Motion- and Uncertainty-aware Path Planning for Micro Aerial Vehicles*

Markus W. Achtelik and Simon Lynen
Autonomous Systems Lab, ETH Zurich, CH-8092 Zurich, Switzerland
e-mail: markus.achtelik@mavt.ethz.ch, simon.lynen@mavt.ethz.ch
Stephan Weiss
Computer Vision Group, NASA Jet Propulsion Laboratory/California Institute of Technology, Pasadena, California 91109
e-mail: stephan.weiss@ieee.org
Margarita Chli
Vision for Robotics Lab, School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, United Kingdom
e-mail: mchli@inf.ed.ac.uk
Roland Siegwart
ETH Zurich, CH-8092 Zurich, Switzerland
e-mail: r.siegwart@ieee.org

Received 5 July 2013; accepted 6 March 2014

Localization and state estimation are reaching a certain maturity in mobile robotics, often providing both a precise robot pose estimate at a point in time and the corresponding uncertainty. In the bid to increase the robots' autonomy, the community now turns to more advanced tasks, such as navigation and path planning. For a realistic path to be computed, neither the uncertainty of the robot's perception nor the vehicle's dynamics can be ignored. In this work, we propose to specifically exploit the information on uncertainty, while also accounting for the physical laws governing the motion of the vehicle. Making use of rapidly exploring random belief trees, here we evaluate offline multiple path hypotheses in a known map to select a path exhibiting the motion required to estimate the robot's state accurately and, inherently, to avoid motion in modes, where otherwise observable states are not excited. We demonstrate the proposed approach on a micro aerial vehicle performing visual-inertial navigation. Such a system is known to require sufficient excitation to reach full observability. As a result, the proposed methodology plans safe avoidance not only of obstacles, but also areas where localization might fail during real flights compensating for the limitations of the localization methodology available. We show that our planner actively improves the precision of the state estimation by selecting paths that minimize the uncertainty in the estimated states. Furthermore, our experiments illustrate by comparison that a naive planner would fail to reach the goal within bounded uncertainty in most cases.

1. INTRODUCTION

Driven by the need for mobile robots that are applicable in real tasks, research in spatial awareness and motion control has experienced great leaps over the past few years. It was not so long ago when simultaneous localization and mapping (SLAM) was shown to be possible with a single handheld camera by Davison, Molton, Reid, and Stasse (2007). Most recently, a micro aerial vehicle (MAV) was demonstrated to perform SLAM onboard, solely using visual and inertial cues (Weiss, Achtelik, Chli, & Siegwart, 2012). However, we are still missing essential functionality before a robot can carry out real missions autonomously from start to finish. With current research pushing for added autonomy, realistic path planning and obstacle avoidance are certainly at the top of the priorities list.

With the seminal works of Kavraki, Švestka, Latombe, and Overmars (1996) and Kuffner and LaValle (2000) paving the way for random-sampling-based path planners, the question arising is how to employ such methods within the navigation pipeline running onboard the mobile robot. While path planning has long been studied on ground vehicles, approaches dealing with MAV navigation in three dimensions are rather sparse. Perhaps this can be attributed to the fact that it is only very recently that sufficiently robust systems, able to perform simple maneuvers and estimate their motion, have started appearing (Fraundorfer et al.,

*The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7) under grant agreements n.266470 (myCopter) and n.600958 (SHERPA).

2012; Shen, Michael, & Kumar, 2011; Weiss et al., 2013). In this work, we choose to address path planning for MAVs, which is one of the most challenging navigation scenarios given both their high agility and the significance of prompt and sound state estimates to avoid crashes.

1.1. Path Planning for MAVs

In the context of MAV path planning, He, Prentice, and Roy (2008) proposed one of the first approaches applicable to a quad-rotor helicopter. Their method is based on sampling in information space, albeit ignoring the vehicle dynamics in the path generation. Similarly, in Prentice & Roy (2009), dynamics were also excluded from the sampling space, but the vehicle pose uncertainty has been taken into account. Using a laser-generated map, Wzorek, Kvarnström, and Doherty (2010) proposed using a list of predefined possible plan strategies to select from at run time, employing a machine-learning approach.

Sacrificing a global minimum, Richter, Bry, and Roy (2013) presented an effective and fast method to compute dynamic trajectories for a MAV. They furthermore reformulated the trajectory optimization method proposed by Mellinger and Kumar (2011) to an unconstrained problem. This allows for a higher number of path segments and improves numerical stability for the high-order polynomials commonly used to represent path segments. Planning toward optimal paths and the formation of multiple heterogeneous MAVs was demonstrated by Mellinger, Kushleyev, & Kumar (2012), while goal assignment for large teams of MAVs was demonstrated by Turpin, Mohta, Michael, & Kumar (2013).

Probably the most high-performing system to date is the very recent SPARTAN (Cover, Choudhury, Scherer, & Singh, 2013), demonstrating online obstacle perception and avoidance using a lidar and a stereo camera onboard a MAV. This method was shown to be able to handle both dense environments and free spaces. Finally, a very interesting concept was demonstrated by Choudhury, Scherer, & Singh (2013): using a rapidly-exploring random tree approach with guaranteed asymptotic optimality (RRT*) planner (Karaman & Frazzoli, 2010), feasible routes are generated and provided to a human operator in case of engine malfunctions to reach possible emergency landing sites. However, in contrast to all of the aforementioned works, the thesis of this work is that both vehicle dynamics and pose uncertainty need to be taken into account in path-planning estimation in order to have a truly generic application of the system in a variety of scenarios.

1.2. State Estimation Considerations in the Loop of Path Planning

In the past, the focus of accurate state estimation has been on the controlled states, such as the 6 DoF (degrees of freedom) pose and the 3 DoF velocity of the vehicle. However, with the recent emergence of self-calibrating power-on-and-go-systems (Kelly & Sukhatme, 2011; Weiss, 2012), it has become increasingly important to ensure fast state convergence at the beginning of the vehicle's motion (or right after reinitialization during the mission). Moreover, for self-calibrating systems, it is crucial to continuously excite the system in such a way that the calibrating states [e.g., biases of the inertial measurement unit (IMU), the visual scale, the transformation between sensors] are accurately estimated throughout the whole mission.

The studies (Achtelik, Achtelik, Weiss, & Siegwart, 2011; Weiss, 2012) on MAV navigation provide approaches for localization of the vehicle through continuous observation of visual landmarks. Thus, in such scenarios all regions are equally preferred during planning in terms of availability of measurements. The sparsity and availability of landmarks has given rise to special planning algorithms favoring areas with more reliable landmarks in the area between the start and goal positions (Bryson, Johnson-Roberson, & Sukkarieh, 2009; He, Prentice, & Roy, 2008; Prentice & Roy, 2009; Roy & Thrun, 1999). However, power-on-and-go systems usually have two additional requirements. First, as shown by Kelly & Sukhatme (2011) and Weiss & Siegwart (2011), these systems need excitation in linear acceleration and angular velocity before all states become observable. This is particularly true for systems estimating their inter-sensor calibration, in addition to the vehicle pose used for control. In hovering mode or while flying on a straight path, a MAV system remains in an unobservable mode preventing correct estimation of all the system states. Secondly, for single-camera systems, if a loss of the visual map occurs (which cannot be ruled out during a real mission), reinitialization of the system has to be performed. After such a reinitialization, the metric scale has to be reestimated quickly to allow continuous and robust vehicle control.[1] Our work aims to provide robustness against such cases in which typical planners would fail to produce a feasible path: We seek to find not only a short path to the destination, but a path along which the states estimated in our system are best observable at all times. Figure 1 illustrates an example path generated by our method, which aims to reduce the position uncertainty of the vehicle at the goal location, with the additional constraint of reaching the target within the given confidence area.

In this work, we study the problem of not just acquiring *any* measurement, but in fact acquiring an *informative* measurement, such that the vehicle always remains in a fully observable mode. Using our previous work (Achtelik, Weiss, Chli, & Siegwart, 2013b) as a basis, we employ the rapidly exploring random belief tree (RRBT) approach (Bry

[1]Monocular systems measure the three-dimensional position only up to an arbitrary scale.

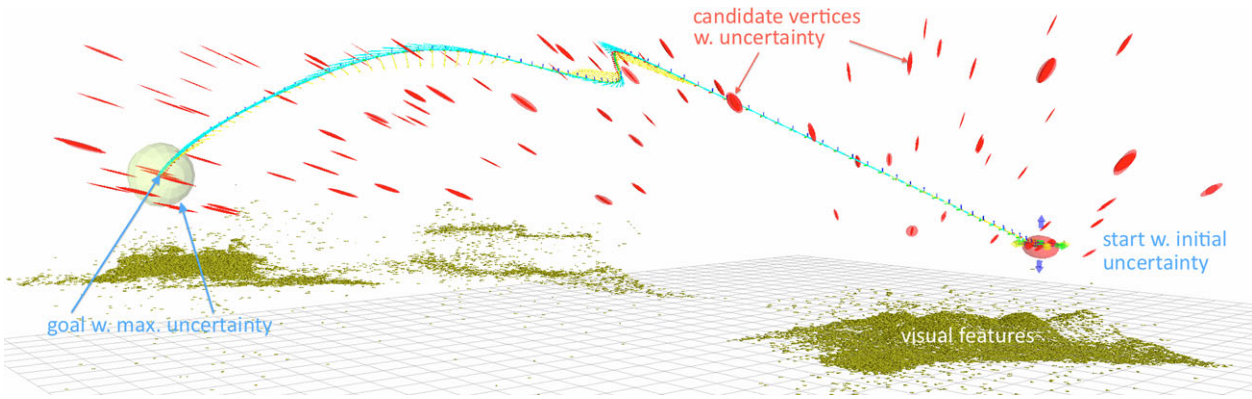


Figure 1. The path planned by the proposed method to navigate in a medium-sized outdoor area from the start (on the right) to the goal (left). The dark green dots denote visual landmarks, and the red ellipsoids denote the uncertainty of intermediate points that the planner has considered during the optimization phase. Control input velocities (cyan) and accelerations (yellow) are computed that minimize the localization covariance. With this methodology, the planner is hence able to avoid flying over areas that do not allow accurate localization while assuring sufficient motion to render all states of our system observable

& Roy, 2011) to plan and follow a path for a MAV on the fly, from the current position to a user- or application-defined destination. We choose to employ RRBT since it can handle *both* dynamic constraints and uncertainty. It is important to note that in order to produce realistic paths for the MAV, the *same* navigation pipeline is used within the planning phase as the one used by the MAV. This navigation framework was introduced by Weiss & Siegwart (2011) and Weiss et al. (2012) and is used to acquire a map of the MAV's workspace to serve as a basis for the planning and navigation phase. Monitoring the uncertainty in 24 states, the proposed system can plan a collision-free path (avoiding static obstacles) incorporating the vehicle's controller dynamics, while being able to cope with nonholonomic constraints of the vehicle.

Exploring the power of the RRBT framework within MAV navigation, we demonstrate how effective path planning can not only reduce the error and uncertainty of the state estimates, but also allow for faster convergence of states initialized far off their true values. Studying this particularly challenging navigation scenario, we aim to highlight the influence of path planning in the overall robustness of navigation when used *in the loop* of the estimation process.

The remainder of the paper is organized as follows: We begin with a description of the MAV navigation framework and the helicopter model used in this work in Section 2, followed by the definition of the problem statement and the assumptions we make in Section 3. Our problem formulation for the RRBT path planner and the realization for the MAV navigation scenario are discussed in Section 4. With the description of the experimental setup in Section 5, we present and analyze the behavior of the method via simulated and real-world experimentation and field tests in Section 6, before drawing conclusions in Section 7.

2. SYSTEM DESCRIPTION

In this section, we provide a brief overview of our framework for MAV navigation as well as the helicopter model, both of which are used later within the path-planning approach. It should be noted that all the systems described here have been tested and verified in various field experiments. Throughout this article, the term *state* has the following three different aspects:

- *filter state* $\boldsymbol{x}_f$: refers to the state used in the estimation processes using the extended Kalman filter (EKF) as described in Section 2.1. This comprises the vehicle pose and velocity used for control, as well as the (inter)sensor calibration parameters.
- *system state* $\boldsymbol{x}_d$: refers to the state used to describe the vehicle dynamics. The pose of the system and its derivatives are of interest in order to grant smooth motion (see Section 2.2).
- *sampling state* $\boldsymbol{x}_s$: refers to the state in the space from which we take samples in order to obtain new state vertices in the RRBT graph structure (see Section 4.3).

