Egospace Motion Planning Representations for Micro Air Vehicles

Thesis by
Anthony T. Fragoso

In Partial Fulfillment of the Requirements for the
degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2018
Defended 11 October 2017

Anthony T. Fragoso
ORCID: 0000-0002-5805-9668

ACKNOWLEDGEMENTS

Robots don't build themselves (at least not yet), and I owe my gratitude to the team of researchers that worked on the system discussed in these pages.

My work on micro air vehicles began four years ago with an email to Prof. Richard Murray and Dr. Larry Matthies, who replied and agreed to jointly serve as my advisers. Since then, the project took on a life of its own — in no small part a result of a connection between JPL and campus that emerged from this work. I thank both Richard and Larry for cultivating an atmosphere of cooperation between lab and campus, as well as for their encouragement and dedication to helping me become a better researcher.

The project started as a rather applied and technical endeavor, and I thank my commitee members for encouraging me to explore its much deeper theoretical and academic components. Beverley McKeon, Soon-Jo Chung, and Mory Gharib contributed their expertise in this regard, and their input was very much appreciated.

I'd like to thank the scientists and engineers of the Micro Air Vehicle Laboratory at JPL for helping turn my ideas into an aircraft that can actually fly. I thank Christian Brommer for lending his impressive technical acumen to this work and for his willingness to always help out on a repair. Jeff Delaune and Stephen Weiss were dedicated to making sure that my system had reliable state estimation — they spent countless hours debugging and tuning, often in the field, and had an unmatched desire to see their system succeed and excel. Roland Brockers, Brandon Rothrock, and Cevahir Cigla were coauthors for much of the work I describe here, and I thank them for their hard work on the perception system and for fascinating research discussions. We also had a group of hardworking student interns around, and they always made manufacturing, repairs, and testing much easier — I'd like to thank Duke Le for manufacturing and repair support, Norbert Nowak for help with electronics, Christian Stewart for OS development, and Connor Lee for his HDR system and his service as a coauthor. Most of the time that went into this research was probably spent at our arroyo test site, where a number of JPL engineers endured long hours in the field to run experiments. Although they are too numerous to list here, I thank them all for putting aside their own work to help me test and produce a better system.

ABSTRACT

Navigation of micro air vehicles (MAVs) in unknown environments is a complex sensing and trajectory generation task, particularly at high velocities. In this work, we introduce an efficient sense-and-avoid pipeline that compactly represents range measurements from multiple sensors, trajectory generation, and motion planning in a 2.5–dimensional projective data structure called an *egospace representation*. Egospace coordinates generalize depth image obstacle representations and are a particularly convenient choice for configuration flat mobile robots, which are differentially flat in their configuration variables and include a number of commonly used MAV plant models. After characterizing egospace obstacle avoidance for robots with trivial dynamics and establishing limits on applicability and performance, we generalize to motion planning over full configuration flat dynamics using motion primitives expressed directly in egospace coordinates. In comparison to approaches based on world coordinates, egospace uses the natural sensor geometry to combine the benefits of a multi-resolution and multi-sensor representation architecture into a single simple and efficient layer.

We also present an experimental implementation, based on perception with stereo vision and an egocylinder obstacle representation, that demonstrates the specialization of our theoretical results to particular mission scenarios. The natural pixel parameterization of the egocylinder is used to quickly identify dynamically feasible maneuvers onto radial paths, expressed directly in egocylinder coordinates, that enable finely detailed planning at extreme ranges within milliseconds. We have implemented our obstacle avoidance pipeline with an Asctec Pelican quadcopter, and demonstrate the efficiency of our approach experimentally with a set of challenging field scenarios. The scalability potential of our system is discussed in terms of sensor horizon, actuation, and computational limitations and the speed limits that each imposes, and its generality to more challenging environments with multiple moving obstacles is developed as an immediate extension to the static framework.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] A. Fragoso, L. Matthies, and R. Murray, "A fast motion planning representation for configuration flat robots with applications to micro air vehicles," in *American Control Conf.*, 2017. DOI: 10.23919/ACC.2017.7963600,
A. Fragoso developed the theoretical content and generated all simulations. Edited portions of this work appear in Chapters 2 and 3.

[2] A. Fragoso, C. Cigla, R. Brockers, and L. Matthies, "Dynamically feasible motion planning for micro air vehicles using an egocylinder," in *Conf. on Field and Service Robotics*, 2017. [Online]. Available: http://www.fsr.ethz.ch/papers/FSR_2017_paper_58.pdf,
A. Fragoso designed and conducted experiments and developed the motion planning content. Edited portions of this work appear in Chapter 4.

[3] R. Brockers, A. Fragoso, B. Rothrock, C. Lee, and L. Matthies, "Vision-based obstacle avoidance for micro air vehicles using an egocylindrical depth map," in *Int. Symp. on Experimental Robotics*, 2016. DOI: 10.1007/978-3-319-50115-4_44,
A. Fragoso designed and conducted experiments and the system performance evaluation. A. Fragoso also developed the motion planning content, and developed adaptive HDR in collaboration with C. Lee. Edited portions of this work appear in Chapters 2 and 4.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Due to recent advances in microcontroller technology, micro air vehicles (MAVs) have emerged as a highly versatile platform for commercial, scientific, and recreational activities, including remote observation, exploration, and reconnaissance in areas inaccessible to humans or ground vehicles. Stabilizing digital controllers are embedded aboard even the smallest of MAVs, which are often capable of vertical take-off and landing (VTOL) and are generally among the easiest of aircraft to pilot (Figure 1.1).

Figure 1.1: Micro air vehicles, colloquially known as "drones", are available in a number of configurations. Quadcopters, such as the Asctec Hummingbird research aircraft pictured here, are perhaps the most familiar and widely used member of the the multirotor vehicle class. Multirotors are inherently unstable in flight without high-frequency onboard estimation and control, but the delegation of these tasks to digital controllers reduces the pilot workload to a motion planning problem in position and yaw angle. The theoretical source of this reduction in workload arises from a deep result in nonlinear control, and is also essential to MAV autonomy (see Chapter 3).

In spite of its simplicity, the MAV-plus-pilot operation model has severe limitations on mission complexity and scope. The poor line-of-sight visibility of small aircraft limits manual operation of MAVs to simple missions, open environments, and short ranges, and technical considerations frequently preclude the use of a pilot altogether. The most high-profile research and development efforts, such as massive-scale delivery of small parcels [1] and planetary exploration [2], [3], are extremely

inefficient or impossible to operate manually. Navigation and motion planning for all but the most trivial missions must allow for a high degree of autonomy that is in turn dependent on lightweight and low-power sensing and computation. These requirements conspire with high speeds, nonlinear dynamics, and the complexity of unknown and unprepared operating environments to produce a challenging setting for the engineering of MAV systems. Obstacle detection and representation, trajectory generation, and motion planning must be performed in a highly compact fashion and designed methodically using the tools of control theory and computer vision.

Research into obstacle avoidance for autonomous aircraft has traditionally been divided into two efforts: the first focuses on obstacle detection and attempts to accurately and compactly represent a scene for presentation to a black-box motion planner, while the second neglects onboard detection and state estimation in order to provide accurate control of aggressive and complex maneuvers. When either aspect of the obstacle avoidance problem is compromised, the ability of a system to operate in general environments becomes severely limited. Off-board state-estimation and obstacle detection allows for efficient navigation through cluttered, but known environments with full dynamics, but does not allow aircraft to stray from a carefully prepared and relatively small area of operation. Only with the separate advances made in each of these efforts have the first self-contained, independent, and practical field obstacle avoidance systems emerged (such as [4], [5]).

1.1 Perception

Sensing regimes that have received use in other areas of robotics are often impractical aboard small aircraft due to weight, detection horizon, or computational expense, and must be selected using a system-level payload-design perspective.

Obstacle sensors interact with their environment either actively, in which radiation is broadcast to obstacles and the return signal is measured to extract depth data, or passively, in which the interaction of exogenous radiation with obstacles — almost always detected visually using cameras — is measured instead. Active sensors, such as portable radar, LIDAR, or structured light, concentrate complexity in hardware and within the sensor itself and have a smaller computational footprint on the general-purpose flight stack. Depth data arrives nearly instantaneously in a single measurement and processing stage.

This emphasis on hardware, however, increases weight and bulk on small aircraft

that already have extreme payload limitations, and the requirement for internally produced radiation emission limits performance at long ranges or in direct sunlight. Laser scanners also have a limited field-of-view, which makes complete coverage of the environment impossible for small aircraft that in practice can only support a single unit. Single unit laser packages often require servo control mechanisms to provide satisfactory coverage for aircraft flight, which only adds additional weight and complexity (Figure 1.2). Modeling and error analysis of active sensors is reviewed in [6], with a detailed specification to laser range-finding.

Figure 1.2: A time-of-flight laser scanner calculates distance data by measuring the time it takes an infrared laser signal to strike an obstacle and return to the scanner in each direction. The model shown here (Hokuyo UTM-30LX, white arrow), is light enough (370 g) for MAV applications on mid-to-large sized vehicles (Asctec Pelican shown), with a detection range of roughly 30 m and an unobstructed field-of-view of about 270° about its axis on a single plane. An additional "nodding" servo mechanism, not provided, is required for 3D coverage. Image modified from [7].

In contrast to the active sensing philosophy, passive visual approaches receive directional information and typically rely mathematically on the comparison of at least two images to the resolve depth ambiguity of a single image. Visual detection has the advantage of lightweight, low-power cameras as sensors and can readily resolve long-range obstacles, but places the burden of the detection problem on computation.

Monocular obstacle detection typically uses some form of optical flow, in which two spatially and temporally separated images from a single camera are compared in order to resolve the range ambiguity associated with directional data (see Chapter 8 of [8] for an extensive review). Accordingly, obstacles can be localized only up to a scale factor that must be determined using inertial cues or an external state estimate. This process has the advantage of extremely simple hardware, requir-

ing at most a single camera and an inertial measurement unit (IMU), but cannot detect obstacles without sufficient vehicle motion. Furthermore, once moving, the camera cannot acquire new depth data in the direction of vehicle velocity because directional information does not change during direct approach of an obstacle — a serious limitation for aircraft that must look where they are going. Learning-based approaches, such as [9], offer the possibility of avoiding this issue by using implicit visual cues to extract a depth image or a direct reactive maneuver without a need for a second spatially separated image, but suffer from the difficulties of training and transfer from a data set and remain an active area of research. Geometric monocular vision also assumes an entirely static scene, and will misrepresent the distance to moving objects without extensive filtering and inertial reasoning.

Stereo vision also compares images to extract depth data, but avoids the velocity-based observability problems of monocular vision by using two or more fixed cameras to simultaneously produce spatially separated images (see [10] for a review of the geometry of stereo imaging and its generalizations). This comparison process, however, limits the minimum and maximum detection ranges to those which can be achieved with the specified camera resolution and separation baseline. For a given resolution, cameras that are too close will be unable to distinguish independent directions to a distant obstacle and fail to calculate a reliable range, and cameras that are too far apart will be unable to capture close obstacles simultaneously in two images. The accuracy of stereo also decays with range, which for the purposes of accurate and efficient motion planning introduces a dependence on the geometry of data representation that can enhance or degrade data quality independently of the accuracy of the data itself. This connection between data quality and representation is a major focus of this work and is explored in later sections.

Although there is as yet no "gold standard" onboard sensing approach that is advantageous in all aspects, most practical systems employ some form of either stereo vision or LIDAR. The recent commercial development of single-purpose stereo integrated circuits and smaller laser scanners for the automotive market offers the expectation that, respectively, the computational and size and power limitations of these sensors will diminish in severity.

1.2 Representation

Regardless of the sensing platform, the motion planning software that eventually must follow a perception suite operates on an internal representation that stores

and indexes the locations of obstacles in an image. To date, MAV motion planners (including [4], [11] and the "state-lattice" technique of [12]) have almost always relied on the creation of a 3D “voxel” map, in which a 3D world is discretized into a grid and occupancy (the presence or absence of an obstacle) is noted at each point. This representation has been successfully used on a quadcopter with stereo vision during a mapping operation, but at low speeds with negligible dynamics [13]. Uniformly discretized voxel-based approaches are simple to index and store, but require a transformation step in which range data is converted into Cartesian coordinates and a 3D grid filled out accordingly (Figure 1.3).

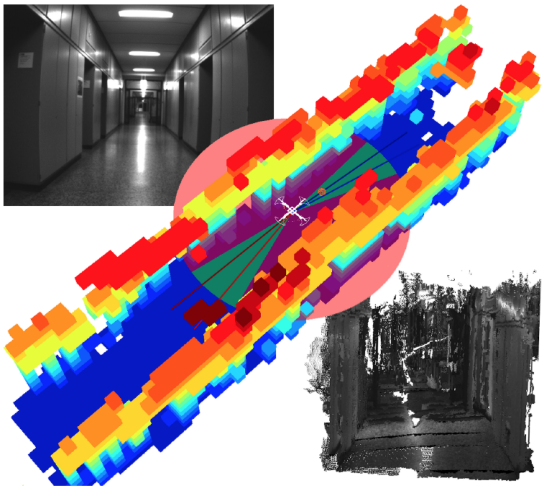

Figure 1.3: A corridor scene (top left) is captured using a stereo camera pair onboard a MAV, and the resulting depth data is converted to a point cloud that is used to populate a probalistic occupancy grid (center). The probabilistic likelihood of an obstacle being located in each cell of the uniform grid, with resolution 25 cm, is noted and updated as points arrive and are appropriately placed in the structure. A textured 3D map recovered from the occupancy grid is at bottom right. Image from ([13]).

Attempts to more compactly represent obstacles have typically relied on a conversion to cylindrical coordinates ([14], [15]), which have an intrinsic advantage of increased resolution at short ranges where it is most needed, and decreased resolution at long ranges where it is less needed. The idea of scaling stereo vision to high speeds by modulating resolution is taken to its limit by [16], who demonstrate a “push broom” technique that only detects obstacles at a single disparity and uses onboard state estimation to propagate their location with no further attempt at detection. This technique is able to achieve extremely high frame rates (120 frames per

second), but is completely blind to obstacles that are first encountered at a distance less than that of the detection horizon and cannot handle obstacle motion.

The JPL closed-loop rapidly exploring random tree system (JPL CL-RRT; [17]) generalizes polar representational structure as part of one of the first obstacle avoidance pipelines to autonomously navigate an aircraft with full nonlinear dynamics through cluttered, unknown environments. It is most notable for introducing a compact obstacle representation based on the depth-image data structure acquired natively from a set of stereo cameras (Figure 1.4), which is implemented using a disparity (inverse depth) parameterization of visual input as a means to extend the usefulness of stereo vision to long ranges. Although not exhibited experimentally on the JPL CL-RRT system, [17] suggest that their depth image structure can be used to fuse stereo data with side-looking monocular optical flow for enhanced field of regard in cluttered environments.

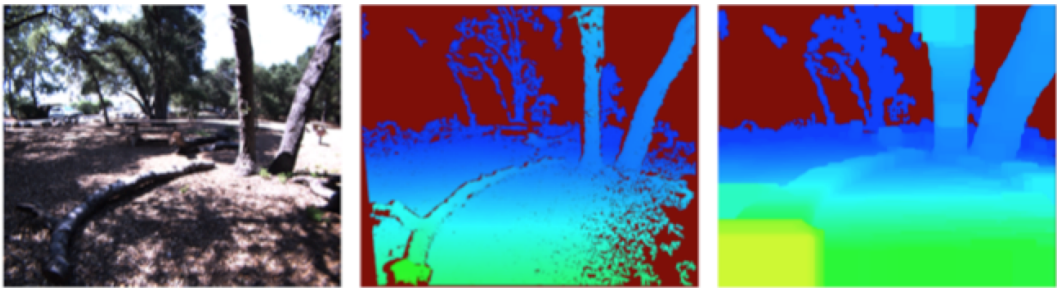

Figure 1.4: The JPL-CRRT system relies on a "C-space expansion" technique that dilates obstacles within an image for efficient collision-checking of trajectories against disparity image data structures as acquired natively by a stereo camera pair. Raw images (left) are captured by the stereo pair, which are then run through a stereo matching algorithm to produce a raw depth map (middle). Obstacles are then expanded by a characteristic vehicle radius (right), which allows the vehicle to be abstracted to a point mass for the purposes of collision-checking by comparison—if a projected point has a disparity equal to that of the expanded map at its pixel coordinates, it is invalid. This approach contrasts with classical occupancy grids in that the geometry of the image remains more-or-less unmodified and retains a radial alignment in order to efficiently vary resolution with distance. This thesis extends the observation that valid vehicle states can be easily identified in an image to a full obstacle avoidance pipeline in which feasible motion plans arise naturally from an image instead. Figure from ([17]).

JPL CL-RRT also contributes a disparity space “C–space expansion” technique, in which difficulty of collision-checking the trajectory of a finite-sized vehicle is bypassed by artificially increasing the size of obstacles in the same disparity co-

ordinates used to detect and most compactly represent them. This step stands in contrast to the usual practice of doing so in world coordinates, in which depth data would be converted into a voxel grid and then expanded in three dimensions. Although both approaches allow the vehicle to be treated as a point mass for motion planning, JPL CL-RRT and also bypasses the need for conversion to a voxel map for collision checking by performing all expansion calculations and intermediate steps within the disparity representation.

Motion plans are collision-checked in JPL CL-RRT by projection into disparity space and comparison to the disparity of obstacles in the scene. This technique is inspired by by the "Z-buffer" procedure of computer graphics, and differs primarily in its use of disparity coordinates rather than depth — if the disparity of the projected path coincides with that of an expanded obstacle in the same pixel, a collision results and the path is invalid. In doing so, the disparity space C-space expansion appears to provide computational savings in its compactness as an obstacle representation for collision-checking. In spite of the representational advantages demonstrated by JPL CL-RRT, however, compactness and efficiency claims have not been rigorously verified and the motion planning portion of the pipeline suffers from high planning latency and slow speeds due to a reliance on onboard numerical integration of dynamics in world coordinates. These issues, as well as a desire to more generally analyze motion planning using the novel representational structure of JPL CL-RRT, provide the primary theoretical and experimental impetus for this thesis.

1.3 Trajectory Generation, Selection and Control

Once an obstacle representation is at hand, dynamical trajectory generation for MAVs typically assumes vehicle plant models with the differential flatness property, which allows all vehicle states and controls to be expressed algebraically in terms of a smaller set of "flat outputs" [18]. For a differentially flat system, any sufficiently smooth trajectory in the flat output space can be followed up to state and control constraints, and the difficulty of online nonlinear trajectory generation can be largely bypassed by transformation. The class of differentially flat vehicles includes quadcopters as well as common car and airplane abstractions, and is useful for MAV motion planning because trajectory generation for differentially flat vehicles does not require direct integration of the state equations — indeed, feasible trajectories in the flat output space are relatively easy to find using simple linear algebraic and simple optimization methods.

For quadcopters, differential flatness has been used to generate real-time "minimum snap" trajectories that penalize the fourth derivative of position to intentionally limit control input aggression [19]. When properly posed as an optimzation problem, trajectories that result from minimizing a net derivative of position between two states have a polynomial form of fixed degree, but with undetermined coefficients and an undetermined final time — a property used heavily later on in this work to simplify motion planning. As a result, selection of a suitable minimum snap trajectory becomes a geometric, rather than control-theoretic, problem and is essentially a search over these undetermined parameters for a solution that will not result in a collision. A minimum snap framework formed the basis of trajectory generation in the quadcopter motion planning algorithm of [20], in which high-speed indoor flight was calculated offline, within a volumetric world model, by seeding a polynomial motion planner with the output of a preliminary, dynamics-free RRT* search. A similar minimum jerk (third derivative of position) scheme was used aboard a quadcopter in ([4], [21]) following a graph search within a fixed-resolution voxel grid representation.

The related notion of motion primitives — short trajectory segments integrated symbolically in advance and pieced together online — also attempts to sidestep online integration of the state equations by abstracting available control trajectories into a finite set of admissible behaviors. In general, these approaches also derive their utility from a conversion of control theoretic problems into geometric problems, which further allows vehicle dynamics to embedded within a "maneuver automaton" decision-making structure in which the state equations are inaccessible to the motion planner itself [22]. In addition to their classical use for ground vehicles [23] and civil engineering [24], motion primitives have extensive precedent in the standardized maneuvers that are at the core of manned aviation. Standard flatness-based approaches, of course, also prescribe a functional form to trajectories and are a subset of the motion primitive concept in some sense.

Regardless of its origin, the actual execution of a trajectory is typically delegated to a low-level controller that is split into a fast inner loop that regulates vehicle attitude and a slower outer loop that regulates vehicle position and speed (for quadcopters, see [25]). This control architecture is well known for autonomous aircraft, and the ability to easily extract bare-metal motor inputs from a prescribed trajectory arises from the differential flatness property as well as the partition of the vehicle dynamics by speed.

1.4 Contribution

The contribution of this thesis is to combine the compactness of image-based obstacle representation and perception with the efficiency of flatness and motion-primitive-based motion planning, and analyze the resulting performance benefits theoretically and experimentally. In Chapter 2 we discuss visual obstacle detection and formalize the notion of an image-based representation to "egospace", which replaces the depth image data structure with a general sensing geometry that can accomodate a range sensor suite of arbitrary type and configuration. This definition is followed with a full theoretical characterization of egospace, including its ability to faithfully represent measured obstacle environments and accomodate point-mass and spherical vehicle abstractions. The compactness advantages of egospace are also demonstrated against uniform Cartesian grids.

In Chapter 3 we analyze motion planning within the egospace structure, relying primarily on an extension of "configuration flat" vehicle dynamics. We characterize the general configuration flat motion planning problem in egospace coordinates, after which we introduce a set of useful approximations and abstractions, with variable dynamic fidelity to actual MAVs, to assess the limits of their performance and applicability. The advantages of trajectory searches in egospace is compared against occupancy grids for commonly-encountered obstacle avoidance scenarios.

In Chapter 4 we consider an experimental quadcopter platform, towards which this thesis contributes motion planning and obstacle avoidance capability in egospace. After specializing the theoretical development of the preceding chapters, we provide experimental verification of the egospace motion planning concept and its performance benefits aboard a real aircraft. A scalability discussion considers the effects of horizon limitations at each step of the obstacle avoidance pipeline, and the applicability of egospace is extended to environments with large numbers of moving obstacles.

Chapter 2

PERCEPTION AND REPRESENTATION

Portions of this chapter have been modified from [26] and [27].

In this chapter, we discuss the obstacle perception and representation process — that is, "seeing" and "believing" in a systematic and algorithmic way. Although perception strategies are not a research focus of this thesis, we first review the various sensing options available for MAVs as the first step of an obstacle avoidance pipeline. We then abstract away this component to a black box and introduce the representation problem, which determines how the vehicle makes sense of the range data and uses it to calculate a configuration space, or "C-space", that formally categorizes the set of valid states to which the vehicle can travel. After introducing this formalism, we discuss the apparent advantages of image-based C-space approaches and generalize to a more formal geometric structure that we call "egospace". A complete characterization follows, including a preliminary discussion of a natural motion planning advantage of egospace that is explored more deeply in Chapter 3. We also derive asymptotic compactness relations for egospace and for an equivalent occupancy grid in a long-horizon, high-resolution field scenario.

2.1 Introduction

Perception

As discussed in Chapter 1, a fundamental aspect of mobile robotics in general is the acquisition of information about an environment as it is traversed. If a robot ventures into an unprepared setting, in order to avoid obstacles and reach a goal it must typically have some way to detect their presence before it can calculate a safe path around them.

The most primitive manner of obstacle detection is based on contact sensing, in which robots are allowed to collide with obstacles and a "bump detector" notes their presence or absence. The obvious advantages of detection at a longer range notwithstanding, contact sensing is an important academic regime in ground robotics, primarily as an input to insect-inspired "bug" wall-following algorithms that assume only local information about an environment (as in [28]; see Chapter 3). Contact sensing is taken to the limits of austerity in [29] for aerial robots, in which small

MAVs in protective roll cages were allowed to collide with obstacles on their way towards a destination without any attempt at detection or avoidance in advance — the only evidence of an obstacle existing is the sudden deceleration registered by a separate state estimation module.

Unless an aerial robot is specifically designed for contact with obstacles, however, collision events usually result in mission failure and the potential for vehicle damage or loss. Accordingly, there is a clear advantage to detecting obstacles well before they are approached by the vehicle, with the necessary sensing horizon for safe operation generally increasing with speed. Sensors with a non-zero detection distance, or "range sensors", detect obstacles by measuring their interaction with an energy source that is either onboard and actively emitting ("active sensing") or exogenous and simply measured onboard ("passive sensing"). The source of energy from active range sensors is typically electromagnetic, although active ultrasound has been used on autonomous MAVs for obstacle detection [30] and altimetry [31]. Active electromagnetic sensors used on micro air vehicles include scanning LIDAR [4], [32] and 3D structured light [21], [33]. Scanning LIDAR, as discussed in Chapter 1, measures the return time of a laser light signal (typically infrared) at a specified angular interval to reconstruct the distance to a reflecting obstacle using the constant speed of light. The units that have received use on MAVs typically have a limited vertical field-of-view (entirely planar up to about 15°) and require servo mechanisms (as in [34]) to achieve complete coverage of an environment. Time-of-flight cameras, or "flash LIDAR" have similar principles of operation, but avoid a scan by simultaneously measuring the return time of a laser pulse at each pixel in its sensor array and returning a dense image at each step. 3D structured light techniques project a known test pattern (also infrared) onto obstacles that is distorted by surface topography, imaged, and compared to extract a depth measurement. Although not yet light enough to be employed on an MAV, portable single-beam radar also offers promise as an onboard range sensor [35].

In addition to the size, weight, and hardware complexity issues mentioned earlier, active sensors suffer from severe range limitations in outdoor environments. Radiation emitted by active sensors must first be sourced from a small onboard power supply, and then compete with ambient radiation to generate a return signal strong enough to be distinguishable upon reflection. Accordingly, maximum ranges of most units are relatively short and limit vehicle speeds. The state-of-the-art scanning LIDAR in [4] is limited to a maximum useful range of approximately 40 m,

and the RGB-D structured light sensor in [21] is limited to about 5 meters.

The primary alternative to active detection, especially for the texture-rich outdoor scenes in which active sensors struggle, is passive visual sensing using cameras. In addition to requiring much simpler and lighter hardware (cameras), visual systems can detect obstacles at extreme ranges with an accuracy dependent on resolution rather than power. Vision approaches used by MAVs include structure-from-motion depth maps [36], reactive optical flow [37], or machine learning [9] within a monocular sensing and reactive control regime, or stereo depth matching with a coordinated two-camera vision module (as in [17]). Monocular approaches have the advantage of a simple single-camera sensor layout, but depth perception has observability issues (depth and optical flow are unobservable if the vehicle is stationary or in the direction of its velocity) and learning has transfer issues that limit generalization. With a sufficiently large baseline as afforded by vehicle motion, however, monocular depth perception can be extended to limitless ranges.

Stereo vision avoids the observability issues of monocular vision and offers instantaneous dense depth maps independent of vehicle velocity, but requires an additional synchronized camera. The use of a fixed and relatively small stereo baseline can severely limit the range of a stereo pair without a corresponding increase in image resolution. At least one attempt [27] has been made to use stereo and monocular depth perception simultaneously to mutually enhance the weaknesses of each: a forward-looking stereo pair is used to acquire depth data in the direction of vehicle velocity, while side-looking optical flow extends the total field of regard based on a scale estimate from overlap with the stereo region. This effort, discussed in more detail in Chapter 4 of this thesis, suffered from computational throughput issues that limited its usefulness and led to frequent loss of optical flow tracking and map fusion.

For the rest of this chapter, we shift our focus away from the details of the perception module and assume that, in spite of a motivation largely influenced by visual approaches, 3D range data is arriving in a general, dense, range-plus-direction format unless otherwise specified. This assumption will cover all of the commonly used sensing regimes and allow representation results to remain general and platform-agnostic.

Representation

The primary objective of the representation module is to conveniently organize range data into sets of valid (safe) and invalid (unsafe) situations a robot can find itself in for a given environment. Once a representation is available, a motion planning module is used to determine a trajectory among the set of valid situations towards a goal specified in a higher layer of the vehicle architecture.

Formally, a robot's disposition in its environment is defined as a *configuration* $\mathbf{q}$ in a *configuration space*, or "C-space" C [38]. C-space consists of all configurations that can be reached using actions $\mathbf{u}$, drawn from an admissible *action space* $U(\mathbf{q})$, that induce transitions from configuration to configuration. The *obstacle region* is a set $O \subset C$ into which no part of the vehicle may enter, with the valid states confined to *free space* $C_{\text{free}} = C \setminus O$ (Figure 2.1).

For a finite-sized vehicle, modeled by a set $\mathcal{A}$, configuration space also takes into account the set of valid orientations — that is, any configuration with $\mathcal{A} \cap O \neq \emptyset$ is invalid. Finite-sized vehicles are commonly abstracted to a point by the technique of "C-space expansion", in which the vehicle is first modeled by a sphere (circle in 2D) of radius r and the position of its center used to collision-check against obstacles. Because this radius r is fixed and the spherical vehicle has no meaningful orientation about its center, any point within a distance r of an obstacle is automatically an invalid location for the center of the vehicle. The obstacle region can be grouped along with this set into an expanded obstacle region O', against which the single center point alone can be checked to verify safety.

C-space *representations*, which are the data structures upon which motion planning will ultimately take place, differ from each other in how O and C_{free} are distinguished mathematically, stored, and accessed. If an environment is prepared in advance and consists of simple shapes, such as polyhedra (polygons, in 2D), cylinders (circles, in 2D), or spheres, obstacles can be represented exactly, to infinite precision, using geometric *primitives* that explicitly and constructively define the sets they occupy using inequalities. All points inside of a 2D polygon, for example, can be described exactly using a set of linear inequality constraints that model the intersection of closed half-planes, and the set of points on or inside a sphere of radius R centered at (x_0, y_0, z_0) can be defined by the single inequality constraint $(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 \leq R^2$.

Detailed sensed data of arbitrary obstacles, however, is almost always too complex to model explicitly with primitives and is instead placed into a discretized structure

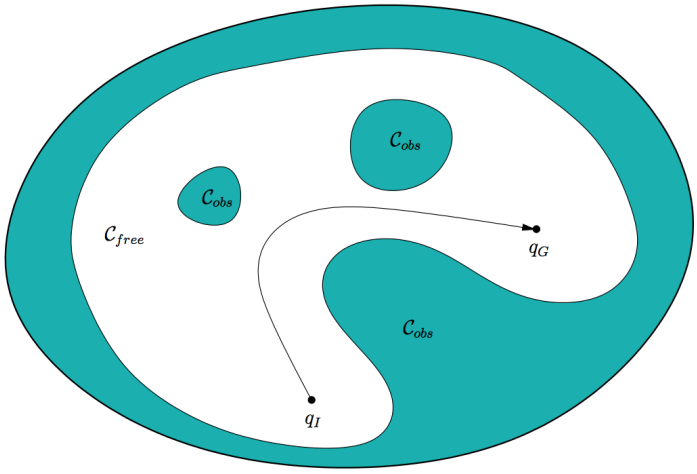

Figure 2.1: A 2D point robot operates in a C-space C consisting of a roughly elliptical region of the plane. An obstacle set O, in cyan, consists of illegal configurations within the space and restricts the position to any points contained in $C_{\text{free}} = C \setminus O$, in white. Here, a motion planner, as discussed in Chapter 3, has also found a valid (obstacle-free) path between the start state $\mathbf{q}_I$ and the goal state $\mathbf{q}_G$. Image from [38].

according to an indexing scheme specified in advance. The classical approach to scene representation for mobile robots is to insert range data into uniform 2D or 3D Cartesian grid maps. Space is discretized into fixed-size cells, known as *voxels* in three dimensions, that are indexed by their position in x, y, and z, and contain estimated probabilities of occupation by an obstacle. Within the MAV literature (such as [13], [4]), the voxel states are typically limited to "occupied" or "unoccupied", with probability 1, and "unknown" otherwise.

