Integrated Perception and Control at High Speed

by

Peter R. Florence

A.B., Princeton University (2012)
M.Phil, Cambridge University (2013)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2017

Author .....
Department of Electrical Engineering and Computer Science
January 31, 2017

Certified by .....
Russ Tedrake
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Theses

Integrated Perception and Control at High Speed

by

Peter R. Florence

Submitted to the Department of Electrical Engineering and Computer Science
on January 31, 2017, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

We present a method for robust high-speed quadrotor flight through unknown cluttered environments using integrated perception and control. Motivated by experiments in which the difficulty of accurate state estimation was a primary limitation on speed, our method forgoes maintaining a map in favor of using only instantaneous depth information in the local frame. This provides robustness in the presence of significant state estimate uncertainty. We compare the method against a benchmark approach using a simulated quadrotor race through a forest at high speeds in the presence of increasing state estimate noise. We then present hardware validation experiments in both indoor and outdoor environments, performing robust obstacle avoidance at speeds of up to 10 m/s, including sustained flight through a forest at 6 m/s. Finally, we add to the memoryless method, and develop a robust obstacle avoidance approach that uses memory without resorting to a maximum-likelihood mapping framework.

Thesis Supervisor: Russ Tedrake
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

There are many people to thank for helping me put together this work at MIT over the past couple of years.

First and foremost I would like to thank my advisor, Russ Tedrake. The list of things that I have picked up from Russ is countless and I thank him for his continued inspiring leadership of the group. In particular I consistently benefit from his drive to focus in on foundational, technical challenges, and his positive example he sets in the lab and in the research community.

I would also like to thank the members of the Robot Locomotion Group, for sharing their wisdom, talents, insights, critiques, laughs, and friendship. In particular, I'd like to thank Andy Barry for helping me settle in at MIT and in the lab, for his knack to do very much with very little, and for teaching me an insane amount of things. John Carter, too, has been tremendously generous with his deep programming talents and growing skills with a banjo. Lucas Manuelli has been a great partner in starting out the robotics quest, and I thank him for constantly teaching me a lot of math. Pat Marion has been magically helpful with his programming skills. Benoit Landry was also a great partner in the quest into the quadrotor world, and Ani Majumdar has also taught me a lot of math and robotics. Vincent Tjeng has also been a uniquely astute undergraduate to have in the lab. Everyone in the group is amazingly talented and it has been great to be in research conversations with all of them, including Aykut Satici, Hongkai Dai, Michael Posa, Robin Deits, Twan Koolen, Frank Permenter, Shen Shen, Geronimo Mirano, and Tao Pang.

It has also been a very valuable experience being a member of the DARPA FLA research team, partnering with the labs of Nick Roy, Jon How, and Emilio Frazzoli at MIT, and with Draper Laboratory. In particular I'd like to thank Brett Lopez, Nick Greene, Jake Ware, and Kris Frey for enlightening research discussions, and to the whole team for being supportive of my research, including Katherine Liu, Steve Paschall, Julius Rose, Rob Truax, Ted Steiner, Scott Rasmussen, Chris Wardman.

Of course I would like to thank my family! Mom and Dad thank you for putting

up with me being out on the East Coast again. TJ it has been great going through PhD world with you. Susanne thank you for everything. Huxley and Murphy, thanks for the licks.

Funding Acknowledgement

This work was supported by the DARPA Fast Lightweight Autonomy (FLA) program, HR0011-15-C-0110. Disclaimer: the views expressed in this proposal are not endorsed by the sponsor.

Contents

Preface

This work is organized in five chapters. Chapter 1 introduces the work, states the contributions, discusses the motivation of the robust, high speed navigation, and reviews related work. Chapter 2 describes the formulation of the novel approach in detail, and Chapter 3 presents hardware validation experiments. Chapter 4 extends the method to include memory, with Depth-Pose-Graph Planning. Chapter 5 briefly discusses future work.

Chapter 2 was originally published as

> Peter R. Florence, John Carter, and Russ Tedrake. Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps. In *Workshop on the Algorithmic Foundations of Robotics*, 2016.

The intent is to submit a journal version of this content together with the hardware experiments of Chapter 3.

Chapter 4 is also a candidate for publication soon, with some more experimental work.

Chapter 1

Introduction

The past few years have seen rapid progress in artificial intelligence and robotics. As of the year 2016, an artificial intelligence (AI) agent can beat the world's best human player in the game of Go [1], and autonomous cars are becoming increasingly accepted to be safer than human drivers [2]. Yet despite the advances we've made, robots still can't match humans in common skills of movement – our best robots cannot walk as well as a toddler, cannot manipulate objects with the dexterity of human hands, and cannot interact with the world in the rich ways humans do every day.

Agile and robust flight in an unknown environment is a prime example of where our best AI agents cannot match nature. Birds are often pointed to as a remarkable example of nature's dominance of this task. An even more direct comparison is offered by the recently emerged sport of quadcopter FPV (first person view) racing. Quadcopter FPV pilots use essentially the same hardware as autonomous quadcopters, but are given less data (only a monocular video stream) than a typical autonomous system. There is no doubt that the skills of the world's best pilots are spectacularly above the skills of any autonomous drone.

What in particular is hard about autonomous flight in cluttered environments? Uncertainty and lack of information are primary difficulties – the game of Go has perfect information, whereas an autonomous drone gets only a limited, noisy view of its world. There are other foundational robotics problems embedded as well: there is a mix of discrete problems *(should I turn left or right?)* as well as continuous prob-

lems *(how wide should I turn around this corner?)*. The difficulty of collecting data without crashing and the immense variability of data in navigating unknown environments make machine-learning-based methods difficult to apply. All of these problems are exacerbated by the small payloads available on lightweight UAVs (unmanned autonomous vehicles), with typically under 1 kg of payload available for sensing and computation.

Figure 1-1: Masters of agile flight in unknown environments: hawks and humans. (a) Red-tailed hawk, and (b) goshawk flying through forests. (c) Renowned quadcopter FPV pilot, Charpu, and (d) quadcopter FPV race through forest.

As has been a motivational theme of our research group for the past several years, agile and robust flight is a robotics problem that is *"hard for the right reasons"* [3]. Due to the inherent subproblems that are foundational to robotics, making advances on this overall problem can translate into broad applicability across robotics – from making autonomous cars increasingly safer, to increasing the dexterity of robotic manipulation, to making helpful robots in the home.

Chapter Organization

In this introductory chapter, the goal is to complement, rather than restate, the motivations and backgrounds as discussed in the later chapters. Section 1.1 presents

contributions, Section 1.2 discusses the motivation from a broad perspective, and Section 1.3 reviews some of the most relevant research in the field.

1.1 Contributions

This thesis develops novel methods combining typically separate perception, control, and state estimation considerations, and applies them to the high speed collision avoidance problem.

In particular, in Chapter 2, to address the limitations on performance that are observed due to state estimation difficulties, a planning-based obstacle avoidance method is developed that avoids a dependence on global position. Simulation experiments are presented in which a benchmark approach cannot provide robust collision avoidance at high speeds, while the presented method enables the quadrotor to navigate a simulated forest environment at 12 m/s even in the presence of significant state estimate noise. This is also the first work known to the author to describe stochastic receding horizon control with depth sensor data for a UAV. In Chapter 3, hardware validations of this method in both indoor warehouse and outdoor forest environments are also presented and analyzed, at speeds up to 10 m/s. These hardware results are among the fastest and most robust performance results achieved by a comparable vehicle. Chapter 4 extends the integrated perception and control approach to incorporate memory. The approach developed is the first known to evaluate a distribution of the robot's obstacle memory in obstacle avoidance decisions.

1.2 Motivation

1.2.1 Separation or Integration of Perception and Control

A hallmark of control theory is the separation principle. In many applied control tasks, from controlling chemical plants to stabilizing a quadcopter at a fixed point, following the separation principle has had broadly useful empirical applications. The principle states that a maximum likelihood estimator and feedback controller can be

computed separately, or as worded by Kalman in his 1960 paper, "Contributions to the theory of optimal control" [4]:

> *One may separate the problem of physical realization into two stages: (A) computation of the "best approximation" $\hat{x}(t_1)$ of the state from knowledge of $y(t)$ for $t \leq t_1$ and (B) computation of $u(t_1)$ given $\hat{x}(t_1)$.*

While the proof of the separation principle is straightforward for the simple case of linear time-invariant systems [5], it also applies to more general, including nonlinear, systems with certain technical conditions [6]. The principle, however, does not generally apply to uncertain systems with general cost functions – perhaps the simplest way to see this is with a small POMDP (Partially Observable Markov Decision Process) toy example, as shown in the figure below.

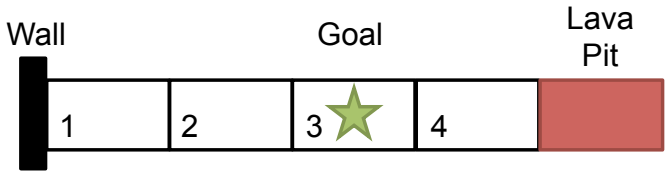


Figure 1-2: Toy example, inspired by Kaelbling et al. [7], of how a separation principle does not generally apply to POMDPs. In this example, the actions available are $WEST$ and $EAST$. Entering the lava pit is irrecoverable, whereas attempting to enter the wall causes the robot to remain in state 1. If the initial probability distribution between states $1, 2, 3, 4$ is $[0.3, 0.4, 0.0, 0.3]$, then the maximum likelihood state is state 2, and the desired action should be $EAST$. Clearly the preferred strategy, however, is to execute $WEST$ until the robot knows it must be in state 1, and then go to the goal.

The general problem of obstacle avoidance in an unknown world, with imperfect sensing subject to field of view (FOV) constraints and occlusions, exemplifies a system that does not fit the sufficient conditions of Kalman's principle. In practice, however, there has been a de facto trend of employing the separation principle in work in this area. For a number of reasons of convenience, including that they have somewhat separate research communities, planning and control are often separated from mapping and estimation.

On the opposite end of the spectrum, the close integration of perception and control has been explored in a variety of forms in robotics research and applications. In

visual servoing, for example, image coordinates rather than full state estimates are used to accomplish tasks [8]. Other work in manipulation with "end-to-end visuomotor" reinforcement learning control has suggested the advantages of allowing all raw vision data in the control loop, rather than derived estimates of object poses [9].

1.2.2 The Conventional Routes to Autonomous UAV Navigation: Map-Plan-Track and Reactive Approaches

Much of the work towards the goal of autonomous UAV flight in unknown environments can be categorized as using a "map-plan-track" type approach, in which mapping, planning, and trajectory-tracking control are run as separate processes. (This area of work is reviewed further in Section 1.3.2.) A primary barrier, however, to the success of these techniques in unknown environments is the difficulty of high-precision GPS-denied state estimation, particularly in regimes of fast, aggressive flight. When these methods are exposed to significant state estimate uncertainty, both mapping and tracking fail. Planning-heavy approaches also tend towards high latency, although offline- [3, 10] or online-computed [11, 12] libraries can enable low-latency response.

Given this status quo, one option to make progress is to focus on improving GPS-denied state estimation [13, 14], but in this thesis an alternate route is investigated. Our approach takes motivation from reactive control techniques, which have blazed their own trail separately from motion planning theory, and for a while now have provided impressive UAV obstacle avoidance capabilities. For example, work by Beyeler et al. [15] in 2009 achieved obstacle avoidance moving at approximately 14 m/s, and they touted their lack of explicit position dependence as a primary advantage of their approach. That said, there are a variety of limitations with reactive control techniques, including that they don't provide rigorous frameworks for reasoning about uncertainty.

The aim of our work was to provide a route to fast, robust flight that combines the best of both these worlds. In particular, from the world of reactive control, we want the property that we don't necessarily need to perform full state feedback – we

do not want to rely on high-precision state estimation. From motion planning theory, we employ state-space tools and reason about uncertainty.

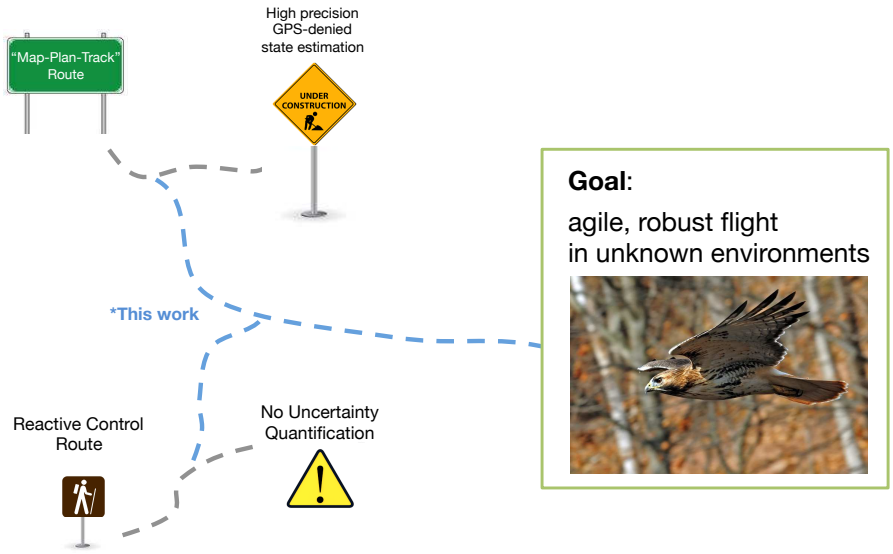
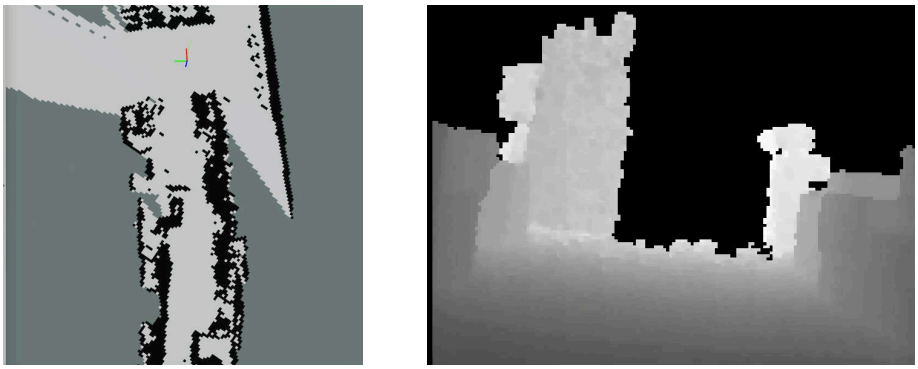


Figure 1-3: Depiction of the possible routes and limits towards our goal. Our work investigates combining inspiration from both traditional approaches.

1.2.3 Motivations from the Realities of Hardware

Our approach was closely motivated by experience with hardware. Our DARPA FLA[1] research team had previously implemented a map-plan-track type approach, and a primary barrier to higher vehicle speed and performance was the difficulty of state estimation in unknown environments, which could cause both mapping and tracking to fail. A realization upon inspecting raw flight data was that even though odometry errors made precise mapping difficult, raw depth images could be clean. An additional realization was provided by an examination of position and velocity estimation from a GPS-denied visual inertial odometry (VIO) state estimator. Empirically, we observe that the variance of position is higher than velocity. This makes sense, since variance increases for each integration of a noisy IMU.

[1]The DARPA Fast Lightweight Autonomy (FLA) program is a multi-research-organization effort with a particular emphasis on speed for UAVs in unknown, GPS-denied environments.



(a) (b)

Figure 1-4: Comparison of (a) skewed map data and (b) raw depth image data (ASUS Xtion structured-light sensor) from a flight at 5.5 m/s down a corridor.

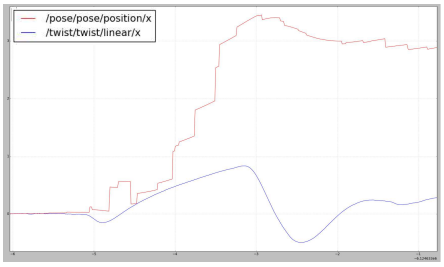


Figure 1-5: Raw data out of a visual inertial state estimator for position and velocity.

Additionally, an important property is that depth sensors measure obstacles in relative coordinates. Raw depth images are commonly available at a high rate (30-60 Hz) – a sufficient rate for making obstacle avoidance decisions. Other works have previously investigated planning-based methods in depth image space [16,17], but use trajectory-tracking controllers. A novel combination of both (i) using only current depth image information, and (ii) using model-predictive-control (MPC) rather than a nominal-plan tracking controller, offered the opportunity to eliminate an explicit dependence on global position. Velocity estimation is still required, but it is lower variance than position with VIO estimators, and our method is designed to handle a Gaussian belief space for velocity. Using only the instantaneous local frame, there is also no explicit dependence on yaw, which is well-known to be hard to estimate for quadrotors without GPS. In contrast to our method, there is a 1:1 correspondence between the hardest-to-estimate parts of state for a quadrotor, and the differentially-flat outputs that are commonly used for trajectory planning and tracking [18].

Additional motivation for this approach came from how an expert quadcopter FPV pilot flies. Expert pilots are able to navigate around obstacles with incredible agility, without ever being able to estimate their global position to within centimeters. Arguably the central components of successful FPV pilot flight are (i) an approximate sense of velocity, (ii) an approximate sense of how control actions affect dynamics, and (iii) fast reactions to sensory inputs. Our method shares these three characteristics, although it uses depth images rather than RGB images.

Figure 1-6: FPV quadcopter pilot Charpu flying through trees. With only monocular video available, expert pilots can navigate complex environments like these at remarkable speed. The inset in each image shows the hands of the pilot sending control inputs to an onboard attitude controller.

1.3 Related Work

First, we review the empirical state of the art for the specific problem of UAVs navigating unknown environments at high speeds. Beginning with the empirical state of the art forces recognition of *what actually works?* We then review the formulations and varieties of the two dominant categories of approaches, (i) map-plan-track, and (ii) reactive. Additionally, we review related areas of theory and practice.

1.3.1 Empirical State of the Art: UAVs Navigating Unknown Environments

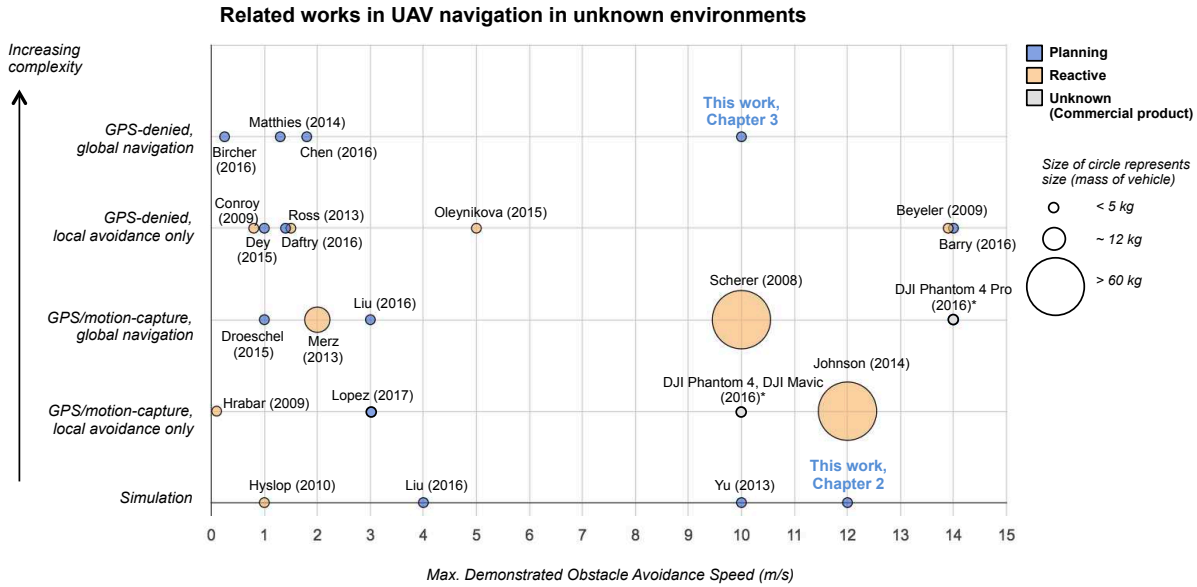


Figure 1-7: Comprehensive comparison of works in UAVs navigating unknown environments that meet the requirements stated below. Asterisk * denotes that the numbers are reported for manual-pilot obstacle assistance, and may be different for fully autonomous navigation.

A vast amount of work has been devoted towards this goal, but the list of works that have been empirically evaluated in *unknown environments* is a manageable list to evaluate in full. The figure above, Figure 1-7, presents a chart that to the best of the author's knowledge is a comprehensive compilation of works [3, 10–12, 15, 17, 19–32] that meet the following requirements: (i) must be demonstrated either in hardware with real perception or in simulation with real perception simulation, for a UAV platform navigating among obstacles in an unknown environment. An additional practical requirement (ii) is that the vehicle velocity must have been stated – there is a small list of works that meet the first requirement but do not provide any velocity numbers [33–38]. There is a much longer list of works that address subproblems but are demonstrated either in simulation or hardware with full obstacle knowledge, use a similar platform such as a ground vehicle instead of a UAV, or in some other way do not meet the requirements stated above. Some of these will be referred to in the

following sections.

A variety of careful considerations are present in Figure 1-7. All works have been categorized as either planning or reactive approaches. The definition of "planning" used was that the method must consider specific paths in configuration space, wheres "reactive" encompasses all approaches that do not. Some works [11, 19, 20, 28], including our own, employ a layered approach, in which a higher-level planner provides global guidance, but were categorized according to the local obstacle avoidance method. The y-axis is ordered in terms of increasing complexity: GPS-denied makes obstacle avoidance significantly harder, in part due to the variance of position estimation. For the x-axis, although speed is not a perfect measure of capabilities, it is clearly quantifiable, and often reported. To be fair to the respective authors, maximum obstacle avoidance speed was not necessarily the goal of each of these works. Perhaps better metrics for quadrotor "agility" are maximum roll angles and roll rates while navigating an unknown environment – the highest demonstrated among these works is probably Lopez et al. [12], with respectively $75.4deg$ and $695.7\frac{deg}{s}$. The most important concept that is not represented in the chart, and is perhaps our method's primary motivation, is robustness – unfortunately, data is not available to compare across separate studies.

Some conclusions are apparent from examining the works of the chart. Of all the planning methods included, ours is the only one that does not have an explicit dependence on position – every single other planning approach performs position-tracking control. Multiple works achieved obstacle avoidance results with reactive approaches at 10 m/s or above several years ago [15, 19], whereas only recently, Barry et al. was the first to achieve this result with a planning method, using stereo vision and a trajectory-library approach [3]. Additionally, as is highly relevant to the arguments regarding memory presented in Chapter 4, of the eight fastest hardware-proven planning methods, five of them use no memory of depth measurements [10–12, 17, 32], one uses only a two-second history of depth measurements [3], and one uses an exponentially decaying history of depth measurements [29]. Another observation is the rapid entrance of industry onto the scene – 2016 saw the first widely available consumer

products with these capabilities.

1.3.2 Map-Plan-Track Approaches

In the map-plan-track paradigm, each process is run separately: (i) producing a map as close as possible to full obstacle knowledge, (ii) receding-horizon planning, and (iii) trajectory-tracking feedback control along the nominal plan. Map-plan-track approaches are very sensible in that they build on a large body of motion planning theory [39]. There has been large amounts of impressive work using map-plan-track methods for UAVs navigating amongst obstacles with a prior map, specific obstacle locations, or other prior knowledge [13, 18, 40–45]. In regimes of good information, such as in motion capture rooms, these methods have work well.

Mapping

Mapping has been intensely studied for decades, and has many applications even with no controller in the loop – for example in virtual reality (VR) applications, or in surveying. The mapping problem is often simultaneously solved with the state estimation problem, known as SLAM (Simultaneous Localization and Mapping). For UAV navigation [3, 29, 31, 32, 44], an overwhelmingly common option is to use occupancy grids [46] as the map representation. A more detailed discussion of related work in mapping is provided in Chapter 4.

Planning

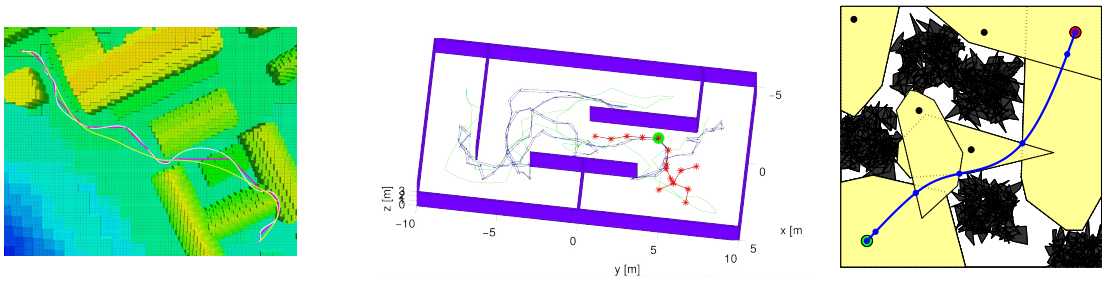


Figure 1-8: Examples of planning approaches used for UAVs, (a) A*, (b) RRT, (c) convex segmentation and mixed-integer optimization. Images from [30, 47, 48].