2.1. Visual-inertial State Estimation for a MAV

In the following, we consider the case of vision-based navigation for MAVs using a single camera and an IMU as the only sensors. We use the keyframe-based monocular visual SLAM framework PTAM (Klein & Murray, 2007) for localization, which we adapted to the requirements of MAV navigation (Weiss et al., 2013). For the system state estimation, we use an EKF approach according to our previous work in Lynen et al. (2013), Weiss & Siegwart (2011), and Weiss et al. (2012). For completeness, we briefly summarize the essentials.

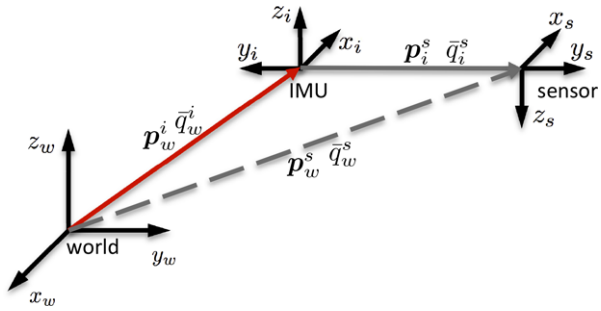


Figure 2. Setup depicting the IMU centered robot body with its sensors with respect to a world reference frame. The state is $\boldsymbol{x}_f = \left[\boldsymbol{p}_w^i \quad \boldsymbol{v}_w^i \quad \bar{q}_w^i \quad \boldsymbol{b}_\omega \quad \boldsymbol{b}_a \quad \lambda \quad \boldsymbol{p}_i^s \quad \bar{q}_i^s\right]$, whereas $\boldsymbol{p}_w^s$ and $\bar{q}_w^s$ denote the robot's sensor measurements of (a possibly scaled) position and attitude, respectively, expressed in the world frame of reference

The state of the filter is composed of the position of the IMU $\boldsymbol{p}_w^i$ in the world frame, its velocity $\boldsymbol{v}_w^i$, and its attitude quaternion $\bar{q}_w^i$ describing a rotation from the world to the IMU frame. We also add the gyro and acceleration biases $\boldsymbol{b}_\omega$ and $\boldsymbol{b}_a$. Since we use a monocular vision approach for localization, the position part of the pose measurement is arbitrarily scaled. Given sufficient motion, this visual scaling factor λ is observable (Kelly & Sukhatme, 2011; Weiss, 2012). This scaling factor is set to an arbitrary value at initialization of the SLAM system. However, the scaling factor drifts slightly during motion of the camera. Therefore, we add it to the state in order to be estimated continuously online. When fusing camera data with IMU data, knowledge of their extrinsic calibration is essential in order to map visual measurements into the IMU frame. Therefore, we add calibration states, namely the rotation from the IMU frame to the camera sensor frame $\bar{q}_i^s$ and the distance between these two sensors, $\boldsymbol{p}_i^s$. How these states relate to each other can be seen in Figure 2. This setup yields a 24-element filter state vector $\boldsymbol{x}_f$:

$$\boldsymbol{x}_f = \left[\boldsymbol{p}_w^{i\,T} \quad \boldsymbol{v}_w^{i\,T} \quad \bar{q}_w^{i\,T} \quad \boldsymbol{b}_\omega^T \quad \boldsymbol{b}_a^T \quad \lambda \quad \boldsymbol{p}_i^{s\,T} \quad \bar{q}_i^{s\,T}\right]^T \tag{1}$$

The following differential equations govern the state:

$$\dot{\boldsymbol{p}}_w^i = \boldsymbol{v}_w^i, \tag{2}$$

$$\dot{\boldsymbol{v}}_w^i = \boldsymbol{C}(\bar{q}_w^i) \cdot (\boldsymbol{a}_m - \boldsymbol{b}_a - \boldsymbol{n}_a) - \boldsymbol{g}, \tag{3}$$

$$\dot{\bar{q}}_w^i = \frac{1}{2}\Omega \cdot (\boldsymbol{\omega}_m - \boldsymbol{b}_\omega - \boldsymbol{n}_\omega) \cdot \bar{q}_w^i, \tag{4}$$

$$\dot{\boldsymbol{b}}_\omega = \boldsymbol{n}_{b_\omega}, \tag{5}$$

$$\dot{\boldsymbol{b}}_a = \boldsymbol{n}_{b_a}, \tag{6}$$

$$\dot{\boldsymbol{b}}_\omega = \boldsymbol{n}_{b_\omega} \quad \dot{\boldsymbol{b}}_a = \boldsymbol{n}_{b_a} \quad \dot{\lambda} = 0 \quad \dot{\boldsymbol{p}}_i^s = 0 \quad \dot{\bar{q}}_i^s = 0, \tag{7}$$

where $\boldsymbol{g}$ is the gravity vector in the world frame and $\Omega(\boldsymbol{\omega})$ is the quaternion multiplication matrix of $\boldsymbol{\omega}$. $\boldsymbol{a}_m$ and $\boldsymbol{\omega}_m$ denote the linear accelerations and angular velocities measured by the IMU, while $\boldsymbol{n}_a$, $\boldsymbol{n}_{b_a}$, $\boldsymbol{n}_\omega$, $\boldsymbol{n}_{b_\omega}$ denote zero mean white Gaussian noise. We assume the scale drifts spatially and not temporally, thus $\dot{\lambda} = 0$.

For the possibly scaled camera position measurement $\boldsymbol{p}_w^s$ obtained from the visual SLAM algorithm, we have the following measurement model, with $\boldsymbol{C}(\bar{q}_w^i)$ denoting the rotation matrix of the IMU's attitude in the world frame:

$$\boldsymbol{z}_p = \boldsymbol{p}_w^s = [\boldsymbol{p}_w^i + \boldsymbol{C}(\bar{q}_w^i) \cdot \boldsymbol{p}_i^s] \cdot \lambda + \boldsymbol{n}_p. \tag{8}$$

For the rotation measurement, we apply the notion of an error quaternion. The vision algorithm yields the rotation from the world frame to the camera frame q_w^s. We can model this as

$$\boldsymbol{z}_q = \bar{q}_w^s = \bar{q}_w^i \otimes \bar{q}_i^s \otimes \delta\bar{q}_n. \tag{9}$$

A nonlinear observability analysis, as suggested by Hermann & Krener (1977) and done in Kelly & Sukhatme (2011); Weiss (2012), reveals that all states are observable, including the intersensor calibration states $\boldsymbol{p}_i^s$ (distance from the IMU to the sensor) and $\bar{q}_i^s$ (rotation from the IMU to the sensor). This is true as long as the vehicle excites the IMU's accelerometer and gyroscopes in at least two axes, as proven in Kelly & Sukhatme (2011) and Mirzaei & Roumeliotis (2008).

2.2. Helicopter Model

To plan the motion of a MAV while ensuring that all states of the aforementioned state estimation filter stay observable, we will have to execute (i.e., propagate forward) this filter along candidate optimized paths. Besides visual measurements ($\boldsymbol{p}_w^s$, $\bar{q}_w^s$), we need to generate the system inputs for the filter in Eqs. (3) and (6), namely acceleration and the angular velocity, both in body-fixed coordinates. Furthermore, we aim to keep the helicopter model generic for quad- or in general multicopter MAVs, such as the hexacopter we used for the experiments in Weiss et al. (2012). Therefore, we define the thrust in terms of acceleration in the z axis of the helicopter, and the body fixed angular velocity $\boldsymbol{\omega}$ as control inputs for the multicopter system. We assume that there exists a low-level controller that maps the individual rotor speeds to angular velocities. Thus, we do not have to care about vehicle-specific details, such as rotor count/alignment or moments of inertia. Since there is an underlying controller for $\boldsymbol{\omega}$, we require a demanded control input to be continuous and differentiable.

To plan appropriate paths for the MAV, we need to consider the capabilities of the helicopter given the aforementioned constraints. The findings of Mellinger and Kumar (2011) regarding the differentially flat outputs $[x\ y\ z\ \psi]$ (i.e., position and yaw) for a quadrotor can be applied to this problem: given a function for the position of the center of

mass of the helicopter and its orientation (yaw), which are sufficiently differentiable, we can always compute the remaining states we need for the helicopter (roll/pitch part of the attitude) and the required control inputs angular velocity $\boldsymbol{\omega}$ and thrust $|\boldsymbol{t}|$. The latter is simply a function of the attitude and rotor speeds. We assume that the IMU is aligned with the center of gravity of the helicopter. According to Mellinger & Kumar (2011), the full attitude $\bar{q}_w^i$ can be computed as follows:

$$\boldsymbol{z}_i = \frac{\boldsymbol{t}}{|\boldsymbol{t}|}, \quad \boldsymbol{t} = \boldsymbol{a} + \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T, \tag{10}$$

$$\boldsymbol{y}_i = \frac{\boldsymbol{z}_i \times \boldsymbol{x}_{\hat{0}}}{|\boldsymbol{z}_i \times \boldsymbol{x}_{\hat{0}}|}, \quad \boldsymbol{x}_{\hat{0}} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \end{bmatrix}^T, \tag{11}$$

$$\boldsymbol{x}_i = \boldsymbol{y}_i \times \boldsymbol{z}_i, \tag{12}$$

$$\Rightarrow \bar{q}_w^i = \bar{q}(\boldsymbol{C}_w^i), \quad \boldsymbol{C}_w^i = \begin{bmatrix} \boldsymbol{x}_i & \boldsymbol{y}_i & \boldsymbol{z}_i \end{bmatrix}, \tag{13}$$

where $\boldsymbol{a}$ denotes the acceleration and ψ is the yaw angle of the helicopter. The thrust vector $\boldsymbol{t}$ defines the direction of the unit vector $\boldsymbol{z}_B$ of a body-fixed coordinate system $[\,\boldsymbol{x}_i\ \boldsymbol{y}_i\ \boldsymbol{z}_i\,]$. This can also be seen as the rotation matrix $\boldsymbol{C}_w^i$ that describes the orientation of the helicopter (IMU) with respect to the world coordinate system. $\boldsymbol{\omega}$ computes as follows (Achtelik, Lynen, Chli, & Siegwart, 2013a; Mellinger & Kumar, 2011):

$$\omega_x = -\boldsymbol{j}^T \cdot \boldsymbol{y}_i \cdot |\boldsymbol{t}|^{-1}, \tag{14}$$

$$\omega_y = \boldsymbol{j}^T \cdot \boldsymbol{x}_i \cdot |\boldsymbol{t}|^{-1}, \tag{15}$$

$$\omega_z = \dot{\psi} \cdot \boldsymbol{z}_w^T \cdot \boldsymbol{z}_i, \tag{16}$$

where $\boldsymbol{j}$ denotes the jerk, i.e., the first derivative of the acceleration. Since we require $\boldsymbol{\omega}_B$ to be continuous and differentiable, we need to ensure that the derivatives of the jerk $\boldsymbol{j}$ and angular rate $\dot{\psi}$ (snap $\boldsymbol{s}$ and $\ddot{\psi}$) are continuous. Therefore, we define the vehicle state x_v as follows:

$$\boldsymbol{x}_v = \begin{bmatrix} \boldsymbol{p}^T & \boldsymbol{v}^T & \boldsymbol{a}^T & \boldsymbol{j}^T & \boldsymbol{s}^T & \psi & \omega_\psi & \dot{\omega}_\psi \end{bmatrix}^T. \tag{17}$$

This state is needed for the local path planning in Section 4.2. To summarize, $\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{a}, \boldsymbol{j}, \boldsymbol{s} \in \mathbb{R}^{3\times 1}$ denote the position and its four derivatives, namely velocity, acceleration, jerk, and snap, respectively, while $\psi, \omega_\psi, \dot{\omega}_\psi \in \mathbb{R}$ denote the yaw angle, the angular rate, and angular acceleration.

This aligns well with our work on position and trajectory control for MAVs (Achtelik et al., 2013a), where we skip the common attitude control loop and directly control the angular rate ω_B and thrust $|\boldsymbol{t}|$ of the helicopter. From the planner described in the next section, we obtain the trajectory for the optimized path as a set of high-order polynomials. We sample these polynomials and their derivatives (from $\boldsymbol{p}$ to $\boldsymbol{j}$) at time intervals of 10 ms, and we use Eqs. (10)–(16) to generate feed-forward commands for $\boldsymbol{\omega}$ and $|\boldsymbol{t}|$. To account for disturbances and modeling errors, an error controller compares the states $\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{a}$ with the reference set by the time-discrete samples of the polynomials. Errors are turned into angular rate and thrust commands by feedback linearization and are fed back to the angular rate controller, keeping the vehicle on the desired trajectory. For the details of this trajectory tracking controller, we refer the reader to Achtelik et al. (2013a).