Within a full probabilistic framework, a Bayesian update step operates on probabilistic sensing and motion models to gradually build up a high-quality map that updates the occupation confidence in each cell [6]. The map-building step is also occasionally skipped in real time MAV operations, as in the instantaneous voxel maps of [4], in order to keep onboard computation to a manageable level.

In addition to providing a natural setting for mapping and exploration missions in and of themselves, Cartesian occupancy grids are also highly useful for multiple-query planning problems in which known areas must be examined repeatedly for new paths. Mobile robots also have state transition models that are almost always expressed most conveniently in terms of the same (x, y, z) coordinates in which these occupancy grids are indexed. These two properties allow essentially the entire

body of literature on motion planning algorithms on Cartesian grids to be used without modification, which includes a wide variety of local, global, reactive, and deliberative approaches that can be tailored to the specific problem at hand. We review the relationship between map coordinates and vehicle states along with the advantages of various motion planning paradigms in Chapter 3.

In spite of their simplicity and extensive use, however, uniform Cartesian grids scale poorly to longer sensing distances (which are required at high speeds) because their resolution does not adapt with distance. Multiple-resolution Cartesian approaches can mitigate the memory blow-up problem using tree [39] or hash-based structures [40] that partition space hierarchically so that finer detail is used only where it is essential. In maintaining different-sized blocks within the same indexing scheme, however, multi-resolution Cartesian approaches sacrifice the simplicity of array indexing and become significantly more expensive to access and update in real time. As a result, most SLAM and obstacle avoidance applications continue to rely on uniform grids when not performed offline. Although they are unable to perform their octree-based SLAM algorithm aboard their quadcopter due to inadequate computational resources, [41] limit its memory consumption to a bounded amount using a rolling map that dynamically activates and inactivates live "tile" submaps for off-board processing.

For single-query problems in which a mobile robot is simply passing through the environment rather than attempting to explore it, mapping infrastructure on the occupancy grid itself is also largely unnecessary and presents an avoidable computational burden. Accoringly, for obstacle avoidance *per se* less-expensive representation and planning algorithms are possible. Often these representations are polar in nature, matching the polar angular resolution of the depth sensors [14], [15], [42], [43]. In [44], depth data from two onboard stereo pairs was fused in a cylindrical inverse range map centered on a ground vehicle. This work introduced C-space obstacle expansion of an image space depth map, though in a limited fashion based on an assumed ground plane. In [17], the C-space expansion was generalized to be based on the actual depth at each pixel, and included the first combination of an image space depth representation with a dynamics-aware motion planner — feasible trajectories were generated in 3D using forward integration and projected into image space to do collision checking by testing for intersections with the C-space expanded depth map. This approach to obstacle representation and collision checking is fast, compact, and showed good potential in experiments, but had a limited field-

of-view. The disparity space representation has been generalized to an "egocylinder" by [45], which offers a 360° field of regard as well as straightforward fusion of range data from stereo and monocular sensors into a single compact structure. [46] extended the image space representation of [17] to include a C-space expansion based on the expected range uncertainty both in front of and behind a visible surface.

The image-based representations discussed thus far capture only the instantaneous visible obstacle surface, however, and will present spurious or dropped data from a given frame to the motion planner without regard to its actual reliability. Although mitigated somewhat by the C-space expansion, which tends to wash out small regions of missing data, partially occluded surfaces are handled poorly and appear as large gaps in the depth data. [5], [47] addressed these issues by equipping a disparity image and an egocylinder, respectively, with a temporal fusion module based on a Gaussian Mixture Model (GMM) representation of the depth hypothesis at each pixel. Measured range data, sourced from a single stereo pair in both implementations, is compared to previous data propagated forward in time using an external pose estimate. Each hypothesis in the GMM is either reinforced or degraded by the comparison operation, and the most likely depth is inserted into the final data structure and presented to downstream modules. When used with an egocylinder, the GMM propagation technique has a "memory" capability that allows previous obstacle data to persist in areas beyond the field of view of the range sensors. As the vehicle moves around its environment, a full representation that includes areas behind the vehicle begins to emerge in the egocylinder — a feature previously available only to occupancy-based approaches with a map update step.

The temporal fusion step also significantly improves the quality of depth data in the measured region of egospace, and leads to a representation that is stable enough for direct use in motion planning. It is particularly useful for filling in flickering image regions, removing spurious detections, and maintaining a representation of obstacles that have drifted closer than the minimum detection distance.

2.2 Theoretical Development

For the rest of this chapter, we extend the compactness and efficiency advantages of disparity-space obstacle representations, such as the disparity image and egocylinder formulations, to a more general "egospace" data structure that can accommodate an arbitrary range sensor configuration.

Egospace Obstacle Representation

The egospace obstacle representation is a generalization of the 2.5–dimensional "depth map" data structure, which assigns to each pixel on an image plane a distance into the plane called "depth".

Traditional depth maps are based on projection onto planes, however, and necessarily have a limited field of view. In order to accommodate arbitrary sensing geometries with a field of view greater than 180 degrees, we define the egospace as a parameterization of a portion of $\mathbb{R}^3$ by *pixel* coordinates (u, v), which parameterize the set of directions (points in S^2), paired with a generalized depth δ that is a smooth and strictly monotonic function of radius from the origin. Intuitively, egospace simply describes the location of a point by a unique generalized direction and distance, of which ordinary spherical coordinates is an immediately accessible special case.

Definition 1. *The egospace coordinates of a point* $\mathbf{x} = (x, y, z)$ *in a region* $R \subseteq \mathbb{R}^3$ *are curvilinear coordinates* (u, v, δ)*, for which the egospace map* $(x, y, z) = \mathcal{E}(u, v, \delta)$ *has a non-zero and smooth Jacobian determinant and the local unit vector* $\mathbf{e}_\delta$ *is parallel to the local radial unit vector* $\mathbf{e}_r$*. An egospace representation of an obstacle environment is any surface in egospace* $\delta = f(u, v)$*.*

The use of a radial coordinate is responsible for the compactness advantages of egospace (as is shown later in this chapter), but as defined it precludes, in general, smooth and invertible egospace coordinates on all of $\mathbb{R}^3$. This issue arises naturally from the geometry of range sensing and is simple to resolve in practice, but impractical to avoid in full generality. Difficulties always arise at the origin, which is projected ambiguously onto itself, as well as from the well-known geometric fact that it is impossible to parameterize S^2 smoothly by the two coordinates (u, v). Although these technicalities prevent a single valid egospace from covering all of $\mathbb{R}^3$, they can be sidestepped either by restricting the representation to a non-singular region of interest or by patching together multiple egospace representations to cover the entire space — for example, two or more spherical sectors can be used represent all of S^2, or the poles and origin can be assigned any convenient unambiguous coordinates. We simply exclude singularities from the definition because they can be resolved in practice by patching, and construct the egospace representation to be invertible to Cartesian coordinates and therefore able to represent any visible obstacle surface in a valid region of interest.

The actual implementation of an egospace representation, which encodes the visible

Table 2.1: Example Egospace Transformations $\mathcal{E}$

Structure			
Depth Image[a] (u, v, Z)	$x = \frac{uZ}{f}$	$y = Z$	$z = \frac{vZ}{f}$
Egocylinder[b] (θ, v, d)	$x = \frac{\cos\theta}{d}$	$y = \frac{\sin\theta}{d}$	$z = \frac{v}{fd}$
Egosphere[c] (θ, ϕ, ρ)	$x = \rho \sin\phi \cos\theta$	$y = \rho \sin\theta \sin\phi$	$z = \rho \cos\theta$

[a] Standard perspective projection with focal length f and depth Z
[b] For the standard cylindrical radius r, $d = 1/r$ [27]
[c] Ordinary spherical coordinates, azimuthal angle θ and polar angle ϕ

surface perceived by an arbitrary configuration of range sensors, takes a form analogous to the Z-buffer of computer graphics — at each pixel, the generalized depth of the first obstacle encountered in that direction is stored. Egospace coordinates are defined on a continuum for the purposes of motion planning, but in all practical implementations the pixel values are discrete with a floating-point generalized depth. If prior map information is available, as is the case when online temporal filtering of depth data is available (as in [5]), the depth and confidence of multiple occluded surfaces can be stored at each pixel instead. The implementation of an egospace representation also depends on the pixel parameterization and the choice of generalized depth (Table 2.1), with a trivial realization being ordinary spherical coordinates (an "egosphere"). Similarly, the depth image example parameterizes a sector of S^2 using a pinhole projection to pixels coupled with ordinary image depth Z (disparity $d = 1/Z$ can be used instead, as in [17]), while the egocylinder implementation of [27] uses a cylindrical projection of S^2 about the vehicle-carried IMU origin with inverse radius $1/r$ (Figure 2.2).

The primary geometric advantage to egospace obstacle representations is that any parameterization by pixels will have maximal spatial resolution near the origin, where obstacles must be dealt with immediately, that decreases with distance as it becomes less useful. In an ordinary depth image, for example, the Euclidean distance between the points (u_0, v_0, Z_0) and (u_0+1, v_0, Z_0) is smaller than the Euclidean distance between the points $(u_0, v_0, 2Z_0)$ and $(u_0 + 1, v_0, 2Z_0)$. Although it is possible to equip a voxel map with a variable resolution that scales well with distance and avoids excessive detail in empty space [39], [40], such structures are expensive to access and have significant storage overhead because they must fundamentally modify the intrinsic constant resolution (constant Jacobian) of the underlying

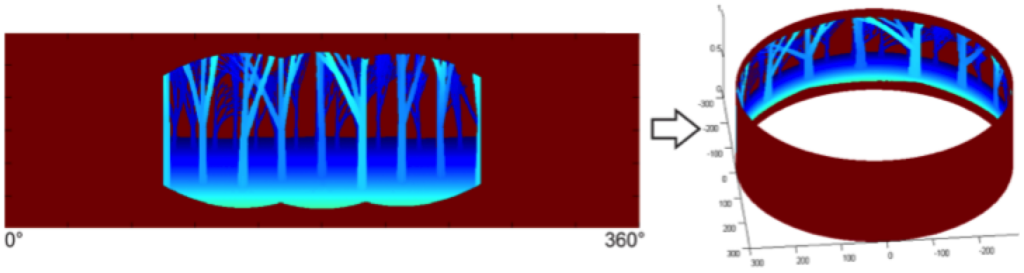

Figure 2.2: A simulated scene (left) is projected into an egospace data structure in order to provide a full 360° representation of an environment. The horizontal coordinate is discretized into uniform angular increments between 0 and 360°. The vertical coordinate, also discretized, is a pinhole image projection with focal length f — as in a standard depth image. In the original egocylinder representation of [27], the generalized depth coordinate is inverse range $1/r$ rather than range r.

Cartesian coordinates. Egospace, on the other hand, acheives a favorable variable resolution in a natural pattern for range sensing with simple array indexing.

Stereo vision has been shown to be particularly well-suited to disparity space and egocylinder representations primarily because a limited baseline and resolution also causes sensor accuracy to decay quickly with range. The choice of generalized depth also plays an important role in partitioning resolution to best suit a sensor — inverse depth, for example, assigns points beyond the sensor horizon a finite value (zero) and extends the useful range of stereo by compressing distant obstacles, which are ranged inaccurately, into a much smaller region of egospace.

The freedom to choose any pixel parameterization also provides the system designer with flexibility to accommodate any range sensor type or configuration, as well as the obstacle layout expected during a mission. The egocylinder, for example, is tailored to cluttered environments with a single dominant ground plane and limited vertical motion. The depth image is tailored to linear scenes with a preferred direction, such as a road or corridor to be followed, or in vehicle configurations where only a single visual range sensing unit is available and it remains coordinated with the velocity vector of the vehicle (as on a fixed-wing aircraft with a range sensor embedded in the nose). A more general environment with significant vertical and horizontal motion, such as an MAV takeoff or landing amongst obstacles, would be best treated using an egosphere to provide coverage of all possible flight paths in a cluttered area. Egospace coordinates can be fixed to external references, such as the local gravity vector, the normal vector of the terrain below the vehicle, or a com-

pass direction, and can also be placed or carried arbitrarily in the world in a fixed or moving frame. In addition to the external reference, multiple range sensors can also be fused into egospace, with overlap, to greatly enhance the robustness of the entire sensing suite. This property is particularly useful for monocular vision, which must be initialized and then tracked continuously to remove a scale ambiguity and produce unambiguous depth data. In the event of a tracking failure or dropped frame, scale can be reinitialized immediately by comparison to intact depth data in regions of overlap with other sensors (as in [27] — see Chapter 4).

Collision-checking in egospace

Egospace also admits an efficient collision-checking approximation for finite-sized vehicles first introduced for disparity image representations in [17]. To simplify collision-checking and motion planning, the vehicle is modeled as a sphere with a characteristic radius and abstracted to a point mass by expanding the apparent size of obstacles *directly* in egospace. In addition to directly accomodating the finite extent of the vehicle, the characteristic radius can be extended to account for sensor uncertainty (as in [46]) and, in experimental settings, introduce a natural "aversion" to obstacles that prevents close approaches.

After first expanding a point in world space to a sphere with radius equal to that of the vehicle, the sphere is replaced by the rectangular region in egospace, at constant generalized depth, that just occludes the sphere from view (Figure 2.3). By insisting on a rectangular expansion (that is, a region in the egospace representation bounded by lines on which either u or v are constant), the entire expansion algorithm is divided into separate horizontal and vertical expansions that are performed consecutively for speed. The performance of the algorithm can be further enhanced with the use of a look-up table for the expansion radii. For a general obstacle configuration in which every data point in a uniform structure must be evaluated and expanded, the cost of visiting each pixel or voxel is compounded by the cost of a depth or occupancy update of all points within the expansion radius. When performed consecutively in each dimension, this cost is dependent on the precision of the representation and scales with the width in pixels M of an egospace representation or the typically much larger size of a uniform voxel map N. Accordingly, the egospace expansion algorithm described is $O(M^3)$ and enjoys a complexity advantage over a uniform voxel map, for which a similar expansion that visits and expands every voxel would be at least $O(N^4)$.

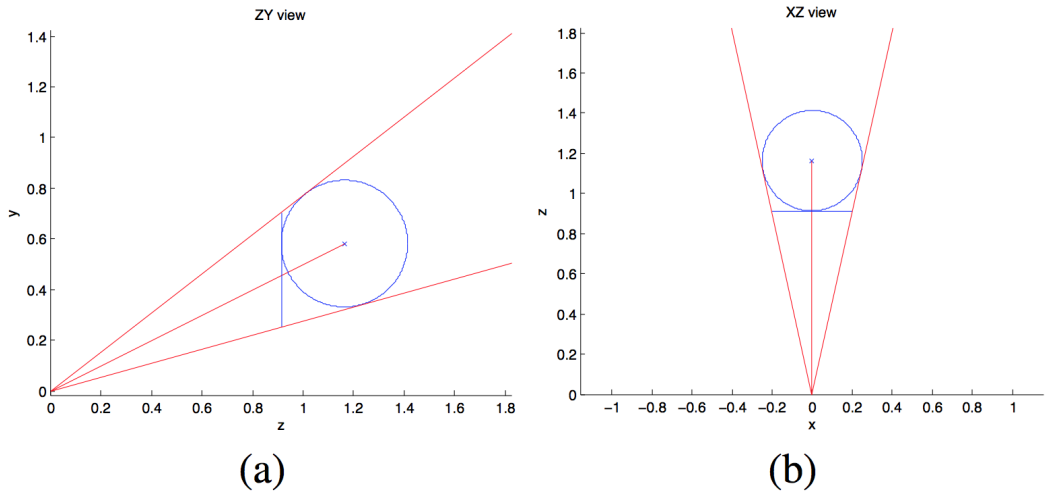

Figure 2.3: The rectangular C-space expansion in depth image coordinates, as described by [17]. After expanding a point in the image by a characteristic vehicle radius (circular region), the constant-depth, rectangular surface that completely occludes the expansion sphere is calculated and loaded into the expanded depth image. The expansion process is divided into separate vertical (a) and horizontal (b) operations for speed. Figure from [17]

Although rectangular expansion is highly efficient and a useful approximation in practice — particularly for the disparity image and egocylinder formulations — it can be difficult to process all of S^2 in this manner because coordinates must be chosen and patched carefully to avoid expanding points through singularities. Furthermore, C-space expansions performed in this manner also exclude more space than is technically required for a spherical vehicle to safely pass, and produce a conservative estimate of the pixel region subtended by an obstacle.

Rectangular regions are also inconvenient for the egosphere in particular because they degenerate into triangles as a pole is approached, due to convergence of meridians, and eventually lose the ability to occlude the expansion sphere using separate horizontal and vertical scans. Even if a rectangle stays away from a pole, it requires the zonal ("east-west") 1–D expansions to be excessively conservative close to the equator in order to maintain occlusion closer to the poles. If a degree of spherical character is required to adequately represent an environment (that is, a representation based on the egosphere or a sector of an egosphere), a more general approach replaces rectangles with the exact pixel region occupied by the projection of the expansion sphere — a circular region of known radius (Algorithm 1 and Figure 2.4; we assume that the poles have been assigned unambiguous pixel coordinates).

Algorithm 1 Expand obstacles on an egosphere (ordinary spherical coordinates)

Input: egospace representation E, expansion radius R,
  for $u = 1 : M$ **do** {pixel coordinates (u, v)}
    for $v = 1 : N$ **do**
      $r \leftarrow \mathcal{E}(E[u, v])$ {acquire range to obstacle}
      $r' \leftarrow r - R$
      $\sigma \leftarrow \arcsin(R/\sqrt{r'^2 + R^2})$
      for i, j such that $d((u, v), (i, j)) \leq \sigma$ **do** {pixel distance metric d}
        $E[i, j] \leftarrow \min(r', E[i, j])$
      end for
    end for
  end for

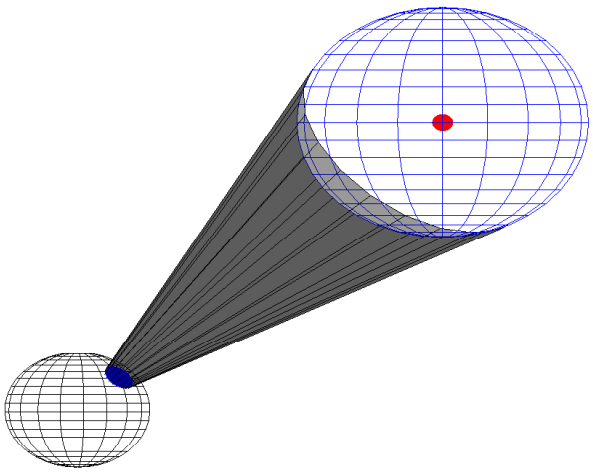

Figure 2.4: Expansion of obstacles for an egospace representation based on spherical coordinates. The red point in space is expanded by a characteristic vehicle radius (blue sphere), which is then projected onto the egospace surface (black sphere).

After the C-space expansion step and a projection, trajectories can be collision-checked immediately by comparing their generalized depth values to those of the acquired imagery — collisions occur whenever the coordinates of trajectories and obstacles coincide. In particular, paths that extend radially from the vehicle can be checked in a single operation that compares the path, which occupies a single pixel, to the generalized depth of the scene at that pixel. For a more general discretized trajectory, points in egospace can be compared to the depth data in a similar manner by comparison. In this way, the simplicity of accessing egospace obstacle data provides a significant advantage over the multi-resolution voxel approaches, which require repeated calls of complex access schemes and limit the number of candidate trajectories that can be practically checked during a planning cycle. This issue is especially apparent in tree-based multi-resolution grids, which when subjected to the above scanning example would incur the expense of a tree transversal as opposed to a simple index scan.

2.3 Egospace compactness

Although not strictly feasible unless the vehicle comes to a complete stop, radial and other attractive, yet infeasible paths are often used to seed fully dynamic motion planners [4], [20], [21] in addition to being useful in their own right at low speeds [48]–[50]. For high-speed obstacle avoidance in unknown environments, radial paths in particular are an intuitive place to start. The arrival of new information (and path continuity) will typically require revised plans that originate at the vehicle's current location, and visibility graph considerations of some sort can be highly useful in the pursuit of expeditious progress around obstacles.

A comparison of the radial path search and identification problem between egospace and Cartesian occupancy grids is illustrative not only of compactness and complexity advantages, but also demonstrates a "natural" obstacle avoidance tendency in egospace that prioritizes collision-free feasible paths within a trajectory search — a tendency that is exploited in Chapter 4 to efficiently incorporate full dynamics into the obstacle avoidance problem on an experimental aircraft. To this end, we consider a two dimensional egospace consisting of an egocylinder restricted to a single plane $(r, \theta, z = 0)$ along with an analogous uniform Cartesian occupancy grid (x, y). We then use both structures to collision-check all radially-aligned paths out to a radius R and compare performance.

For a given angular increment θ_0 of the egocylinder, we first generate an "equiva-

lent" uniform Cartesian occupancy grid by identifing the maximum size of a Cartesian grid cell that can acheive the same angular resolution at the distance R. As an estimate of the cost required to collision-check all radial paths, we count the number of operations needed to access each cell within R of the vehicle and check its occupancy (Figure 2.5). For clarity, we neglect the overhead required to maintain a representation of each radial path for later use as well as the fact that a uniform Cartesian grid must sacrifice precision at short ranges to achieve this minimum resolution at R (therefore, we are estimating a lower bound on cost).

We note that for large R and small θ_0, the length of a side of a Cartesian grid cell that subtends the angle θ_0 at the distance R is approximately $dx = \theta_0 R$. Comparing the area covered by the egospace, $A_e = \pi R^2$, to the area of a Cartesian grid cell, $dx^2 = \theta_0^2 R^2$ gives an estimate of $N = \pi/\theta_0^2$ grid cells that must be collision-checked. Letting $M = 2\pi/\theta_0$ be the corresponding number of egocylinder pixels that must be scanned during the collision-checking process gives the relation

$$N = \frac{M^2}{4\pi}, \tag{2.1}$$

which is an increase in the square of angular resolution. Assuming that element access requires an equal number of operations in both egospace and in the occupancy grid, we take the actual expense of the collision-checking process to be multiples of N and M by the same constant.

We may also consider the relative compactness of this 2D represenation scenario by estimating the memory usage of egospace and the equivalent uniform occupancy grid. We suppose that the egospace stores, up to the sensor horizon R, a 32-bit floating-point depth value in each of the M pixels for a total storage of $32M$ bits. The deterministic occupancy grid, on the other hand, stores an 8-bit, single-byte addressable boolean value at each of its N cells for a total storage of $8N$ bits.

For the purposes of a compactness estimate, rather than a radial collision-checking analysis, an occupancy grid indexed in discretized Cartesian coordinates and containing the circle of radius R would be a square with side length $2R$ — otherwise, conditions would need to be placed on the indices i and j that restrict their coverage to the circular region and are never performed in practice. Accordingly, the size of the occupancy grid is a slightly larger value $N = (2R/dx)^2 = (2/\theta_0)^2 = M^2/\pi^2$, for a total memory consumption of $8M^2/\pi^2$ bits. Using the egocylinder from the experimental regime of ([5]; discussed in Chapter 4), there are $M = 660$ pixels covering each two-dimensional plane. Assuming a sensor horizon of approximately

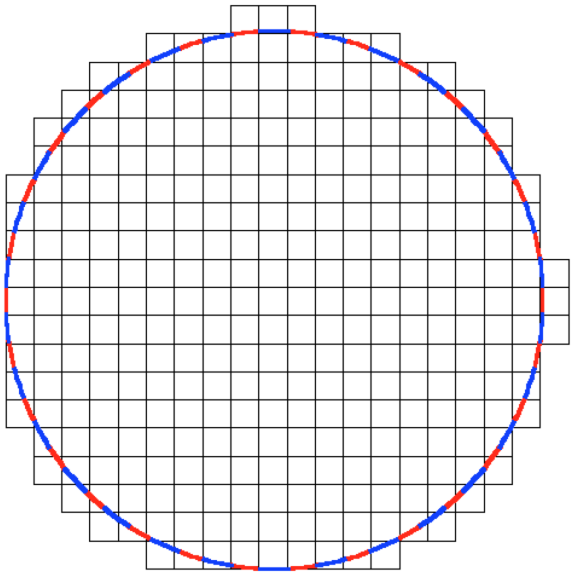

Figure 2.5: Egospace greatly simplifies a search for a collision-free radial path. For a uniform 2D Cartesian grid, identifying a radial path at an angular resolution equal to that of the alternating red and blue arcs requires collision-checking all squares inside the circle. The same search can be done in egospace over the much smaller set of arcs themselves. The number of squares that must be searched increases with the the square of angular resolution, but for egospace the search size remains linear.

$R = 50$ m as typical for reliable stereo detection, the planar structure has a total planar size of 2.6 kB. The resolution-matching occupancy grid has cells no larger than $dx = \theta_0 R = 0.48$ m on a side, which results in a grid with 44,000 cells and 44 kB of memory. A switch to egospace for the two dimensional scenario saves an order of magnitude of storage *and* provides increased resolution at closer ranges.

In three dimensions, we consider an egosphere of radius R partitioned into M pixels of uniform area, and match resolution at the same sensor horizon with a unform Cartesian voxel grid made up of cubes with side length dx. For the solid angle $\Omega = 4\pi/M$, we must have $dx^2 = \Omega R^2$ and, thus, $dx^3 = \Omega^{3/2} R^3$. Dividing the volume of the sphere by the volume of a cell gives $N = M^{3/2}/(6\sqrt{\pi})$ cells within the sphere that need to be collision-checked to consider all radial paths, for total size of the full cubic voxel grid equal $(2R/dx)^3$.

Although the rate of increase of N with M is slower than in the two-dimensional case, a full 3D egospace is typically much larger than its 2D analogue and still demonstrates a clear practical advantage. Using the dimensions of the previous egocylinder as a rough estimate for the size of an egosphere, we assume a sensor

Table 2.2: Structure Size and Memory Consumption — 50 m Sensing Radius

Structure	**# Elements**	**Memory Consumption**
Egosphere (660 × 660)	435,600	1.7 MB
Uniform Voxel Map (Cube)	5.8×10^7	50.8 MB

horizon of $R = 50$ m with $M = 660 * 660 = 435,600$ pixels for a total egospace structure size of 1.7 MB. The corresponding occupancy grid has an approximate cell size of $dx = 0.27$ m on a side, which generates a grid with 50.8 million cells and, accordingly, 50.8 MB of storage (Table 2.2). A fairer comparison, of course, would use a more representative resolution because cell size must be considerably smaller to capture the detail of egospace at close ranges near the vehicle.

We can also consider the other possible occupancy grid trade-off, sacrificing resolution to keep memory usage manageable at long range, by considering the cell size required to generate an occupancy grid equal in memory usage to our example egosphere. A 1.7 MB occupancy grid over a sensor horizon of $R = 50$ m would have 1.7 million cells covering a volume of 1,000,000 m^3, for a cell side length of approximately 0.84 m and a volume of 0.59 m^3 — a cell volume 30 times larger than that of the full resolution case.

The actual collision-checking process for radial paths on an occupancy grid is also much more onerous in practice than the previous estimates suggest because a number of time- and space-consuming issues have been neglected in our analysis for clarity. First, Cartesian grids must be populated by range sensors, which requires a grid-filling step to proceed first. For a non-probabilistic, instantaneous uniform grid like that of [4], this process combines an iteration over egospace with an unprojection into Cartesian coordinates at each pixel. In contrast, the egospace iteration could have immediately returned a radial path directly. Second, cells must be searched for *identifiable* radial paths that can be looked up and used later, which leads to additional indexing overhead compared to the simple iteration procedure that we assumed — for example, the construction of an angular histogram as in the VFH motion planning algorithm ([51]; reviewed in Chapter 3), or a large lookup table of the cells encountered in each radial path. Cartesian cells also do not readily line up with specfic angular regions, which must be calculated by considering the angles subtended by problematic cells and requires more operations than a cell-by-cell iteration.

Summary

In this chapter, we have constructed the egospace representation, which exploits a radially-aligned geometry to more compactly represent obstacles. We demonstrated how the C-space expansion technique allows egospace to be efficiently collision-checked for a spherical approximation of aircraft shape, which was made possible by a generalization to egosphere representations with singularities at the poles. We then provided a compactness comparison between egospace and the most widely used alternative, uniform voxel grids, that will motivate efficient motion planning in both a theoretical (Chapter 3) and experimental (Chapter 4) sense.

Chapter 3

CONFIGURATION FLAT MOTION PLANNING AND TRAJECTORY GENERATION IN EGOSPACE

Portions of this chapter have been modified from [26].

In this chapter, we build on the results of Chapter 2 to characterize, through theory and simulation, the use of egospace representations to inform motion planning with and without vehicle dynamics. After discussing configuration flat aircraft models and extending their traditional use in world coordinates to egospace coordinates, we develop motion planning over configuration flat dynamics in egospace coordinates and demonstrate basic performance guarantees.

The rest of the chapter highlights a number of useful approximations and specializations of the full configuration flatness problem and discusses their limits of validity. We first evaluate the conditions under which vehicle dynamics are significant to a motion planning problem at all, which depend on the operating environment as much as they depend on the vehicle performance, and leads to an estimate of obstacle sparsity relative to the dynamics. We then discuss the requirements of complete motion planning, as distinguished from pure obstacle avoidance, and illustrate the adaptation of complete world-space algorithms to egospace. We conclude by considering the related notions of of motion primitives and configuration flat trajectories, which allow online integration of state equations to be entirely avoided by exploiting predefined problem stucture.

3.1 Introduction

Motion Planning

We consider a robot, without dynamics for the time being, operating in a C-space C and located at an initial configuration $\mathbf{q}_0 \in C_{\text{free}}$. A *motion planner* seeks to identify a trajectory of valid configurations, $\mathbf{q}(t) \subset C_{\text{free}}$, such that the robot proceeds from $\mathbf{q}(0) = \mathbf{q}_0$ to a *goal state* $\mathbf{q}(t_f) = \mathbf{q}_g$, at a potentially unspecified time t_f. Each such trajectory may also be assigned a cost J. A rich variety of motion planning algorithms that can solve this problem have been developed over the past few decades of research, and the performance of one algorithm versus another is largely dependent on the nature of the obstacle representation and the state transition process.