A few categories of planning approaches have been most prominently used for UAVs. First, (i) discrete graph search approaches such as Dijkstra's algorithm, A*, or D* operate on a graph of discretized space, and have been abundantly used for UAV platforms in at least part of an overall planning pipeline [11, 19, 20, 28, 31, 32, 35, 47]. Discrete search offers many convenient properties, but are limited by discretization effects, the difficulty of incorporating dynamic vehicle constraints, and the curse of dimensionality which makes sizable 3D environments difficult. Second, (ii) sampling-based planners, such as the classic PRM [49] or RRT [50], or preferably variants that offer asymptotic optimality guarantees such as RRT* [51] or FMT* [52] offer another route to planning through complex configuration spaces, scale better than discrete planning to higher dimensions, and have been commonly used for UAV navigation [17, 24, 42]. Methods based on using a library of motion primitives [3, 10, 29] are a form of sampling-based planning. Third, (iii) there are mixed-integer optimization planning variants [43, 48, 53, 54]. The idea of using convex free-space segmentation, rather than half-space constraints, was proposed by Deits. et al. [48] and has inspired methods that although do not use mixed-integer programming, use convex segmentation around an initial plan to provide constraints for trajectory optimizations through free space [31, 32].

Particularly for quadrotors, an abundantly popular tool has been optimizing polynomial trajectories $x(t), y(t), z(t), \psi(t)$ for the differentially-flat outputs (position and yaw) of a quadrotor. Optimization tools can easily minimize higher-order time-derivatives. This method was first developed for quadrotors by Mellinger et al. [18] in 2011, and produced amazingly acrobatic quadrotor flight in motion capture environments. To navigate among obstacles, polynomial optimization needs good constraints, which are often provided by some previous planning step – for which each of the three methods mentioned above have been used [37, 42, 47, 48, 54].

Tracking

Theory has studied for decades the problem of stabilizing a dynamical system to a time-invariant fixed point, or a trajectory through time [55]. Theory has transitioned

well to practice for UAVs, particularly for quadrotors, which since the influential work of Mellinger et al. [18] have been able to perform aggressive maneuvers using position and velocity feedback control. We do not review specific tracking methods in detail, but note that with a good position estimate (i.e., in motion capture) and a dynamically feasible trajectory, position-tracking for UAVs is a solved and proven technology. One option for quadrotors that is a nice application of optimal control theory is to time-varying LQR to directly come up with motor commands [43], but in practice inner-loop attitude controllers are often run on separate hardware closing a feedback loop at high rate (200+ Hz) with an IMU. For fixed-wings, time-varying LQR can supply good tracking performance [3, 56]. Recent work has developed impressive tracking controllers for tail-sitters [57]. The open problems in tracking control are in rigorously dealing with estimation uncertainty, particularly with nonlinear UAV models. Exciting recent work has gone into verifying controllers of such models, given bounds on state estimation and disturbances [56, 58].

1.3.3 Reactive Approaches

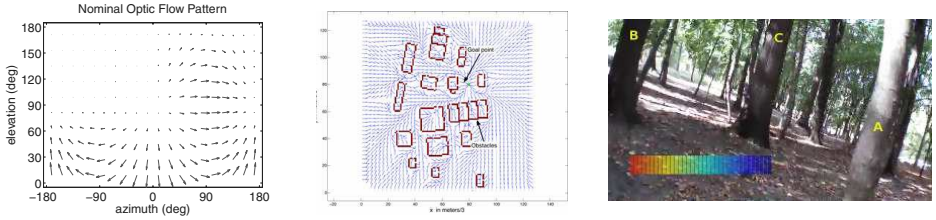


Figure 1-9: Examples of reactive obstacle avoidance approaches, (a) optic flow, (b) apf, (c) imitation learning. Images taken from Hyslop et al. [22], Scherer et al. [19], and Ross et al. [23].

Most reactive control techniques can be divided into four categories that have been used on UAVs: (i) optic flow [15, 20–22], (ii) artificial potential fields [19, 28] and closely-related approaches like the vector field histogram [36], (iii) imitation-learning approaches [23], and (iv) approaches that use hand-designed geometric relations but do not specifically perform state-space planning [25–27, 34]. It has been a common approach to use reactive approaches in layered formulations, with a higher-level planning

system [19, 20, 28]. A notable point is that optic flow methods have been performed at ∼4 kHz, orders of magnitude faster than common depth or RGB sensors [15]. It has been common to point to low-latency as an advantage of reactive approaches, but we believe an under-appreciated component of reactive approaches is that many of them have no explicit dependence on the full state estimate of the vehicle.

1.3.4 Planning Under Uncertainty

We briefly discuss the vast topic of planning under uncertainty. For a more comprehensive review of planning under uncertainty, we refer the reader to the pair of 2006 books by LaValle [39] and Thrun, Burgard, and Fox [59], which should be coupled with more up-to-date reviews of continuous motion planning under uncertainty [56] and POMDP solvers [60]. A general model of planning under uncertainty, POMDPs (Partially Observable Markov Decision Processes), is discussed more in Chapter 4, and Kaelbling et al. [7] provides a good introduction. POMDP solvers have been used for models of aircraft collision avoidance [61, 62]. A large variety of other works have made investigated continuous belief space planning. Common ideas have been to complement RRTs with notions of belief space [63–66], or use other forms of sampling-based algorithms with a notion of belief space [67, 68]. Others have used trajectory optimization tools such as direct transcription [69]. Quadrotors can be approximated with linear models, and so Linear-Gaussian models of uncertainty are useful – which have been widely used [66]. Two options for considering uncertainty are to consider an unbounded probability distribution of state, or consider worst-case bounds. Bounded uncertainty with nonlinear models has produced elegant formulations of verifiable planning [56, 70], where sequentially composed offline-computed funnels enables low-latency control decisions [56].

Path Collision Probability and Probabilistic Collision Detection

A component of planning under uncertainty may be approximating the collision probability of a path through configuration space. A good review of this topic is provided

by Janson et al. [71], who propose smart sampling strategies for making Monte Carlo calculations more efficient, but also review time-sampling methods. In particular, Patil et al. provide a conditional multiplicative method that is empirically the best among time-sampling methods [72]. Path collision probability approximations have been coupled with RRT planners [73]. Instead of entire paths, the concept of probabilistic collision detection between point clouds or meshes [74] has been studied and used in planning systems [75].

Chance Constrained Optimization

One approach to planning under uncertainty is chance constrained optimization, which has roots in the 1950s in operations research [76], and has been well developed in the context of robotic motion planning by a series of theoretical works by Blackmore et al. using mixed-integer optimization [77–79]. Related to these mixed-integer optimization methods, the concept of chance-constraining an RRT planner has been explored in a variety of works [80–82].

Planning with Limited Field of View (FOV) and Occlusions

The concept of planning in unknown environments where the assumption is made that unknown space is occupied is a common assumption. The concept of Inevitable Collision States (ICS) by Fraichard et al. well encompasses this idea [83]. Lesperance et al. provides a nice framework for thinking about a hierarchy of safety in motion planning, generalizing previous ideas about the assumptions of static or dynamic environments [84]. That said, the full implications of efficiently planning with limited field of views and occlusions of available hardware sensor data like depth images, is probably an underappreciated concept, although has been addressed in some works [11, 17].

1.3.5 Local Frame and Depth Image Space Planning

The benefits of performing obstacle avoidance in a local frame, rather than a global frame, have been well appreciated by many others besides us, and are often pointed to in studies of autonomous navigation [16, 17, 24, 85]. Moore et al. offers a formal evaluation of the benefits of maintaining separate state estimates for local frame and global frame planning [86]. A particular form of planning in the local frame is planning directly in depth image space, which has been explored in a number of works [16, 17].

1.3.6 Motion Primitive Libraries

The concept of a library of motion primitives is a widely used tactic for robotics, including UAVs navigating among obstacles [3, 10, 29, 45, 56, 87]. In practice, there are a variety of practical benefits of motion primitive libraries. For one, primitives can be verified ahead of time to be sensical. Additionally, motion primitives offer a way to avoid non-convex continuous optimization, by simply choosing the best among a discrete set. A nice theoretical framework is provided by Frazzoli et al. with their Maneuver Automaton [88]. Often libraries are computed offline [3,10,29,56], for which the advantage is allowing rigorous offline verification [56], or can be computed online for simple models [11, 12], which allows each primitive to have an initial condition that is exactly the current estimate. Without this latter property, offline libraries require careful thought about transitioning between motions [56]. Often, the choice of motion primitives is done by hand and not subjected to rigorous analysis, although some work has looked into optimization of motion primitive libraries [89].

1.3.7 Depth Sensor Hardware

The capabilities of available depth sensors has rapidly increased in recent years, and merits discussion. As of 2015, Intel's release of the RealSense R200 provided a global shutter stereo pair that can provide 480×360 resolution depth images at 60 Hz, weighs only 34 grams, costs only $\sim$ \$100, and can provide a range of 15-25 meters in high-texture, well-lit scenarios such as a forest. These specifications were simply not

commercially available only a couple of years ago. With its IR CCDs, however, in our hardware testing the RealSense is highly sensitive to material texture, and is blind to low-texture surfaces, like the sides of buildings. For inside applications, the Microsoft Kinect and ASUS Xtion sensors provide depth sensors with better robustness to a variety of surface textures. Outside of vision sensors, the year 2016 saw the first 3D lidar weighing under 600 grams with the Velodyne VLP Puck Lite, although the $\sim \$10,000$ cost is difficult to swallow for robots deliberately flying towards trees. The year 2016 also saw the emergence of radar sensors weighing approximately 1 kg, with range on the order of a kilometer. It is certainly an exciting time for the development of hardware, which are critical for geometric planning algorithms.

1.3.8 Obstacle Avoidance and Navigation Without Depth Sensors

In parallel to the growing capabilities of depth sensors, there is also growing interest in the use of non-geometric data for obstacle avoidance, including for mobile robots navigating unknown environments. Ross et al. was probably the first to demonstrate this capability for UAVs, with their imitation-learning approach [23]. Supervised learning of classifying paths in forests has enabled UAVs to follow paths in forests using only RGB data [90]. The idea of training to navigate using vision data in simulation is a commonly talked about idea in the research community, although has not been notably demonstrated on UAV hardware. Modern autonomous vehicles perform a variety of vision-based classifications in order to navigate their world.

Chapter 2

Evaluating Collision Avoidance Maneuvers Without Maps

2.1 Introduction

A primary challenge in improving robot performance is to increase robustness in regimes of fast motion, proximity to obstacles, and significant difficulty of estimating state. A robotics platform that is at the center of all of these challenges is a UAV navigating quickly through unknown, cluttered environments. Although compelling progress has been made [3, 10, 32, 91], the goal of autonomous, robust, agile flight into unknown environments remains an open problem.

In this paper, we present an integrated approach for perception and control, which we apply to the high-speed collision avoidance problem. Our approach departs from the paradigm of building maps, optimizing trajectories, and tracking trajectories. Central to the approach is considering routes to achieve control objectives (fly fast, and don't crash into obstacles) and taking advantage of model-based state-space without relying on full-state feedback.

Our approach is directly motivated by the success of reactive control that is "straight from sensors to control input" but uses tools from more rigorous state space control. We show that in order to get the performance of a motion planning system, the robot doesn't need to build a map, doesn't need precise estimates of its full state,

and doesn't need to heavily optimize trajectories.

A key insight we explore in this paper is that we can both estimate the probability of collision for any action without building a locally consistent map, and execute that action without the use of position-control feedback. The basics steps of our method are: evaluate maneuvers probabilistically for collision and impose field of view constraints, choose a maneuver based on an unconstrained objective combining collision avoidance and navigation, and execute this at high rate with a model-predictive control type approach. This method offers a mapless collision avoidance approach that does not depend on position, rigorously considers robustness, is amenable to low-latency implementations, and integrates seamlessly with arbitrary navigation objectives. We note, however, that the mapless method cannot escape dead-ends by itself without a layered global planner.

Our primary contribution is the novel synthesis of our approach combining typically separate perception, control, and state estimation considerations. This synthesis is implemented for robustness at speed by a combination of: local frame estimation of path collision probabilities that considers field of view (FOV) constraints, motion primitives defined in the local frame, acceleration by spatial partitioning, and high-rate robust model-predictive control that doesn't depend on trajectory-tracking. This is also the first paper known to the authors to describe stochastic receding horizon control with depth sensor data for a UAV. Additionally, we present simulation experiments in which a benchmark approach cannot provide robust collision avoidance at high speeds, while our method enables the quadrotor to navigate a simulated forest environment at 12 m/s even in the presence of significant state estimate noise.

2.2 Related Work

The close integration of perception and control, where the realities of perceptual information inform the control approach, is a concept of active interest in robotics. Visual servoing methods for robotic manipulation [8], for example, are an application where control is designed to work with partial information (relative positions in image

space) rather than full-state feedback.

In the application area of UAV navigation in unknown environments, the predominant approach is to instead impose the separation principle between perception and control, and separately build a map, plan an optimal trajectory in that map, and execute trajectory-tracking feedback control along the nominal plan. In this map-plan-track paradigm, the goal is to produce a map as close as possible to full obstacle knowledge and produce highly accurate estimates of full state. These methods work well in regimes of good information, such as motion capture rooms with pre-prescribed obstacle locations. They are particularly fragile, however, when exposed to significant state estimate uncertainty, causing mapping and tracking to fail. Planning-heavy approaches also tend towards high latency, although offline-computed libraries enable low-latency response [3, 10].

A different approach to UAV navigation is offered by reactive control, which has achieved some of the most impressive obstacle avoidance results demonstrated to date [15,19,23]. Three primary types of reactive approaches have shown success: optic flow methods [15, 21, 22], artificial potential fields [19, 28], and imitation learning [23]. Reactive methods by definition do not fit into the map-plan-track paradigm since they do not plan a time-sequence of states into the future, but are also generally characterized by not performing full-state feedback.

In that our method neither builds a map nor executes trajectory-tracking control, it departs from the map-plan-track paradigm. In that it does not perform position-control feedback, it is more similar to the mentioned reactive methods, yet it does plan states in the local frame into the future and reason about state-space uncertainty, which does not fit the definition of a reactive method.

The theory of motion planning under uncertainty has been well studied, at least in the domain of full obstacle knowledge. One approach is that of chance-constrained optimization [79, 80, 92, 93], in which the probability of collision at any time is upper-bounded as a constraint in an optimization. In the planning portion of our approach we use a variant where collision avoidance is included in the objective, not as a constraint, and we estimate collision probabilities for entire paths, then choose among

a finite library. An important component of this approach requires path collision probability estimation, which has been well studied [71].

Several other works are notably related to various components of our integrated approach. One related method for online stochastic receding-horizon control is that of "funnel" computation and sequential composition [56, 94, 95], which notably can handle nonlinear models. The focus of those works, however, is not on integrated perception and control considerations, as ours is here. A somewhat related work is by Matthies et al. [17] since it presents field-of-view-limited planning with depth image information for collision avoidance, but their approach is a map-plan-track approach, and doesn't consider uncertainty. Probabilistic collision detection in point clouds has been studied [74] and integrated with sampling-based motion-planners [75], but not to our knowledge has been applied to the collision avoidance problem with field-of-view constraints. Another complementary approach aims to learn, through supervised training in simulation, collision probabilities outside of conservative field of view approximations [96].

2.3 Generalized Formulation for Collision Avoidance

First, we consider the problem of estimating the probability of collision for a time-varying distribution of configurations using only instantaneous depth information. We then present approximation methods that enable fast computation for collision avoidance at high speeds. Additionally, we discuss the use of spatial partitioning data structures and the incorporation of global navigation objectives. This section is generalized to allow for application to an arbitrary robot. In the next section, a particular implementation for a quadrotor is presented.

2.3.1 Evaluating Collision Probabilities from Instantaneous Depth Information

We wish to evaluate the probability of collision for:

$$P\big(\text{Collision during } t \in [0, t_f] \mid \mathbf{D}, p_t(\mathbf{q})\big) \tag{2.1}$$

where $p_t(\mathbf{q})$ is the time-varying distribution of configuration, t_f is the final time, and $\mathbf{D}$ is a vector of depth sensor returns $[\mathbf{d}_0, ..., \mathbf{d}_n]$. This probability cannot be calculated with certainty, due to the large amount of unknown space $\mathcal{U} \subset \mathbb{R}^3$ caused by occlusions and the finite FOV (field of view) of the depth sensor. Each depth return corresponds to an occupied frustum $\mathcal{F}_{\mathbf{d}_j} \subset \mathbb{R}^3$ whose volume is defined by the image resolution, depth return distance, and sensor discretization. Together these occupied frustums comprise the known occupied subset of space, $\mathcal{O}_{known} = \bigcup_j \mathcal{F}_{\mathbf{d}_j}, \mathcal{O}_{known} \subset \mathbb{R}^3$. Each depth return also creates a portion of unknown space $\mathcal{F}_{(\text{occluded by } \mathbf{d}_j)} \subset \mathcal{U}$ which is a frustum that joins the unknown space at the sensor horizon. For handling the FOV constraints, the conservative route is to make the assumption that all unknown space $\mathcal{U}$ is occupied ($\mathcal{U} \cup \mathcal{O}_{known} = \mathcal{O}$), which provides a mapping from $\mathbf{D} \rightarrow \mathcal{O}$ that is strictly conservative.

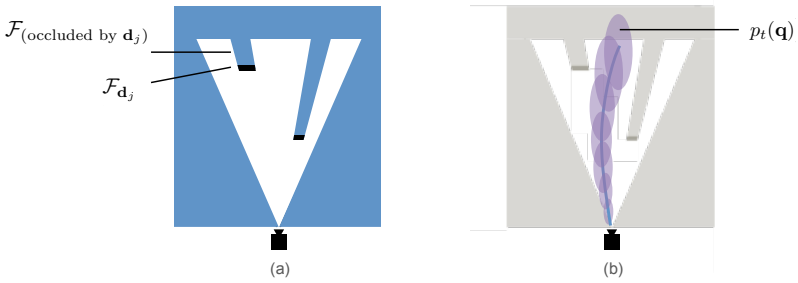


Figure 2-1: Depictions of (a) depth measurements (black) and conservative assumption of unknown space as occupied (blue), and (b) time-varying distribution of configuration (purple).

At any given point in time and given the distribution $p_t(\cdot)$ over robot configuration $\mathbf{q}$, the probability of collision is obtained by the probability that the robot is in

collision with any of the sensor returns or occupies any unknown space:

$$P\big(\text{Collision, } p_{t_i}(\mathbf{q})|\ \mathcal{O}_{known}, \mathcal{U}\big) = P\big(\mathbf{q}(t_i) \in \{\mathcal{O}_{known} \text{ or } \mathcal{U}\}\big) \tag{2.2}$$

Note that the probabilities are not disjoint, since for any non-zero-volume robot, a given configuration can be in collision with multiple frustums, occupied or unknown. To evaluate this probability, an integral over all possible configurations must be integrated. Even given a solution to this integral, however, this only provides an evaluation of one possible distribution of configuration at some future time, and hence the probability of collision for the time-varying distribution of configuration is still difficult to evaluate, given that all future positions in time are dependent on previous positions in time. One route to estimating this probability is through Monte Carlo simulation, but approximations offer computationally efficient routes. Although the literature does not typically account for FOV constraints, a good review of available options for estimating path collision probabilities with full obstacle knowledge is included in a recent paper by Janson et al. [71].

Additionally, even with the conservative assumption, the form of $\mathcal{U}$ (large subsets of space) is of a different form than $\mathcal{O}_{known}$ (small frustums). Our current formulation addresses this by converting $\mathcal{O}_{known}$ into a point cloud and evaluating the probability distribution $p_t(\mathbf{q})$ at these points, whereas for $\mathcal{U}$ we perform a binary evaluation of the mean of $p_t(\mathbf{q})$ entering $\mathcal{U}$. Future work could evaluate both of these probabilities more rigorously by integrating the probability distribution $p_t(\mathbf{q})$ over the volumes of both $\mathcal{O}_{known}$ and $\mathcal{U}$, at additional computational cost.

2.3.2 Fast Approximation of Maneuver Collision Probabilities

Given the goal to evaluate collision probabilities in real time for the purpose of collision avoidance, some approximations are in order. Although these are significantly simplifying assumptions, the simulation results presented in this paper suggest that even these approximations offer a significant improvement over deterministically collision-checking trajectories. We consider maneuvers of the form: $\mathcal{M} =$

$\{\mathbf{u}(t), p_t(\mathbf{q})\}$, i.e. control inputs as a function of time $\mathbf{u}(t)$ that produce a time-varying distribution of configurations $p_t(\mathbf{q})$. Our choice of open-loop maneuvers is a choice that represents our control decision to not depend on position-control feedback.

For estimating the probability of collision for the entire maneuver we use an independence approximation. Future positions are sampled in time, and the maneuver's probability of collision is approximated as the subtraction from unity of the product of the no-collision probabilities at each sampled time t_i:

$$P\big(\text{Collision}, p_t(\mathbf{q})\big) \approx 1 - \prod_{i=1}^{n_t} \big[1 - P(\text{Collision}, p_{t_i}(\mathbf{q}))\big] \tag{2.3}$$

For the evaluation of the no-collision probabilites at each time t_i, we assign a no-collision probability of 0 (definite collision) if the mean of $p_{t_i}(\mathbf{q})$ is in $\mathcal{U}$, and otherwise evaluate the probability of collision with the point cloud. Evaluating only the mean in $\mathcal{U}$ is a large oversimplification, but avoids integrating over many small occluded frustums:

$$\big[1 - P(\text{Collision}, p_{t_i}(\mathbf{q}))\big] = \begin{cases} 0, & \text{if } \mu(p_{t_i}) \in \mathcal{U} \\ \prod_{j=1}^{n_d} \big[1 - P(\text{Collision}, p_{t_i}(\mathbf{q}), \mathbf{d}_j)\big], & \text{otherwise} \end{cases} \tag{2.4}$$

Checking if $\mu(p_{t_i}) \in \mathcal{U}$ can be done by a projective transform into depth image space, and checking if the projection is either out of bounds of the depth image (outside FOV), or a depth return at that pixel has less depth (occluded). If not in $\mathcal{U}$, the probability of collision with $\mathcal{O}_{known}$ is approximated by an additional independence approximation: each collision with all n_d depth returns is assumed an independent probability event. To evaluate each event $P(\text{Collision}, p_{t_i}(\mathbf{q}), \mathbf{d}_j)$ above, we must choose a dynamic model with uncertainty. Thus far, the discussion has been generalizable to any model. In Section 2.4 we describe how we evaluate this term for a simplified model with Gaussian noise.

Naively, the complexity of the computation above is $O(n_{\mathcal{M}} \times n_t \times n_{\mathbf{d}})$. Even for a "low-resolution" depth image, the number of depth points can be high, for example a 160 x 120 image is $n_{\mathbf{d},\text{Total}} = 19,200$ points. Only the closest depth returns to

the mean of the robot's distribution, however, will have the highest probability of impact, and this additionally offers a route to lower computational complexity. Thus, rather than evaluate Equation 4.9 for all depth returns, we query only the closest $n_d < n_{\mathbf{d},\text{Total}}$ points with a k-d-tree.

In contrast to deterministic collision checking, collision probability approximation significantly benefits from three-dimensional spatial partitioning as opposed to operating directly on the depth image. This is because with the probabilistic collision checking, we care about "long-tails" of the robot position distribution, rather than just deterministically collision-checking the mean. To deterministically collision check, there is no faster way than using the raw depth image [17], but in order to consider long-tail positions in the direct depth image method, a large block of pixels needs to be checked. The depth image structure provides information about proximity in two dimensions (neighboring pixels), but not the third (depth). We also note, however, that since the direct depth image method requires no building of a new data structure, highly parallelized implementations may tip computational time in its favor (as opposed to sequentially building a k-d-tree, then searching it).

Briefly, we analyze the limitations of the approximation accuracy. In the context of full obstacle knowledge, the independence approximation over time has been shown to provide overly conservative estimates of collision probability [71]. Additionally, the independence approximation between depth returns contributes to more overestimation, and picking only one point from each cluster has been recommended to reduce this overestimation [74]. In our method, the FOV constraints contribute even more to over-conservatism, but there is not available information to improve this approximation without adding risk going into the unknown. Learned priors, however, can intelligently minimize this risk [96]. We note that with our unconstrained formulation, it is the relative differences between maneuver collision probabilities (see Figure 2-4b), not their absolute scale, that impacts control decisions.

At additional computational cost, additional accuracy could be achieved through Monte Carlo (MC) evaluation, whereby randomly sampled trajectories are deterministically collision-checked and the proportion of collision-free trajectories is the collision

probability. In the limit of infinite samples the probability is exact, but the computational cost is approximately $n_{MC} \times T_D$, where T_D is the time to deterministically collision-check, and n_{MC} is the number of samples. As we show in Table 1 (Section 6), deterministic collision-checking takes approximately the same amount of time as our independence approximation evaluation. Hence, naive MC evaluation is slower than our method by approximately the factor n_{MC}. Smart MC sampling strategies have been demonstrated to enable path collision probability approximations on the order of seconds for reasonable models [71], but our requirement is a few orders of magnitude faster (milliseconds) to replan at the rate of depth image information (30-150 hz).