Figure 3 depicts the whole system setup. All flight-relevant data are processed onboard on the appropriate hardware, depending on computational power and real-time constraints. Paths generated by our planner are sent to the onboard computer, where the "trajectory sampling" part generates feed-forward control signals and reference trajectories as described previously in this section.

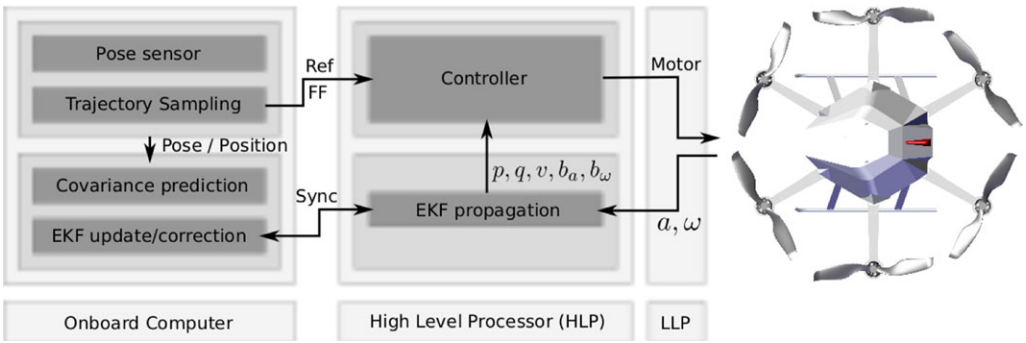


Figure 3. System setup showing the controller and state estimation components of our system. The arrows denote the information flow between high-level and low-level controllers and the state estimation framework. All flight-relevant data are processed onboard, while paths generated by our planner are sent to the onboard computer, where the "trajectory sampling" part generates feed-forward controls and reference trajectories

3. PROBLEM STATEMENT

As in most path-planning approaches, the aim here is to (a) plan the shortest or most energy-efficient path possible, (b) ensure that this is a collision-free path, and lastly (c) incorporate the dynamic constraints of the vehicle in the planning. In this work, we go a step further, and in addition to the aforementioned constraints, our planner is required to be motion- and state-uncertainty-aware, such that a realistic path is planned for the vehicle in question. As a result, in order to take the quality of the estimated state into account, we constrain the planner to (d) excite the system such that certain states of $\boldsymbol{x}_f$ (biases, camera-IMU calibration, scale) become observable and converge fast, and finally, (e) plan a safe path incorporating the availability and the configuration of visual landmarks used for our monocular visual SLAM system.

To handle all of these requirements, we chose to employ rapidly exploring random belief trees by Bry & Roy (2011)—dubbed "RRBT" by the authors—as a planning back-end. This method is able to handle both the dynamic constraints of the vehicle and uncertainty in the vehicle state estimate. Furthermore, due to the sampling-based nature of the approach, we do not need to make any assumptions on discontinuities in measurement uncertainty with respect to the position of our vehicle in the workspace. This aspect becomes particularly important when incorporating (e), the distribution of visual landmarks as a constraint within the path-planning framework.

We note that, as the method presented here is a global planner, we require a *previously known* map, providing information about visual landmarks and obstacles. The maps used in this work were prebuilt using our vision-based SLAM framework (Weiss et al., 2013) and man-in-the-loop waypoint setting. Given the difficulty of the task at hand, here we consider static obstacles (i.e., as outlined in the known map) and offline path planning, while extending to dynamic obstacles and online map creation and online path (re)planning are natural future directions for a generic system.

Looking at the application of the aforementioned requirements within our framework, (a) and (b) essentially drive the planning calculations, stating the problem we aim to solve. During the local path planning within each RRBT iteration, (c) is incorporated, while applying the findings of vehicle dynamics and the helicopter's differential flatness (see Section 2.2). The property of differential flatness allows us to reduce the sampling space to position and yaw only, which reduces the complexity of the approach. Details of the sampling strategy are explained in Section 4.3.

The last two constraints, (d) and (e), corresponding to novel contributions of this work, are formulated in terms of uncertainty, obeying a set of rules. First, for a path planned up to a given stage, if insufficient motion has been planned for, the uncertainty of the states requiring motion is not reduced. The planner, therefore, favors paths exhibiting motion and thus reducing uncertainty. As power consumption is directly dependent on additional motion, the constraints (a) and (d) compete for optimality. This forces the planning framework to trade off between increasing excitation, to reduce the uncertainty of the vehicle's states, and reducing motion, for energy efficiency. As a result, the vehicle is just as excited as necessary to reach the goal within the specified uncertainty region. Secondly, the visual landmark configuration and availability are incorporated during EKF updates along local connections in the RRBT graph (cf. "propagate" in Section 4.1): areas with no visual features available or bad feature configurations will not update the state covariance, or will update it in a certain direction only.

An example of a realistic system that could employ the approach would be a "fly-over and land" maneuver of an unmanned aerial vehicle (UAV) in an unknown environment. Before landing at a site, the vehicle first does a fly-over maneuver at an obstacle-free altitude using GPS as the primary source of information for navigation. Upon building the map of the area using cameras, the planner then plans the approach and landing trajectory, where all system states remain observable, obstacles are avoided, and the terrain provides sufficient information for the visual localization system.

4. PATH PLANNING TOWARD OPTIMIZED STATE ESTIMATION

In the following, we discuss the main properties of the RRBT algorithm as proposed by Bry & Roy (2011), and we describe how it can be employed in our visual-inertial MAV state estimation framework. The RRBT algorithm was chosen as a planning back-end, since it can handle both the dynamic constraints of the vehicle as well as uncertainty of the state estimation.

4.1. The RRBT Approach

The basic idea of RRBT is to interleave graph construction and search over the graph. Similarly to known planners, such as rapidly exploring random graphs (RRG), the algorithm operates on a set of *state vertices* V connected by edges E, defining a graph in *state space*. In addition to the state $v.x$, each state vertex $v \in V$ owns a set N of so-called *belief nodes* $n \in N$. Each belief node contains properties such as state estimate covariance Σ, a distribution over state estimates Λ, and a cost c. Furthermore, it has pointers to its owning vertex v and to a parent belief node: Being at a current vertex v_c, a unique path with the properties Σ, Λ, and c through the graph is described by successively following the belief nodes' parents, starting from v_c until v_{start}. With each belief node having a pointer to its owning vertex, we can also reconstruct the "physical" path from v_{start} to v_c. Multiple belief nodes at one state vertex are possible since there could be

multiple candidates for optimized paths to that vertex, e.g., one with smaller cost c and another with better Σ.

A RRBT iteration starts similar to a RRG iteration. After sampling a new state v_{new} vertex, an approximate connection is made to the nearest state vertex v_{nearest}. In addition, given a successful (collision-free) connection, if there exists one belief at v_{nearest} that can be *propagated* without collision along the newly created edge e_{new}, v_{new} gets added to V. *Propagate* means that a state estimation filter, such as the EKF in Section 2.1, is initialized with the state at a starting vertex and the properties of a belief node (Σ)[2]. This filter is executed along e_{new}, and a collision check is performed that takes the uncertainty along that edge into account. Upon success, exact connections are made forth and back to a set of near vertices within a certain radius (Bry & Roy, 2011; Karaman & Frazzoli, 2010), which also get propagated. After successful propagation, a new belief node is added to the corresponding vertex, if it is not dominated by an existing belief node at that vertex. Finally, a search queue keeps track of all belief nodes involved in the previous steps, and updates, expands, or removes them from the graph.

This approach allows us to set not only a start and goal state, but also the uncertainty of the system at the start state and a maximum uncertainty region at the goal. The goal is not reached until both state *and* uncertainty constraints are met. By keeping track of the system uncertainty in the way described above, the algorithm plans a *safe* path to the goal region, where it can *localize* itself while providing *sufficient excitation* to keep the system's states observable.

4.2. Local Path Planning

To connect a newly sampled state vertex to the nearest state vertex, and to the set of near vertices, we need a local path planner. While for algorithms such as rapidly exploring random trees (RRT) an *approximate* connection from v_{nearest} to v_{new} is sufficient,[3] we need to make additional *exact* connections between v_{new} and V_{near}, and back from v_{new} to v_{nearest}. The main difficulty is that multicopters are underactuated systems, while we require the fourth derivative of the position (snap) to be continuous (cf. Section 2.2). We want to be able to include actuator constraints (snap), but also constraints such as maximum velocity and acceleration. Furthermore, the vehicle should fly smooth paths without having to stop at state vertices. As a further constraint, many of those connections need to be created during graph construction. Thus, we cannot afford sophisticated optimization techniques.

A simple and fast solution was proposed by Webb & van den Berg (2012), but it requires linearization of the dynamics around the hovering point. Motion constraints other than actuator limitations are not easy to set. To enforce motion constraints, it sounds tempting to ignore those during local planning and to treat a violation simply as a collision. The result would be "no connection" and the next sample would be taken. This may work for simpler approaches such as RRT and just results in more samples. However, for RRG or RRBT, this would lead to missed exact connections between existing state vertices that are actually collision-free.

We decided to adapt the *minimum snap trajectory* approach (Mellinger & Kumar, 2011) to our needs. This approach uses Nth-order polynomials with more degrees of freedom than initial and final (equality) constraints. It leaves the remaining constraints to a solver, optimizing a quadratic cost function and inequality constraints. Since we need to enforce continuity to the fourth derivative of position (cf. Section 2.2), we need at least 10 parameters (ninth-order polynomials) plus some degrees of freedom for the optimizer. Unlike in Mellinger & Kumar (2011), we chose to optimize over the integral of the squared norm of the acceleration instead of the snap. The motivation for this choice was to plan energy-efficient paths. Compared to snap, acceleration directly translates into permanent additional thrust that all the motors have to provide, while snap just causes particular motors to spin up/down quickly. This results in the following optimization problem:

$$f(t) = \boldsymbol{t} \cdot \boldsymbol{c},\ \boldsymbol{t} = \begin{bmatrix} 1 & t & t^2 \cdots t^{N-1} \end{bmatrix}, \tag{18}$$

$$\min \int_{t_0}^{T} \left\| \frac{d^2 f(t)}{dt^2} \right\| \text{ s.t.}, \tag{19}$$

$$\frac{d^n f(t_0)}{dt^n} = \frac{d^n (v_{\text{start}} \,.\, x)}{dt^n};\ n = 0, \ldots, 4, \tag{20}$$

$$f(T) = (v_{\text{end}} \,.\, x \,.\, p), \tag{21}$$

$$\frac{d^n f(T)}{dt^n} = \frac{d^n (v_{\text{end}} \,.\, x)}{dt^n} \text{ or free};\ n = 1, \ldots, 4, \tag{22}$$

where $f(t)$ is an Nth-order polynomial with coefficients $c_0, \ldots, c_N$ of the position, and p_{start} and p_{end} are the position at the starting and ending state vertex, respectively. A simplification is that we only need to connect two states, i.e., we do not have any intermediate points and thus we can keep $t_0 = 0$, while T is the time needed to fly through the local path. We apply the same methodology for yaw. We optimize the integral over the angular rate while enforcing continuity up to the second derivative of yaw. For a discussion on when to keep the constraints in Eq. (22) fixed or leave them free, we refer to Section 4.3.

A beneficial property of the above optimization problem is the absence of inequality constraints, which makes it solvable with a closed-form solution. However, this still depends on a fixed time T to travel along the trajectory.

[2]We omit Λ here since we assume sufficient measurements and the vehicle stays close to the nominal trajectory.

[3]In this case, v_{new} only serves as a direction to grow the tree toward, so it can simply be set to the state to where we were actually able to steer.