For a mobile point robot, the simplest motion planning problems are on a discrete C-space, in which the set of valid configurations is finite or countable. A robot can move from configuration to configuration using a state transition, to which is associated a cost that measures its expense or desireablilty. The motion planner attempts to to return a sequence of transitions $\mathbf{u}_k$ and configurations $\mathbf{q}_k$ that brings the vehicle from the start to the goal, perhaps with a minimal (or maximal) cost $J = \sum J_k$. This problem generally has the structure of a graph search, for which a number of extremely efficent algorithms have been developed (see [38] for a survey). Popular examples include Dijkstra's algorithm, which is guaranteed to find an optimal path [52], or the related A^* search [53], which attempts to speed up Djikstra's algorithm by considering a future cost heuristic and is optimal under mild conditions.

A motion planner is said to be *complete* if it will always return a path from start to goal if a valid path exists, or return a failure in finite time if such a path does not.

On a continuous C-space, such as a point aircraft translating in $\mathbb{R}^3$ or a point ground vehicle in $\mathbb{R}^2$, it is impossible to visit every point in the continuum and compromises must often made according to the obstacle representation. The first distinction that must be made is between reactive approaches, which are essentially local in nature and return instantaneous actions, and deliberative approaches, which determine a future trajectory over a non-zero planning horizon.

The simplest local planning algorithms are contact-based "bug algorithms" in two dimensions (such as [28]), which attempt a direct path to the goal if possible, but in the event of contact with an obstacle revert to a wall-following behavior until a direct path is once again available — the criteria used to determine "availability" distinguishes the different versions. Only a contact sensor is required to use the basic bug algorithms, but the "tangent bug" [48] generalization also assumes that a sensor with finite detection range is available. Although they are easily shown to be complete in two dimensions (a consequence of the Jordan curve theorem; [54]), bug algorithms generalize poorly to three dimensions because it is impossible to partition $\mathbb{R}^3$ into two sets using a single space curve.

Potential field methods calculate a local action by analogy to electrostatics, in which the robot is assumed to be a "charged particle" under the influence of an attractive, opposite-charged goal and repulsive, like-charged obstacles [38]. The robot follows the negative gradient of a "potential field", as in the relation between electric field and electric potential, calculated from the attractive and repulsive regions of an *a priori* known C-space. Potential field methods produce elegant paths when

obstacles are widely spaced and are applicable to a wide variety of explicit and discretized representations. A naive choice of potential can produce local minima in which a gradient-following robot would find itself stuck, however, and potential fields are also somewhat unpredictable in general environments — oscillatory behavior and poor corridor identification is often observed in cluttered areas such as doorways (as noted in the introduction of [51]). A related class of two-dimensional methods, the vector field histogram [51] and its modifications [55], [56], address the loss of directional information associated with compression into a potential field by condensing the local obstacle layout, as represented by a Cartesian occupancy grid, into an angular-indexed histogram that records the obstacle density in each direction. At each instant, the robot chooses a safe path from the available angles in the histogram, with the more advanced methods restricting the choice to dynamically feasible trajectories and offering completeness guarantees using higher-level lookahead. For MAVs, the mapping operation in [13] employs VFH+ on horizontal 2D slices of a 3D, uniform occupancy grid combined with a wall-following behavior.

Deliberative approaches, on the other hand, take into account the global structure of the environment and produce a plan that is collision-checked in advance, for safety, and reused over a number of data acquisition and control cycles. These methods are mostly divided into combinatorial path planning, which assume an explicit representation of the environment and construct exact paths, and sampling-based path-planning, which consider only a subset of possible configurations in C-space chosen using a probabilistic or determinsitic selection criterion.

Combinatorial methods consider the exact geometry of a region as given by explicit geometric primitives and always offer at least a completeness guarantee, but are often impractical for field operations and are primarily of academic interest. Most combinatorial methods rely on the explicit construction of a *roadmap* of C-space, which is a set of collision-free paths between important points in C-space onto which a vehicle can merge in to order to proceed from an arbitrary starting state to an arbitrary goal state. Popular roadmap generation techniques include visibility graphs or cellular decompositions for polygonal or polyhedral obstacles, onto which connections from the start and goal states are attempted. Once a roadmap is at hand, a graph search determines a path from the start to the goal.

Sampling-based methods sacrifice the classical notion of completeness in order to make progress in complex environments, and are used in most real mobile robotic systems. Because it is impossible to evaluate every sequence of points in C-space,

a set of configurations is chosen as a sample set to which motion planning is restricted. The most straightforward way to do so is to sample in a deterministic manner — for example, to overlay C-space with a Cartesian grid and consider only grid points — and attempt a discrete graph search among the sampled points. Probabilistic single-query searches, such as the rapidly exploring random tree [49] and its variants (particularly, the asymptotically optimal RRT* algorithm of [57]) use random sampling to build a search tree that "grows" into the space and more efficiently explores for a path. Multiple-query options typically attempt to generate a roadmap from samples, as in the probabilistic roadmap (such as the PRM algorithm of [50] and its widely used relative Lazy PRM: [58]). Modified completeness criteria, such as resolution completeness (if a path exists, it can be found in the limit of infinite sample resolution), and probabilistic completeness (if a path exists, it can be found with probability 1 as the number of samples grows to infinity), are typically used for performance evaluation of sampling based methods, and are generally an acceptable compromise in the pursuit of progress for otherwise intractible problems.

Motion planning and dynamics

We now generalize to a robot with dynamics — that is, it posesses a state vector $\mathbf{x}(t)$ that includes at least the configuration variables $\mathbf{q}$ (for a general 3D rigid body, $\mathbf{q} \in \text{SE}(3)$) along with a set of internal variables (for our rigid body, this could include derivatives of $\mathbf{q}$). The state vector is contained in *phase space* that evolves in time according to the dynamical state equations $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, where $\mathbf{u} \in \mathbb{R}^m$ is the vector of m control inputs to the state equations.

Although any consistent representation of obstacles can, in theory, serve as an input to an optimal control problem, motion planning over dynamics is often performed using a "seed" approach that attempts to modify a dynamics-free motion plan until it satisfies the state equations. A popular approach for aerial robots is convexification, in which a preliminary candidate path is found using dynamics-free methods and used to build a set of continguous, obstacle-free convex regions in which it is contained. The optimal control problem then proceeds as usual over the restricted convex regions, which simplifies its domain and assists the optimization procedure. These approaches differ primarily in how they seed the optimal control calculation and the assumed structure of the control problem. For spacecraft, SE-SCP [59] uses a novel sampling step to explore C-space and identify spherical obstacle-free regions, over which minimum fuel paths calculated using a sequential convex pro-

gram. The quadcopter motion planner of [4], [21] builds a set of ellipsioids around the result of a graph search over the cells of a Cartesian occupancy grid, which are then grown into convex obstacle-free regions. A minimum jerk polynomial trajectory is then is calculated over the convex regions using a quadratic program on its coefficients.

A different seeding strategy avoids the online optimal control problem altogether by discretizing the set of vehicle actions into feasible *motion primitives* of a fixed form — the maneuvers are then linked in sufficiently smooth fashion online and simply replace straight lines in most local and sampling-based methods. In particular, smooth 3D turns for an aircraft with agility constraints were used in combination with optimal aggressive turn-around procedures in [60] for flight in a cluttered forest environment. Motion primitives for a quadcopter are calculated offline and fixed to a grid in [61], with feasible sequences determined online using a graph search.

Trajectory Generation

The trajectory generation problem is the identification of an open-loop control input $\mathbf{u}(t)$ ($\mathbf{u}[k]$ for the discrete-time case, $k \in \mathbb{Z}$) that will bring the system from its current state $\mathbf{x}_0$ to a future state $\mathbf{x}_f$ according to its dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ (in the discrete case, $\mathbf{x}[k+1] = f(\mathbf{x}[k], \mathbf{u}[k])$. This step is the defining feature of the two degree-of-freedom (2-DOF) control paradigm, in which the open loop input is used in combination with a feedback loop. This structure improves control performance and allows for tighter system operation than is possible using feedback error signals alone.

Typically, $\mathbf{u}$ is determined using optimal control techniques. The Pontryagin minimum (or maximum) principle [62] offers necessary conditions for a trajectory to be optimal, and in simpler cases can be used to calculate optimal inputs by explicitly solving the two-point boundary value problem (BVP) it generates. For a typical nonlinear plant model, however, this approach has no closed form solution and numerical methods are required.

In many practical cases, however, additional problem structure allows progress to be made without resorting to direct numerical solution of a two-point BVP. For a differentially flat system (as briefly mentioned in Chapter 1 and discussed in detail below), a set of basis functions, typically polynomials, can be used to represent trajectories in a smaller set of variables known as *flat outputs*. Coefficients are determined by the boundary conditions and propagated through the plant model to

determine the other states as well as the control inputs. The set of basis functions can also reduce a trajectory optimization procedure to a problem with a much simpler form — particularly, [19] assumes a polynomial form for space trajectories of a quadcopter in $\mathbb{R}^3$ in order to minimize snap using a quadratic program. Although not reliant on differential flatness, trim primitives solve the trajectory generation problem [22] by selectively enabling constant control inputs and integrating forward, possibly numerically, to produce a set of maneuvers with known input and known output.

A trajectory generation procedure that calculates a control input out to a time horizon T that is subsequently revised before it is completely executed is known as *receding horizon* (for nonlinear systems, see [63]). At a replanning time t_0, for which the system state is $\mathbf{x}(t_0)$, a receding horizon controller will calculate an optimal control input $\mathbf{u}^*(t)$ to a desired horizon state $\mathbf{x}(t_0 + T)$ over the time interval $[t_0, t_0 + T]$. The optimal control input is executed until the next replanning time (which occurs before $t_0 + T$). The receding horizon paradigm seeks to introduce a feedback component into trajectory generation by solving the optimization problem over the future states at each planning step — online measurements and, potentially, a system identification component can then be incorporated into the planning process.

3.2 Theoretical development

Because obstacles can be represented compactly in egospace without having to construct a 3D world model, it is a natural extension of the egospace data structure to also use the same coordinates for motion planning and obstacle avoidance. The use of visual features and coordinates to plan motion has extensive precedent in visual servoing and manipulation literature, including airborne manipulators aboard MAVs [64]. Other image-based motion planning schemes have used optical flow for reactive corridor-following with a quadcopter [37] as well as an image-plane representation used to plan motion of a ground vehicle with negligible dynamics [44]. Existing visual methods for MAVs do not readily accommodate a feedforward control component that can be efficiently collision-checked in advance, however, which is essential for accurate trajectory following and deliberative planning for MAVs at high speeds. In this chapter, we specialize to the class of configuration flat aerial robots and develop deliberative, feedforward motion planning in egospace using the geometric results derived previously.

Configuration flat aerial vehicles are well-equipped to handle real-time dynamical obstacle avoidance using visual coordinates because their trajectories and controls can be expressed in terms of the same configuration variables in which obstacles are represented (see Chapter 4 of [65]).

Definition 2. *A robot with state variables* $\mathbf{x}$*, control inputs* $\mathbf{u}$*, and state equations* $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ *is called configuration flat if the configuration variables (see Chapter 2)*

$$\mathbf{q} = \alpha(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ldots, \mathbf{u}^{(k)}),$$

are a smooth function α *of the states and control inputs, with smooth functions* β *and* γ *such that*

$$\mathbf{x} = \beta(\mathbf{q}, \dot{\mathbf{q}}, \ldots, \mathbf{q}^{(j)}),$$

$$\mathbf{u} = \gamma(\mathbf{q}, \dot{\mathbf{q}}, \ldots, \mathbf{q}^{(j)}).$$

The configuration variables are known as flat outputs of the system, with obstacle and configuration spaces $O \subset C = \{\mathbf{q}\}$ *and a free space* $C_{free} = C \setminus O$ *open in which valid configuations lie.*

Because their spatial obstacles can be expressed equivalently in $\mathbb{R}^3$ or in egospace, trajectories and controls for a point configuration flat robot can also be expressed equivalently in terms of egospace coordinates.

Proposition 1. *A point-mass robot that is configuration flat in world coordinates is also configuration flat in egospace wherever the egospace map is well-defined.*

Proof. By definition of configuration flatness, all state variables $\mathbf{x}$ and controls $\mathbf{u}$ can be written uniquely in terms of the configuration variables $\mathbf{q}$ and their derivatives — in this case, a sufficiently smooth trajectory in world coordinates $\mathbf{r}(t) \in \mathbb{R}^3$ so that

$$\mathbf{r} = \alpha(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ldots, \mathbf{u}^{(k)}),$$

$$\mathbf{x} = \beta(\mathbf{r}, \dot{\mathbf{r}}, \ldots, \mathbf{r}^{(j)}),$$

$$\mathbf{u} = \gamma(\mathbf{r}, \dot{\mathbf{r}}, \ldots, \mathbf{r}^{(j)}).$$

Assuming that the egospace map is sufficiently smooth, we calculate the trajectory in egospace coordinates $\mathbf{s}(t) = \mathcal{E}^{-1}(\mathbf{r}(t))$, and after inversion and composition exhibit the smooth flatness relations

$$\mathbf{s} = \mathcal{E}^{-1} \circ \alpha(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ldots, \mathbf{u}^{(k)}),$$

$$\mathbf{x} = \beta(\mathcal{E}(\mathbf{s}), \frac{d}{dt}[\mathcal{E}(\mathbf{s})], \ldots, \frac{d}{dt^j}[\mathcal{E}(\mathbf{s})]),$$
$$\mathbf{u} = \gamma(\mathcal{E}(\mathbf{s}), \frac{d}{dt}[\mathcal{E}(\mathbf{s})], \ldots, \frac{d}{dt^j}[\mathcal{E}(\mathbf{s})]).$$

□

Configuration flat dynamics are particularly advantageous for sampling-based motion planning algorithms because only the flat outputs and their derivatives must be specified and considered for waypoint and trajectory generation. Smooth configuration flat robots also inherit the existence of time optimal trajectories between points from differential flatness [66], which greatly restricts the number of trajectories connecting two sampled points (possibly a single trajectory for certain plant models and boundary conditions) and can accommodate the extreme maneuvers that may be required to successfully evade obstacles. Insisting on time optimality avoids the need to sample over the entire space of control inputs and also allows for any feasible trajectory to be approximated by a finite sequence of segments.

Lemma 1. *Any continuous feasible path in the flat outputs of a configuration flat robot* $\mathbf{q}(t)$ *with finite duration is arbitrarily close to a sufficiently smooth and feasible sequence of finitely many locally time-optimal segments* $\{\mathbf{q}_i^*\}$.

Proof. For a configuration flat robot, time-optimal trajectories exist between any two points in the absence of obstacles and, by construction, satisfy the vehicle dynamics and constraints. Furthermore, any sufficiently smooth sequence of locally time-optimal trajectories will also satisfy the dynamics and the constraints.

Let $\mathbf{q}(t)$ be any feasible trajectory in the flat outputs of a configuration flat robot on the interval $t \in [0, t_f]$. Because $\mathbf{q}(t)$ is uniformly continuous on the closed time interval (a consequence of the Heine-Cantor theorem), for any $\epsilon > 0$, there exists $\delta > 0$ such that, for all $s \in (0, t_1)$ with $t_1 < t_f$,

$$t_1 < \delta \implies \|\mathbf{q}(s) - \mathbf{q}(t_1)\| < \frac{\epsilon}{2}. \tag{3.1}$$

For a configuration flat robot, there also exists a uniformly continuous time optimal trajectory $\mathbf{q}_1^*(t)$ between $\mathbf{q}(0)$ and $\mathbf{q}(t_1)$ with duration $t_1^* \leq t_1$. Accordingly, there exists $\delta^* > 0$ such that if $t_1 < \delta^*$ and $s^* \in (0, t_1^*)$,

$$\|\mathbf{q}_1^*(s^*) - \mathbf{q}(t_1)\| < \frac{\epsilon}{2}. \tag{3.2}$$

Adding Equations 3.1 and 3.2 and applying the triangle inequality gives

$$t_1 < \min(\delta, \delta^*) \implies \|\mathbf{q}(s) - \mathbf{q}_1^*(s^*)\| < \epsilon, \tag{3.3}$$

which implies that the time optimal segment and the non-optimal segment can be made arbitrarily close by choice of t_1. Similarly, a sufficiently accurate time-optimal approximation originating at some other time $t_i \in (0, t_f)$ may be found by considering the trajectory $\mathbf{q}(t - t_i)$ and repeating the above analysis.

We now exhibit a continuous sequence of finitely many time optimal segments $\{\mathbf{q}_i^*\}$, between $\mathbf{q}(0)$ and $\mathbf{q}(t_f)$, that is at every point no further than a tolerance $\epsilon_q > 0$ from some point on the trajectory $q(t)$. Partition $[t_0 = 0, t_f]$ into $N + 1$ times $\{t_0, t_1, t_2, ..., t_f\}$. For each $i \in \{1, ..., N\}$ calculate a time optimal trajectory segment $\mathbf{q}_i^*$, with duration t_i^*, between $\mathbf{q}(t_{i-1})$ and $\mathbf{q}(t_i)$. Refine the partition until for each i, $\|\mathbf{q}(s) - \mathbf{q}_i^*(s^*)\| < \epsilon_q$ for all $t_{i-1} < s < t_i$ and $0 < s^* < t_i^*$.

By uniform continuity, $N \leq \frac{t_f}{\min_i (t_i - t_{i-1})}$ is finite. □

Combining the flat output sample space with an insistence on optimal connections allows for a resolution-complete and dynamically feasible motion planning algorithm to be constructed from any resolution-complete algorithm that neglects dynamics.

Theorem 1. *Suppose the flat outputs of a smooth configuration flat robot with bounded inputs are sampled and connected according to any resolution-complete motion planning algorithm without regard to dynamic feasibility. If a modified algorithm attempts to make these connections with time-optimal trajectories* $\mathbf{q}_i^*$ *and rejects any that result in a collision, then it is also resolution-complete over the vehicle dynamics.*

Proof. The sampling and proposed connection of points is performed in a resolution-complete fashion. In the limit of infinite resolution, every sequence of sampled points — and, accordingly, every possible smooth sequence of time optimal trajectories in the modified algorithm — is eventually evaluated, if necessary, by definition.

Let $\mathbf{q}(t) \subset C_{\text{free}}$ be any feasible path from start to goal. By the preceding lemma, we may find a smooth and feasible sequence of time optimal trajectories $\{\mathbf{q}_i^*\}_\epsilon(t)$ such that, for any $\epsilon > 0$, $\|\mathbf{q}(t) - \{\mathbf{q}_i^*\}_\epsilon(t)\| < \epsilon$ at each t — where we have also

reparameterized each time optimal segment so that its duration matches that of the feasible path segment to which it runs parallel.

Because C_{free} is open, we may select $\epsilon = \epsilon^*$ sufficiently small such that $\{\mathbf{q}_i^*\}_{\epsilon^*}(t) \subset C_{\text{free}}$.

By resolution completeness, some such sequence $\{\mathbf{q}_i^*\}_{\epsilon^*}(t)$ is eventually considered and returned by the modified algorithm. □

Additionally, there is never a need to leave egospace to accommodate a spatial motion planning algorithm over configuration flat dynamics, because completeness and soundness properties always carry over from world space.

Theorem 2. *Any complete spatial motion planning algorithm is also complete in egospace coordinates wherever the egospace map is well-defined.*

Proof. A complete motion planning algorithm always returns, possibly in a resolution or probabilistic limiting sense, a collision-free sequence of trajectory segments $\{\mathbf{q}_i(t)\}$ between the start and goal if such a path exists. By invertibility of the egospace map, if a motion plan exists its egospace representation $\{\tilde{\mathbf{q}}_i(t)\}$ is also collision-free, so after a composition we may return the individual segments $\tilde{\mathbf{q}}_i(t)$ directly and exhibit a complete motion planning algorithm in egospace coordinates. □

Corollary 2.1. *If the robot is also configuration flat in world coordinates then all controls and state variables can be determined algebraically in terms of the egospace motion plan and its derivatives.*

Proof. The corollary follows directly from Proposition 1. □

The preceding two theorems suggest a natural way to proceed with motion planning in egospace coordinates that uses both the invertibility and configuration flatness structures. We may simply adapt standard world coordinate motion planning algorithms from the literature, generating equivalent trajectory segments in egospace instead. The first theorem also hints at an efficient way of incorporating dynamics: the use of time optimal primitives, also expressed in egospace and linked online. Configuration flatness allows control inputs to be extracted immediately.

To illustrate the simplicity of this approach, we now introduce a reactive obstacle avoidance scheme under trivial dynamics (infinite agility), and sequentially modify

the example to identify and accommodate the conditions under which full motion planning and configuration flat dynamics must be considered.

3.3 Reactive Obstacle Avoidance

We distinguish obstacle avoidance, in which the objective is to simply avoid collisions while not necessarily reaching a destination, from motion planning, in which the objective is to always reach the destination if possible and a completeness guarantee is required in some sense. A lightweight and successful obstacle approach that motivates our analysis is that of [45], in which an egocylindrical obstacle representation was used to steer a quadcopter, assumed to have infinite agility, onto collision-free radial paths through a forest towards a goal. In this section, we generalize this simple obstacle avoidance scheme to arbitrary egospace geometries and establish the vehicle and obstacle regimes in which an infinite agility approximation is valid.

Avoidance under Infinite Agility

Because egospace always has a coordinate parallel to the radial unit vector, trajectories that extend radially from the origin occupy a single pixel location (u_0, v_0) in egospace. An egospace trajectory is also, in general, collision-free if it nowhere has the same pixel coordinates and depth as an obstacle, so it follows immediately that a radial path around an obstacle can be collision-checked simply by comparing the furthest point in the path to the depth of the egospace obstacle representation at that pixel. Accordingly, an infinitely agile vehicle can navigate around obstacles and maintain a time-to-contact constraint, chosen as a design parameter to ensure a margin of safety, by simply scanning the pixels of egospace and choosing an appropriate target to aim at.

The vehicle searches the egospace obstacle representation for the target pixel, closest to the destination, that also satisfies the time-to-contact criterion — a low-level controller then aligns the velocity vector with the chosen target (algorithm 2). If the time-to-contact constraint cannot be satisfied after a scan, the vehicle repeats the selection process with a decreased speed. It is clear that the simple avoidance procedure presented here is collision-free in the infinite agility limit — because the lookahead horizon decreases to zero with time-to-contact, the vehicle can always reduce the time-to-contact search constraint until it can identify and turn onto an acceptable path.

Although this approach is adequate for avoiding isolated obstacles that are sparse

Algorithm 2 Infinite agility reactive obstacle avoidance

Input: egospace E, depth threshold δ_c, destination pixels (u_{dest}, v_{dest})
Output: target pixel (u_t, v_t)

```
while true do
  d_min ← ∞
  for u = 1 : M do
    for v = 1 : N do
      if E[u,v] ≥ δ_c then {collision-check: assume δ(r) increases with r}
        d ← sqrt((u − u_dest)^2 + (v − v_dest)^2)
        if d < d_min then
          (u_t, v_t) ← (u, v)
          d_min ← d
        end if
      end if
    end for
  end for
  if d_min < ∞ then
    return (u_t, v_t){pick closest collision-free pixel to destination}
  else
    δ_c ← δ'_c {δ'_c < δ_c}
  end if
end while
```

relative to the vehicle dynamics, on a real system collisions become possible as speeds increase and obstacles become denser. To determine the limits of validity quantitatively, we evaluate the ability of a quadcopter, in constant speed, level flight, to follow a weaving trajectory. Assuming that the roll axis is always aligned with the velocity vector and that the only permitted acceleration is directed normal to it (to maximize agility), we consider the simple plant model

$$\dot{x} = v\cos(\theta), \tag{3.4a}$$

$$\dot{y} = v\sin(\theta), \tag{3.4b}$$

$$\dot{\theta} = \frac{g}{v}\tan(\phi), \tag{3.4c}$$

$$\ddot{\phi} = u. \tag{3.4d}$$

where v is the constant vehicle speed, θ is the heading angle, ϕ is the roll angle, and g is the acceleration due to gravity. The control input and roll angle are also subject to the saturation constraints $|u| \leq u_{\max}$ and $|\phi| \leq \phi_{\max}$, which are related to the mass, moment of inertia, and maximum thrust of the vehicle and ultimately limit its agility.

We consider the ability of the vehicle to track a sinusoidal trajectory in heading, $\theta(t) = A\sin(\omega t)$, in which the parameters A and ω are estimates of maximum turn severity and frequency in a particular environment. If the reference trajectory saturates the constrained variables at the most severe point of the trajectory, we conclude that it cannot be tracked by the vehicle without slowing down and that the infinite agility assumption can no longer be safely made. We ignore the x and y dynamics, which play no role in determining the agility of the vehicle, and for clarity assume that level flight limits the vehicle roll angle to small values — in addition to providing a conservative and simpler estimate of vehicle agility, roll angles must remain small for the vehicle to simultaneously execute a sharp turn and also support its own mass against gravity anyway. Accordingly, linearizing about $\phi = 0$ gives the linearized plant model

$$\dot{\theta} = \frac{g}{v}\phi, \tag{3.5a}$$

$$\ddot{\phi} = u. \tag{3.5b}$$

To identify the fastest speed for which the trajectory can still be tracked, we substitute the reference trajectory and assuming that the constrained variables saturate at the most severe point of the trajectory. This produces two distinct failure criteria in ϕ and u,

$$v = \frac{g}{A\omega}\phi_{\max}, \tag{3.6a}$$

$$v = \frac{g}{A\omega^3}u_{\max}, \tag{3.6b}$$

which are limited by, respectively, the ability of the vehicle to provide sufficient thrust to track the turn and its ability to roll quickly enough to bring its thrust to bear.

Independently of whether or not the dynamics satisfy the tracking criteria, however, collision-free obstacle avoidance does not imply that the vehicle will always reach its destination in any environment — the vehicle can become trapped by either entering a closed loop or reducing its own velocity to zero. In the next section we remedy the limitations of the naive scheme and present complete egospace motion planning over full configuration flat dynamics.

3.4 Complete Motion Planning and Full Dynamics

In obstacle environments that are two-dimensional and require no altitude changes to reach a destination, the infinite agility procedure can easily be made complete by

also including a wall-following behavior — as is done in the popular "bug" algorithms for motion planning. This solution is not entirely satisfactory for MAVs in general environments, however, because complete wall-following behavior is inefficient on a platform with limited battery life and does not easily generalize to three dimensions. Deliberative planning in egospace is a much simpler option, and can accommodate popular approaches with all the same representation and collision-checking advantages of the reactive scheme.

Infinite-agility motion planning is readily extended to the full dynamic case through the use of motion primitives, which are trajectory segments that are precomputed either numerically or analytically. The primitives are expressed directly in egospace coordinates and simply replace the straight radial segments used in previous sections with trajectories that satisfy a more realistic vehicle model. Although trajectory generation itself can always be performed directly in egospace coordinates, the primitives described here are generated in the typically much simpler world coordinate plant model and then projected, algebraically and in advance, using the egospace map. The primitives are then used to connect waypoints in egospace as part of the trajectory generation segment of any motion planning procedure, which remains valid at higher speeds with minimal extra overhead.

After modifying the infinite-agility obstacle avoidance procedure to include a completeness guarantee, we introduce egospace motion primitives and combine the two to provide a general approach to the MAV motion planning problem.

Completeness under Infinite Agility

As shown in an earlier section, any motion planning algorithm that is complete in world coordinates is also complete in egospace — care must be taken, however, to properly sample points in egospace and connect them with lines properly. Selecting u and v coordinates separately from uniform distributions, for example, will not generally produce a uniform distribution of pixels (u, v) in egospace, and the equation of a linear interpolation between points in egospace will be different in general from a linear interpolation between points in Cartesian coordinates.

Once an appropriate sampling or point selection procedure is chosen, however, motion planning can proceed under the slight modifications suggested by Theorem 1. For a sampling-based algorithm, the correct choice of a distribution in egospace immediately satisfies completeness and follows an identical procedure as in world coordinates — albeit with the more efficient collision-checking scheme by com-

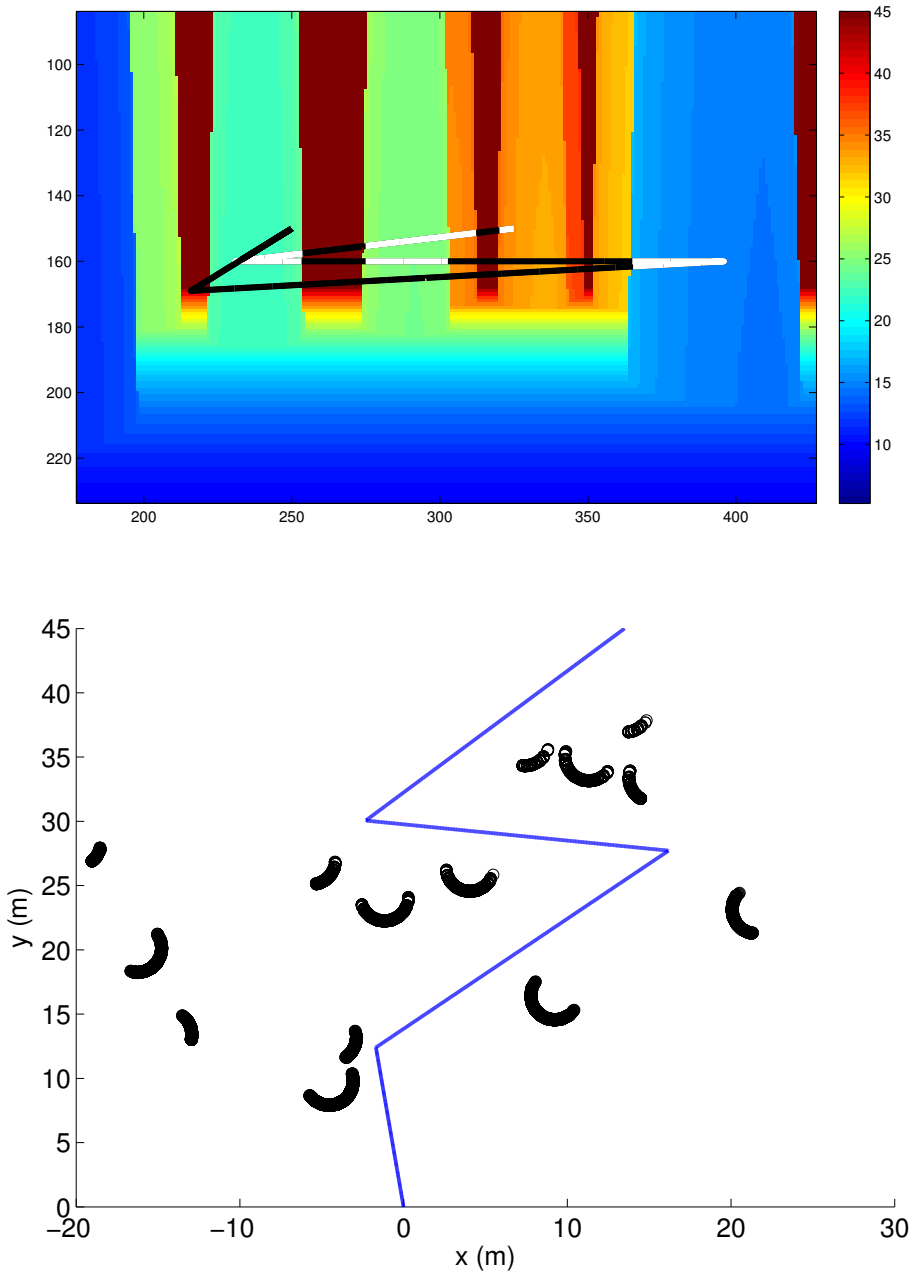

Figure 3.1: A Lazy PRM motion planner implemented entirely in depth-image coordinates (top) for an MAV with negligible dynamics in a simulated forest environment. In this depth image, pixels with cooler colors (blue trees, foreground) are closer than pixels with warmer colors (green, yellow, and orange trees with red background at sensor horizon). An equivalent point cloud and world-space representation is also shown (bottom, viewed from above). Points are sampled, selected and connected entirely in the egospace coordinates, for which the projected pixel coordinates of the motion plan are shown. White segments are obscured by obstacles, but pass safely behind them by a predetermined safety margin. The motion plan is calculated for a particular view and followed until updated imagery becomes available.

parison as mentioned in the introduction. For a simulated obstacle data set with no apriori known global structure, we exhibit an egospace implementation of the Lazy PRM algorithm [58] generated using this approach (Figure 3.1) with an assumed safety margin for trajectory segments that pass into unknown areas behind obstacles.

