Integrated Perception and Control at High Speed: Evaluating Collision Avoidance Maneuvers Without Maps

Pete Florence[1], John Carter[1], and Russ Tedrake[1]

MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA
{peteflo,jcarter,russt}@csail.mit.edu

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. Additionally, we present approximation methods augmented with spatial partitioning data structures that enable low-latency, real-time reactive control. The probabilistic formulation provides a natural way to integrate reactive obstacle avoidance with arbitrary navigation objectives. We validate the method using a simulated quadrotor race through a forest at high speeds in the presence of increasing state estimate noise. We pair our method with a motion primitive library and compare with a global path-generation and path-following approach.

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 [1–4], 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 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 [5], 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 [1, 3].

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 [6–8]. Three primary types of reactive approaches have shown

success: optic flow methods [7,9,10], artificial potential fields [6,11], and imitation learning [8]. 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 [12–15], 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 [16].

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 [17–19], 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. [20] 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 [21] and integrated with sampling-based motion-planners [22], 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 [23].

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.

3.1 Evaluating Collision Probabilities from Instantaneous Depth Information

We wish to evaluate the probability of collision for:

$$P\big(\text{Collision during } t \in [0, t_f] \mid \mathbf{D}, p_t(\mathbf{q})\big) \tag{1}$$

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.

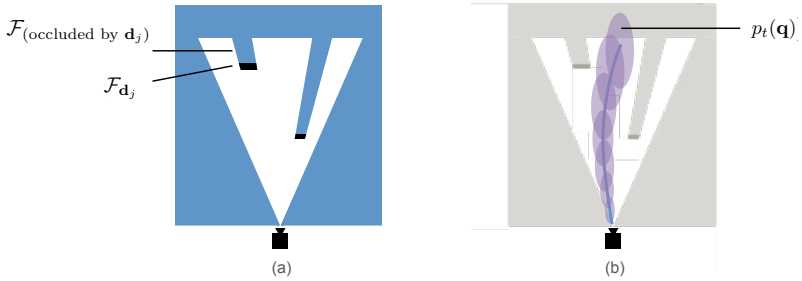


Fig. 1: Depictions of (a) depth measurements (black) and conservative assumption of unknown space as occupied (blue), and (b) time-varying distribution of configuration (purple).

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.

$$P\big(\text{Collision}, p_{t_i}(\mathbf{q}) |\, \mathcal{O}_{known}, \mathcal{U}\big) = P\big(\mathbf{q}(t_i) \in \{\mathcal{O}_{known} \text{ or } \mathcal{U}\}\big) \tag{2}$$

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. [16].

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.

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:

$$P\big(\text{Collision}, p_t(\mathbf{q})\big) \approx 1 - \prod_{i=1}^{n_t} \big[1 - P(\text{Collision}, p_{t_i}(\mathbf{q}))\big] \tag{3}$$

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. The mean in $\mathcal{U}$ is a large oversimplification, but avoids integrating over many small occluded frustums:

$$\big[1 - P(\text{Collision}, p_{t_i}(\mathbf{q}))\big] = \begin{cases} 0, & \text{if } \mu(p_{t_i}) \in \mathcal{U} \\ \prod_{j=1}^{n_d} \big[1 - P(\text{Collision}, p_{t_i}(\mathbf{q}), \mathbf{d}_j)\big], & \text{otherwise} \end{cases} \tag{4}$$

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 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 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 [20], 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 [16]. 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 [21]. 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 [23]. We note that with our unconstrained formulation, it is the relative differences between maneuver collision probabilities (see Figure 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 [16], but our requirement is a few orders of magnitude faster (milliseconds) to replan at the rate of depth image information (30-150 hz).

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:

$$\mathbb{E}\big[R(\mathcal{M}_i)\big] = P(\text{No Collision}, \mathcal{M}_i)R_{nav}(\mathcal{M}_i) + P(\text{Collision}, \mathcal{M}_i)R_{collision} \quad (5)$$

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 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:

$$\mathcal{M}^* = \underset{i}{\text{argmax}} \;\; \mathbb{E}\big[R(\mathcal{M}_i)\big] \quad (6)$$

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.

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 [17, 18, 24], 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.

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. Given a desired acceleration $\mathbf{a}_i$, geometry defines the mapping to {roll, pitch thrust} required to produce such an acceleration, given any yaw.