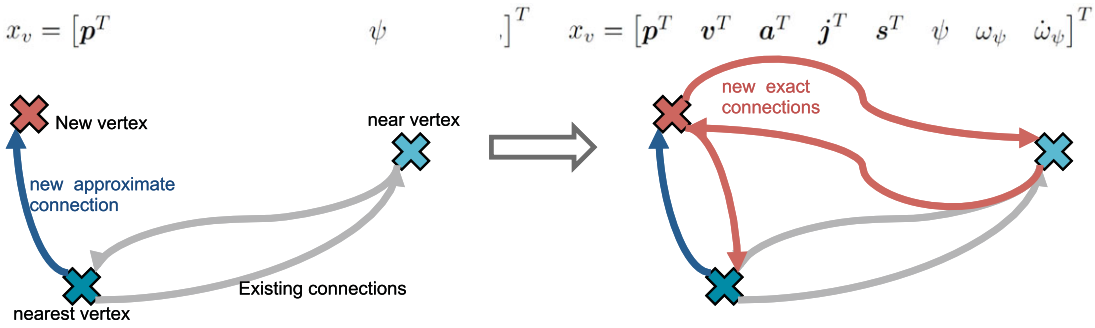


Figure 4. Principle of the local-planning and sampling strategy: the proposed method picks samples from a four-dimensional (position and yaw) state-space only, and creates an *approximate* connection from the nearest vertex using our local path-planning method. The remaining state variables are left as free variables to the local planning method. Once the approximate connection is made, the local planner fixes the free variables at the new vertex. This way, we "sample" the remaining state variables (derivatives of position and yaw) in the direction of motion, avoiding useless state vertices. Then, *exact* connections are created back to the nearest vertex and to the set of near vertices

Nonlinear optimization techniques with time as an additional parameter and inequality constraints are costly (and numerically problematic), therefore we were seeking a simple and fast solution: We solve the aforementioned problem with a conservative estimate for T and compare this solution to predefined maximum values for each derivative of the position. Then, we scale the path time according to the derivative that is closest to its motion constraint:

$$c(n) = \text{abs}\left(\left(\frac{d^n f(t)}{dt^n}\right) \Big/ c_{n,\max}\right), n = 1, \ldots, 4, \quad (23)$$

$$T_{\text{new}} = T \cdot \max(c(n))^{1/\text{argmax}(c(n))}. \quad (24)$$

We then recompute the optimization problem with the new path time T_{new}, which is fast since we do not have inequality constraints in the optimization problem. Note that the time-scaling as proposed in Mellinger and Kumar (2011) does not work in our case, since it also scales the boundary conditions that are nonzero in our case.

4.3. Sampling of State Vertices

We decided to sample in the space of the differentially flat outputs, position, and yaw angle, as previously defined. The sampling state becomes

$$x_s = [x \quad y \quad z \quad \psi]^T. \quad (25)$$

First, this choice is motivated by reducing the complexity of the sampling space. Second, the states of the vehicle are tightly coupled and we want to sample in a way that is physically reasonable. As an example, it is questionable to sample a state in the positive direction with respect to a current state while sampling a negative velocity. Therefore, to create a connection from the nearest (in the sense of position and yaw) state vertex v_{nearest} to the newly sampled state vertex, v_{new} is created by applying the method from Section 4.2 while leaving the derivatives of position in the final condition from Eq. (22) free. After optimization of the local path, these free variables result from the optimization and define the full state at v_{new}. Since we optimize over the acceleration, the optimizer will not decelerate the vehicle toward the end of a local path, thus we implicitly sample the velocity and its derivatives in the direction of motion (Figure 4 illustrates the process).

4.4. Covariance Comparison

From the ordering of partial paths represented by belief nodes, a belief node n_a dominates (is "better" than) a node n_b, as described by Bry & Roy (2011), if

$$n_a < n_b \Leftrightarrow n_a.c < n_b.c \wedge n_a.\Sigma < n_b.\Sigma \wedge n_a.\Lambda < n_b.\Lambda, \quad (26)$$

where $n.c$, $n.\Sigma$, and $n.\Lambda$ are the properties of a belief node (Section 4.1). Within an RRBT iteration, a new belief node n_b is only added to the set of belief nodes $v.N$ if it is not dominated (is not "worse") by any existing belief node $n_a \in v.N$ of its state vertex v. In other words, a new belief node is added to $v.N$ if at least one of its properties $(n.c, n.\Sigma, n.\Lambda)$ is better than in any of the existing belief nodes. After n_b is added to $v.N$, another check is performed if n_b dominates any existing belief node at v, and it can therefore prune it. In that case, n_b has to be better in all properties.

An important issue is how to compare two covariance matrices, i.e., how to judge if one is "better" than the other. While this may be intuitive for systems with two or three dimensions in position, it becomes difficult for our system with 24 states, not only because of the dimensionality, but also because the states are not referring to the same physical quantity or unit. As an example, how could an improvement

of uncertainty in position be compared with an improvement for the gyro biases? A very conservative measure for $n_a \,.\, \Sigma$ dominating $n_b \,.\, \Sigma$ would be

$$n_a \,.\, \Sigma < n_b \,.\, \Sigma \Leftrightarrow \min(\text{eig}(n_b \,.\, \Sigma - n_a \,.\, \Sigma)) > 0. \quad (27)$$

This in return means that any new belief node n_b, whose covariance is better in only one dimension, would be added to $v \cdot N$. This results in many belief nodes being added, having to be propagated and thus requiring additional computational resources. Especially in our high-dimensional case, it is questionable if, for instance, a slight improvement in gyro biases justifies adding a new belief node, while position uncertainty has grown. Other options are comparing the determinants, which could be thought of as the volume of the covariance ellipsoid, or the trace. However, both cannot make a distinction between ellipsoids with a high or low ratio of their axes. Furthermore, since our states are referring to different physical quantities, states with a smaller order of magnitude would dominate using the determinant method, and vice versa using the trace method.

We decided to use the Kullback-Leibler (KL) divergence (Kullback & Leibler, 1951) with respect to a reference covariance matrix Σ_{ref} as a performance measure for comparing belief nodes. That is, we want to find out how similar to Σ_{ref} the other covariance matrix is. In our case, the KL divergence for a normal distribution with zero mean is computed as

$$D_{\text{KL}} = \frac{1}{2}\left[\text{trace}\left(\Sigma_{\text{ref}}^{-1} \cdot n \,.\, \Sigma\right) - \ln\left(\frac{\det(n \,.\, \Sigma)}{\det(\Sigma_{\text{ref}})}\right) - N\right], \quad (28)$$

where N is the dimension of our filter (error) state, which is 22 in our case (see Section 2.1). For Σ_{ref}, we chose a diagonal matrix with entries in the order of magnitude from a covariance matrix of a fully converged state estimation filter that we obtained during the experiments in Lynen et al. (2013) and Weiss et al. (2012). The motivation for this choice is to normalize the different orders of magnitudes of the filter state variables while preferring round covariance ellipsoids over those with a high axis ratio. Another advantage of this method is that the KL divergence can be computed at the creation time of its belief nodes rather than in Eq. (27) at every belief comparison, which occur a lot during each RRBT iteration.

As stated by Bry & Roy (2011), applying Eq. (26) whenever a new belief is added would result in a robot infinitely circling in information-rich environments, since that would always improve uncertainty slightly. Therefore, Bry and Roy propose using a small tolerance factor ϵ to block these useless paths:

$$n_a \lesssim n_b \Leftrightarrow (n_a \,.\, c < n_b \,.\, c) \wedge (n_a \,.\, \Sigma + \epsilon I < n_b \,.\, \Sigma) \wedge (n_a \,.\, \Lambda < n_b \,.\, \Lambda + \epsilon I). \quad (29)$$

4.5. RRBT for MAVs with Motion-dependent State Estimation

In this section, we summarize our findings and show how these are applied within the RRBT framework for a MAV. At the beginning of each iteration, we sample a new state vertex v_{new} for position and yaw (25) of the helicopter from a uniform distribution. An approximate connection from the nearest (position and yaw) state vertex v_{nearest} to v_{new} is created by applying the method from Section 4.3. The result of this approximate connection defines the full state [Eq. (17)] at v_{new}. This corresponds to the CONNECT($v_{\text{nearest}} \,.\, x$, $x_{\text{v,rand}}$) function in Bry & Roy (2011).

Having a collision free connection, i.e., an edge e_{new}, there needs to be at least one belief $n \in v_{\text{nearest}} \,.\, N$ that can be propagated without collision along e_{new}. This is done by applying our state estimation filter from Section 2.1 on the measurements, and system inputs (cf. Section 2.2) that were generated while creating e_{new}. The initial state $x_{\text{f,init}}$ of the filter is set to

$$\boldsymbol{x}_{\text{f,init}} = [v_{\text{nearest}} \,.\, x \,.\, \boldsymbol{p}^T, v_{\text{nearest}} \,.\, x \,.\, \boldsymbol{v}^T, \bar{q}(v_{\text{nearest}} \,.\, \boldsymbol{a}^T, v_{\text{nearest}} \,.\, \psi), \boldsymbol{b}_\omega^T, \boldsymbol{b}_a^T, \lambda, \boldsymbol{p}_i^s, \bar{q}_i^s]^T. \quad (30)$$

Position and velocity can be directly obtained from $v_{\text{nearest}} \,.\, x$ while the attitude quaternion $\bar{q}$ is a function from the acceleration and yaw angle at v_{nearest}, as explained in Section 2.2. The remaining states can be set constant.[4] This works since we are only interested in the evolution of the state covariance during filter execution along the new edge.[5] The initial state covariance for the filter is simply $n \,.\, \Sigma$. During propagation along e_{new}, the path is also checked for possible collisions by taking the state covariance into account. This corresponds to the PROPAGATE(e, n) function in Bry & Roy (2011).

We define the cost for flying along an edge e as an integral over the thrust $\int |\boldsymbol{t}|$, minimizing the power consumption and time necessary to reach the goal (cf. Section 4.2). This seems to contradict the need for excitation of the vehicle for its states to stay observable. However, this is a tradeoff between excitation reducing the uncertainty of the vehicle's states and energy efficiency. That is, the vehicle gets just as excited as necessary to reach the goal within the defined uncertainty region.

Upon the success of the previous propagation step, new edges are created from v_{new} to v_{nearest}, from v_{new} to V_{near}, and from V_{near} to v_{new}. This time exact connections are created by fixing all constraints in Eq. (22). Whenever an outgoing edge is added to an existing vertex, all its belief nodes are added

[4] This would be states such as intersensor calibration or sensor biases, which do not change with the vehicle's dynamics.

[5] A known issue of the EKF covariance estimate is that the estimated value may not reflect a correct uncertainty due to linearization, as discussed by Huang, Mourikis, & Roumeliotis (2011). For the sake of simplicity, we neglect this issue in this paper.

Figure 5. The area where parts of the field tests were conducted as seen from the onboard camera (right) with colored dots indicating estimated landmark locations from the SLAM map. Start point (red cross) and end point (blue circle) of the experiments are indicated. A series of containers, trees, and a large house create a realistic environment as found in many commercial applications of UAVs, such as mapping, inspection, or aerial imagery (left image: Bing Maps)

to a search queue, which is then exhaustively searched. Its AppendBelief() function [cf. Bry & Roy (2011)] uses the methods discussed in Section 4.4 to decide if a belief is added to a state vertex (29) and if it dominates and thus can prune existing beliefs (26).

Similarly to the approach in Webb & van den Berg (2012), we want to reach the goal state exactly and centered in the goal region. This is due to the region also defining the maximum uncertainty at the goal. Also, we do not want to wait until this state gets sampled by chance in the goal state, since we believe that a nonoptimized path to the goal in an early iteration is better than no path. In contrast to the work of Webb & van den Berg (2012), we cannot simply add the goal state to the set of *near* vertices. In this stage, the goal state is not part of the set of state vertices and thus has no belief nodes that can be propagated to/from the newly sampled state. Therefore, we explicitly try to "sample" the goal state before an iteration of the RRBT. If a successful connection can be made, we proceed with a RRBT iteration, using the goal as a newly sampled state. If this connection fails, we proceed with a regular RRBT iteration by sampling a random state.

We aim to keep the (tuning) parameter space small: In fact, most parameters are system parameters, such as maximum velocity or acceleration, which are easy to figure out depending on the system at hand. The search radius r around v_{new} for near vertices is determined by $r = [\log(n)/n]^{(1/d)}$ according to Bry & Roy (2011) and Karaman & Frazzoli (2010), where n is the number of state vertices and d is the dimension of the sampled state x_s. ϵ in Eq. (29) is probably the most interesting and only tuning parameter. It determines how many (almost similar) belief nodes are considered in the planning process. We set this value to a small percentage of the reference ellipsoid's measure (cf. Section 4.4).