Motion Primitives, Configuration Flatness, and Full Generality

Motion primitives for planning with aerial robots were introduced in [22] as "trim primitives" inspired by aircraft maneuvers in which one or more inputs are held at a constant value for a period of time. We continue to leverage precomputation by employing trim primitives integrated in advance and converted into egospace coordinates for immediate collision-checking. For the most frequently used MAV models, the trim primitives can be readily integrated forward in closed form when inputs are constant. For both this reason and the typically increased complexity of plant models in egospace coordinates, we develop the set of trim primitives explicitly in terms of the initial vehicle state, the constant input value, and the final vehicle state and *then* convert into egospace coordinates to produce a set to be used online for motion planning. This strategy also allows for the simultaneous use of primitives in world coordinates elsewhere to simplify low-level vehicle control — state estimation, hardware inputs, and mission-level objectives are all measured or expressed most conveniently in terms of world coordinates and can immediately use the set of world primitives without a pass through the projection function, while obstacle avoidance and motion planning can use egospace primitives that are more convenient to collision-check.

As a simple example, we exhibit the set of egospace trim primitives for a simplified quadcopter model, based on a Dubins car, that remains torsion-free but can be constrained to any plane in $\mathbb{R}^3$ rather than just the x–y plane. In addition to being much simpler than the Dubins airplane generalization of [67], this "Dubins helicopter" abstraction is more appropriate for the multicopter platforms that have come to dominate research in MAV autonomy — unlike the Dubins airplane, which must maintain a predetermined groundspeed and can only climb using helical maneuvers, the Dubins helicopter model allows for purely vertical ascents as well as a flexible groundspeed that can equal zero without disrupting trajectory generation. Dubins helicopter trajectories are defined by a starting point $\mathbf{r}_0$ with an initial velocity vector $\mathbf{v}_0$ and a relative destination $\mathbf{x}_f$ vector that together determine a unit normal vector $\mathbf{n} = (\mathbf{v}_0 \times \mathbf{x}_f)/\|\mathbf{v}_0 \times \mathbf{x}_f\|$ to the plane in which the trajectories lie.

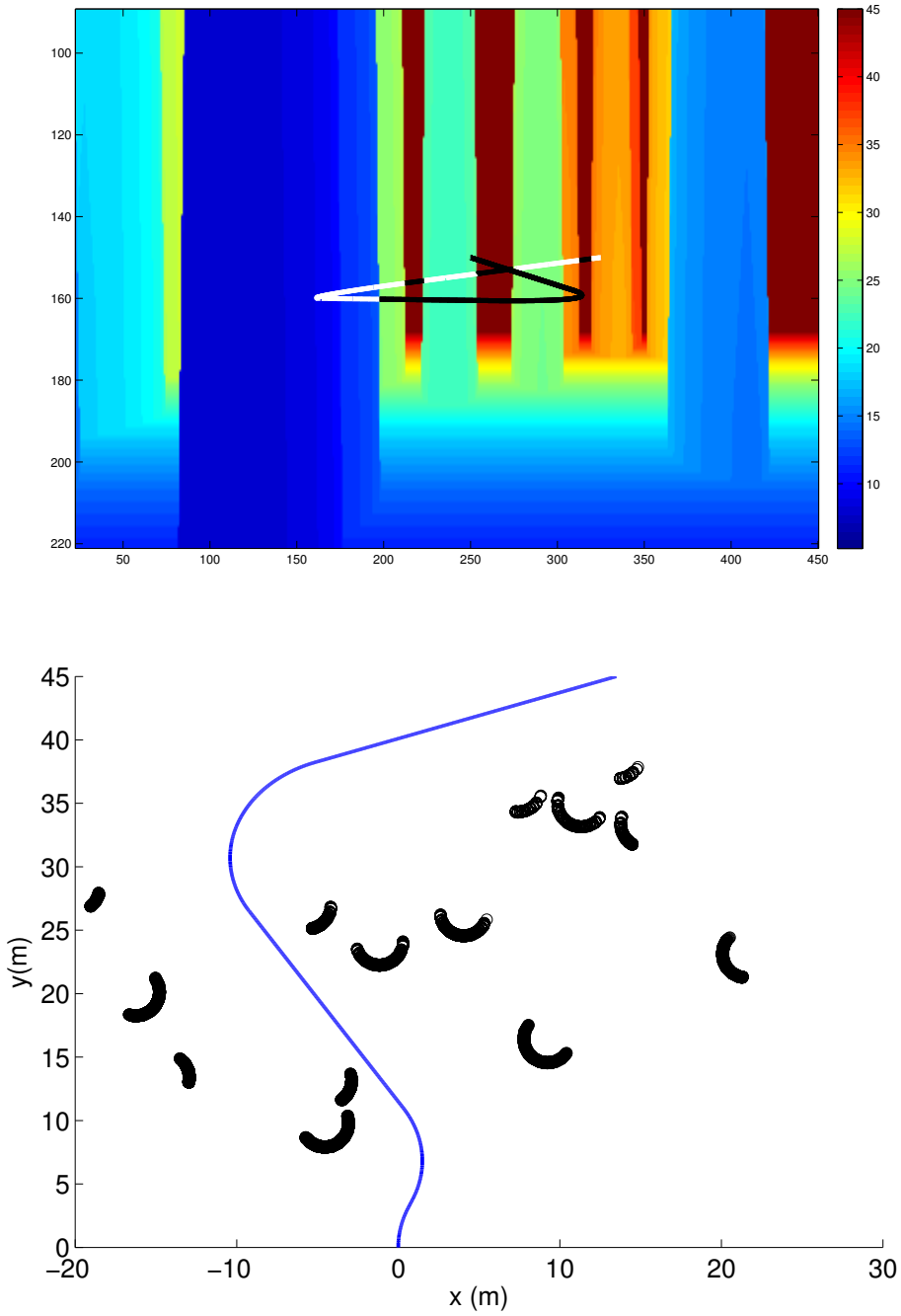

Figure 3.2: An egospace Lazy PRM algorithm adapted to Dubins helicopter dynamics by replacing straight-line segments in egospace with the equivalent projected 3D torsion-free Dubins paths (top), with the equivalent point cloud and feasible path in world space (bottom).

Using the planar assumption and the Rodrigues rotation formula gives the simple Dubins helicopter plant model

$$\dot{x} = v_{x0}\cos(\theta) + (b_x)\sin(\theta), \tag{3.7a}$$

$$\dot{y} = v_{y0}\cos(\theta) + (b_y)\sin(\theta), \tag{3.7b}$$

$$\dot{z} = v_{z0}\cos(\theta) + (b_z)\sin(\theta), \tag{3.7c}$$

$$\dot{\theta} = u_\theta, \tag{3.7d}$$

where (v_{x0}, v_{y0}, v_{z0}) are the constant components of the initial velocity vector $\mathbf{v}_0$, (b_x, b_y, b_z) are the constant components of the vector $\mathbf{b} = \mathbf{n} \times \mathbf{v}_0$, and $|u_\theta|$ is limited to a maximum value κ. In addition to having available all the analytic advantages of the regular Dubins car, the Dubins helicopter plant model also always has $\theta = 0$ at $t = 0$ to match the specified initial velocity. Setting the control input u either off and equal to zero or, on at a constant value, and integrating produces two distinct classes of trim primitives: straight-line trajectories, which include straight and level flight as well as pure climbs and descents, and circular turns that are either level or include a climbing component (at its most extreme, a full vertical loop) depending on $\mathbf{n}$. It also follows immediately from the Pontryagin minimum principle that if the control inputs are allowed to saturate, any time-optimal path between two points in space must be made up of a combination of these two primitives. The primitives are integrated from the constant control inputs with a general initial vehicle state and projected into egospace (Tables I and II) to replace the straight line trajectory segments proposed in the previous section.

Table 3.1: Dubins helicopter trim primitives in Cartesian (x, y, z) coordinates.

	(x, y, z)
Straight Line $(u_\theta \equiv 0)$	$x(t) = x_0 + v_{x0}t$ $y(t) = y_0 + v_{y0}t$ $z(t) = z_0 + v_{z0}t$
Circular Turn $(u_\theta \equiv \pm\kappa)$	$x(t) = x_0 + \dfrac{v_{x0}\sin(\kappa t) \pm b_x(1 - \cos(\kappa t))}{\kappa}$ $y(t) = y_0 + \dfrac{v_{y0}\sin(\kappa t) \pm b_y(1 - \cos(\kappa t))}{\kappa}$ $z(t) = z_0 + \dfrac{v_{z0}\sin(\kappa t) \pm b_z(1 - \cos(\kappa t))}{\kappa}$

Table 3.2: Dubins helicopter trim primitives in depth image (u, v, Z) coordinates.

	(u, v, Z)
Straight Line $(u_\theta \equiv 0)$	$u(t) = f\dfrac{x_0 + v_{x0}t}{y_0 + v_{y0}t}$ $v(t) = f\dfrac{z_0 + v_{z0}t}{y_0 + v_{y0}t}$ $Z(t) = y_0 + v_{y0}t$
Circular Turn $(u_\theta \equiv \pm\kappa)$	$u(t) = f\dfrac{\kappa x_0 + v_{x0}\sin(\kappa t) \pm b_x(1 - \cos(\kappa t))}{\kappa y_0 + v_{y0}\sin(\kappa t) \pm b_y(1 - \cos(\kappa t))}$ $v(t) = f\dfrac{\kappa z_0 + v_{z0}\sin(\kappa t) \pm b_z(1 - \cos(\kappa t))}{\kappa y_0 + v_{y0}\sin(\kappa t) \pm b_y(1 - \cos(\kappa t))}$ $Z(t) = y_0 + \dfrac{v_{y0}\sin(\kappa t) \pm b_y(1 - \cos(\kappa t))}{\kappa}$

The primitives are linked to form trajectories by enforcing smooth transitions, which determine the initial state of each primitive, and solving for durations. Instead of attempting to connect points with straight lines as in a kinematic motion planning algorithm, dynamically feasible paths can be generated from primitives and are used to explore the space. If this is done according to conditions of Theorems 1 and 2 (that is, the primitives allow for control saturation), this substitution allows for resolution-complete motion planning that simultaneously satisfies the vehicle's dynamics and control constraints. In Figure 3.2, we adapt the lazy egospace PRM of the previous section to use Dubins helicopter paths, which can be found algebraically in advance in terms of the destination in egospace as well as the initial vehicle state and simply concatenated smoothly online. The egospace representation of the motion primitives can also be substituted into the reactive algorithm described above if high-speed obstacle avoidance, rather than a complete motion planning algorithm, is adequate for the needs of a mission.

Configuration Flat Trajectory Segments

The motion primitive concept is closely related to standard flatness-based trajectory generation in that control inputs are immediately available for the planned trajectory in C-space. We now specialize to a configuration flat plant and follow the method of [68], in which space curves of a vehicle are restricted to a polynomial of sufficiently high degree that can be smoothly linked — thus replacing the trim primitives described above. Given an initial configuration $\mathbf{q}(0) = \mathbf{q}_0$, final time

t_f, and final configuration $\mathbf{q}(t_f) = \mathbf{q}_f$, we may find a polynomial trajectory of degree n by enforcing the initial and final states and solving a linear system for the coefficients. Assuming a one-dimensional particle with specified position and velocity at the start and end of a maneuver, for example, we construct the polynomial $q(t) = a_3t^3 + a_2t^2 + a_1t + a_0$ and specify its values at the endpoints:

$$q(0) = q_0 \quad \dot{q}(0) = u_0,$$

$$q(t_f) = q_f \quad \dot{q}(t_f) = u_f.$$

Immediately we have $a_0 = q_0$ and $a_1 = u_0$, and substitute $q(t)$ into the remaining conditions to obtain a 2×2 linear system for a_2 and a_3:

$$\begin{pmatrix} q_f - u_0 t_f - q_0 \\ u_f - u_0 \end{pmatrix} = \begin{pmatrix} t_f^3 & t_f^2 \\ 3t_f^2 & 2t_f \end{pmatrix} \begin{pmatrix} a_3 \\ a_2 \end{pmatrix}.$$

For non-zero t_f, we obtain the closed form solution

$$a_2 = \frac{3(q_f - q_0) - t_f(2u_0 + u_f)}{t_f^2},$$

$$a_3 = \frac{2(q_0 - q_f) + t_f(u_0 + u_f)}{tf^3}.$$

Indeed, for a polynomial of finite degree n, we can determine all coefficients in advance in a similar manner. This property is used in Chapter 4 to extract feasible trajectories with minimal computation performed onboard. We then proceed exactly as in the Dubins helicopter example, but instead of using the time-optimal primitives, use the expression for each coefficent and find a suitable polynomial at each linkage step.

When used to construct space curves $(x(t), y(t), z(t))$ with a specified final time, polynomials of particular degree are also readily shown to minimize net derivatives of position between fully determined endpoints.

Proposition 2. *Let $t \in [0, t_f]$, $t_f > 0$, and $M, N, k \in \mathbb{N}^0$ such that $M + N = 2k$. The space curve $\mathbf{q}(t)$ in $\mathbb{R}^3$ that minimizes the integral*

$$J = \int_0^{t_f} \frac{1}{2} \frac{d^k \mathbf{q}(t)}{dt^k}^T \frac{d^k \mathbf{q}(t)}{dt^k} dt,$$

subject to the boundary conditions

$$\left.\frac{d^m \mathbf{q}(t)}{dt^m}\right|_{t=0} = \mathbf{q}_0^{(m)}, \qquad 0 < m < M, \quad \mathbf{q}_0^{(m)} \in \mathbb{R}^3, \tag{3.10a}$$

$$\left.\frac{d^n \mathbf{q}(t)}{dt^n}\right|_{t=t_f} = \mathbf{q}_f^{(n)}, \qquad 0 < n < N, \quad \mathbf{q}_0^{(n)} \in \mathbb{R}^3, \tag{3.10b}$$

is a polynomial of degree $2k-1$.

Proof. Let $\mathcal{L} = \frac{1}{2}\frac{d^k \mathbf{q}(t)}{dt^k}^T \frac{d^k \mathbf{q}(t)}{dt^k}$. Using a generalization of the Euler-Lagrange equations to higher derivatives [69]:

$$\frac{d^k}{dt^k}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{q}^{(k)}}\right) = \frac{d^k}{dt^k}\left(\mathbf{q}^{(k)}\right) = \mathbf{q}^{(2k)} = 0,$$

which can be repeatedly integrated using the boundary conditions (3.10) to generate a polynomial solution of degree $2k-1$. □

Accordingly, the degree-three example not only provides a feasible path from start to finish, but also minimizes net acceleration through the single tunable parameter t_f that determines the value of the optimization objective J from Proposition 2.

Summary

In this chapter, we have characterized the nature of motion planning in egospace using configuration-flat dynamics and constructed examples of its use. We established that literature motion planning algorithms, which are typically run in (x, y, z) coordinates, extend readily to egospace and can be used together with configuration-flat dynamics. We provided a number of useful approximations to the full motion planning problem, which were evaluated for applicability for use in later experimental implementations (see Chapter 4).

Chapter 4

EXPERIMENTAL IMPLEMENTATION AND RESULTS

Portions of this chapter have been modified from [27] and [5].

In this chapter we introduce a quadcopter experimental testbed and evaluate the performance of two approaches to the egospace obstacle avoidance problem. The first evaluates the safety of reactive velocity selection under the infinite agility approximation of Chapter 3, using an egocylinder with a fusion of forward-looking stereo data and side-looking optical flow. The second approach generalizes to deliberative planning on full dynamics, and replaces the reactive turning controller of the first approach with receding horizon, feasible trajectories that can be collision-checked in advance. Depth data from a stereo pair is reinforced using GMM temporal fusion and propagated away from the sensor field of view to provide complete coverage of several complex and challenging environments.

4.1 Introduction: Quadcopters

The term "drone" has no precise technical meaning and is rarely used in the engineering literature. Within a colloquial context, however, the unqualified term refers most often to the class of *multirotor* vehicles, particularly the four rotor *quadcopter* (also, *quadrotor*) configuration. Quadcopters have come to dominate the consumer model aircraft sector due to their simplicity, inexpensiveness, and ease of use.

Quadcopters and multirotors are also an ideal testbed for research in aerial robotics. Their small size, ability to hover in place, and vertical takeoff and landing (VTOL) qualities allow for easy and safe operation within indoor and outdoor laboratory setting, especially when interaction with the environment (including grasping tasks, as in [64], and ball juggling, as in [70]) and flight near obstacles [71] is expected. On a deeper engineering level, these capabilities also instill the platform with a natural aptitude for high-level autonomy, including single- and multiple-agent construction tasks [72], complex formation flight [73], and mapping operations [13].

Principles of Operation

Here, we focus on the quadcopter platform, following loosely the system characterization of [25]. More general multirotors are based on the same physical principles,

but with a more complex actuator mapping due to either the redundancy of having more than four rotors or the additional actuation mechanism required to fly with three.

Quadcopters move through space using four electric motors, arranged in an "X", that each spin a propeller (or *rotor*) on rotation axes parallel to the body z axis (Figure 4.1). We assume for simplicity that the arms of the X are oriented at ninety degrees to each other and aligned with the axes of the vehicle body frame. The rotors are modelled by identical and constant vertical force coefficients k_F such that the vertical force (thrust) contribution F_i from the ith rotor is given by

$$F_i = k_F \omega_i^2, \tag{4.1}$$

where ω_i is the angular velocity of rotor i. The drag on rotor i also generates a reaction moment about its rotation axis

$$M_i = k_M \omega_i^2. \tag{4.2}$$

The thrust and moment coefficients k_F and k_M are found empircally through system identification. The assumption that k_F and k_M are constant also tacitly assumes a constant air density ρ, which is acceptable for the small altitude changes of a typical quadcopter operation but can be relaxed by using the lift and drag coefficients k_L and k_D instead (as in the aerodynamics of helicopter main rotors; see [74]). For a relevant surface area S and rotor radius R,

$$k_F = \rho R^2 S k_L, \tag{4.3a}$$

$$k_M = \rho R^3 S k_D. \tag{4.3b}$$

The motors spin up the propellers to speed according to the linear model

$$\dot{\omega}_i = k_m(\omega_i^{des} - \omega_i), \tag{4.4}$$

where ω_i^{des} is a desired feedforward motor speed and k_m is an empirical motor gain. For a typical quadcopter, however, the motor response is much faster than the dynamics of the vehicle and we can instead safely assume $\omega_i = \omega_i^{des}$ at all times.

The four motors rotors together produce a collective thrust

$$u_{\text{collective}} = \sum_{i=1}^{4} F_i \tag{4.5}$$

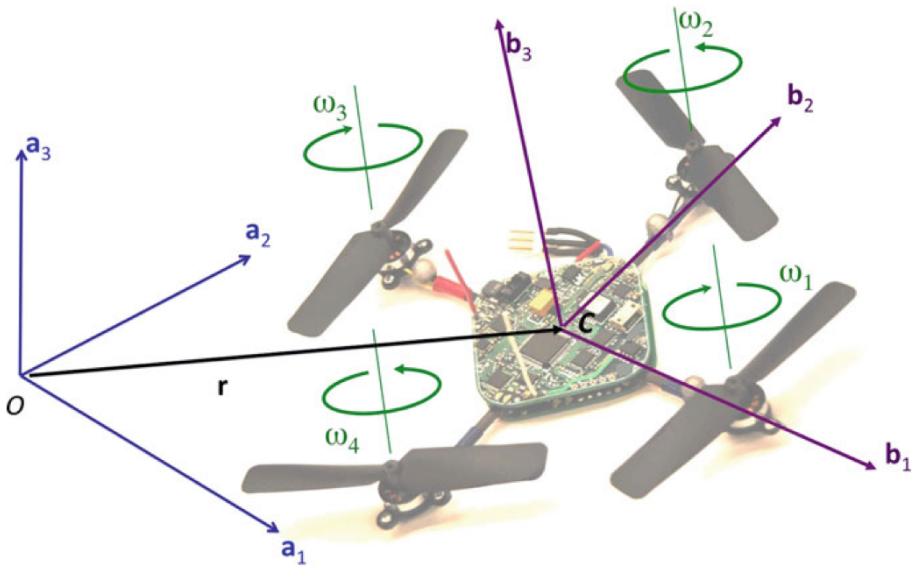

Figure 4.1: The four motors of a quadcopter have axes of rotation aligned with the body z axis — here, denoted $\mathbf{b}_3$ — with the x ($\mathbf{b}_1$) and y ($\mathbf{b}_2$) axes aligned with the arms of the vehicle. The world coordinate axes ($\mathbf{a}_i$) follow the east north up (ENU) flat earth convention. Image from [25].

that supports the weight of the vehicle against gravity and provides translational acceleration, and a net moment about the vehicle center of mass aligned with the rotation axes

$$M_{com} = \sum_{i=1}^{4} (-1)^{(i+1)} M_i. \tag{4.6}$$

Here, we have assumed that motors 1 and 3 are opposite from one other and spin clockwise (thus inducing a positive, counterclockwise reaction torque about the vehicle z axis) and motors 2 and 4 spin counterclockwise in order to cancel out the reaction moments and prevent the vehicle from spinning about the z axis during a hover.

The lifting surfaces of the rotors are usually cambered and have some degree of spanwise twist, and are manufactured in distinct "right-handed" and "left-handed" versions that are not interchangeable between motor spin directions. As a result, reverse thrust is unavailable on off-the-shelf platforms — a research exception is the fixed-speed, variable-pitch quadcopter of [75], which can actuate the rotors to a negative angle-of-attack configuration and maintain a stable inverted hover.

The quadcopter maneuvers by separately selecting ω_i to vector the collective thrust and change its magnitude (Figure 4.2). Assuming that a balanced vehicle starts in

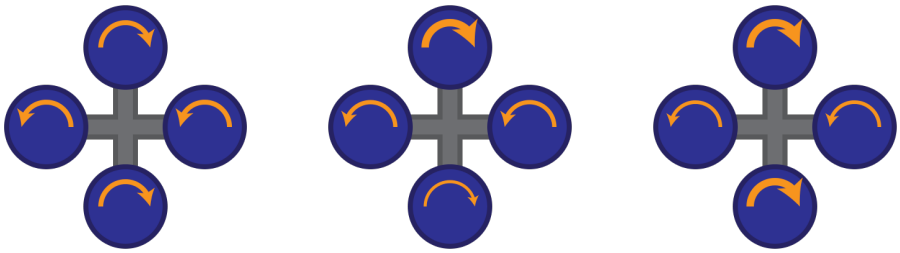

Figure 4.2: A quadcopter modulates the speeds of four spinning rotors to change its position in space. By increasing or decreasing the speed of the motors at once (left), the quadcopter changes the magnitude of its collective thrust and can ascend and descend. The collective thrust can be vectored by pitched by pitching or rolling the vehicle (center), which is accomplished by increasing the speed of one motor and decreasing the speed of its opposite. The vehicle can yaw about its center of mass (right) by increasing the speed of the clockwise or counterclockwise motors while decreasing the speed of the opposite motors.

a hover with equal motor speeds ω, a simultaneous and equal increase or decrease in ω_i will cause $u_{\text{collective}}$ to increase or decrease without a change in its direction of application. Pure rolling and pitching moments are generated by maintaining a constant collective thrust but increasing the speed of one motor while decreasing the speed of the opposite motor — for two opposite motors placed at a radius r from the vehicle center of mass, this command produces a net torque $\Delta T r$ for the prescribed thrust differential ΔT. Pure yawing moments are generated, while again maintaining a constant collective thrust, by increasing the speed of the clockwise or counterclockwise motors and decreasing the speeds of their counterparts. If motors 1 and 3 (and thus, 2 and 4 by elimination) had not been opposite from each other, this procedure would also have produced rolling and pitching moments. It is also clear that a pure yawing moment will not result in translation of the vehicle because it changes neither the magnitude nor the direction of the collective thrust. This ability to yaw the vehicle separately from the x, y, z dynamics causes the yaw angle ψ to be a flat output of the system.

Attitude Dynamics

A fully general derivation of the dynamics of a quadcopter on SO(3) is given in [25], which neglects motor dynamics but includes the capability for aerobatic flight at extreme orientations. For the purposes of outdoor unprepared obstacle avoidance, however, these maneuvers are largely beyond the capability of the current state-of-

the-art in onboard visual state estimation. A simpler linear attitude mode is derived in [76], used in [17] as the basis of a numerical trajectory generator, and described as linear approximation to the full geometric controller in [25]. The linear model has been used extensively, is the default autopilot mode for the off-the-shelf experimental aircraft described later in this chapter, and provides adequate performance for outdoor obstacle avoidance missions.

We represent the attitude of the vehicle — that is, the transformation of a body frame vector $\mathbf{v}^B$ into a world frame vector $\mathbf{v}^W = R_B^W \mathbf{v}^B$ — using sequential Z-X-Y Euler angles (where we have modified the rotation sequence and body axis convention in [76] to match that of [25]). A particular orientation can be described, beginning with aligned body and world frames, with a rotation first by a yaw angle ψ about the body z axis, then a roll angle ϕ about the (now different) intermediate body x axis, followed by a pitch angle θ about the (once again different) intermediate body y axis. In our coordinate system, which is common in the MAV literature but differs from the standard aeronautics convention, thrust is directed in the positive z body direction and position and orientation are referenced against an inertial, flat earth, east north up (ENU) world frame.

We also assume that the roll and pitch angles are small, which is acceptable for level, non-aerobatic flight. In this limit the Euler angles commute, are orthogonal, and can be modeled as a set of independent, linear, second-order equations of motion:

$$\ddot{\psi} = u_{\text{yaw}}, \tag{4.7a}$$

$$\ddot{\theta} = u_{\text{pitch}}, \tag{4.7b}$$

$$\ddot{\phi} = u_{\text{roll}}. \tag{4.7c}$$

Here, u_{yaw}, u_{pitch}, and u_{roll} are prescribed angular accelerations that align with the principal axes of inertia and can be related to the thrust and moment differentials, using a diagonal inertia tensor, as discussed above.

We note that all of the experimental results that follow can be just as well implemented with either the linear attitude model or the fully general geometric model, and proceed with the linear model for clarity and ease of implementation. The controllers differ only in their tracking performance and are essentially equivalent for

the purposes of general motion planning — both the model discussed here and the more complex geometric one in [25] are differentially flat in position and yaw angle, and up to constraints have the same smoothness criteria for trajectory following.

Position Dynamics and Differential Flatness

To derive the position dynamics of the quadcopter, we rotate $(0, 0, u_{\text{collective}}/m)^T$ into the world frame using the conventions of the previous section and add the effect of gravity:

$$\ddot{x} = (\sin\theta\cos\psi + \cos\theta\sin\phi\sin\psi)\,\frac{u_{\text{collective}}}{m}, \tag{4.8a}$$

$$\ddot{y} = (\sin\theta\sin\psi - \cos\theta\sin\phi\cos\psi)\,\frac{u_{\text{collective}}}{m}, \tag{4.8b}$$

$$\ddot{z} = -g + (\cos\phi\cos\theta)\,\frac{u_{\text{collective}}}{m}. \tag{4.8c}$$

Here, m is the mass of the vehicle, g is the acceleration due to gravity, and the Euler angles evolve according to the plant model given in Equations 4.7. Given a trajectory $(x(t), y(t), z(t), \psi(t))$ and its derivatives, we can use the combined state equations to algebraically extract the required roll and pitch angle trajectories $\phi(t), \theta(t)$ as well all the control inputs: $u_{\text{collective}}$, u_{yaw}, u_{roll}, u_{pitch}.

Proposition 3. *The plant model given by Equations 4.7 and 4.8 is differentially flat in* (x, y, z, ψ)

Proof. We prove flatness by explict construction of expressions for $\phi(t)$, $\theta(t)$, $u_{\text{collective}}(t)$, $u_{\text{yaw}}(t)$, $u_{\text{pitch}}(t)$, and $u_{\text{roll}}(t)$ in terms of $x(t)$, $y(t)$, $z(t)$, and $\psi(t)$ and their derivatives.

Immediately, $u_{\text{yaw}} = \ddot{\psi}$ and $u_{\text{collective}} = \sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2}$. Because reverse thrust is unavailable on our quadcopter and a freefalling vehicle lacks control authority in x and y, we select the positive square root and take $u_{\text{collective}} > 0$.

We then rotate $\ddot{x}$ and $\ddot{y}$ into the yawed frame (that is, the intermediate body frame after the first Euler angle rotation) and solve for pitch and roll. This rotation operation gives the pitch relation

$$\sin\theta = \frac{m}{u_{\text{collective}}}(\ddot{x}\cos\psi + \ddot{y}\sin\psi), \tag{4.9}$$

and the roll relation

$$\sin\phi = -\frac{m}{u_{\text{collective}}\cos\theta}(-\ddot{x}\sin\psi + \ddot{y}\cos\psi), \tag{4.10}$$

where we have taken $\cos\theta > 0$ to avoid a gimbal lock condition.

The control inputs u_{roll} and u_{pitch} follow from 4.9 and 4.10 after differentiating twice. □

We note that the independence of vehicle translation from the yaw angle trajectory arises from the calculation of roll and pitch angles in terms of rotated frame accelerations: $\ddot{x}' = \ddot{x}\cos\psi + \ddot{y}\sin\psi$ and $\ddot{y}' = -\ddot{x}\sin\psi + \ddot{y}\cos\psi$. Accordingly, we may set the desired yaw angle to any convenient value and extract roll and pitch angles that will produce an invariant world frame acceleration. This property allows a point abstraction of the quadcopter to be treated as a configuration flat vehicle in the C-space variables x, y, z, leaving the yaw angle trajectory (which has a trivial configuration space) to high-level considerations. Possible options include a fixed yaw angle, which is mathematically the most convenient, or a "coordinated" yaw angle that keeps the rotors aligned with the planar Frenet-Serret frame, which is more useful for obstacle avoidance and control. The latter choice allows a forward-looking range sensor to be aligned with the velocity vector and provides better tracking performance by allowing the full rolling torque contribution of a motor pair to be directed into a turn.

