sin^2\left(\frac{\delta}{2}\right)}$$

and a length $L(\delta) = 2\sqrt{\frac{\delta}{\sigma}}$ of the clothoid segment. We can construct the connecting segment from two clothoid segments, one starting with curvature zero at the start point to an intermediate point with heading $\frac{\delta}{2}$ and a segment mirrored at the end point of the first segment. Figure 8.3 illustrates the segment construction. Finally, the resulting normalized connecting segment is scaled and projected back into the 3D space. To simplify the calculations, we do not restrict the maximum curvature. This ensures that a continuous solution can always be found even if locally not dynamically feasible. We leave finding a globally feasible solution to the subsequent optimization stage.

To allow for arbitrary spaced sampling, we fit cubic splines to the resulting path consisting of straight line segments and clothoid transition segments. In contrast to fitting splines to the initial path containing only straight line segments, we achieve a much closer approximation of the path. Thus, the spline-based path is always collision-free if the planned path is collision-free.

Model-based Initialization

The planned path is a timingless list of 4D (x, y, z, yaw) spatial waypoints. After initialization of the trajectory optimizer with the planned path, which is optimal given the plan discretization and dimensionality, we need to re-parameterize the planned path from a discrete-space to a discrete-time representation to match the fixed-duration timesteps of the parameter vector. We rediscretize the path according to a simple analytical motion model with bounded acceleration to a 10 Hz time resolution. To get an easy to compute closed-form solution for our discretization, we assume that the MAV starts with a maximum acceleration $a(0) = a_{max}$, stops with a maximum deceleration $a(T) = -a_{max}$ at the end of the trajectory, and a linear transition between these states. With an estimated flight duration of T for the whole trajectory, we can derive a simple motion model of the MAV for acceleration $a(t)$, velocity $v(t)$, and position $x(t)$

$$a(t) = -2\frac{a_{max}}{T}t + a_{max}, \tag{8.1}$$

$$\int a(t)dt = v(t) = -\frac{a_{max}}{T}t^2 + a_{max}t, \tag{8.2}$$

$$\iint a(t)dt = x(t) = -\frac{a_{max}}{3T}t^3 + \frac{1}{2}a_{max}t^2, \tag{8.3}$$

at time $t \in [0, T]$.

With $x(T) = L$, given a total length L of the planned path, and Equation (8.3) we can calculate the estimated flight duration T as

$$L = -\frac{a_{max}}{3T}T^3 + \frac{1}{2}a_{max}T^2 \quad \Rightarrow \quad T = \sqrt{\frac{6L}{a_{max}}}.$$

With $T = (N+1)\Delta t$, we get the necessary number of time steps N for our trajectory discretization.

A uniform discretization of the planned path into these N timesteps can serve as an input to the optimizer. Such a discretization would assume a constant velocity and, thus, zero acceleration along the trajectory. These derivatives are far from optimal. Hence, a large amount of optimization effort is spent on optimizing the timing of the trajectory.

The position $x(t) \in [0, L]$ is the part of the planned path that has been traversed until time t. This can be used to rediscretize the path with a better initial guess about velocities and accelerations that reduce the control costs over the complete trajectory. Instead of equal-length path segments per time step, we set the discrete time steps

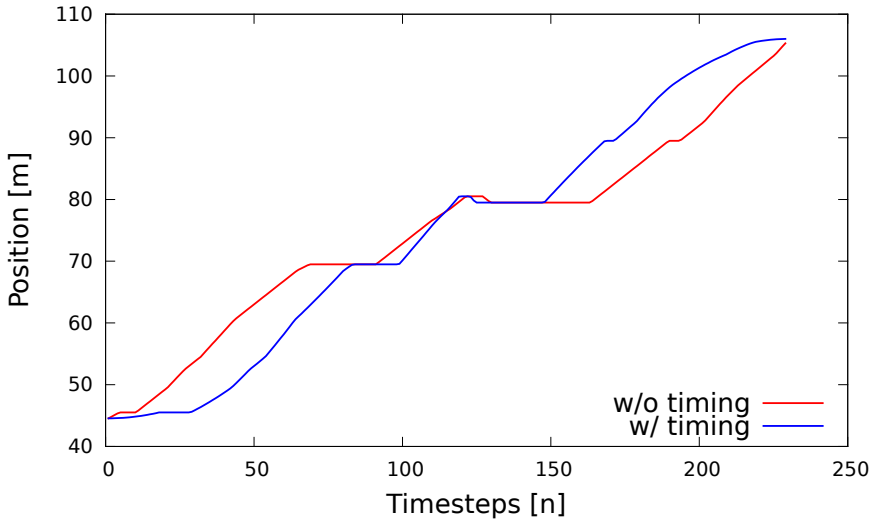


Figure 8.4: Uniform plan discretization (red) vs. discretization according to motion model (blue) of the MAV trajectory in the x-coordinate. Discretizing according to a motion model facilitates faster convergence of the trajectory optimizer.

$t = i\Delta t$ along the path according to $x(t)$. Figure 8.4 shows the effect of the timing-based discretization on the initial parameter vector. The velocities at the start and goal are lower and the velocity in the middle of the trajectory is higher.

8.3 FOV-AWARE TRAJECTORY OPTIMIZATION

Similar to the planning approaches described in Chapter 7, an important objective is to perceive obstacles with the MAV onboard sensors. While we can employ a path planner to plan paths in the field of view (FoV) of the sensors in the initialization step, we have to also modify the trajectory optimization objective function to preserve the sensor coverage property after optimization. The trajectory ascent and descent angles should stay within the vertical FoV of our obstacle sensor. The resulting trajectories contain velocity and acceleration information employed by our low-level controller to accurately follow the intended paths.

To enforce the visibility constraints, we look at the local path triangles defined by a segment between two trajectory points Θ_{i-1}, Θ_i and its projection to the x-y-plane $\Theta_{i-1,xy}$, $\Theta_{i,xy}$. Let Θ_{i-1} and Θ_i be two consecutive trajectory points in Θ^d. Then the visibility constraint for a sensor apex angle of ϕ is defined as

$$\left|atan2\left(\Theta_{i,z} - \Theta_{i-1,z}, \left\|\Theta_{i,xy} - \Theta_{i-1,xy}\right\|\right)\right| \leqslant \frac{\phi}{2}.$$

If this constraint is violated, we locally modify the trajectory points to flatten the path triangle. Simultaneously, we reduce the altitude

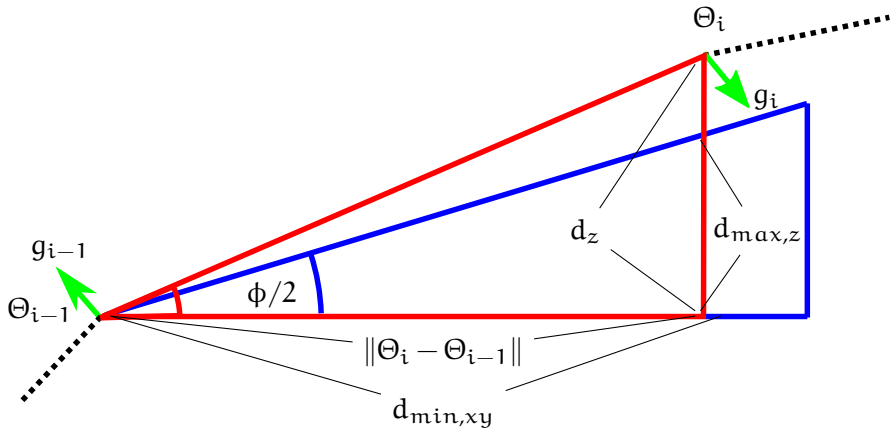


Figure 8.5: Gradients for FoV-aware optimization. If the ascent (or descent) angle between two consecutive waypoints Θ_{i-1} and Θ_i is out of the sensor FoV ϕ, this case is depicted by the red triangle, then the altitude change d_z is modified such that the constraint violation $d_z - d_{max,z}$ is reduced to half. The remaining violation is mitigated by stretching the planar projection of the movement $\Theta_{i,xy} - \Theta_{i-1,xy}$ to reduce the difference to $d_{min,xy}$. Thus, the trajectory becomes reshaped towards the blue triangle by the gradients g_{i-1}, g_i.

difference d_z and stretch the movement in the x-y-plane, depicted in Figure 8.5. The partial gradients g_{i-1} and g_i to modify the trajectory points are defined as

$$
\begin{aligned}
d_{min,xy} &= \frac{|d_z| - d_{max,z}}{\tan(\phi/2)} - \left\|\Theta_{i,xy} - \Theta_{i-1,xy}\right\|, \\
g_{i-1,x} &= w_v \cos(\alpha) \frac{d_{min,xy}}{2}, \\
g_{i-1,y} &= w_v \sin(\alpha) \frac{d_{min,xy}}{2}, \\
g_{i-1,z} &= w_v \operatorname{sgn}(-d_z) \frac{|d_z| - d_{max,z}}{4}, \\
g_i &= -g_{i-1},
\end{aligned}
$$

where w_v is a weighting factor and α is the direction angle of the path segment projected to the x-y-plane. $d_{min,xy}$ denotes the minimum planar distance to reach the angular constraint with a given d_z and $d_{max,z}$ is the maximum allowed distance in z to reach the constraint with a given planar distance. Thus, half of the constraint violation is distributed to the altitude gradients and the other half is used to elongate the path. As a result, the optimized paths can lunge out to reduce the ascent/descent angles. Figure 8.6 shows the resulting trajectory for an ascent in place and Figure 8.7 shows resulting trajectories with and without visibility constraints after the optimization step in an outdoor map. To reach the target position close to a

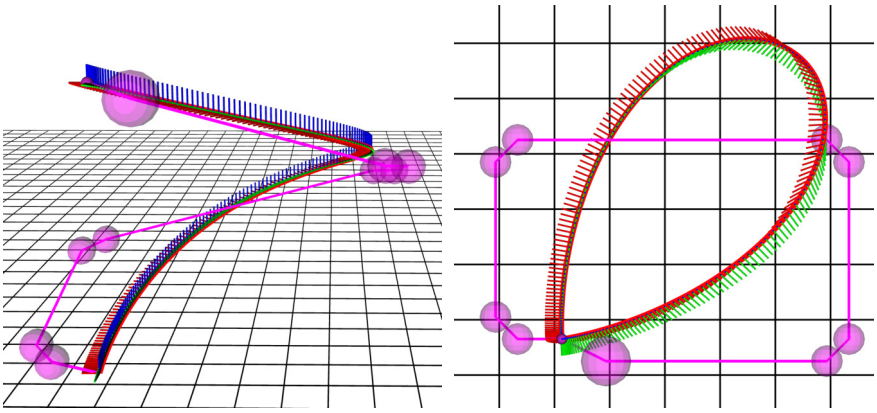


Figure 8.6: Optimized trajectory for an ascent in place. We initialize the trajectory optimization with a planned path (purple) with transition segments (purple spheres). The result after optimization yields a smooth spiral (colored axes). Left: Perspective. Right: Top-down ortho projection.

building our approach generates a spiraling descent. Please note that the sensor visibility constraint is satisfied along the whole trajectory if it is satisfied in the discrete trajectory points by construction.

8.4 FREQUENT REOPTIMIZATION

To react on deviations from the planned trajectory and to avoid obstacles perceived with onboard sensors the trajectory is continuously re-optimized during flight. Optimizing a high-dimensional trajectory with a time resolution of 10 ms is prohibitively slow. Nevertheless, the high time resolution at fixed duration is only required for the prediction horizon of the MAV controller. If we perform replanning sufficiently fast, the remainder of the trajectory can be represented at a lower time resolution, even with varying durations of trajectory points. We employ a local multiresolution time discretization—detailed in Section 2.3 and extending ideas from Behnke (2004) and Steffens et al. (2014). The prediction horizon is represented at a constant high resolution, after that the resolution decreases linearly. This resembles the uncertainty in the trajectory execution and perception in the future. As result, it is computationally feasible to plan at the control rate for the prediction horizon of a model predictive controller and at lower resolution to the global goal configuration. The duration of trajectory point i is $d(i) = \Delta t$ for $i \leqslant i_{fix}$ and $d(i) = (1 + c \cdot (i - i_{fix})) \cdot \Delta t$ for $i > i_{fix}$ with multiresolution factor

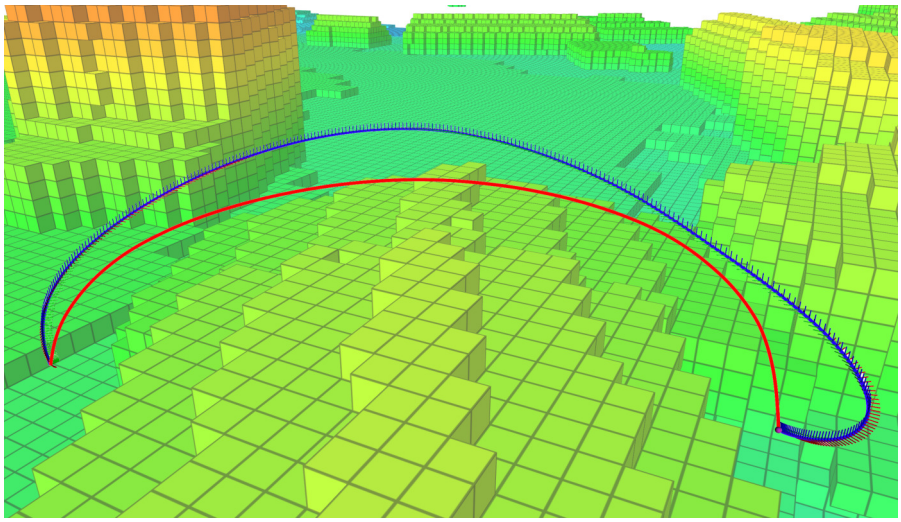


Figure 8.7: Optimized trajectories without (red) and with visibility constraints (blue). The constrained trajectory lunges out to keep the ascent and descent angles within the field of view of the onboard sensors. The flight starts on the left.

$c = 0.1$ in our implementation. This results in an overall duration from the trajectory start to point i of $D(i) = i\Delta t$ for $i \leqslant i_{fix}$ and

$$D(i) = i_{fix}\Delta t + \left(\Delta i + \sum_{k=1}^{\Delta i} ck\right)\Delta t \tag{8.4}$$

$$= i_{fix}\Delta t + \left(\Delta i + c\frac{\Delta i^2 + \Delta i}{2}\right)\Delta t \tag{8.5}$$

with $\Delta i = i - i_{fix}$ for $i > i_{fix}$. We set i_{fix} to 10, hence replanning has to be performed in 100 ms.

To calculate the derivatives for our trajectory given an arbitrary time discretization, we employ finite differencing (Fornberg, 1988). Due to the non-equal time difference between consecutive trajectory points, it is necessary to compute a differencing filter per trajectory point. Nevertheless, the time discretization is fixed over the whole optimization process, thus these filters can be precomputed. To allow for numerical differentiation of the first trajectory points, an additional padding of six time steps of fixed resolution is added at the beginning and end of the trajectory. The padding is fixed and not altered during the optimization process. In the first run the paddings are initialized with the start and goal configurations of the trajectory, respectively.

During execution of the trajectory, we shift the trajectory padding corresponding to the elapsed time. Hence, the trajectory padding always contains the past six trajectory states representing the dynamic state in the past that led to the current state. Consequently, the trajectory optimization finds feasible followup trajectories implicitly. The remaining trajectory is mapped to the new time parameterization by means of linear interpolation between trajectory points. We initialize

the optimizer with the current remaining flight trajectory shortened by the estimated reoptimization duration. The duration estimate is based on the last reoptimization duration with an added 10 % overhead as the duration is dominated by the remaining trajectory length which gets shorter during flight. Finally, the reoptimized part of the trajectory is merged with the currently executed trajectory.

After this reparameterization, some subsequent optimization steps are necessary to find the new local optimal solution, even if the MAV was not disturbed. In our implementation, we have restricted the reoptimization iterations to one fifth of the initial optimization iterations. As the trajectory points in the padding are not differentiable with the finite differences method, their derivatives have to be represented explicitly. Thus, our intermediate trajectories are 12+1 dimensional, containing 4D poses, velocities, accelerations plus duration.

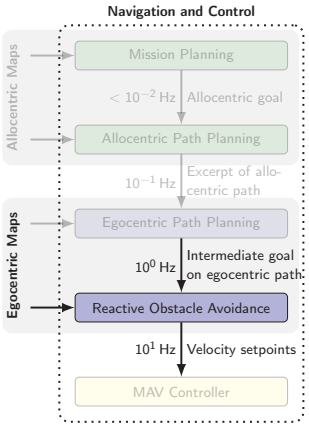
The trajectory can be reoptimized with few iterations for small changes in the environment, e.g., small or dynamic obstacles. The optimizer is initialized with the already optimized trajectory from the previous iteration and the updated environment model. Consequently, the initialization steps necessary for the first trajectory can be omitted and the trajectory converges fast to a new local optimum.

In case of disturbances of the MAV state while following an optimized trajectory, e.g., by strong gusts of wind, the remaining trajectory has to be adapted quickly. As the MAV dynamic state is not altered and thus the motion direction cannot be changed immediately, we move the initial part of the old trajectory to the deviated current MAV pose. We distribute the trajectory error over the remaining part of the trajectory. This part is then used as initialization for optimization yielding locally optimal trajectories to the goal.

If complete replanning is necessary during the trajectory execution due to topological changes in the environment that cannot be solved by reoptimization, the initialization steps including allocentric planning have to be repeated. In contrast to the initial planning, the trajectory padding of the sampled trajectory is now filled with a future part of the currently executed trajectory to take the dynamic state of the MAV into account for the followup trajectory.

The optimization cannot leave a local optimum if another trajectory would be closer to a global optimum. This can happen by newly perceived obstacles blocking or influencing the old locally optimal trajectory. To avoid this, we perform global replanning from points on the trajectory in the more distant future to the goal and newly initialize and optimize the remaining part of the trajectory, similar to the initial trajectory planning.

9 FAST REACTIVE OBSTACLE AVOIDANCE



We use reactive obstacle avoidance as a low-level safety layer complementing the deliberative path planning layers. For our application, reactive obstacle avoidance has two important properties—compared to fast local planning (Vanneste et al., 2014), or optimization-based approaches (Israelsen et al., 2014). First, it has the ability to elude approaching dynamic obstacles, depicted in Figure 9.1. This might include leaving a hover position or even moving into the opposite direction of the commanded flight path and not only modifying future flight paths up to a stop in front of obstacles. Second, a hazard minimizing solution will always be found even if the distance constraints are violated. Furthermore, reactive obstacle avoidance is computationally cheap and, consequently, can be executed with the lidar frequency of 10 Hz.

Another use-case that is facilitated by reactive obstacle avoidance is assisted flight. The obstacle avoidance layer can help a human pilot to prevent collisions in complicated situations, e.g., while flying through a narrow passage (Figure 9.1).

In this chapter, we summarize general artificial potential fields and detail our obstacle avoidance approaches with and without trajectory prediction.

9.1 ARTIFICIAL POTENTIAL FIELDS

The major design goal of our approach to obstacle avoidance is to react quickly on newly perceived obstacles. For this purpose, we extend artificial potential fields (Ge and Cui, 2002) and operate on aggregated egocentric sensor data at approximately the frequency they are processed. The artificial potential field approach is inspired by physical potential fields. In general, the robot is modeled as a particle passively moving through a field induced by attractive and repulsive forces. One or more goals induce attractive forces. In this thesis, we consider only one goal, the waypoint w_t, inducing the attractive force vector F_a. This attractive force directs the MAV towards the goal. A waypoint is either selected from a planned path or from user input. When employing a planner, it has to ensure that local minima are avoided. Perceived obstacles induce repulsive forces on the flying robot. The magnitude of the repulsive force F_r of the closest obstacle o at a position p is calculated as

$$\|F_{r,p}\| = \text{costs}\left(\operatorname{argmin}_o\left(\|o - p\|\right)\right),$$

Figure 9.1: Reactive obstacle avoidance. Our local obstacle avoidance algorithm acts as a safety measure between control inputs given by a planning layer or a human pilot remotely controlling the robot. This ensures safe operation in challenging situations. Left: A human pilot manually steers the MAV through a narrow passageway between vegetation and scaffolding leading to a collision that could have been avoided by obstacle avoidance. Right: A person approaches the MAV hover position, the MAV avoids this dynamic obstacle.

where costs(.) is a function that is zero in a user-defined safe distance to obstacles and raises with decreasing distance. This yields a repulsive force vector

$$F_{r,p} = \|F_{r,p}\| \frac{p - o}{\|p - o\|}.$$

The resulting force at a discrete position is now the weighted sum of the attractive and repulsive forces

$$F_p = aF_{a,p} + bF_{r,p},$$

with user-defined weighting factors a, b. A 2D example of the resulting potential field is depicted in Figure 9.2.

In general, the input to our obstacle avoidance algorithm is an aggregated robot-centered 3D occupancy grid based on measurements from a laser rangefinder (LRF) or an egocentric 3D point cloud from, e.g., stereo cameras and a target position or velocity vector, depending on the application. As most of the cells in our obstacle map are free space, we do not precalculate the forces for every cell, but only for cells that intersect with the bounding box of the MAV. For efficiency reasons, we calculate the force affecting one particle by selecting the closest obstacle. The effects caused by this simplification are mitigated by extending the standard potential field-based approach by relaxing the assumption that the robot is one idealized particle. We account for the shape of the MAV by discretizing it into cells (blue grid cells in Figure 9.3). Every cell is considered as one particle in the algorithm. This leads to a robot model containing particles in the center of the

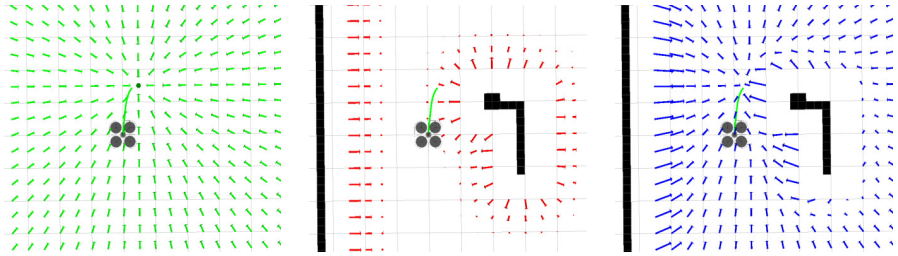


Figure 9.2: Artificial potential field. The local trajectory of the robot (green) is influenced by a weighted sum of attractive (left) and repulsive forces (middle). This induces an artificial potential field to navigate collision-free to an intermediate goal (right). The margins around obstacles without arrows depict the minimum safety distance.

cells p_i. The force affecting this model is the average of all individual forces

$$F_p = \frac{1}{N} \sum_i^N F_{p_i}.$$

Thus, we do not need to enlarge obstacles and avoid oscillations caused by the discretization.