5. EXPERIMENTAL SETUP

5.1. Navigation in an Outdoor Scenario

We conducted our field experiments at a medium-sized maintenance area in which a large house, several 20-feet containers, and trees provided a realistic scenario (see Figure 5).

In our experiments, we use the visual SLAM algorithm *parallel tracking and mapping* (PTAM) (Klein & Murray, 2007) in an adapted version. We work with the same modifications to this SLAM algorithm as described in our previous work (Weiss et al., 2013), which increase the robustness and performance for the MAV navigation scenario. The algorithm performs SLAM by building a local map and performing bundle-adjustment over it to estimate a map containing landmarks from which the camera pose can be calculated using PnP. Given the adaptations we carried out, the framework allows us to perform large-scale navigation and mapping with constant computational complexity computed on the platform's embedded processor. The 6 DoF pose measurements estimated with respect to this local map are fused online with IMU measurements in an EKF for low latency pose estimation. These online estimates are used for control of the MAV at 1 kHz.

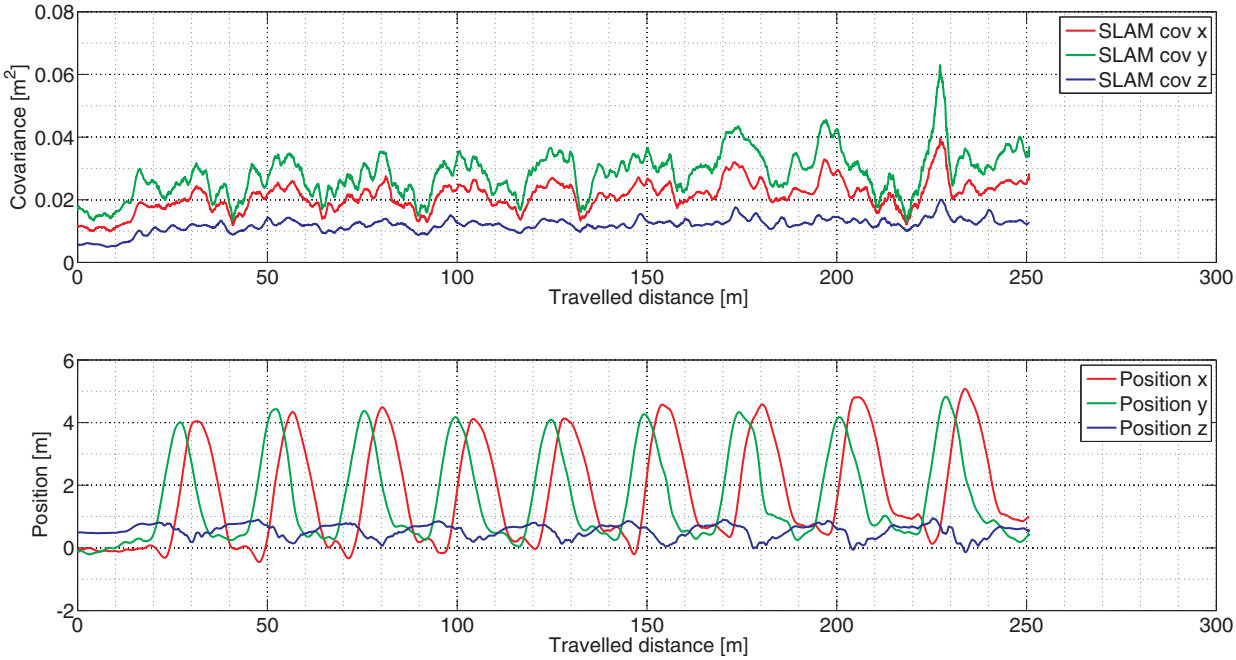


Figure 6. The covariances as reported from the SLAM system when traversing the same area multiple times (closed-loop flying in circles using the proposed framework for trajectory planning). Because we keep only a limited number of key frames, the local map is rebuilt when revisiting the scene. Different structures at the ground provide less visual information than others. This is reflected in the covariances, which for certain parts of the trajectory are higher than others. This repetitive pattern can be seen in the figure, highlighting the fact that the localization is repeatable over multiple observations of the same scene. The SLAM-landmark estimates can be stored in a map to later allow trajectory planning that takes the localization uncertainty into account

5.2. Obtaining the Pose Covariance from a Key-frame-based Visual SLAM System

For both simulation and real-world experiments, we assume that a map of the environment is available for path planning. For our experiments, we employ the same SLAM framework to build this map as we do for online navigation of the MAV (see Figure 6).

In contrast to EKF SLAM as in Davison et al. (2007), the landmarks in key-frame-based SLAM systems are commonly not represented with uncertainty. Instead of deriving the uncertainty in the camera pose from the propagation of the pose and the visible landmarks as in EKF SLAM, the covariance of the camera-pose is obtained from *bundle adjustment*. This nonlinear optimization solves simultaneously for both three-dimensional (3D) map points in the world and the camera locations over time. This is done by minimizing the weighted-least-squares reprojection errors of the 3D map points across all images (augmented by noise with constant covariance). This provides an estimate of the pose covariance Σ_{cam} from the pseudoinverse $\boldsymbol{S}^{+}$ of the innovation covariance $\boldsymbol{S}$ (Hartley & Zisserman, 2004), as $\Sigma_{\text{cam}} = \boldsymbol{S}^{+}$. The matrix $\boldsymbol{S}$ is a block matrix of dimensionality $n \times n$ (n is the number of visible landmarks) with blocks $\boldsymbol{S}_{ik}$ defined as

$$\boldsymbol{S}_{jk} = -\sum_{i} \boldsymbol{W}_{ij} \boldsymbol{W}_{ik}^{T}, \tag{31}$$

where $\boldsymbol{W}$ is the matrix of weighted Jacobians of the projection with respect to the camera location parameters a such that $\boldsymbol{W}_{ij} = [\partial x_{ij}/\partial a_j]^T \Sigma_{x_{ij}}^{-1}$. To build $\Sigma_{x_{ij}}$, the reprojection errors are weighted with a robust weighting function and commonly augmented with (reprojection) noise, which is independent of time and position of the map-point in the world-frame.

We can estimate the covariance for an arbitrary pose in the map by holding the landmarks fixed and applying PnP. This pose is then provided to the planner, which uses it as measurement in the state estimation pipeline. We compute the pose-measurement covariance as in Eq. (31) without minimizing the reprojection errors and adjusting the landmark locations. This provides the covariance of the sampled pose T_{map}^{k} with respect to the local map. This is the same algorithm carried out online by the pose tracker of the employed SLAM system (Klein, 2006).

It is common knowledge that already three known, well-distributed 3D landmarks in the map constrain a

camera pose in 6 DoF. The covariance computed from the landmarks therefore drops significantly as soon as a minimal set of points is in the field of view. To account for larger reprojection errors, false triangulation, and uncertain tracker performance during the online-localization, we scale this covariance by the number of features in the field of view to give more weight to more densely populated areas with evenly distributed features. This and an additional scaling factor were discovered to match the covariances found during experimental tracking runs with the employed SLAM system. Another promising approach for obtaining covariance estimates by learning through likelihood estimation was very recently proposed by Vega-Brown, Bachrach, Bry, Kelly, & Roy (2013). However, we leave this for future enhancements.

The proposed approach of using PnP and weighting by feature count allows us to take feature-distribution, lack of features, and their estimation uncertainty into account for planning. However, since we only simulate the pose estimation process, we are not able to estimate the localization uncertainty under the influence of lighting or structural changes. As with any feature detector, the recall of the employed FAST-like detector will deteriorate as the images experience large changes in contrast. While we do not handle this case explicitly, lighting changes result in a reduction of feature density that is handled by the planner.

This means that poorly textured areas, or those affected by lighting-induced contrast loss, are avoided either by flying around them or ascending to altitudes from which a wider area can be overlooked, as shown in Figure 17. On the other hand, areas with a higher density of features (feature-rich), and a more homogeneous feature distribution over the image, result in a pose estimate with lower covariance as expected from PnP, and they are therefore preferred by the planner.

The tracker implementation inside PTAM does not take into account the quality of the landmarks (e.g., number of observations) when estimating the camera pose with respect to the local map. This does not allow us to weight landmarks differently depending on their quality when estimating the pose-measurement covariance. We therefore do not add poorly constrained landmarks to the map of the environment in the first place, so that respective areas in the map appear as unmapped.

From these data, the planner can then minimize the covariance of the MAV state from start to goal. Using a known map of the environment, the planner can optimize the trajectory also for areas that are not in the field of view from the current location of the MAV, and therefore provide a *globally* optimal trajectory from start to goal.

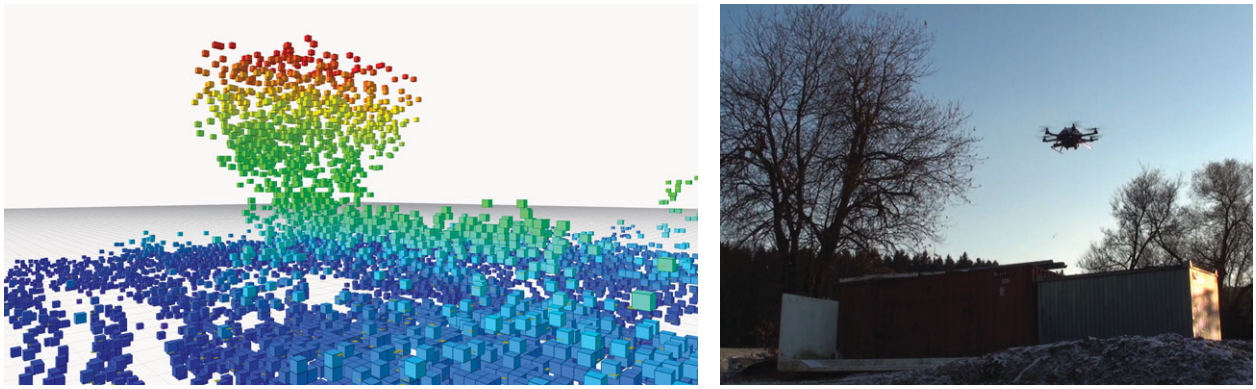
To obtain the known map in which to perform path planning, we flew the helicopter by remote control for several flights, each about 8 min in length (close to the MAV battery lifetime), observing and mapping an area of 30×50 m^2 multiple times. Flying a helicopter manually is, however, not a requirement for the proposed method. Instead, one could imagine a scenario in which a GPS-based flyover maneuver of the scene at an obstacle-free altitude could provide a rough map, which is then used by the same vehicle for approach and landing maneuvers closer to the ground. The map would be used both for obstacle avoidance and finding a path that provides sufficient information to perform the requested maneuver. As future work, we plan to use sensor information gathered using other platforms to reconstruct the scenery offline. Using Google Earth 3D, satellite images, or data captured by airplanes, one could construct a map of the environment in question, allowing path-planning and obstacle avoidance in previously unvisited areas.

Because the maps from different platforms or even different times of a day would make data-association challenging, we decided not to localize from the known map directly when executing the planned trajectory. We rather use it as a source of information about how well certain areas of the map support vision-based localization. The actual maneuver is then carried out with the SLAM system in the loop, but building a new map while following the trajectory that was optimized for the known map. Figure 6 shows an experiment about repeatability of covariance computation along a trajectory across different maps. Obviously there will be differences in scene appearance when flying at different altitudes. Our SLAM system, however, partially mitigates this fact by detecting and tracking features at different image scales.

5.3. Obstacle Avoidance

Obstacle avoidance is performed while making local connections between two state vertices, as described in Section 4.2. We check local paths for collisions and do not add them to the tree if a collision occurs. Furthermore, during uncertainty propagation along the edges, we inflate the bounding box around the vehicle by a multiple of the standard deviation of position in order to plan a safe path. This factor depends on the risk of hitting an obstacle, and is commonly set to 3σ.

The collision checks are performed using OctoMap (Hornung, Wurm, Bennewitz, Stachniss, & Burgard, 2013) representing the environment, allowing for efficient storage and fast look-ups. A straightforward creation of the OctoMap could be done by adding the triangulated and refined landmarks from the visual SLAM system to the map. However, even after bundle-adjustment, the points in the SLAM map contain a significant number of outliers, resulting in many spurious points blocking parts of the free space. OctoMap provides an option to insert laser scans to the map, consisting of a 3 DoF position and a set of points perceived from this location. We can use this interface to add virtual scans consisting of key-frame locations plus respective landmark observations. The statistical model included in OctoMap is then able to remove outlier points if they are



(a) (b)

Figure 7. OctoMap representation (a) of a tree and two containers placed on its right. The photo on the right depicts the real scenario. We observed that this map is sufficiently dense in order to plan paths around larger structures. The OctoMap is created from sparse visual landmarks reconstructed during a flyover at a safe altitude

intersected by rays from these virtual scans. Thereby, we can significantly reduce the number of outliers in the map and make it useful for 3D obstacle avoidance.