We also observe that expressions for u_{roll} and u_{pitch} involve the fourth derivatives of position — pitch and roll depend on $\ddot{x}$ and $\ddot{y}$ explicitly and $\ddot{z}$ through $u_{\text{collective}}$, and are differentiated twice to determine their corresponding feedforward inputs. Accordingly, feasible trajectories must be at least four times differentiable, not necessarily continuously, to be tracked exactly by the attitude controller. In practice, however, the position dynamics of the quadcopter are much slower than the attitude dynamics, and trajectory tracking can be split into a slow position controller and a fast attitude controller with somewhat relaxed smoothness criteria. The position controller uses the flatness relations to request attitude and thrust values, which are then maintained by the attitude controller at a rate at least an order of magnitude faster.

4.2 Reactive Motion Planning using an Egocylinder: Infinite Agility Approximation

In this section, we experimentally demonstrate the reactive, infinite agility egospace planner discussed in the previous chapter, and assess its performance using an intervention metric.

System Architecture

Figure 4.3 illustrates the sensing, processing, and algorithm architecture of our approach, as implemented on an AscTec Pelican quadcopter (Figure 4.7). The architecture is divided between two onboard embedded computers and a closed-source autopilot from AscTec — state estimation, motion planning, and position/velocity control are performed on an ARM-based Odroid U3 [77], visual obstacle detection and representation are performed on an Intel-based Asctec Mastermind [78], and low-level linearized motor control is performed on a proprietary Asctec autopilot [79]. With the exception of high-level triggering of the obstacle avoidance pipeline, which is relayed to the vehicle over a 5 GHz WiFi connection, all sensing and computing is performed aboard the vehicle and navigation from the start to goal is accomplished without human input of any kind.

The various modules of the pipeline are written in C/C++ and coordinated using the Robot Operating System [80]. The vision and motion planning modules are linked over a local Ethernet network between the Mastermind and the Odroid, which serves as a master computer within the ROS framework. After a motion plan is extracted, a control node calculates an instantaneous desired acceleration command, which is in turn converted to a proprietary serial packet format using an Asctec control ROS wrapper and sent to the autopilot over a serial UART link.

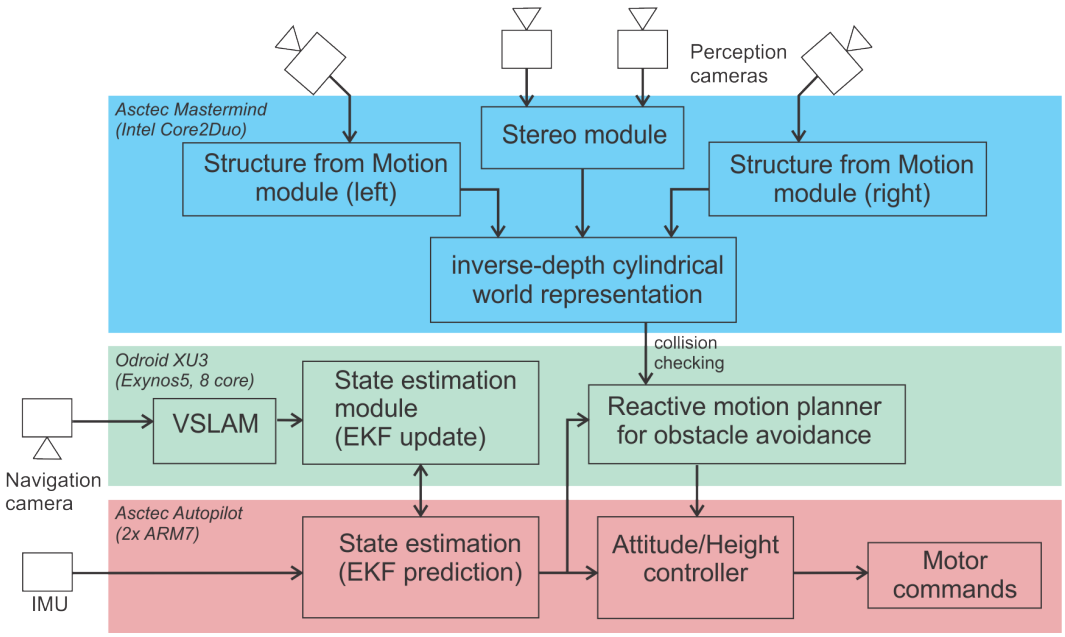

Figure 4.3: System architecture.

To minimize the sensor hardware while still acheiving a large field of regard, we augment a forward-looking stereo pair with monocular cameras aimed 45° to each side for approximately 180° of coverage. The shutter of each camera is triggered to fire simultaneously with that of the left stereo camera, which serves as a master. Stereo depth data is extracted using local block matching for speed, which produces dense disparity maps in our test environments. The obstacle detection sensing suite is linked to the Mastermind using USB, which provides camera control, receives the raw image data, and performs all vision and representation processing.

To obtain depth perception with the side-looking cameras, we examined several two-frame optical flow algorithms, as well as the LSD-SLAM incremental SfM algorithm [81]. LSD-SLAM constrains optical flow search to epipolar lines computed from estimated camera motion, and incrementally improves depth resolution by using each new image to update depth estimates in keyframes. After evaluating several alternatives, LSD-SLAM proved to be much less noisy than unconstrained optical flow algorithms and was selected for inclusion in the sensing suite. Because monocular SfM has an unobservable scale factor (as discussed in Chapter 1), we estimate scale by comparing SfM range measurements with range from stereo in the image overlap areas between the outer SfM cameras and the stereo cameras.

To provide a unified depth representation, we project and fuse the stereo and scaled SfM depth maps onto a cylinder (the "egocylinder") centered on the aircraft IMU, fixed to the body frame, and aligned with the body z axis (as in the configuration of Chapter 4). The vertical pixel coordinate retains the projective character of a standard depth map with focal length f, while the horizontal pixel coordinate is based on a uniform discretization of the interval $[0, 2\pi)$ to admit a representation with a full 360° field of regard. We use inverse range ($1/r$ — the "radial disparity") as a generalized depth coordinate, rather than range itself, because it assigns a finite value (zero) to objects beyond the maximum range of the sensors, and discretized inverse range matches the uncertainty characteristics of vision-based depth estimation. The egocylinder is implemented internally as an array data structure in which the nearest object in the direction of the pixel is recorded (Figure 4.4).

Once the raw depth data is projected onto the cylinder, a C-space expansion is performed on the egocylinder using a rectangular method similar to that of [17] and discussed in Chapter 2. This step essentially reduces the range at each pixel, and widens objects in the depth map by a characteristic radius of the aircraft plus a safety margin. A precomputed look-up table of expansion radii, in pixels, is used

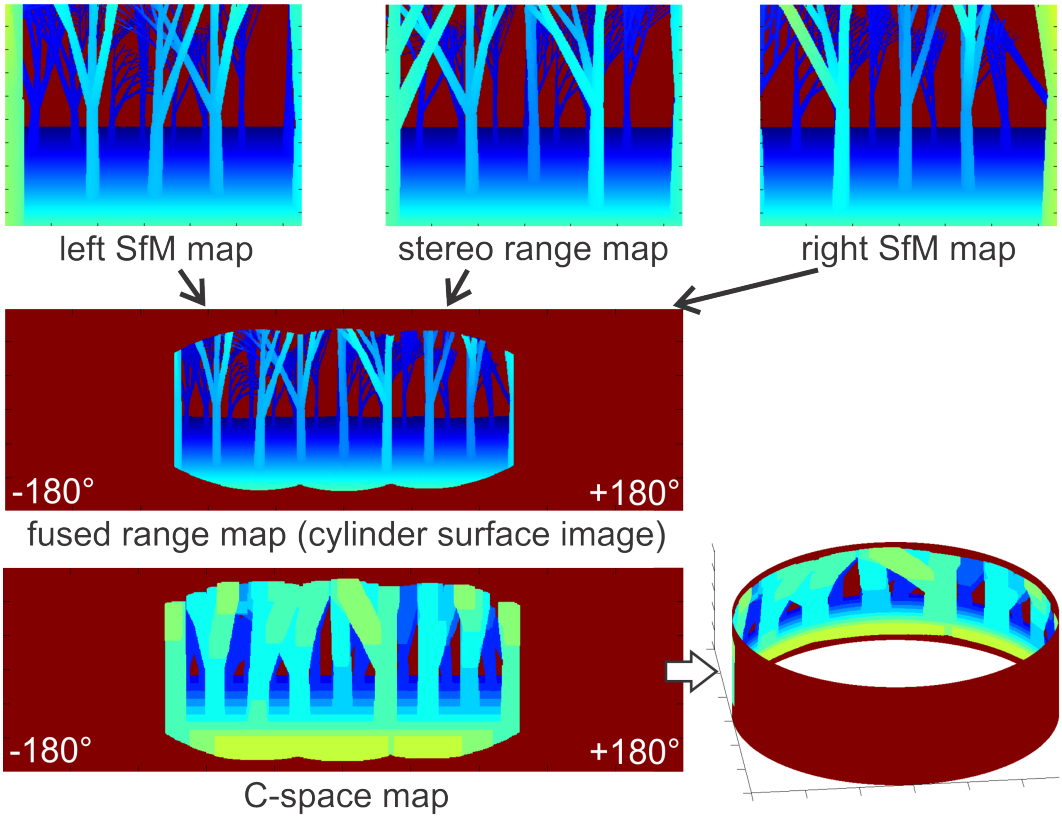

Figure 4.4: Schematic illustration of stereo and SfM depth maps fusion into the egocylinder representation, and C-space expansion of the egocylinder. Using inverse range, the expansion widens closer objects more than farther objects.

to keep onboard computation during the expansion to a minumum. The expanded egocylinder then allows the aircraft to be treated as a point for collision checking, and is sent to an obstacle avoidance module running on the Odroid U3. Images are processed at 384 × 240 pixel resolution, with stereo running at 5 fps, LSD-SLAM at 10 fps, and the egocylinder updated at the stereo frame rate of 5 fps.

For the rest of this section, we evaluate the innovations in the perception and representation system with a simple, fast avoidance algorithm that is safe if there are no major perceptual errors. At the obstacle densities and velocities considered here, we employ a reduced dynamical model in which the vehicle can turn with infinite agility but requires a finite distance to come to a stop. Accordingly, we restrict the set of possible vehicle trajectories at any instant to the set of straight lines extending radially from the vehicle — that is, the radial paths discussed at the end of Chapter 2 and related to the infinite agility approximation of Chapter 3.

Collision-free radial paths are extracted from the egocylinder by first transforming

a desired vehicle cruise speed into an inverse range safety horizon. In order to always maintain collision-free operation, the vehicle must look far enough ahead that it could still safely come to a stop if an obstacle were detected in the proposed flight path direction during the subsequent planning cycle. For the current velocity, this safety horizon is determined by the time required to stop $t_{\text{stop}}^{(V)}$, the time until new range data appears t_{plan} (for our implementation, equal to the 0.2 s egocylinder update time), and a small margin of error dt — these time values are estimated, added, and used to estimate the lookahead distance R (and its equivalent inverse range value $d_{\text{look}} = 1/R$). For a vehicle cruise speed V,

$$d_{\text{look}} = \frac{1}{V(t_{\text{stop}}^{(V)} + t_{\text{plan}} + dt)}. \tag{4.11}$$

Paths are selected by comparing the lookahead horizon d_{look} to each pixel of the egocylinder. After first checking the direction to the goal in order to avoid a search if possible, the planning horizon is checked against the pixels of the egocylinder to cull flight directions that violate the safety horizon constraint and would result in an unacceptably short time-to-contact with an obstacle. Altitude restrictions are implemented by directly restricting the pixel search region, in egospace, to prevent the vehicle from selecting an excessively high or low target target. Of the remaining pixels, all of which are collision-free, the motion planner returns the one that is closest to the goal direction for presentation to the control module (Figure 4.5).

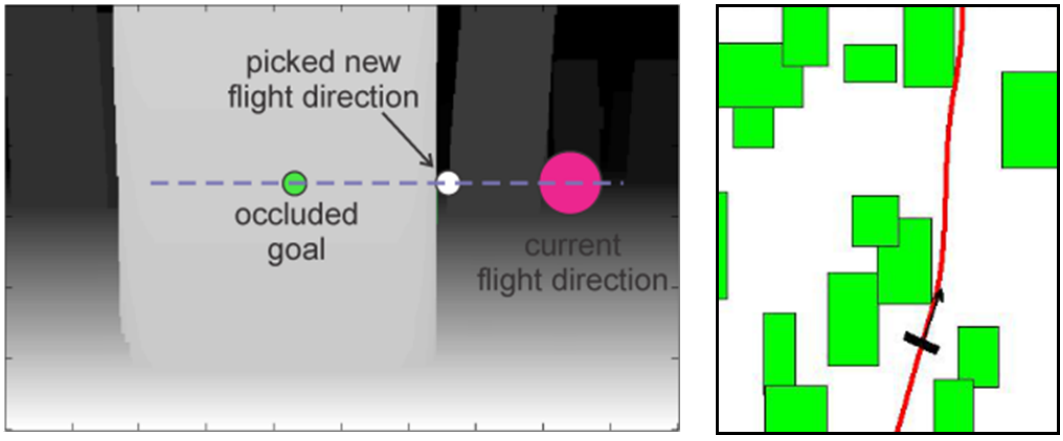

Figure 4.5: Motion planning schematic and simulation. Left: selected flight direction to avoid obstacle. Right: simulated flights through cluttered environments without dead-ends were successful up to speeds over 15 m/s (top view).

To maximize safety and visibility of the scene ahead, a low-level controller executes the command by yawing the cameras towards the requested direction while separate

PID loops extract acceleration commands that maintain forward velocity and eliminate side slip around the turn. The acceleration commands are converted to a set of Euler angles and a collective thrust using differential flatness, which are sent to the Asctec autopilot over the serial link and in turn converted to motor commands using its proprietary firmware.

We have also implemented a simple temporal filtering feature over the extracted motion plan that provides robustness against noisy or missing depth data. Once a point on the planning horizon is chosen, it is propagated forward with the motion of vehicle for a few cycles and assigned a priority lower than the goal direction but higher than that of the egospace pixel scan — that is, the motion planner considers the destination, the reused motion plan, and the output of a scan in that order and exits the search once a collision-free path is found. In addition to reducing latency by allowing the planning pipeline to be bypassed most of the time, target filtering tends to smooth trajectories in complex environments where the target would otherwise change frequently and prevents dropped frames or other gaps in visual input from disrupting the planning process. This entire planning approach is very fast, safe, and allows us to focus on evaluating perception at the cost of sacrificing algorithmic completeness and strict satisfaction of the full vehicle dynamics. Planning takes under a millisecond to verify that the current direction is still safe, and a few milliseconds if it is necessary to search for a new direction — therefore, it occurs at the egocylinder update rate of $5Hz$. The approach is generalized later in this chapter to a more sophisticated image-space motion planning algorithm that can accommodate vehicle dynamics.

Visual-inertial state estimation is performed using a nadir-pointed camera using methods from [82]. Operating outdoors in areas with bright sunlight and deep shadow is particularly difficult, because it creates very large intra-scene (within the same image) and inter-scene (between successive images) illumination variations that greatly exceed the linear dynamic range of available cameras. This has been especially problematic in the experiments we have conducted in a grove of trees using the nadir-pointed camera.

The most straightforward way to address these issues is to improve dynamic range at the sensor level. Some approaches acquire multiple images separated in time and combine these in software, which is impractical for our low-altitude flight envelope. Another approach uses hardware design within the imager to provide a "multilinear" exposure mode that approximates a logarithmic image response. This mode

is implemented in hardware within the onboard Matrix Vision mvBlueFOX-200w CMOS cameras and can extend the total dynamic range from 55 dB to 110 dB. These cameras have three linear segments in their photometric response function, where the slope and transition point of the second and third segments is controlled by a two sets of knee point parameters. Creating a good exposure for given scene conditions requires choosing the total exposure time and setting appropriate knee point parameters.

Figure 4.6: Non-HDR (left) and HDR (right) images in a forest scene. Large areas are saturated or under-exposed in the non-HDR image. The HDR image has a better distribution of intensity values, which leads to better performance of vision algorithms.

We have taken a first step toward exploiting this multi-linear HDR mode using a camera initialization procedure which is run once at the start of an experiment (Figure 4.6). First, we acquire a series of images while adjusting exposure time via gradient descent to push the average intensity of the image stream towards a target intensity in the middle of the pixel brightness range. Next, we fix the total exposure time while seeking the parameters of each knee point that maximize image entropy. In an iterative process, each knee point is set sequentially to maximize local entropy using a standard coordinate descent optimization. This does not simultaneously optimize the setting of both knee points, but it avoids extra parameters and has shown to improve feature tracking performance. Once the exposure parameters are initialized, they are fixed for the duration of the flight, which has been adequate in our test conditions to date. Ideally, exposure should be optimized on every frame; however, our current optimization procedure is too slow for that and large changes of exposure have potential to require changes to feature tracking algorithms to maintain landmark tracking across exposure discontinuities.

Results

We have conducted low-speed ($< 1m/s$) experimental trials in a grove of trees that provided a relatively high obstacle frequency (Figure 4.7). Flights totaled over 500 meters in aggregate length, during which 65 trees were encountered. This area had very difficult illumination conditions due to the combination of brightly sunlit and deeply shadowed areas in the same image. Figure 4.8 shows results of the vision pipeline at several points during such a run, as well as a 2D map generated offline from logged data.

Figure 4.7: Top: grove of trees test area. Bottom: AscTec Pelican with 4-camera obstacle detection unit and downward-looking state estimation camera (orange box).

The saturated and underexposed areas of the images in Figure 4.6 illustrate the dynamic range problem with these illumination conditions. While the C-space expansion effectively fills in many areas that have missing data in depth maps from stereo and SfM, this forest environment was particularly challenging for the visual-inertial state estimation system. Therefore, we focused HDR experiments on the state estimation camera, where use of the HDR mode improved the average percentage of map landmarks that could be matched in each frame from 61% to 79%. Nevertheless, the floor of the forest had many very small, self-similar features, and

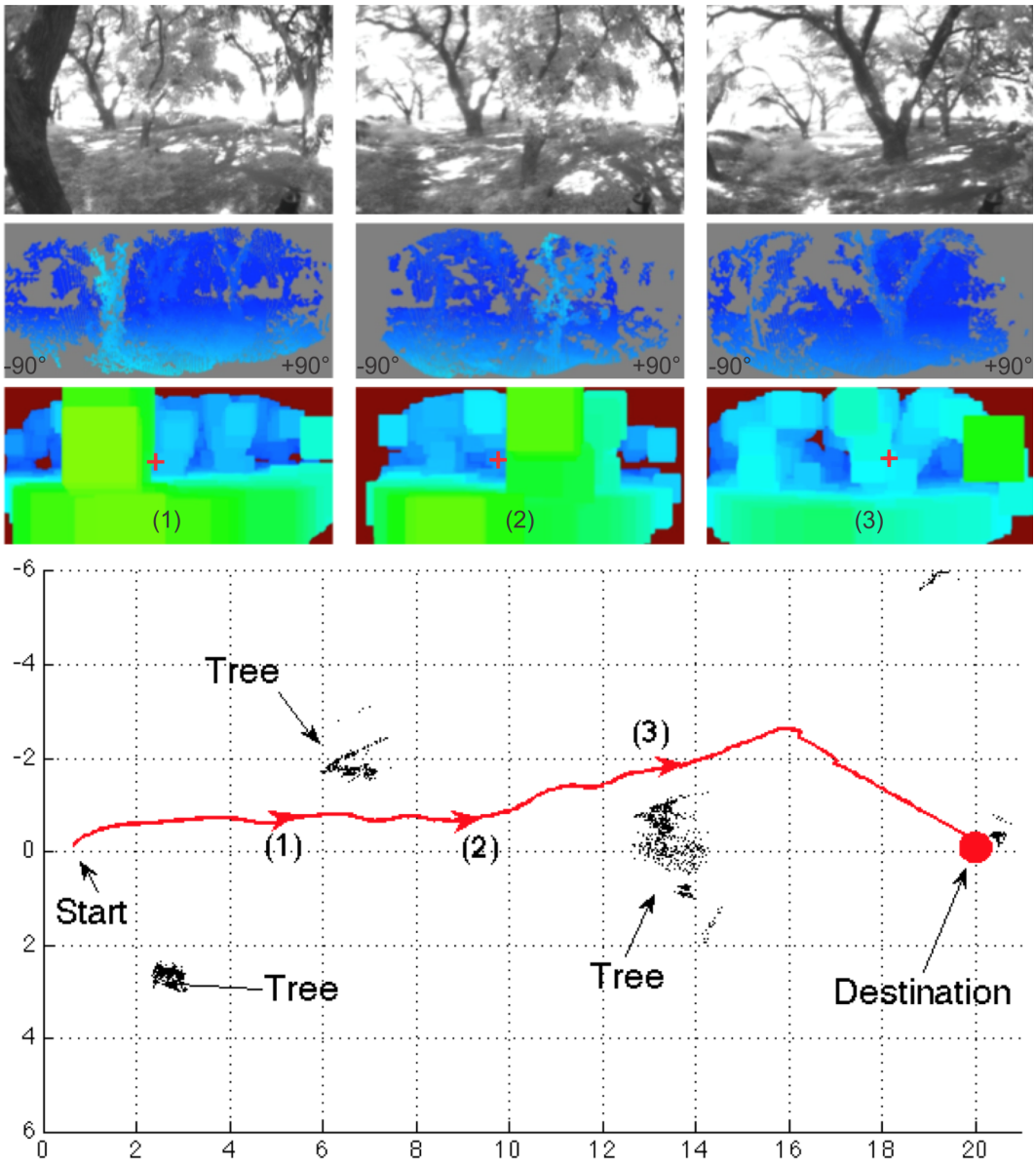

Figure 4.8: Results of a 20m experimental flight through a grove of trees. Top: the results of the perception system for three different locations on the run, showing the left rectified stereo image, the fused egocylinder, and the C-space expanded egocylinder with selected direction of flight (red crosses). This only shows the central 180° of the egocylinder. Bottom: a top down 2D plot of the trajectory and nearby obstacle pixels from the egocylinder over the whole run. Arrows and numbers on the trajectory show where the three images above were acquired. Vehicle speed was $1m/s$ throughout.

doing state estimation with a nadir-pointed camera while flying low (< 2 m above the ground) in this environment still made state estimation by far the weakest link

in the system. Because state estimation is beyond the scope of this thesis, for the rest of this section we have not included state estimation failures within our performance evaluation.

The detection and evasion portions of the architecture were very reliable in the performance evaluation experiments, which were analyzed quantitatively by noting the frequency and cause of any human intervention required to avoid a collision. These modules were responsible for only a single intervention event during the 521 meters recorded, which resulted in successful avoidance of 64 out of 65 trees for an success rate of 98%. The intervention was attributed to a missed detection in which the vehicle had drifted to within the minimum detection distance of our stereo unit. Because of the large baseline of the stereo pair, the obstacle had a disparity beyond the limited search range of the block matching algorithm and registered as empty space — the motion planner interpreted this region as a valid direction of flight and selected a path directly into the obstacle. A pilot intervention was then required to avoid a collision. This error mode, which is inevitable for any stereo pair at some minimum distance, is discussed and addressed later in this chapter using a temporal fusion module, which maintains a representation of obstacles even as they drift closer than the minimum detection distance.

LSD-SLAM failed to adequately track features about 25% of the time, after which it would drop out and become unavailable to the egocylinder fusion step until its recovery a few frames later. With data logging, the LSD-SLAM frame rate dropped to about 8 Hz, which is too slow for the algorithm to generate reliable depth data. Although this limitation prevented the side-looking data from being of much use for path planning, it did not severely impact overall obstacle avoidance performance because the control policy of first turning the stereo cameras towards the flight direction and incorporating a small amount of path hysteresis provided a high degree of robustness to missed left or right camera LSD-SLAM depth maps. The presence of an overlapping stereo region, however, allowed scale estimates to be immediately propagated over to the monocular data, which was seamlessly reacquired beginning on the next frame.

Main Experimental Insights

Using C-space expansion of image space depth maps for collision checking and motion planning is a very new approach to obstacle avoidance for MAVs. In our experiments to date, obstacle avoidance has been quite successful — in 521 meters

of flight in challenging conditions, only one obstacle detection or avoidance intervention was needed in 65 encounters with obstacles, and no problems with false alarms in freespace were apparent. This is significant, since the approach so far does not include explicit temporal fusion for false alarm suppression or filling in missing data, unlike approaches based on voxel maps. Nevertheless, work is in progress to add temporal fusion to image space representations to address the finite probability that these problems will eventually occur.

LSD-SLAM was successful as a source of side-looking depth data, but it required a high frame rate ($> 10Hz$) and accurate calibration of camera extrinsics to maintain its usefulness for obstacle detection, both of which were problematic in this implementation. Side-looking stereo cameras might be easier to use, but would lack the potential of exploiting increasing motion baselines to improve depth resolution that exists with recursive approaches to structure from motion. Ultimately, combining both may be a good approach, as is explored in a recent stereo extension of LSD-SLAM [83]. Although our current level of computational resources prevented our approach from reaping the full benefits of a combined monocular and stereo sensor suite, the egocylinder showed promise as an underlying environment for depth data fusion. In particular, the robustness of monocular sensing was significantly enhanced by the propagation of scale information from the stereo regions, and allowed for immediate recovery of range data. The possibility of a higher monocular framerate on future embedded computing systems offers the potential for a much more reliable and effective perception and representation system.

By far the biggest performance problem in this system, however, is with visual state estimation. Using a nadir pointed camera while flying low (< 2 m above ground) in a scene with a very high dynamic range of illumination and many small, self-similar features (leaves) on the forest floor seems to be at the heart of the problem. We plan to address this in several ways in ongoing work, including visual odometry methods with direct components (such as SVO: [84]) and tightly-coupled approaches in which image features are directly incorporated into a vehicle state vector (such as MSCKF: [85]).

The disparity-space reactive planner was able to extend the advantages of the C-space expansion method and egocylinder to the planning regime — potential trajectories are selected and executed in a highly economical fashion by employing the same framework that allows the egocylinder to represent obstacles compactly and efficiently. Overall, this choice of representation demonstrates decreased planning

latency and complexity compared to world-coordinate methods.

There is a close connection between vehicle velocity, uncertainty in the range data, and successful obstacle avoidance. This has not emerged as an issue for the slow speeds of our experiments so far, but for reliable obstacle detection to scale to high speeds, this interplay will require further study. Several approaches may improve the maximum range and range resolution of the system to support higher velocities, including the use of higher resolution imagery and potentially the use of temporal fusion of depth maps for improved range resolution. Scenes involving moving obstacles will require extensions of both the perception and the planning elements of this system.

4.3 Dynamically Feasible Obstacle Avoidance

Egospace obstacle representations have been demonstrated to simplify reactive obstacle avoidance because their underlying geometry reduces any radially-aligned path to a single pixel — these pixels can be, in turn, evaluated and selected against depth data using a lightweight pixel scan. Egospace also admits full configuration flat dynamics, of which the quadcopter and commonly-used car and airplane models are examples, and allow all control inputs and vehicle states to be determined uniquely in terms of the trajectories as they appear, in egospace coordinates, in relation to obstacles.

In this section, we extend reactive egospace trajectory selection for micro air vehicles with negligible dynamics (as in [27], as well as the previous section of this chapter) to incorporate full configuration flat dynamics. We present experimental verification on a quadcopter platform in *a priori* unknown environments, using stereo obstacle detection, temporal filtering of depth data, and an egocylindrical obstacle and motion planning representation.

Related Work

Traditional approaches for MAV motion planning rely on flatness-based trajectory generation (as in [86]) over a three-dimensional voxel model of the environment. [21] populates an instantaneous, uniform resolution occupancy representation of an unstructured environment, after which a minimum-jerk trajectory is identified using a convex segmentation seeded with the result of an A* search over the voxel indicies themselves. A layered short-range and long range planner then calculates a dynamically feasible path to the goal. A modified version [4] of the same motion planning framework employs a more efficient graph search algorithm. For a known indoor

environment, [20] achieves high-speed flight within a volumetric world model by seeding a polynomial motion planner with the output of a preliminary, dynamics-free RRT* search. Although essentially any standard motion-planning method can be used within a voxel representation, uniform grids scale poorly in size in large outdoor environments, and complex access schemes are required to distribute resolution in a more favorable way (as in [39]).

Egospace representations, on the other hand, can encode collision-free radial paths using a single pixel and offer an efficient trajectory search space for dynamics-free, reactive motion planning in unstructured environments [27]. [17] uses a disparity image representation to efficiently collision-check the segments of a closed loop RRT (CL-RRT) motion planner after a projection, but perform trajectory selection and generation in world coordinates using forward integration of a nonlinear plant model at severe computational expense. Although reactive methods in egospace [27] are extremely lightweight, they scale poorly to high speeds because full trajectories are neither known nor collision-checked in advance, and the reactive plans are strictly infeasible — even if an instantaneous collision-free action is identified and can be followed by the vehicle closely enough to be safe, there is no way to practically verify if the future actions that follow will also be safe. Deliberative motion planning performed entirely within egospace representations, rather than simply projected into egospace for collision-checking, is introduced and characterized in [26] for robots with configuration flat dynamics. Completeness properties are established for the egospace equivalents of motion planners in world coordinates, and motion primitives expressed directly in egospace coordinates extend the advantages of world trajectory libraries to also encompass fast collision-checking and trajectory selection.

Depth data can be obtained from active depth sensors or passive stereo matching. Stereo matching approaches work well both indoors and outdoors and have an adjustable depth range that provides computational flexibility for the design of the perception system. Depth estimation is typically performed on each frame independently, which introduces errors and holes due to environmental factors as well as stereo matching failures. As a result, some degree of temporal fusion is required to propagate reliable data over time and generate a complete, reliable, and stable scene representation for later motion planning steps — unfused egospace approaches, such as [27], require additional side-looking cameras to achieve a large field-of-regard, and suffer from flicker and missed-obstacle incidents that result in

collisions. Fusion can be performed during the matching step [84], [87], [88], in image space [47], [89], [90] or on 3D voxel representatio [11], [91].

In [87] and [88], temporal fusion incorporates temporal data in cost functions during estimation as an extension to spatial aggregation. SLAM techniques [84] also constrain temporal consistency in the estimation framework, where online updates are applied in key frames. In [84], a simple Gaussian model represents depth measurements and limits the search range according to the standard deviation of prior hypotheses. Multi-view filtering is another approach that improves depth maps by removing outliers and compensating holes. In [89], the frames within a specified time window (a fixed number of the most recent frames, for example) are mapped to the most recent frame according to the related depth maps. These hypotheses are merged probabilistically via probability density estimation. [47], [90] use Gaussian Mixture Models (GMM) to represent depth observations in a compact manner, with an online update and propagation approach that decreases computational complexity and memory requirements. These methods tend to produce more reliable depth maps than can be obtained using filter-based fusion approaches.

Three-dimensional models [11], [91] are also commonly used to merge multiple depth map observations and provide temporal fusion. In this regime, depth data is mapped to voxels [11] and accumulated in a surface representation [91] that produces highly accurate maps at the expense of memory usage and computational efficiency.

System Overview

The perception, representation, and motion planning modules are implemented within a pipeline architecture (Figure 4.9) aboard an AscTec Pelican quadcopter, with vision and perception tasks performed on an AscTec Mastermind embedded computer and motion planning and state estimation performed on an Odroid XU4 [92]. With the exception of a slightly more advanced Odroid board, the computational architecture is identical to that of the previous section.