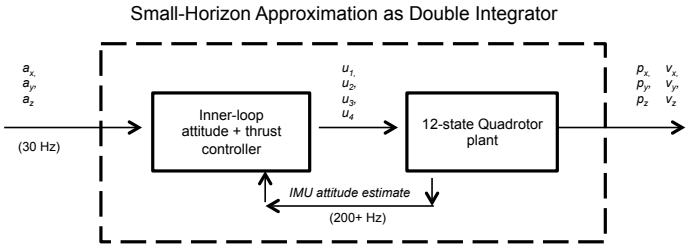


Fig. 2: Dynamics approximation considered: the quadrotor is modeled in feedback with the inner loop attitude and thrust controller.

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:

$$\mathbf{p}_i(t) \sim \mathcal{N}\left(\frac{1}{2}\mathbf{a}_i t^2 + \mathbf{v}_{0,\mu} t,\ t^2 \Sigma_{v_0}\right) \tag{7}$$

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

where $\mathbf{p}_i(t)$ is a random variable defining the distribution referred to as $p_t(\mathbf{q})$ in Section 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:

$$\mathbf{j}_i = \frac{\mathbf{a}_i - \mathbf{a}_0}{t_{jf}} \tag{8}$$

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

$$\mathbf{p}_i(t) \sim \mathcal{N}\left(\frac{1}{6}\mathbf{j}_i t^3 + \frac{1}{2}\mathbf{a}_0 t^2 + \mathbf{v}_{0,\mu} t, t^2 \Sigma_{v_0}\right) \quad \forall t \in [0, t_{jf}] \tag{9}$$

and for $t \in [t_{jf}, t_f]$ the double integrator model (Equation 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.

4.3 Maneuver Library and Attitude-Thrust Setpoint Control

We use a finite maneuver library (Figure 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.

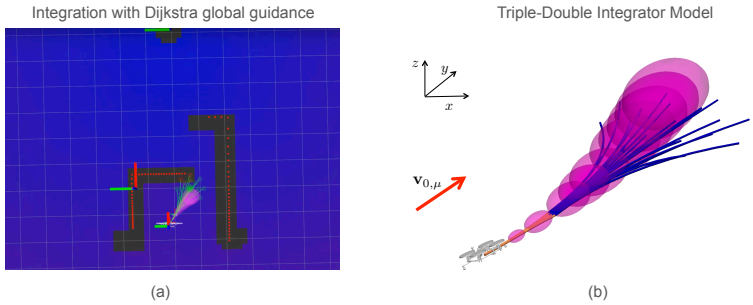


Fig. 3: (a) Visualization of integrating Dijkstra global guidance, where $\mathcal{R}_{nav}$ is the cost-to-go (blue is lower, purple is higher) of the final maneuver position. (b) Visualization of the piecewise triple-double integrator maneuver library. The library of maneuvers is shown with a positive 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.

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 tf $= 1.0$ 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:

$$P(\text{Collision, } \mathbf{p}_i(t)) \approx V_r \times \frac{1}{\sqrt{\det(2\pi\Sigma_p)}} \exp\Big[-\frac{1}{2}(\mathbf{p}_{i,\mu} - \mathbf{d}_j)^T \Sigma_p^{-1} (\mathbf{p}_{i,\mu} - \mathbf{d}_j)\Big] \tag{10}$$

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 [12]. 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 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}:

$$R_{nav}(\mathcal{M}_i) = ||\mathbf{p}_0 - \mathbf{p}_{goal}|| - ||\mathbf{p}_i(t_f) - \mathbf{p}_{goal}|| + R_v(v_f) \tag{11}$$

$$R_v(v_f) = \{0 \text{ if } v_f < v_{target},\ kv_f \text{ if } v_f \geq v_{target}\} \tag{12}$$

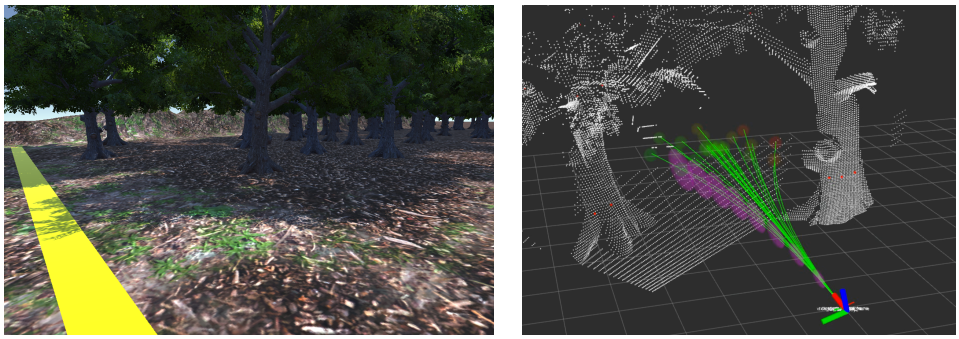
Where we used $k = 10$, and $R_{collision} = -10,000$.