The current approach handles static, previously known obstacles only. Avoiding dynamic obstacles could be handled in two stages: First, when a new obstacle is detected, local (reactive) obstacle avoidance would steer around the obstacle. Second, for small deviations from the originally planned path, a local path toward the nearest state vertex on the optimized path could be planned and followed from then on. It may even be possible to use the proposed RRBT approach locally with the newly detected obstacle. For larger deviations or failure of local replanning, the whole path will have to be replanned.

The density of the obstacles represented in the map naturally depends on the amount of visual structure of these objects. While we observed reconstructions of trees or larger structures using our SLAM pipeline to be sufficiently dense for obstacle avoidance, our system is not able to handle structures with little/no visual texture, e.g., power lines. An example OctoMap representation of a tree can be seen in Figure 7. However, given that the proposed system is not dependent on the source of map-information, this is not a limitation of the work presented in this article, as one could use, for example, dense-stereo or laser-based sensory information to handle these situations. A strength of this work is the ability to take into account the limitations of the sensor-suite and the navigation and map-building modules available to the system at the time of path planning.

6. EXPERIMENTS: FROM SIMULATION TO FIELD TESTS

The requirements of the planner are manifold. First, we want to plan paths that enable our motion-dependent states to converge better and faster. This can be shown by comparing a direct connection between start and goal and a path that our planner found and considered as optimal in the sense that uncertainty gets reduced while just using as much excitation as necessary. Secondly, the planner has to be able to plan more complex paths for navigating in a real-world scenario in the presence of (static) obstacles. Finally, we are interested in planning paths where the onboard sensors are able to localize reliably, or in other words, maximizing the information. We start by showing simulation experiments and then describe in detail how this can be applied in real-world scenarios.

6.1. Motion-aware Path Planning

In the following, we pick a few representative states of our state estimation filter and show how our proposed method improves its estimates and convergence by exciting the vehicle in the right way. The setup is the following: we want to fly the MAV from a starting location along the x axis to a goal location in a distance of 10 m. The resulting path from our method can be seen in Figure 8, which differs only slightly from a direct path.

We set an initial state covariance that we obtained during real experiments with our framework described in Lynen et al. (2013). We obtain the system inputs (acceleration, angular velocity) and measurements for our state estimation framework (Section 2.1) from the values computed in Section 2.2 based on the position and its derivatives along the partial paths. Measurement uncertainty for 3 DoF position and 3 DoF attitude are kept uniform in this first experiment for simplicity and in order to cancel out side effects. Since the presented approach is a sampling-based technique, different measurement uncertainties can be incorporated, as shown in Section 5.2 and in subsequent experiments.

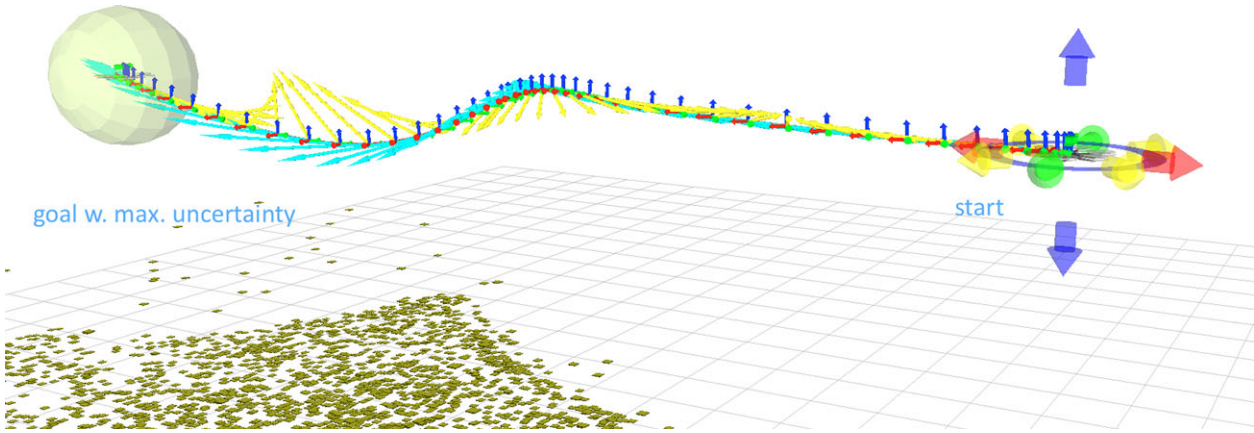


Figure 8. The proposed method computes and optimizes a trajectory taking into account the vehicle dynamics and generating velocity (cyan) and acceleration (yellow) control inputs to follow the desired trajectory. The system uses the features of the SLAM map to localize and estimate the covariance of pose updates on the path to the goal. This information is input to the planner, which optimizes a nondirect path that reduces the uncertainty on biases and position. This allows the system to reach the goal region defined by the transparent green sphere with sufficiently low covariance

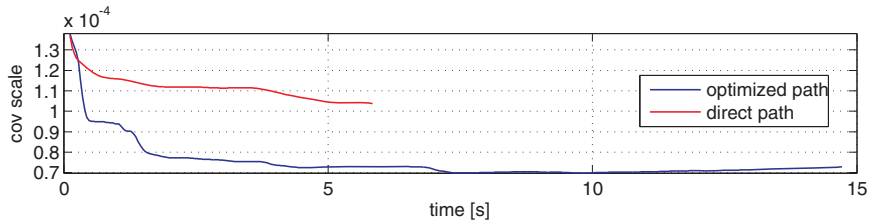


Figure 9. Comparison of the evolution of uncertainty of the visual scale over time for the direct path (red) and the optimized path (blue). While following the direct path takes substantially less time, the final scale covariance prevents reaching the goal region with the required certainty, which in contrast can be achieved by following the optimized path

Figure 9 shows the evolution of the uncertainty of the visual scale along the direct path and the optimized path. The uncertainty is not only significantly lower for the optimized path, but also converges faster. The visual scale directly influences the uncertainty and the quality of the position estimate. This is important when the vehicle moves away from its origin, since the visual scale influences the position estimate multiplicative [see Eq. (8)]. As a result, even if the position was measured correctly and without any drift, the position uncertainty grows with increasing distance if the visual scale is uncertain. This is shown in Figure 10: in the top left, the uncertainty of the position in the x axis for the optimized path (blue) p_x decreases until $t = 5$ s, which corresponds to the improvement of the visual scale uncertainty in Figure 9 due to excitation of the system. After $t = 5$ s, the visual scale has converged. Since the system moves away from the origin in the positive x direction (Figure 10, bottom left), the uncertainty for p_x starts growing again. The same applies to the direct path (red). Due to the lack of excitation, the described behavior happens notably faster. Since the trajectories for p_y (bottom right) stay close to 0 on the right side of Figure 10, the uncertainty (top right) for both the direct and the optimized path decreases, while the optimized path is performing remarkably better.

The intersensor calibration state $\bar{q}_i^s$ plays another important role (see Figure 11). This is a quaternion rotating pose measurement from the sensor's reference frame into the reference frame of the IMU. Errors in this state would cause IMU readings to be misaligned with pose measurements, and thus affect the accuracy of other states. Also for this state, the optimized path outperforms the direct path in terms of uncertainty and convergence speed. The different times of the optimized and direct path result from actuator limitations that we set.

Figure 12 shows the evolution of the cost and the path length over the iterations of the algorithm. Note that the small number of iterations denotes the number of updates of the optimized path. During the simulation, 150 state vertices were sampled in a volume around the direct path, having 750 belief nodes. The cost decreases (top) as expected while the path length (bottom) occasionally increases. This is natural, since we chose the integral of the squared norm

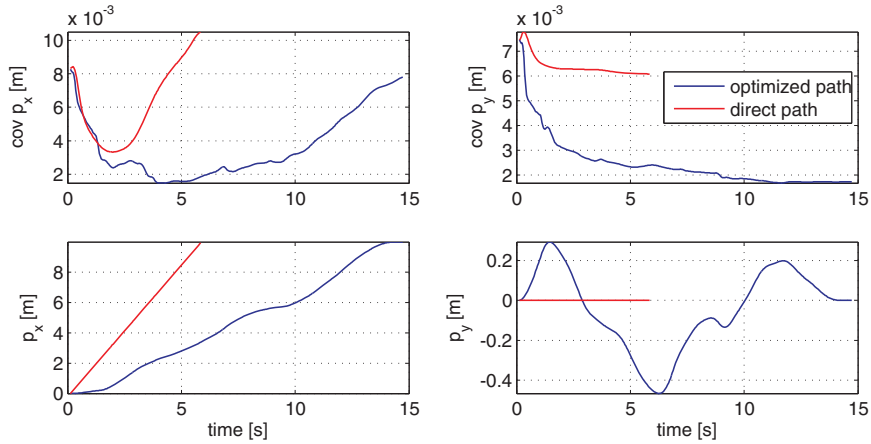


Figure 10. Behavior of the uncertainty for the direct path (red) and the optimized path (blue) in position for the x axis (left) and the y axis (right). While the time required to follow the direct path is substantially lower than for the optimized path, the optimized path leads to a substantially lower final position uncertainty. Since the visual scale error is multiplicative, the uncertainty of the position grows after the initialization phase with increasing distance to the origin. The trajectory in the y axis stays close to zero, which is why the uncertainty decreases

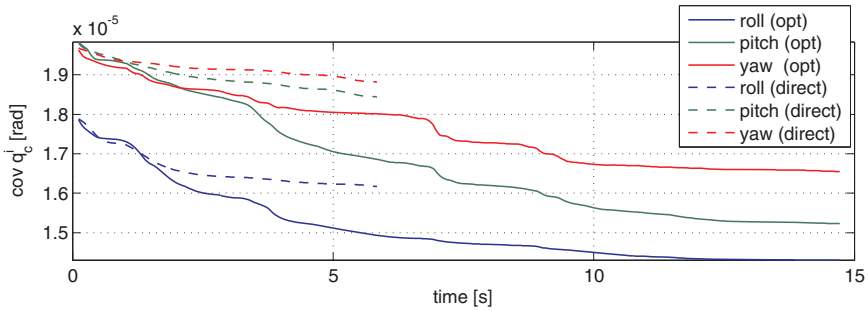


Figure 11. Comparison of the intersensor calibration state $\bar{q}_i^s$. The optimized path outperforms the direct path in terms of uncertainty and convergence speed. This state relates the measurements of the visual SLAM system to the IMU frame of reference, which means its uncertainty and convergence rates are of major interest

of the acceleration as cost. This graph shows that we obtain major improvements in the quality of the state estimation, with only a slightly longer path (≈ 0.6 m). In fact, the direct path would not have reached the goal with its covariance ellipsoid within the uncertainty region around the goal.

6.2. Obstacle Avoidance

Figure 13 shows a path-planning scenario with two obstacles. The figures show the first successful path, an intermediate path, and the final optimized path (from left to right). Cyan arrows indicate velocity while yellow arrows refer to the acceleration. Figure 14 shows the uncertainty on the left and the corresponding cost on the right (in Figure 13: left, blue; middle, red; right, magenta). It can be seen that the planner trades cost for uncertainty, as long as the final uncertainty is within the predefined bounds. Figure 14 also shows the relation between uncertainty and the required motion: During the almost straight sections of Figure 13, middle/right, uncertainty grows, while it decreases during acceleration/deceleration phases at the beginning and at the end.