After acquiring stereo imagery of a scene using a set of forward-looking cameras, depth data is calculated, temporally filtered, and used to populate an egocylinder structure in which obstacles are artificially expanded in order to abstract the vehicle to a point mass for planning. The expanded egocylinder is then passed to a two-stage motion planner (detailed below), which first performs a pixel scan to identify a candidate flight path, and then attempts to identify a dynamically feasible, collision-

free maneuver onto the candidate path using minimum-jerk paths directly in image space. Once a motion plan is successfully found, a low-level flatness-based controller calculates feedfoward inputs and tracks the reference trajectory until a new motion plan is required. Vehicle pose is provided to both the temporal filtering and motion planning modules using visual odometry from a downward-looking camera (Semi-dense Visual Odometry SVO; [93]) fused with IMU data using the SSF framework [82].

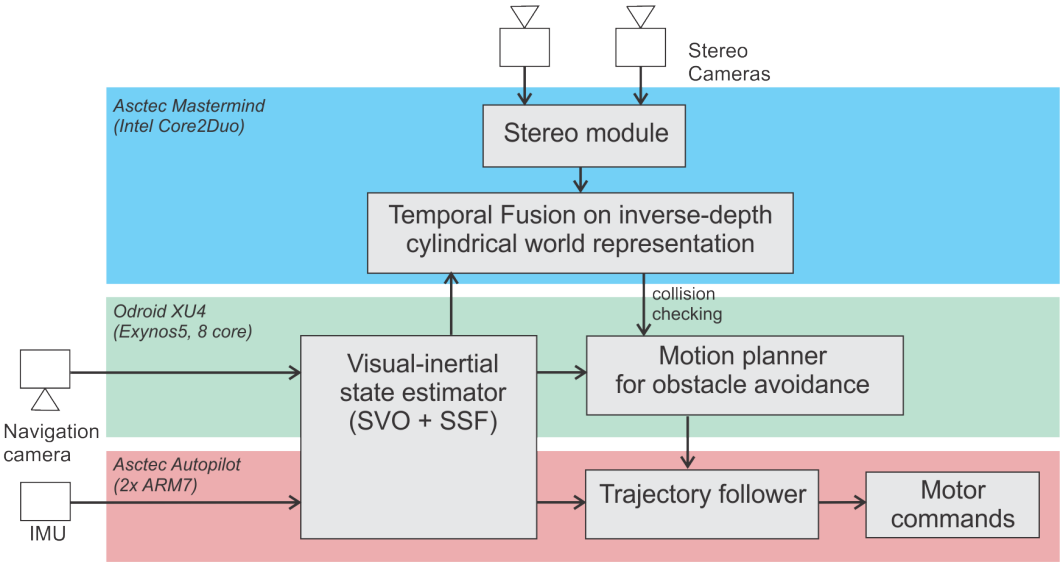

Figure 4.9: System architecture for the implementation on an Asctec Pelican quadrotor that is equipped with an Asctec Mastermind and an Odroid XU4 computing board.

Perception and Representation

The egocylinder representation of the scene [27] is used to represent the environment for collision-checking, which avoids much of the memory and computation expense used by voxel maps to represent free space. A full 360° field-of-regard is achieved by integrating depth maps forward as the vehicle moves within the environment.

Visual perception for collision avoidance is based on the approach shown in Figure 4.10. The most recent stereo vision-based depth map estimate is first mapped onto an egocylinder representation fixed to the frame of the left camera – the optical center of the left camera maps to the center of the structure, with the forward direction aligned with its viewing axis. The internal representation of the egocylinder is a disparity image, specifically inverse range, which wraps around the cylinder

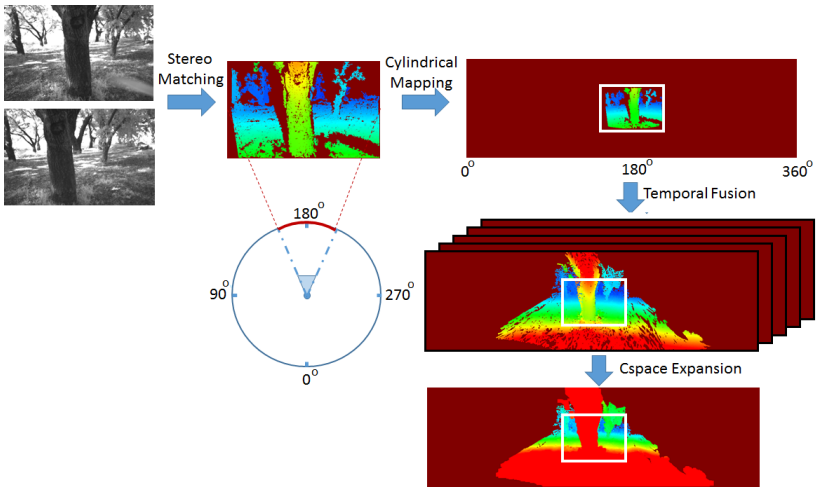

Figure 4.10: The visual perception pipeline uses stereo matching to acquire depth data, which is then mapped onto an egocylinder for temporal fusion and C-space expansion directly in egocylinder coordinates. Areas of the egocylinder visible to the stereo cameras are enclosed within a white rectangle for illustration.

structure and is called the egocylinder image.

Obstacles are then expanded within configuration space (C-space; [17], [27]) by a characteristic radius of the vehicle, directly on the coordinates of the fused egocylinder map, in order to abstract the vehicle to a point mass and simplify path planning. As motion planning proceeds and the vehicle continues to move through the environment, new depth data is observed in the central area of the egocylinder corresponding to the field-of-view of the stereo cameras.

We use GMM-based fusion [47], [90] to efficiently merge the current depth map with past observations directly in image space. In this approach, each pixel is modeled by a mixture of Gaussian distributions, in pixel coordinates (u,v) and disparity d, and propagated to the recent depth map using vehicle pose estimates. The final depth map is obtained by constraining both the standard deviation of the disparity models and an observation count to remove flicker and rely on frequently observed depth values. [47] demonstrates that GMM-based fusion improves depth quality by removing large depth errors such as false obstacles, which are typically due to errors in stereo matching, and by assigning reliable depth values for empty pixels in the recent depth map. Temporal fusion in this region of the egocylinder mitigates sensing failures, such as incorrect depth estimates or empty pixels, by constraining the temporally accumulated data. Regions of the egocylinder not visible to the stereo sensors do not receive new depth data, and are instead updated by transfer-

ring temporally accumulated models subject to a forgetting factor at each update. As the vehicle moves around and encounters obstacles at a variety of viewing angles, a more complete representation of the scene emerges as temporal data is fused into the structure.

Motion Planning

The first stage of the motion planning module replicates the approach of [27] in which collision-free, yet dynamically infeasible radial paths are represented by their pixel locations and identified using a scan over the expanded egocylinder structure. During this scan, the distance to obstacles at each pixel is compared to a predetermined planning horizon distance h, which is chosen to maintain a safe time-to-contact towards obstacles at the desired speed and can take values up to the maximum reliable sensor range depending on the expected complexity of the environment. The collision-free pixel that is closest to that of the destination is selected as a candidate path (Figure 4.11), and if no safe candidate path is found the process is repeated with a lower desired speed and shorter planning horizon.

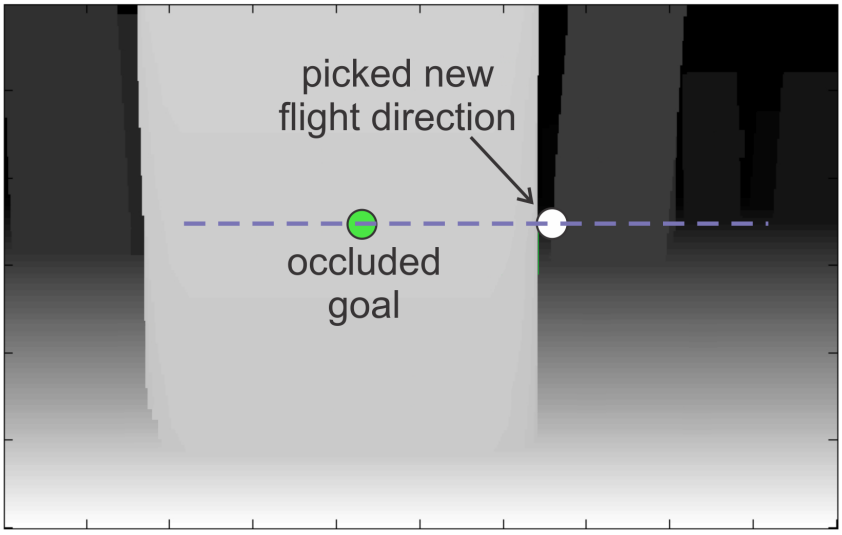

Figure 4.11: The first stage of the planner identifies collision-free flight paths by checking the pixels of the egocylinder against a planning horizon chosen to guarantee a minimum time-to-contact. A goal is projected into the egocylinder (green, occluded) and the nearest collision-free pixel (white) is selected as a candidate path and passed to the trajectory generator. Here, brighter pixels are closer to the camera.

In the second stage, a trajectory generation module attempts to identify a feasible maneuver that can smoothly merge the vehicle onto the infeasible candidate path at the desired final velocity V. In addition to generalizing the infinite-agility approach

by adding full dynamics and deliberative capability, the second stage also greatly narrows the set of admissible trajectories that actually need to be calculated and collision-checked. The straight-line segment that follows the turning maneuver is known to be collision-free from the first stage and need not be checked again, while the required acceleration, velocity, and position are automatically specified at the beginning and end of the turning maneuver (for position, up to an undetermined range):

$$\ddot{\mathbf{x}}(0) = \mathbf{a}_0, \quad \dot{\mathbf{x}}(0) = \mathbf{v}_0 \quad \mathbf{x}(0) = \mathbf{x}_0,$$
$$\ddot{\mathbf{x}}(t_f) = 0, \quad \dot{\mathbf{x}}(t_f) = V\hat{\lambda} \quad \mathbf{x}(t_f) = (\lambda h)\hat{\lambda}.$$

Here, the final flight direction $\hat{\lambda}$ is chosen in the first planning stage, t_f is the total maneuver time, and $\lambda \in (0, 1]$ is a free parameter that modulates the distance to the end of the maneuver.

The set of possible trajectories is further narrowed by insisting on a minimum jerk maneuver, which can be shown to consist of fifth degree polynomials $(x(t), y(t), z(t)) = \sum_{i=0}^{5} \mathbf{a}_i t^i$ using a straightforward application of the Pontryagin minimum principle. Once the endpoints are prescribed, the maneuver has only two remaining free parameters, t_f and λ, that can either be modulated independently or constrained to each other to produce a one-parameter family of admissible trajectories – for example, the total time can be chosen to maintain an average vector velocity associated with the candidate path (fixed speed, fixed direction) such that $t_f = (\lambda h)/V$. Minimum-jerk polynomials can be shown by direct substitution to satisfy the differentially flat quadcopter plant model discussed in the introduction to this chapter,

$$\ddot{x} = (\sin\theta\cos\psi + \cos\theta\sin\phi\sin\psi)\,\frac{u_{\text{collective}}}{m},$$
$$\ddot{y} = (\sin\theta\sin\psi - \cos\theta\sin\phi\cos\psi)\,\frac{u_{\text{collective}}}{m},$$
$$\ddot{z} = -g + (\cos\phi\cos\theta)\,\frac{u_{\text{collective}}}{m},$$
$$\ddot{\phi} = u_{\text{roll}},$$
$$\ddot{\theta} = u_{\text{pitch}},$$
$$\ddot{\psi} = u_{\text{yaw}},$$

where (ϕ, θ, ψ) are Z-X-Y Euler angles and $u_{\text{collective}}$ is the collective motor thrust. The trajectories $(x(t), y(t), z(t))$ correspond to unique control inputs, subject to actuator saturation and a yaw angle trajectory, that can be chosen independently of the translation of the vehicle and determined algebraically using the associated flatness relations.

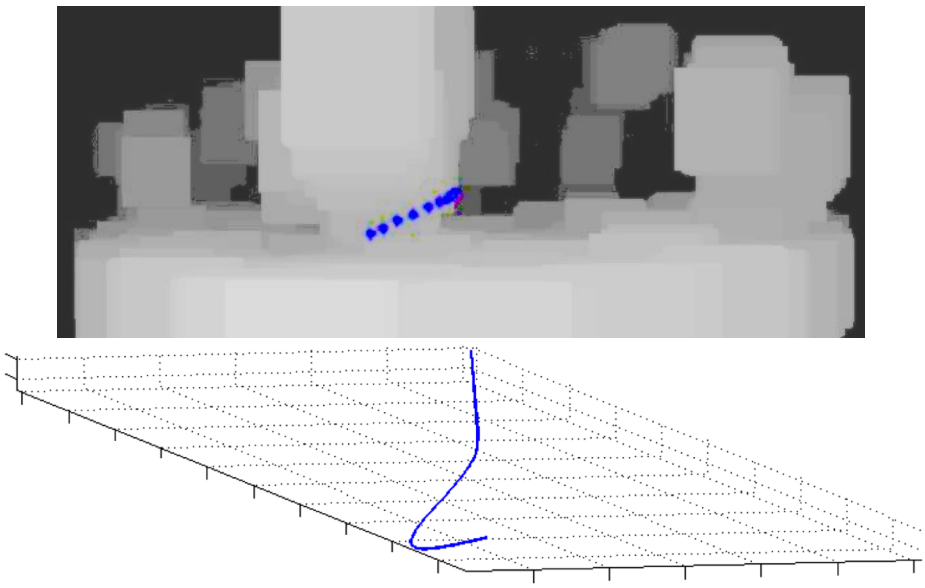

Figure 4.12: Closed-form trajectories are simultaneously maintained in egocylinder coordinates for collision-checking and world coordinates for control. Here, a trajectory (top, blue dots proceeding towards the right) has been selected to avoid a tree (brighter pixels are closer) using a pixel scan and collision-check in egocylinder coordinates. The trajectory is tracked in world coordinates (blue line, bottom) by the flatness-based controller.

The coefficients of the minimum-jerk polynomials are determined symbolically, in advance, as a solution of a small linear system of equations in terms of the initial and final vehicle states and the search parameter λ. This solution step dramatically reduces the amount of computation that needs to be performed onboard and allows for a large number of potential trajectories to be considered during a planning cycle (see Section 4.3 for runtime statistics). Evaluation for control saturation and collisions is further enhanced by a simultaneous dual representation of the trajectories (Figure 4.12) in world coordinates and egocylinder coordinates, which both have a simple closed form and are determined by the single search parameter — the world coordinate trajectories can be easily checked for saturation and converted into control inputs, while the egocylinder coordinates can be immediately collision-checked using a pixel-by-pixel comparison against the egocylinder data structure.

Once a proposed candidate path is available, the trajectory search itself begins by attempting to merge the vehicle onto the candidate radial path at a distance corresponding to the planning horizon (Figure 4.13). The proposed maneuver is collision-checked, evaluated against the control constraints, and returned if successful – otherwise, the distance to the end of the maneuver is decreased along with

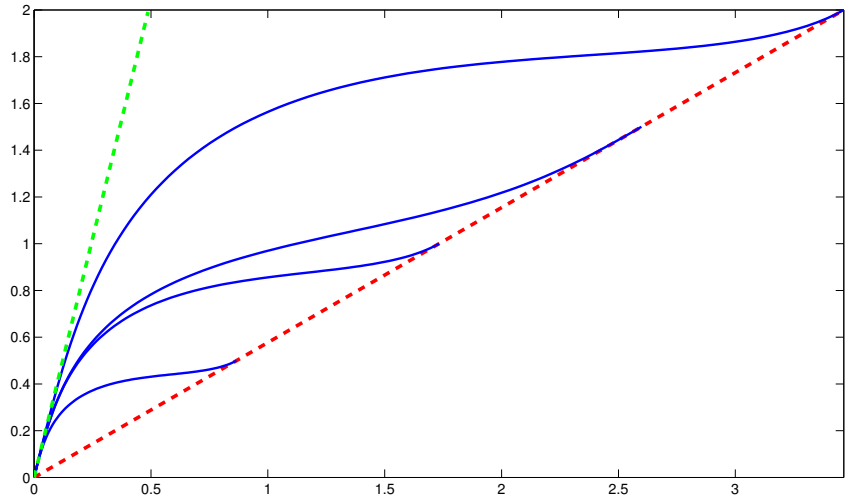

Figure 4.13: The trajectory generator, shown in 2D for clarity, attempts to merge the vehicle smoothly from the green heading (left, dashed) to the red heading (right, dashed) using minimum-jerk polynomials (blue) that are parametrized by the distance covered before merging. By decreasing the length of the maneuver, the trajectory generator can provide a more aggressive turn if a collision is detected.

the total time until either a valid trajectory is identified or the controls saturate, in which case either the total time is increased independently or a different candidate radial path is attempted.

Once the trajectory is available, a low level controller tracks the reference using a standard two degree-of-freedom feedforward design ([25], using a linearized attitude controller), with nested position/velocity and attitude loops corresponding to the slow and fast dynamics of the vehicle model. Although the discussion in Section 4.1 requires trajectories to be at least four times differentiable to be strictly trackable, in order to strike a balance between performance and simplicty we have instead insisted on piecewise C^4 trajectory segments with at least C^2 smoothness throughout. This approach is able to prevent an attitude discontinuity between the turn and the straight-line maneuver, which is inelegant and can disrupt state estimation, while also maintaining the minimum-jerk, low-degree character of the polynomial segments without the additional overhead of an optimization over their coefficients. It is also a justified approximation for the performance regime achievable with our current attitude controller, which has angular velocity dynamics that are significantly faster than position control and only requires feedforward in attitude itself rather than its derivatives. A more general approach increases the de-

gree of the polynomial, perhaps to $n = 7$, which would allow for optimization-free minimum-snap maneuvers with continuity of angular velocity between the turn maneuver and the straight-line segments. This approach would be consistent with aggressive geometric attitude controllers from the literature (as in [19]), which do not provide an angular acceleration feedforward signal in practice and would treat the discontinuity as a step input.

Comparison to Other Approaches

Our receding-horizon approach does differ from a seeded RRT-type formulation, like that of [20], in that it lacks a completeness guarantee in its own right and can become stuck in dead ends. For obstacle avoidance in unstructured and unknown environments where frequent replanning is required, however, complete methods can bottleneck the pipeline (as in the CL-RRT of [17]) and limit flight speeds. For a pure obstacle avoidance scenario in which environments are complex and constantly changing, it is also difficult to justify the overall practicality of placing this extra overhead within a low-level avoidance architecture as compared to a higher level in which it is more straightforward to implement.

We contrast our approach primarily with graph-based, voxel grid approachs such as [4], in which a complete planning algorithm operates over an instantaneous map that is flushed and repopulated entirely anew at each cycle. Although these methods do produce a deliberative plan that can be stably followed towards the goal for a long duration, only the visible obstacle surface is represented or known with any certainty and no additional information is maintained beyond what could be presented in an *unfiltered* egospace. A significant amount of computational effort is spent either on entirely unknown areas or in obviously empty regions, which requires replanning times on the order of hundreds of milliseconds on state-of-the-art embedded computers.

Unless a fully-intensive SLAM or exploration component is required, egospace with temporal filtering can be a considerably more compact and efficient alternative. Our approach is specifically designed to represent visible surfaces, which is ultimately the only data available in any instantaneous representation, and efficiently find paths around them. With the use of receding-horizon trajectory generation (see below for a discussion of planning horizon) and temporal fusion, we share some degree of deliberative action and map reinforcement with complete SLAM-based planning algorithms, and offer forward collision-checking unavailable to reactive

approaches such as [27].

Because our vehicle attempts to transit through an area as expeditiously as possible, unnecessary exploration of occluded areas is a hindrance to progress towards the goal and ought to be avoided as much as possible. For this reason, our approach attempts to restrict deliberative action to visible areas under the assumption that most of the unknown areas in an environment will never be encountered, and need only be considered once they become absolutely necessary (and therefore, known). An unprepared, cluttered environment with a limited sensor horizon will be occluded almost in its entirety after single observation — if the environment is also unstructured, as is the case in an outdoor field setting using range sensing, an exploration-based global motion planner will be forced to rely heavily on pure speculation in its early stages. Accordingly, we leave exploration, mapping operations, and the enforcement of theoretical performance guarantees to higher levels of the mission architecture.

Results

The visual perception pipeline is implemented on the Asctec Mastermind embedded computer equipped with a 1.86 GHz Intel Core2Duo processor. The forward-looking stereo cameras are installed with a baseline of 25 cm, and frame-wise stereo disparity maps (376x240) are calculated by block matching with a search range of 100 pixels. The resolution of the egocylinder image is 660x200. The full pipeline maintains a 10 Hz update rate using both cores of the processor, with computation times for each step in the visual perception pipeline given in Table 1. Typical results for temporal fusion on the egocylinder are illustrated in Figure 4.14. The left stereo image, unfused disparity map, and the corresponding egocylinder images are shown sequentially as the vehicle approaches an obstacle. Temporal fusion yields a more complete scene representation compared to the unfused disparity maps — known regions that would be otherwise discarded at each frame are instead retained in the egocylinder. The propagation process is particularly important for obstacles at close range, which eventually become invisible to stereo matching due to a limited search range and would otherwise present a missed-detection hazard (Figure 4.15). Motion planning experiments were conducted at a number of challenging sites, including a 40 meter course through a forested area, a small dead-end alcove in a built-up area, and an open, obstacle-free wash with distant buildings.

While maintaining a speed of 1 m/s, the vehicle encountered and successfully

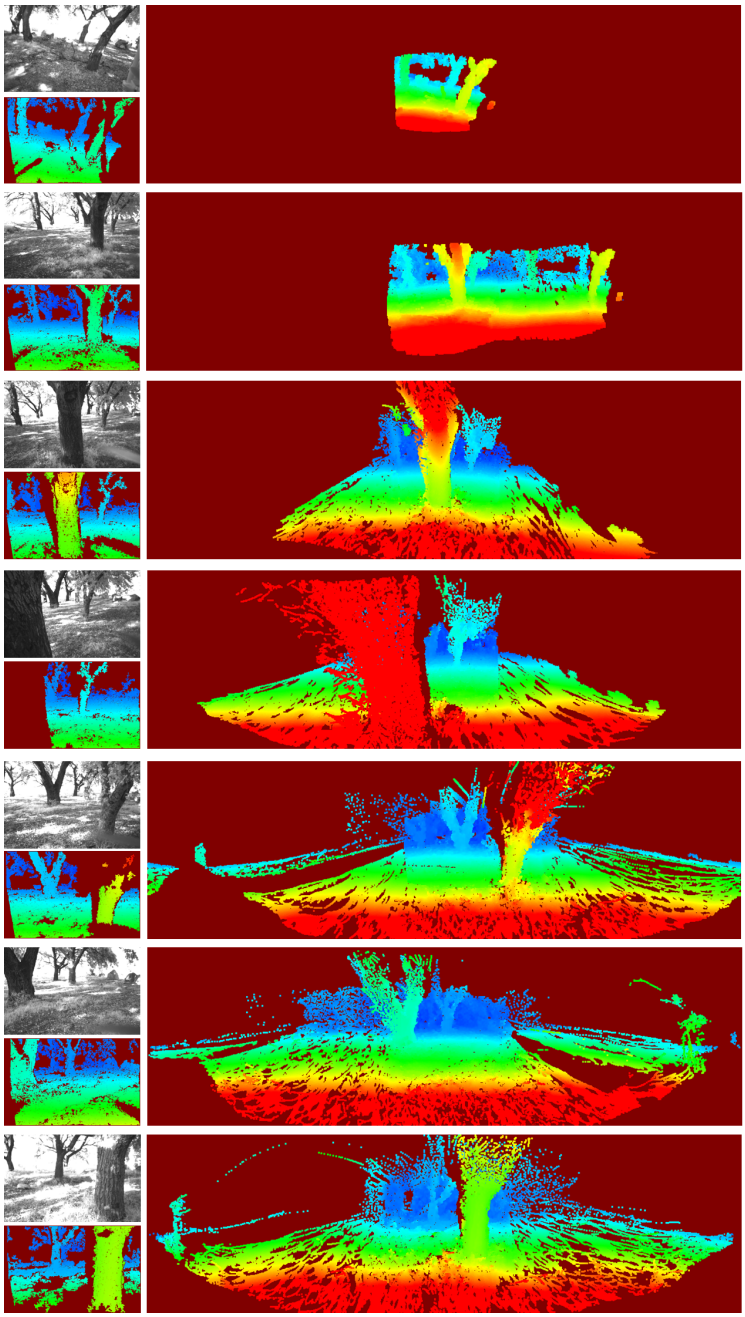

Figure 4.14: The perception pipeline was tested in a forested environment. In the first column are raw images (grayscale) from the left stereo camera and the unfused disparity map, and in the second are full egocylinder maps with temporal fusion. Here, warmer colors are closer, cooler colors are farther, and the maroon background represents no data. In the fourth row, temporal fusion prevents a collision at close range (first column) by propagating obstacle data forward from an earlier detection (second column).

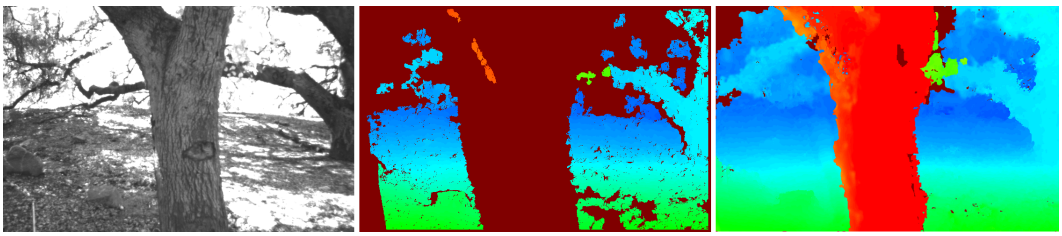

Figure 4.15: While the maximum disparity calculated by the stereo algorithm is too low to resolve the nearby obstacle, temporal fusion propagates the previously observed obstacle, which is then used in the motion planner. Left: left rectified image, Middle: stereo disparity map, Right: temporally fused disparity map.

Table 4.1: Onboard computation times for each step in the visual perception pipeline

Step	Time (ms)
Stereo Matching	100
Cylindrical Mapping	14.4
Temporal Fusion	52.6
C-Space Expansion	4.5

avoided three trees in the forest using the minimum-jerk trajectories of Section 4.3 executed in a receding-horizon fashion (Figure 4.16). This environment required frequent replanning as new obstacles became visible, which was performed efficiently with average runtimes on the Odroid XU4 listed in Table 4.2.

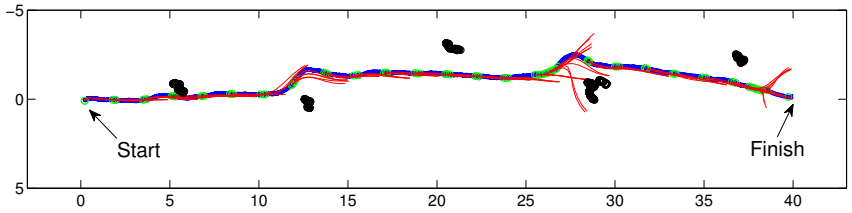

Figure 4.16: While navigating through a forested environment (tree trunks at 2 m height shown as black point clouds), the vehicle took evasive maneuvers around three trees and successfully reached a predetermined destination 40 m away. The path (blue, vehicle travels from left to right) consists of segments of dynamically feasible trajectory segments (red arcs) calculated using the two-step procedure of Section 4.3 and followed in a receding-horizon fashion. Axis scale is in meters.

The egocylinder also proved highly useful in preventing the vehicle from becoming stuck in confined areas when paired with temporal propagation of depth data. Af-

Table 4.2: Onboard computation times for each step in the motion planning pipeline

Step	Time (ms)
Egocylinder Scan	4.0[a]
Trajectory Generation	0.02
Collision-Checking	0.2

[a] For a complete replan (motion planner can reuse old targets)

ter exploring a small alcove and discovering a dead end (Figure 4.17), the motion planner was able to extricate the vehicle towards an open area to its left that was invisible to the stereo pair, but already stored within the egocylinder structure and available to the motion planner.

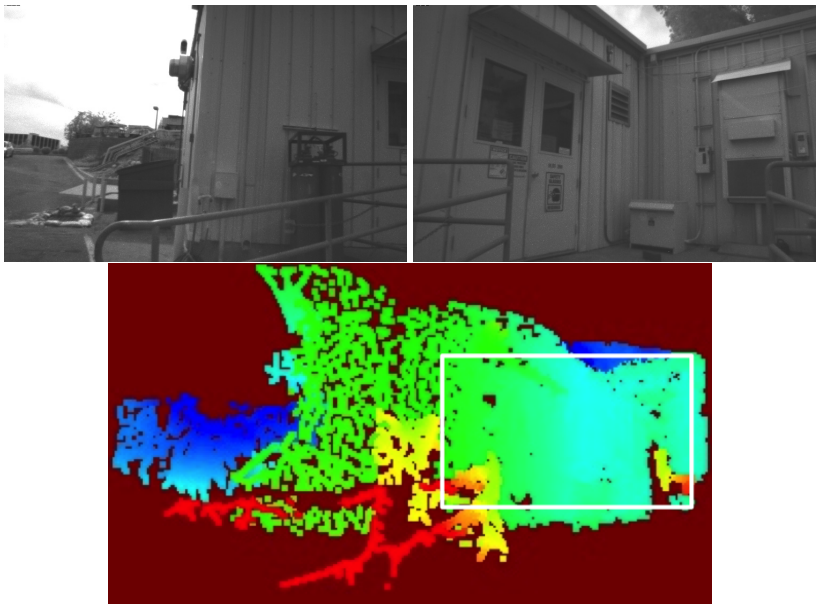

Figure 4.17: When combined with temporal fusion, the egocylinder facilitates exploration of confined areas by propagating knowledge of open areas when they become invisible to the stereo camera. Here, the vehicle has reached a dead end within an alcove (top, raw images shown), but continues to represent an exit to its left within the egocylinder (blue area, bottom left) beyond the visible field-of-regard of unfused stereo (white rectangle). An escape route is also available by climbing over the building, but was excluded by a mission-level altitude ceiling.

By identifying a restricted trajectory search space and exploiting the natural resolution pattern of the egocylinder, the motion planning module was also able to efficiently generate detailed, dynamically feasible trajectories at ranges near the limits of reliable stereo detection. When presented with a destination waypoint inside

a parking garage over 40 meters away, within 5 milliseconds the motion planner identified a 5 meter x 2 meter opening into the garage as the best path towards the target and calculated a collision-free and dynamically feasible path into the building (Figure 4.18).