Our formulation allows to evaluate the effects of the potential field on the MAV orientation. As the MAV is represented as a discretized 3D model, we can calculate the angular momentum on the MAV center by all artificial forces F_{p_i} applied to the individual robot cells i and their respective relative positions p_i:

$$M = \sum_i p_i \times F_{p_i}.$$

From the three rotational velocities, only the yaw velocity can be chosen independently of the linear velocities of the MAV. Hence, we project all p_i and F_{p_i} to a plane parallel to the ground. Thus, we get an acceleration around the z-axis resulting in an angular velocity. This can be used to orient less circular robots, e.g., an MAV with a sensor pole, away from obstacles. As our MAV is approximately circular, we give precedence to the orientation commanded by higher layers.

The motion command sent to the low-level MAV controller is calculated according to the resulting artificial forces. Output is a safe position or velocity vector, respectively. Maintaining the MAV attitude based on these outputs is the task of the underlying low-level controllers.

9.2 OBSTACLE AVOIDANCE WITH TRAJECTORY PREDICTION

In contrast to the idealized massless particle assumed in the potential field approach, frequent acceleration and deceleration of the MAV

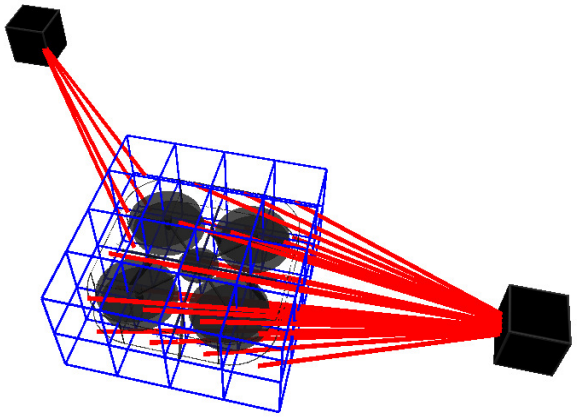


Figure 9.3: We discretize our MAV into cells (blue) and calculate the forces per cell. The artificial force applied to the MAV is the average of all forces. The nearest obstacles to the cells are depicted by red lines.

to follow the most cost-efficient path through the field is disadvantageous. To be able to totally change the motion direction at every discrete position would require low velocities. Hence, we accept suboptimal paths, as long as they do not lead to a motion state that will cause a collision in the future, e.g., if the MAV becomes too fast in the vicinity of obstacles. We take the special properties of MAVs in contrast to earthbound vehicles into account, by extending the classic potential field approach to collision avoidance with a prediction of the outcome of a chosen control for a fixed time horizon. This improves the navigation performance and closes the gap between pure reactive control and planned motions.

As additional input for our algorithm, we consider the current motion state x_t of the MAV at discrete time step t. The motion state x_t consists of the current linear velocity $v_t = (v_x, v_y, v_z)$, the rotational velocity around the z-axis ω_t, and the attitude of the MAV $R_t = (R_x, R_y)$ in an egocentric coordinate frame aligned to the floor. The target waypoint w_t is defined by a 3D position and 1D orientation $w_t = (x, y, z, \psi)$.

To account for the dynamic state x_t of the MAV, we predict its future trajectory Θ_t given the current linear velocities v_t, the attitude R_t, and the probable sequence of motion commands $u_{t:t+n}$ given in the next n time steps (Figure 9.4). The trajectory is predicted employing a learned motion model of the MAV, detailed in Section 9.3, and the expected resulting forces along the trajectory, given the current environment representation.

These predicted forces are used to influence the motion command selection. Larger forces indicate that the MAV will come close to an obstacle in the future while following the trajectory started by the current motion command. Hence, the next velocity commanded to

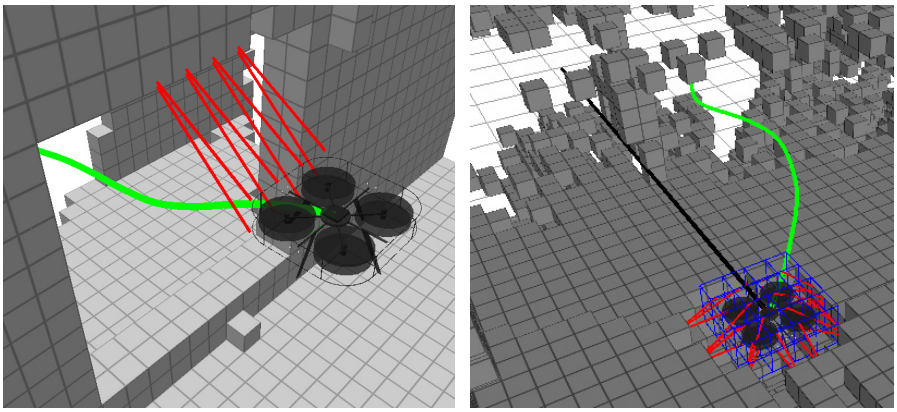


Figure 9.4: Trajectory rollout. We predict the influence of a motion command by rolling out the robot trajectory (green) using a learned motion model. The current artificial repulsive forces are depicted in red. Left: MAV flying through a window. Right: An obstacle is blocking the direct path (black) out of a garage.

the low-level controller needs to be reduced, accordingly. This is implemented in our algorithm as a reduction of the maximum velocity for the next motion command, if the magnitude of the forces along the trajectory becomes too large. For the prediction of the trajectory, this new maximum velocity is assumed as the maximum in the predicted future.

The prediction of the trajectory Θ_t is implemented as follows

$$\begin{aligned} \Theta_t &= p_{t:t+n} = (p_t, p_{t+1}, \ldots, p_{t+n}), \\ p_{i+1} &= Ax_i + Bu_i + p_i \qquad i \in [t : t+n-1], \\ u_i &= CF_{p_i}, \end{aligned}$$

where A, B denote matrices based on our motion model (described in the next section) to estimate a pose difference given the dynamic state and a control input u. C denotes a mapping of a force vector to a velocity command. Given the estimated sequence of future positions $p_{t:t+n}$, we search the smallest index $i \in (t : t+n)$ for which the magnitude of the force exceeds a threshold, i.e., $\|F_{p_i}\| > F_{max}$. If such an i exists, we reduce the maximum velocity v_{max} to

$$v_{new} = \left(\frac{1}{2} + \frac{i}{2n}\right) v_{max}.$$

Hence, while approaching an obstacle the maximum velocity commanded to the MAV is gradually reduced.

In the case that no trajectory with sufficiently small predicted forces can be found, the MAV stops. Here, we exploit the property of multicopters that, in contrast to fixed-wing unmanned aerial vehicles (UAVs), the dynamic state of the system can be changed completely within a short time. As a result, the look-ahead needed to estimate the effects of a control input is tightly bound. We have chosen the

length of the trajectory rollout as the time the multicopter needs to stop, which has been empirically measured to be one second for the employed control parameters.

9.3 LEARNING A MOTION MODEL

To obtain a motion model of our MAV, we fly our multicopter remote-controlled within a motion capture (MoCap) system. This system provides ground-truth data of the robot position and attitude at an average rate of 100 Hz allowing for the derivation of the robot dynamic state. Due to inconsistent delays within the MoCap system, captured data is often noisy and unsuitable for simple delta-time differentiation needed to correctly calculate the dynamic state. Thus, after capturing, all data is processed using a low-pass filter allowing for more accurate estimates of instantaneous velocities.

To determine the influence of control inputs to the system, we have to synchronize the user commands to the filtered state measurement. Due to the aforementioned MoCap time delays and the architecture of the data capturing system, which does not allow for explicit time synchronization, these two components are initially captured using different computer systems and fused later in a postprocessing step to avoid additional network delays. As the capture times of both components rarely coincide, we match the interpolation of the filtered state with control commands. Allocentric MoCap measurements are transformed from the capture frame into the egocentric MAV frame, the control inputs are normalized to the interval $[-1, 1]$. The final state estimate data can be used to derive the motion model parameters.

Several effects from flying within the MoCap system must be considered before an appropriate motion model can be derived. Due to a restricted capture volume, external thrust effects can be observed due to ground, ceiling, and wall planes interacting with the propeller-generated wind. These effects are minimized when flying within the central region of the capture volume, thus data acquisition is only performed within this restricted volume. Additionally, some maneuvers are impossible to capture due to both safety and acceleration constraints. However, as flying advanced maneuvers is not targeted in this work, this issue can be safely ignored. Using this simple set of capture constraints, external factors to the motion of the MAV can be minimized.

We model the flight dynamics as a time-discrete linear dynamical system (LDS) $x_{t+1} = Ax_t + Bu_t$ that predicts the state of the MAV at the time $t+1$ given the current state estimate x_t—i.e., attitude R_t, position p_t, angular ω_t and linear v_t velocities, and thrust T_t—and the user control input u_t. The state transition model A and control input model B are fitted to the captured data using ordinary least squares. Additionally, the yaw component of the model is considered

to be independent of the remaining parameters. Due to the properties of a multicopter, the attitude in the horizontal plane (i.e., roll and pitch) when maintaining altitude is nearly proportional to the acceleration along the corresponding axis for small angles (Beul and Behnke, 2016). Furthermore, for small vertical accelerations the thrust is dominated by the hover thrust compensating the gravity vector. We neglect the effects caused by drag, as our MAV flies at relatively low speed. With these simplifications and considering only the horizontal planar velocity and attitude, i.e., x- and y-axis, the transition model A for the corresponding velocities has the general form:

$$\begin{bmatrix} D_{xx} & D_{xy} & A_{xx} & A_{xy} \\ D_{yx} & D_{yy} & A_{yx} & A_{yy} \end{bmatrix} \text{ with state input } x_t = \begin{bmatrix} v_x \\ v_y \\ R_x \\ R_y \end{bmatrix},$$

where D_{ij} represents the dampening effects in the i-axis given the j-axis velocity and A_{ij} represents the acceleration effects from attitude in the i-axis given the j-axis attitude. Generally the dampening terms D_{xx}, D_{yy} are close to 1; D_{xy}, D_{yx} are close to 0 while the acceleration terms A_{xy}, A_{yx} correspond to the proportionality constant of the attitude; A_{xx}, A_{yy} are close to 0 (note that an attitude in one axis affects the acceleration of the perpendicular axis). Additionally, due to a non-symmetric inertia tensor of the multicopter, these matrix values may also not be symmetric resulting in different accelerations and dampening for the forward and lateral movements. The reduced state input consists of the planar velocity (v_x, v_y) and the corresponding rotations roll and pitch (R_x, R_y).

Through similar analysis, the control-input model B in the horizontal plane velocity and attitude has the general form:

$$\begin{bmatrix} \Phi_{A_{x\alpha_x}} & \Phi_{A_{x\alpha_y}} \\ \Phi_{A_{y\alpha_x}} & \Phi_{A_{y\alpha_y}} \\ \delta_{xx} & \delta_{xy} \\ \delta_{yx} & \delta_{yy} \end{bmatrix} \text{ with control input } u_t = \begin{bmatrix} \alpha_x \\ \alpha_y \end{bmatrix},$$

where $\Phi_{A_{ij}}$ represents the integrated accelerations from control input angles over the time period used for model learning in the i-axis from control input j. δ_{ij} approximates the reaction constants of the multicopter system to reach a desired attitude in the i-axis from the j-axis desired input. The control input is the desired attitude, i.e., roll and pitch angles. Both $\Phi_{A_{x\alpha_x}}$, $\Phi_{A_{y\alpha_y}}$ are relatively large while $\Phi_{A_{x\alpha_y}}$, $\Phi_{A_{y\alpha_x}}$ are small showing an independence between multicopter rotation axes. The δ values also show an independence between rotation axes. However, their values are specific to the multicopter control reaction speed, control delay, thrust output, and time period used for learning.

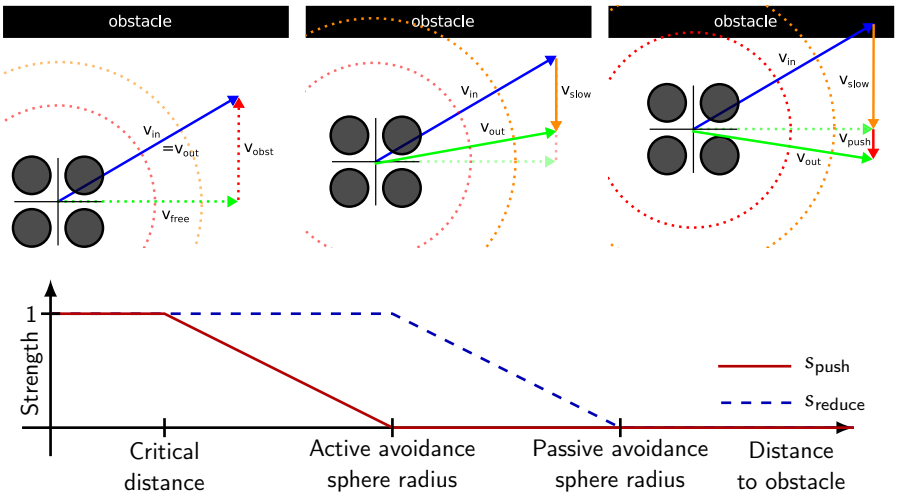


Figure 9.5: Reactive obstacle avoidance. Top-Left: The MAV velocity setpoint vector v_{in} is split into the projection towards an obstacle v_{obst} and the remainder v_{free}. If the MAV is not close to obstacles, the output velocity v_{out} is equal to the setpoint. Top-Middle: When an obstacle is in the passive avoidance sphere (dotted orange), v_{in} is reduced by $v_{slow} = -s_{reduce} v_{obst}$. Top-Right: Obstacles in the active avoidance sphere (dotted red) induce an additional repulsive force resulting in the pushing velocity v_{push} directing the MAV into free space. For simplicity, we depict velocity vectors, the pose modification vectors follow straightforward. Bottom: Scaling factors in relation to the obstacle distance.

In addition to predicting the trajectory for the purpose of collision avoidance, this model can be utilized in the low-level velocity controller (Achtelik et al., 2009), for kinodynamic motion planning (Şucan and Kavraki, 2008), and to predict away system delays due to the time difference of control input and control execution. Furthermore, we built a simulation environment using the model to ensure realistic behavior of the simulated multicopter.

9.4 OBSTACLE AVOIDANCE WITH DIRECTION-BASED VELOCITY REDUCTION

Based on our obstacle avoidance with trajectory prediction, we developed a simplified version without the need to learn an MAV motion model and a reduced set of parameters. This modified algorithm facilitates smoother flight in narrow spaces by adding two spheres of influence around the MAV, depicted in Figure 9.5. Obstacles in the passive avoidance sphere with radius d_p cause a reduction of the MAV motion into the direction of the obstacles. In the active avoidance sphere with radius d_a, obstacles exert artificial repulsive forces, increasing with proximity, that push the MAV away. By dividing the obstacle avoidance into these two phases, we achieve a stable equi-

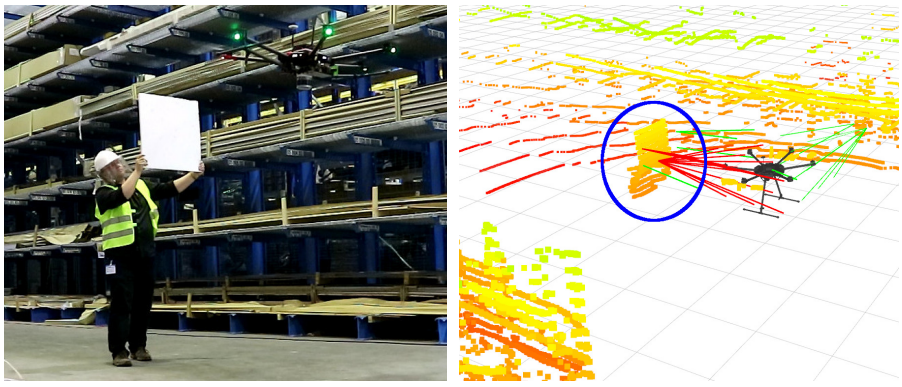


Figure 9.6: Laser scan of a person avoided by reactive obstacle avoidance. A person (circled blue in the laser scan) approaches the MAV. The MAV is repelled by the artificial forces (red lines) and dodges the obstacle. Green lines depict the influence of obstacles in the passive avoidance distance.

librium distance between obstacles and MAV regardless of the MAV control inputs without influencing the motion into orthogonal directions in the passive sphere—e.g., the MAV can follow an exploration pattern along a shelf even if the commanded pattern is too close to the shelf due to protruding goods. Figure 9.6 shows the influence of obstacles in the two spheres during a flight in a warehouse.

Our new obstacle avoidance layer can operate on—in addition to egocentric velocities as before—allocentric and egocentric target positions to allow a more flexible selection of planning and control layers. For simplicity of notation, all further calculations are depicted in an egocentric MAV frame to omit the localization transform matrices, allocentric positions and velocity vectors follow straightforward. If both spheres are obstacle-free, we execute the commands from the planning layer unaltered. Egocentric targets farther away than $1\,\mathrm{m}$ are first normalized, shorter vectors are processed without prior normalization to avoid a speed up of the MAV while approaching an obstacle. The new egocentric target position t_{new} is calculated as

$$t_{new} = t_{orig} - c_o s_{push} + f_o s_{reduce}.$$

Here, c_o is the projection of the current target t_{orig} onto the direction of the obstacle, thus, the part of the command that steers the MAV closer to the obstacle. The artificial force direction f_o is a normalized vector pointing away from the obstacle. The magnitudes of the slow down strength s_{reduce} and the push back strength s_{push}, depicted in Figure 9.5, are calculated as

$$s_{reduce} = \frac{d_p - d}{d_p - d_a},$$
$$s_{push} = \frac{d_a - d}{d_a - d_c},$$

with distance d to the obstacle and critical distance d_c. Both results are clipped to the interval $[0, 1]$ afterwards. Their value is a linear interpolation between free-space distance and the safety distance, and an interpolation between safety distance and critical distance, respectively.

Overall, this approach allows for less conservative safety distances—facilitating inspection and mapping closer to obstacles—while still maintaining safe navigation.

10 EVALUATION

In this chapter, we evaluate the individual components of our planning hierarchy as well as the integrated micro aerial vehicle (MAV) systems. We report quantitative and qualitative results from simulated experiments and flight tests with the different MAVs detailed in Section 5.2.

10.1 SIMULATION ENVIRONMENTS

For testing and evaluation, we employ different simulation environments. While several aspects of the system are simulated in very basic simulators, e.g., assuming that all actions are executed perfectly and making obstacles visible to the algorithms when in proximity, we also employ three different physics-based simulators for the integrated systems. In this section, we briefly describe these simulators.

Gazebo with Motion Model

Our first simulation environment is based on the physics-based simulation framework Gazebo from N. Koenig and Howard (2004). The Simulator is well integrated into the robotics middleware Robot Operating System (ROS). Gazebo can simulate complex 3D environments, actuators, and many sensors, e.g., cameras, laser rangefinders (LRFs), and ultrasonic sensors. Nevertheless, the physics engine of Gazebo is not aiming at simulating flying robots. Thus, we extended the simulator to move objects according to a black box motion model, given external control inputs and timings from simulation. Here, we can use a learned motion model, detailed in Section 9.3, of an MAV as plugin to move it realistic, without loosing other capabilities of the simulator like the simulated sensors or collision detection. Furthermore, we extended Gazebo with modules to support multi-echo LRFs (e.g., Hokuyo UTM-30LX-EW) and electromagnetic grippers. In our experiments, we employed a learned motion model of the AR.Drone. This simulator is mainly used for simulating an AR.Drone and our Pixhawk-based MAVs *MoDCopter* and *AIRCopter*.

RotorS

RotorS from Furrer et al. (2016) is an extension to Gazebo targeted at simulating flying robots, in particular multirotors. Instead of using a learned motion model, RotorS simulates the flight dynamics based on

physical parameters, e.g., the MAV layout and inertia, and the rotor torque. This allows for a flexible design of MAVs, but requires more knowledge about the flight dynamics and low-level controllers. The real MAVs employed in this thesis already abstract from this low-level layer by taking attitude or linear velocities as control inputs. To close the gap, we employ a modified version of the model predictive controller (MPC) by Kamel et al. (2017) as part of the simulator to obtain a velocity interface. Furthermore, RotorS provides additional sensors that we employed in testing, e.g., global navigation satellite system (GNSS) and inertial measurement unit (IMU) sensors. We used this environment mainly to simulate the *ChimneySpector* MAV.

DJI Hardware-in-the-Loop

To simulate the flight dynamics of the DJI MAVs, we employ a combination of Gazebo and the hardware-in-the-loop (HIL) simulator DJI Assistant 2 provided by the manufacturer. The DJI Assistant 2 runs on a dedicated PC which is connected to the MAV flight controller and simulates the MAV movement and sensor data, i.e., IMU and GNSS measurements. In addition, the onboard PC is connected to the flight controller similar to operating the real MAV. Thus, from a control perspective the simulation is completely transparent and can be used to simulate MAV flights not relying on other sensor input. Nevertheless, as the DJI Assistant 2 simulator can not simulate environment structures or sensors like LRFs, we feed the position, velocity, and attitude data from the flight controller to the Gazebo simulator. This data is then used to move and orient the simulated MAV accordingly, comparable to the black box model described above.

10.2 PATH PLANNING

Allocentric Planning

First, we evaluate individual properties of our path planner quantitatively. We investigate the influence on the planning duration of our design decision to employ a longer open list, i.e., not resorting the heap on key decreases, on the planning time. In our planners, we employ the binary heap implementation gheap[1]. The planning times are measured in a series of plans requiring from 737 to 665 194 node expansions. Table 10.1 reports the durations for a subset of the plans. The results show that a longer open list reduces the planning time by several orders of magnitude when using a heap-based priority queue by omitting the costly resorting of the heap structure.

When planning with sensor field of view (FoV) constraints, we employ a tailored heuristic. The largest impact on the number of ex-

1 https://github.com/valyala/gheap

Table 10.1: Planning duration with short and long open list (in s). The reported durations for the short open list are the minimum from three runs.