2.3.3 Integrating Reactive and Navigation Objectives

A benefit of the probabilistic maneuver evaluation approach is that it naturally offers a mathematical formulation that integrates reactive-type obstacle avoidance with arbitrary navigation objectives. Whereas other "layered" formulations might involve designed weightings of reactive and planning objectives, the probabilistic formulation composes the expectation of the reward, $\mathbb{E}\big[R\big]$. Given some global navigation function that is capable of evaluating a reward $R_{nav}(\mathcal{M}_i)$ for a given maneuver, the expected reward is:

$$\mathbb{E}\big[R(\mathcal{M}_i)\big] = P(\text{No Collision}, \mathcal{M}_i) R_{nav}(\mathcal{M}_i) + P(\text{Collision}, \mathcal{M}_i) R_{collision} \qquad (2.5)$$

As we show in the simulation experiments, $R_{nav}(\mathcal{M}_i)$ may not even need to consider obstacles, and collision avoidance can still be achieved. The global navigation function can be, for example, just Euclidean progress to the global goal for environments with only convex obstacles, or for environments with dead-ends could for example be a cost-to-go using Dijkstra's algorithm (Figure 2-3a). A key point is that with the instantaneous mapless approach handling collision avoidance, $R_{nav}(\mathcal{M}_i)$ can be naive, and/or slow, although a good $R_{nav}(\mathcal{M}_i)$ is only a benefit. One parameter that must be chosen, and can be tuned up/down for less/more aggressive movement around

obstacles, is the cost (negative reward) of collision, $R_{collision}$,

Given a library of maneuvers, the optimal maneuver $\mathcal{M}^*$ is then chosen as:

$$\mathcal{M}^* = \underset{i}{\text{argmax}} \ \ \mathbb{E}\big[R(\mathcal{M}_i)\big] \tag{2.6}$$

2.4 Implementation for High Speed Quadrotor Flight

The formulation presented above is generalizable for different robot models and for evaluating different types of discrete action libraries. In this section we present a specific implementation for high-speed quadrotor control.

2.4.1 High-Rate Replanning with a Motion Primitive Library

We use an approach similar to a traditional trajectory library, except our library is generated online based on a simplified dynamical model. In the sense that a model is used for real-time control, and we use no trajectory-tracking controller, this is MPC (Model Predictive Control), but since we perform no continuous optimization but rather just select from a discrete library, this is a motion primitive library approach. This high-rate replanning with no trajectory-tracking controller offers a route to controlling collision avoidance without a position estimate. Since the uncertainty of the maneuvers is considered open-loop, this can be categorized as OLRHC (open-loop receding horizon control). Another control approach is to "shrink" the future uncertainty with a feedback controller [56, 66, 94], but this assumes that a reasonable position estimate will be available. It is crucial to our method that we do not shrink the uncertainty in this way, since this enables sensible avoidance decisions and control without ever needing a position estimate.

2.4.2 Dynamical Model and Propagating Uncertainty

To build intuition of our simple quadrotor model, we first describe the basic version of a constant-input double-integrator (constant-acceleration point-mass) modeled around the attitude controller. This version approximates the quadrotor as a

point-mass capable of instantaneously producing an acceleration vector of magnitude $||\mathbf{a}|| \leq a_{max}$ in any direction. Together with gravitational acceleration, this defines the achievable linear accelerations. This model is applied with the inner-loop attitude and thrust controller in feedback, as depicted in Figure 2-2. Given a desired acceleration $\mathbf{a}_i$, geometry defines the mapping to {roll, pitch thrust} required to produce such an acceleration, given any yaw.

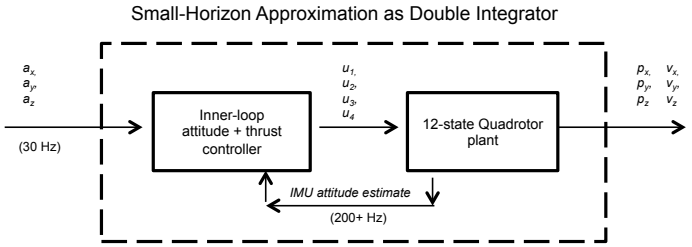


Figure 2-2: Dynamics approximation considered: the quadrotor is modeled in feedback with the inner loop attitude and thrust controller.

A motivating factor for this model is that the overwhelmingly ubiquitous implementation for quadrotor control involves a high-rate (∼200+ Hz) inner-loop attitude and thrust controller. The desirability of quickly closing a PID or similar loop around the IMU makes this an attractive control design choice.

The only source of uncertainty we consider is the state estimate. In particular, since the quadrotor's initial position is by definition the origin in the local frame, we only consider uncertainty in the velocity estimate. We use Gaussian noise for the initial linear velocity estimate $\mathbf{v}_0 \sim \mathcal{N}(\mathbf{v}_{0,\mu}, \Sigma_{v_0})$ which gets propagated through the linear model. We use the notation $\mathbf{p} \in \mathbb{R}^3$ to refer to the configuration since it is just position (point-mass is rotation-invariant). Accordingly we have:

$$\mathbf{p}_i(t) \sim \mathcal{N}\left(\frac{1}{2}\mathbf{a}_i t^2 + \mathbf{v}_{0,\mu} t,\ t^2 \Sigma_{v_0}\right) \tag{2.7}$$

$$\text{for maneuver } \mathcal{M}_i = \{\mathbf{a}_i, \mathbf{p}_i(t)\}, t \in [0, t_f]$$

where $\mathbf{p}_i(t)$ is a random variable defining the distribution referred to as $p_t(\mathbf{q})$ in Section 2.3. The chosen acceleration $\mathbf{a}_i$ defines the maneuver $\mathcal{M}_i$.

Extension to Piecewise Triple-Double Integrator Model

The limitations of the constant-acceleration model are clear, however: it does not consider attitude dynamics, even though they are fast (∼100-200 ms to switch between extremes of roll/pitch) compared to linear dynamics. It is preferable to have a model that does include attitude dynamics: for example, the initial roll of the vehicle should affect "turn-left-or-right" obstacle-dodging decisions.

Accordingly, we use a triple integrator for the first segment, and a double integrator for the remaining ("triple-double" integrator for short). Each maneuver $\mathcal{M}_i$ is still defined uniquely by $\mathbf{a}_i$, but during $t \in [0, t_{jf}]$, we use a jerk $\mathbf{j}_i$ that linearly interpolates from the initial acceleration $\mathbf{a}_0$ to the desired acceleration:

$$\mathbf{j}_i = \frac{\mathbf{a}_i - \mathbf{a}_0}{t_{jf}} \tag{2.8}$$

During the initial constant-jerk $t \in [0, t_{jf}]$ period, this gives

$$\mathbf{p}_i(t) \sim \mathcal{N}\left(\frac{1}{6}\mathbf{j}_i t^3 + \frac{1}{2}\mathbf{a}_0 t^2 + \mathbf{v}_{0,\mu} t, t^2 \Sigma_{v_0}\right) \quad \forall t \in [0, t_{jf}] \tag{2.9}$$

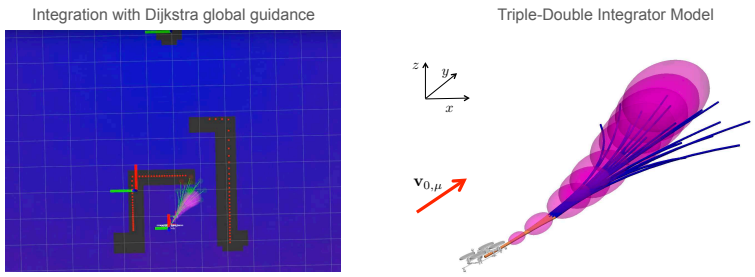
and for $t \in [t_{jf}, t_f]$ the double integrator model (Equation 4.7) is used with the appropriate forward-propagation of position and velocity. Note that for the constant-jerk portion, an initial acceleration estimate, $\mathbf{a}_0$ is required. We assume this to be a deterministic estimate. Since roll, pitch, and thrust are more easily estimated than linear velocities, this is a reasonable assumption.

The maneuvers produced by this piecewise triple-double integrator retain the properties of being closed-form for any future $t \in [0, t_f]$, of being linear with Gaussian noise, and cheap to evaluate. Although the actual attitude dynamics are nonlinear, a linear approximation of the acceleration dynamics during the constant-jerk period is an improved model over the constant-acceleration-only model. We approximate t_{jf} as 200 ms for our quadrotor.

2.4.3 Maneuver Library and Attitude-Thrust Setpoint Control

We use a finite maneuver library (Figure 2-3b), where the maneuvers are determined by a set of desired accelerations $\mathbf{a}_i$ for the piecewise triple-double integrator. Our method is compatible for a 3D library, but for the purposes of the simulation comparison against a global-planning 2D method in the next section, we use a library constrained to a single altitude plane. To build a suitable discrete set of maneuvers, we approximate the maximum horizontal acceleration and sample over possible horizontal accelerations around a circle in the horizontal plane. The max horizontal acceleration is approximated as the maximum thrust vector (T_{max}) angled just enough to compensate for gravity: $a_{max} = \frac{\sqrt{T_{max}^2+(mg)^2}}{m}$. By sampling both over horizontal accelerations with just a few discretizations (for example, $[a_{max}, 0.6a_{max}, 0.3 * a_{max}]$) and just 8 evenly spaced θ over $[0, 2\pi]$, this yields a useful set in the horizontal plane. We also add a $[0, 0, 0]$ acceleration option, for 25 maneuvers total in the plane, and use $t_f = 1.0$ seconds.

Executing the chosen maneuver is achieved by commanding a desired roll and pitch to the attitude controller. For this 2D-plane implementation, a PID loop on z-position maintains desired altitude by regulating thrust. We allow for slow yawing at 90 degrees per second towards the direction $\mathbf{p}(t_f) - \mathbf{p}_0$, which in practice has little effect on the linear model and allows for slow yawing around trees.



(a) (b)

Figure 2-3: (a) Visualization of integrating Dijkstra global guidance, where $\mathcal{R}_{nav}$ is the cost-to-go (blue is lower, purple is higher) of the final maneuver position. (b) Visualization of the piecewise triple-double integrator maneuver library. The library of maneuvers is shown with a positive x, positive y initial velocity $\mathbf{v}_{\mu,0}$, and the 1-σ of the Gaussian distribution is shown for one of the maneuvers. The $t_{jf} = 200$ ms constant-jerk period shown in orange. Note that due to the initial roll-left of the vehicle, it can more easily turn left than right.

2.4.4 Evaluation of Collision Probability and Global Navigation

Each maneuver is sampled at n_t positions (we use $n_t = 20$), for a total of 500 positions to be evaluated in our $n_{\mathcal{M}} = 25$ library. To allow for speeds past 10 m/s, given $t_f = 1.0$ s, we do not consider positions beyond our simulated depth image horizon of 10 meters to be in collision. All mean robot positions are evaluated for n_d nearest neighbors in the k-d-tree. In practice we have found success with $n_d = 1$, although larger n_d is still fast enough for online computation, as shown in Table 1 in Section 6.

For each robot position mean $\mathbf{p}_{i,\mu}$ evaluated, we use a small-volume approximation of the probability that a depth return point $\mathbf{d}_j$ and the robot are in collision, by multiplying the point Gaussian probability density by the volume V_r of the robot's sphere:

$$P(\text{Collision, } \mathbf{p}_i(t)) \approx V_r \times \frac{1}{\sqrt{\det(2\pi\Sigma_p)}} \exp\big[-\frac{1}{2}(\mathbf{p}_{i,\mu} - \mathbf{d}_j)^T \Sigma_p^{-1} (\mathbf{p}_{i,\mu} - \mathbf{d}_j)\big] \quad (2.10)$$

where Σ_p is the covariance of the robot position as described by the model. This small-volume spherical approximation has been used in the chance-constrained programming literature [92]. If the above equation evaluates to > 1 (possible with the approximation), we saturate it to 1. A key implementation note is that using a diagonal covariance approximation enables the evaluation of Equation 2.10 approximately an order of magnitude faster than a dense 3×3 covariance. Rather than use online-estimated covariances of velocity, we choose linear velocity standard deviations $\sigma_{v\{x,y,z\}}$ that scale with linear velocity.

For our quadrotor race through the forest, since the obstacles are all convex and so navigating out of dead-ends is not a concern, we use a simple Euclidean progress metric as our navigation function R_{nav}, plus a cost on terminal speed $v_f = ||\mathbf{v}_i(t_f)||_2$ if it is above the target max speed, v_{target}:

$$R_{nav}(\mathcal{M}_i) = ||\mathbf{p}_0 - \mathbf{p}_{goal}|| - ||\mathbf{p}_i(t_f) - \mathbf{p}_{goal}|| + R_v(v_f) \tag{2.11}$$

$$R_v(v_f) = \{0 \text{ if } v_f < v_{target},\ kv_f \text{ if } v_f \geq v_{target}\} \tag{2.12}$$

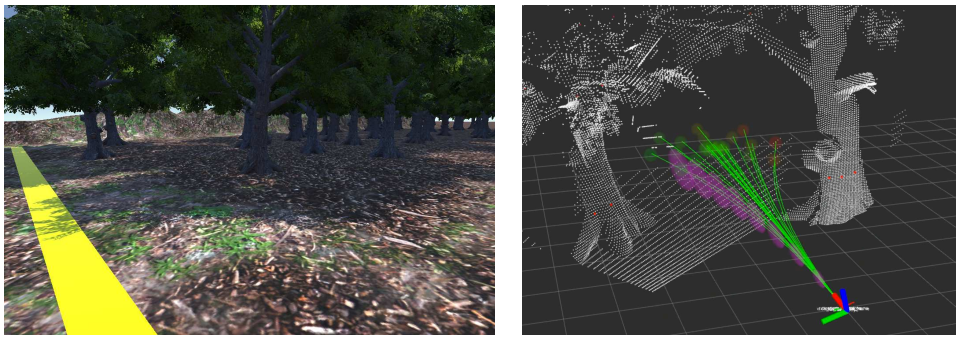
Where we used $k = 10$, and $R_{collision} = -10,000$.

2.5 Simulation Experimental Setup

2.5.1 Simulator Description

To facilitate the comparison study, simulation software was developed to closely mimic the capabilities of our hardware platform for the Draper-MIT DARPA FLA (Fast Lightweight Autonomy) research team. The sensor configuration includes a depth sensor that provides dense depth information at 160x120 resolution out to a range of 10 meters, with a FOV (field of view) limited to 58 degrees horizontally, 45 degrees vertically. A simulated 2D scanning lidar provides range measurements to 30 meters. Both sensors are simulated at 30 Hz.

Drake [97] was used to simulate vehicle dynamics using a common 12-state nonlinear quadrotor model [98] while the Unity game engine provides high fidelity simulated perceptual data that includes GPU-based depth images and raycasted 2D laser scans. The flight controller uses a version of the Pixhawk [99] firmware running in the loop (SITL) that utilizes an EKF over noisy simulated inertial measurements to estimate attitude and attitude rates of the vehicle.



(a) (b)

Figure 2-4: (a) Screenshot from our race-through-forest simulation environment in Unity. (b) Screenshot from Rviz which shows the evaluation of the 25-maneuver real-time-generated motion library. The chosen maneuver and the 1-σ of the Gaussian distribution over time are visualized. The small sphere at the end of each maneuver indicates approximated collision probabilities from low to high (green to red).

2.5.2 Experimental Setup

The experiments were carried out in a virtual environment that consists of an artificial forest valley that is 50 meters wide and 160 meters long. The corridor is filled with 53 randomly placed trees whose trunks are roughly 1 meter in diameter. A timer is started when the vehicle crosses the 5 meter mark and stopped either when a collision occurs or when the 155 meter mark is reached. If the vehicle is able to navigate the forest without colliding with any of the trees or terrain in under a predetermined amount of time, the trial is considered a success. Collisions and time-outs are considered failures.

The experiments were repeated for each algorithm at various target velocities $v_{target} = \{$ 3, 5, 8, 12$\}$ meters per second and with increasing levels of state estimate noise for $x, \dot{x}, y, \dot{y}$. We do not simulate noise in the altitude or in the orientations since these are more easily measurable quantities. To simulate noise that causes position to drift over time, we take the true difference in x, y over a timestep, $\Delta\mathbf{p}_{x,y}$, and add zero-mean Gaussian noise which is scaled linearly with the velocity vector. The three noise levels we use are $\sigma = \{0, 0.1, 1\}$ which is scaled by $\frac{\sigma}{10}\mathbf{v_{true}}$. This linearly increases noise with higher speed. We also add true-mean Gaussian noise to $\dot{x}$ and $\dot{y}$, with standard deviations that are the same as for position noise. Accordingly we

have:

$$\mathbf{p}_{noisy}[i+1] \sim \mathcal{N}(\mathbf{p}_{true}[i+1] - \mathbf{p}_{true}[i], \frac{\sigma}{10}\mathbf{v_{true}}) \tag{2.13}$$

$$\mathbf{v}_{noisy}[i] \sim \mathcal{N}(\mathbf{v}_{true}[i], \frac{\sigma}{10}\mathbf{v_{true}}) \tag{2.14}$$

The total time taken and the trial outcome was recorded for 10 trials at each noise and speed setting, for a total of 360 simulation trials.

2.5.3 Dijkstra's Algorithm with Pure Pursuit Description

We compare our method to a typical map-based robotics navigation solution that consists of a global path planner that is paired with a path following algorithm. The particular implementation we chose functions by maintaining a global probabilistic occupancy grid (Octomap [46]) with a 0.2 meter voxel size. At a specified rate, a horizontal slice of the map is extracted and a globally optimal path is computed using Dijkstra's algorithm. The path planning includes a soft cost on proximity to obstacles. We then use a pure pursuit algorithm to command a vehicle velocity along the resulting path to the goal. This approach has been heavily tested on our hardware, and shown considerable success in complex environments in the range of 2.0 to 5.5 m/s with little state estimate noise.

2.6 Simulation Results and Discussion

The key metric for our comparison of the three methods is the no-collison success rate of reaching the finish line, and is presented in Figure 2-5. Additional data is presented in Figure 2-6: average time to goal for successful trials, and example paths at various noise levels.

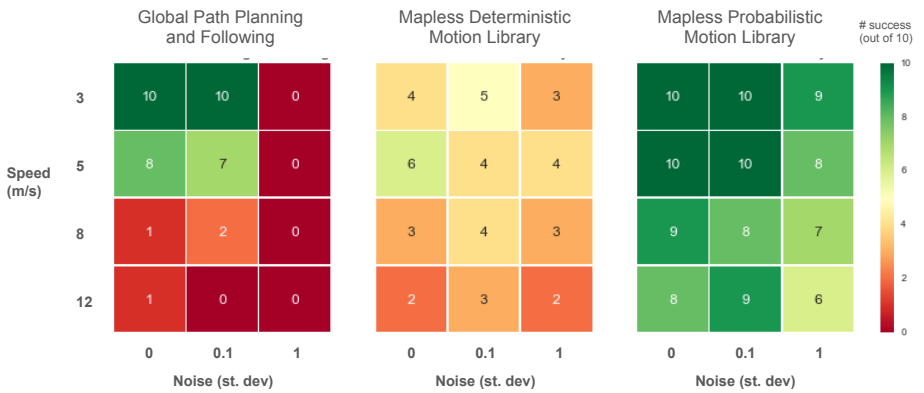


Figure 2-5: Comparison summary of number of successful collision-free trials for the different approaches tested in in our simulated quadrotor race through the forest. Ten trials were run for each of the three approaches, for four different speeds {3, 5, 8, 12} meters per seconds, and for three different levels of 2-dimensional state estimate noise as described in Section 2.5.2.

The results for the global path planning and following approach show both the limitations on handling higher speed, and on handling higher state estimate noise. The approach was not able to handle any of the severe noise ($\sigma = 1$) for any of the speeds and was only able to reliably reach the goal at 5 m/s and below, with zero or little state estimate noise. These limits on speed and state estimate noise match well our experimental results in hardware. Primary inhibiting factors for this approach's success are (i) dependence on a global position estimate, (ii) latency incurred by processing sensor data into a global map (up to ∼50 ms), (iii) latency incurred by path planning on the local map (up to ∼200 ms), and (iv) neglect of vehicle dynamics, which are increasingly important for obstacle avoidance at higher speeds.

For comparison, we also compare with the approach of deterministically collision-checking our motion primitive library. For this deterministic method, the average time to goal on a successful run was faster than the probabilistic method by approximately 14%. The deterministic nature of the collision checking, however, causes the method to leave little margin for error while navigating around obstacles. Thus, small inaccuracies in the linear planning model (which approximates the nonlinear model used for simulation) or in the state estimate can lead to fatal collisions.

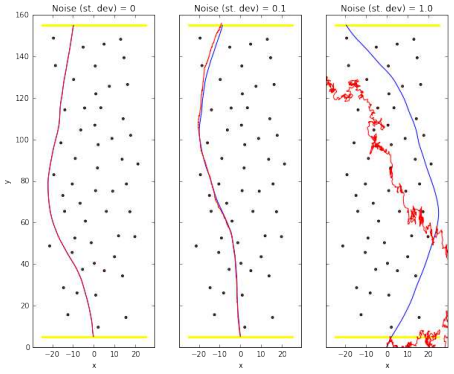
The results for the probabilistic method demonstrate a marked increase in robustness at higher speeds and with noise levels an order of magnitude higher than was

manageable by the path following approach. The sacrifice in average time to goal compared to the deterministic method is outweighed by the gains in robustness.

Average time to goal for successful trials (s)

Speed (m/s)	Global Path Planning and Following	Mapless Deterministic Motion Library	Mapless Probabilistic Motion Library
3	60 +/- 6	51 +/- 1	59 +/- 2
5	36 +/- 2	29.7 +/- 0.3	34 +/- 1
8	24	18.1 +/- 0.1	21 +/- 2
12	15	12.1 +/- 0.2	14.1 +/- 1

(a)



(b)

Figure 2-6: (a) Comparison summary of the average time to goal for successful trials for $\sigma = 0$, which all methods were at least able to get 1 trial across the finish line. (b) Visualization of the different noise levels $\sigma = \{0, 0.1, 1.0\}$ and representative paths for the probabilistic motion library navigating successfully through the forest at 12 m/s. The path of the noisy x, y state estimates (red) are plotted together with the ground truth path (blue). The brown circles represent the tree obstacles at the flying altitude of 1.8 m.

Additionally, an important practical consideration is that, given our fast collision probability approximations, the total computation times of the probabilistic and deterministic methods are nearly identical ($\sim$3-4 ms total), as is displayed in Table 1. This is a strong argument for replacing deterministic collision checking with fast collision probability approximation in a wide number of scenarios. We also emphasize that these approximate computation times are achievable on our actual flight vehicle hardware, which uses an Intel i7 NUC.

	Deterministic, N=1		**Probabilistic, N=1**		**Probabilistic, N=10**	
Subprocess	Average time (μs)	Percentage time (%)	Average time (μs)	Percentage time (%)	Average time (μs)	Percentage time (%)
Building kd-tree	1900 +/- 700	50.5	2000 +/- 500	57.8	1900 +/- 400	42.6
Evaluating future positions from real-time generated 25-maneuver motion library	40 +/- 10	1.0	40 +/- 10	1.1	40 +/- 10	0.9
Evaluating collision probabilities with N-nearest neighbor search on kd-tree	1800 +/- 800	47.9	1400 +/- 600	40.5	2500 +/- 1000	56.1
Evaluating expected reward, given R_{nav}	2 +/- 1	0.1	2 +/- 1	0.1	2 +/- 1	0.0
Calculating attitude setpoint for attitude controller	17 +/- 5	0.5	17 +/- 5	0.5	17 +/- 5	0.4

Table 2.1: Measured averages and standard deviations of subprocess latencies, from one representative run each. Implementation on single-thread Intel i7.

2.7 Future Work

There are several components to this line of work that we would like to extend. For one, we plan to present validation experiments of the method in hardware. Additionally, the highly parallel nature of the fast collision probability approximation algorithm is amenable to data-parallel implementations on a GPU. We also plan to expand on the motion primitive library, including true 3D flight, increased variety of maneuvers, and analysis of the accuracy of the model. We also plan to characterize the performance of the collision probability approximation with more elaborate global navigation functions.

Chapter 3

Hardware Validation

In this chapter, we analyze hardware validation experiments of the approach discussed in the previous chapter.

Most notably, results are presented for the fastest known sustained UAV flight through a dense forest. Previous top speeds demonstrated for sustained flight through a forest were 1.5 m/s [10, 23], whereas we present flights 4x faster, at up to 6 m/s, and dodge up to 39 obstacles in one continuous flight. We also present fast obstacle avoidance in an indoor warehouse up to 10 m/s, and other near-building flight up to 7 m/s. These results are among the fastest and most robust results ever demonstrated for autonomous UAVs navigating unknown environments. In particular, the outdoor results are presented using a GPS-denied visual inertial odometry (VIO) estimator, for which robust obstacle avoidance is especially difficult at speed. For the outdoor flights, the obstacle avoidance system is demonstrated with a global planner that enables the vehicle to get itself out of maze-like dead ends.