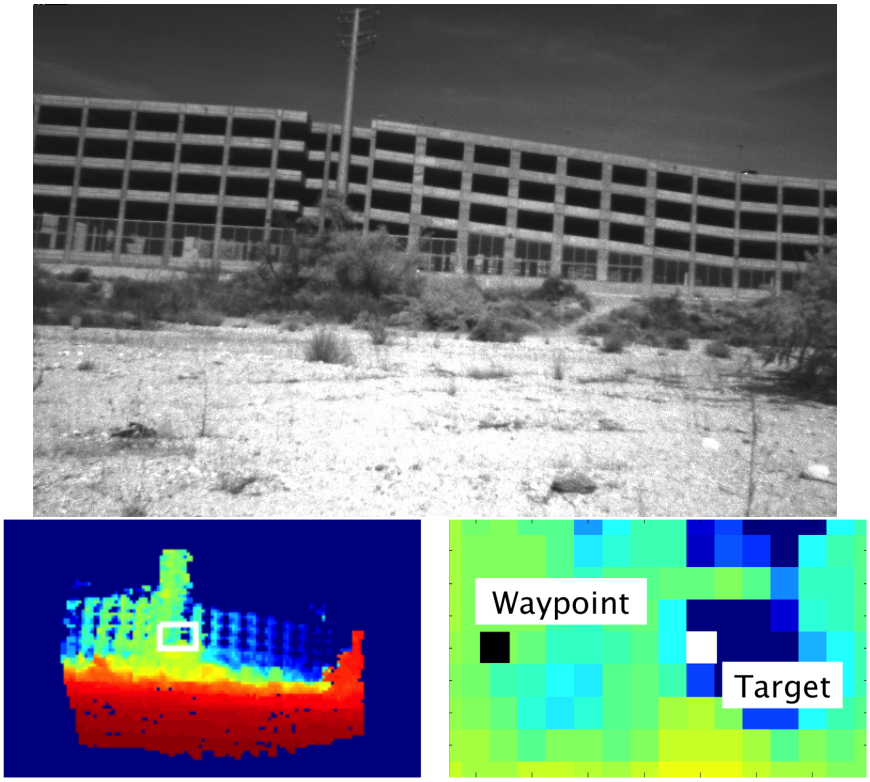

Figure 4.18: Egospace representations are particularly well-suited for producing finely detailed trajectories at extreme ranges. Here, the motion planner has been assigned a waypoint behind a pillar inside a parking garage 45 meters away (raw image, top). After generating and expanding an egocylinder representation (original, bottom left, zoomed for clarity, right), the motion planner projects the waypoint (black square, bottom right) into the image and identifies the closest collision-free target (white square, bottom right) as a candidate flight path for the trajectory generator.

Experimental Conclusion

We have demonstrated an obstacle avoidance pipeline over the full dynamics of a micro air vehicle using a temporally fused egocylinder representation. In addition to being simple to access and compact to store, the egocylinder representation admits an extremely lightweight trajectory selection proceedure, based on a parametrization of flight paths by pixels and differential flatness, that can quickly generate dynamically feasible turn maneuvers towards targets at the range limit of

stereo sensing. This advantage is made more reliable and flexible with temporal fusion of obstacle data within the egocylinder geometry, which improves the accuracy of sensed data and accumulates a 360° representation with a single set of stereo cameras that is useful for navigation in confined and cluttered environments.

Our perception framework also addresses a number of weaknesses of stereo vision, primarily due to the limited and fixed baseline of MAV stereo applications, and significantly increases its power as an obstacle detection tool. The resolution pattern of the egocylinder, which decreases favorably with distance without additional overhead, allows stereo data to inform motion planning and remain useful even at long ranges where it is less accurate. Detection errors at close ranges, which eventually become inevitable for raw stereo due to a limited field of view and disparity search, are prevented by propagation and storage of sensed data with real-time temporal fusion.

4.4 Scalability

We evaluate the scalability of egospace-based approaches, as speeds and obstacle density increase, by considering the effect of lookahead distance on trajectory speed, stability, and safety. Lookahead distance encompasses two distinct, yet related horizons that govern the motion planning problem: the maximum range of the obstacle sensor (the sensing horizon), which is more-or-less fixed by the capabilities of the onboard hardware, and the acceptable time-to-contact criterion (the planning horizon), which is an engineering parameter that depends partially on the vehicle performance and grants the designer some tuning flexibility. In this section, we determine how the expected operating conditions suggest lookahead distance specifications, determine the acceptable vehicle cruise speed, and motivate the receding horizon formulation.

Sensing Horizon

We now specialize to 2D for clarity, and assume a planar egocylinder structure $(r, \theta, z = 0)$. Because the egocylinder is divided into fixed angular increments, it is clear that choosing a planning horizon beyond the sensing horizon offers no safety benefit. If the sensors can detect all obstacles within a distance R, with further points unknown, collision-checking is entirely inconclusive beyond the sensor horizon. We simply deem a path "acceptable" if it remains collision-free up to the horizon, and derive no useful information beyond it. Although seemingly obvious, this observation yields an important relation between perception and dynamic con-

straints: for a general environment, we ought to require that the vehicle be able to come to a stop (for a quadcopter; fixed wing aircraft must instead perform aggressive turnaround maneuvers, as in [60]) by the sensor horizon, lest there be no avenue of escape from the arrival of impenetrable obstacles.

To estimate the speed limit afforded by an obstacle sensing suite with horizon R, we consider the ability of our quadcopter model (Equations 4.12) to come to a stop from a cruise velocity V in level flight, with initial velocity vector $(V,0,0)$, initial yaw angle equal zero, and initial pitch and roll equal zero. The vehicle then uses only its pitch and thrust authority to come to a level stop as quickly as possible (that is, time optimally). This stopping procedure involves two trim primitive segments — one in which u_{pitch} is held at its minimum saturated value $-u_{\max}$, and a second in which pitch is held at the minimum value $\theta_{\min}$ capable of sustaining level flight. Due to the time-scale separation between attitude control and position control that motivates our trajectory tracking architecture, for clarity we further assume that minimal braking is achieved during the pitching maneuver. As a result, it can be treated as a small additional time-delay (as in [60]), which is determined by integrating forward the constant control input until $\theta_{\min}$ is acheived. We consider actuation latency later in this section.

The value for $\theta_{\min}$ itself arises from substitution of the saturated inputs into the expression $\ddot{z}$ — we set $\phi = 0$, $\theta = \theta_{\min}$, $u_{\text{collective}}$ to its maximum, $T_{\max}$, and enforce level flight: $\ddot{z} = 0$. Accordingly,

$$\theta_{\min} = -\arccos\left(\frac{mg}{T_{\max}}\right). \tag{4.12}$$

For our experimental system, $m = 1.7$ kg, $T_{\max} = 30$ N, and $\theta_{\min}$=-57°. Substituting into the equation for $\ddot{x}$ and integrating twice gives

$$\dot{x}(t) = V - t\sqrt{\frac{T_{\max}^2}{m^2} - g^2}, \tag{4.13a}$$

$$x(t) = Vt - \frac{t^2}{2}\sqrt{\frac{T_{\max}^2}{m^2} - g^2}, \tag{4.13b}$$

where we have used the identity $\sin(\arccos(x)) = \sqrt{1 - x^2}$ and assumed without loss of generality that the vehicle starts at the origin. The vehicle requires a time

$$t_f = \frac{V}{\sqrt{\frac{T_{\max}^2}{m^2} - g^2}} \tag{4.14}$$

to come to a stop, over which it covers a distance $x(t_f)$. Setting $x(t_f) = R$ allows the sensor horizon required to accomodate the cruise velocity V to be specified, and relates the perception and dynamical parameters in a single expression:

$$R = \frac{V^2}{2\sqrt{\frac{T_{\max}^2}{m^2} - g^2}}, \tag{4.15}$$

which provides an estimate of the required sensing horizon and can be used during the design process to assist in sensor selection. From a performance analysis perspective, R is prescribed by hardware constraints and V must be determined instead (Figure 4.19). The expression is easily invertible, and the above experimental stereo-based system is capable of stopping within 75 m from a speed of 46 m/s if it applied maximum braking power — a speed limit that could never actually be acheived by our system and instead arises from its generous thrust-to-weight ratio $T_{\max}/(mg) = 1.8$.

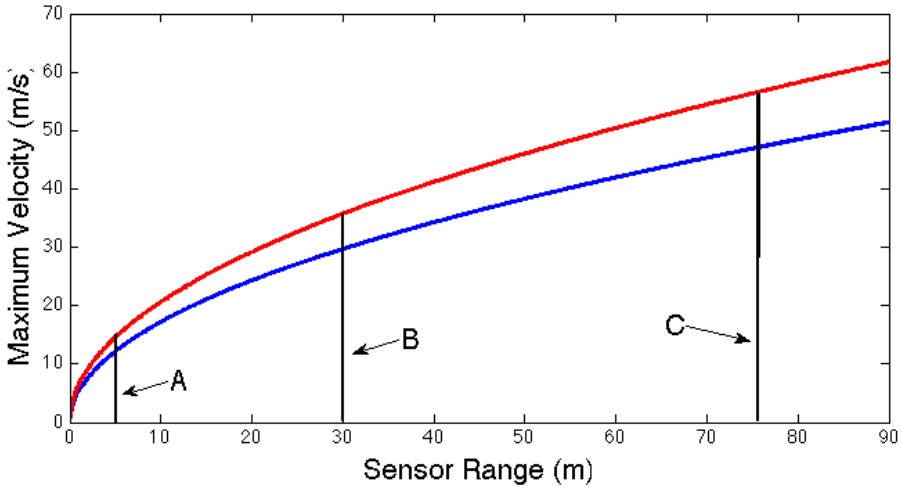

Figure 4.19: The maximum velocity afforded by sensor horizon for an Asctec Pelican (blue — thrust-to-weight ratio 1.8) and an Asctec Hummingbird (red — thrust-to-weight ratio 2.3). Sensor A is the structured light sensor of [21], with a range of 5 m, B is the Hoyuko UTM-30LX, with a range of 30 m, and C is the stereo system presented in the previous section, with a range of 75 m (at one full pixel of disparity). In the absence of latency the resulting speed limits are 12 m/s, 30 m/s, and 46 m/s respectively on the Pelican, and 15 m/s, 35 m/s, and 56 m/s on the Hummingbird. With the exception of the structured light sensor, the sensing horizon is not the limiting factor for vehicle speed.

Although the vehicle can theoretically stop by the horizon at extreme speeds, no room is left for error or the characteristics of real sensors, computational resources,

and actuators. On the perception side, depth data arrives at discrete intervals with some latency and is never instantaneous. For the 10 Hz perception system with runtimes given in Table 4.1, at the moment depth data arrives for motion planning it is already almost 200 ms old, which reduces the effective sensor horizon by a corresponding amount. Because the vehicle must also commit fully to a stop the moment unfavorable depth data arrives, even if no obstacles are detected within the horizon the possibility of their presence a small distance ϵ further beyond must not be discounted — the vehicle may not fly so fast as to prevent itself from stopping in time, if it must, on the next measured frame. If an impentrable obstacle set lies at $R + \epsilon$, it will actually have advanced closer by a distance proportional to the measurement interval t_{frame} by the time it is actually detected. Accordingly, we take ϵ down to zero and subtract Vt_{frame} from the actual sensor horizon.

The motion planning pipeline has negligble latency once the depth data is available to use, but on the control side it takes roughly 100 ms for the vehicle to perform the pitch maneuver that preceeds braking. Consequently, a total latency time t_l elapses before the vehicle is able to actually respond to visual input. During this time it covers a distance Vt_l, which leads to a quadratic expression in V:

$$R - Vt_{frame} = Vt_l + \frac{V^2}{2\sqrt{\frac{T_{\max}^2}{m^2} - g^2}}. \tag{4.16}$$

For our system, latency and the discrete frame rate further reduces the stereo speed limit to just under 42 m/s. Operation at such a high speed, if it could even be acheived under the vehicle aerodynamics, would be marked by frequent braking maneuvers that are extremely inelegant and wasteful of battery life. In spite of its clear aesthetic downsides, however, the ability of such a system to react to obstacles would ultimately not be limited by the sensing regime.

We can consider sensor-level design charcteristics that result from our analysis. For stereo vision suite with a fixed baseline (which is ultimately limited by what can be practically carried on a small vehicle) and fixed optics, maximum range increases linearly with the resolution of the camera in each row (which is along the epipolar line for rectified images). Continuously increasing camera resolution in an attempt to improve sensor horizon, however, will eventually overwhelm the limited onboard computational resources entirely, increase latency to unacceptably high levels, and counterproductively reverse the increase in speeds afforded by the longer sensor horizon. For a given design speed, we can specify the sensor horizon along with

both the camera resolution and the computational resources (through the latency tolerance and frame rate) required to achieve it.

Planning Horizon

The sensor horizon ultimately limits the range to which motion plans can be collision-checked and verified in a concrete sense, but the planning horizon eventually used is also dependent on the obstacle layout in a given environment. Because the planning horizon is incrementally decreased when a search for a collision-free direction fails, an overly optimistic choice is wasteful and will require multiple fruitless egospace scans until a more appropriate horizon is reached.

We may estimate the minimum planning horizon needed to safely avoid obstacles using a similar analysis as used to determine the sensor horizon. We assume that a point vehicle is initially at the origin with a velocity vector $(V,0)$, and flies in the xy plane near a C-space expanded wall lying entirely right of the line $x = h$ with $V,h > 0$. To determine the minimum planning horizon h required to safely avoid the wall while maintaining the speed V, we order a 90 degree circular turn at the origin with the full roll, thrust, and yaw authority of the vehicle and calculate the closest h that will not produce a collision. Assuming that the planning horizon is less than the sensor horizon, we may ignore latency in vision (the vehicle knows the wall is coming, and simply decides when to react) and treat the initial roll maneuver as a time delay as in the sensor horizon discussion. After the time delay t_l, the vehicle holds $\phi_{\max}$ for the first 90 degrees of the circular turn, after which the wall has been successfully avoided. Balancing the maximum thrust with gravity gives

$$\phi_{\max} = \arccos\left(\frac{mg}{T_{\max}}\right). \tag{4.17}$$

Assuming that the vehicle's acceleration is directed towards the center of the circular turn using a coordinated yaw command, the vehicle turning radius is

$$R = \frac{V^2}{\sqrt{\frac{T_{\max}^2}{m^2} - g^2}}, \tag{4.18}$$

where we have used the identity $\sin(\arccos(x)) = \sqrt{1-x^2}$ and the Newtonian equation for centripetal acceleration $a_c = V^2/R$. A 90 degree circular turn will cause the vehicle to travel a distance R in x, to which we add the effect of latency and determine h:

$$h = Vt_l + \frac{V^2}{\sqrt{\frac{T_{\max}^2}{m^2} - g^2}}. \tag{4.19}$$

We note by comparison with Equation 4.15 that a circular evasion maneuver is a good deal less effective than a full vehicle stop in preventing collisions — after latency, a circular turn will require twice the distance of a hard braking maneuver — but is capable of maintaining the vehicle cruise speed and is more typical of the smoother operation expected of planned motion.

In an environment that requires less extreme maneuvering than an infinite wall, a more realistic planning horizon is obtained by restricting our analysis to a wall of finite extent. We now suppose that the vehicle attempts to maneuver around a wall given by the set $\{(x, y) : x = h, y \in (-L, L)\}$ using a circular arc that, for $L < R$, is generally less than 90 degrees. Taking account the latency required to enter a turn but not to undo it (the ability to undo a turn has no impact on obstacle avoidance capability in this case), we set the planning horizon equal to distance at which the wall first intersects the circular turn — any closer and a collision will result (Figure 4.20).

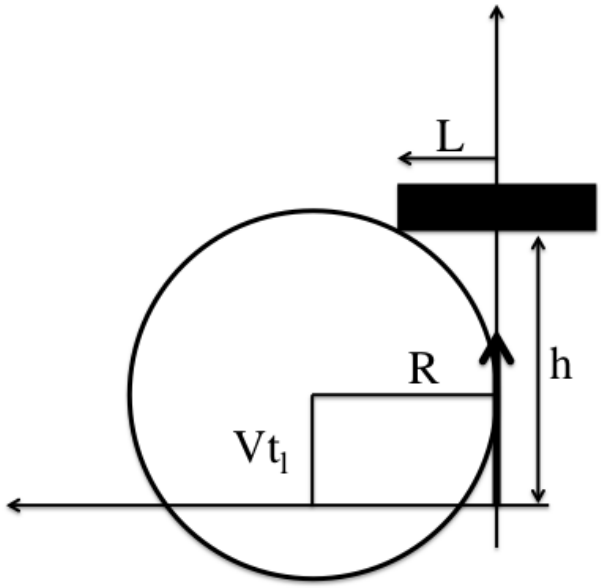

Figure 4.20: For a wall of finite extent $L < R$, a more aggressive minimum planning horizon h may be found using the geometry of circular turns. A collision results if the wall penetrates the circular turn, which relates t_l and V, through R, to h.

The latency and $\phi_{\max}$ criteria determine the equation of the circle, $(x - Vt_l)^2 + (y - R)^2 = R^2$, into which the position of the corner (h, L) is substituted. Solving for h and requiring $R > L$ (otherwise, the 90 degree turn solution holds) gives

$$h = Vt_l + \sqrt{L(2R - L)}, \tag{4.20}$$

which is asymptotically linear in V upon substitution of Equation 4.18 for R. For the forested test site described previously, the half-width L of a typical tree is approximately 30 cm and the performance profile is considerably more favorable than in the infinite wall case (Figure 4.21).

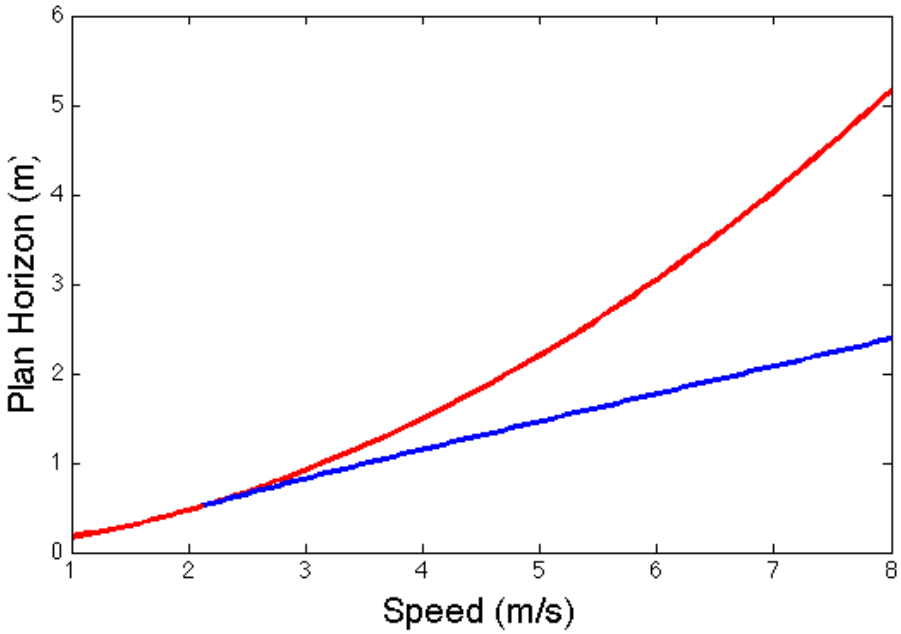

Figure 4.21: The minimum planning horizon at each velocity for the infinite wall case (red) and for a 30 cm radius tree (blue) for our experimental Asctec Pelican system.

In a sparse obstacle environment longer horizons help avoid (but do not eliminate) the possibility of becoming trapped with a command of zero velocity, which would require the invocation of a higher-level extrication routine (Figure 4.22). Longer planning horizons also allow for higher speeds by a line of reasoning similar to the sensor horizon discussion: by collision-checking further out, a longer verified motion plan gives the vehicle more time to stop or revise its trajectory before hitting new obstacles. In the interest of compromise between the more certain existence of short-horizon paths and the benefits of longer horizons, an adaptive selection scheme is appropriate. The vehicle attempts to increase the planning horizon by a small amount each time until it is equal the lesser of the sensor horizon or the remaining distance to the goal (lest it be repeatedly overshot).

Motion planning for systems in *a priori* unknown environments, such as ours, is typically run in a receding-horizon fashion. Although the endpoints of each motion

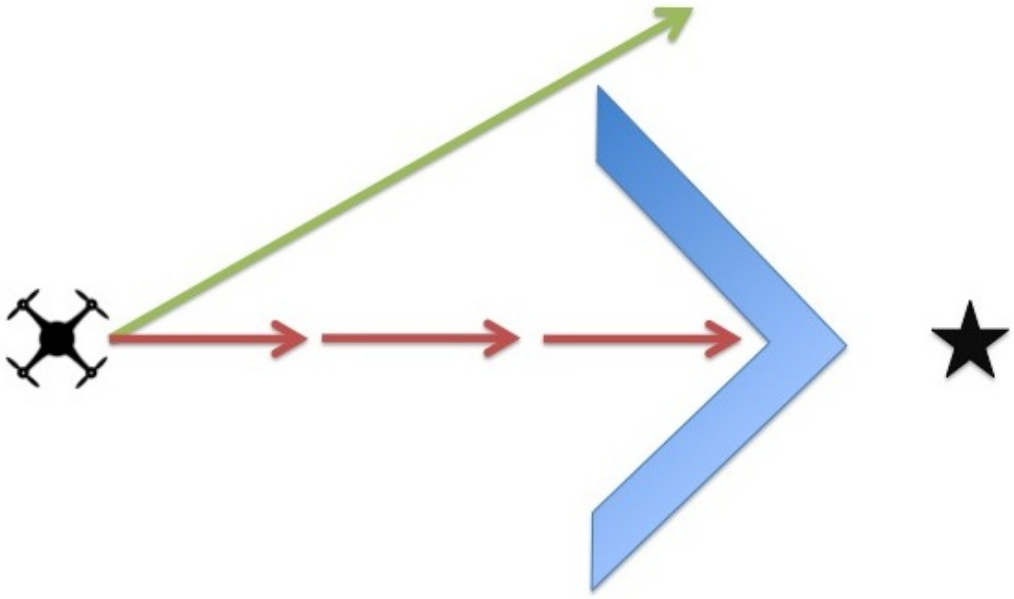

Figure 4.22: Longer planning horizons can diminish the potential for the vehicle to become trapped on its way to a destination (black star). Whereas a long-horizon planner was able to find a clear path to the left of the alcove, the short-horizon planner was unaware of the trap until it could no longer extricate itself without resorting to high-level considerations.

plan are not necessarily reached, how they are selected and how the plans themselves are revised has important effects on performance. It is clear that the experimental systems described in the previous sections rely on occasional replanning to actually reach their destinations in a timely manner (Figure 4.23) — following a long radial path to its endpoint without intermediately checking for a faster route to the destination reliably prevents collisions, but can lead to highly uneconomical paths that may not converge to the destination at all. At the same time, excessively frequent replanning can lead to trajectory and control instability, especially in the presence of depth and state estimation noise.

On our experimental system, which was tested for the most part around thin obstacles (trees), a hand-tuning approach was sufficient to determine an appropriate replanning horizon in each test scenario. As an extension to general environments, however, a more systematic approach places the receding horizon endpoint at the destination itself, rather than on the radial path, and relies on the collision-checking routine to determine the timing of revisions. Once a trajectory is no longer known to be collision-free for a sufficiently long time duration, as determined by the collision-

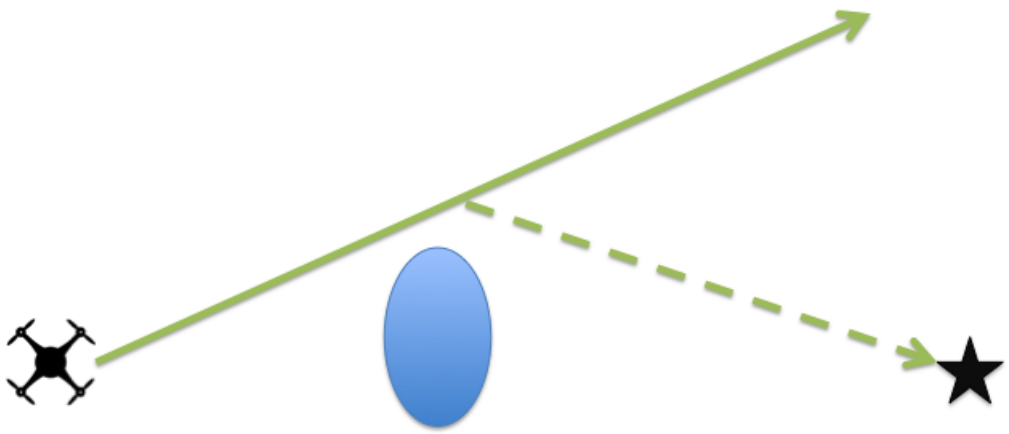

Figure 4.23: Although the long-horizon radial path (solid) allows a collision-free trajectory to be determined in a single planning cycle and followed for a considerable distance, its endpoint is very far from from the destination. A forced replan (dashed) would have led to a much more economical path.

checker, a new plan towards the destination is ordered. This approach requires planning to be permitted to proceed in unknown, occluded areas, which in turn requires a heuristic assumption of obstacle structure behind the visible surface. One solution is to use the simulated egospace approaches discussed in Chapter 3, which have the additional benefit of probabilistic completeness guarantees for each structural heuristic. A simpler approach motivated by the scanning and seeding philosophy discussed in this chapter continues to use candidate radial paths, but with an additional modification step in which a feasible path also smoothly *departs* the path and reaches the destination (Figure 4.24). The departure point is tuned to maintain a collision-checking horizon that allows for safe operation at the vehicle cruise speed.

4.5 Velocity Space and Moving Obstacles

The experimental representation and planning pipelines discussed in this chapter ultimately derive their efficiency from a close connection between egospace and the vehicle's "velocity space" — the pixels of egospace can be thought of as encoding the set of velocity vectors available to the vehicle at each instant, with the generalized depth coordinate serving as a safety metric. This equivalence is direct for the infinite agility case, in which the pixel coordinates completely and unambiguously describe the actual future path of the vehicle.

For the fully dynamic case, pixel coordinates instead index variable-aggression, fea-

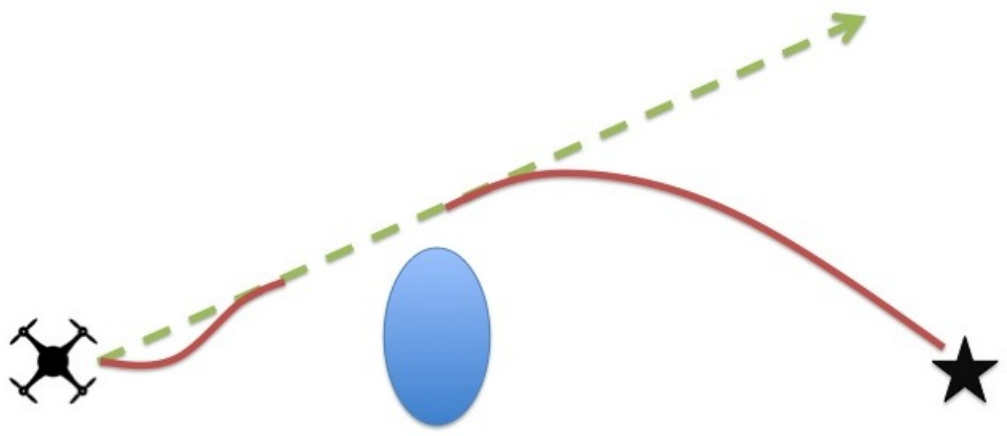

Figure 4.24: The collision-checker can be used to determine replanning timing. In scenario, the the candidate radial path (green, dashed) is modified to include both a merger and a departure maneuver that satisfy the vehicle dynamics (red). The departure maneuver ends at the destination, but passes through an unknown area — it is imperative that either some structual information about the occluded region is available or the vehicle can collision-check far enough in advance to maintain safe operation. The modified path can be followed directly to the destination if necessary, but if a collision is detected a replanning cycle is triggered.

sible maneuvers. The additional aggression tuning structure we have imposed on the experimental motion planner earlier in this chapter, however, allows the exact functional form of each maneuver to be largely abstracted away after collision-checking. By mutually restricting the final velocity and the maneuver time to maintain an average velocity vector over the course of the turn, higher-level considerations need only consider the average velocity vector to which each trajectory corresponds — once it is verified to be collision-free, the exact functional form of a turn maneuver is immaterial to its eventual kinematic outcome. Accordingly, the direct equivalence between pixels and velocity vectors can be recovered, even in the presence of full dynamics, at distances beyond the merger of feasible turn maneuvers with their straight-line targets.

The equivalence we have observed allows classical multi-agent motion planning techniques, which operate in the space of radial vehicle trajectories in the absence of static obstacles, to be merged immediately into egospace static obstacle avoidance with minimal additional structure. Cartesian occupancy grids, on the other hand, require an additional time dimension to adequately represent moving obstacles for motion planning. The extra coordinate usually emerges either as part of a generalization to a 4–D structure in space-time (see Chapter 7.1 of [38]) or through

velocity scheduling [94], in which a path among static obstacles is chosen and its *timing* modified to accomodate obstacle motion. In egospace, the connection between velocities and pixel coordinates collapses the moving obstacle problem and the static obstacle problem into the same 2.5–dimensional egospace structure that we have developed in our previous analysis.

For the rest of this chapter, we develop moving obstacle avoidance in egospace using the experimental methods described above. After introducing the modifications required to merge static and moving obstacles into a unified egospace representation, we exhibit simulation results that add the challenge of moving, non-compliant, and non-communicating agents to the experimental regime we have considered in the static case.

Velocity Obstacles

Multi-agent collision avoidance is typically dealt with using a velocity obstacle formulation [95], in which the relative velocity and size of each vehicle pair is used to calculate the set of velocities that will ultimately result in a collision. This procedure is based on the well-known maritime principle of constant bearing, decreasing range (CBDR; Figure 4.25) — if a vessel appears to be approaching but lingers in the same relative direction when viewed from second ship, a collision will result unless evasive action is taken. More formally, two vehicles are defined to be on a collision course if the relative velocity vector of the pair, when placed at the first vehicle, is directed at the second vehicle.

The formal version of the CBDR principle can be used to systematically calculate a *velocity obstacle* region, in the space of instantaneous vehicle velocity vectors, that contains the set of velocity vectors that will result in a collision and must be avoided. For clarity, we consider a two-dimensional engagement between two circular vehicles, one of which is controlled (vehicle A with variable velocity $\mathbf{v}_A$) and the other non-compliant (vehicle B, with a known, fixed velocity $\mathbf{v}_B$). We may abstract vehicle A to a point by performing a C-space expansion on vehicle B, which we now assume to have a radius r_B enlarged by that of vehicle A.