Expansions	4k	43k	114k	175k	295k	551k	665k
Duration (long)	0.04	0.13	0.26	0.37	0.63	1.16	1.31
Duration (short)	0.06	0.59	3.85	9.05	21.69	121.76	233.90

panded nodes in the A* search is expected, if the major difference between start and goal pose is a change in altitude without moving in the plane. We evaluate the heuristic in a grid with a horizontal edge length of 25 cm and a vertical sensor FoV of 30°, equal to the FoV of the Velodyne Puck LITE. For an ascent of 7 m in place the planner expands 943 505 nodes before reaching the goal with a Euclidean distance heuristic. Our FoV-aware heuristic reduces the number of expanded nodes to 285 411, which is approximately 30 % of the baseline. For the trajectories depicted in Figure 10.16 and Figure 10.19 the node expansions compared to the baseline are reduced to 63 percent (5 907 649 vs. 9 443 491 expansions) and 88 % (3 766 025 vs. 4 255 730 expansions), respectively.

For the evaluation of the three different employed obstacle cost representations, detailed in Section 7.1.1, we consider two different OctoMaps with a resolution of 0.25 m. The first OctoMap is a map of a building facade, a part of a courtyard, and some vegetation (see Figure 7.2) built from LRF measurements. This map contains 165 972 occupied octree leaves from which 30.2 % can be pruned to larger octree nodes (maximum depth 15: 115 804 nodes; depth 14: 6143 nodes; depth 13: 16 nodes). The second OctoMap represents the Frankenforst manor house depicted in Figure 10.3. This map is based on a level of detail (LoD) 2 model and does not contain the ground from the corresponding digital elevation model (DEM). Here, only 0.8 % of the 39 366 leaves can be pruned. Table 10.2 shows the results of this evaluation.

We evaluate three different cases for employing MAV models with multiple levels of abstraction:

- refinement of the model if it is within the maximum distance where an obstacle induces costs,
- refinement of the model only if it is so close to an obstacle that a cell becomes unreachable, and
- always using the finest MAV model.

The main advantage of the multilevel k-d tree representations is the reduced memory consumption. For the facade map the reduction is 27 %. For large, densely occupied maps, this is advantageous. Nevertheless, the overhead for the evaluation of cell costs introduced by the multilevel k-d tree representations is higher than the advantage

Table 10.2: Comparison of obstacle cost representations. For the abstraction levels, we report results for refining the MAV model in the case an obstacle induces any costs (refine max.), if a cell would be unreachable without refinement (refine min.), and for always using the finest MAV model. Durations for initializing (init) and for evaluating the obstacle cost (eval) are in ms. Free denotes the number of cells with obstacle costs lower than the maximum.

Representation	Facade map			Manor house map		
	Init	Eval	Free	Init	Eval	Free
Uniform	239.5	1133.3	283k	490.0	1932.0	355k
Multilevel	82.4	1324.9	283k	23.0	1931.5	355k
Abstraction						
- Refine max.	81.1	4131.6	296k	22.1	3089.5	361k
- Refine min.	71.4	2777.1	296k	16.9	2047.8	361k
- Only finest	74.6	8396.4	294k	18.3	13522.0	358k

of searching in smaller trees with our employed maps. The long initialization duration of our uniform obstacle representation is dominated by a full expansion of the octree in the initialization step. This duration could be reduced by expanding only the occupied space if necessary.

It can be seen that using three levels of abstraction strongly reduces the time required to determine the obstacle costs for the whole grid in contrast to always employing the finest MAV model. A further reduction can be achieved by only refining the model if a cell would not be reachable otherwise. This can lead to an overestimation of obstacle costs in other cells. In the facade map, these representations yield 4.5 % more reachable cells, especially close to the ground and on top of protruding building parts. For the manor house map this increase is lowered to 1.7 % due to the not represented ground. All results are averaged over ten runs.

We show the necessity for frequent replanning on the allocentric layer in some qualitative experiments. The experiment shown in Figure 10.1 is inspired by a typical situation in one of the university buildings. Depicted is a situation in which a previously open gate is closed and blocks a passageway. This change in the environment is too large, such that the local planner cannot find a local detour. Consequently, the global map needs to be updated with local sensor information. By frequent replanning, a new globally consistent path is found shortly after the blockage is incorporated into the allocentric map. Figure 10.2 shows resulting plans towards a goal at different stages of the flight. The general applicability of our planning hierarchy was tested by simulated following of globally planned paths

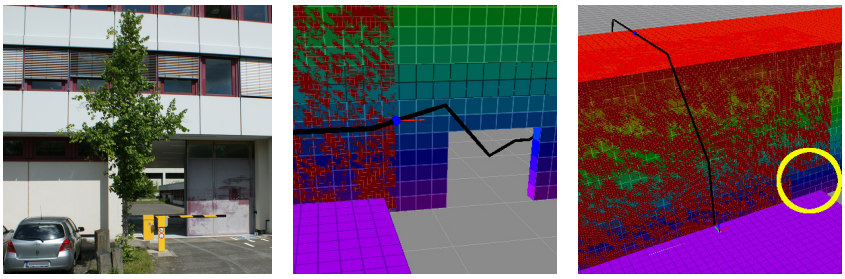


Figure 10.1: Example of frequent replanning on the allocentric planning layer. New sensor information (red dots) is incorporated into the allocentric map (colored voxels) to update the global plan. Left: Photo of a passageway with a closing gate. Middle: In the global world model the gate is open, the initial plan goes through the passageway. Right: While approaching the closed door (circled yellow) the MAV perceives it and updates the model. An alternative path over the building is planned.

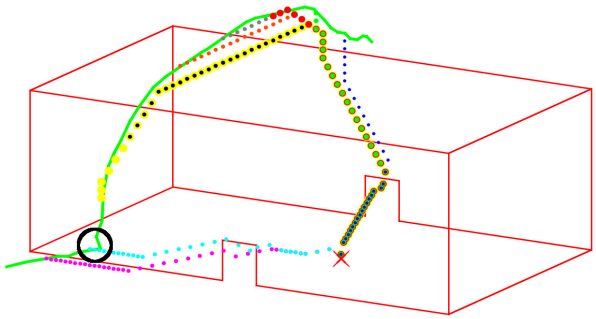


Figure 10.2: Updated global plans at different stages of the flight depited in Figure 10.1. The green line depicts the MAV trajectory, starting on the left side. The dotted lines depict plans from the respective MAV position towards the goal (red cross). The black circle marks the MAV position where the closed passage is perceived and the global plan is updated.

based on a city model represented in an $100\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$ grid with $1\,\mathrm{m}$ cell size. Our combination of allocentric and egocentric planners is able to achieve the required replanning frequencies on the *MoDCopter* onboard PC.

Figure 10.3 shows results from planning a flight path with different wind speeds. Whereas the MAV plans the shortest path in the absence of wind, it plans a qualitatively different path in the case of strong wind. In the second case, the path leads the MAV through the areas with lower wind speeds behind a large building.

Egocentric Planning

Local multiresolution planning acts as a layer to make frequent replanning feasible. Thus, we evaluate the reduction in the required

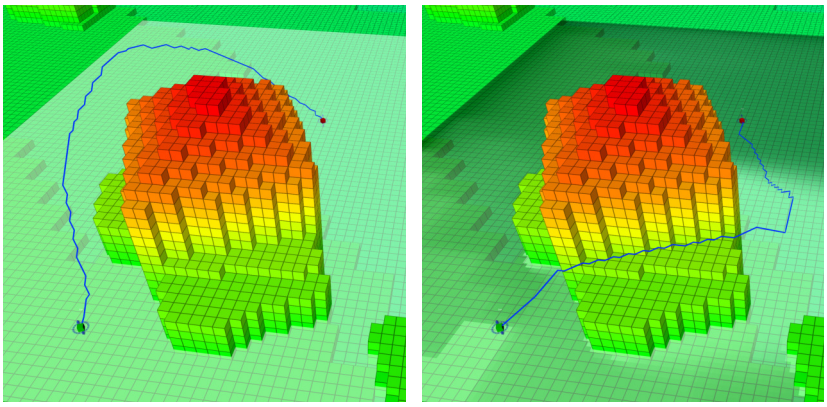


Figure 10.3: Planning with wind. The MAV plans qualitatively different paths depending on the wind speed. Left: No wind. Right: Strong wind from the left. The obstacle map (colored voxels) is overlaid with a slice from the wind model. Darker cells depict stronger wind.

planning time when employing multiresolution representations. On most of the employed MAVs, the onboard PC sends motion commands to the low-level controller with 10 Hz. Full 3D environment updates are acquired with 2 Hz with the rotating LRFs, whereas partial updates are available with up to 40 Hz. Thus, replanning times below 500 ms allow reacting on newly perceived obstacles in every full 3D scan. Durations below 100 ms allow replanning at the frequency of our obstacle avoidance layer.

We compared our path planner with local multiresolution grid with two uniform grid representations. The MAV follows a list of waypoints in simulation with an unknown obstacle which has to be surrounded locally. Figure 10.4 shows the example situation and resulting planning times during the path following. Input to the local planner is the next waypoint on the allocentric path or, if outside of the egocentric planning volume, the point where the allocentric path intersects with the planning volume. The multiresolution grid has an inner cell size of 0.25 m and the duration of one planning cycle is on average 12 ms, measured on the *MoDCopter* onboard PC. The maximum measured duration is 35 ms. With a uniform grid with cell size 0.25 m, the average planning time of 26 ms is still acceptable. Nevertheless, the maximum planning time of 3.4 s exceeds the desired time window of 100–500 ms by a factor of seven. Increasing the cell size to 1 m reduces the planning times to 4 ms in average and 20 ms in maximum. Our multiresolution approach results in 3 % longer trajectories than the higher resolution uniform approach. Using the lower resolution uniform grid results in 9 % longer trajectories. Table 10.3 summarizes the results.

We tested the tighter coupling of our global and local path planning layers to evaluate the computation time and the resulting flight

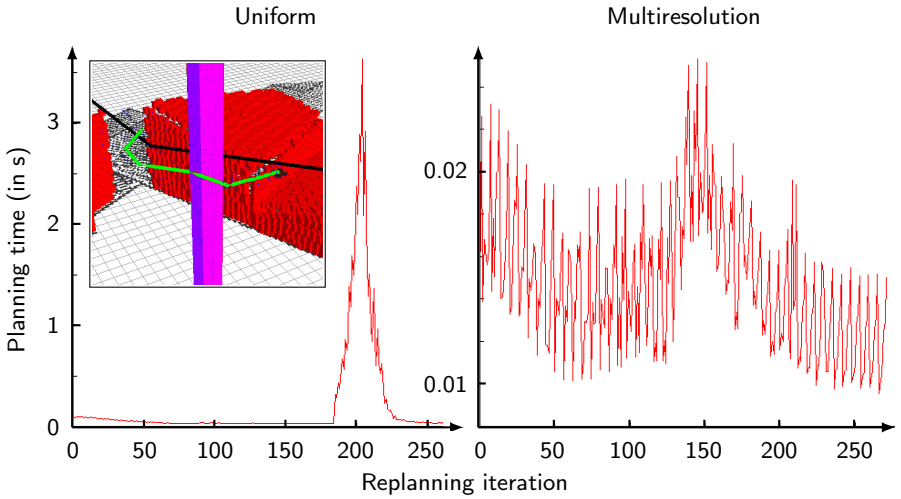


Figure 10.4: Plot of planning times per iteration. Inset: The MAV is approaching an obstacle (purple) where it has to deviate from the globally planned path (black). The local path is depicted by green arrows. Left: If the uniform grid with 0.25 m cell size is used the available planning time is exceeded substantially. Right: The local multiresolution planner is able to replan without a strong increase in planning time.

paths. As before, the MAV follows a globally planned path and has to avoid obstacles that are not in the a priori known world model. For evaluation, we manually removed obstacles from our allocentric map and plan global paths based on this modified map and local paths based on our local map incorporating laser measurements. Figure 10.5 shows plans while locally surrounding one of the shelves in the indoor evaluation area. When newly perceived obstacles have to be avoided, the planning time for a uniform grid with high resolution can substantially exceed the time window for replanning. By contrast, the local multiresolution planning is always fast enough for continuous replanning.

Table 10.3: Planning time and trajectory lengths of local path planner without trajectory coupling. Lenghts are normalized to the length of the shortest trajectory.

Grid representation	Cell size	Planning time (in ms)		Length
	(in m)	Min.	Max.	
multiresolution	0.25	12	35	1.03
uniform	0.25	26	3395	1.00
uniform	1.00	4	20	1.09

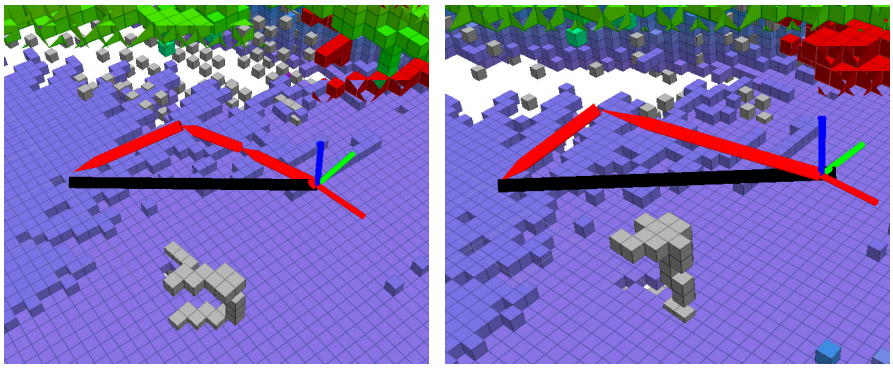


Figure 10.5: Local plan around an obstacle. The allocentric path planner (black line) plans globally consistent paths based on our allocentric map (blue and green boxes). Our local path planner (red arrows) is coupled to the allocentric plan, but surrounds obstacles in the vicinity of the MAV based on our local obstacle map (gray and red boxes). The MAV pose is depicted by the axes.

Table 10.4: Planning time of local path planner with trajectory coupling. Case A: Maximum planning time if the allocentric plan can be followed. Case B: Maximum planning time when deviating from the allocentric plan.

Grid representation	Cell size (in m)	Planning time (in ms)		
		Min.	Max. (A)	Max. (B)
multiresolution	0.25	10	40	60
uniform	0.25	10	210	720
uniform	1.00	10	210	8200

We compared the computation times of our local planner when employing different planner representations: two uniform grids and our local multiresolution grid with $8 \times 8 \times 8$ cells per level. All grids have a size of $32\,\mathrm{m} \times 32\,\mathrm{m} \times 32\,\mathrm{m}$. Here, we measured the runtimes on the *MoDCopter* onboard PC employing data recorded during autonomous operation for our integration experiments in a decommissioned car service station described in Section 10.5. Table 10.4 summarizes the results. While the planning times remain sufficiently low for frequent replanning when employing the multiresolution grid, the maximum planning times become prohibitively long when using the uniform grids. The minimum planning time with all representations is approximately 10 ms. For the maximum planning time, we report two results: A) maximum planning time if the allocentric plan can be followed, B) maximum planning time when deviating from the allocentric plan. In case A, the planning with the multiresolution grid takes up to 40 ms, both uniform grid-based planners need up to 210 ms. In case B, the maximum planning time with the multireso-

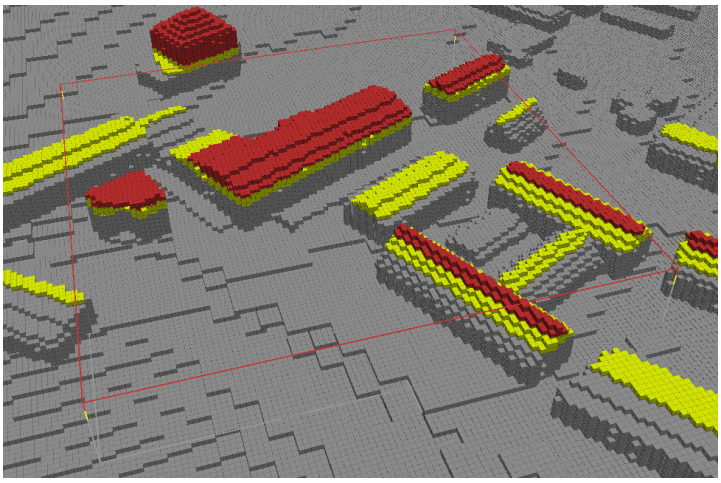


Figure 10.6: OctoMap of the evaluation area. We have restricted the maximum allowed altitude for our experiments to approximately 10 m over ground, otherwise flying at higher altitudes always yields shortest paths. Yellow: Obstacles influencing the MAV at 10 m altitude. Red: Obstacles the MAV cannot overfly.

lution grid increases to 60 ms, whereas the maximum planning times with the uniform grids increase to prohibitively long 720 ms (1 m), and 8.200 ms (0.25 m), respectively.

10.3 TRAJECTORY OPTIMIZATION

We evaluate the individual components of our trajectory optimization pipeline, which are

- the influence of our spline- and motion model-based initialization techniques and the continuous curvature transition segments (CCTSs), detailed in Section 8.2,
- optimization with sensor visibility constraints (see Section 8.3),
- and frequent reoptimization with multiresolution in time (see Section 8.4).

Furthermore, we show the applicability of our approach by simulated flights of MAVs in the RotorS simulator as well as with the HIL simulator, both detailed in Section 10.1. We perform proof-of-concept flight experiments with the *ChimneySpector* MAV in a motion capture system and with the DJI Matrice 600 outdoors.

Initialization and Optimization

In these experiments, we evaluate the improvements of plan- and spline-based initialization for our trajectory optimizer without CCTSs. The experiments are performed employing an outdoor map containing buildings from a village and a farm area, depicted in Figure 10.6. In this environment, shortest paths are often direct connections at a

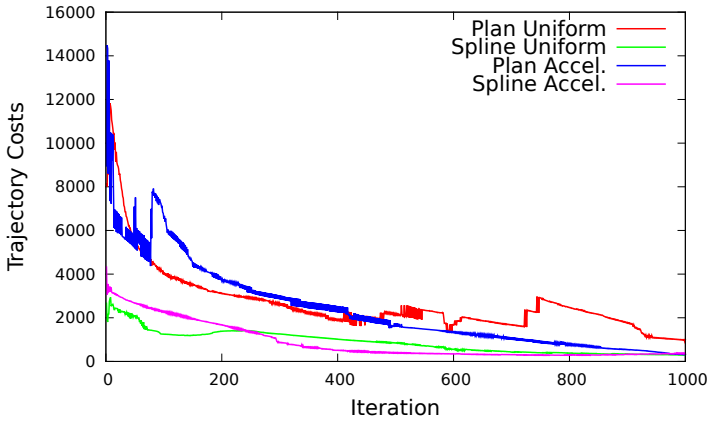


Figure 10.7: Trajectory costs (state and control costs) per iteration of the optimizer. Whereas the optimizer converges to nearly the same value for all initializations, spline-based initializations (green/pink) reach a low value much faster. Also, the combination of spline-based initialization with non-uniform accelerations (pink) reduces the convergence time.

certain height. To avoid this simple solution, we restrict the allowed flight altitude to a fixed absolute height. Depending on the terrain elevation, this limit is 10–14 m above ground-level. Some buildings are higher than this allowed altitude and have to be surrounded by the MAV. The path planning grid has a size of 100 m × 100 m × 14 m and a cell size of 1 m. Our distance field is 3 m larger in every dimension to allow for correct gradient calculations and has a resolution of 20 cm. The higher resolution of the distance field compared to the planner grid is exploited in the following optimization step. Here, the allowed minimum distance to obstacles is 2 m, the maximum distance influenced by an obstacle is 4 m. The generation of the initial distance field from an OctoMap takes 6.1 s. All timings are evaluated on a single core of the *MoDCopter* MAV onboard computer.

In the first experiment, we plan a path of 229 m for further optimization. A valid path is found in 0.46 s. Our second step, the calculation of timings and spline-based-trajectories runs in under 1 ms. Figure 10.7 shows the convergence of the trajectory optimizer in our four evaluation cases:

- uniform sampling of the planned path as initialization for the optimizer,
- sampling of the planned path according to a motion model,
- spline interpolation with uniform sampling,
- combining spline interpolation and motion model.

The costs of a trajectory are a sum of state and control costs. The control costs penalize the change in control input, i.e., minimize the jerk of the trajectory. State costs incorporate obstacle costs, accelerations, and velocities of the MAV. Clearly, the spline-based initializations start with much lower trajectory costs and converge faster to the local

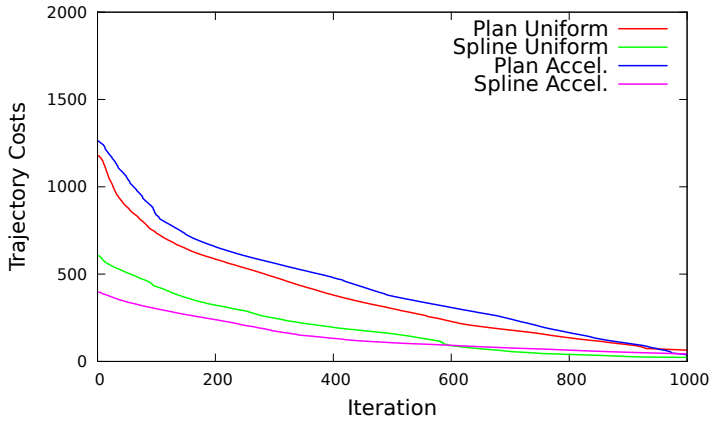


Figure 10.8: Summed control costs per iteration of the optimizer. After initialization the overall control costs of the trajectory could be reduced by 25–34 % by employing better timings and by approximately 75 % by using splines in this example.

optimum. The influence of the motion model-based timing correction can be seen in the first iterations of the plan-based initializations. Nevertheless, it has no observable effect if combined with splines at the beginning of the optimization. This can be explained by a larger overshoot causing higher velocities and obstacle costs in parts of the initial trajectory increasing the state costs while reducing the control costs. We depict the control cost part without state costs in Figure 10.8. Here, the initialization with motion model-based timings is clearly better in both cases, with and without a combination with splines. We achieve 25 % and 34 % lower control costs in the beginning by using improved timings and approximately 75 % lower initial costs by using splines. In combination, the initial control costs can be reduced by 77 %. In normal operation, we stop the optimization after 500 iterations. The optimization process takes 0.96 s for trajectory points with a Δt of 0.05 s.

Figure 10.9 shows a comparison of the position trajectories for (x, y, z) after initial planning, spline interpolation and optimization.