Chapter Acknowledgements

The hardware experiments presented in this chapter were the result of a team effort, representing many individual contributions towards the overall goals of our Draper-MIT FLA team. Among these contributions, the author would particularly like to acknowledge John Carter and Jake Ware for their extensive development of the overall autonomous system, and the many hours spent dedicated to testing the particular

contributions discussed in this chapter. Brett Lopez contributed to overall planning and control components. Jake Ware and John Carter integrated the global planner into the system. Nick Greene contributed to autonomous mission system intelligence. The visual inertial odometry (VIO) state estimation system was developed by Ted Steiner and Rob Traux. The Gaussian Particle Filter (GPF) with a prior map estimation system was integrated by John Carter and Jake Ware. Scott Rasmussen led the hardware design and building. Kris Frey handled the RealSense sensor integration and filtering. Steve Paschall and Julius Rose contributed to overall program and testing management. This chapter will focus on evaluating the specific performance of the system contributed by the author, which handles local planning and obstacle avoidance, and its integration with the rest of the system.

Chapter Organization

The first set of results, Section 3.1, tests the approach in hardware in an indoor warehouse environment, with no software changes of note from the integrated perception and control approach that was evaluated in the simulation experiments of the previous chapter. This system has no prior obstacle information given, but the state estimation system does, and is performed with a Gaussian Particle Filter (GPF) by matching laser scans against a prior map [41, 42]. No global planning is used, and obstacles are dodged at up to 10 m/s.

The second set of results, Section 3.2, extends the work to outdoor environments with a VIO state estimator, and tests the approach in a variety of outdoor forest and near-building environments. In order to transition to outdoor, robust flight, a variety of supplementary components are added to the approach, which are presented. A layered global planner is used, and sustained flight through a forest is achieved at 6 m/s. Obstacle avoidance in near-building and outdoor/indoor transition flight is also demonstrated, at up to 7 m/s.

3.1 Indoor Warehouse Flight

3.1.1 Experimental Setup for Indoor Flight

Hardware

The quadrotor hardware used in this work is shown in Figure 3-1. The frame is the standardized platform for the DARPA FLA program: a DJI Flamewheel F450 airframe, with DJI E600 motors and 12" propellers. Onboard computation is provided by a dual-core Intel NUC5i7RYH. The total vehicle weight is 2.8 kg. The vehicle can hover at approximately 61% throttle with its 6S LiPo battery. The sensor used for obstacle perception was the ASUS Xtion sensor, mounted with 15 *deg* tilt up. The Xtion provides depth images with ∼8-10 m range at 30 Hz at VGA (640 × 480) resolution, which were downsampled to 160 × 120 resolution. An onboard 2D laser, the Hokuyo UTM-30LX, was used for state estimation, as well as a downward-facing single-point LIDAR (LidarLite v2). Attitude control and an onboard InvenSense MPU-6000 IMU was provided by a 3DR Pixhawk, running the ETH Pixhawk software stack[1]. The chassis is a custom 3D print.

Figure 3-1: Draper-MIT quadrotor in configuration used for indoor flight experiments, flying at speed (left, and top right), and the fleet of airframes (bottom right).

[1]https://pixhawk.ethz.ch/software/start

State Estimation

For state estimation, a Gaussian Particle Filter (GPF) [41, 42] was used that fused measurements from the Hokuyo laser scans matched against a prior-built map, the downward-facing single-point laser, and the onboard IMU. Only the state estimation – not obstacle perception or planning – has any prior map information available.

Environment

Indoor warehouse results are presented from testing inside of the dedicated Draper-MIT FLA testing facility. The obstacles in this environment are floor-to-ceiling pillars, which are approximately 1 meter in diameter, and are spaced in a grid with center-to-center distance of approximately 7 meters. Since the pillar obstacles are all convex, there is no need for a global planner.

3.1.2 Results from Indoor Flight

Three autonomous flights, varying only by the target max speed v_{target}, are presented. The start, goal, and environment were the same for each flight. A goal location is given to the vehicle 32 m away at a 10 deg angle, so the vehicle is forced to cross one row of pillars. The vehicle is programmed to autonomously navigate to the goal at 1.4 m altitude, and return the start location, with $v_{target} = \{5, 8, 10\}$ $\frac{m}{s}$.

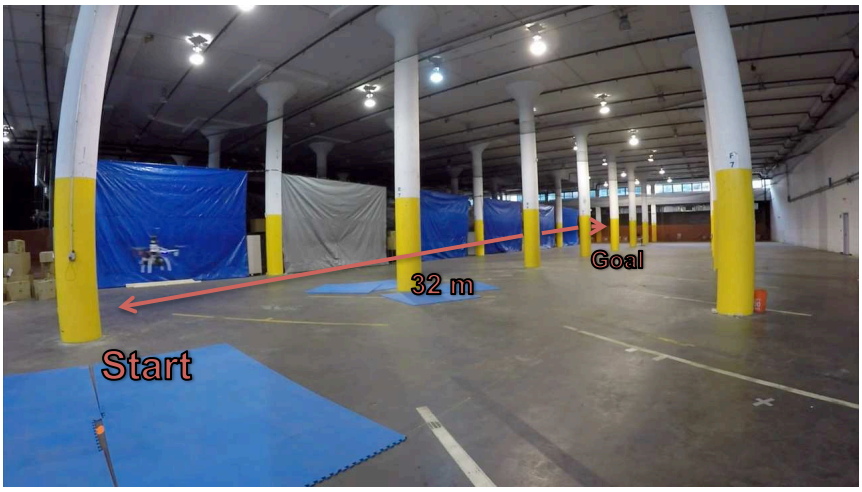


Figure 3-2: Setup of warehouse environment, and locations of start and goal (32 m away, behind the row of pillars).

The vehicle was able to successfully navigate the unknown environment, and achieve the target max speed in each scenario. A summary of the results is provided in Table 3.1. The maximum estimated speeds achieved for the three flights were respectively 5.5, 8.4, and 10.2 $\frac{m}{s}$. The speed profiles over time for each flight are plotted in Figure 3-3. The beginning of the third flight in particular showcases the system's capability to intelligently slow down and speed up again while rounding a sharp corner around the pillars. For the most part, each one-way trip produced two autonomous dodges of the pillars[2]. The maximum estimated roll angles achieved for the three flights were respectively 25.3, 37.4, and 42.6 *deg*.

	Duration (s)	**Distance Traveled** (m)	**Max Speed** ($\frac{m}{s}$)	**Max Roll** (deg)	**Obstacles Dodged**	**Result**
Flight i	32.4	75.3	5.5	25.3	5	Autonomous return and land
Flight ii	31.8	80.3	8.4	37.4	4	Autonomous return and land
Flight iii	27.5	76.4	10.2	42.6	4	Autonomous return and land

Table 3.1: Summary of three flights of increasing speed in warehouse.

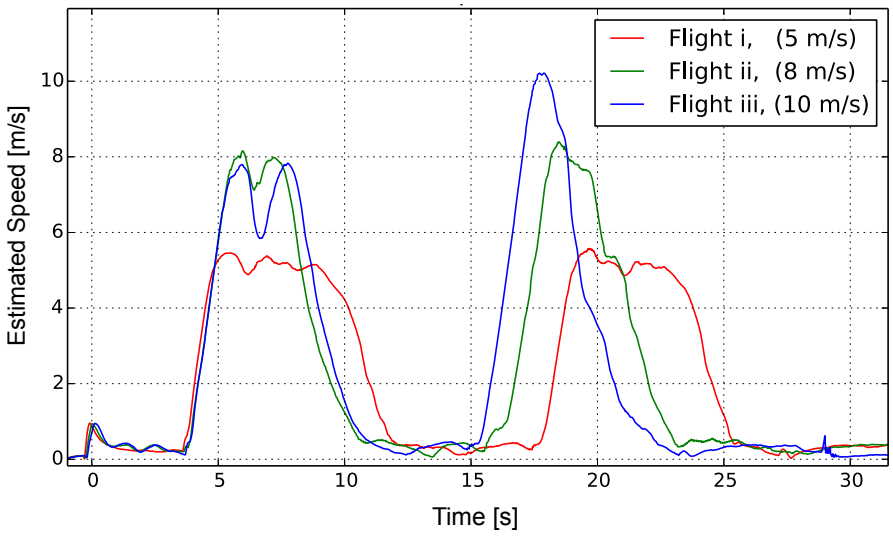


Figure 3-3: Speed over time for the three warehouse flights. When the vehicle speed is near 0, it is turning around at the goal.

[2]A dodged pillar was counted if the vehicle deliberately chose to avoid it.

A sampling of the obstacle avoidance maneuvers are analyzed in step-by-step detail in Figures 3-4, 3-5, and 3-6. For each figure, three sequential moments are represented by a pair of an RGB image (left) which helps provide the reader with context, and a 3D visualization (right) of the integrated perception and control. In the visualization, for each motion primitive path (light green), the approximated collision probability of each primitive is visualized with the sphere at the end of each primitive, (red is high collision probability green is low, i.e. $RGB = [P_{collision}, 1 - P_{collision}, 0]$). The point cloud produced by each Xtion depth image is rendered so that color corresponds to depth (red is close, blue is far). The laser scans (white) are also shown, which help give a sense of the attitude of the vehicle. The chosen motion primitive (purple) is shown with the σ of the time-varying distribution of configuration.

Note that the Xtion sensor was tilted (pitched) 15 *deg* up, and so images that look near-level are actually during forward-pitch flight.

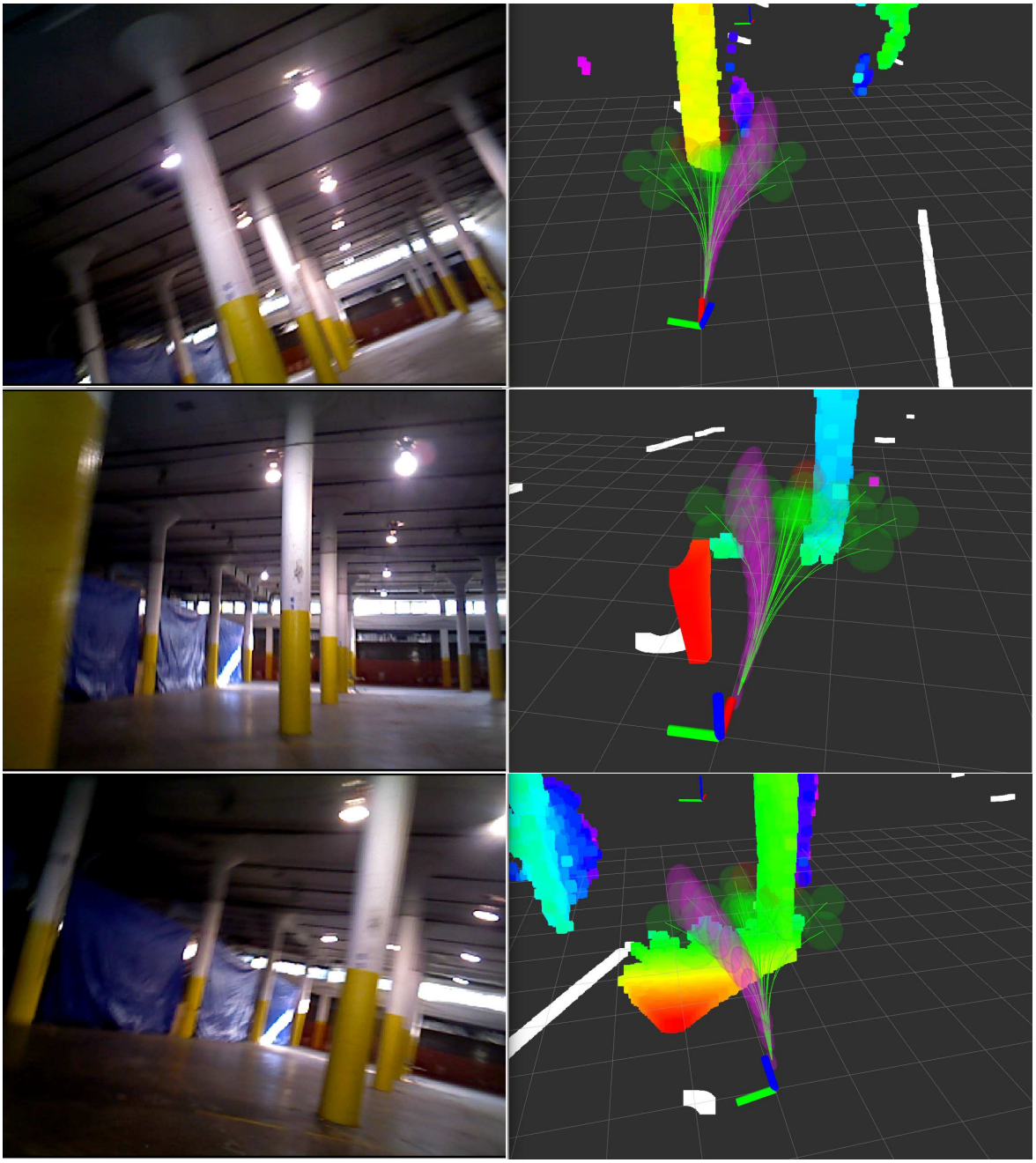


Figure 3-4: Outbound flight at 5 m/s. The vehicle approaches the second pillar at speed (top), then begins to roll left as soon as the previous pillar is passed (middle), taking this turn with a roll angle of approximately 25 *deg* (bottom).

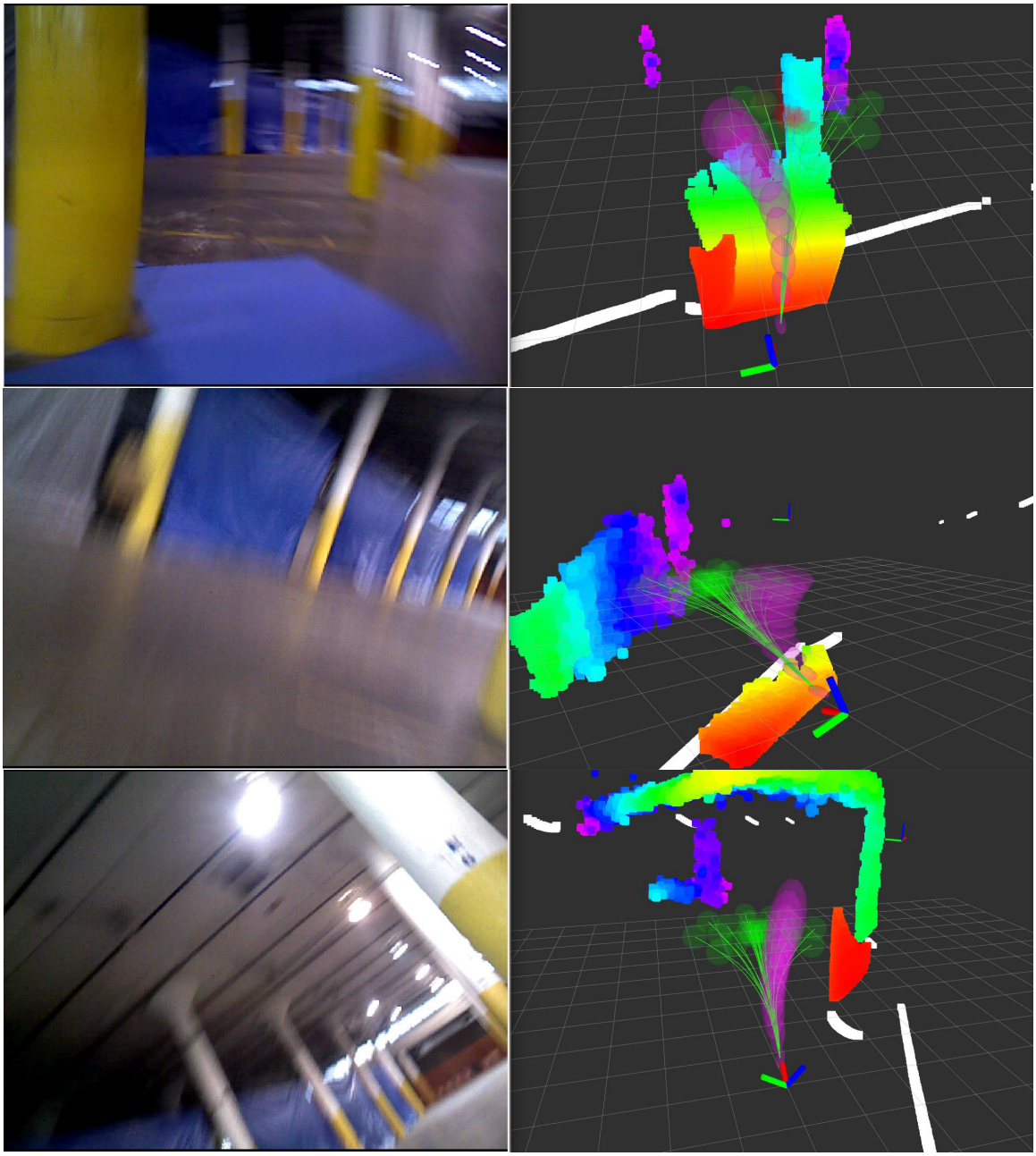


Figure 3-5: Outbound flight at 8 m/s. The vehicle approaches the first pillar and takes a different route than the previous flight, cutting sharply around the first pillar (top), quickly rolling and then choosing to come out of the turn (middle), aggressively coming out of this swerve at approximately 37 *deg* roll (bottom). Rotational motion blur in the RGB images is significantly higher than the 5 m/s flight.

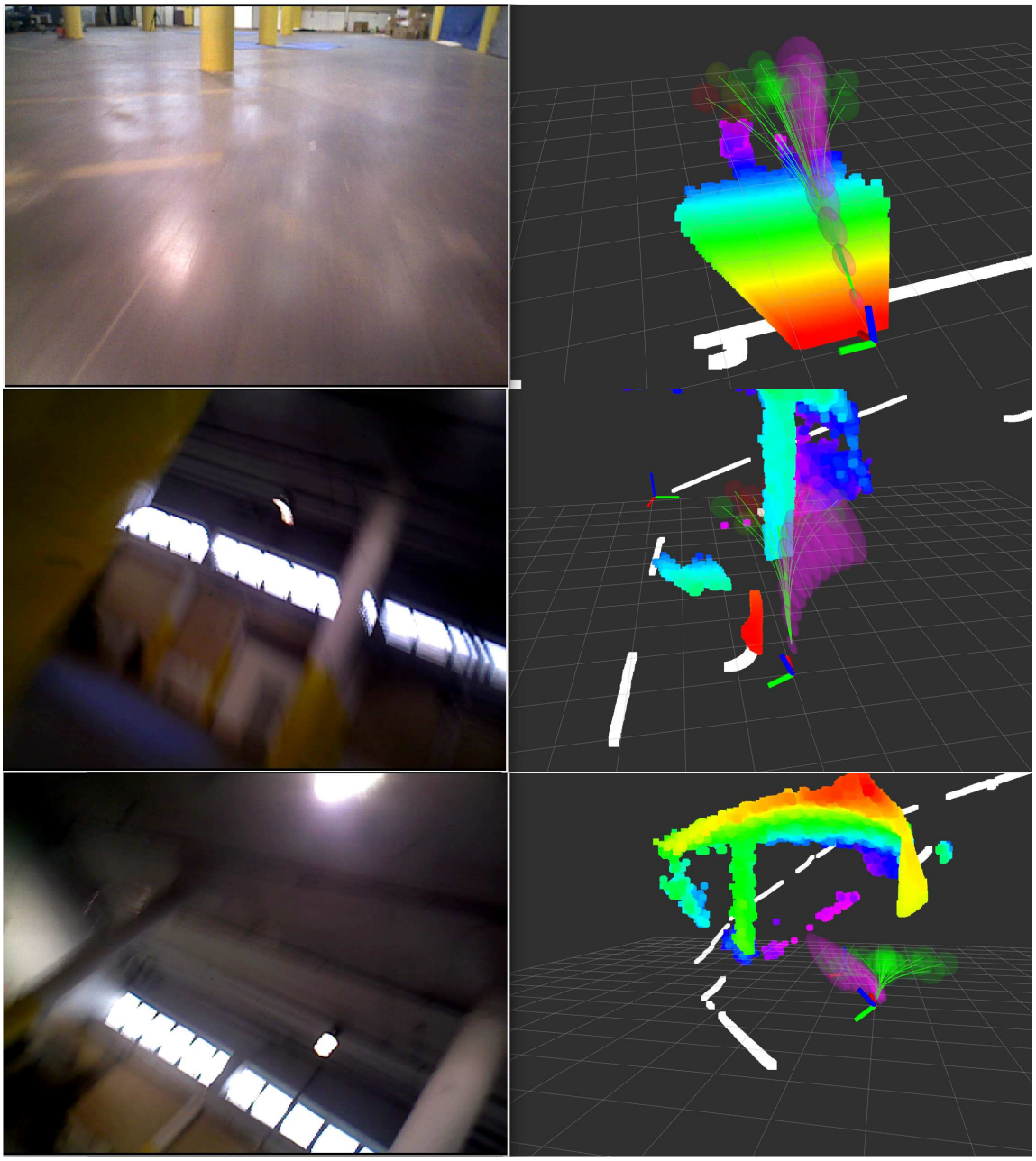


Figure 3-6: Return flight at 10 m/s. The motion blur from the linear velocity of 10 $\frac{m}{s}$, at 1.4 m altitude, is noticeable, and the 40 *deg* pitch causes obstacles to almost be out of the FOV, even with a 15 *deg* tilt angle for the Xtion (top). The vehicle decides which distance to keep from the pillar according to its collision probabilities (top), then starts rolling and pitching back (middle) and execute a 42.6 *deg* roll to come out of the turn around the second to last pillar (bottom).

3.2 Outdoor Forest and Near-Building Flight

3.2.1 Adaptations for Outdoor Flight

Transitioning to outdoor flights in more complex environments and with a visual inertial odometry (VIO) state estimation system presented a variety of difficulties beyond the scenario presented for the indoor flights. Accordingly, a variety of adaptations were implemented for the integrated perception and control system, which are outlined below.

Smooth Flight with Chance-Constrained Optimal Primitive

A non-optimal result of the motion primitive library previously presented, in which primitives are sampled over the acceleration input space, is that discretization effects can cause non-smooth flight alternating between primitives. Particularly for the benefit of visual inertial odometry, it was preferable to have vehicle motion that avoids unnecessary rapid attitude changes, which makes visual feature tracking difficult. To address this, an additional primitive was generated which rather than sampling over an acceleration input, is calculated to be the approximate optimal input, in the absence of obstacles. The minimum-time input for a double integrator system subject to a nonlinear actuator constraint cannot be calculated in closed form, but instead an approximation was used that works well in practice. Given the planning horizon time, t_f, the position in the local frame at $t = t_f$ with no control input is easily calculated: $\mathbf{p}_{t=t_f} = \mathbf{v_0} t_f$. The optimal acceleration is then approximated as the acceleration that will get the vehicle moving in the direction $\Delta\mathbf{p}_{f,desired} = \mathbf{p}_{goal} - \mathbf{p}_{t=t_f}$ at the desired top speed, i.e. $\mathbf{v}_{f,desired} = \frac{\Delta\mathbf{p}_{f,desired}}{|\Delta\mathbf{p}_{f,desired}|} \times |v_{max}|$. With the double integrator approximation, this optimal acceleration is $\mathbf{a}^* = \frac{\mathbf{v}_{f,desired} - \mathbf{v}_0}{t_f}$. If the optimal acceleration was above the chosen acceleration limits, then it was scaled down to within the acceleration limit, i.e. if $|\mathbf{a}^*| > |a_{max}|, \text{then } \mathbf{a}^* := \frac{\mathbf{a}^*}{|\mathbf{a}^*|} \times |a_{max}|$. A similar process is applied for when the vehicle is within stopping distance of the goal.

To supply clear constraints, the optimal primitive was chosen in a chance-constrained approach. This ensured it was chosen when viable, rather than subject its selection

to the mixing of other rewards (from obstacles, global guidance, and terminal velocity cost). In particular, the optimal primitive was chosen if $P_{collision}(\mathcal{M}^*) < \epsilon$, where ϵ is some small probability ($\epsilon = 0.02$ was used). If not, then all primitives were evaluated with all mixed rewards.

Global Planner Integration

To be able to navigate complex, maze-like environments, global guidance was provided by a 2D A* planner from the `global_planner` module of the open-source `navigation` ROS package[3]. Its integration was designed to allow slack for the local planning system to "selectively listen" to the global guidance only when needed. This desire for loose integration was due to: (i) mapping errors causing spurious obstacles to appear in the global map, and (ii) the inability of the A* paths to represent dynamic constraints. Rather than follow the global path exactly, as in a map-plan-track type approach, we wanted only the global planner to give a general direction for the local planning component, and when the vehicle was stuck in a dead-end, to get it turned around and moving in the correct direction. The solution used was to have the local goal for the local planning system to be a "carrot" on the global planner's path, where the carrot distance d_{carrot} scaled by the vehicle's velocity, $|v|$. The relation used was $d_{carrot} = 1 + 4|v|$, with a minimum carrot distance of 1. Rigorous analysis of any global-local planning integration is difficult, but empirically this architecture provided strong results. Global frequency was limited to 2 Hz, with approximately 100 ms latency. Conversely the local obstacle avoidance was performed at the depth image rate (60 Hz) with approximately 1 ms latency. The global-local interaction is analyzed more later.