6.3. Closed-loop Experiments

First, we evaluated the tracking performance of the controller in the Flying Machine Arena (Lupashin, Schöllig, Sherback, & D'Andrea, 2010), which is equipped with a Vicon motion capture system, providing us with ground-truth. To isolate the evaluation of the performance of the proposed controller from potential visual SLAM inaccuracies, we use the Vicon system for pose measurements as well. However, we distort the Vicon readings by adding delay, reducing the frame rate, and adding noise. We planned fast trajectories as shown in Figure 15, reaching track speeds of up to 3.5 $\frac{\mathrm{m}}{\mathrm{s}}$. The figure on the left is flown in a plane only,

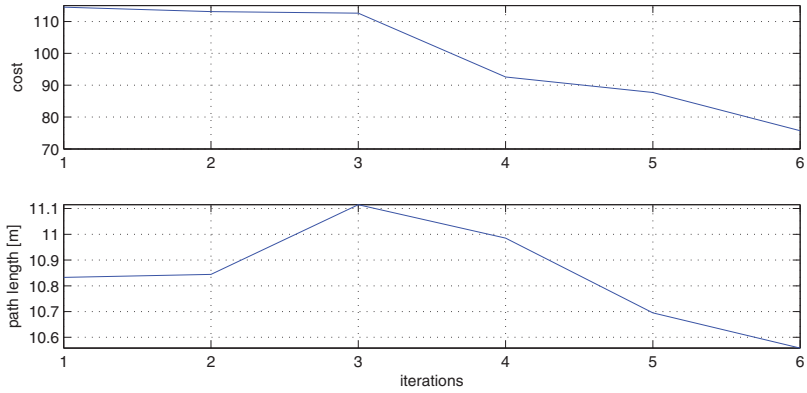
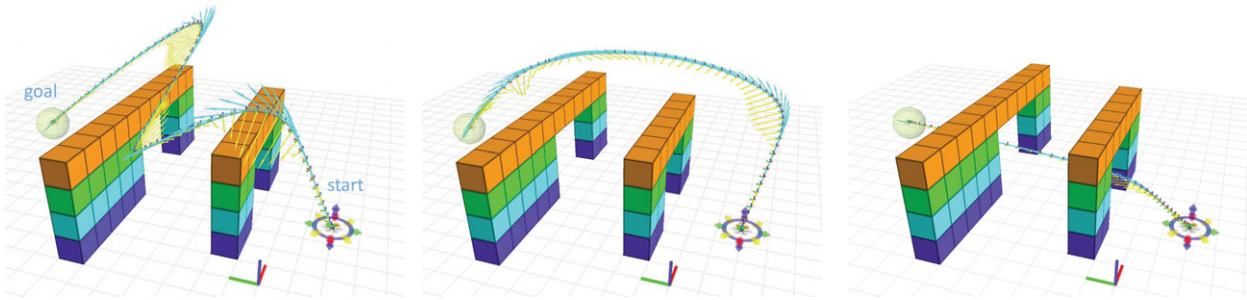


Figure 12. Cost and path length over the number of optimized path updates. During the simulation, 150 state vertices were sampled, resulting in 750 belief nodes. Finally, a path slightly longer than the direct path yields remarkably better results for our state estimation filter



(a) (b) (c)

Figure 13. First, intermediate, and best path found by the planner. These paths correspond to the blue, red, and magenta lines/bars in Fig. 14

while on the right, we added the third dimension. The polynomial path segments representing position and its derivatives are sampled at discrete time intervals of 10 ms, generating reference trajectories and feed-forward commands for the trajectory tracking controller. In these experiments, we obtained a root-mean-square (RMS) error of 11.8 and 7.7 cm for the trajectories, respectively, which is sufficiently accurate for our outdoor experiments.

With these results at hand, we tested the entire system at the outdoor test-site detailed in Section 5.1. We planned paths using obstacle as well as covariance information from the map with the approaches presented in this article. These paths, represented by polynomial segments according to Section 4.2, are uploaded to the vehicle. On the vehicle, we executed the full navigation pipeline consisting of visual localization (PTAM), state estimation, and control as presented in Sections 2.1 and 2.2. We conducted flights in the outdoor area depicted in Figure 5, all from the same start location to the same goal. The paths correspond to subsequently improved solutions of the optimized path from the planner. The flights were successfully conducted from the start to the goal—we show two paths with their uncertainties in Figure 16. The left side shows the covariances computed by the planner (bold) and the covariances reported by the state estimation framework. The graphs on the right denote the trajectories for position for each path (blue: top right, red: bottom right). We can see that the uncertainties from the planning phase closely match the actual uncertainties reported during the experiments, confirming the simulation results. Also, similar observations as for the obstacle avoidance can be made: during straight path segments, i.e., no acceleration, covariance grows as, for instance, during $0 < t < 12$ s for the first path and during $0 < t < 8$ s for the second path. Covariance, in contrast, shrinks during acceleration phases, such as for $12 < t < 26$ s and around $t = 10$ s for the first and the second path, respectively.

Our visual-inertial navigation system onboard the vehicle was thoroughly tested in our previous work (Achtelik et al., 2012; Weiss et al., 2013). In addition, we showed successful trajectory flights in this section first. Secondly, we showed that the vehicle is able to follow trajectories planned with the proposed method and that the estimated

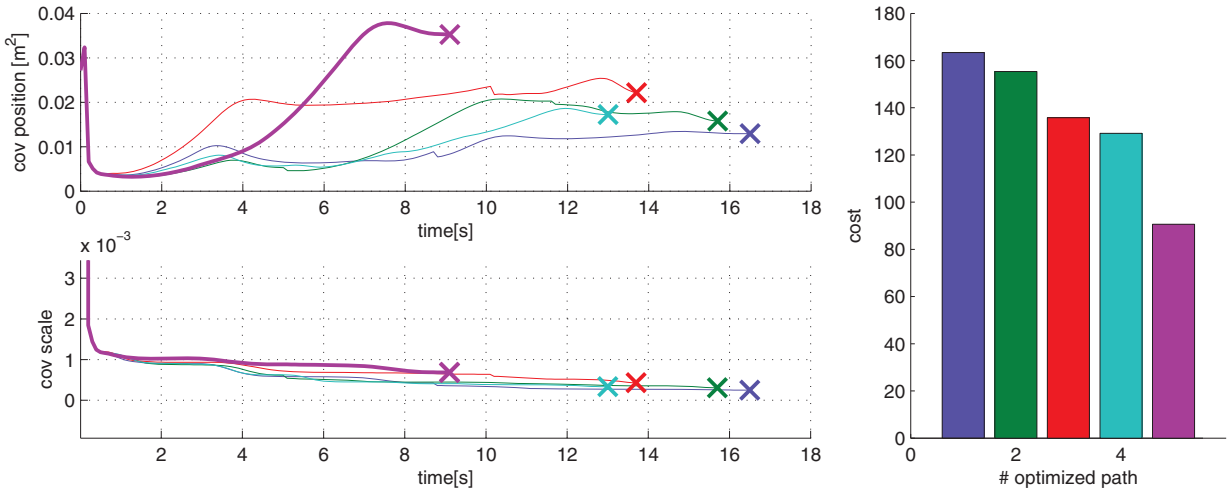


Figure 14. The left side shows the uncertainty in the 3D position and scale over several iterations of the optimized path, while the right plot shows the corresponding cost. Corresponding colors refer to one version of the optimized path. Blue refers to Figure 13(a), red to Figure 13(b), and magenta to Figure 13(c). Note that while the cost has to decrease, the uncertainty may be higher for the best path, compared to the other iterations, as long as it stays within the defined goal-region

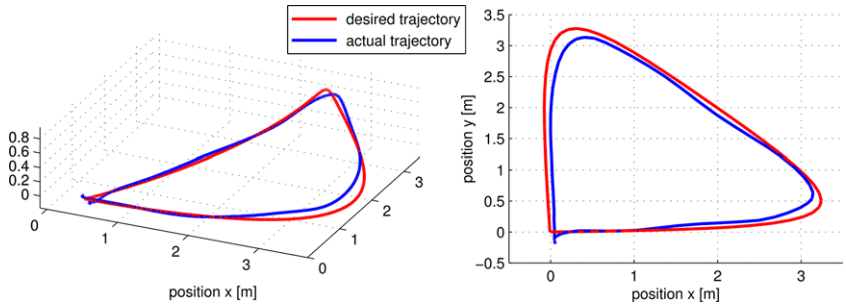


Figure 15. Fast trajectory flight. The track speed reached up to 3.5 $\frac{\text{m}}{\text{s}}$ and we obtained a RMS error of 11.8 and 7.7 cm, respectively, for left and right (Achtelik et al., 2013a)

covariances align well with the covariances from the real experiments. Having these results let us show the properties of our approach in more complex scenarios in the following experiments, where we focus on the outcome of the planning method in the given real scenario.

6.4. Uncertainty-aware Path Planning

In this experiment, we show that our approach is capable of planning feasible paths not only around obstacles, but also where the vision system is realistically able to localize appropriately. To improve clarity, we discuss here the planning module isolated from the other parts of the system. We require a minimum of five features to be successfully projected into the field of view before actually computing the measurement covariance as described in Section 5.2. Fewer features are treated as a "collision" in the planner, and the corresponding edge is removed. As a side note, one could allow traversing featureless areas if there is sufficient free space around the vehicle. This would be required, since integrating the IMU readings during this period without incorporating external measurements would result in a significant increase in position uncertainty. However, this assumes that the localization system is able to relocalize reliably after traversing this area—a risk that we want to avoid. Figure 17 shows one of the experiments: On the left, landmarks (features) are well distributed and the vehicle can fly an almost straight path. On the right, we cut out a stripe of landmarks. A possible scenario of that is, for example, crossing an asphalt road or a river, which usually have very poor visual features. In this case, the planner has to increase altitude such that landmarks on the boundary region of that stripe become visible in the field of view of the onboard camera.

Figure 18 shows an example in which a path around a featureless area is chosen. Since there is a way around this

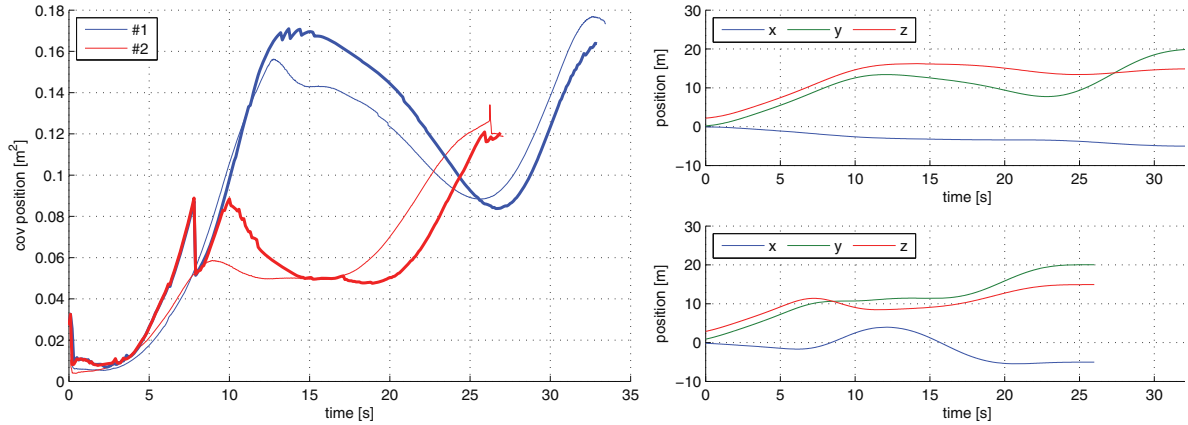


Figure 16. Closed-loop experiments in the outdoor scenario detailed in Section 5.1, using the full navigation pipeline. The left side shows the covariances for two different paths, while the right side shows the corresponding trajectories for position (blue: top right, red: bottom right). Bold lines denote the covariances computed by the planner, while thin lines correspond to the covariances reported by the state estimation framework