In the second experiment, we generate trajectories for pairs of obstacle-free start and goal poses uniformly distributed over the evaluation area. We omit trajectories between poses with less than 70 m distance. This results in 1,216 trajectories with an average length of 101 m. The shortest planned path was about 72 m, the longest path was about 155 m. We stop the optimization after 500 iterations and evaluate the trajectory cost reduction with our proposed initializations during optimization. We calculate the cost reduction r_i after optimizer iteration i as

$$r_i = \left(1 - c_i / c_i^{base}\right) \cdot 100\%.$$

Here, c_i^{base} is the average cost of a trajectory after iteration i with the baseline algorithm and c_i is the average cost with the evaluated

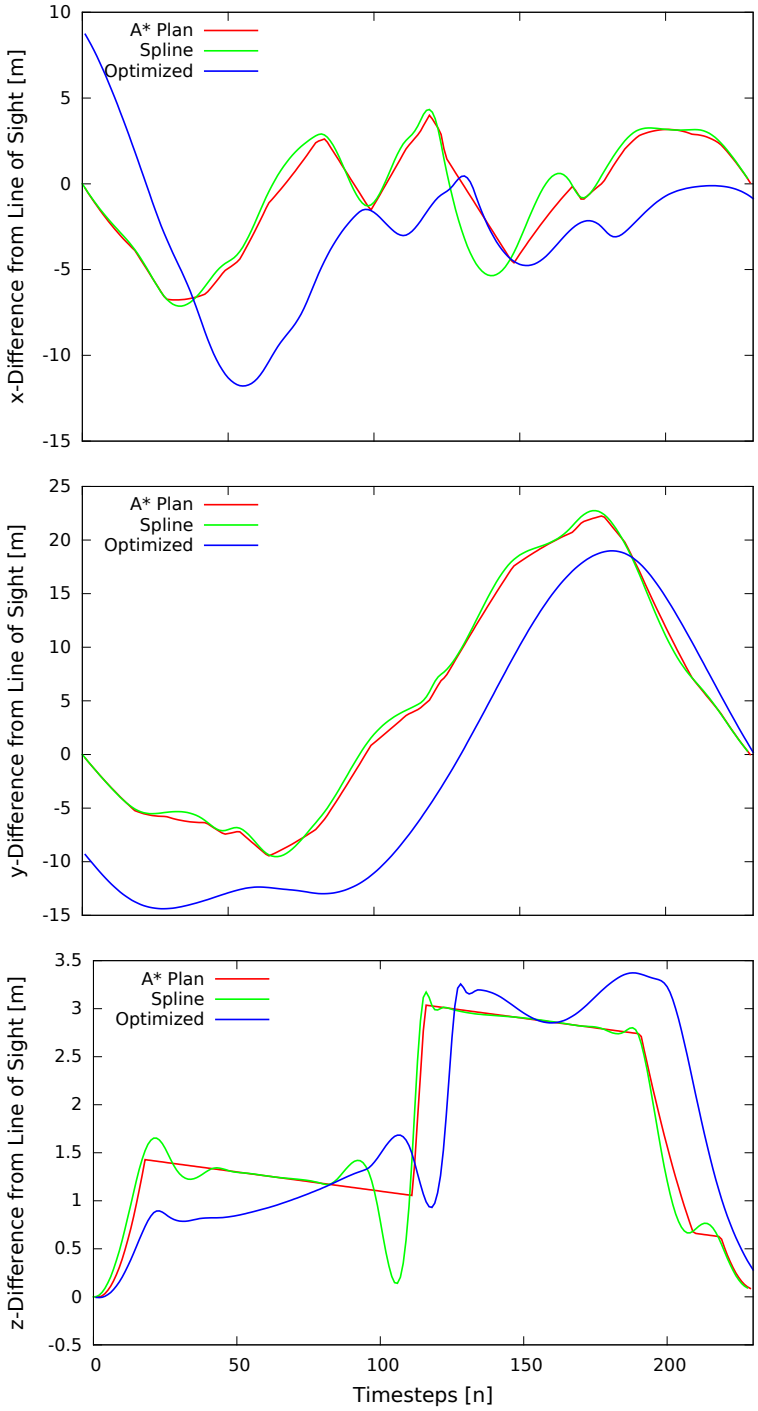


Figure 10.9: Comparison of trajectories for individual position dimensions at different stages of optimization. Planned paths (red) with applied timing correction require still large accelerations when the movement direction changes. Spline interpolation (green) mitigates these effects, but tend to overshooting and are bound to the initial plans sampling points. The optimized trajectories (blue) are much smoother and reduce the necessary control effort.

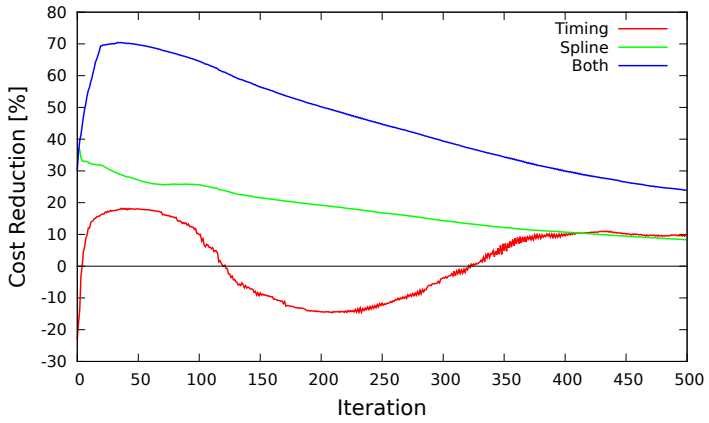


Figure 10.10: Reduction of trajectory costs compared to baseline (initialization with A* planned path) in each optimizer iteration. The results are an average over 1216 trajectories.

initialization. Figure 10.10 shows the trajectory cost at each iteration compared to the baseline, i.e., direct initialization with the plan from the grid-based planner. With enough iterations, the cost reduction converges to zero as the optimization initialized with the baseline approach will finally converge to the local optimum, but this makes frequent planning infeasible. Especially the spline-based initialization reduces the initial cost drastically. Combined with the motion model-based timing correction, after 500 iterations the trajectory is still less costly than without. By means of spline interpolation, the initial costs can be significantly reduced. This leads to faster convergence, resulting in 20–45 % less costly trajectories after 250 iterations and 9–24 % less costly trajectories after 500 iterations, compared to the baseline.

Employing motion model-based timings reduces the cost at some iterations when directly applied to a planned path. But higher velocities and accelerations in other iterations can lead to much higher control costs in cases where connections between plan segments have to be traversed with high speeds. This results in low improvements or even negative effects. A positive effect can be observed in the long run, after the initial plan has been smoothed enough by optimization.

Table 10.5 shows the average and maximum runtimes of the planning and trajectory optimization. In 98.8 % of the optimization runs, the summed planning and optimization times are below 1 s. Some more complex trajectories take longer planning and optimization time (maximum 1.45 s), yielding an average total optimization time of 0.64 s.

In the next experiments, we evaluate our clothoid-based CCTSs in combination with our motion model- and spline-based initializations.

Figure 10.11 shows the reduction of the maximum accelerations at transitions between consecutive path segments. The initial planned path has discontinuities in the derivative of the trajectory causing

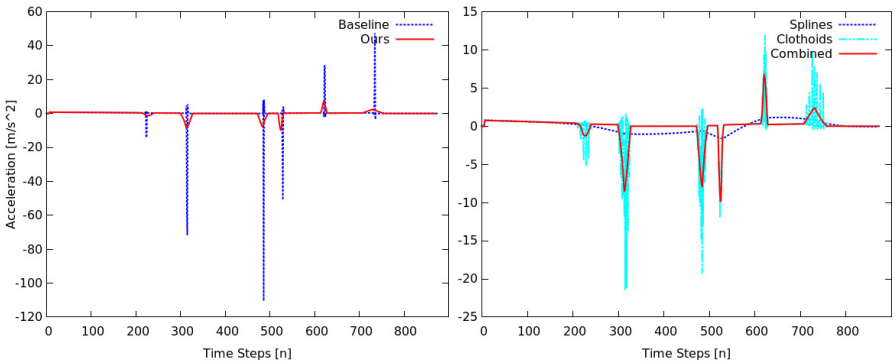


Figure 10.11: Accelerations of initial trajectory. Left: Straight line trajectories have discontinuities causing large accelerations at the transition points (Baseline, planned path from A* planner). Our approach—a combination of continuous curvature transition segments with spline interpolation—reduces these accelerations much closer to feasible accelerations (approximately $3\,\mathrm{m/s^2}$ for our MAV). Better initial guesses for the trajectory lead to faster convergence. Right: Spline interpolation combined with clothoid transitions further mitigate effects caused by discretization. Splines fitted to the initial trajectory result in even smaller accelerations, but cannot assure obstacle-freeness of the resulting path. Our approach alters the path only locally and only if in obstacle-free volumes.

large accelerations for single time steps. In contrast, our approach—a combination of CCTSs with spline interpolation—reduces the maximum acceleration at transition points significantly. Spline interpolation alone—without clothoid segments—has the potential to reduce the accelerations further by dropping guaranteed obstacle-freeness. Our approach preserves obstacle-freeness by inserting local changes into the plan in an obstacle-free volume. We depict the acceleration reduction factors in dependence of the transition angle and radius of the free-space spheres in Figure 10.12. We generated 10 m long trajectories with one transition between line segments in the middle with uniform time discretization. Resulting acceleration reductions are re-

Table 10.5: Runtimes of planning and optimization averaged over 1216 trajectories.

Phase	Duration (in s)		
	Mean	Std. dev.	Maximum
Path Planning	0.07	0.03	0.29
Optimization	0.58	0.13	1.16
Total	0.64	0.14	1.45

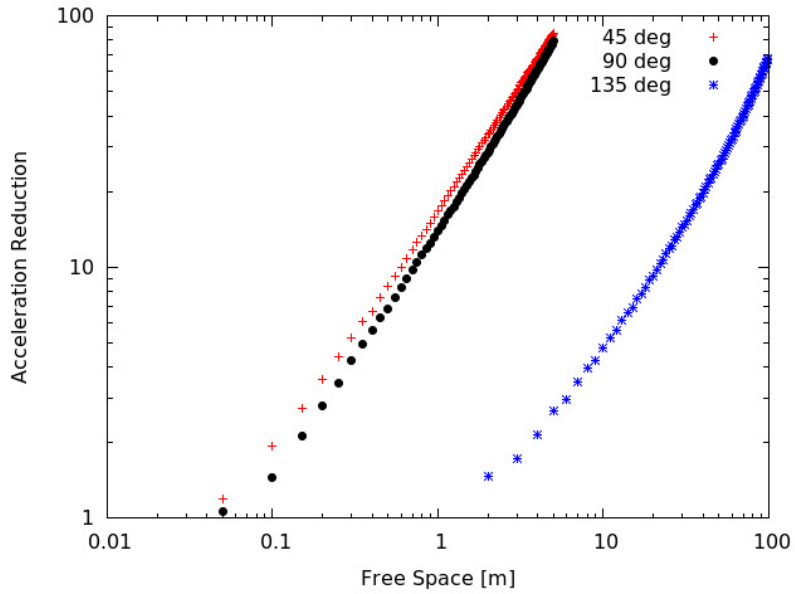


Figure 10.12: Acceleration reduction with CCTSs. Depicted is the reduction factor $a_{max}^{baseline}/a_{max}^{ours}$ of the maximum accelerations encountered with the baseline trajectory—planned with the A* planner—and ours on a double logarithmic scale. An order of magnitude of reduction can already be observed at less than one meter of free space with smaller transition angles. A transition with an angle of 135° still causes large accelerations when generated in a small free-space sphere due to the high necessary maximum curvature. In non-artificial scenarios angles beyond 90° are uncommon.

ported for free space from 5 cm up to 5 m and typical angles of 45°, 90°, and 135°. The initial accelerations are already an order of magnitude smaller within a free-space volume of 1 m with our approach for typical heading changes. Only the angle of 135° shows less reduction, because the maximum curvature for large heading changes still causes large accelerations when performed in a small radius. Nevertheless, such large heading changes are very uncommon with a grid-based planner in real world scenarios.

In simulation, the *ChimneySpector* MAV followed trajectories around a power plant that is part of the RotorS simulator. We defined a path by 12 via-points that constrain the optimization topological to a trajectory traversing eight apertures in the building. Our preprocessing calculates free-space bounding spheres at the via-points and inserts CCTSs. Finally, the path is optimized and executed with frequent reoptimization. Figure 10.13 shows an example trajectory. We show a comparison of the convergence behavior with and without CCTSs in Video 10.1[2].

2 Video 10.1: www.nieuwenhuisen.de/thesis/optimization-ccts.mp4

To test the applicability of our approach, we performed proof-of-concept real robot experiments. We followed trajectories generated by our approach with uniform timing trajectory optimization with $\Delta t = 10\,ms$ with the *ChimneySpector* MAV and the MPC from Kamel et al. (2017). State feedback was provided by a motion capture system. The complete optimized trajectory is given to the MPC as an input without further reoptimization. In four experiments the commanded trajectories could be followed with only a small time delay. Thus, the motion dynamics model employed during optimization resembles the real MAV flight dynamics. Figure 10.14 shows the results of one exemplary trajectory execution. The shape and dynamics of the reference trajectories are qualitatively well-matched by the MAV, showing the applicability of our approach to robot navigation. Video 10.2[3] shows two more experimental flights with the *ChimneySpector* MAV.

FoV-aware Trajectory Optimization

We analyze the trajectory planning and optimization in the sensor FoV qualitatively in the outdoor farm map and regarding the ascent and descent angles with and without our approach.

With our approach the ascent and descent angles of the trajectories are bounded by the FoV of the onboard sensor. When ascending or descending in place, as depicted in Figure 10.15, the shortest path yields angles close to 90° for the whole flight, clearly not covered by the onboard sensor. The resulting spiral motion after optimization facilitates a very smooth ascent with angles always close to the allowed maximum. The right part of Figure 10.15 shows the angles per pair of trajectory points along the path after optimization.

A more realistic example in the outdoor farm map where buildings block the line-of-sight between start and target poses is depicted in Figure 10.16. As the start pose is close to an L-shaped building, the MAV has to fly away from the facade first and perform a partial spiraling motion to gain altitude. After passing the building through a cut-in between higher parts of the roof, the descent is smoothly distributed along the remaining trajectory. In comparison to the planned path—which is also valid w.r.t. visibility constraints—the optimized trajectory can be flown at higher velocity since it does not contain sharp turns. Thus, the optimized trajectory is less compact. The corresponding angles between consecutive trajectory points for this example are depicted in the left part of Figure 10.17. The right part of Figure 10.17 shows the optimized trajectory without constraints as a reference. It can be seen that without constraints the trajectory goes up nearly vertical, reduces the ascent angle nearly linearly until it descents nearly vertical again The visibility constraints are violated for

3 Video 10.2: www.nieuwenhuisen.de/thesis/optimization-chimneyspector.mp4

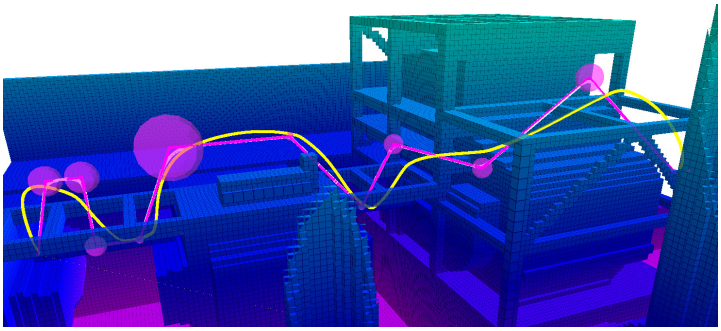


Figure 10.13: Processing steps of trajectory optimization. To compute dynamically feasible trajectories, we initialize the optimization with a collision free path of straight line segments (violet). We construct CCTSs (white) bound by obstacle-free spheres (violet spheres). After estimating velocities and accelerations along the trajectory with a simple MAV dynamics model, we optimize this initial trajectory to a dynamically feasible local optimum preserving the topological constraints (yellow).

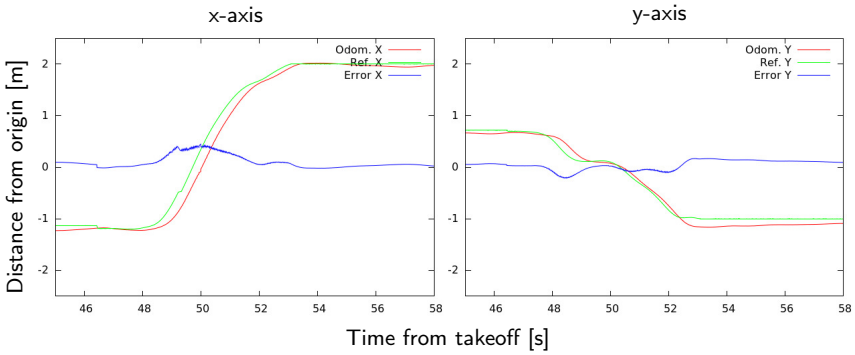


Figure 10.14: Trajectory following with *ChimneySpector* MAV. We tested the dynamic feasibility of the generated trajectory for direct execution on an MAV while flying in a motion capture system. The green graph depicts the reference trajectory in the x and y axes, the red graph pose measurements from the motion capture system. Blue depicts the trajectory error.

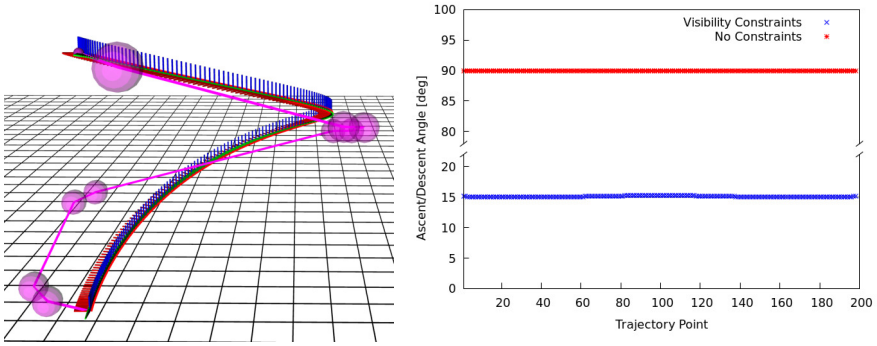


Figure 10.15: Angles for ascent in place. In the example depicted on the left the MAV ascents continuously with 15° with our approach (blue graph). Without constraints the ascent angle is 90° (red graph).

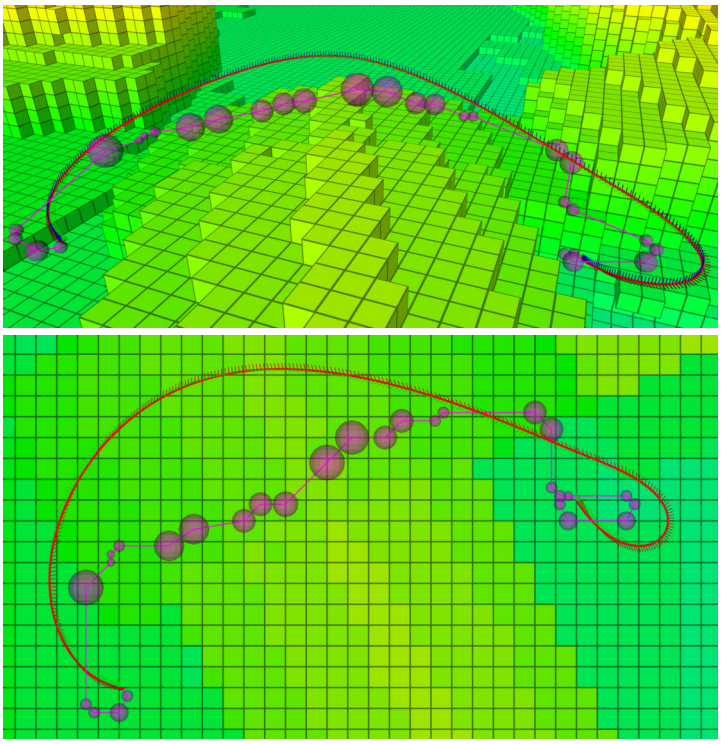


Figure 10.16: Plan and trajectory in outdoor map. Whereas the planned path (purple) is more compact and shorter, the optimized trajectory allows higher velocities due to a smoother flight path. The flight direction is from right to left.

approximately 75 % of the flight time, resulting in a large collision hazard. With enabled visibility constraints the ascent and descent are within the maximum allowed band.

We evaluate the applicability of our approach for MAV control with our DJI Matrice 600 MAV. In addition to outdoor experiments, we employ our HIL simulator. The optimized trajectories are executed by the MPC from Beul and Behnke (2017). Input to the controller are the next trajectory point position and velocity with 10 Hz. The commands are sent open-loop according to the calculated timings. By predicting intercept points given the target position and velocity the MPC is able to track the trajectory accurately without large contouring error. We report absolute trajectory errors (ATEs) between optimized trajectories and the pose estimates of the MAV during simulated flight in Table 10.6. The ATEs are averaged over ten flights per example. Spiral and Flight 1 are the trajectories depicted in Figure 10.15 and Figure 10.16, respectively. Flight 2 and Flight 3 are longer trajectories with different start and end points in the same map. The MAV reaches velocities of up to 2.43 m/s from an allowed maximum of 3 m/s in the controller. Thus, the resulting trajectories are within the dynamic limits of the MAV without slowing down the MAV too much.

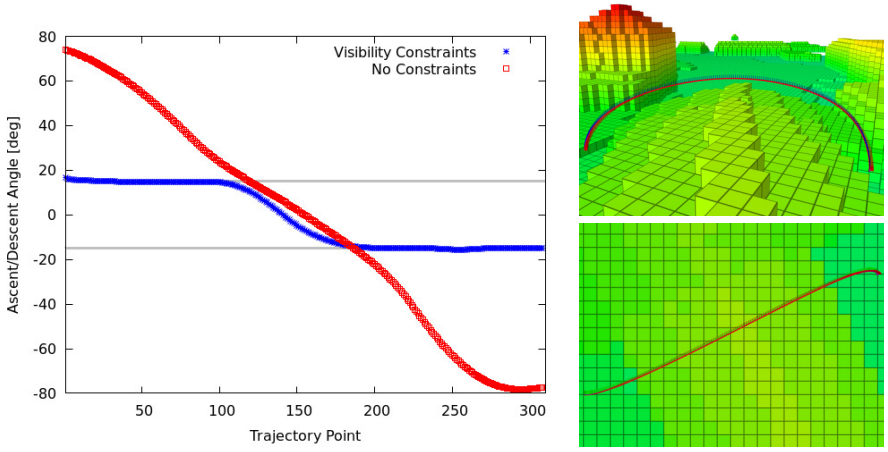


Figure 10.17: Angles for outdoor trajectory. Without constraints, the ascent and descent angles of the MAV trajectory, depicted on the right, change nearly linear from 75° to −80° (red) caused by an arc-shaped trajectory over the building. With enabled visibility constraints, the trajectory is divided into an ascent, flight, and descent phase (blue). The angles stay within the band defined by the FoV of the sensor (gray lines).