A simple prior map was used which did not include individual obstacle information but biased the vehicle in the correct general direction (Figure 3-7a). This map would clear out and fill with obstacles as the vehicle progressed (Figure 3-7b). Due to high noise of the RealSense sensor, only the Hokuyo laser was used to clear obstacles.

[3]http://wiki.ros.org/navigation

Figure 3-7: Prior map (a) and map on return flight (b). The inset shows an onboard image at the start location pointing north (a), and on the return flight in the forest (b). The 2D A* global plan is shown in light green. The colored view cone is the FOV of the RealSense sensor, which will give returns out to 60 m, even though only the first ∼10-20 m are typically useful.

Vertical Oscillations to Help Monocular State Estimator

Since the monocular VIO state estimation system had difficulties in handling the case of constant forward-velocity flight, which is an otherwise ideal flight regime for the vehicle, vertical oscillations were commanded to help overall system performance. The cause of this difficulty is a well known aspect of monocular visual odometry: when the velocity vector is parallel to the camera axis [100], the scale factor cannot be estimated well. During accelerations, an IMU can provide a visual-inertial estimator with a sense of scale, but not if there are no accelerations, such as during constant forward-velocity flight. Hence, until our estimation system can handle these degenerate cases, a stopgap solution used was to sinusoidally oscillate the z setpoint for the vehicle, according to $z(t) = A * cos(t * \frac{2\pi}{T})$, where the amplitude A used was 0.5 m, and period T was 3 seconds.

Robust 2D Flight with Laser and Stereo Combination

Given the difficulties of vision-based obstacle detection for certain difficult lighting or low-texture environments, it was determined that a laser-based perception system was

needed to complement the vision-based system. Even if the vision-based perception from the RealSense was highly capable at perceiving natural high-texture obstacles such as trees, it was essentially blind to important obstacles, like the broad sides of low-texture walls. Complementary perception was provided by a 2D Hokuyo laser sensor, which even in worst-case lighting conditions (direct sunlight) can reliably detect obstacles at approximately $10 - 15\ m$ (although it is specified for 40 m range, this only applies in indoor / low-light conditions). This sensor offers reliable detection, but its rigid mounting and lack of a vertical field of view presents difficulties for flight with aggressive rotations. To address this, laser-data-processing assumed the world is mostly 2.5 dimensions – i.e., a the 3D laser point cloud data was projected into the 2D plane at the altitude of the vehicle. To avoid projecting the ground, points were only projected up to the vehicle plane if they were within 0.5 meters below the vehicle, or any distance above. To incorporate both laser and vision data, the collision probability approximation used the $n = 1$ closest laser point and the $n = 1$ closest depth image point, with an independence approximation between them. The FOV constraints for the depth image were still imposed. Accordingly, for each time-sampled distribution of configuration $p_{t_i}(\mathbf{q})$, we have:

$$\big[1-P(\text{Collision}, p_{t_i}(\mathbf{q}))\big] = \begin{cases} 0, & \text{if } \mu(p_{t_i}) \in \mathcal{U} \\ \prod_{j=\{laser,stereo\}} \big[1 - P(\text{Collision}, p_{t_i}(\mathbf{q}), \mathbf{d}_j)\big], & \text{otherwise} \end{cases} \tag{3.1}$$

Emergency Stop Maneuver

There is a worst-case failure mode for the approach presented previously, when all collision probabilities are very large (i.e., the vehicle believes it has no chance of avoiding a wall). To address this, an emergency-stop maneuver was implemented. To avoid false-positive detection of the need for an emergency stop, only the laser data, which is typically cleaner than the vision data, was used to determine the emergency stop. If all motion primitives, according to the laser only, had a higher collision

probability than some threshold ($P_{collision} = 0.7$ was used), then the stop maneuver was executed.

3.2.2 Experimental Setup for Outdoor Flight

Hardware

The hardware used for the outdoor experiments were almost identical, with only slight modifications. Rather than an ASUS Xtion depth sensor, an Intel RealSense R200 (mounted with 0 pitch angle) was used, which provided depth images at 60 Hz and 480×360 resolution, which were then median-filtered to reduce spurious obstacles and down-sampled to 120×90 resolution. The Hokuyo was still present, but this time its point cloud was actually used for obstacle detection rather than state estimation. For these flights, a PointGrey Flea3 (FL3-U3-13Y3M-C) monocular camera, and a navigation-grade ADIS 16448 IMU was added for the VIO state estimation system. With additional camera and IMU hardware, and board to read and synchronize this data, total vehicle weight for these flights was approximately 3.2 kg.

Figure 3-8: Draper-MIT quadrotor in configuration used for outdoor flight experiments, including with front-mounted Intel RealSense sensor and bottom-mounted Point Grey Flea3 camera. (Images courtesy of Jon How)

State Estimation

State estimation was provided by a monocular visual inertial odometry (VIO) graph-based smoother, "Samwise", developed by Draper Laboratory. Samwise leverages

GT-SAM[4] for solving its pose graph.

3.2.3 Results for Outdoor Flight

Sustained Flight Through Dense Forest

Three autonomous flights in a forested environment are presented, in which the target speed was respectively $v_{target} = \{5, 6, 6\}\frac{m}{s}$. The start and goal locations were the same for each flight. A goal location was given to the vehicle 270 m away, and after taking off pointing north with a fiducial for orientation (Figure 3-7a,), the vehicle was programmed to autonomously navigate to the estimated goal location at a 2 m altitude, and return. A portion of the flight (approximately half) was through an edge-of-forest clearing, while the other half was through the forest (Figure 3-9).

Figure 3-9: Overhead imagery of the flight path. The vehicle was instructed to navigate 270 m, of which approximately half of the flight was along a clearing next to the forest, and the other half was in the forest. (Image taken from Google Maps.)

The vehicle successfully navigated into and out of the forest and achieved the target max speed in each scenario. A total of 84 obstacles (trees, shrubs, vehicle),

[4]https://bitbucket.org/gtborg/gtsam

were dodged along the way[5]. A summary of the results is provided in Table 3.2. The maximum estimated speeds achieved for the three flights were respectively 5.3, 6.4, and 6.2 $\frac{m}{s}$. The speed profiles over time for each flight are plotted in Figure 3-10. The maximum estimated roll angles achieved for the three flights were respectively 25.2, 33.5, and 44.2 *deg*.

	Duration (s)	**Distance Traveled** (m)	**Max Speed** ($\frac{m}{s}$)	**Max Roll** (deg)	**Obstacles Dodged**	**Result**
Flight i	192.3	716.4	5.3	25.2	39	Autonomous return and land
Flight ii	158.3	711.5	6.4	33.5	27	Autonomous return and land
Flight iii	127.8	508.6	6.2	44.2	18	Safety pilot land

Table 3.2: Summary of three flights through forest.

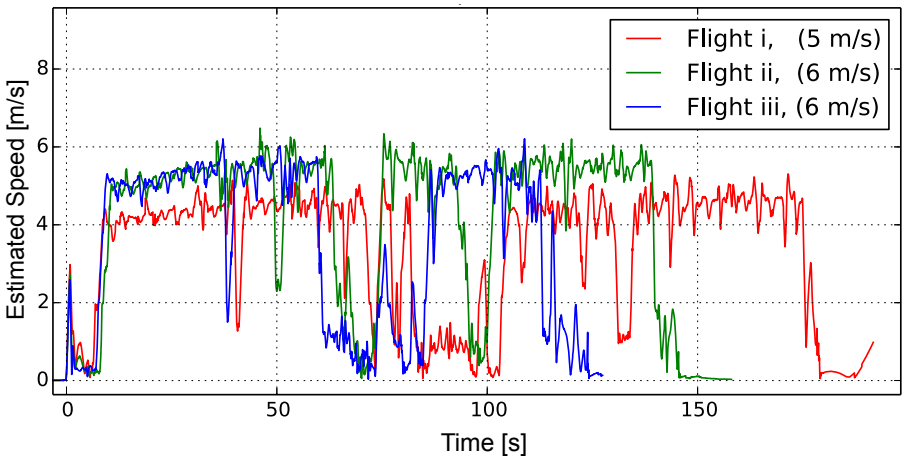


Figure 3-10: Speed over time for the three flights through the forest. The moments where vehicle speed drops to near 0 are either when the vehicle is turning around at the goal, or is stuck in a dead end.

Before analyzing the vehicle's flight, it is helpful to get a sense of the raw depth sensor data that was available. In the high-texture forest, the RealSense sensor performs notably well, detecting depth returns with reasonable reliability in the 10-20 m range, although noisy. Figure 3-11 displays this raw data from a variety of moments throughout the test environment.

[5]Distinct obstacles were counted for each separate object (tree, shrub, etc) that the vehicle intentionally dodged in its flight path.

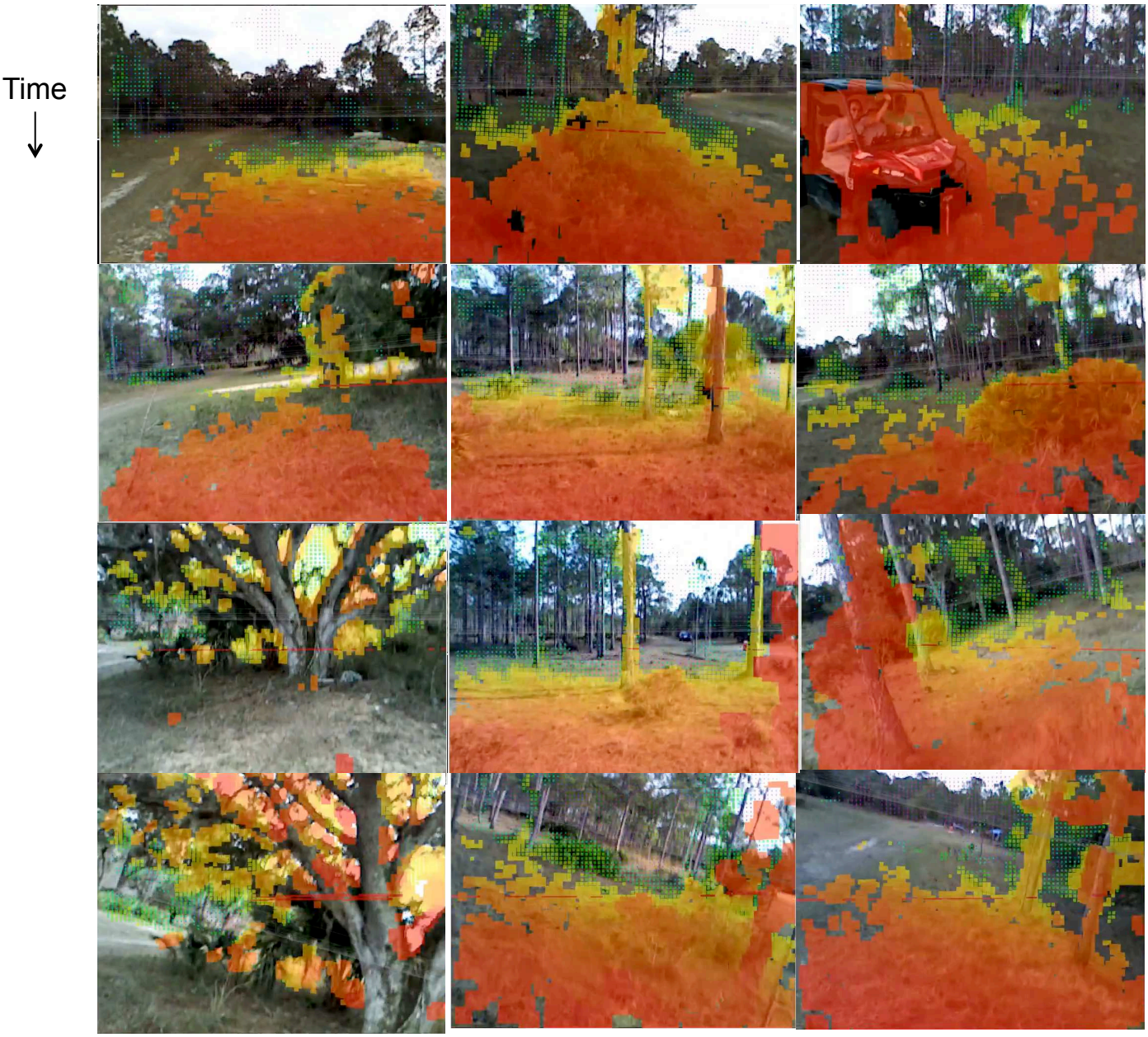


Figure 3-11: Raw RGBD data from the RealSense sensor, from a variety of moments throughout Flight i. The point cloud from the depth image is displayed as an axis color map (red is close, green is far) and is projected onto the RGB image. Time progresses down each column, starting with the left, then moving to the middle and right columns. Note that many natural textures, including thin trees (middle column), grass, and shrubs (right column, second from top) are detected. A Polaris vehicle (top right) was also detected and dodged. The bark of the large tree, however (bottom left), is blind to the depth sensor – as an IR sensor, it is very surface-sensitive.

A sampling of the obstacle avoidance maneuvers are analyzed in step-by-step detail in Figures 3-12 through 3-16, etc. The visualizations are similar to what was provided for the indoor flights, except the global A* path (light green), the velocity-scaled "carrot" (orange), and global 2D map are also provided in the image.

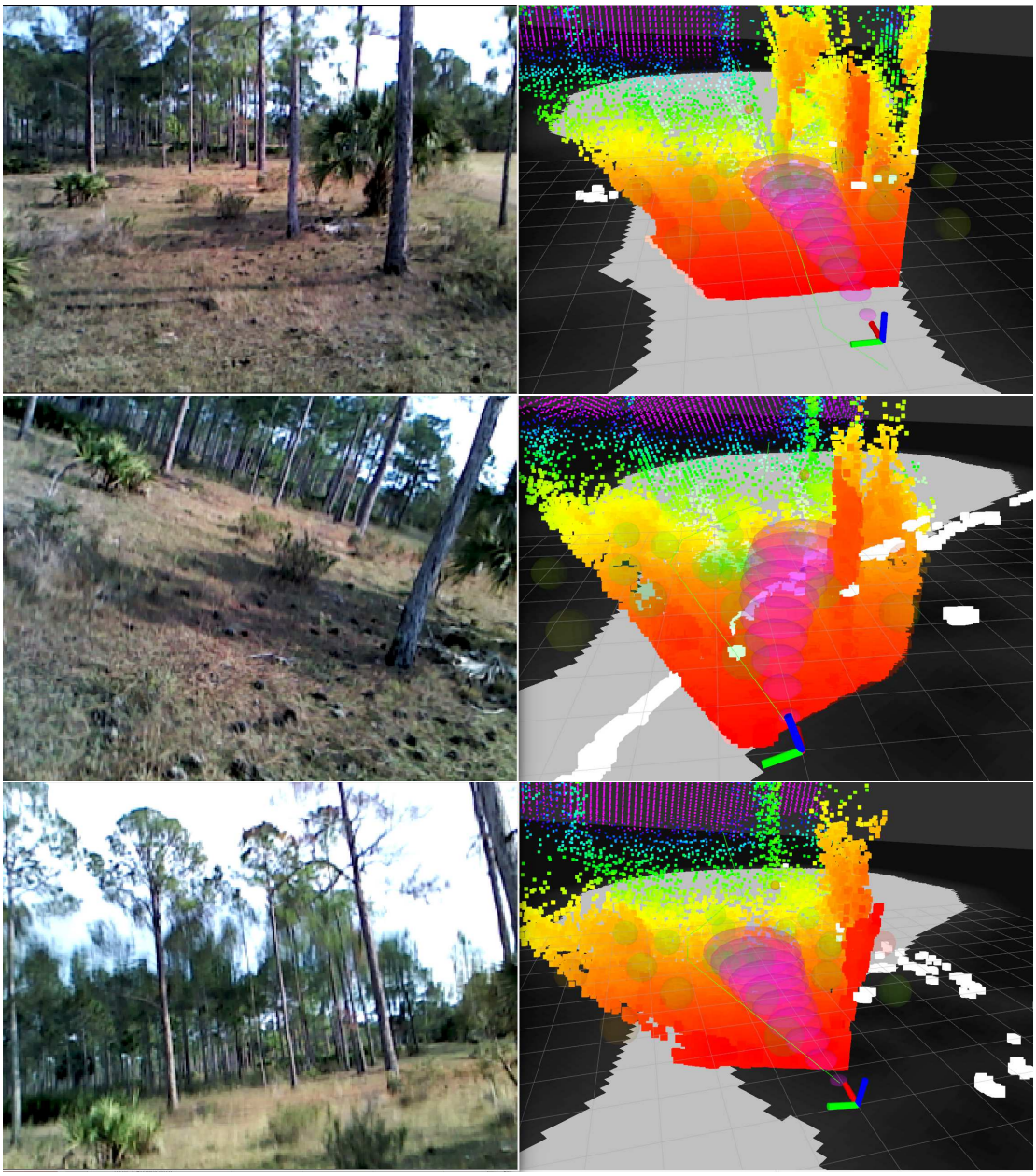


Figure 3-12: Outbound, 5 m/s from Flight i. Vehicle approaches a pair of trees (top), rolls left 21.2 *deg* and then rolls right out of the dodge (middle), and returns to forward flight (bottom). Note that although the global plan veers off to the left due to mapping errors, the local planner chooses to fly through the open.

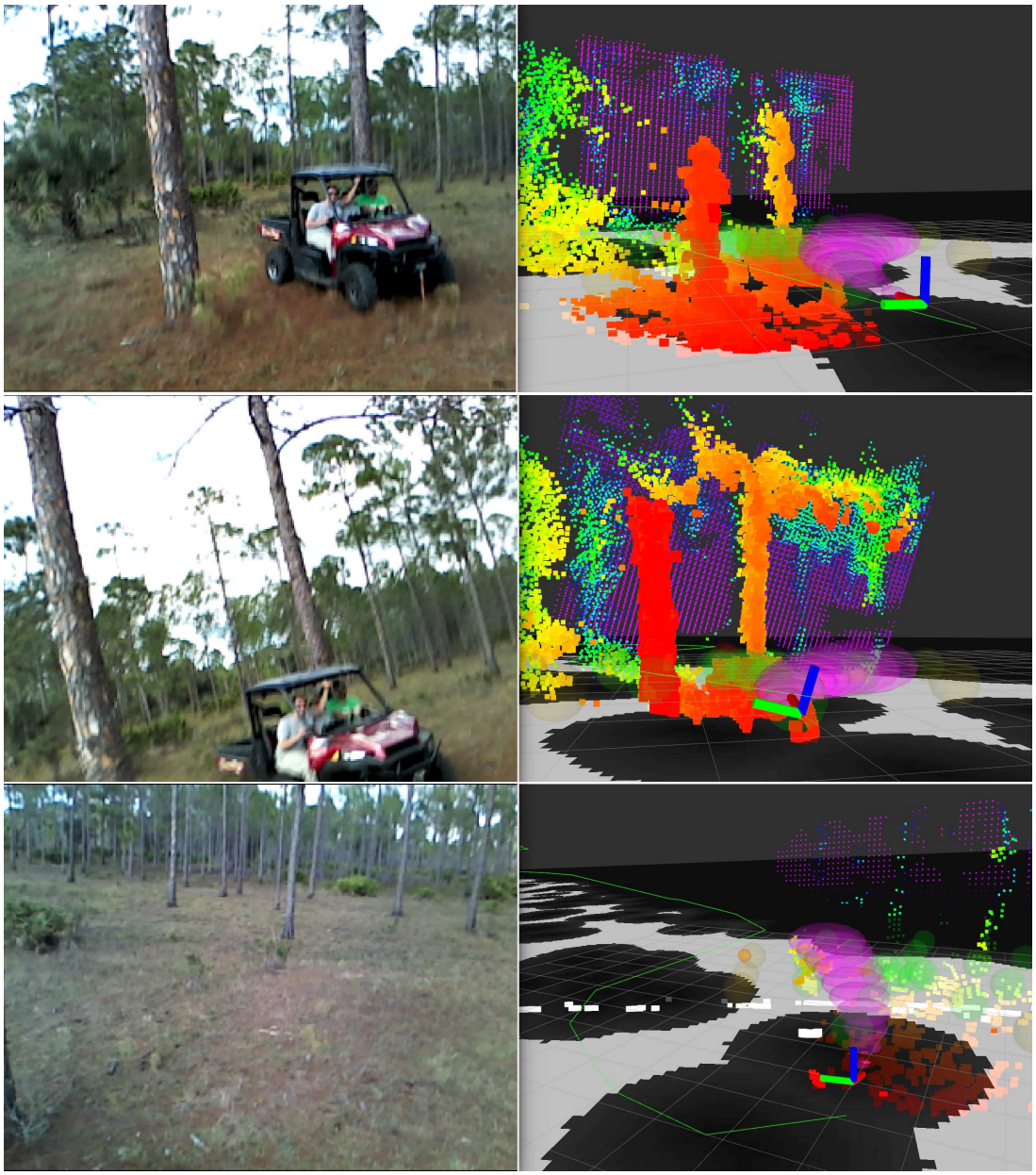


Figure 3-13: Return, 5 m/s from Flight i. Quadrotor approaches a pair of obstacles (top), not that although the initial velocity is to the right, the A* planner does not encode dynamic constraints and plans a path to the left. The quadrotor instead dynamically avoids the car to the right, passing the car (bottom) and returning to level flight (bottom).

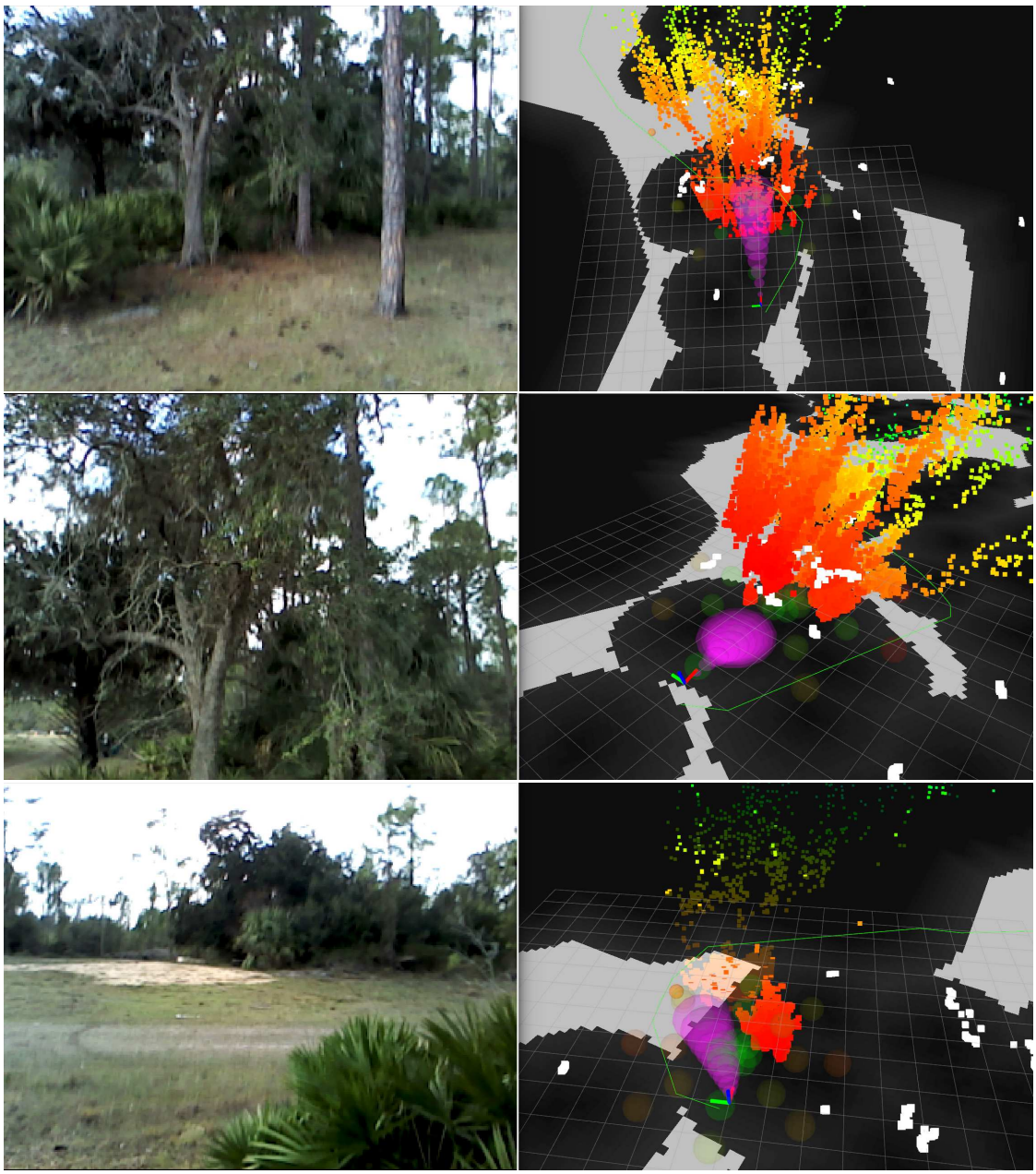


Figure 3-14: Demonstration of the global planner helping the vehicle get out of a dead end. The vehicle approaches a dead end (top) at 5 m/s (Flight i), chooses to stop to avoid collision (middle), and the global planner then guides the vehicle around to the left, out of the dead end (bottom).