A collision will result if the relative velocity vector, when placed at A, is directed at B — the set of inadmissible relative velocity vectors, $\mathbf{v}_{AB} = \mathbf{v}_A - \mathbf{v}_B$ is a cone $C \subset \mathbb{R}^2$. Adding $\mathbf{v}_B$ to each vector in C gives the *velocity obstacle* in $\mathbf{v}_A$: any vector $\mathbf{v} \in C \bigoplus \mathbf{v}_B$ is an invalid velocity for A because it is on a collision course with B (Figure 4.26). Because B cannot be expected to take evasive action, the motion

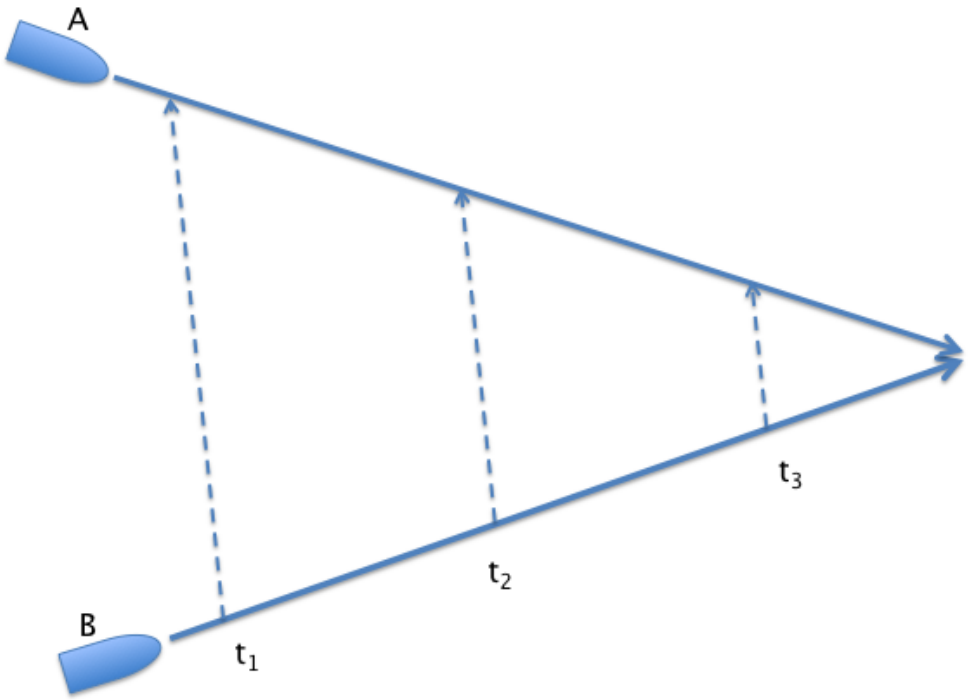

Figure 4.25: If vessel A remains at the same bearing relative to the velocity vector of vessel B and also closes in distance, a collision is inevitable if neither yields.

planning objective for A is to select a velocity vector that is not in this set.

We now assume that vehicle A has infinite agility and attempts to maintain a fixed cruise speed V, seeking only a collision-free instantaneous direction θ. Because the cruise velocity is fixed, we need only check the circle of radius V in velocity space against the velocity obstacle. If the circle is contained entirely inside the velocity obstacle, a collision is inevitable at that speed. The intersection of a cone with a circle, if it occurs, has a straightforward analytic solution equal at least one but no more than two intervals $[\theta_0, \theta_1]$ and $[\theta_2, \theta_3]$. Any θ outside of these intervals represents a collision-free velocity vector.

A three-dimensional engagement at a fixed speed V, on the other hand, has an exact solution corresponding to the region of a sphere contained in the cone — except for the most trivial cases (such as static obstacles, which intersect with the sphere in a circle, or inevitable collisions, in which the entire sphere is contained inside the cone), this involves a quartic equation in x and y paired with a quadratic equation in x, y, z [96].

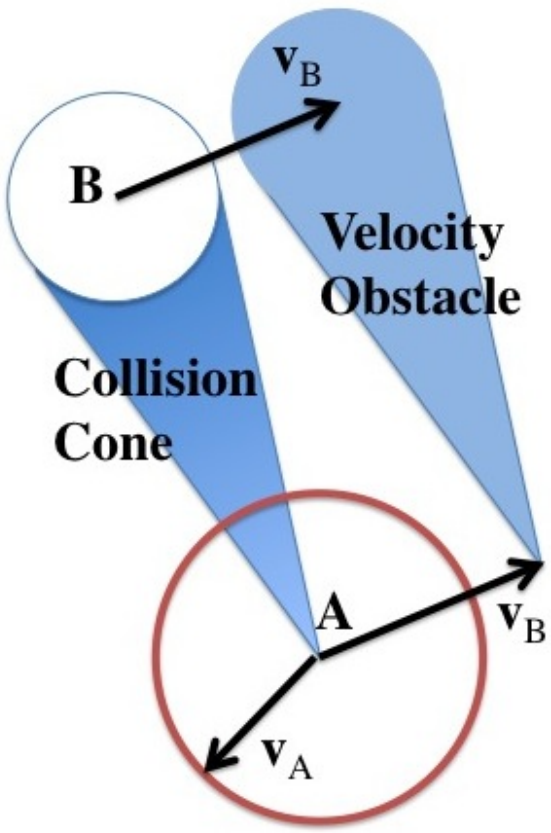

Figure 4.26: Velocity obstacle formalism is the classical approach to multiagent coordination problems. For two vehicles in two dimensions, denoted A and B, a C-space expansion first abstracts A to a point and B to a circle. The set of relative velocities $\mathbf{v}_{AB} = \mathbf{v}_A - \mathbf{v}_B$ that are directed at B result in collisions and lie in the *collision cone*. Adding v_B to the entire collision cone set gives the set of inadmissible $\mathbf{v}_A$, which is known as a *velocity obstacle*. The red circle centered on A is the set of $\mathbf{v}_A$ with constant magnitude V, which in this case does not intersect the velocity obstacle and will not produce a collision.

The first step towards a solution determines whether the sphere intersects with any cones at all (there may be more than one) or is located entirely inside any of them (which indicates an inevitable collision). This is a classic problem in collision-checking and helps avoid needless computation if an intersection is not actually present. If a non-trivial intersection is detected, we may avoid a simultaneous numerical solution of the quartic and quadratic by instead sampling points in S^2, for which we ultimately we will use the set of pixels in egospace (see below). This reduces the problem to an embarrassingly parallel scan that checks which points of the sphere lie inside the cone. Given $\mathbf{v}_A$, $\mathbf{v}_B$, r_B, and the position separation vec-

tor towards B, $\mathbf{x}_{AB}$, all represented in the instantaneous body frame of A, we may represent the cone using an equality constraint

$$(\mathbf{u} \cdot \mathbf{d})^2 - (\mathbf{u} \cdot \mathbf{u}) \cos^2 \theta \leq 0, \tag{4.21}$$

where $d = \mathbf{x}_{AB}/\|\mathbf{x}_{AB}\|$, θ is the cone half-aperture, and $\mathbf{u} = (x, y, z)^T - \mathbf{v}_B$. Each point on the sphere is checked against the inequality with the assistance of a look-up table for the transformation from (V, θ, ϕ) to (x, y, z), and marked "unsafe" if the inequality holds. The scan is highly efficient — for a C++ implementation on a 2.8 GHz dual-core Intel Core i7 processor with a 200x660 pixel egocylinder, it takes less than one millisecond to process each moving obstacle.

Velocity obstacles and egospace

When a vehicle attempts to maneuver around a moving obstacle with a fixed speed, the set of unsafe directions is indexed by coordinates on S^2 (S^1 in 2D) and can be merged immediately with the angular layout of egospace and its representation of static obstacles. We construct an artificial obstacle in egospace at the pixel locations corresponding to these directions, and insert it into the structure as a mask over the static obstacle data (Figures 4.27 and 4.28)— for the purposes of this thesis, we do not speculate upon the source of moving obstacle detection and tracking. Once the artificial obstacle is in place, motion planning can proceed as if the moving obstacle were actually static (with an effective location and extent, of course, that differs from any of its instantaneous positions in egospace) using the equivalence between pixels and velocity space. Accordingly, we have merged the moving and static obstacles into a single representation without the use of an additional time coordinate or explicit velocity scheduling.

The modified velocity space principle can be employed used when the dynamics of vehicle A are significant, but with the minor additional requirements that the initial turn maneuver be completed before any interaction with vehicle B is expected, and that any departure turn maneuver also be completed afterwards. This step occurs after the candidate seed path is chosen, and is equivalent to the turn aggression tuning operation of the static case.

The method described here is based on the *first order* velocity obstacle assumption, which does not take into account the acceleration of vehicle B and anticipates that its future action will be to continue on its current velocity vector — an assumption of neutral non-compliance. Maneuverability of B incorporated into the planning problem with a measurement and update of v_B at each step.

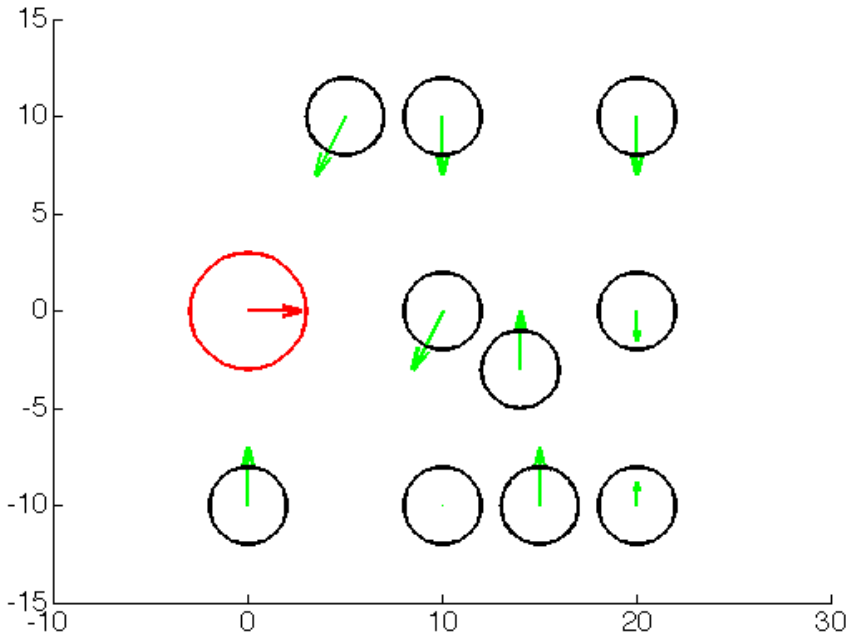

Figure 4.27: A top-down view of the moving obstacles encountered in the engagement of Figure 4.28 — the expanded obstacles (black circles) are non-compliant with instantaneous velocity vectors (green arrows; xy planar projection shown), while the controlled vehicle must select a velocity vector of predetermined magnitude from the red circle. We note that the actual engagement is three-dimensional, and that the moving obstacle located at $(10, -10)$ has a velocity vector directed out of the page (not shown).

Summary

In this chapter, we have specialized the theoretical development of motion planning in egospace to obstacle avoidance on a real quadcopter aircraft. Through a performance evaluation based on accepted metrics of online runtime and human intervention rate, we demonstrated that vision-based egospace obstacle avoidance is safe and extremely efficient with entirely onboard computing and visual hardware. We then conducted a scalability analysis that asctertains the limits of egospace motion planning in terms of sensor and actuator performance, and provided an extension to moving obstacles based on a connection between egospace and the concept of the velocity obstacles.

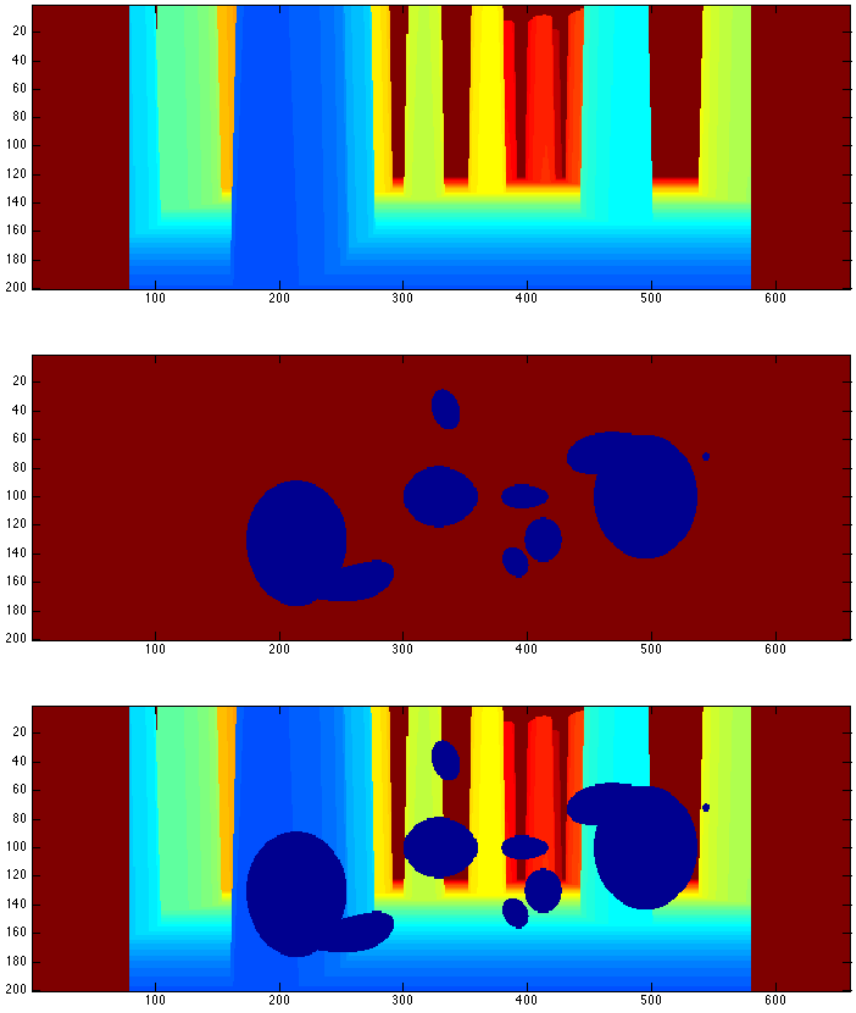

Figure 4.28: The vehicle is asked to proceed through the simulated forest towards a destination (static depth data, top), but with the addition of 10 moving and non-compliant obstacles in the same environment (Figure 4.27) that produce a collective velocity obstacle (middle, unsafe directions in blue). The velocity space obstacles denote invalid flight directions, and serve as a mask over the static data that further restricts the set of vehicle actions for a given speed (bottom). For each obstacle and proposed speed, it takes less than one millisecond to generate a velocity obstacle representation in egospace.

Chapter 5

CONCLUSIONS

In this thesis, we have started from the idea of the depth image as a C-space representation and developed a lightweight obstacle avoidance pipeline based on a general range sensing geometry and full configuration flat dynamics. Our analysis followed from two structural observations — that the radially-aligned geometry of egospace offers efficient collision-checking, compactness advantages, *and* a natural set of candidate flight paths, and that configuration flatness in world coordinates (x, y, z) immediately extends to egospace coordinates (u, v, δ). Combining these two characteristics allowed for the construction of a highly efficient obstacle avoidance algorithm that can generate dynamically feasible, collision-free trajectories at extreme ranges with minimal computational overhead. The computational savings afforded to our experimental approach ultimately arose from the use of egospace coordinates to restrict the size of the trajectory search space, which includes a highly efficient 2D pixel scan followed by a search over a single maneuver aggression parameter.

We performed a theoretical analysis of the egospace motion planning problem in a more abstract sense, which allowed a number of performance and applicability guarantees to be established. We first showed that any traditional Cartesian obstacle representation can be converted invertibly to egospace by construction, which provides a theoretical basis for its ability to faithfully represent environments. Compactness and search efficiency asymptotics were then compared for both egospace and for the uniform Cartesian grids that are the principal alternative in the literature for MAV use cases. We then considered the motion planning problem in egospace, in which both configuration flatness and the use of complete motion algorithms from the literature were extended to general egospace coordinates. A number of approximations to the full configuration flat problem — including a bypass of vehicle dynamics entirely, relaxed completeness requirements, and fixed-form motion primitives — were also demonstrated and the conditions of their applicability determined.

An experimental analysis, motivated by the theoretical results of the preceding chapters, was performed on a quadcopter testbed equipped with visual range sen-

sors. The reliability of the representation and obstacle avoidance pixel scan method were determined through a flight test suite in a challenging forested environment. Dynamically feasible, configuration flat trajectories were also specialized to the quadcopter plant model and run in a series of field environments. This testing regime taxed the efficiency of our trajectory selection and generation algorithm, as well as the ability of our egospace represention to capture detail at extreme ranges with large fields of view. Scalability requirements were then considered, including the limitations of sensing, planning horizon, and vehicle actuators, along with an extension to moving obstacle environments using a connection between egospace and candidate vehicle velocities. Egospace obstacle avoidance methods were shown to be safe and efficient on a real aircraft, and appear to offer the potential for more widespread unsupervised use of field aircraft for applications beyond basic research using current-generation embedded computing.

Future Work

Our work on obstacle avoidance occurs at a relatively low level within a vehicle architecture, however, and also leaves open the potential for study of its interaction with higher levels of motion planning and operation. Our approach has been developed specifically for *a priori* unknown environments that must be sensed as the vehicle encounters them, but a number of situations in which higher-level supervisory input may be required have been neglected. The completeness guarantees that we have obtained assume that an accurate estimate environmental structure is available, which was required in order to allow the vehicle to safely pass through occluded areas. Any such assumption is actually the result of high-level decision making, and if unavailable or incorrect would ultimately require high-level extrication instructions to be issued to the obstacle avoidance layer.

Micro air vehicles are also becoming increasingly complex and can incorporate significant aerodynamic effects and exotic bioinspired propulsion (for example, the "Bat Bot" of [97]). The possibility of moving beyond the class of differentially flat vehicles and incorporating complex effects into motion planning is an open question.

BIBLIOGRAPHY

[1] Amazon.com. (2017). Amazon Prime Air, [Online]. Available: https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011 (visited on 09/20/2017).

[2] E. Landau. (2015). Helicopter could be 'scout' for mars rovers, [Online]. Available: https://www.jpl.nasa.gov/news/news.php?feature=4457 (visited on 09/20/2017).

[3] L. Matthies. (2017). Titan aerial daughtercraft, [Online]. Available: https://www.nasa.gov/content/titan-aerial-daughtercraft (visited on 09/20/2017).

[4] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Battacharya, C. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3D complex environments," *Robotics and Automation Letters*, vol. 2, no. 3, 2017.

[5] A. Fragoso, C. Cigla, R. Brockers, and L. Matthies, "Dynamically feasible motion planning for micro air vehicles using an egocylinder," in *Conf. on Field and Service Robotics*, 2017.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, 1st ed. MIT Press, 2005.

[7] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV," in *IEEE Intl. Conf. on Robotics and Automation*, 2014.

[8] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2011.

[9] D. Dey, K. Shankar, S. Zeng, R. Mehta, M. Agcayazi, C. Eriksen, S. Daftry, M. Hebert, and J. Bagnell, "Vision and learning for deliberative monocular cluttered flight," in *Conf. on Field and Service Robotics*, 2015.

[10] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[11] W. Dryanovski, W. Morris, and J. Xiao, "Multi-volume occupancy grids: an efficient probabilistic 3D mapping model for micro aerial vehicles," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2010.

[12] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous obstacle avoidance and maneuvering on a vision-guided MAV using on-board processing," in *IEEE Intl. Conf. on Robotics and Automation*, 2011.

[13] F. Fraundorfer, L. Heng, D. Honegger, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor MAV," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2012.

[14] E. Bakolas and P. Tsiotras, "Multiresolution path planning via sector decompositions compatible to on-board sensor data," in *AIAA Guidance, Navigation, and Control Conf.*, 2008.

[15] H. Yu and R. Beard, "Vision-based local-level frame mapping and planning in spherical coordinates for miniature air vehicles," in *IEEE Conf. on Decision and Control*, 2011.

[16] A. Barry and R. Tedrake, "Pushbroom stereo for high-speed navigation in cluttered environments," in *IEEE Intl. Conf. on Robotics and Automation*, 2015.

[17] L. Matthies, R. Brockers, Y. Kuwata, and S. Weiss, "Stereo vision-based obstacle avoidance for micro air vehicles using disparity space," in *IEEE Intl. Conf. on Robotics and Automation*, 2014.

[18] M. Fleiss, J. Levine, P. Martin, and P. Rouchon, "Flatness and defect of nonlinear systems: introductory theory and examples," *Int. J. Control*, vol. 61, no. 6, 1995.

[19] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE Intl. Conf. on Robotics and Automation*, 2011.

[20] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Intl. Symp. of Robotics Research*, 2013.

[21] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *IEEE Intl. Conf. on Robotics and Automation*, 2016.

[22] E. Frazzoli, M. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Trans. Robotics*, vol. 21, no. 6, 2005.

[23] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, 1957.

[24] AASHTO, *Policy on Geometric Design of Highways and Streets*, 4th ed. American Association of State Highway and Transportation Officials, 2001.

[25] C. Powers, D. Mellinger, and V. Kumar, "Quadrotor kinematics and dynamics," in *Handbook of Unmanned Aerial Vehicles*, Springer Netherlands, 2015.

[26] A. Fragoso, L. Matthies, and R. Murray, "A fast motion planning representation for configuration flat robots with applications to micro air vehicles," in *American Control Conf.*, 2017.

[27] R. Brockers, A. Fragoso, B. Rothrock, C. Lee, and L. Matthies, "Vision-based obstacle avoidance for micro air vehicles using an egocylindrical depth map," in *Int. Symp. on Experimental Robotics*, 2016.

[28] V. Lumelsky and A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1-4, 1987.

[29] Y. Mulgaonkar, G. Cross, and V. Kumar, "Design of small, safe and robust quadrotor swarms," in *IEEE Intl. Conf. on Robotics and Automation*, 2015.

[30] M. Becker, R. Sampaio, S. Bouabdallah, V. Perrot, and R. Siegwart, "In flight collision avoidance for a mini-UAV robot based on onboard sensors," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 2, no. 12, 2012.

[31] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2012.

[32] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," in *IEEE Intl. Conf. on Robotics and Automation*, 2011.

[33] A. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Intl. Symp. of Robotics Research*, 2011.

[34] M. Nieuwenhuisen, D. Droeschel, M. Buel, and S. Behnke, "Obstacle detection and navigation planning for autonomous micro aerial vehicles," in *Intl. Conf. on Unmanned Aircraft Systems*, 2014.

[35] K. Sarabandi, M. Vahidpour, M. Moallem, and J. East, "Compact beam scanning 240 GHz radar for navigation and collision," in *Micro- and Nanotechnology Sensors, Systems, and Applications VIII*, 2011.

[36] D. Daftry, D. Dey, H. Sandhawalia, S. Zeng, A. Bagnell, and M. Hebert, "Semi-dense visual odometry for monocular navigation in cluttered environments," in *IEEE Intl. Conf. on Robotics and Automation Workshop on Recent Advances in Sensing and Actuation for Bioinspired Agile Flight*, 2015.

[37] J. Keshavan, G. Gremillion, H. Alvarez-Escobar, and J. Humbert, "Autonomous vision-based navigation of a quadrotors in corridor-like environments," *Micro Air Vehicles*, vol. 7, no. 2, 2015.

[38] S. Lavalle, *Planning Algorithms*, 1st ed. Cambridge University Press, 2006.

[39] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, 2013.

[40] O. Kahler, V. Prisacariu, J. Valentin, and D. Murray, "Hierarchical voxel block hashing for efficient integration of depth images," in *IEEE Intl. Conf. on Robotics and Automation*, 2016.

[41] L. Heng, D. Honegger, G. Lee, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous visual mapping and exploration with a micro aerial vehicle," *Journal of Field Robotics*, vol. 31, no. 4, 2014.

[42] M. Bajracharya, A. Howard, L. Matthies, B. Tang, and M. Turmon, "Autonomous off-road navigation with end-to-end learning for the LAGR program," *Journal of Field Robotics*, vol. 26, no. 1, 2009.

[43] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for MAV flight," in *IEEE Intl. Conf. on Robotics and Automation*, 2015.

[44] M. Otte, S. Richardson, J. Mulligan, and G. Grudic, "Path planning in image space for autonomous robot navigation in unstructured outdoor environments," *Journal of Field Robotics*, vol. 26, no. 2, 2009.

[45] R. Brockers, A. Fragoso, and L. Matthies, "Stereo vision-based obstacle avoidance for micro air vehicles using an egocylindrical image space representation," in *Micro- and Nanotechnology Sensors, Systems, and Applications VIII*, 2016.

[46] G. Dubey, S. Arora, and S. Scherer, "DROAN — disparity-space representation for obstacle avoidance," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2017.

[47] C. Cigla, R. Brockers, and L. Matthies, "Gaussian mixture models for temporal depth fusion," in *IEEE Winter Conf. on Applications of Comp. Vision*, 2017.

[48] I. Kamon, E. Rivlin, and E. Rimon, "Range-sensor based navigation in three dimensions," in *IEEE Intl. Conf. on Robotics and Automation*, 1999.

[49] S. Lavalle and J. Kuffner Jr., "Randomized kinodynamic planning," *Intl. Journal of Robotics Research*, vol. 20, no. 5, 2001.

[50] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for motion planning in high-dimensional configuration spaces," *Trans. on Robotics and Automation*, vol. 12, no. 4, 1996.

[51] J. Borenstein, "The vector field histogram — fast obstacle avoidance for mobile robots," *IEEE Trans. Robotics*, vol. 7, no. 3, 1991.

[52] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, 1959.

[53] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, 1968.

[54] V. Lumelsky, *Sensing, Intelligence, Motion: How Robots and Humans Move in an Unstructured World*. Wiley, 1970.

[55] I. Ulrich and J. Borenstein, "VFH+: reliable obstacle avoidance for fast mobile robots," in *IEEE Intl. Conf. on Robotics and Automation*, 1998.

[56] ——, "VFH*: local obstacle avoidance with look-ahead verification," in *IEEE Intl. Conf. on Robotics and Automation*, 2000.

[57] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Intl. Journal of Robotics Research*, vol. 30, no. 7, 2011.

[58] R. Bohlin and L. Kavraki, "Path planning using lazy prm," in *IEEE Intl. Conf. on Robotics and Automation*, 2000.

[59] F. Baldini, Bandyopadhyay, R. Foust, S. Chung, A. Rahmani, J. de la Croix, A. Bacula, C. Chilan, and F. Hadaegh, "Fast motion planning for agile space systems with multiple obstacles," in *AIAA/AAS Astrodynamics Specialist Conf.*, 2016.

[60] A. Paranjape, K. Meier, X. Shi, C. S., and S. Hutchison, "Motion primitives and 3D path planning for fast flight through a forest," *Intl. Journal of Robotics Research*, vol. 34, no. 3, 2015.

[61] M. Pivtoraiko, D. Mellinger, and V. Kumar, "Incremental micro-UAV motion replanning for exploring unknown environments," in *IEEE Intl. Conf. on Robotics and Automation*, 2013.

[62] A. Bryson and Y. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*, 1st ed. CRC Press, 1975.

[63] D. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 35, no. 7, 1990.

[64] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar, "Toward image based visual servoing for aerial grasping and perching," in *IEEE Intl. Conf. on Robotics and Automation*, 2014.

[65] M. Rathinam, "Differentially flat nonlinear control systems," PhD thesis, California Institute of Technology, 1997.

[66] T. Faulwasser, V. Hagenmeyer, and R. Findeisen, "Constrained reachability and trajectory generation for flat systems," *Automatica*, vol. 50, no. 4, 2014.

[67] H. Chitsaz and S. LaValle, "Time-optimal paths for a Dubins airplane," in *IEEE Conf. on Decision and Control*, 2008.

[68] K. Astrom and R. Murray, *Feedback Systems*. Princeton University Press, 2008.

[69] R. Courant and D. Hilbert, *Methods of Mathematical Physics*, 1st ed. Interscience, 1953.

[70] M. Muller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2011.

[71] M. Pope and M. Cutkosky, "Thrust-assisted perching and climbing for a bioinspired UAV," in *Conf. on Biomimetic and Biohybrid Systems*, 2016.

[72] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. Mueller, J. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The flight assembled architecture installation: cooperative construction with flying machines," *IEEE Control Systems*, vol. 34, no. 4, 2014.

[73] T. Nageli, C. Conte, A. Domahidi, M. Morari, and O. Hilliges, "Environment-independent formation flight for micro aerial vehicles," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2014.

[74] J. Leishman, *Principles of Helicopter Aerodynamics*, 1st ed. Cambridge University Press, 2000.

[75] B. Michini, J. Redding, N. Ure, M. Cutler, and J. How, "Design and flight testing of an autonomous variable-pitch quadrotor," in *IEEE Intl. Conf. on Robotics and Automation*, 2011.

[76] J. How, B. Behihke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, vol. 28, no. 2, 2008.

[77] Odroid. (2017). Odroid U3, [Online]. Available: `http://www.hardkernel.com/main/products/prdt_info.php?g_code=g138745696275` (visited on 09/20/2017).

[78] AscTec. (2017). Asctec Mastermind, [Online]. Available: `http://www.asctec.de/en/asctec-mastermind` (visited on 09/20/2017).

[79] ——, (2017). Asctec Autopilot, [Online]. Available: `http://wiki.asctec.de/display/AR/AscTec+AutoPilot` (visited on 09/20/2017).

[80] ROS. (2017). Ros, [Online]. Available: `www.ros.org` (visited on 09/20/2017).

[81] J. Engel, T. Schops, and D. Cremers, "LSD-SLAM: large-scale direct monocular SLAM," in *Euro. Conf. on Computer Vision*, 2014.

[82] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, "Monocular vision for long-term micro aerial vehicle state estimation: a compendium," *Journal of Field Robotics*, vol. 30, no. 5, 2013.

[83] J. Engel, J. Stuckler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, Sep. 2015.

[84] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *IEEE Intl. Conf. on Computer Vision*, 2013.

[85] A. Mourikis and S. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *IEEE Intl. Conf. on Robotics and Automation*, 2007.

[86] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE Intl. Conf. on Robotics and Automation*, 2011.

[87] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N. Dodgson, "Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid," in *Eur. Conf. on Comp. Vision*, 2010.

[88] A. Hosni, C. Rhemann, M. Bleyer, and M. Gelautz, "Temporally consistent disparity and optical flow via efficient spatio-temporal filtering," in *Pac. Rim Symp. on Image and Video Tech.*, 2011.

[89] C. Unger, E. Wahl, P. Sturm, and S. Ilic, "Probabilistic disparity fusion for real-time motion stereo," *Machine Vision and Applications*, vol. 25, 2011.

[90] C. Cigla, R. Brockers, and L. Matthies, "Image-based visual perception and representation for collision avoidance," in *IEEE Intl. Conf. on Computer Vision and Pattern Recognition, Embedded Vision Workshop*, 2017.

[91] C. Hane, C. Zach, J. Lim, A. Ranganathan, and M. Pollefeys, "Stereo depth map fusion for robot navigation," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2011.

[92] Odroid. (2017). Odroid XU4, [Online]. Available: `http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825` (visited on 09/20/2017).

[93] C. Forster, M. Pizzoli, and S. D., "SVO: fast semi-direct monocular visual odometry," in *IEEE Intl. Conf. on Robotics and Automation*, 2014.

[94] K. Kant and S. Zucker, "Toward efficient trajectory planning: the path-velocity decomposition," *Intl. Journal of Robotics Research*, vol. 5, no. 3, 1986.

[95] P. Firoini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Intl. Journal of Robotics Research*, vol. 17, no. 7, 1998.

[96] E. Weisstein. (2017). Wolfram mathworld: cone-sphere intersection, [Online]. Available: `http://mathworld.wolfram.com/Cone-SphereIntersection.html` (visited on 09/20/2017).

[97] A. Ramezani, S.-J. Chung, and S. Hutchison, "A biomimetic robotic platform to study flight specializations of bats," *Science Robotics*, vol. 2, no. 3, 2017.