Table 10.6: Absolute trajectory errors (ATE) during trajectory execution simulation (in m). The ATEs are averaged over ten flights. v_{max} is the maximum reached velocity along the trajectory in m/s.

	Spiral	Flight 1	Flight 2	Flight 3
ATE	0.22	0.46	0.59	0.67
RMSE	0.14	0.30	0.34	0.37
v_{max}	1.22	2.43	2.26	2.34

Figure 10.18 shows a comparison of the trajectory setpoints and the flown trajectories for the first run of each of the examples.

The outdoor experiments with the DJI Matrice 600 MAV were performed in a free-space area augmented with artificial obstacles in the map. Figure 10.19 shows an example with a high wall with an opening at a height of 4 m. To overcome the wall without violating the sensor FoV constraint the MAV flies two connected partial spirals. A second performed experiment includes an artificial wall with a uniform height of 4 m. In these experiments the MAV plans and optimizes two qualitatively different trajectories, depending on the exact start condition. The trajectories can either be of a shape comparable to the experiment with the opening or have a U-shape with roughly straight ascent and descent segments. The third conducted experiment is an ascent in place similar to the spiral depicted in Figure 10.15.

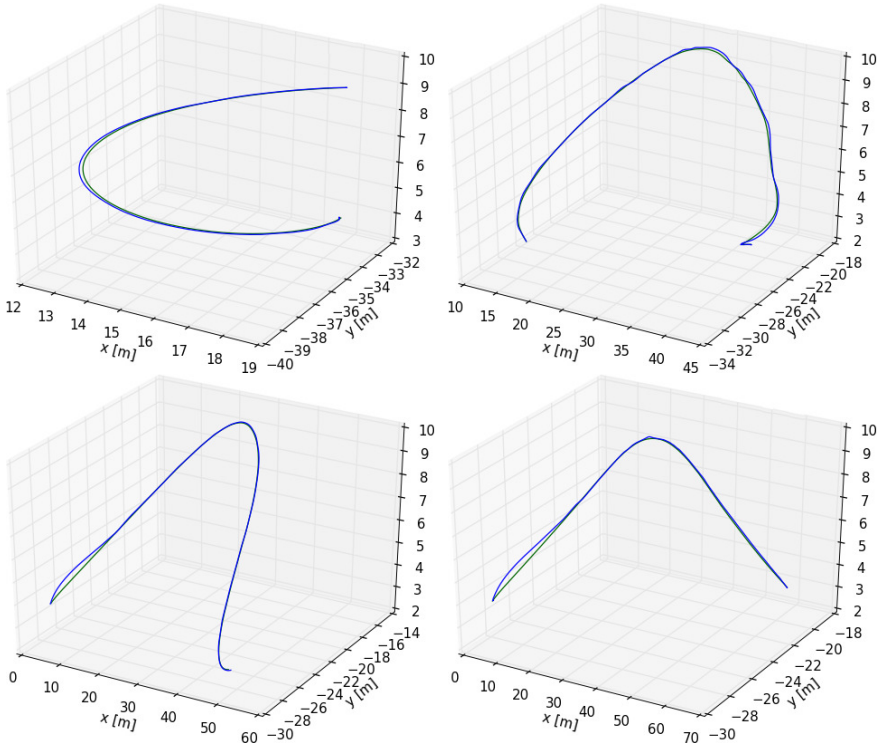


Figure 10.18: Trajectory tracking during flights. Green depicts the trajectory setpoints, blue the odometry estimates during the four example flights Spiral (top-left) and Flight 1-3.

For state estimation in these experiments, we employ the onboard filter of the DJI flight control incorporating global positioning system (GPS) and IMU measurements. As no ground truth apart from this is available, the ATEs reported in Table 10.7 represent the trajectory tracking error based on the onboard state estimate. Video 10.3[4] shows footage of our outdoor experiments and results from the simulation experiments.

Frequent Trajectory Reoptimization

First, we evaluate the reoptimization of trajectories with uniform resolution. We disturb the MAV flight by simulating strong gusts of wind while the MAV follows a trajectory. As baseline, we apply the transformation between the old MAV pose and the disturbed one to an initial fixed part of the trajectory and perform complete replanning and optimization from the endpoint of that fixed trajectory part to the goal. The fixed part is the part the MAV will follow during replanning due to its current dynamic state. Figure 10.20 shows the cost reduction of the trajectory during initial optimization and while reoptimizing the trajectory after two gusts of wind, each pushing the MAV away 4.25 m. Reoptimization yields a close-to-optimal cost trajectory in less

4 Video 10.3: www.nieuwenhuisen.de/thesis/optimization-constraints.mp4

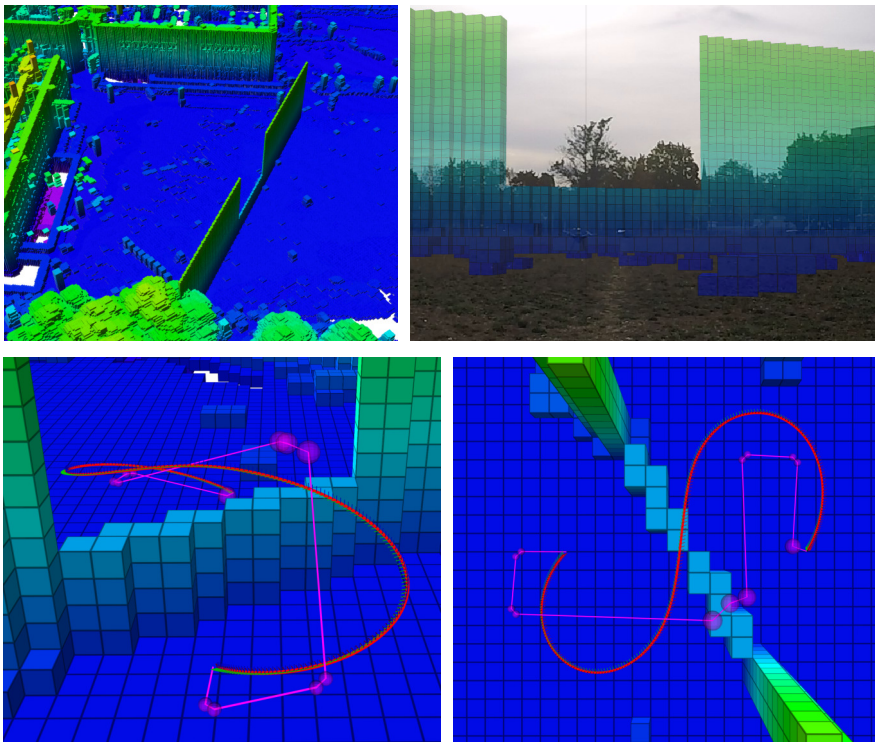


Figure 10.19: Real world experiment for FoV-aware optimization. A free-space area on the campus is augmented with artificial obstacles. Top-left: OctoMap of the environment. Top-right: Photo of the environment overlaid with artificial wall. Bottom: Our MAV plans and optimizes a trajectory to overcome the wall (flight from front/left to rear/right). The optimized trajectories are successfully followed by the DJI Matrice 600 MAV. The depicted voxels have an edge length of 1 m.

than 100 iterations. By contrast, the complete replanning needs about 500 iterations. To evaluate the overall compute time, we simulated MAV flights, disturbed every second by strong gusts of wind. On average the reoptimization finished in 18.3 % of the time a complete reoptimization required. The maximum was 28.3 % and the minimum 14.8 %. When no disturbances occur the trajectory is improved with every reoptimization step, further converging towards a local minimum.

Figure 10.21 shows the resulting trajectories in a simulated experiment where the MAV is pushed away 3 m from its current position every second. The resulting pose error is distributed to the trajectory followed by reoptimization.

In the second experiment, we evaluate reoptimization to react on new environment perceptions by placing an unknown cuboid obstacle of size $4\,\mathrm{m} \times 4\,\mathrm{m} \times 4\,\mathrm{m}$ randomly in the environment. We constrain the center point of the obstacle to lie within a corridor with radius 1 m to the line of sight between the start and goal pose of the MAV outside its sensor range. The line of sight is the best trajectory

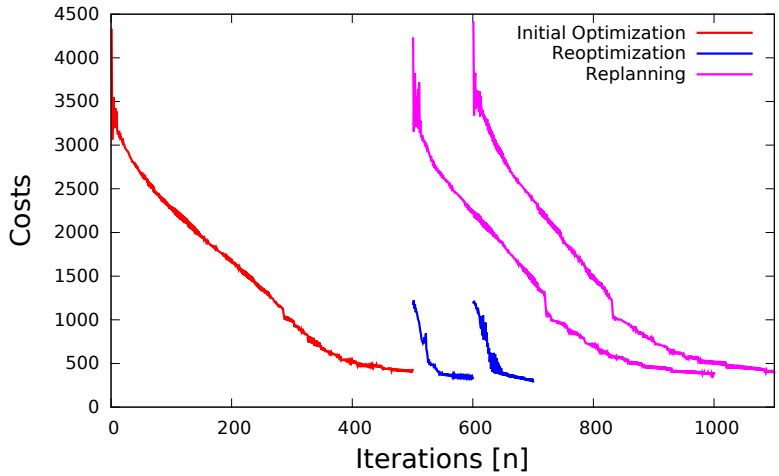


Figure 10.20: Costs per iteration during frequent trajectory optimization. Frequent reoptimization allows for quick reactions on deviations while following a trajectory. The initial trajectory is planned and optimized for 500 iterations (red). Reoptimizing the old trajectory yields a close-to-optimal new trajectory with fewer iterations than complete replanning.

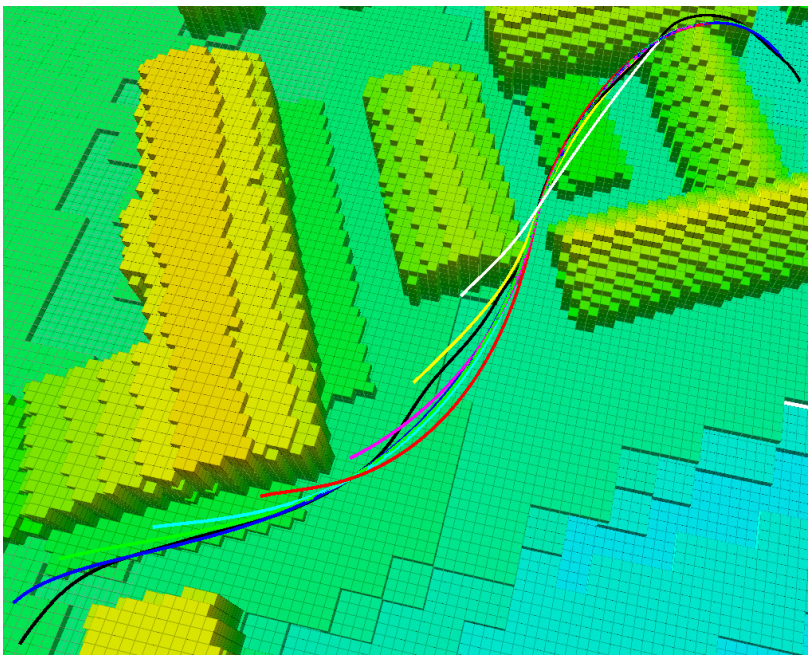


Figure 10.21: We employ frequent optimization to recover from disturbances during trajectory execution. In this example the MAV follows an initial trajectory from start to goal (black). Gusts of wind push it away from the trajectory, the colored trajectories depict the newly optimized trajectories after every gust of wind.

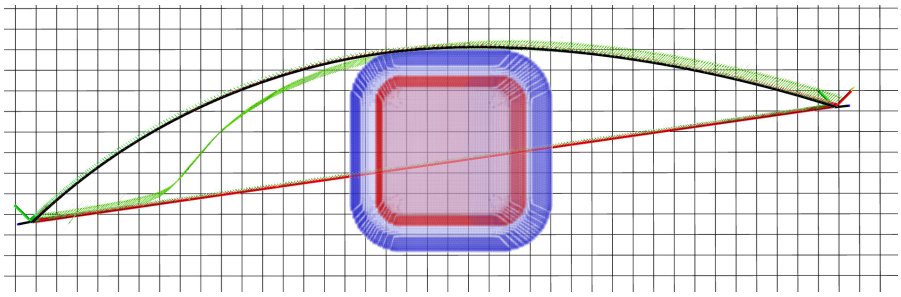


Figure 10.22: Continuous reoptimization allows for surrounding of previously unknown obstacles. The red line depicts the initial trajectory; the green arrows depict the actual flown trajectory. The black line shows the resulting optimized trajectory if the obstacle is known in advance for reference. The obstacle is depicted by the isosurfaces for minimal and safe distance. The flight direction is from left to right.

Table 10.7: ATEs during trajectory execution in real MAV experiments (in m). The ATEs are for individual flights. v_{max} is the maximum reached velocity along the trajectory in m/s.

	Spiral	Wall	Opening
ATE	0.29	0.26	0.17
RMSE	0.41	0.28	0.19
v_{max}	1.89	1.79	1.60

found by initial optimization. The scanner range of the MAV is reduced to 15 m to avoid early detection of the obstacle. Figure 10.22 shows the initial optimized trajectory and the actual flown trajectory with reoptimization for an example experiment. The experiment is shown in Video 10.4[5]. With 10 iterations per reoptimization it took on average 110 ms depending on the remaining trajectory length, with a maximum of 500 ms for the full trajectory. This is sufficient to find a feasible trajectory in a safe distance while approaching the obstacle. Further reduction of this duration is possible with multiresolution, which we did not employ here. The timing measurements and trajectory following were performed on the DJI Matrice 600 onboard PC employing the HIL simulator.

The next experiment evaluates the refinement of multiresolution trajectories during the flight employing the RotorS simulation of the *ChimneySpector* MAV. When using multiresolution, especially with a high multiresolution factor c in Equation (8.4), the resulting optimized trajectories diverge from the uniform discretized trajectory with increasing duration from the start pose. This effect is mitigated by frequent reoptimization as the diverged trajectories are refined

5 Video 10.4: www.nieuwenhuisen.de/thesis/optimization-reoptimization.mp4

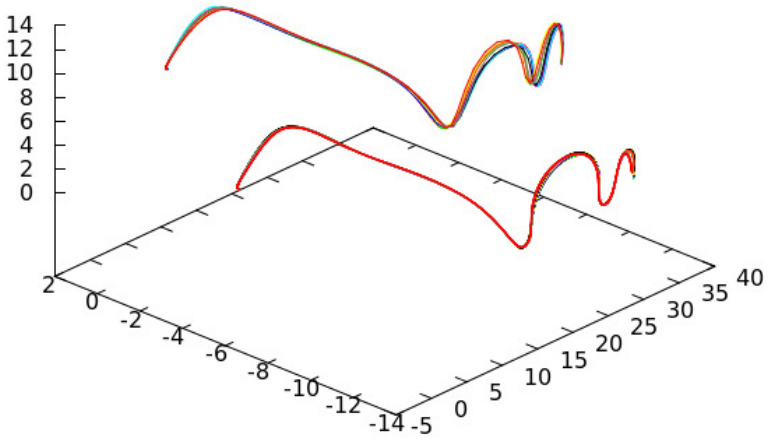


Figure 10.23: Comparison of initially planned and executed multiresolution trajectories. The top bundle of trajectories depicts the multiresolution trajectories from start (left) to goal (right) for multiresolution factors c from 0.1 to 1. The trajectories diverge with increasing duration caused by the different discretizations. Nevertheless, the bottom bundle shows that the resulting trajectories, when executed, are very similar due to frequent reoptimization during execution. For better visibility the two bundles are plotted with an artificial offset. All axes are in meters.

before the MAV executes them. For our evaluation, we reoptimized the trajectory every ten time steps, i.e., every 100 ms, during execution. The top part of Figure 10.23 shows the resulting initial trajectories with different multiresolution factors c. Plots of the actual flown trajectories are depicted in the bottom part of the figure. These trajectories are very similar to each other, showing that frequent reoptimization mitigates the effects of larger multiresolution factors facilitating faster optimizer iterations. For initial optimization, we employ 500 iteration steps as in the first experiment. Figure 10.24 shows the duration of these initial optimizations depending on the selected multiresolution factor c and the uniform discretization of the trajectory. All timings are measured on a computer with Intel Core i7 940 CPU. It can be seen that whereas the uniform time resolution has to be reduced by at least an order of magnitude to achieve acceptable optimization speed, introducing multiresolution allows for real-time replanning with high resolution in the controller prediction horizon. In our implementation, we use a multiresolution factor of 0.1. As a result, the optimization of a multiresolution trajectory with an initial resolution of 0.01 s requires approximately the same amount of

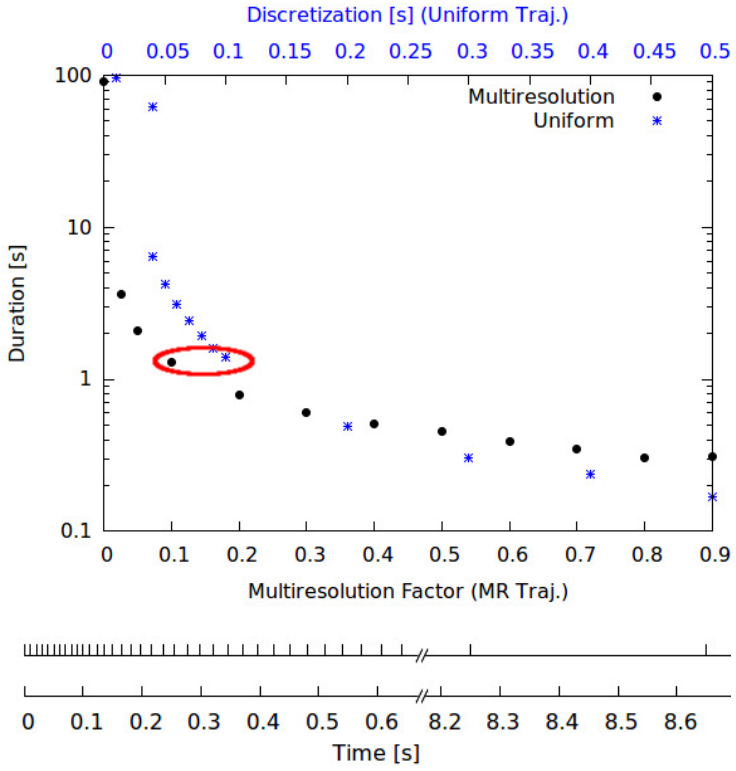


Figure 10.24: Optimization duration with and without multiresolution. Top: Black dots depict the duration of 500 optimizer iterations depending on the multiresolution factor c in Equation (8.4). Blue stars depict the duration of non-multiresolution optimization depending on the uniform discretization of the trajectory. The optimization with $c = 0.1$ requires approximately the same amount of time as a uniform discretization of 0.1 s (circled red). Bottom: Comparison of the time discretization for the circled cases. The bottom line depicts the uniform time discretization. The top line depicts the multiresolution discretization.

time as with a uniform time discretization of 0.1 s The bottom part of Figure 10.24 compares the time discretization for these cases. The bottom line depicts the uniform time discretization which is an order of magnitude coarser than needed by low-level control of the *Chimney-Spector* MAV. The top line depicts the multiresolution discretization which is much finer at the beginning and gets coarser in the future. Video 10.5[6] shows the optimization process and trajectory following with local multiresolution.

Map Updates

As a prerequisite for reoptimization the employed distance field has to be updated. Even though the implementation of efficient environment representations is out of scope of this work, we per-

6 Video 10.5: www.nieuwenhuisen.de/thesis/optimization-mr.mp4

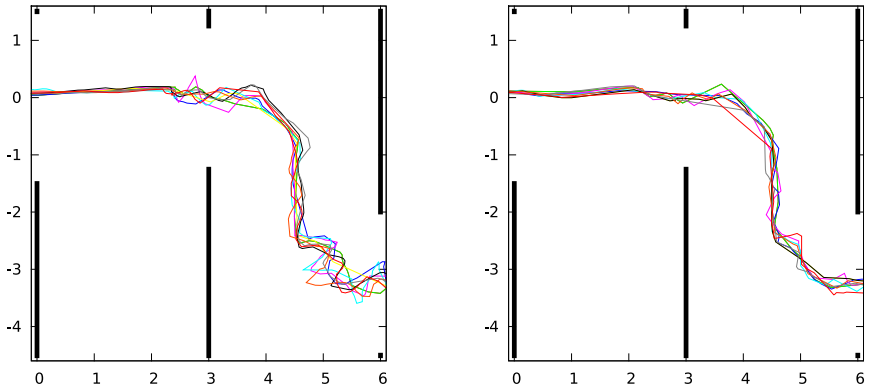


Figure 10.25: Comparison of MAV trajectories (in the xy-plane, in meters) while flying through a scene containing three walls with window-like openings (black bars). Left: Trajectories without our approach. Right: Trajectories with adaptive velocity reduction and a 1 s look-ahead. The look-ahead leads to smoother trajectories, especially in the vicinity of narrow passageways (e.g., lower right corner).

formed a proof-of-concept experiment to assess the general possibility of frequent updates. We employed a rotating LRF that—with modifications—is the main sensor for most of the used MAVs. This scanner measures 1080 distances per scan line at 40 Hz. Thus, the time window for incorporating a single scan line into the distance field is 25 ms. The maximum required time measured on the *ChimneySpector* onboard PC is 29 ms, exceeding the time window by 4 ms. However, when incorporating complete 3D scans of a cluttered environment with obstacles in any direction into an empty distance field, the updates are possible in real-time on average. The mean time per scan line for the first complete 3D scan of an environment—a half rotation of the laser scanner, yielding 20 scan lines—is approximately 10 ms. With only small changes in the environment and no movement of the MAV, distance field updates are performed in 0.2–5 ms per scan line. This shows that distance field updates are possible with the required frequency.

10.4 OBSTACLE AVOIDANCE

We evaluate the accuracy of our learned motion model and the performance and reliability of our predictive collision avoidance module in simulation and on the real MAV systems.

We tested obstacle avoidance with trajectory prediction in waypoint following scenarios in simulation.

We compared our approach with a classic potential field approach without look-ahead in a scenario containing several walls with window-like openings of different size. Furthermore, we evaluated the effect of a fixed reduction of the velocity with and without trajec-

Table 10.8: Effect of slowing down with and without prediction compared to the standard potential field approach (baseline). Fixed slow down due to predicted future forces leads to a slight increase in the flight time, but a decrease in the average repulsive force applied to the MAV. Adapting velocities according to the predicted duration of the flight until the force threshold is reached mitigate the slow down effects.