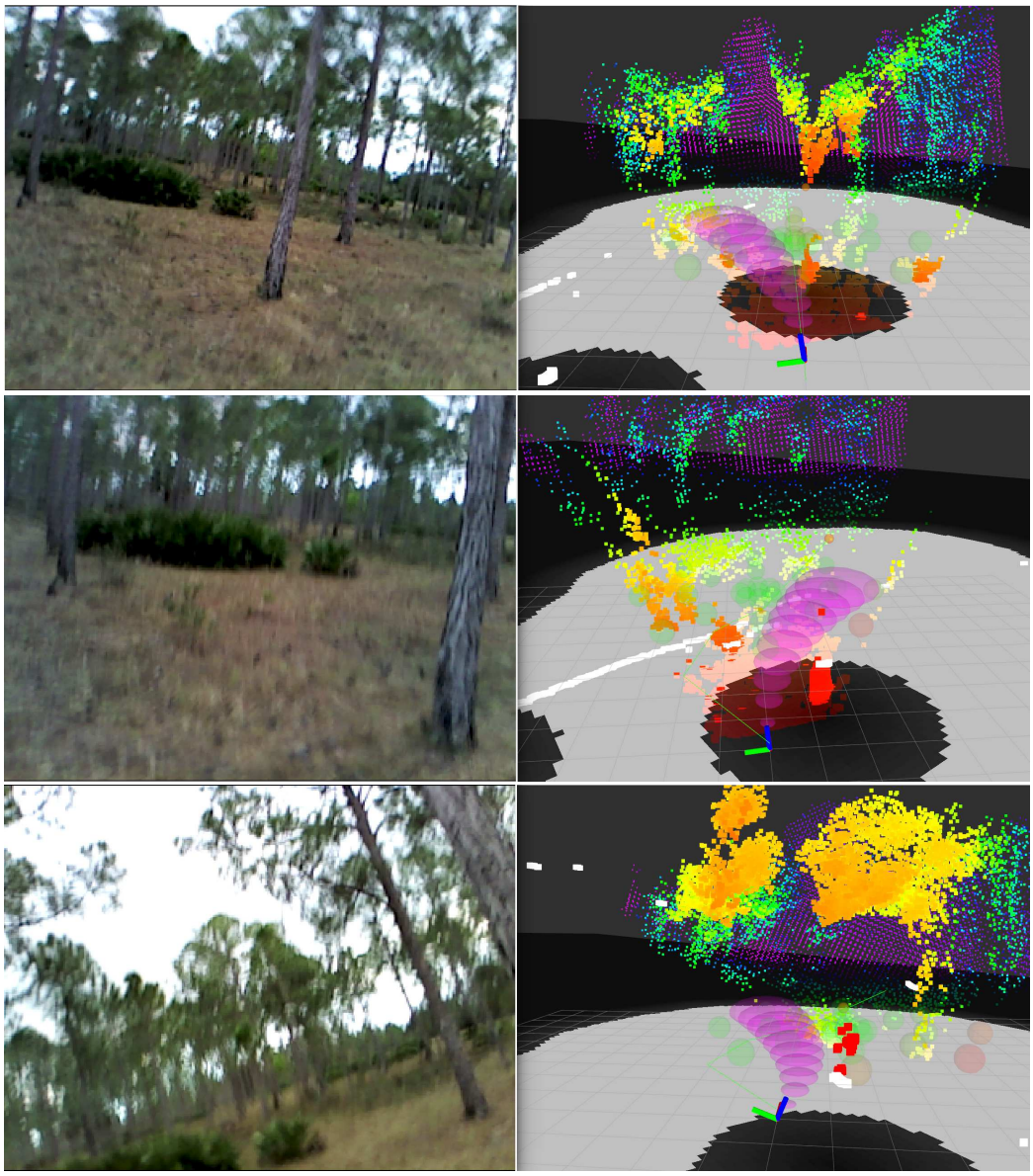


Figure 3-15: Outbound, 6 m/s from Flight ii. Vehicle approaches tree (top) and rolls to left, then chooses to snap around tree as soon as it is safely past (middle), rolling 28.9 *deg* as it comes around (bottom).

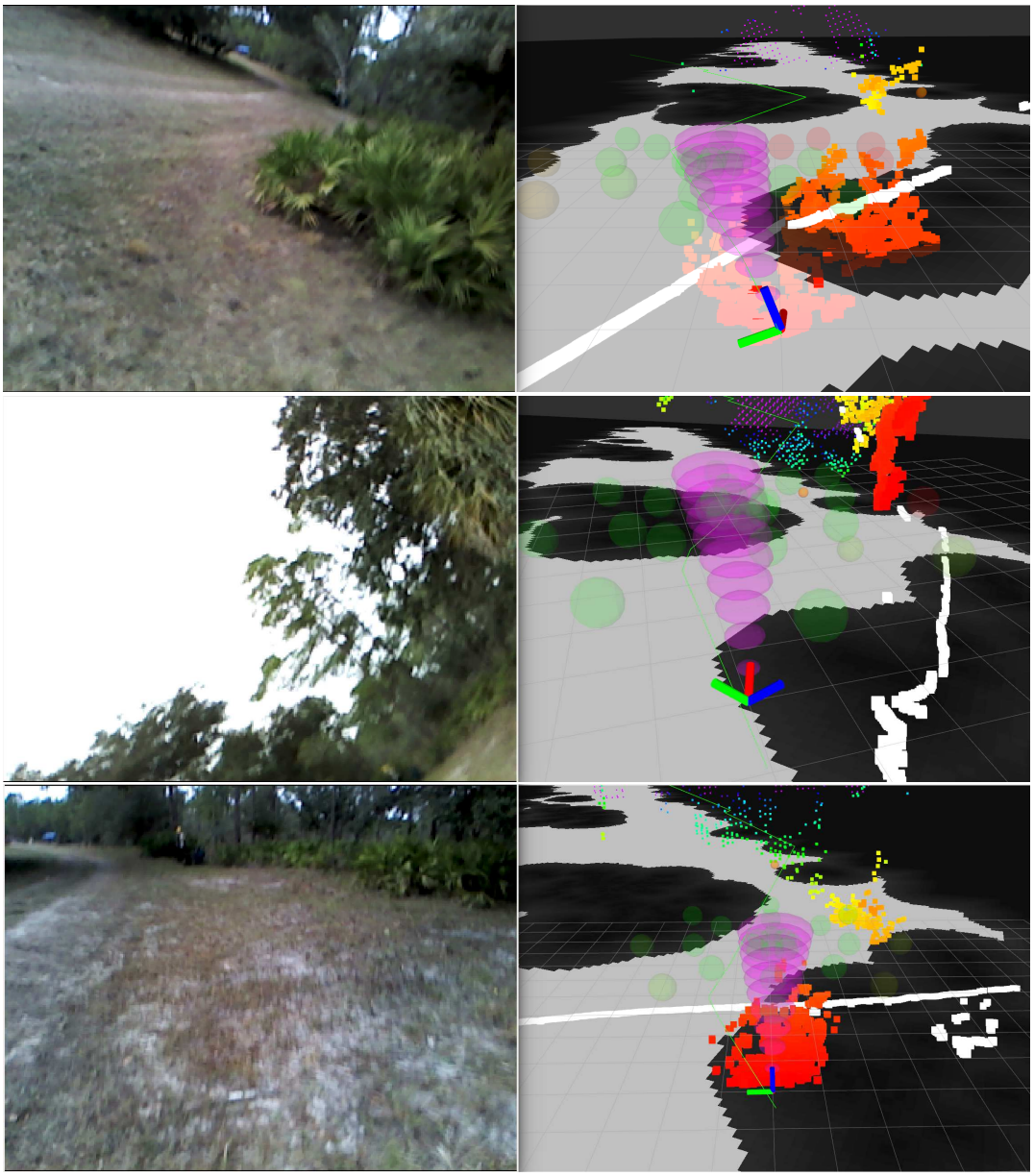


Figure 3-16: Outbound, 6 m/s from Flight iii. Vehicle approaches shrubs and rolls left around them (top), snaps around the corner with a 44.2 *deg* roll (middle), then returns to level flight (bottom).

The robustness of the method is apparent from the 84 obstacles that were consecutively dodged over the course of these three flights.

One item to note is the interaction of the global planner and local planner. In a variety of figures above, the local planner overrides the global planner's guidance, since it has a more detailed local understanding of obstacle information, and encodes the dynamic constraints (including initial velocity and acceleration of the vehicle). As demonstrated in Figure 3-14, however, the global planner is critical for enabling the vehicle to turn around out of dead ends.

Near-Building Flight: Rounding Building Corners and Exiting Buildings

In addition to the forest flights, we present results from flying near buildings outside at speed. Flying near large walls at speed presents its own challenges, different than those of flying in the forest. In part, this is due to the perception difficulty: since man-made walls are often low-texture, they are hard for the vision sensors (i.e., RealSense) to perceive well. In our case, this meant relying on the 2D laser to perceive these walls. Also, however, the obstacle avoidance is also particularly challenging: small lateral dodges, as work with trees, are not always viable. One common obstacle avoidance method that is suited well both for trees, and for walls, can be difficult. Since most of the duration of the following two flights did not involve obstacles, so we focus only on the obstacle avoidance moments. Figure 3-17 shows the vehicle coming around the corner of a building at 7 m/s, in which mostly the laser data is relied upon. Figure 3-18 shows the vehicle navigating out of a door, at 6 m/s.

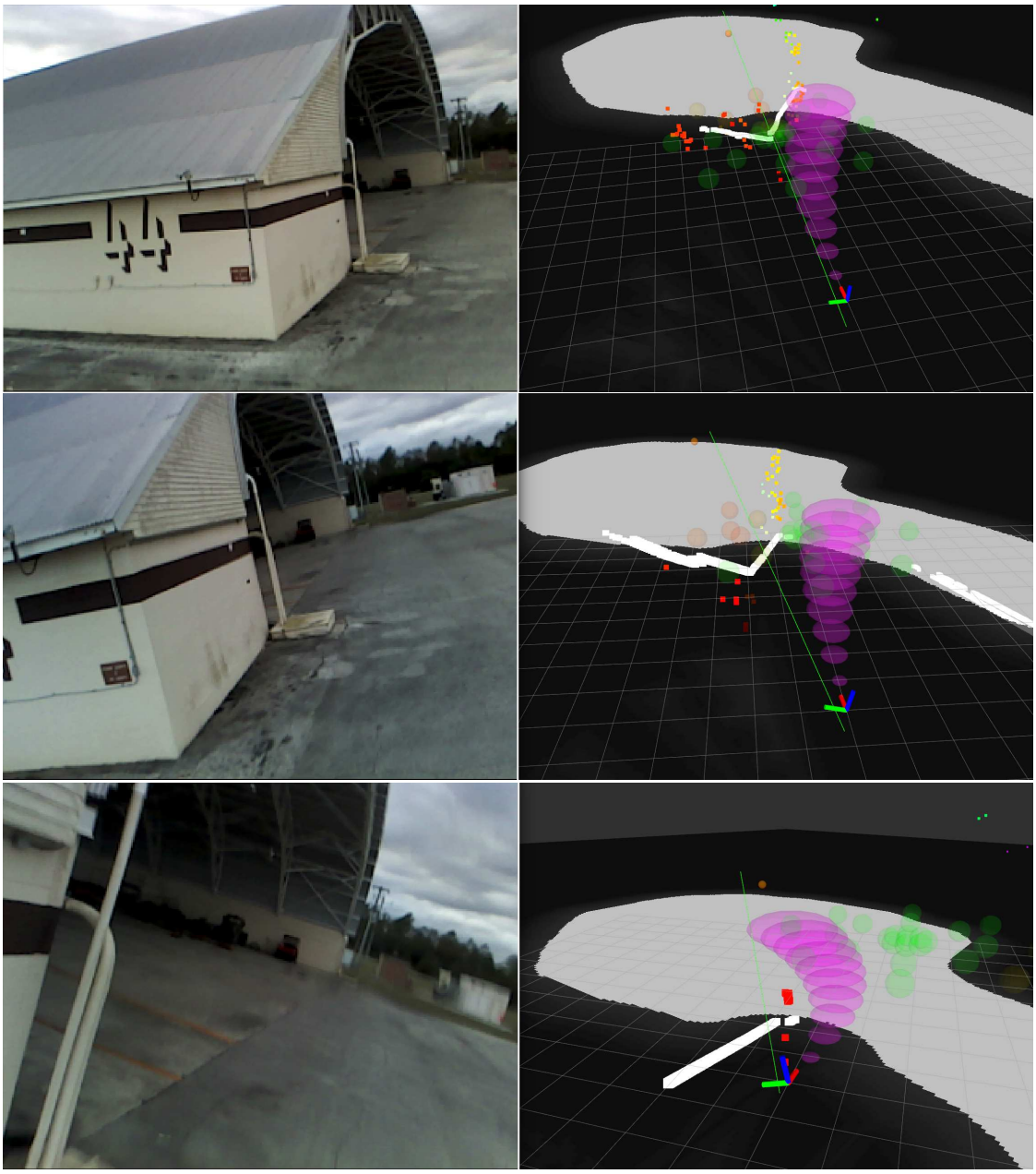


Figure 3-17: Flight at 7 m/s around the edge of a hangar. With mostly laser returns and only sparse RealSense returns, the vehicle rolls right to avoid the wall (top). Despite a bug in the global planner that caused it to propose bad plans when in false-positive occupied space (middle), the local planner is responsible for navigating the vehicle to safety. As soon as the vehicle has dynamically cleared the wall (bottom), it snaps around it with a left roll.

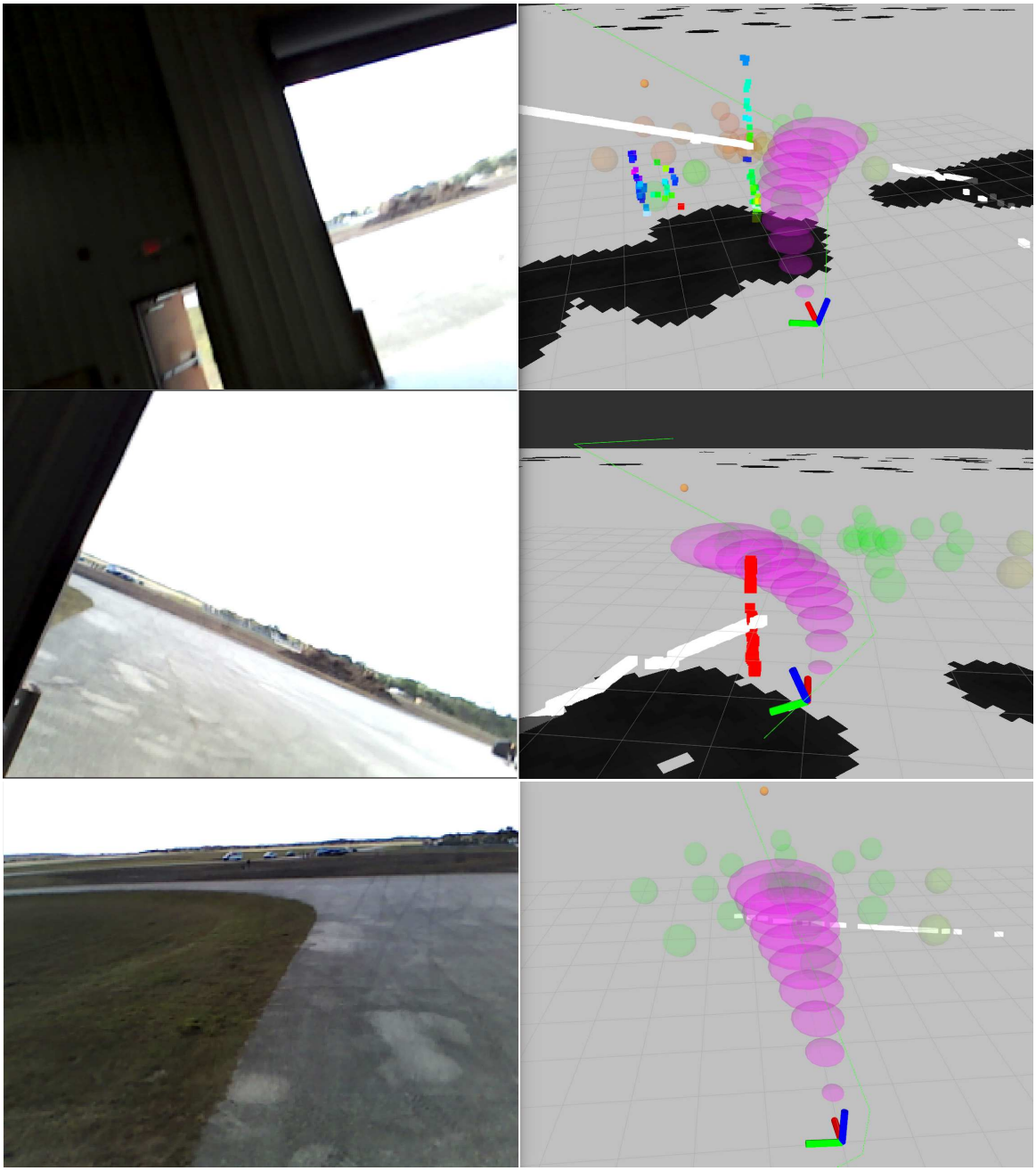


Figure 3-18: Flight at 6 m/s exiting a warehouse to outdoors. The vehicle relies on mostly laser perception (top) to see the opening, then snaps around it (middle), and comes to level flight after clearing the exit (bottom).

3.3 Discussion

We have demonstrated some of the fastest and most robust flight, in difficult environments, at speeds of 5 - 10 m/s. As a highlight, we have increased the fastest sustained flight ever achieved through a forest by a factor of 4.

Despite the successes we have observed in hardware experimentation, there are a variety of areas that offer room for improvement and further research. Many of these opportunities are outside the realm of capabilities addressed here – for example, improving raw perception data. Other possible improvements would be to work on tuning small things, such as tuning to make flight smoother. It is expected that eliminating the vertical oscillations in altitude would help reduce flight jerkiness in outdoor environments.

Perhaps the largest opportunity for the most tangible increase in performance would be to incorporate some memory into the obstacle avoidance. There have been a variety of times where the vehicle will oscillate due to forgetting what it has just seen. This topic is addressed in the next chapter.

Chapter 4

Robust Obstacle Avoidance Beyond Maximum-Likelihood Maps: Depth-Pose-Graph Planning

4.1 Introduction

Robust, fast motion near obstacles is an open problem that is central in robotics, with applications spanning across manipulation, autonomous cars, and UAV navigation in unknown environments. To address this problem, this chapter explores how to efficiently and robustly use a short history of depth measurements, particularly in regimes of difficult state estimation for UAVs. The common approach is to build a map from a history of maximum-likelihood estimated poses, but in regimes of significant state estimation uncertainty, mapping errors can be the downfall of planning motions around obstacles [11, 32]. Accordingly, a notable trend in the state of the art has been to develop state-space planning approaches to obstacle avoidance that use only the most recent depth sensor measurement [10–12, 17, 32], which effectively eliminates pose estimation uncertainty from the problem.

It would seem that using memory of depth sensor measurements should be strictly superior to a memoryless approach, since additional information should only benefit

decision-making. In every map-based obstacle avoidance method known to the authors, however, the representation of the world used in the planning process is either a maximum-likelihood estimate map batched over some time interval, $\hat{\mathcal{M}}_{MLE}$, for example in a SLAM (Simultaneous Localization And Mapping) method allowing loop closures, or is a map built incrementally from maximum-likelihood poses. Whenever a maximum-likelihood estimate does not represent the distribution well (for example, distributions that are high-variance and/or multi-modal), mapping errors can be difficult to recover from (even if future measurements are good).

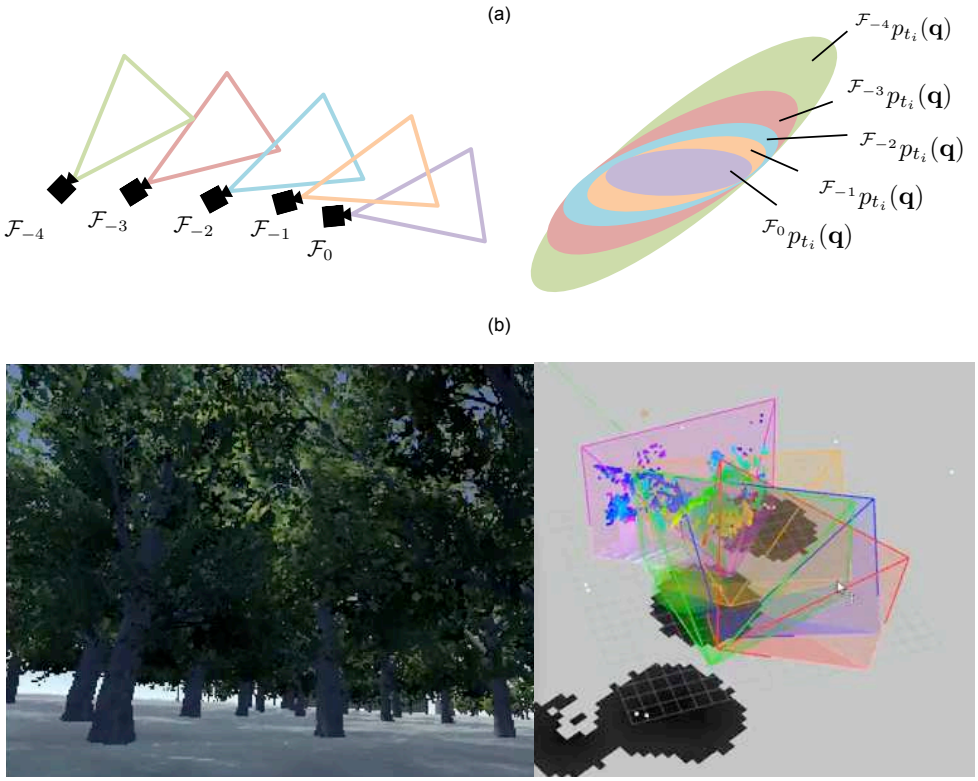


Figure 4-1: (a) Depiction of frame-specific configuration uncertainty $^{\mathcal{F}_{-j}}p_{t_i}(\mathbf{q})$ for a series of depth image field of views (FOVs). (b) Visualization of using Depth-Pose-Graph Planning to navigate a simulated forest (simulated image on left, visualization of subset of recent FOVs on right).

In this chapter, we present a novel formulation, Depth-Pose-Graph Planning, that enables robust obstacle avoidance with memory, through incorporating a distribution of the world in the obstacle avoidance decision-making process. Rather than operating on a summarized representation of the world as in mapping based methods,

this approach uses the entirety of the information available to evaluate motions: raw depth images and a pose graph (as in a pose graph SLAM framework). This structure respects both the natural field of view (FOV) and occlusions of the information available, and also enables appropriate modeling of uncertainty. In this framework, raw depth sensor data is not fused in order to create a map, but rather the raw depth sensor data and the pose graph itself are used to collectively score motions. Since we do not build a map but rather only use sensor data to the extent that it scores motion plans, we think of this as a type of "perception in the service of control".

To those familiar with robotic mapping and planning, the idea to use a distribution of the world state rather than a maximum-likelihood estimate seems desirable, but expensive computationally. Efficient inference, however, can be done using both the raw depth image data and uncertainty measures incorporating pose estimation, velocity estimation, and depth sensing. In particular when the structure of the problem is exploited through a hierarchy of graph search and spatial partitioning, it can be tractable for real-time implementation. The computational efficiency is notably desirable when loop closures are allowed: mapping approaches require reinsertions for each N depth measurements in memory, i.e. $\mathcal{O}(N)$ insertion complexity for uniform grid maps or $\mathcal{O}(N \log N)$ for octree structures, but there is no increase in complexity for loop closures with our method.

In terms of contributions, this is the only work known to the authors that addresses obstacle avoidance where the representation of the obstacles incorporates a history of pose estimation uncertainty into a distribution for the world state. Among the new ideas demonstrated in this work are: (i) using frame-specific uncertainty for planning with depth sensors, (ii) efficiently using independently spatially partitioned depth measurements, and (iii) searching a history of recent raw depth measurements to satisfy field of view constraints.

4.2 Background and Related Work

The POMDP (Partially Observable Markov Decision Process) provides a general framework for belief space planning [7], where decisions are made based on a probability distribution of what the state of the world is. General-purpose solvers exist for POMDPs, and although there have been gains made in the size of problems they can address [60, 101–103], discretizing the world state with a fine resolution leads to an intractable number of states, and complexity scales exponentially with the number of time steps in the planning horizon. Belief space planning approaches that exploit particular problem structure at the expense of restricting themselves from general-purpose solvers have been addressed through a large variety of works, with a general sampling-based framework provided by the Belief Roadmap (BRM) [67]. Belief space planning with Linear-Gaussian belief spaces, as we use here, has also been used broadly [66]. To varying extents, however, all previous works involving avoiding obstacles with belief space planning require some prior knowledge about the obstacles of the environment, or use a deterministic world state rather than including a distribution of the world state.