5 Simulation Experimental Setup

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 [25] was used to simulate vehicle dynamics using a common 12-state nonlinear quadrotor model [26] 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 [27] 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)

Fig. 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).

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:

$$\mathbf{p}_{noisy}[i+1] \sim \mathcal{N}(\mathbf{p}_{true}[i+1] - \mathbf{p}_{true}[i], \frac{\sigma}{10}\mathbf{v_{true}}) \tag{13}$$

$$\mathbf{v}_{noisy}[i] \sim \mathcal{N}(\mathbf{v}_{true}[i], \frac{\sigma}{10}\mathbf{v_{true}}) \tag{14}$$

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.

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 [28]) 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.

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 5. Additional data is presented in Figure 6: average time to goal for successful trials, and example paths at various noise levels.

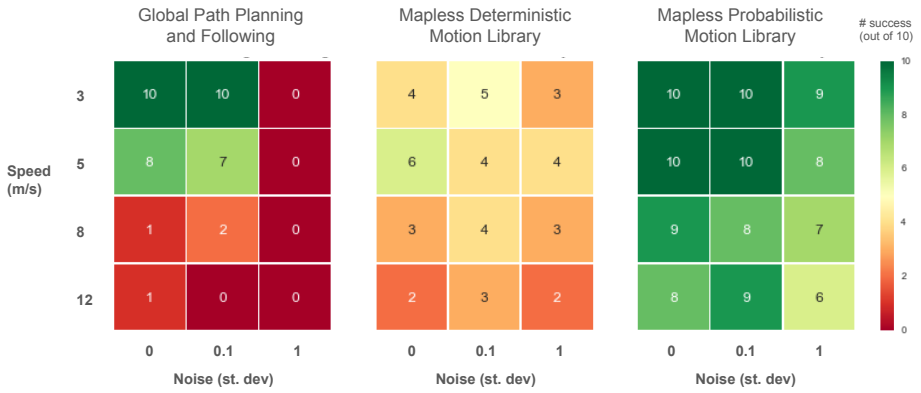


Fig. 5: Comparison summary of number of successful collision-free trials for the different approaches tested in in our simulated quadrotor race through the forest. Ten trials were run for each of the three approaches, for four different speeds {3, 5, 8, 12} meters per seconds, and for three different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

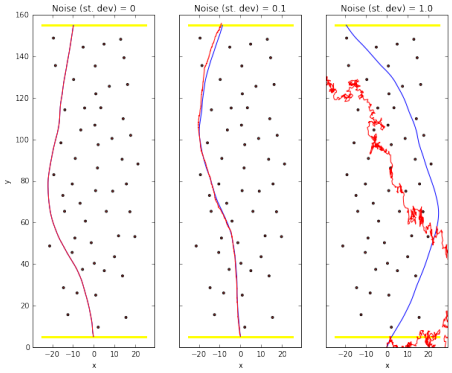
For comparison, we also compare with the approach of deterministically collision-checking our motion primitive library. For this deterministic method, the average time to goal on a successful run was faster than the probabilistic method by approximately 14%. The deterministic nature of the collision 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)

Fig. 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 (∼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 1: Measured averages and standard deviations of subprocess latencies, from one representative run each. Implementation on single-thread Intel i7.

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.

8 Acknowledgements

This work was supported by the DARPA Fast Lightweight Autonomy (FLA) program, HR0011-15-C-0110. We also thank Brett Lopez and William Nicholas Greene for fruitful discussions on modeling, control, and depth images.