	Flight time	Experienced force	
	(in s)	average	% of baseline
Baseline	11.90 (0.5)	0.44 (0.06)	100
Slow down	12.56 (0.8)	0.43 (0.04)	98
Slow down 1 s	14.30 (1.7)	0.28 (0.04)	64
Adaptive velocity 1 s	12.90 (0.8)	0.30 (0.01)	68

tory prediction, i.e., reduced velocity if a force threshold is reached now or in the predicted time horizon, respectively. Example trajectories from the test runs are depicted in Figure 10.25. We summarize the average repulsive forces, a measure of the proximity of obstacles during a flight, and the average durations of the test runs in Table 10.8. No collisions occurred during these test runs. The prediction of the near future outcome of motion commands leads to smoother trajectories, keeping the MAV further away from obstacles than the same potential field approach without trajectory prediction while allowing comparable velocities as the classic approach.

We evaluated the learned motion model by comparing the propagated dynamic state of an MAV given a user input with ground truth data from the motion capture (MoCap) system (Figure 10.26). The predicted state of the MAV matches the real state well for time periods sufficiently long for our predictive approach.

Our collision avoidance approach runs at approximately 100 Hz on a single core of an Intel Core 2 processor, which includes data acquisition and map building. Hence, this collision avoidance algorithm is particularly well suited for MAVs with relatively small processing power or complex scenarios where the onboard computer has to carry out many other processing tasks in parallel.

We evaluated the reactive obstacle avoidance system quantitatively as part of the European Robotics Challenge, combined with stereo camera obstacle perception instead of the laser obstacle map. The MAV was steered velocity controlled through free space and towards obstacles of different shape and size. The maximum linear velocity was 1 m/s. Our *ChimneySpector* MAV was able to safely stop in front of every obstacle: In 40 % of the evaluated cases in the optimal distance range, in the other cases not more than 0.4 m off, yet still in a safe range. This error was measured with a MoCap system by the

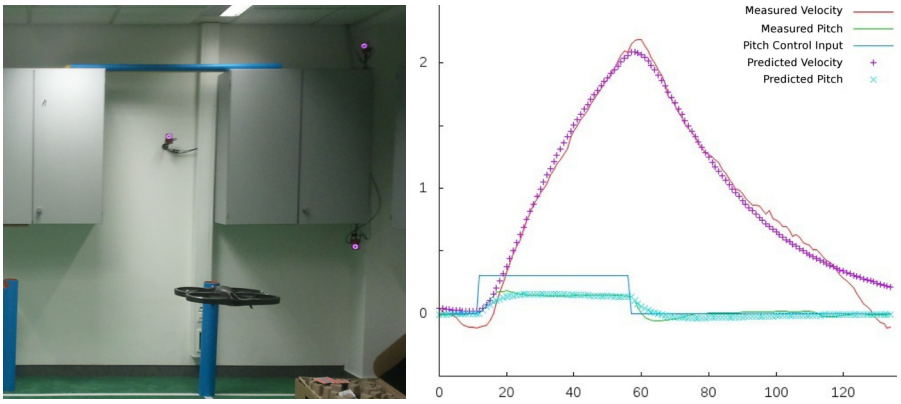


Figure 10.26: Learning a motion model. Left: Experimental setup, AR.Drone flying in a MoCap volume. Right: Comparison of predicted pitch angle (in rad) and resulting linear velocity (in m/s) with ground-truth given the same initial state and user input. The linear model is not able to model the transient responses to user input changes and the final dampening exactly, but models the system well during majors parts of the test flight.

challenge hosts and encompasses the error of visual perception and reactive obstacle avoidance.

We show qualitative results of the reactive obstacle avoidance in two supplemental videos. In Video 10.6[7], we show experiments with the *MoDCopter* MAV and our potential field-based method from Section 9.2. The prediction horizon for the MAV movement is set to zero, thus, no trajectory is rolled out in these experiments. Our obstacle avoidance approach with velocity reduction, detailed in Section 9.4, is presented in Video 10.7[8] employing the DJI Matrice 600.

10.5 INTEGRATED SYSTEMS

We evaluate our components as integral parts of several integrated MAV systems. In this section, we describe the performed experiments in indoor and outdoor scenarios.

Mapping of a Manor House

One of the first applications for the *MoDCopter* MAV was the autonomous mapping of an old manor house, as described in Section 6.1. The first performed mission included the inspection of a rainwater gutter along one side of the house. Four observation poses were defined employing the operator graphical user interface (GUI) from Loch-Dehbi et al. (2013) based on a semantic LoD 2 model. Before takeoff for the actual inspection mission, the defined observation

7 Video 10.6: www.nieuwenhuisen.de/thesis/avoidance-outdoor.mp4
8 Video 10.7: www.nieuwenhuisen.de/thesis/avoidance-indoor.mp4

poses—plus a return pose at a fixed height above the start pose—are processed by a mission control layer, incorporating the mission planner. After takeoff, the global planner begins to continuously plan paths to the next mission relevant pose. The allocentric world model employed for path planning—represented in an OctoMap—is continuously updated during the mission. Localization of the MAV is performed by using GPS. In a first inspection flight, two of these poses were inspected. Due to strong GPS drift the next allocentrically defined poses were too close to the building and could not be reached—the reactive obstacle avoidance held the MAV in a safe distance to the facade—such that these poses were skipped and the MAV returned autonomously to the takeoff position. This experiment is shown in the first part of Video 10.8[9]. In future missions, we allowed an adaption of the view poses along the surface normal to cope with inaccurate allocentric localization. We show a successful inspection mission covering the whole gutter as part of a demonstration to experts in the second part of the video.

Furthermore, we performed a fully autonomous mapping mission to acquire data for postprocessing by the 3D simultaneous localization and mapping (SLAM) system from (Droeschel et al., 2014a). The scenario involves vegetation such as trees and bushes and is difficult to traverse from all sides by humans. Therefore, a manual flight by a human pilot controlling the MAV was not possible in this scenario. The user manually defined a set of mission-relevant view poses given the coarse LoD 2 world model of the environment. In addition to poses configured in the operator GUI, some view poses were trained by manual flight to good observation poses in this experiment. When the pilot gave a command, the operator added the current MAV pose to the list of observation poses remotely These view poses where roughly specified to cover the building facade from all sides.

Figure 6.2 shows a solution for the mission described here. The MAV successfully traversed the building in all experiments without colliding with an obstacle. The overall flight duration was approximately 8 min. Figure 10.27 shows a local deviation from the allocentric path while returning to the start position employing local planning.

Processing the data to build maps is out of the scope of this thesis. The resulting 3D maps based on the data acquired during these flight experiments are reported by Droeschel et al. (2016).

Inspection of a Decommissioned Car Service Station

To evaluate the applicability of our approaches for navigation in indoor environments, we tested our components as part of an MAV mapping and inspection system. The main goal of this experiment

9 Video 10.8: www.nieuwenhuisen.de/thesis/mapping-outdoor.mp4

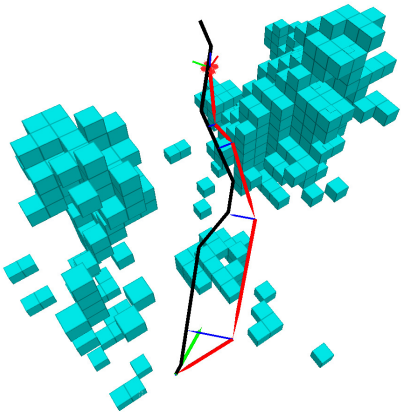


Figure 10.27: Deviation from allocentric plan. The local plan (red) is coupled with the allocentric plan (black). Based on an updated local map (cyan voxels) the MAV surrounds an obstacle locally while approaching its target position. The blue lines depict the deviation vectors at example points, the green arrow indicates the goal for local planning.

was to autonomously navigate to certain predefined waypoints in a hall, employing solely means of localization that are available in both indoor and outdoor environments. During flight, the MAV mission was to detect visual features (AprilTags (Olson, 2011)), representing interesting locations near the trajectory.

Local navigation and control were performed using visual odometry (Geiger et al., 2011) and allocentric localization was performed employing laser pose tracking (Droeschel et al., 2014b).

First, a pilot navigated the MAV manually through a hall, a garage, and an outdoor part connecting these two buildings. Subsequently, we derive an OctoMap for mission and path planning from the 3D map that has been built from the data collected during this flight. In applications where such an initial flight is not feasible, building construction plans could be used instead, similar to the coarse models in outdoor environments.

Second, we defined a mission with six observation poses—plus a return pose 2 m above the start pose—in the smaller building part, a decommissioned car service station. Our mission planner plans paths between every pair of mission poses and determines the best visiting order. After takeoff, the global planner begins to continuously plan paths to the next mission-relevant pose. The local obstacle avoidance keeps the MAV successfully away from obstacles like hanging cables and debris, lying around in the hall. In these experiments, we planned allocentric paths in a grid with a cell size of 0.5 m. An excerpt from the map and the inspection poses are shown in Figure 10.28. The MAV successfully reached all poses in the experiments without col-

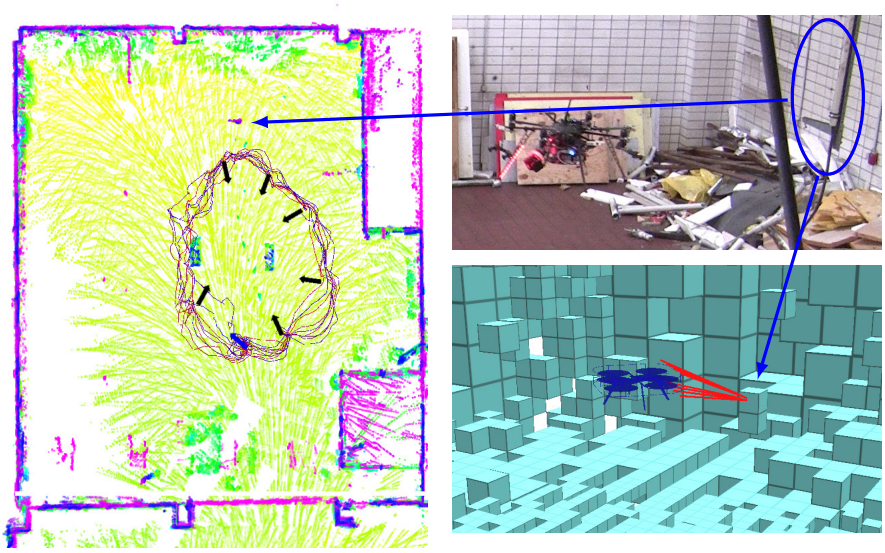


Figure 10.28: Inspection mission in a car service station. Left: The view poses (black arrows) are overlaid over the map of the area and the flown trajectories from 10 flights. The blue arrow depicts the MAV return pose. Right: A small pipe structure (circled blue) hangs from the ceiling close to one observation pose. The bottom figure shows how the obstacle is perceived by the MAV. Red lines depict the artificial repelling forces of the reactive collision avoidance.

liding with an obstacle. We show the experiment in Video 10.9[10]. In experiments without running localization module, the MAV still was able to avoid dynamic and static obstacles with the local collision avoidance layer, since this layer solely relies on egocentric velocity estimates, e.g., from an integration of visual odometry, accelerometers, and gyroscopes.

Figure 10.28 shows a situation from one of the runs where the MAV is flying towards a mission view pose. Close to the view pose, a long thin structure is hanging from the ceiling. The structure is perceived with the onboard sensors and avoided by means of artificial repelling forces. In our test setup, many smaller and larger structures obstruct the free space. In all cases, the MAV deviated from the direct path or moved away from a hover position to avoid a collision. Figure 10.29 shows the magnitudes of repelling forces while fulfilling a planned mission. The shown positions are from the ten autonomous flights shown in Figure 10.28 plus one additional mission with observation poses closer to some obstacles. Samples for illustration are taken every 500 ms.

10 Video 10.9: www.nieuwenhuisen.de/thesis/integration-indoor.mp4

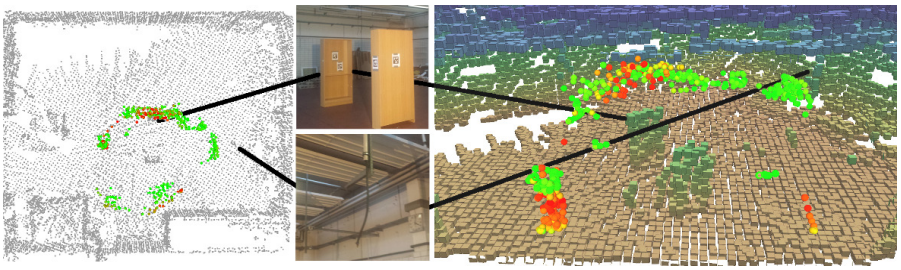


Figure 10.29: Artificial forces from the reactive collision avoidance. Shown is the strength of repelling forces pushing the MAV away from obstacles at samples along the trajectories of ten autonomous flights. Green: small magnitude / Red: large magnitude. The photos show some obstacles and their approximate position in the maps.

Chimney Inspection

We tested and demonstrated our integrated system in two close-to-application scenarios: Inspection and reconstruction of a narrow chimney mock-up and a decommissioned industrial chimney. Both scenarios were defined in close collaboration with a chimney inspection service contractor. We employed the *ChimneySpector* MAV for these flights. In Video 10.10[11] and Video 10.11[12], we show footage from our experiments.

Experiments in Chimney Mock-up

First, we used an octagonal chimney mock-up with an inradius of 1.8 m and a height of 4.4 m, shown in Figure 6.5. The mock-up consisted of eight wooden panels with styrofoam structures on the inner sides resembling the stonework and concrete patterns that can be found in many industrial chimneys. Single structure elements were of size 1.0×0.5 m and each of the stone walls—except of the wall containing an entry to the mock-up—was plastered with a single type of elements resulting in repetitive patterns. In addition, one panel carried a rusty iron surface as found in chimneys with a metal alloy on the inner side. Some bricks in the styrofoam elements were carved out to represent defects. These experiments were performed with an SR300 sensor instead of the stereo camera setup employed in later experiments. This sensor requires a distance of 0.8–1.0 m from the surface for good data acquisition. Consequently, this yielded a remaining safe navigation space with a diameter of only ≈1.2 m. The mock-up was designed and built by the chimney inspection service contractor to facilitate the transferability to real inspection applications.

11 Video 10.10: www.nieuwenhuisen.de/thesis/chimneyspector-coverage.mp4
12 Video 10.11: www.nieuwenhuisen.de/thesis/chimneyspector-inspection.mp4

2

We started with an initial coverage flight to acquire RGB-D data of the chimney surface. For surface coverage, the MAV followed a horizontally and vertically spiraling pattern with the RGB-D sensor directed to the nearest surface in order to enable loop closings in the later surface reconstruction.

The flight to acquire data for the majority of the chimney surface took seven minutes. We covered $36\,\mathrm{m}^2$ of the chimney surface with a single charge of batteries. All defects were covered by that area. The flight was fully autonomous, except for start and landing.

After the coverage flight, a downsampled version of the recorded video stream was transferred to the ground control station. Here, an operator can identify and store poses for a more detailed inspection in the video stream. In the test case, the operator could identify all ten defects in the images. The MAV poses, corresponding to the images showing defects were saved for a second flight to reinspect those defects. To exemplify the targeted inspection of previously identified defects, the MAV then autonomously planned an inspection mission and followed a path, adopting all stored poses in a useful order and held its position for several seconds to demonstrate reaching the desired position. All poses were successfully reached and the defects were clearly identified in the data captured during the second flight. After the targeted inspection, the acquired data was transferred to the ground control station for further offline processing.

On average, we acquired depth information for 95.5 % of the pixels, thus, the sensor was almost always positioned in an optimal distance to the surface.

The acquired data was used by Quenzel et al. (2018) to reconstruct the surface of the chimney walls. The surface reconstruction is out of the scope of this thesis, results of the reconstruction are presented in their article.

Experiments in a Decommissioned Industrial Chimney

Based on the experiments in the chimney mock-up, we improved our system setup and transferred our system to a scenario closer to the inspection of actual industrial chimneys as the next step. We evaluated and demonstrated our system in a chimney of a decommissioned coking plant at the Zollverein Coal Mine Industrial Complex in Essen, Germany. The total height of the chimney is $98\,\mathrm{m}$. The inradius at the bottom is $2.75\,\mathrm{m}$ and tapers to $2.2\,\mathrm{m}$ on the top. An improved camera setup allowed us to increase the distance to the surface to $1.5\,\mathrm{m}$, resulting in a larger covered area. Furthermore, we could omit the smaller vertical spirals for better coverage of the surface. Both improvements yielded a much higher surface coverage speed. The MAV flew at a maximum speed of $1.5\,\mathrm{m/s}$. In total, the area covered with a single charge of batteries could be increased to $140\,\mathrm{m}^2$ due to

Figure 10.30: Photos of warehouse. Left: Overview over the two aisles used for our evaluation. Right: Autonomous flight along an aisle for stocktaking.

the faster coverage procedure. The autonomous flight time was 3:27 minutes, plus an additional minute for manual start, landing, and hovering phases. As a result, we covered seven times the surface area per flight time than in the previous experiment.

Similar to the inspection of the chimney mock-up, an operator selected poses of interest based on the data captured during the initial flight. During the final evaluation, an operator selected three poses for reinspection. All of these poses could be reached and the defects were clearly visible in the captured video data.

The total surface of the chimney is approximately 1520 m^2, thus, we could cover the complete surface with 11 of such flights, resulting in 38:30 minutes of coverage time. With an estimated overhead of 2 min per flight for ascent, descent, and battery change, an MAV-based inspection system could cover this exemplary industrial chimney in approximately one hour.

Stocktaking in a Warehouse

We evaluated our *AIRCopter* and DJI Matrice 600 MAVs in a large, active warehouse to facilitate autonomous stocktaking, detailed in Section 6.2. The building area of the warehouse, depicted in Figure 10.30, is 100 $\mathrm{m} \times 60\,\mathrm{m}$ with approximately 12 000 m^2 storage front.

First, we evaluated the *AIRCopter* MAV as part of a demonstration to an external reviewer by flying autonomous missions in a warehouse. The MAV visits several manually defined observation poses on different heights along a shelf based on an allocentric map created with the SLAM approach from Droeschel et al. (2014a). Figure 10.31 shows one example mission. The MAV successfully accomplished multiple missions with a duration between ~2–5 min and a total trajectory length of ~40–80 m each.

To evaluate the local obstacle avoidance, we control the MAV with egocentric velocity commands, i.e., a zero velocity setpoint for movements in the plane and rotations, and a small descent velocity to keep the MAV close to the ground. The obstacle avoidance keeps the MAV

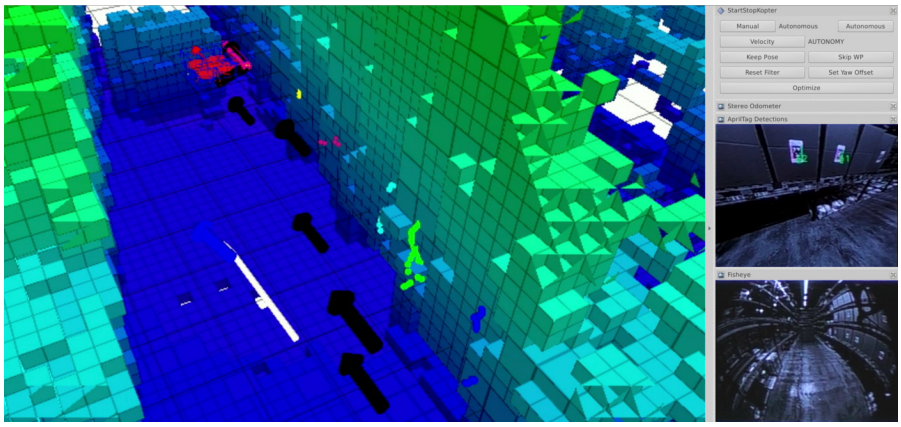


Figure 10.31: Flight operator view for stocktaking missions. The command and control interface depicts in the main window the allocentric obstacle map, the MAV pose (red shape), future mission waypoints (black arrows), obstacle-induced forces if applicable, and 3D coordinates of detected tags. An operator can switch between manual, velocity controlled, and fully autonomous operation.

at a safe distance to the ground. Figure 10.32 shows an experiment where a person approaches the MAV. The MAV avoided all static and dynamic obstacles based on the 3D laser scans. Video 10.12[13] shows the autonomous mission execution including reactive obstacle avoidance.

A second demonstration was performed with the DJI Matrice 600 MAV. In contrast to the first experiment, the mission planner now employs poses and coverage patterns requested by the warehouse management system (WMS). This is achieved by aligning our obstacle map with a semantic map containing storage units. To demonstrate autonomous stocktaking operations, we specified a mission containing the complete inventory of one shelf row and the inspection of a single storage unit in another row employing the WMS. Thus, the mission included following a coverage pattern and planning an obstacle-free path to the remote shelf row. The MAV executed this mission autonomously multiple times while avoiding static obstacles, e.g., the shelves and stock protruding from the shelves. The MAV reached velocities up to 2.1 m/s. As no dynamic obstacles above the MAV were to be expected in this demonstration, we neglected the planning with visibility constraints in favor of faster mission execution.

To test the avoidance of dynamic obstacles, we let the MAV hover at a height of 2 m above the ground after finishing the stocktaking mission. A person approached the MAV, which avoided the dynamic obstacle by means of our reactive obstacle avoidance, shown in Figure 9.6. Furthermore, a person stepped into the way of the MAV while

13 Video 10.12: www.nieuwenhuisen.de/thesis/integration-warehouse.mp4

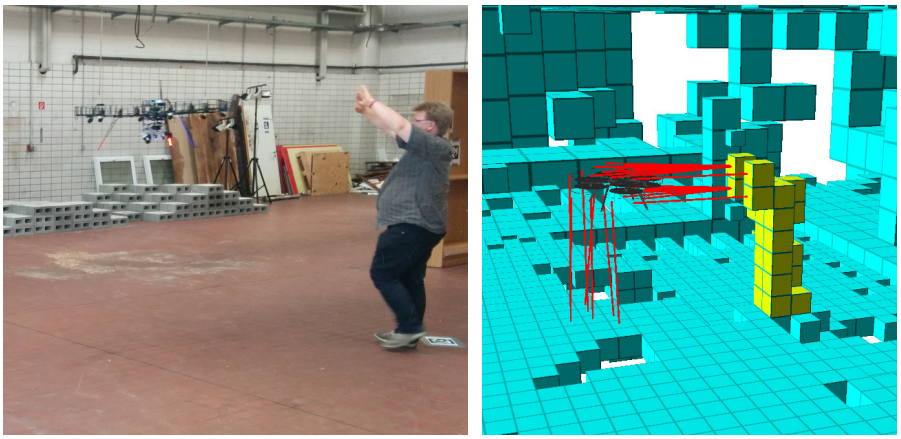


Figure 10.32: The MAV is pushed away from an approaching person and the ground by potential field-based obstacle avoidance. Red lines in the right figure depict forces induced by the local obstacle map on the MAV. Voxels corresponding to the person are colored yellow for illustration.

it approached a waypoint. The MAV stopped at a safe distance in all cases. Video 10.13[14] shows the experiments in the warehouse.