A few related works share some features of using pose estimation uncertainty in planning, but do not address planning around obstacles in unknown environments. Previous works have used directly the uncertainty of a pose graph framework for planning but have a critical limitation that they only plan over graphs of pre-known poses [104, 105]. Other work seeks to develop generalized belief space that includes distributions over worlds, but there are no obstacles in these worlds, only landmarks for navigation [106]. Another related work includes a sampling of depth perception estimates (a discrete probability distribution), but inserts them into a map structure using maximum-likelihood poses [29].

Rather than deal with the belief space of previous poses, the predominant approach for incorporating memory has been to ignore pose uncertainty, and use a maximum-likelihood mapping approach [37, 44]. Mapping-based approaches benefit from extensive decades of research into the robot mapping problem. When they are

well constrained, many SLAM approaches are able to create precise maps that are the maximum likelihood estimate map $\hat{\mathcal{M}}_{MLE}$ from the fusion of a variety of noisy depth sensor, RGB, and other sensor data. There are a variety of different ways to formulate a map – the most common version are occupancy grids, which are used ubiquitously [46]. Occupancy grids can probabilistically incorporate depth sensor measurements (multiple measurements can be required for a cell to be occupied), but this doesn't address pose estimation uncertainty. Other forms include polar maps, and for some dense SLAM techniques, surfel maps are used. The accuracies of map structures are subject to the limits of discretization (i.e., voxel and polar maps), or are limited by parametric representations of geometry (i.e., surfel maps).

A different and popular approach to the obstacle avoidance problem under significant state estimation uncertainty is to essentially cut pose estimation out of the equation, which can be done via a method that uses no memory of depth sensor measurements. In addition to planning-based approaches that exhibit this property [10–12, 16, 17, 32], any obstacle avoidance approaches that are considered reactive approaches may inherently have this property as well. Reactive approaches, including optic flow methods [15], reactive imitation-learning [23], and non-planning-based geometric approaches [27] have demonstrated considerable success at obstacle avoidance for UAVs. The limitations of memoryless obstacle avoidance have been well noted, however [16, 23]. Related approaches have limited map-building to very short time horizons [3], or have used map structures that exponentially decay old depth sensor measurements [29].

4.3 Problem Formulation

We consider the robust obstacle avoidance problem as a particular type of planning problem. This problem, which is concerned with latency on the order of seconds and a spatial area only the size of the robot's stopping distance, has a set of constraints that differs by orders of magnitude from the navigate-the-maze planning problem, which can tolerate seconds of latency and has to operate over whichever spatial area

is of concern to the robot (potentially kilometers or more). Accordingly, we consider a layered approach [11, 19, 28, 107] in which a higher-level planner provides a global navigation function $\mathcal{R}_{nav}(\pi)$, for any policy π, without itself needing to consider collision risk rigorously.

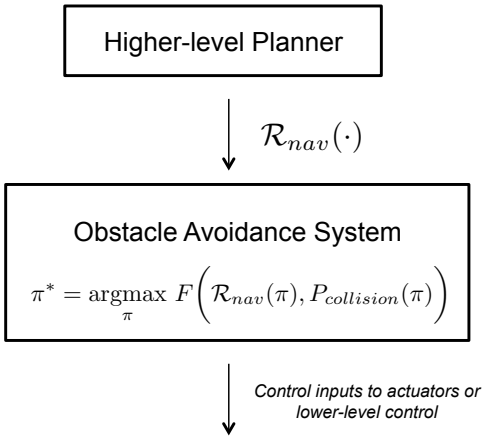


Figure 4-2: Layered planning structure that abstracts the obstacle avoidance problem away from the higher-level planning problem.

Given a higher-level navigation function, the obstacle avoidance problem reduces to determining the risk of collision for each policy π. In a probabilistic framework, this means evaluating the collision probability $P_{collision}(\pi)$. If this quantity can be estimated for any policy, then the optimal policy may be chosen by optimization over some chosen mapping F:

$$\pi^* = \underset{\pi}{\text{argmax}}\ F\left(\mathcal{R}_{nav}(\pi), P_{collision}(\pi)\right) \tag{4.1}$$

In particular in this work, we consider a finite set of policies $\Pi = \{\pi_0, \pi_1, ..., \pi_K\}$ as in popular motion library approaches, which have a variety of practical benefits including avoiding nonconvex optimization. This enables the optimal policy to be chosen as simply the best from the discrete set evaluated, $\pi^* = \underset{\pi_i}{\text{argmax}}\ F(\cdot)$.

We also specifically consider the obstacle avoidance problem where all of the information $\tilde{\mathcal{W}}$ about the true world state $\mathcal{W}$ is given in the form of a depth-pose-graph. The depth-pose-graph is defined fully in Section 4.4.1, but the two main components are (i) a sequence of depth images, and (ii) a pose graph. Since a maximum-likelihood

map can be produced from a depth-pose-graph, this set of information is general enough to allow typical mapping-based approaches.

4.4 Depth-Pose-Graph Planning

Before describing the details, we provide a brief overview of the key ideas. When each depth image is received, the raw depth image and a k-d-tree of its associated point cloud are stored in memory together with the pose from which they came. To evaluate motions for collision risk, this memory is searched to see if a queried point in space is found in the viewed free space of a depth image. If such a depth image view is found, the configuration distribution is transformed backwards in time through the uncertain pose graph. Now in the frame of a previous depth image, efficient querying of the closest obstacle is provided by the previously constructed k-d-tree associated with each depth image (within its frame, it does not need to be re-spatially-partitioned, despite a changing pose graph).

In the following subsections, we define a depth-pose-graph, describe searching a depth-pose-graph, describe how to evaluate motions using the search of the graph, and complete the description with a model, motion library, and policy for selection motions.

4.4.1 The Depth-Pose-Graph

A depth-pose-graph is the correlated combination of two types of information commonly available in robotic hardware systems that have a depth image sensor, and perform graph-based SLAM. In a depth-pose-graph, the depth images are paired with time-synchronized poses in a pose graph.

> The graph $G_{depth-pose} = (\{x, \mathcal{D}\}, \mathcal{E})$ is a combination of a set of depth images $\mathcal{D}$ and pose graph $G = (x, \mathcal{E})$ such that each node in the graph $\{x_i, \mathbf{d}_i\}$ is the pairing of the depth image $\mathbf{d}_i$ that corresponds to the timestamp of that pose x_i. It is assumed that depth images each have a corresponding time-stamped pose.

The two components are specifically defined as follows:

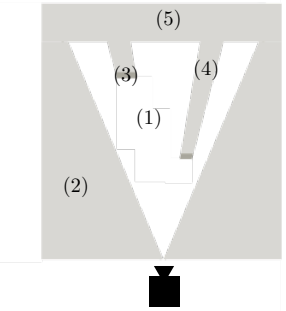
1. A set of n depth images, $\mathcal{D} = \{\mathbf{d}_0, \mathbf{d}_{-1}, ..., \mathbf{d}_{-n}\}$. Each depth image has these following key characteristics: $r \times c$ pixels, finite range R_{range}, camera intrinsic matrix K which defines the finite field of view (FOV), and obstacles create occlusions for the depth sensor (the depth $d_{r,c}$ at each pixel is the measure to the closest obstacle, and is blind to objects behind the obstacle).

2. A pose graph $G = (x, \mathcal{E})$, whose vertices x are n poses $x_0, x_{-1}, ..., x_{-n}$, and whose edges $\mathcal{E}$ are noisy measurements $\tilde{x}_{ij}$ of pairwise relative transforms $x_{ij} = (t_{ij}, R_{ij})$ between poses, $x_{ij} \triangleq x_i^{-1} x_j (i \neq j)$.

We assume that the edges of the pose graph, the noisy pairwise relative transforms $\tilde{x}_{ij}$, are provided by a separate SLAM front-end, which are available in many varieties and are often packaged separate than the back-end graph optimization solver [108]. We note that pose-graph-based SLAM is only one type of approach to SLAM – other approaches can use a simple filter, for example, or there also exist dense SLAM techniques without a pose graph [109]. The pose graph, however, contains all of the information that would be available to any other SLAM technique – one could simply run a filter, for example, on the sequential information from the pose graph.

4.4.2 Greedy Search on a Depth-Pose-Graph

Each depth image is subject to FOV limitations as shown in Figure 4-3, and so is only able to percept a limited, non-necessarily-convex polyhedron of free space.



(1)Free space
(2)Laterally outside FOV
(3)Inside occupied frustum
(4)In occluded frustum
(5)Beyond sensor horizon

Figure 4-3: The five subsets of space partitioned by one depth image.

Greedy search provides a heuristic for which depth-pose to use: the first depth image which has the mean of the queried mean point $\mu(t = t_i)$, inside its FOV. This equates to calling only a fast subroutine, `IsInFOV()`, for each node of the depth-pose-graph searched. This subroutine is fast, as shown below, where π is the projective transform, $(u, v) = (\frac{x}{z}, \frac{y}{z})$, in the right-down-forward Cartesian frame of the depth image, and $(x, y, z) = K\mu$ with K the camera intrinsics matrix.

$$\texttt{IsInFOV()} = \begin{cases} true, & \text{if } 0 < u < r \text{ and} \\ & \quad 0 < v < c \text{ and} \\ & \quad z < d_{r,c} \text{ and} \\ & \quad z < R_{range} \text{ and} \\ & \quad z > 0 \\ false, & \text{otherwise} \end{cases} \tag{4.2}$$

In some cases, multiple depth images may all have a view of a particular subset of space. Each depth-pose could be used to contribute to the overall evaluation, but it is helpful to limit the expensive querying of each k-d tree (which for 160x120 images, has 19,200 nodes) to only when the depth image is the most useful.

We use a hierarchy of breadth-first-search (BFS) graph search and spatial partitioning for a fast use of the depth-pose-graph. In the greedy search, the first depth image found that evaluates `IsInFOV()` $= true$ is used. This method is depicted in the figure below.

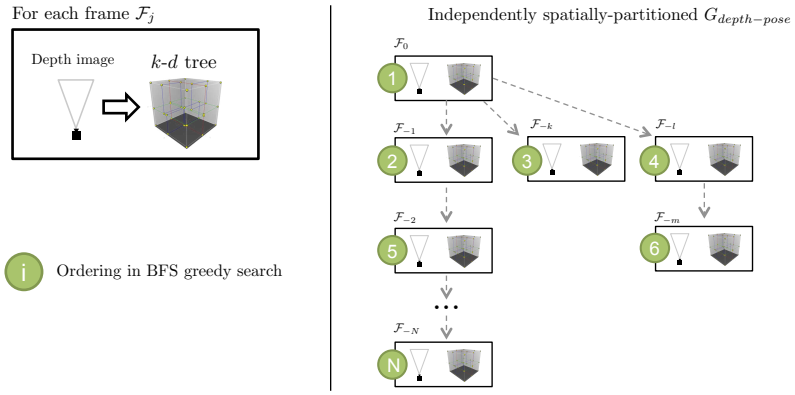


Figure 4-4: Greedy search on the depth-pose-graph.

When there are no loop closures in the pose graph, the greedy search reduces to a search down an array of depth-pose nodes, where temporal preference (more recent is better) is used as a heuristic for which depth image to use.

4.4.3 Evaluating Motions With A Depth-Pose-Graph

We consider a type of sampling-based motion planning with a depth-pose-graph, in which a discrete library of motions is evaluated. Previous works have had success using a small library of motion primitives and replanning at high rate (i.e., 30-60 Hz) [11, 12]. The motion primitives we consider are open-loop input trajectories, of the form $\mathbf{u}(t)$ over some finite horizon $t \in [0, N]$, and are executed in a Model-Predictive-Control (MPC) type fashion, without any position-tracking controller [11].

A key step is to determine a time-varying distribution of configuration, for any motion primitive. (Since only configuration space matters for collisions, only a time-varying distribution of configuration is needed, rather than full state.) We consider a finite set of K open-loop input trajectories, $\mathbb{U} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K\}$. Future states are determined with the given dynamical model, $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$. In the current frame of the robot $\mathcal{F}_0$, there is no initial position or orientation uncertainty. Accordingly to determine the time-varying distribution of configuration in frame $\mathcal{F}_0$, for a second order system only the velocity estimates are needed. With a current velocity estimate $\tilde{\mathbf{v}}_0$, together with the open-loop input trajectory, a time-varying distribution of configuration can be determined $\{\tilde{\mathbf{v}}_0, \mathbf{u}(t)\} \rightarrow p_t(\mathbf{q})$.

Unless we are performing Monte Carlo evaluation[1], we need to sample the time-varying configuration distributions over a sequence of times:

$$\begin{array}{c} \{\tilde{\mathbf{v}}_0, \mathbf{u}(t)\} \quad \underset{\text{sample over times } t_1, t_2, ..., t_N}{\rightarrow} \\ p_{t_1}(\mathbf{q}), p_{t_2}(\mathbf{q}), ..., p_{t_N}(\mathbf{q}) \quad \text{all in frame } \mathcal{F}_0 \end{array} \tag{4.3}$$

[1] As mentioned in the previous chapter, Monte Carlo evaluation was deemed too slow for our purposes, although recent work has shown that highly parallel implementations of Monte Carlo can enable reasonable computation times [110].

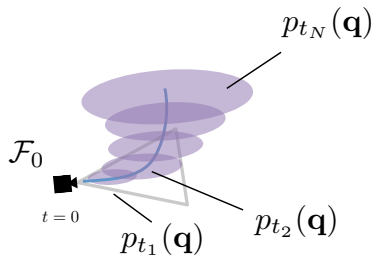


Figure 4-5: In current frame $\mathcal{F}_0$, sampling time-varying distribution of configuration at times $t_1, t_2, ..., t_N$. The FOV of the depth sensor is shown with the gray triangle.

We then use a simple independence approximation to combine evaluations of multiple times in order to evaluate the whole trajectory. This is not a perfect approximation, and its accuracy limitations have been previously addressed [71], but empirically it has shown to be useful [11]. With the independence approximation, the probability of collision for the whole trajectory is

$$P_{collision}(p_t(\mathbf{q})) \approx 1 - \prod_{i=0}^{N} \Big(1 - P_{collision}(p_{t_i}(\mathbf{q}))\Big) \tag{4.4}$$

The problem then reduces to determining the $P_{collision}$ at each time step, for which the entire depth-pose-graph may be used:

$$P_{collision}\big(p_{t_i}(\mathbf{q}) \mid G_{depth-pose}\big) \tag{4.5}$$

The efficient, uncertainty-conscious evaluation of the above expression is the central part of this work. The previous subsection described searching the depth-pose-graph – once the appropriate depth-pose has been found, its frame-specific uncertainty is determined as described in the next section.

4.4.4 Determining Frame-Specific Uncertainty

Each time-sampled configuration distribution is originally considered always in the current robot frame $\mathcal{F}_0$, and so transforming this distribution through the uncertain pose graph edges $\tilde{x}_{i,j}$ creates a different uncertainty distribution ${}^{\mathcal{F}_j}p_{t_i}(\mathbf{q})$ in the frame $\mathcal{F}_j$ of each different pose.

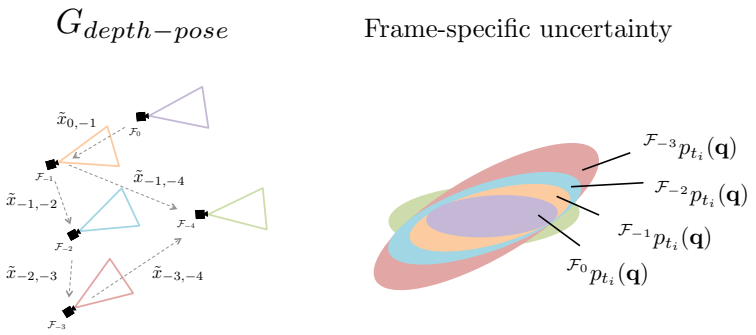


Figure 4-6: The uncertain frame transformations $\tilde{x}_{i,j}$ are used to determine the frame-specific time-sampled distributions of configuration.

The colors in the above depiction match each pose (and corresponding depth image) with their respective uncertainty of the time-sampled distribution of configuration.

We make the simplifying assumption that rotations are known, and there is only uncertainty is in the transformations. Although incorporating rotational uncertainty would be more rigorous, this assumption is actually the same made for least-squares-based pose graph SLAM, which is commonly used.

In the current frame $\mathcal{F}_0$, for a Linear-Gaussian model we have mean $\mu(t)$ and covariance $\Sigma(t)$:

$$^{\mathcal{F}_0}\mathbf{p}_i(t) \sim \mathcal{N}\Big(\mu(t), \Sigma(t)\Big) \tag{4.6}$$

Now for frame $\mathcal{F}_j$ associated with pose j we incorporate the mean translation t_{0j} and its covariance Σ_{0j}, which both sum (they are Linear-Gaussian):

$$^{\mathcal{F}_j}\mathbf{p}_i(t) \sim \mathcal{N}\Big(\mu(t) + t_{0j}, \Sigma(t) + \Sigma_{0j}\Big) \tag{4.7}$$

This completes the determination of frame-specific configuration uncertainty.

4.4.5 Evaluation Within Each Frame, With Inverse-Depth Gaussian Noise

Given the frame-specific configuration uncertainty, the task is to evaluate the probability of collision, given only the depth image in that frame. Previous work has addressed fast planning with current-depth-image-only FOV constraints accelerated

by spatial partitioning [11]. We offer an improvement over this previous method for handling one depth image – we now incorporate a model of depth image uncertainty. We do this by adding an uncertainty component to the covariance which is only in the direction of depth – in a right-down-forward convention of the Cartesian frame of a depth image, this corresponds to a 3×3 covariance Σ_{depth} whose only non-zero component is the bottom right entry. We use inverse-depth Gaussian noise, so if the depth at a particular pixel is $d_{r,c}$, we have:

$$\Sigma_{depth_{3,3}} = \frac{1}{\frac{1}{d_{r,c}} + \sigma}$$

where σ can be determined for the specific sensor. This is added to the other two sources of uncertainty previously discussed:

$$\Sigma(t) + \Sigma_{0j} + \Sigma_{depth}$$

Although inverse-depth Gaussian noise does not perfectly model the real failure modes of depth sensors (which may have binary "don't see object at all" failure modes rather than a Gaussian decay of depth uncertainty), it is an improved model over modeling no depth image uncertainty. Operating in inverse-depth is a well-established technique when dealing with range data computed from triangulation, for example in stereo/monocular vision [111].

4.4.6 Marginalization and Pruning of the Pose Graph

For computational purposes it is useful to not keep a fully dense pose graph. Local obstacle avoidance does not require a long history of depth information – only a short history is necessary. Specifically, we consider a small handful of seconds (10) to be sufficient. Given a framerate frequency $f_{framerate}$ in Hz, this would be $10 \times f_{framerate}$ depth images, but not every depth image is needed. Instead, we consider that remembering only 5 Hz of depth information is sufficient. Thus our depth-pose-graph will have 50 nodes, which is reasonably small complexity.

Although a back-end graph optimization tool is not required for our approach, and is not involved in the critical-path latency (complex graphs can sometimes take seconds to solve). To reduce a high framerate of depth-pose nodes down to 5 Hz requires pruning – but rather than throw away some of the odometry information, it is better to perform marginalization of the full pose graph. Sparsification can leverage well-developed methods for marginalization from SLAM back-ends, which can run in a parallel thread to our planning thread.

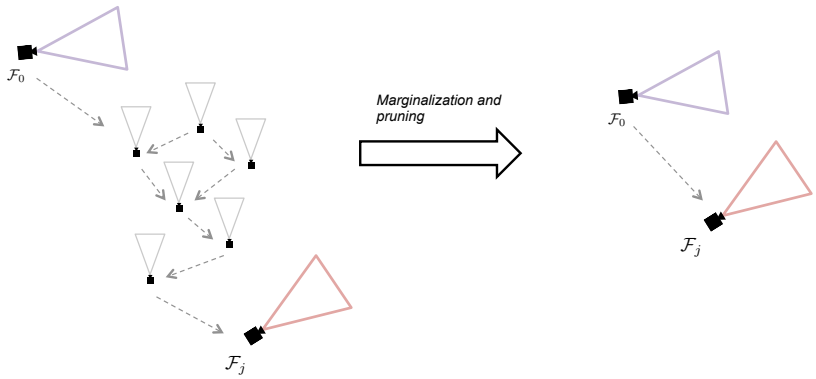


Figure 4-7: Sparsification of a subgraph of a pose graph, which can be provided by a SLAM back-end optimizer.

4.4.7 Quadrotor Obstacle Avoidance with a Triple Integrator Model

For the robot of interest, a specific Linear-Gaussian dynamics model needs to be chosen. We use a previously described quadrotor model, in which the closed loop around the inner-loop attitude controller is approximated as a triple integrator, as described in previous work [11]. This triple-integrator Gaussian-noise model is:

$$\mathbf{p}_i(t) \sim \mathcal{N}\left(\frac{1}{6}\mathbf{j}_i t^3 + \frac{1}{2}\mathbf{a}_0 t^2 + \mathbf{v}_{0,\mu} t, t^2 \Sigma_{v_0}\right) \quad \forall t \in [0, t_{jf}] \tag{4.8}$$

4.4.8 Collision-Probability-Constrained Motion Library

We consider a particular F (see the Problem Formulation), similar to chance-constrained programming, which favors the maximum reward policy R_{nav} that meets a collision

constraint[2]. The collision constraint, however, is defined for the entire policy, rather than any specific time (as is done in chance-constrained programming). If the collision constraint cannot be met, we choose the minimum-$P_{collision}$ policy.

$$\pi^* = \begin{cases} \underset{\pi_i}{\text{argmax}} \;\; \mathcal{R}_{nav}(\pi_i), & \forall\ \pi_i \text{ with } P_{collision}(\pi_i) < \epsilon \\ \underset{\pi_i}{\text{argmin}} \;\; P_{collision}(\pi_i), & \text{if} \quad \nexists\ \pi_i \text{ with } P_{collision}(\pi_i) < \epsilon \end{cases} \tag{4.9}$$

With the chance-constrained formulation, an arbitrary $\mathcal{R}_{nav}$ will not allow purposefully allow the collision probability to exceed some threshold ϵ.

4.5 Comparison with Memoryless and Maximum-likelihood Mapping Approaches

Now that Depth-Pose-Graph Planning has been formulated, and we have provided a complete description of a Greedy Depth-Pose-Graph Planning algorithm, we analyze its properties compared to the two alternatives mentioned: (i) memoryless current-depth-image-only planning, and (ii) maximum-likelihood mapping-based planning.

First, we ignore computation time, and analyze their properties in the absence of computational limits. Then we perform time complexity analysis. We note that space complexity is not a limit in practice: with 32 GB of RAM available on our hardware platform, the entirety of sensor data can be stored in memory for many minutes before running out of space. As mentioned before, for obstacle avoidance we only care about a small number of seconds of memory, which is of insignificant size.

Comparison of properties, ignoring computational complexity

First, we note that both (i) memoryless planning, and (ii) maximum-likelihood mapping-based planning, can both be reduced from the full information of (iii) planning with

[2]In previous work, we have used unconstrained objective functions, but the problem is potentially cleaner with a constrained objective. An unconstrained objective is subject to relative scaling problems, whereas in a constrained objective the navigation function $\mathcal{R}_{nav}$ can be scaled arbitrarily.

a depth-pose-graph. In other words, the entirety of information available to both of these other approaches is only a subset of information available to planning with a depth-pose-graph. In that sense, planning with a depth-pose-graph is strictly superior, modulo computational complexity. This is not true, however, of the Greedy Depth-Pose-Graph Planning method, which is the tractable variant presented.

One potentially useful aspect of mapping-based planning is the sensor fusion of depth data over multiple time steps. In general, however, planning with a depth-pose-graph can also achieve combining data over multiple points in time. The depth-pose-graph framework even allows the opportunity to selectively only do sensor fusion – "lazy sensor fusion" – which is not a property of a mapping-based approach, which exhaustively fuses sensor data even if it is never used for the planning system.

Another key distinction is that mapping-based planning methods offer the ability to do discrete-space planning, which can leverage discrete planning algorithms such as A^*. This may seem like a disadvantage of depth-pose-graph approaches, but upon further analysis, this is not much of a limitation. For one, discrete-space planning approaches do not offer great ways of handling dynamic constraints of a robot, which is critical for obstacle avoidance. Additionally, depth-pose-graph methods can still use discrete planning tools, but over a graph of motion primitives rather than over graph of discretized space. Sampling-based planning is fully available to all three of these methods.

Finally, we note that, ignoring uncertainty, the raw frustum information of depth images is closer to "polygon world" that classical planning approaches would refer to operate with. If uncertainty is either bounded or eliminated from the situation, then path collision evaluation can be performed without sampling (a whole continuous segment of a path through configuration space can be said to be obstacle-free). This is a nice property of depth-pose-graphs which may be attractive in particular situations.