References

1. Barry, A.J.: High-speed autonomous obstacle avoidance with pushbroom stereo, PhD Thesis, MIT (2016)
2. Barry, A.J., Tedrake, R.: Pushbroom stereo for high-speed navigation in cluttered environments. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2015) 3046–3052
3. Daftry, S., Zeng, S., Khan, A., Dey, D., Melik-Barkhudarov, N., Bagnell, J.A., Hebert, M.: Robust monocular flight in cluttered outdoor environments. arXiv preprint arXiv:1604.04779 (2016)
4. Liu, S., Watterson, M., Tang, S., Kumar, V.: High speed navigation for quadrotors with limited onboard sensing. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2016) 1484–1491
5. Bateux, Q., Marchand, E.: Histograms-based visual servoing. IEEE Robotics and Automation Letters **2**(1) (2017) 80–87
6. Scherer, S., Singh, S., Chamberlain, L., Elgersma, M.: Flying fast and low among obstacles: Methodology and experiments. The International Journal of Robotics Research **27**(5) (2008) 549–574
7. Beyeler, A., Zufferey, J.C., Floreano, D.: Vision-based control of near-obstacle flight. Autonomous robots **27**(3) (2009) 201–219
8. Ross, S., Melik-Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A., Hebert, M.: Learning monocular reactive uav control in cluttered natural environments. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE (2013) 1765–1772
9. Conroy, J., Gremillion, G., Ranganathan, B., Humbert, J.S.: Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. Autonomous robots **27**(3) (2009) 189–198
10. Hyslop, A.M., Humbert, J.S.: Autonomous navigation in three-dimensional urban environments using wide-field integration of optic flow. Journal of guidance, control, and dynamics **33**(1) (2010) 147–159

11. Nieuwenhuisen, M., Behnke, S.: Hierarchical planning with 3d local multiresolution obstacle avoidance for micro aerial vehicles. In: ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of, VDE (2014) 1–7
12. Du Toit, N.E., Burdick, J.W.: Probabilistic collision checking with chance constraints. IEEE Transactions on Robotics **27**(4) (2011) 809–815
13. Blackmore, L., Ono, M., Williams, B.C.: Chance-constrained optimal path planning with obstacles. IEEE Transactions on Robotics **27**(6) (2011) 1080–1094
14. Ono, M., Pavone, M., Kuwata, Y., Balaram, J.: Chance-constrained dynamic programming with application to risk-aware robotic space exploration. Autonomous Robots **39**(4) (2015) 555–571
15. Luders, B., Kothari, M., How, J.P.: Chance constrained rrt for probabilistic robustness to environmental uncertainty. (2010)
16. Janson, L., Schmerling, E., Pavone, M.: Monte carlo motion planning for robot trajectory optimization under uncertainty. arXiv preprint arXiv:1504.08053 (2015)
17. Majumdar, A., Tedrake, R.: Funnel libraries for real-time robust feedback motion planning. CoRR **abs/1601.04037** (2016)
18. Majumdar, A., Tedrake, R.: Robust online motion planning with regions of finite time invariance. In: Algorithmic Foundations of Robotics X. Springer (2013) 543–558
19. Tedrake, R., Manchester, I.R., Tobenkin, M., Roberts, J.W.: Lqr-trees: Feedback motion planning via sums-of-squares verification. The International Journal of Robotics Research (2010)
20. Matthies, L., Brockers, R., Kuwata, Y., Weiss, S.: Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2014) 3242–3249
21. Pan, J., Chitta, S., Manocha, D.: Probabilistic collision detection between noisy point clouds using robust classification. In: International Symposium on Robotics Research (ISRR). (2011) 1–16
22. Pan, J., Manocha, D.: Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing. The International Journal of Robotics Research (2016) 0278364916640908
23. Richter, C., Vega-Brown, W., Roy, N.: 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)
24. Van Den Berg, J., Abbeel, P., Goldberg, K.: Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. The International Journal of Robotics Research **30**(7) (2011) 895–913
25. Tedrake, R.: Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems (2014)
26. Mellinger, D., Michael, N., Kumar, V.: Trajectory generation and control for precise aggressive maneuvers with quadrotors. Int. J. Rob. Res. **31**(5) (April 2012) 664–674
27. Meier, L., Tanskanen, P., Heng, L., Lee, G.H., Fraundorfer, F., Pollefeys, M.: Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. Auton. Robots **33**(1-2) (August 2012) 21–39
28. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: Octomap: An efficient probabilistic 3d mapping framework based on octrees. Auton. Robots **34**(3) (April 2013) 189–206