14 Video 10.13: www.nieuwenhuisen.de/thesis/integration-warehouse-m600.mp4

11 CONCLUSION

In this part of the thesis, we detailed a complete navigation hierarchy for micro aerial vehicles (MAVs) from slow deliberative planning to fast reactive obstacle avoidance. The allocentric and egocentric planning layers can be complemented by trajectory optimization to acquire dynamically feasible trajectories including velocity commands.

With tailored mission planners, we can operate MAVs in a variety of application domains. We plan shortest routes between observation poses for outdoor 3D mapping or targeted inspection in chimneys. For the mapping of chimney surfaces or autonomous stocktaking in warehouses, we plan coverage patterns suiting the requirements of onboard sensors. We combine these two approaches for the inspection of building parts, where coverage patterns are connected by planned path segments, yielding an overall shortest coverage tour.

By employing a global-to-local approach in our navigation planning pipeline, we achieve replanning frequencies that match the rate of expected changes in the environment model on different layers without losing the property of globally consistent plans. The plans on all layers are updated with appropriate frequencies and newly acquired information can be incorporated into globally optimal plans after a short time period.

We presented an approach to speed up trajectory generation for MAVs based on a grid-based path planner followed by trajectory optimization. The optimized trajectories are smooth in position, velocity, and acceleration of the MAV, facilitating higher possible execution speed. Key for accelerating the optimization process is a good guess for an initial trajectory and its derivatives. We employ a simple motion model derived from the acceleration capabilities of the MAV, combined with cubic spline interpolation and clothoid-based continuous curvature transition segments (CCTSs). The local modifications by the CCTSs smooth the trajectory while ensuring the guaranteed obstacle-freeness of the initial allocentric plan. This significantly reduces acceleration peaks and thus expedites the convergence of trajectories to a locally optimal and globally feasible trajectory.

The trajectory optimization is sped up by means of a local multiresolution discretization of the time dimension along trajectories. This makes frequent reoptimization feasible and allows to plan trajectories with the same time discretization as the low-level controller generating attitude and thrust setpoints for the MAV.

Planning MAV trajectories imposes new challenges due to the ability for omnidirectional movement not only in the plane, but also in

height. Whereas the environment for ground vehicles can be covered relatively well with onboard obstacle sensors, the movement directions combined with a limited payload prohibits complete and high-frequency coverage of the space around an MAV for many applications. We extended our combined planning and trajectory optimization approach with the ability to plan allocentric paths within the field of view (FoV) of planar omnidirectional 3D sensors with a restricted vertical apex angle. The resulting optimized trajectories are thus safe and dynamically feasible.

We showed that an MAV is able to follow the optimized trajectories with two different model predictive controllers (MPCs) in real world experiments with our *ChimneySpector* and Matrice 600 MAVs and in simulation employing a DJI flight control unit in the loop.

We developed a fast, reactive collision avoidance layer to quickly react on new measurements of nearby obstacles. It serves as a safety measure between higher planning layers or commands given by a human pilot and the low-level control layer of the MAV. We investigated how the limitations of standard potential field-based approaches—making the assumption that the motion of a vehicle can be changed immediately at any position in the field—can be overcome. One possibility is the prediction of the future trajectory resulting from the current dynamic state and the artificial potential field. This leads to safer and smoother trajectories for a multicopter based on learned motion models.

We simplified and robustified this approach for safe operation in the vicinity of obstacles by influencing velocities on the commanded flight direction in addition to the perceived obstacles. This leads to more stable behavior for conservative flight dynamics as required by most of our applications. The efficacy of our obstacle avoidance is shown in many experiments with our integrated systems and quantitatively as part of a competition.

Overall, the components developed within this thesis have been employed in many real MAV experiments as part of integrated systems where the only manual interactions were the starting and landing phases. Thus, the systems are able to complete missions fully autonomous. We demonstrated that multilayered navigation planning results in a high capability to cope with dynamically changing environments and perpetually new obstacle perceptions. The applicability and reliability of our components as part of several integrated systems was proven by I) autonomously accomplishing a mission to map a building and its surroundings while flying in the vicinity of buildings, trees, cables and other potential obstacles and sources for collision, II) flying in a warehouse for stocktaking including the avoidance of dynamic obstacles, and III) capturing data for chimney inspection by smooth coverage flights in a chimney mock-up and in

an industrial chimney and reinspecting the identified defects at more detail in a second mission.

Part III

DISCUSSION AND FUTURE WORK

DISCUSSION AND FUTURE WORK

In this thesis, we covered different approaches to operate and navigate in dynamic environments for mobile ground robots and micro aerial vehicles (MAVs). A key for the safe and efficient operation of robot platforms in dynamic environments is the ability to react to changes in an adequate timeframe. The duration of an adequate timeframe is defined by multiple factors: the expected frequency of changes in the environment, the closeness of the robot to these changes, hazards caused by the dynamics, and the time-sensitiveness of the task to fulfill. Even though stopping and planning a new action sequence in case of changes might look like an option always available in the first place, this behavior might become dangerous or at least disadvantageous in various situations. For example, dynamic obstacles can come closer and the robot has to actively perform actions to avoid a collision. In other situations, e.g., soccer games, waiting too long before selecting the next actions can hinder the robot to fulfill its task successfully as the opponent can take advantage of these delays and take the ball or even score a goal.

Already, these two examples show, that it can be necessary to select the next actions quickly even though they are not optimal. To avoid suboptimal performance, the quick action selection has to be complemented by close to optimal long-term planning. In this thesis, we investigated different approaches to combine quick action selection with deliberative long-term planning.

One approach investigated here is to reduce the time required for long-term planning to the timeframe required to select actions. This comes generally at the price of a reduced planning accuracy, e.g., coarse environment representations or long steps between consecutive actions. These effects can be mitigated by accepting these coarse plans as long as they are not to be executed in the near future and use exact plans in the proximity of the robot. Local multiresolution planning, detailed in Section 2.3, is one solution to get a continuous transition between high- and low-resolution plans. Examples for this approach are our planner for soccer robots (see Chapter 3), the egocentric planning layer in our MAV navigation hierarchy (see Section 7.2), and the continuous trajectory reoptimization, detailed in Section 8.4. The main advantage of employing local multiresolution planning is that a consistent sequence of actions—including all transitions—is planned based on common data. Nevertheless, the major disadvantage is that the planning horizon is restricted as otherwise the future planning steps become too coarse to be beneficial. Furthermore, the

planning duration will exceed the allotted timeframe eventually with increasing planning horizons.

Another option is to split the planning problem into layers with different properties on optimality and achievable action selection frequency. These layers can be loosely or tightly coupled. A typical example for loose coupling is that long-term plans define a coarse list of intermediate points that the short-term action selection has to reach to avoid local minima. We employ this loose coupling in our MAV navigation hierarchy, e.g., between our mission planner for outdoor 3D mapping (see Section 6.1) and the allocentric planner. The order of observation poses is optimal w.r.t. the initial world model, but the allocentric planner, detailed in Section 7.1, can deviate from the assumed initial plan in any way. A prerequisite for loose coupling is that the world model of the more local planner is at least as complete as the model of the more global planner. Otherwise, the more local planner could plan infeasible shortcuts.

Tighter coupling can be achieved by penalizing deviations from the long-term plan. In this case the local plan can only deviate from the global plan if necessary, e.g., if the action sequence defined by the global plan can not be executed based on an updated or more exact world model. The requirement for a complete world model for the local planner can be relaxed with tighter coupling as infeasible detours can only be planned if the allocentric plan is no longer valid. Such a failure condition has to be resolved by new long-term planning on an updated world model. We employ this tighter coupling in our egocentric MAV planning layer, detailed in Section 7.2.

This layered approach to navigation allows us to be as precise and optimal as necessary on the top layers at a price of long planning times and as quick as necessary on the lower layers at the price of locality. By adding more layers these properties can be traded off gradually. In contrast to local multiresolution planning, multiple representations of the world have to be maintained suiting the individual layers of the hierarchy.

For the task of bin-picking, detailed in Chapter 4, we split the planning problem into phases—from planning feasible grasps to the arm motion required for removal of an object from the box—instead of hierarchical layers. Phases that can be preprocessed offline are precalculated independent of the actual problem. All other phases are ordered according to the probability of a failure in the corresponding phase. This allows us to avoid planning for phases that are costly, but likely to succeed before valid solutions to the more critical phases have been found. For example, the final part of a grasping motion is much more likely to fail than the much longer trajectory before.

All the planners mentioned above have in common that the objective is to find a less costly solution to a given problem. One major objective is to minimize the resulting overall path length while avoid-

ing obstacles. Depending on the application, also other objectives play a role. In this thesis, we extended the objective functions by restricting the path to stay locally in the sensor field of view (FoV) to ensure safe obstacle avoidance. Furthermore, we presented cost models to incorporate wind, the ability to communicate, and structure required for laser-based localization into our planner objectives.

In Chapter 3, we showed that neglecting the dynamics in the world can lead to longer paths to reach a target pose. We incorporated the expected movement of another moving agent into our world model to avoid necessary changes to the path in the near future. To reduce the planning complexity, we proposed to reduce the time dimension to a single value per geometric state and investigated the effects.

To facilitate smooth robot trajectories geometric planning is not sufficient. Instead of full kinodynamic planning, we employ geometric planning in conjunction with trajectory optimization. This combination has the advantage that the planning problem itself stays tractable and still optimal given the reduced dimensionality. Still, the resulting trajectories can be executed by a robot without slowing down at intermediate waypoints. Further possible options are discussed in the next section.

OUTLOOK AND FUTURE WORK

The methods presented in this thesis yield working solutions for the presented application domains. Nevertheless, ample directions for future research extending our ideas remain open. Here, we suggest some potential follow-up improvements building upon this thesis.

So far, we addressed the dynamics in the environment by fast and frequent replanning of paths and trajectories in the case of MAV navigation. As one starting point to improve the resulting plans, we propose to incorporate planning in the time dimension with explicit representation of expected obstacle movements. To achieve this, the approximation techniques developed for the soccer domain from Chapter 3 could be ported to the employed planners for MAV planning. Instead of intention projection—or projecting intentions only in special cases—the movement of obstacles can be estimated by tracking and predicting their current motion. Ideally, a combination with semantic classes—cars move differently than pedestrians or animals—would yield a good estimate of the environment changes in the near future.

In Chapter 8, we generate dynamically feasible trajectories that can be followed by the low-level controller open-loop. Nevertheless, this is achieved by conservative assumptions about the performance of the controller, e.g., not exhausting the dynamic limits of the MAV. We propose to integrate the trajectory rollout part of the controller from Beul and Behnke (2017) into the planning layers such that the resulting trajectories would be much closer to the achievable flight dy-

namics. One option would be to connect states in a kinodynamic planner by employing trajectories generated by the controller and check for obstacle-freeness of these rollouts. To keep the planning problem tractable, hierarchical planning and multiresolution techniques are inevitable. A geometric coarse path planned by our search-based planning hierarchy could define a safe corridor in which a kinodynamic trajectory could be planned by, e.g., sampling-based planning. The sampling resolution would correspond to the local environment complexity.

Another option without explicit integration of the controller into the planners is to learn cost functions based on sampled trajectory rollouts. This would lead to plans that could be followed at higher velocities as the MAV can, e.g., lunge out before or after direction changes. Maximum velocities in the intermediate waypoints would be calculated in a post-processing step in this scenario. Apart from the characteristics of the underlying controller other objectives could be learned. Gräve (2015) teaches a robot arm tasks like grasping an object by demonstration. Similar techniques could be applied on MAVs to learn to fly like a human pilot.

Bottlenecks of high-dimensional large-scale planning are the required space to store planning representations and the planning time to cover the large state space. Even though modern hardware mitigates these effects partially, the problem instances one tries to solve grow. In Chapter 7, e.g., we plan in the sensor FoV. This adds a rotation dimension. Planning with explicit representation of obstacles adds a time dimension. Thus, approximations for some dimensions have to be found, similar to the velocity obstacle in Chapter 3. To combine the power of high-dimensional planning with the efficient use of resources of the approximations, both could be combined by adapting the planning dimensionality locally, complementary to adapting the resolution depending on the local complexity of the environment.

For planning and optimization of trajectories with sensor visibility constraints, we make the assumption that the sensor FoV is only dependent on the MAV 4D pose (x, y, z, yaw). This approximation is only valid if accelerations during the flight are considerably low as the accelerations directly affect the MAV attitude. As we already optimize trajectories including accelerations, we propose to employ this information to modify the sensor FoV accordingly along the trajectory. This would ensure safe obstacle avoidance also for very dynamic flights.

LIST OF FIGURES

LIST OF TABLES

LIST OF VIDEOS

Download links and players for all videos listed here can be found online at www.nieuwenhuisen.de/thesis.

ACRONYMS

AMCL adaptive Monte Carlo localization

ATE absolute trajectory error

BIT batch informed trees

CCTS continuous curvature transition segment

CHOMP covariant Hamiltonian optimization and motion planning

CityGML City Geography Markup Language

DEM digital elevation model

DoF degree of freedom

EKF extended Kalman filter

EuRoC European robotics challenges

FoV field of view

GNSS global navigation satellite system

GPS global positioning system

GUI graphical user interface

HIL hardware-in-the-loop

IMU inertial measurement unit

ITOMP incremental trajectory optimization for real-time replanning

KPIECE kinodynamic motion planning by interior-exterior cell exploration

LDS linear dynamical system

LoD level of detail

LRF laser rangefinder

LUT look up table

MAV micro aerial vehicle

MoCap motion capture

MPC model predictive controller

OMPL Open Motion Planning Library

RFID radio frequency identification

ROS Robot Operating System

RRBT rapidly-exploring random belief tree

RRT rapidly-exploring random tree

SLAM simultaneous localization and mapping

SPARTAN sparse tangential network

SPL Standard Platform League

STOMP stochastic trajectory optimization for motion planning

TSP traveling salesman problem

UAV unmanned aerial vehicle

WMS warehouse management system

BIBLIOGRAPHY

Achtelik, Markus, Abraham Bachrach, Ruijie He, Sam Prentice, and Nicholas Roy (2009). "Autonomous navigation and exploration of a quadrotor helicopter in GPS-denied indoor environments." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Achtelik, Markus, Simon Lynen, Stephan Weiss, Margarita Chli, and Roland Siegwart (2014). "Motion- and uncertainty-aware path planning for micro aerial vehicles." In: *Journal of Field Robotics* 31.4, pp. 676–698. ISSN: 1556-4967.

Ackerman, Evan (2014). "When drone delivery makes sense." In: *Spectrum, IEEE* 25.

Andert, Franz, Florian-Michael Adolf, Lukas Goormann, and Jörg S. Dittrich (2010). "Autonomous vision-based helicopter flights through obstacle gates." In: *Selected papers from the 2nd International Symposium on UAVs*. Springer, pp. 259–280.

Andreasson, Henrik, Jari Saarinen, Marcello Cirillo, Todor Stoyanov, and Achim J. Lilienthal (2014). "Drive the drive: From discrete motion plans to smooth drivable trajectories." In: *Robotics* 3.4, pp. 400–416.

Applegate, David, Robert Bixby, Vasek Chvatal, and William Cook (2006). *Concorde TSP solver*.

Baca, Tomas, Daniel Hert, Giuseppe Loianno, Martin Saska, and Vijay Kumar (2018). "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Beer, Kristina (2016). *DHL: Erfolgreiche Tests mit Paketkopter, der selbstständig be- und entlädt*. URL: www.heise.de/newsticker/meldung/DHL-Erfolgreiche-Tests-mit-Paketkopter-der-selbststaendig-be-und-entlaedt-3198938.html.

Beetz, Michael, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth (2011). "Robotic roommates making pancakes." In: *Proceedings of the International Conference on Humanoid Robots*.

Behnke, Sven (2004). "Local multiresolution path planning." In: *RoboCup 2003: Robot Soccer World Cup VII*, pp. 332–343.

Behnke, Sven and Jörg Stückler (2008). "Hierarchical reactive control for humanoid soccer robots." In: *International Journal of Humanoid Robots* 5.3, pp. 375–396.

Bentley, Jon Louis (1975). "Multidimensional binary search trees used for associative searching." In: *Communications of the ACM* 18.9, pp. 509–517.

Berner, Alexander, Jun Li, Dirk Holz, Jörg Stückler, Sven Behnke, and Reinhard Klein (2013). "Combining contour and shape primitives for object detection and pose estimation of prefabricated parts." In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*.

Beul, Marius and Sven Behnke (2016). "Analytical time-optimal trajectory generation and control for multirotors." In: *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*.

— (2017). "Fast full state trajectory generation for multirotors." In: *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*.

Beul, Marius, David Droeschel, Matthias Nieuwenhuisen, Jan Quenzel, Sebastian Houben, and Sven Behnke (2018). "Fast autonomous flight in warehouses for inventory applications." In: *IEEE Robotics and Automation Letters* 3 (4), pp. 3121–3128.

Beul, Marius, Nicola Krombach, Yongfeng Zhong, David Droeschel, Matthias Nieuwenhuisen, and Sven Behnke (2015). "A high-performance MAV for autonomous navigation in complex 3D environments." In: *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*.

Bipin, Kumar, Vishakh Duggal, and K. Madhava Krishna (2014). "Autonomous navigation of generic quadrocopter with minimum time trajectory planning and control." In: *Proceedings of the IEEE International Conference on Vehicular Electronics and Safety (ICVES)*.

Blanco, Jose Luis and Pranjal Kumar Rai (2014). *nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) wih KD-trees*. https://github.com/jlblancoc/nanoflann.

Bley, Florian, Volker Schmirgel, and Karl-Friedrich Kraiss (2006). "Mobile manipulation based on generic object knowledge." In: *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*.

Borenstein, Johann and Yoram Koren (1991). "The vector field histogram—fast obstacle avoidance for mobile robots." In: *IEEE Transactions on Robotics and Automation* 7.3, pp. 278–288. ISSN: 1042-296X.

Borst, Christoph, M. Fischer, and Gerd Hirzinger (1999). "A fast and robust grasp planner for arbitrary 3D objects." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Bouabdallah, Samir, Pierpaolo Murrieri, and Roland Siegwart (2004). "Design and control of an indoor micro quadrotor." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Chang, Lillian, Joshua R. Smith, and Dieter Fox (2012). "Interactive singulation of objects from a pile." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Chitta, Sachin, E. Gil Jones, Matei Ciocarlie, and Kaijen Hsiao (2012). "Perception, planning, and execution for mobile manipulation in unstructured environments." In: *IEEE Robotics & Automation Magazine* 19.2, pp. 58–71.

Cohen, Benjamin J., Gokul Subramanian, Sachin Chitta, and Maxim Likhachev (2011). "Planning for manipulation with adaptive motion primitives." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Costante, Gabriele, Jeffrey Delmerico, Manuel Werlberger, Paolo Valigi, and Davide Scaramuzza (2018). "Exploiting photometric information for planning under uncertainty." In: ed. by Antonio Bicchi and Wolfram Burgard, pp. 107–124.

Dijkstra, Edsger W. (1959). "A note on two problems in connexion with graphs." In: *Numerische Mathematik* 1.1, pp. 269–271.

DJI (2017a). *Mavic Pro user manual v2.0*.

— (2017b). *Phantom 4 Pro/Pro+ user manual v1.2*.

Domae, Yukiyasu, Haruhisa Okuda, Yuichi Taguchi, Kazuhiko Sumi, and Takashi Hirai (2014). "Fast graspability evaluation on single depth maps for bin picking with general grippers." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Droeschel, David, Matthias Nieuwenhuisen, Marius Beul, Dirk Holz, Jörg Stückler, and Sven Behnke (2016). "Multi-Layered Mapping and Navigation for Autonomous Micro Aerial Vehicles." In: *Journal of Field Robotics* 33.4, pp. 451–475.

Droeschel, David, Michael Schreiber, and Sven Behnke (2013). "Omnidirectional perception for lightweight UAVs using a continuous rotating laser scanner." In: *International Arch. Photogramm. Remote Sens. Spatial Inf. Sci. (ISPRS)*. Vol. XL-1/W2, pp. 107–112.

Droeschel, David, Jörg Stückler, and Sven Behnke (2014a). "Local multi-resolution surfel grids for MAV motion estimation and 3D Mapping." In: *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*.

— (2014b). "Local multiresolution representation for 6D motion estimation and mapping with a continuously rotating 3D laser scanner." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Eling, Christian, Lasse Klingbeil, Markus Wieland, and Heiner Kuhlmann (2013). "A precise position and attitude determination system for lightweight unmanned aerial vehicles." In: *International Arch. Photogramm. Remote Sens. Spatial Inf. Sci. (ISPRS)*. Vol. XL-1/W2, pp. 113–118.

Englot, Brendan and Franz Hover (2010). "Inspection planning for sensor coverage of 3D marine structures." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Fang, Zheng, Shichao Yang, Sezal Jain, Geetesh Dubey, Stephan Roth, Silvio Maeta, Stephen Nuske, Yu Zhang, and Sebastian Scherer (2017). "Robust autonomous flight in constrained and visually degraded shipboard environments." In: *Journal of Field Robotics* 34.1, pp. 25–52.

Ferguson, David and Anthony Stentz (2006). "Multi-resolution field D*." In: *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*.

Fischler, Martin A. and Robert C. Bolles (1981). "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography." In: *Communications of the ACM* 24.6, pp. 381–395.

Florence, Pete, John Carter, and Russ Tedrake (2016). "Integrated perception and control at high speed: evaluating collision avoidance maneuvers without maps." In: *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

Fornberg, Bengt (1988). "Generation of finite difference formulas on arbitrarily spaced grids." In: *Mathematics of computation* 51.184, pp. 699–706.

Fox, Dieter, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun (1999). "Monte Carlo localization: Efficient position estimation for mobile robots." In: *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Fox, Dieter, Wolfram Burgard, and Sebastian Thrun (1997). "The dynamic window approach to collision avoidance." In: *IEEE Robotics & Automation Magazine* 4 (1), pp. 23–33.

Fraichard, Thierry and Alexis Scheuer (2004). "From Reeds and Shepp's to continuous-curvature paths." In: *IEEE Transactions on Robotics* 20.6, pp. 1025–1035.