Comparison of computational complexity

The comparison of computational complexity depends on a number of parameters. We note them below:

Sensor and memory parameters

- $N_{pixels} = r \times c = \#$ pixels in each depth image
- $N_{mem} = \#$ depth images to remember

Motion planning parameters

- $N_{samples} = \#$ time-sampled configurations for motion planning

Mapping parameters

- $N_{voxels} = \#$ voxels in discretized map

For (i), the computational complexity of memoryless planning as described in [11], we have the following expression. In our implementation, with $N_{pixels} = 19,200$, F_1 is dominated by k-d tree building, and F_2 is dominated k-d tree lookup. Our total latency has previously been measured at $\sim$2 ms:

$$F_1(N_{pixels}) + N_{samples} * F_2(N_{pixels})$$

For (ii), the computational complexity of mapping-based planning, there is a sizable difference in whether the map is built incrementally, or in a batch fashion (which is required in order to allow loop closures). When the map is built incrementally, we have the following expression, where F_3 represents map-building, and the motion-planning latency is assumed to be dominated by collision-detection in the map structure.

$$F_3(N_{pixels}, N_{voxels}) + N_{samples} \times K_{lookup}$$

We have previously measured the above latency on the order of 100 ms for the first term only, and this does not allow for loop closures. When batched maps, rather than incremental maps, are used, the latency increases significantly:

$$F_3(N_{pixels} \times N_{mem}, N_{voxels}) + N_{samples} \times K_{lookup}$$

For (iii), greedy planning with a depth-pose-graph, we have a similar complexity to (i), with only the addition of another term, where $K_{FOVcheck}$ is the fast `IsInFOV()` subroutine described before.

$$F_1(N_{pixels}) + N_{samples} \times N_{mem} \times K_{FOVcheck} + N_{samples} * F_2(N_{pixels})$$

Initial experimentation has suggested that, due to $K_{FOVcheck}$ being small, that the total worst-case latency of planning with $N_{mem} = 50$ rather than 1, only increases the total latency to $\sim$3 ms.

4.6 Simulation Experiments

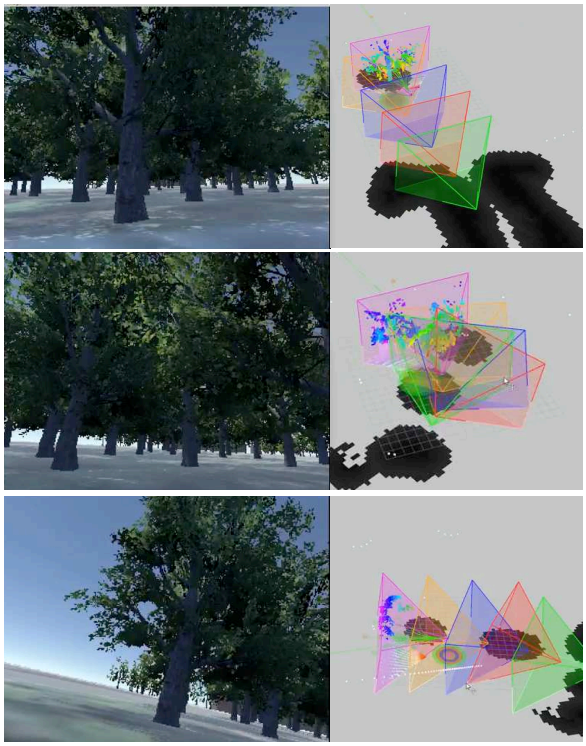


Figure 4-8: Screen capture from three consecutive moments of a simulated quadrotor flying through a forest using our method.

Initial simulation experiments have demonstrated the robust ability of a field-of-view constrained quadrotor to navigate complex environments (Figure 4-8). Among the types of maneuvers that are capable with robustly adding memory: (i) the quadro-

tor can pass by a tree, and remember that it was on its left, not its right, (ii) fast horizontal obstacle avoidance can strafe, without needing first to yaw, (iii) during decelerations while the FOV is pointed up in the air due to vehicle's pitch, it can remember the obstacle in front of it. These types of behaviors make the vehicle's motion more graceful, and even under pose estimation noise allows the vehicle to reason about which areas outside its current FOV are safe and which are not.

4.7 Discussion

We have presented a formulation that efficiently applies a robust planning viewpoint to planning with memory, and initial simulation experiments have suggested its practical capabilities. Testing this approach in hardware is future work that we hope will be under way soon.

There are a number of other areas in which this work could be expanded. Although a Depth-Pose-Graph Planning framework coupled with a sampling-based exploration strategy such as a PRM [49] or RRT [50] offers the ability to navigate arbitrary maze-type environments, the performance problem we are most interested in is local robust collision avoidance, rather than global minimum-cost planning.

It would be ideal, as an alternative to greedy search, to be able to use a spatial partitioning of depth image views in order to determine which depth image to use. Unfortunately, although there exist fast methods for querying distances to convex polyhedra [112] and spatially partitioning them [113], the polyhedra of free space associated with each depth image has many faces, is not necessarily convex. Since we can limit the size of the graph with pruning and marginalization we resort to greedy search.

Our current implementation only simulates noise that would come from a real SLAM front-end, and so it would be of course interesting to integrate with a real SLAM framework. We also discussed but have not implemented the use of a back-end to marginalize a pose graph. It would be interesting to explore what are good metrics for choosing which keyframes are good to keep – this is a different problem

than just pure pose-graph SLAM, since we want to keep view cones of free space, not just minimize maximum-likelihood estimation error.

Chapter 5

Discussion and Future Work

We have presented a novel approach to collision avoidance, demonstrated the method in hardware, and extended it to address the first iteration's largest opportunity for improvement to robustly incorporate memory. Of note, we have demonstrated for the first time the basic viability of a UAV navigating among obstacles using planning-based methods but only executing commands via model predictive control, without any tracking controller. Further, the results are among the fastest and most robust results ever demonstrated for a comparable system. More generally applicable across robotics platforms, we have demonstrated the utility of an approach where perception and control are closely integrated – in particular, when control specifically plays to the strengths of available sensor data, while still applying rigorous techniques.

A surprisingly interesting question to reflect on has been: *what did we think was going to be a problem, but turned out to not be a primary issue?* There are a variety of questions that, before the hardware results were obtained, would have seemingly been questions of primary concerns. This included: will a control approach that is so model-dependent, and has no trajectory-tracking controller even work? Especially when there has been essentially no proper system identification performed? Surprisingly, not once in hardware testing have we thought that we are limited by the inaccuracy of our dynamic model. This speaks potentially to the benefits of planning for robustness (the cushion allowed for velocity estimation also provides a cushion for imperfections of the dynamic model), but also may speak to the simplicity of the dynamics of the

quadrotor.

Additionally, after formulating the technique and its first implementation in software, an area we thought would have been low-hanging fruit for dramatically improving performance was massively parallel implementations on a GPU. Such implementations could allow us to increase by orders of magnitude the number of motion primitives evaluated, and also enable improved path collision probability approximations. It turns out, though, that a very small number of motion primitives has done quite well for us, and it is unclear if orders of magnitude more would have significantly improved performance. Another area that we would have expected to be an exciting and rich next step was fully exploring 3D flight. Although not presented, we actually have implemented and tested in hardware full 3D flight with 3D motion primitive libraries. There have been a number of issues, however. The primary issue was perception – the only sensor we had that could reliably detect low-texture surfaces was our 2D laser, and so we had to limit flight to 2D without a fully reliable 3D sensor. The constraints of the field of view are also more difficult to manage for 3D flight – our new implementation incorporating memory, however, should help with this limitation, as we can explore more in future testing. It also has turned out that fast, aggressive 2D flight still is an exciting and difficult problem to work on.

Robustness to wind and disturbances is also something that our method does not address. This is not an issue in indoor environments, but surprisingly it has turned out that in approximately 100 flights outdoors, in moderate wind less than 20 mph, on only a few flights has the lack of wind-disturbance rejection been an issue for a flight. Implementing an observer to estimate the wind online would enable robustness to wind.

From the overall perspective of our vehicle's flight performance and where the biggest opportunities for improvement lie for improving robust and agile UAV flight, planning is not the primary roadblock. Perception is probably the biggest area of opportunity. State estimation is also difficult, but there appears to be a clear path to making VIO estimation incrementally better. Perception has more frontiers – for example, achieving human-level scene understanding, using not just depth information

but raw RGB vision information.

In terms of particular future work, a couple areas are outlined below.

Integrated Perception and Control on other Platforms

The overall approach of closely integrating perception and control, and attempting to break de facto trends in separating them, is a concept that offers opportunities for other systems in addition to UAVs – for example, manipulation and walking. In manipulation, rather than attempting the solve the grasping problem in one global frame with full state estimation, but also rather than trying to only use image-coordinate visual servoing, approaches could be developed that apply rigorous state-space tools yet play to the strengths of actual sensor information. This could for example be exciting to incorporate in approaches that combine both visual and tactile sensing.

Robust Perception

In our approach, we have primarily focused on robustness to state estimation, but a key assumption has been that each depth image is a dependable source of information. In Chapter 4, we added a modification to allow for inverse-depth Gaussian noise in the depth image, and although this is preferable to modeling no uncertainty in the depth image, it is far from perfect. Empirically, the failure modes of depth sensors are not due to easily-modeled Gaussian decay. Rather, due to poor lighting conditions, or low surface texture, a vision-based sensor may entirely miss an object, and think an entire area of the depth image is empty of objects, which is non-ideal for obstacle avoidance. Even laser-based depth sensors also have varying sensing quality based on environmental conditions and surface textures. Increasing our capabilities to understand the limits of perception systems, and using that understanding to increase the robustness of such systems, is an exciting route for further research.

Bibliography

[1] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[2] James M Anderson, Kalra Nidhi, Karlyn D Stanley, Paul Sorensen, Constantine Samaras, and Oluwatobi A Oluwatola. *Autonomous vehicle technology: A guide for policymakers*. Rand Corporation, 2014.

[3] Andrew J Barry. High-speed autonomous obstacle avoidance with pushbroom stereo. PhD Thesis, MIT, 2016.

[4] Rudolf Emil Kalman et al. Contributions to the theory of optimal control.

[5] Geir E Dullerud and Fernando Paganini. *A course in robust control theory: a convex approach*, volume 36. Springer Science & Business Media, 2013.

[6] Tryphon T Georgiou and Anders Lindquist. The separation principle in stochastic control, redux. *IEEE Transactions on Automatic Control*, 58(10):2481–2494, 2013.

[7] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[8] Quentin Bateux and Eric Marchand. Histograms-based visual servoing. *IEEE Robotics and Automation Letters*, 2(1):80–87, 2017.

[9] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[10] Shreyansh Daftry, Sam Zeng, Arbaaz Khan, Debadeepta Dey, Narek Melik-Barkhudarov, J Andrew Bagnell, and Martial Hebert. Robust monocular flight in cluttered outdoor environments. *arXiv preprint arXiv:1604.04779*, 2016.

[11] Pete Florence, John Carter, and Russ Tedrake. Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps. In *Algorithmic Foundations of Robotics XII*. 2016.

[12] Brett T. Lopez and Jonathan How. Aggressive 3-d collision avoidance for high-speed navigation. In *ICRA (Accepted but not published*, 2017.

[13] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Vision-based state estimation for autonomous rotorcraft mavs in complex environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1758–1764. IEEE, 2013.

[14] Davide Falanga, Elias Mueggler, Matthias Faessler, and Davide Scaramuzza. Aggressive quadrotor flight through narrow gaps with onboard sensing and computing. *arXiv preprint arXiv:1612.00291*, 2016.

[15] Antoine Beyeler, Jean-Christophe Zufferey, and Dario Floreano. Vision-based control of near-obstacle flight. *Autonomous robots*, 27(3):201–219, 2009.

[16] Michael W Otte, Scott G Richardson, Jane Mulligan, and Gregory Grudic. Path planning in image space for autonomous robot navigation in unstructured environments. *Journal of Field Robotics*, 26(2):212–240, 2009.

[17] Larry Matthies, Roland Brockers, Yoshiaki Kuwata, and Stephan Weiss. Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3242–3249. IEEE, 2014.

[18] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011.

[19] Sebastian Scherer, Sanjiv Singh, Lyle Chamberlain, and Mike Elgersma. Flying fast and low among obstacles: Methodology and experiments. *The International Journal of Robotics Research*, 27(5):549–574, 2008.

[20] Stefan Hrabar and Gaurav Sukhatme. Vision-based navigation through urban canyons. *Journal of Field Robotics*, 26(5):431–452, 2009.

[21] Joseph Conroy, Gregory Gremillion, Badri Ranganathan, and J Sean Humbert. Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. *Autonomous robots*, 27(3):189–198, 2009.

[22] Andrew M Hyslop and J Sean Humbert. Autonomous navigation in three-dimensional urban environments using wide-field integration of optic flow. *Journal of guidance, control, and dynamics*, 33(1):147–159, 2010.

[23] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert. Learning monocular reactive uav control in cluttered natural environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1765–1772. IEEE, 2013.

[24] Huili Yu and Randy Beard. A vision-based collision avoidance technique for micro air vehicles using local-level frame mapping and path planning. *Autonomous Robots*, 34(1-2):93–109, 2013.

[25] Torsten Merz and Farid Kendoul. Dependable low-altitude obstacle avoidance for robotic helicopters operating in rural areas. *Journal of Field Robotics*, 30(3):439–471, 2013.

[26] Eric N Johnson and John G Mooney. A comparison of automatic nap-of-the-earth guidance strategies for helicopters. *Journal of Field Robotics*, 31(4):637–653, 2014.

[27] Helen Oleynikova, Dominik Honegger, and Marc Pollefeys. Reactive avoidance using embedded stereo vision for mav flight. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 50–56. IEEE, 2015.

[28] Matthias Nieuwenhuisen and Sven Behnke. Hierarchical planning with 3d local multiresolution obstacle avoidance for micro aerial vehicles. In *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*, pages 1–7. VDE, 2014.

[29] Debadeepta Dey, Kumar Shaurya Shankar, Sam Zeng, Rupesh Mehta, M Talha Agcayazi, Christopher Eriksen, Shreyansh Daftry, Martial Hebert, and J Andrew Bagnell. Vision and learning for deliberative monocular cluttered flight. In *Field and Service Robotics*, pages 391–409. Springer, 2016.

[30] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon" next-best-view" planner for 3d exploration. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1462–1468. IEEE, 2016.

[31] Jing Chen, Tianbo Liu, and Shaojie Shen. Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1476–1483. IEEE, 2016.

[32] Sikang Liu, Michael Watterson, Sarah Tang, and Vijay Kumar. High speed navigation for quadrotors with limited onboard sensing. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1484–1491. IEEE, 2016.

[33] Abraham Bachrach, Samuel Prentice, Ruijie He, Peter Henry, Albert S Huang, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Estimation, planning, and mapping for autonomous flight using an rgb-d camera in gps-denied environments. *The International Journal of Robotics Research*, 31(11):1320–1343, 2012.

[34] I. Lenz, M. Gemici, and A. Saxena. Low-power parallel algorithms for single image based obstacle avoidance in aerial robots. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 772–779, Oct 2012.

[35] Brian MacAllister, Jonathan Butzke, Alex Kushleyev, Harsh Pandey, and Maxim Likhachev. Path planning for non-circular micro aerial vehicles in constrained environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3933–3940. IEEE, 2013.

[36] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4557–4564. IEEE, 2012.

[37] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran. Continuous-time trajectory optimization for online uav replanning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5332–5339, Oct 2016.

[38] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 528–535. IEEE, 2016.

[39] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[40] Ruijie He, Sam Prentice, and Nicholas Roy. Planning in information space for a quadrotor helicopter in a gps-denied environment. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1814–1820. IEEE, 2008.

[41] Adam Bry, Abraham Bachrach, and Nicholas Roy. State estimation for aggressive flight in gps-denied environments using onboard sensing. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1–8. IEEE, 2012.

[42] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research*, pages 649–666. Springer, 2014.

[43] Benoit Landry, Robin Deits, Peter R Florence, and Russ Tedrake. Aggressive quadrotor flight through cluttered environments using mixed integer programming. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016.

[44] Michael Burri, Helen Oleynikova, Markus W Achtelik, and Roland Siegwart. Real-time visual-inertial mapping, re-localization and planning onboard mavs

in unknown environments. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1872–1878. IEEE, 2015.

[45] Aditya A Paranjape, Kevin C Meier, Xichen Shi, Soon-Jo Chung, and Seth Hutchinson. Motion primitives and 3d path planning for fast flight through a forest. *The International Journal of Robotics Research*, 34(3):357–377, 2015.

[46] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Auton. Robots*, 34(3):189–206, April 2013.

[47] Matthias Nieuwenhuisen and Sven Behnke. 3d planning and trajectory optimization for real-time generation of smooth mav trajectories. In *Mobile Robots (ECMR), 2015 European Conference on*, pages 1–7. IEEE, 2015.

[48] Robin Deits and Russ Tedrake. Efficient mixed-integer planning for uavs in cluttered environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–49. IEEE, 2015.

[49] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[50] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[51] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[52] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International journal of robotics research*, 34(7):883–921, 2015.

[53] Tom Schouwenaars, Bart De Moor, Eric Feron, and Jonathan How. Mixed integer programming for multi-vehicle path planning. In *Control Conference (ECC), 2001 European*, pages 2603–2608. IEEE, 2001.

[54] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 477–483. IEEE, 2012.

[55] Karl Johan Aström and Richard M Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[56] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *CoRR*, abs/1601.04037, 2016.

[57] Robin Ritz and Raffaelo D'Andrea. A global strategy for tailsitter hover control. *International Symposium on Robotics Research (ISRR)*, 2015.

[58] Joseph Moore, Rick Cory, and Russ Tedrake. Robust post-stall perching with a simple fixed-wing glider using lqr-trees. *Bioinspiration & biomimetics*, 9(2):025013, 2014.

[59] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[60] Yuanfu Luo, Haoyu Bai, David Hsu, and Wee Sun Lee. Importance sampling for online planning under uncertainty.

[61] Selim Temizer, Mykel Kochenderfer, Leslie Kaelbling, Tomas Lozano-Pérez, and James Kuchar. Collision avoidance for unmanned aircraft using markov decision processes. In *AIAA guidance, navigation, and control conference*, page 8040.

[62] Haoyu Bai and David Hsu. Unmanned aircraft collision avoidance using continuous-state pomdps. *Robotics: Science and Systems VII*, 1:1–8, 2012.

[63] Romain Pepy and Alain Lambert. Safe path planning in an uncertain-configuration space using rrt. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5376–5381. IEEE, 2006.

[64] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 723–730. IEEE, 2011.

[65] Brendan Burns and Oliver Brock. Sampling-based motion planning with sensing uncertainty. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3313–3318. IEEE, 2007.

[66] Jur Van Den Berg, Pieter Abbeel, and Ken Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, 2011.

[67] Samuel Prentice and Nicholas Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, 2009.

[68] Patrycja E Missiuro and Nicholas Roy. Adapting probabilistic roadmaps to handle uncertain maps. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1261–1267. IEEE, 2006.

[69] Robert Platt Jr, Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. 2010.

[70] Shridhar K Shah, Chetan D Pahlajani, Nicholaus A Lacock, and Herbert G Tanner. Stochastic receding horizon control for robots with probabilistic state constraints. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2893–2898. IEEE, 2012.

[71] Lucas Janson, Edward Schmerling, and Marco Pavone. Monte carlo motion planning for robot trajectory optimization under uncertainty. *International Symposium on Robotics Research (ISRR)*, 2015.

[72] Sachin Patil, Jur Van Den Berg, and Ron Alterovitz. Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3238–3244. IEEE, 2012.

[73] Wen Sun, Luis G Torres, Jur Van Den Berg, and Ron Alterovitz. Safe motion planning for imprecise robotic manipulators by minimizing probability of collision. In *Robotics Research*, pages 685–701. Springer, 2016.

[74] Jia Pan, Sachin Chitta, and Dinesh Manocha. Probabilistic collision detection between noisy point clouds using robust classification. In *International Symposium on Robotics Research (ISRR)*, pages 1–16, 2011.

[75] Jia Pan and Dinesh Manocha. Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing. *The International Journal of Robotics Research*, page 0278364916640908, 2016.

[76] Abraham Charnes and William W Cooper. Chance-constrained programming. *Management science*, 6(1):73–79, 1959.

[77] Lars Blackmore, Hui Li, and Brian Williams. A probabilistic approach to optimal robust path planning with obstacles. In *American Control Conference, 2006*, pages 7–pp. IEEE, 2006.

[78] Lars Blackmore, Masahiro Ono, Askar Bektassov, and Brian C Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE transactions on Robotics*, 26(3):502–517, 2010.

[79] Lars Blackmore, Masahiro Ono, and Brian C Williams. Chance-constrained optimal path planning with obstacles. *IEEE Transactions on Robotics*, 27(6):1080–1094, 2011.

[80] Brandon Luders, Mangal Kothari, and Jonathan P How. Chance constrained rrt for probabilistic robustness to environmental uncertainty. In *AIAA Guidance, Navigation, and Control Conference, Guidance, Navigation, and Control*, 2010.

[81] Georges S Aoude, Brandon D Luders, Joshua M Joseph, Nicholas Roy, and Jonathan P How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35(1):51–76, 2013.

[82] Mangal Kothari and Ian Postlethwaite. A probabilistically robust path planning algorithm for uavs using rapidly-exploring random trees. *Journal of Intelligent & Robotic Systems*, pages 1–23.

[83] Thierry Fraichard and Hajime Asama. Inevitable collision states?a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[84] B. L?Espérance and K. Gupta. Safety hierarchy for planning with time constraints in unknown dynamic environments. *IEEE Transactions on Robotics*, 30(6):1398–1411, Dec 2014.

[85] Raia Hadsell, Pierre Sermanet, Marco Scoffier, Ayse Erkan, Koray Kavackuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, February 2009. Video.

[86] David C Moore, Albert S Huang, Matthew Walter, Edwin Olson, Luke Fletcher, John Leonard, and Seth Teller. Simultaneous local and global state estimation for robotic navigation. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3794–3799. IEEE, 2009.

[87] Mihail Pivtoraiko, Daniel Mellinger, and Vijay Kumar. Incremental micro-uav motion replanning for exploring unknown environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2452–2458. IEEE, 2013.

[88] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE transactions on robotics*, 21(6):1077–1091, 2005.

[89] Debadeepta Dey, Tian Y Liu, Boris Sofman, and Drew Bagnell. Efficient optimization of control libraries. Technical report, DTIC Document, 2011.

[90] Alessandro Giusti, Jérôme Guzzi, Dan C Cireşan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.

[91] Andrew J Barry and Russ Tedrake. Pushbroom stereo for high-speed navigation in cluttered environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3046–3052. IEEE, 2015.

[92] Noel E Du Toit and Joel W Burdick. Probabilistic collision checking with chance constraints. *IEEE Transactions on Robotics*, 27(4):809–815, 2011.

[93] Masahiro Ono, Marco Pavone, Yoshiaki Kuwata, and J Balaram. Chance-constrained dynamic programming with application to risk-aware robotic space exploration. *Autonomous Robots*, 39(4):555–571, 2015.

[94] Anirudha Majumdar and Russ Tedrake. Robust online motion planning with regions of finite time invariance. In *Algorithmic Foundations of Robotics X*, pages 543–558. Springer, 2013.

[95] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 2010.

[96] Charles Richter, William Vega-Brown, and Nicholas Roy. Bayesian learning for safe high-speed navigation in unknown environments. In *Proceedings of the International Symposium on Robotics Research (ISRR 2015)*, Sestri Levante, Italy, 2015.

[97] Russ Tedrake. Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems, 2014.

[98] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *Int. J. Rob. Res.*, 31(5):664–674, April 2012.

[99] Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Auton. Robots*, 33(1-2):21–39, August 2012.

[100] Damien Dusha and Luis Mejias. Error analysis and attitude observability of a monocular gps/visual odometry integrated navigation filter. *The International Journal of Robotics Research*, 31(6):714–737, 2012.

[101] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces.

[102] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[103] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In *Advances in neural information processing systems*, pages 1772–1780, 2013.

[104] Rafael Valencia, Marti Morta, Juan Andrade-Cetto, and Josep M Porta. Planning reliable paths with pose slam. *IEEE Transactions on Robotics*, 29(4):1050–1059, 2013.

[105] Ernesto H Teniente, Rafael Valencia, and Juan Andrade-Cetto. Dense outdoor 3d mapping and navigation with pose slam. 2011.

[106] Vadim Indelman, Luca Carlone, and Frank Dellaert. Towards planning in generalized belief space. In *Robotics Research*, pages 593–609. Springer, 2016.

[107] Jason Israelsen, Matt Beall, Daman Bareiss, Daniel Stuart, Eric Keeney, and Jur van den Berg. Automatic collision avoidance for manually tele-operated unmanned aerial vehicles. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6638–6643. IEEE, 2014.

[108] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[109] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. Elasticfusion: Dense slam without a pose graph. In *RSS 2015*.

[110] Brian Ichter, Edward Schmerling, Ali-akbar Agha-mohammadi, and Marco Pavone. Real-time stochastic kinodynamic motion planning via multiobjective search on gpus. *arXiv preprint arXiv:1607.06886*, 2016.

[111] Javier Civera, Andrew J Davison, and JM Martinez Montiel. Inverse depth parametrization for monocular slam. *IEEE transactions on robotics*, 24(5):932–945, 2008.

[112] Ming C Lin and John F Canny. A fast algorithm for incremental distance calculation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1008–1014. IEEE, 1991.

[113] Stephen A Ehmann and Ming C Lin. Accelerated proximity queries between convex polyhedra by multi-level voronoi marching. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 3, pages 2101–2106. IEEE, 2000.