Furrer, Fadri, Michael Burri, Markus Achtelik, and Roland Siegwart (2016). "RotorS–A modular Gazebo MAV simulator framework." In: *Robot Operating System (ROS): The complete reference*. Ed. by Anis Koubaa. Vol. 1. Chap. 23, pp. 595–625.

Ge, Shuzhi Sam and Yun J. Cui (2002). "Dynamic motion planning for mobile robots using potential field method." In: *Autonomous Robots* 13.3, pp. 207–222.

Geiger, Andreas, Julius Ziegler, and Christoph Stiller (2011). "StereoScan: Dense 3D reconstruction in real-time." In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*.

Gräve, Kathrin (2015). "Lernen komplexer Aufgaben aus Demonstration und eigener Erfahrung." PhD. Mathematisch-Naturwissenschaftliche Fakultät, Universität Bonn.

Green, William E. and Paul Y. Oh (2008). "Optic-flow-based collision avoidance." In: *IEEE Robotics & Automation Magazine* 15.1, pp. 96–103.

Gröger, Gerhard, Thomas H. Kolbe, Angela Czerwinski, and Claus Nagel (2008). *OpenGIS city geography markup language (CityGML) encoding standard*. Standard. Open Geospatial Consortium Inc.

Gross, Horst-Michael, Hans-Joachim Boehme, Christof Schroeter, Steffen Mueller, Alexander Koenig, Erik Einhorn, Christian Martin, Matthias Merten, and Andreas Bley (2009). "TOOMAS: Interactive shopping guide robots in everyday use - final implementation and experiences from long-term field trials." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Grzonka, Slawomir, Giorgio Grisetti, and Wolfram Burgard (2012). "A fully autonomous indoor quadrotor." In: *IEEE Transactions on Robotics* 28.1, pp. 90–100.

Gupta, Megha and Gaurav S. Sukhatme (2012). "Using manipulation primitives for brick sorting in clutter." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Hart, Peter E., Nils J. Nilsson, and Bertram Raphael (1968). "A formal basis for the heuristic determination of minimum cost paths." In: *IEEE Transactions on Systems Science and Cybernetics* 4.2, pp. 100–107.

Heester, Robert D., Murat Cetin, Chetan Kapoor, and Delbert Tesar (1999). "A criteria-based approach to grasp synthesis." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Heng, Lionel, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys (2014). "Autonomous visual mapping and exploration with a micro aerial vehicle." In: *Journal of Field Robotics* 31.4, pp. 654–675. ISSN: 1556-4967.

Hernandez, Carlos, Mukunda Bharatheesha, Wilson Ko, Hans Gaiser, Jethro Tan, Kanter van Deurzen, Maarten de Vries, Bas van Mil, Jeff van Egmond, Ruben Burger, Mihai Morariu, Jihong Ju, Xander Gerrmann, Ronald Ensing, Jan van Frankenhuyzen, and Martijn Wisse (2017). "Team Delft's robot winner of the Amazon Picking Challenge." In: *RoboCup 2016: Robot World Cup XX*. Ed. by Sven Behnke, Raymond Sheh, Sanem Sarıel, and Daniel D. Lee. Springer International Publishing, pp. 613–624.

Holz, Dirk, Matthias Nieuwenhuisen, David Droeschel, Michael Schreiber, and Sven Behnke (2013). "Towards multimodal omnidirectional obstacle detection for autonomous unmanned aerial vehicles." In: *International Arch. Photogramm. Remote Sens. Spatial Inf. Sci. (ISPRS)*. Vol. XL-1/W2, pp. 201–206.

Holz, Dirk, Angeliki Topalidou-Kyniazopoulou, Francesco Rovida, Mikkel Rath Pedersen, Volker Krüger, and Sven Behnke (2015). "A skill-based system for object perception and manipulation for automating kitting tasks." In: *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*.

Hornung, Armin, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard (2013). "OctoMap: An efficient probabilistic 3D mapping framework based on octrees." In: *Autonomous Robots*. DOI: 10.1007/s10514-012-9321-0.

Hößler, Tom and Tom Landgraf (2014). "Automated traffic analysis in aerial images." In: *Proceedings of the International Conference on Computer Vision and Graphics (ICCVG)*, pp. 262–269.

Hrabar, Stefan, Gaurav Sukhatme, Peter Corke, Kane Usher, and Jonathan Roberts (2005). "Combined optic-flow and stereo-based navigation of urban canyons for a UAV." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Ikeuchi, Katsushi, Berthold K.P. Horn, Shigemi Nagata, Tom Callahan, and Oded Feirigold (1983). "Picking up an object from a pile of objects." In: *Proceedings of the International Symposium on Robotics Research*.

Israelsen, Jason, Matt Beall, Daman Bareiss, Daniel Stuart, Eric Keeney, and Jur van den Berg (2014). "Automatic collision avoidance for manually tele-operated unmanned aerial vehicles." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Jiang, Guangying and Richard Voyles (2013). "Hexrotor UAV platform enabling dextrous aerial mobile manipulation." In: *Proceedings of the International Micro Air Vehicle Conference and Flight Competition (IMAV)*.

Johnson, Eric N. and John G. Mooney (2014). "A comparison of automatic nap-of-the-earth guidance strategies for helicopters." In: *Journal of Field Robotics* 31.4, pp. 637–653. ISSN: 1556-4967.

Jones, Joseph (2006). "Robots at the tipping point: The road to iRobot Roomba." In: *IEEE Robotics & Automation Magazine* 13.1, pp. 76–78.

Kaden, Steffen, Heinrich Mellmann, Marcus Scheunemann, and Hans-Dieter Burkhard (2013). "Voronoi based strategic positioning for robot soccer." In: *Proceedings of the International Workshop on Concurrency, Specification and Programming (CS&P)*.

Kaelbling, Leslie P. and Tomás Lozano-Pérez (2011). "Hierarchical task and motion planning in the now." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Kalakrishnan, Mrinal and Ken Anderson (2009). *MoveIt: Propagation distance field*. Online available: github.com/ros-planning/moveit_core.

Kalakrishnan, Mrinal, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal (2011). "STOMP: Stochastic trajectory optimization for motion planning." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Kamel, Mina, Thomas Stastny, Kostas Alexis, and Roland Siegwart (2017). "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system." In: *Robot Operating System (ROS): The complete reference (Volume 2)*. Ed. by Anis Koubaa. Springer, pp. 3–39.

Karkowski, Philipp and Maren Bennewitz (2016). "Real-time footstep planning using a geometric approach." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Khatib, Oussama (1999). "Mobile manipulation: The robotic assistant." In: *Journal of Robotics and Autonomous Systems* 26 (2–3), pp. 175–183.

Klingbeil, Ellen, Deepak Rao, Blake Carpenter, Varun Ganapathi, Andrew Y. Ng, and Oussama Khatib (2011). "Grasping with application to an autonomous checkout robot." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Klingbeil, Lasse, Matthias Nieuwenhuisen, Johannes Schneider, Christian Eling, David Droeschel, Dirk Holz, Thomas Läbe, Wolfgang Förstner, Sven Behnke, and Heiner Kuhlmann (2014). "Towards autonomous navigation of an UAV-based mobile mapping system." In: *Proceedings of the International Conference on Machine Control & Guidance (MCG)*.

Koenig, Nathan and Andrew Howard (2004). "Design and use paradigms for Gazebo, an open-source multi-robot simulator." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Koenig, Sven and Maxim Likhachev (2005). "Fast replanning for navigation in unknown terrain." In: *IEEE Transactions on Robotics* 21.3, pp. 354–363.

Lagoudakis, Michael G. and Anthony S. Maida (1999). "Neural maps for mobile robot navigation." In: *Proceedings of the International Joint Conference on Neural Networks*.

Lan, Menglu, Shupeng Lai, Yingcai Bi, Hailong Qin, Jiaxin Li, Feng Lin, and Ben M. Chen (2016). "BIT*-based path planning for micro aerial vehicles." In: *Proceedings of the Annual Conference of the IEEE Industrial Electronics Society (IECON)*.

Laskey, Michael, Jonathan Lee, Caleb Chuck, David Gealy, Wesley Hsieh, Florian T Pokorny, Anca D Dragan, and Ken Goldberg (2016). "Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations." In: *IEEE International Conference on Automation Science and Engineering (CASE)*.

LaValle, Steven M. (1998). *Rapidly-exploring random trees: A new tool for path planning*. Tech. rep. Computer Science Dept., Iowa State University.

LaValle, Steven M. and James J. Kuffner (2001). "Randomized kinodynamic planning." In: *The International Journal of Robotics Research* 20.5, pp. 378–400.

Likhachev, Maxim, Geoffrey J. Gordon, and Sebastian Thrun (2004). "ARA*: Anytime A* with provable bounds on sub-optimality." In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 767–774.

Lin, Shen and Brian Wilson Kernighan (1973). "An Effective Heuristic Algorithm for the Traveling-Salesman Problem." In: *Operations Research* 21.2, pp. 498–516.

Liu, Ming-Yu, Oncel Tuzel, Ashok Veeraraghavan, Yuichi Taguchi, Tim K Marks, and Rama Chellappa (2012). "Fast object localization and pose estimation in heavy clutter for robotic bin picking." In: *The International Journal of Robotics Research* 31.8, pp. 951–973.

Loch-Dehbi, Sandra, Youness Dehbi, and Lutz Plümer (2013). "Stochastic reasoning for UAV supported reconstruction of 3D building models." In: *International Arch. Photogramm. Remote Sens. Spatial Inf. Sci. (ISPRS)*. Vol. XL-1/W2, pp. 257–261.

Longega, L., Stefano Panzieri, Federica Pascucci, and Giovanni Ulivi (2003). "Indoor robot navigation using log-polar local maps." In: *Proceedings of the International IFAC Symposium on Robot Control (SyRoCo)*.

Lozano-Pérez, Tomás and Michael A. Wesley (1979). "An algorithm for planning collision-free paths among polyhedral obstacles." In: *Communications of the ACM* 22.10, pp. 560–570.

MacAllister, Brian, Jonathan Butzke, Alex Kushleyev, Harsh Pandey, and Maxim Likhachev (2013). "Path planning for non-circular micro aerial vehicles in constrained environments." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Maciejewski, Anthony A and Charles A Klein (1985). "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments." In: *The International Journal of Robotics Research* 4.3, pp. 109–117.

Majumdar, Anirudha and Russ Tedrake (2017). "Funnel libraries for real-time robust feedback motion planning." In: *The International Journal of Robotics Research* 36.8, pp. 947–982.

Meagher, Donald (1980). *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-D objects by computer*. Tech. rep. Rensselaer Polytechnic Institute.

Meier, Lorenz, Petri Tanskanen, Lionel Heng, GimHee Lee, Friedrich Fraundorfer, and Marc Pollefeys (2012). "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision." In: *Autonomous Robots* 33.1-2, pp. 21–39.

Miller, Andrew T., Steffen Knoop, Henrik I. Christensen, and Peter K. Allen (2003). "Automatic grasp planning using shape primitives." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Mori, Tomoyuki and Sebastian Scherer (2013). "First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Mueller, Mark W., Markus Hehn, and Raffaello D'Andrea (2015). "A computationally efficient motion primitive for quadrocopter trajectory generation." In: *IEEE Transactions on Robotics* 31.6, pp. 1294–1310.

Nieuwenhuisen, Matthias and Sven Behnke (2014a). "Hierarchical planning with 3D local multiresolution obstacle avoidance for micro aerial vehicles." In: *Proceedings of the Joint International Symposium on Robotics (ISR) and the German Conference on Robotics (ROBOTIK)*.

— (2014b). "Layered mission and path planning for MAV navigation with partial environment knowledge." In: *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*.

— (2015). "3D planning and trajectory optimization for real-time generation of smooth MAV trajectories." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*.

— (2016). "Local multiresolution trajectory optimization for micro aerial vehicles employing continuous curvature transitions." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

— (2019). "Search-based 3D planning and trajectory optimization for safe micro aerial vehicle flight under sensor visibility constraints." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Nieuwenhuisen, Matthias, David Droeschel, Marius Beul, and Sven Behnke (2016). "Autonomous navigation for micro aerial vehicles in complex GNSS-denied environments." In: *Journal of Intelligent & Robotic Systems* 84.1, pp. 199–216.

Nieuwenhuisen, Matthias, David Droeschel, Dirk Holz, Jörg Stückler, Alexander Berner, Jun Li, Reinhard Klein, and Sven Behnke (2013a). "Mobile bin picking with an anthropomorphic service robot." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Nieuwenhuisen, Matthias, David Droeschel, Johannes Schneider, Dirk Holz, Thomas Läbe, and Sven Behnke (2013b). "Multimodal obstacle detection and collision avoidance for micro aerial vehicles." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*.

Nieuwenhuisen, Matthias, Jan Quenzel, Marius Beul, David Droeschel, Sebastian Houben, and Sven Behnke (2017). "ChimneySpector: Autonomous MAV-based indoor chimney inspection employing 3D laser localization and textured surface reconstruction." In: *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*.

Nieuwenhuisen, Matthias, Mark Schadler, and Sven Behnke (2013c). "Predictive potential field-based collision avoidance for multicopters." In: *International Arch. Photogramm. Remote Sens. Spatial Inf. Sci. (ISPRS)*. Vol. XL-1/W2, pp. 293–298.

Nieuwenhuisen, Matthias, Ricarda Steffens, and Sven Behnke (2012a). "Local multiresolution path planning in soccer games based on projected intentions." In: *RoboCup 2011: Robot Soccer World Cup XV*. Ed. by Thomas Röfer, N. Michael Mayer, Jesus Savage, and Uluç Saranlı. Vol. 7416. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 495–506.

Nieuwenhuisen, Matthias, Jörg Stückler, Alexander Berner, Reinhard Klein, and Sven Behnke (2012b). "Shape-primitive based object recognition and grasping." In: *Proceedings of the German Conference on Robotics (ROBOTIK)*.

Nikolic, Janosch, Jörn Rehder, Michael Burri, Pascal Gohl, Stefan Leutenegger, Paul T. Furgale, and Roland Siegwart (2014). "A synchronized visual-inertial sensor system with FPGA preprocessing for accurate real-time SLAM." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

NMA (2016). *Robot tours in the national museum Australia*. URL: www.nma.gov.au/engage-learn/robot-tours.

Nuske, Stephen, Sanjiban Choudhury, Sezal Jain, Andrew Chambers, Luke Yoder, Sebastian Scherer, Lyle Chamberlain, Hugh Cover, and Sanjiv Singh (2015). "Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers." In: *Journal of Field Robotics* 32.8, pp. 1141–1162.

Ok, Kyel, Sameer Ansari, Billy Gallagher, William Sica, Frank Dellaert, and Mike Stilman (2013). "Path planning with uncertainty: Voronoi uncertainty fields." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Oleynikova, Helen, Michael Burri, Zachary Taylor, Juan Nieto, Roland Siegwart, and Enric Galceran (2016). "Continuous-time trajectory optimization for online UAV replanning." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Olson, Edwin (2011). "AprilTag: A robust and flexible visual fiducial system." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Papazov, Chavdar, Sami Haddadin, Sven Parusel, Kai Krieger, and Darius Burschka (2012). "Rigid 3D geometry matching for grasping of known objects in cluttered scenes." In: *The International Journal of Robotics Research* 31.4, pp. 538–553.

Park, Chonhyon, Jia Pan, and Dinesh Manocha (2012). "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments." In: *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Pavlichenko, Dmytro and Sven Behnke (2017). "Efficient stochastic multicriteria arm trajectory optimization." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Pieper, Donald Lee (1968). "The kinematics of manipulators under computer control." PhD. Stanford University.

Puls, Tim, Markus Kemper, Reimund Küke, and Andreas Hein (2009). "GPS-based position control and waypoint navigation system for quadrocopters." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Quenzel, Jan, Matthias Nieuwenhuisen, David Droeschel, Marius Beul, Sebastian Houben, and Sven Behnke (2018). "Autonomous MAV-based indoor chimney inspection with 3D laser localization and textured surface reconstruction." In: *Journal of Intelligent & Robotic Systems*. Available online.

Quigley, Morgan, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng (2009). "ROS: An open-source robot operating system." In: *Proceedings of the ICRA Workshop on Open Source Software*.

Rahardja, Krisnawan and Akio Kosaka (1996). "Vision-based bin-picking: Recognition and localization of multiple complex objects using simple visual cues." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Richter, Charles, Adam Bry, and Nicholas Roy (2013). "Polynomial trajectory planning for quadrotor flight." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

RoboCup Technical Commitee (2010). *RoboCup standard platform league rule book*.

Röfer, Thomas, Tim Laue, Judith Müller, Armin Burchardt, Erik Damrose, Alexander Fabisch, Fynn Feldpausch, Katharina Gillmann, Colin Graf, Thijs Jeffry de Haas, Alexander Härtl, Daniel Honsel, Philipp Kastner, Tobias Kastner, Benjamin Markowsky, Michael Mester, Jonas Peter, Ole Jan Lars Riemann, Martin Ring, Wiebke Sauerland, André Schreck, Ingo Sieverdingbeck, Felix Wenk, and Jan-Hendrik Worch (2010). *B-Human team report and code release 2010*. www.b-human.de/downloads/bhuman10_coderelease.pdf.

Ross, Philip E. (2015). *Watch this Tesla drive itself*. URL: spectrum.ieee.org/cars-that-think/transportation/self-driving/watch-this-video-of-teslas-autopilot-in-action.

Ross, Stéphane, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew Bagnell, and Martial Hebert (2013). "Learning monocular reactive UAV control in cluttered natural environments." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Roy, Nicholas, Wolfram Burgard, Dieter Fox, and Sebastian Thrun (1999). "Coastal navigation – Mobile robot navigation with uncertainty in dynamic environments." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Schmid, Korbinian, Philipp Lutz, Teodor Tomić, Elmar Mair, and Heiko Hirschmüller (2014). "Autonomous vision-based micro air vehicle for indoor and outdoor navigation." In: *Journal of Field Robotics* 31.4, pp. 537–570.

Schwarz, Max, Anton Milan, Christian Lenz, Aura Munoz, Arul Selvam Periyasamy, Michael Schreiber, Sebastian Schüller, and Sven Behnke (2017). "NimbRo Picking: Versatile part handling for warehouse automation." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Sermanet, Pierre, Raia Hadsell, Marco Scoffier, Urs Muller, and Yann LeCun (2008). "Mapping and planning under uncertainty in mobile robots with long-range perception." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Shen, Shaojie, Nathan Michael, and Vijay Kumar (2011). "Autonomous multi-floor indoor navigation with a computationally constrained MAV." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Silva, Grimaldo and Thierry Fraichard (2017). "Human robot motion: A shared effort approach." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*.

Stefas, Nikolaos, Patrick A. Plonski, and Volkan Isler (2018). "Approximation algorithms for tours of orientation-varying view cones." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Steffens, Ricarda (2010). "Multiresolutions Pfadplanung in dynamischer Umgebung für die Standard Platform League." Bachelor thesis. Rheinische Friedrich-Wilhelms-Universität Bonn.

Steffens, Ricarda, Matthias Nieuwenhuisen, and Sven Behnke (2014). "Continuous motion planning for service robots with multiresolution in time." In: *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*.

Stentz, Anthony (1994). "Optimal and efficient path planning for partially-known environments." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Stückler, Jörg, Dirk Holz, and Sven Behnke (2012). "RoboCup@Home: Demonstrating everyday manipulation skills in RoboCup@Home." In: *IEEE Robotics & Automation Magazine* 19.2, pp. 34–42.

Stückler, Jörg, Ricarda Steffens, Dirk Holz, and Sven Behnke (2011). "Real-time 3D perception and efficient grasp planning for everyday manipulation tasks." In: *Proceedings of the European Conference on Mobile Robots (ECMR)*.

Şucan, Ioan A. and Lydia E. Kavraki (2008). "Kinodynamic motion planning by interior-exterior cell exploration." In: *Algorithmic Foundation of Robotics VIII, Int. Workshop on the Algorithmic Foundations of Robotics (WAFR)*. Vol. 57. Springer Tracts in Advanced Robotics. Springer, pp. 449–464.

Şucan, Ioan A., Mark Moll, and Lydia E. Kavraki (2012). "The Open Motion Planning Library." In: *IEEE Robotics & Automation Magazine* 19.4, pp. 72–82.

Szulewski, Piotr (2017). "Towards self-organizing production environments." In: *Mechanik* 7.

Thrun, Sebastian, Wolfram Burgard, and Dieter Fox (2006). *Probabilistic Robotics*. MIT Press.

Ulrich, Lawrence (2016). "Top ten tech cars 2016." In: *Spectrum, IEEE* (Apr. 2016), pp. 34–45.

Vahrenkamp, Nikolaus, Tamim Asfour, and Rüdiger Dillmann (2012). "Simultaneous grasp and motion planning: Humanoid robot ARMAR-III." In: *IEEE Robotics & Automation Magazine* 19.2, pp. 43–57.

Vanneste, Simon, Ben Bellekens, and Maarten Weyn (2014). "3DVFH+: Real-time three-dimensional obstacle avoidance using an OctoMap." In: *Proceedings of the Workshop on Model-Driven Robot Software Engineering (MORSE)*.

Whalley, Matthew S., Marc D. Takahashi, Jay W. Fletcher, Ernesto Moralez, LTC Carl R. Ott, LTC Michael G. Olmstead, James C. Savage, Chad L. Goerzen, Gregory J. Schulein, Hoyt N. Burns, and Bill Conrad (2014). "Autonomous Black Hawk in flight: Obstacle field navigation and landing-site selection on the RASCAL JUH-60A." In: *Journal of Field Robotics* 31.4, pp. 591–616. ISSN: 1556-4967.

Xue, Zhixing, Alexander Kasper, J. Marius Zoellner, and Rüdiger Dillmann (2009). "An automatic grasp planning system for service robots." In: *Proceedings of the International Conference on Advanced Robotics (ICAR)*.

Zeng, Andy, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, Nima Fazeli, Ferran Alet, Nikhil Chavan Dafle, Rachel Holladay, Isabella Morena, Prem Qu Nair, Druck Green, Ian Taylor, Weber Liu, Thomas Funkhouser, and Alberto Rodriguez (2018). "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Zhang, Ji, Rushat Gupta Chadha, Vivek Velivela, and Sanjiv Singh (2018). "P-CAP: Pre-computed alternative paths to enable aggressive aerial maneuvers in cluttered environments." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Zucker, Matthew, Nathan Ratliff, Anca Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher Dellin, J. Andrew (Drew) Bagnell, and Siddhartha Srinivasa (2013). "CHOMP: Covariant Hamiltonian optimization for motion planning." In: *The International Journal of Robotics Research* 32.9-10, pp. 1164–1193.