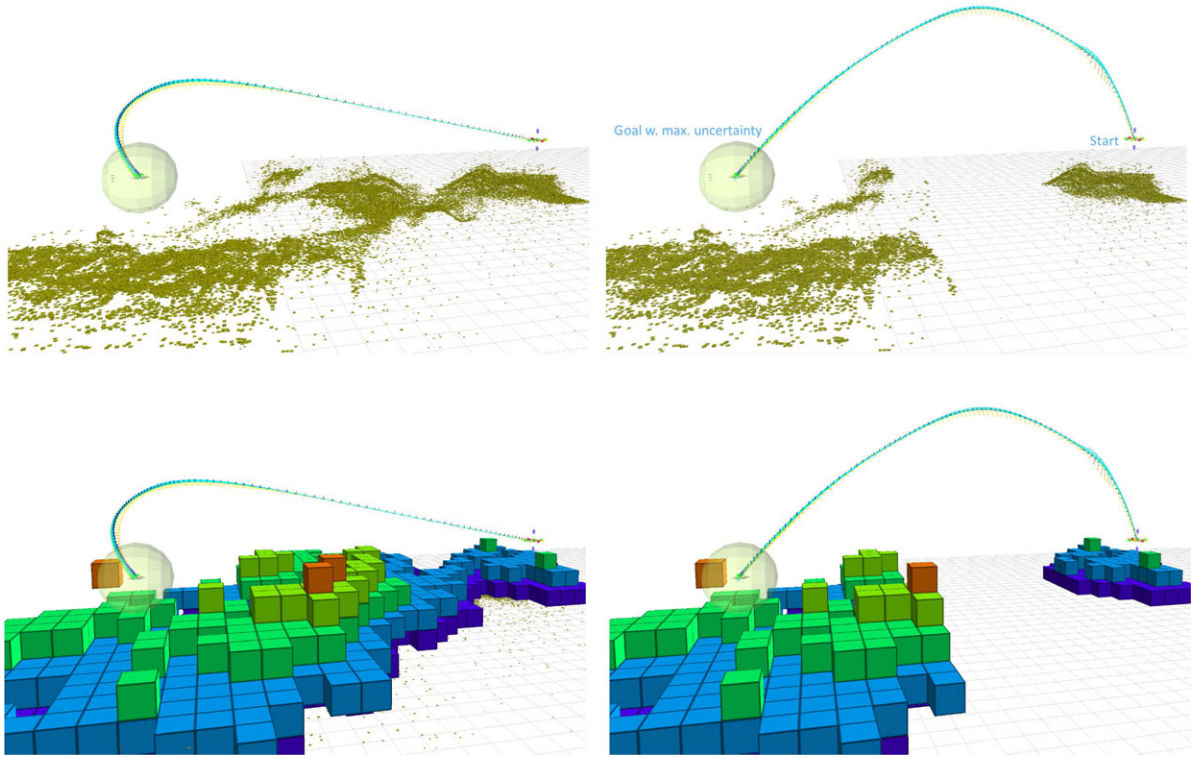


Figure 17. Taking into account the uncertainty and visibility of landmarks (green dots) in the planning phase, the proposed approach provides trajectories that allow the vehicle to safely traverse featureless areas. Given equally distributed landmarks (left), the vehicle can fly a relatively straight path to the goal region denoted by the transparent sphere 20 m away. However, if the map contains featureless areas (here artificially truncated), the planner calculates a path that comprises an increase in altitude for more landmarks to become visible. This situation might occur, for instance, when crossing asphalt roads or rivers, which usually have very few visual features. The landmarks are also used in an OctoMap (shown in the lower row images) for obstacle avoidance, which is computed according to Section 5.3

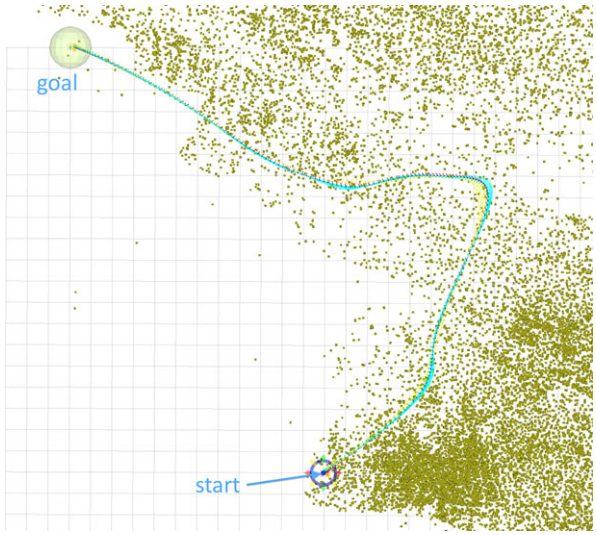


Figure 18. Example of navigating around featureless areas from start (circle) to goal (sphere) instead of across at higher altitude, saving energy. At the same time, the planner minimizes the covariance by choosing a path providing sufficient excitation. Corresponding covariances and costs are shown in Figure 19

spot, energy for increasing altitude is conserved, and paths around are planned sideways. The evolution of uncertainties, cost, and computation time can be seen in Figure 19. We also added a straight line connection for comparison: As soon as the feature-rich area is left, uncertainty increases rapidly. The uncertainty ellipsoid collapses shortly before the goal to reasonable values again—however, like above, this would require a perfectly working relocalization. Furthermore, the bounding box for obstacle checking would have to be highly enlarged, since crossing this area is essentially "blind" free-flight. It is also important to notice how the scale uncertainty gets reduced for the nondirect paths. In contrast, for the direct path, the scale uncertainty remains unchanged. This is because the scale has no time-dependent dynamics, combined with no updates being performed during the "blind" phase.

One could argue that featureless areas could be modeled as obstacles in general. However, this would either entirely exclude that area without the possibility of traversing it in higher altitude, or it would raise the question of how high this obstacle would have to be modeled (i.e., how untraversable it is), which would translate to an extra ad hoc tuning parameter. In contrast, by tightly incorporating measurement uncertainty in the planning phase, the planner finds an appropriate altitude to traverse planning around featureless areas and avoiding bad feature configurations (e.g., all features close to a line)—this would also permit seamless integration of additional sensors in the estimation.

In a final experiment, we demonstrate how the presented approach can handle heterogeneous environments, consisting of all the scenarios analyzed thus far. Figure 20 shows a path starting in front of the house (see Figure 5), navigating around a featureless area similar to Figure 18 first, before ascending to a safer altitude similar to Figure 17 over a sparser area. This is performed while avoiding obstacles, using the OctoMap-based occupancy grid described in Section 5.3, which finally guides the vehicle to the goal safely. The overall path length for this experiment was 67 m.

6.5. Computational Cost and Speedups

Computational complexity with respect to the number of belief nodes and iterations is discussed in Bry & Roy (2011). The main cost in our approach originates from two parts: first, from the computation of the uncertainty of the pose measurements in the real map (Section 5.2), and second, from the propagation of covariances along the edges of the RRBT. Since edges get revisited repetitively after adding state vertices and the following expansion of beliefs, we can cache the measurement uncertainties computed during the first propagation along that edge. An implementation using KD-trees ensures efficient and fast look-up of these cached covariances. Adding new edges to the cache requires rebalancing the KD-tree, but since edges get visited many times, this additional computational effort pays off. The other less obvious option was proposed by Prentice & Roy (2009) with the so-called "one-step transfer function" for propagating state uncertainties over edges within their proposed belief road maps. This one-step transfer function is computed during the first propagation along an edge based on the measurement and system input uncertainties. Once computed, this transfer function can propagate any covariance of an initial belief at once onto the final belief at the end of the edge. This also captures the expensive measurement uncertainty computation and its caching.

The drawback of the one-step transfer function method is that we lose the ability to perform collision checks along the edges based on our current state uncertainty. Since the uncertainty gets propagated in one step, we cannot inflate the bounding box around the vehicle by the state uncertainty (according to the risk the operator is willing to take) at intermediate steps of the edge. For this reason, we retain caching measurement uncertainties in the current implementation.

All path computations from this paper were computed on a recent Intel®Core™i7-based laptop within the range of minutes if not seconds for some smaller problems. Some computation times can be seen in Table I. It is important to notice that a first path is found relatively fast, and then gets gradually refined. Future work will focus on effective model reduction and an efficient search through the belief nodes, reducing complexity and enabling real-time operation.

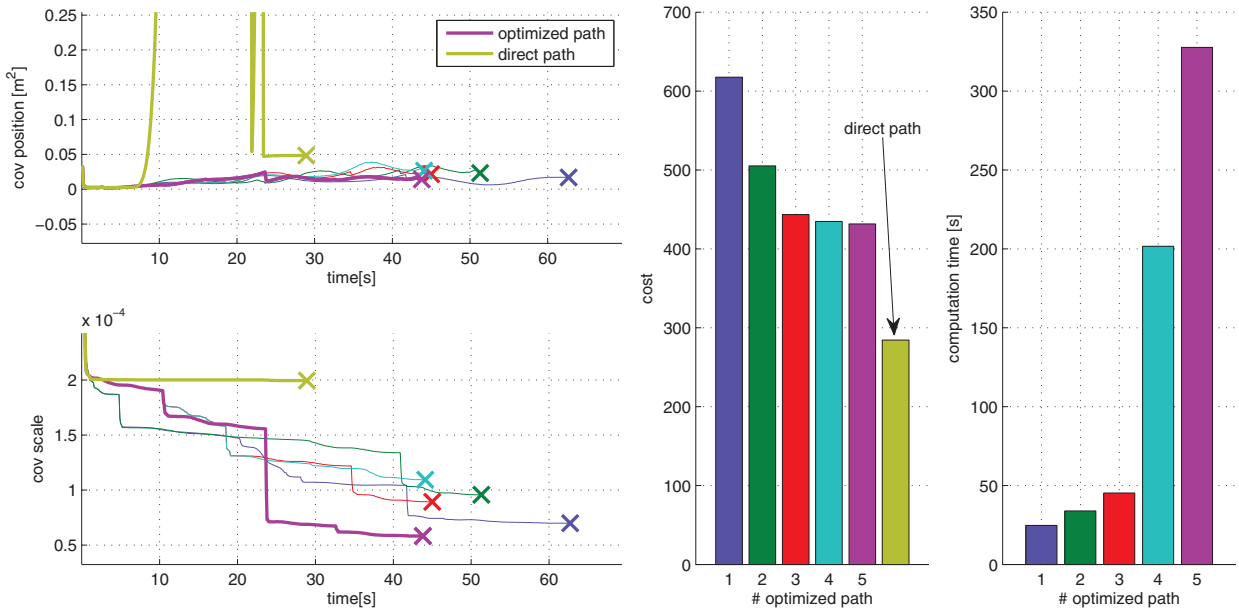
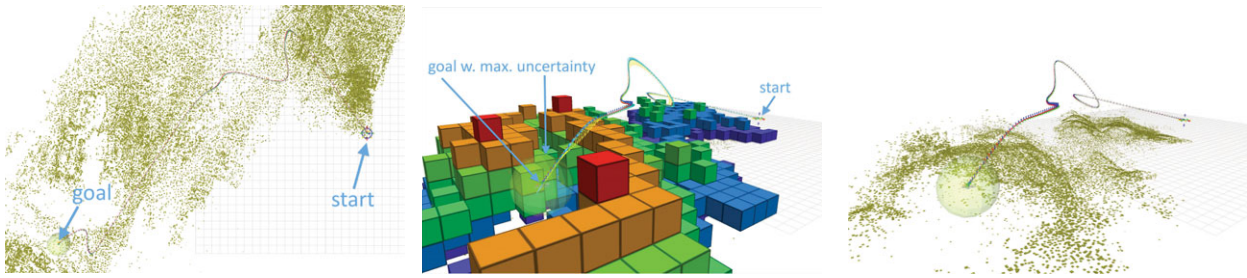


Figure 19. Uncertainty, cost, and computation times for avoiding featureless areas as shown in Figure 18. At the same time, the optimized paths provide sufficient excitation to reduce the position and scale covariance. The graphs also show the results of a direct path to the goal: at $t \approx 9$ s, no visual features are visible in the camera field of view. Without any visual updates, the covariance grows fast—a situation that can be effectively avoided by using our method



(a) (b) (c)

Figure 20. Full path from the take-off area in front of the house (see Fig. 5), avoiding a feature-less area and flying over a group of containers to the goal area. The overall traversed path length for this experiment was 67 m. In (a), it can be seen that the path is preferably planned over feature rich areas that allow precise localization. (b) and (c) show the landmarks and the resulting OctoMap representation which is used for obstacle avoidance

7. CONCLUSIONS

In this work, we showed how to combine a state-of-the-art path-planning framework with a complex state estimation framework for self-calibrating systems used on MAVs. As such, power-on-and-go systems need excitation before all their states can be rendered observable. In this article, we demonstrated how effective path planning can not only improve the error in the state estimates, but it can also allow for faster convergence after (re)initialization, all while planning a collision-free path (avoiding static obstacles). To our knowledge, this is the first work employing a path-planning framework such as RRBT, which takes into account the helicopter dynamics *while* maximizing the information along the path with the complex state estimation framework on the MAV. Thus far, only the one or the other has been shown.

We do not provide another path-planning method. Instead, we formulated our problem of MAV navigation, given the dynamic constraints of the MAV and its state estimation and visual localization framework, to be solved with a generic path-planning framework. Within this formulation, a fast and effective local planning method connecting state vertices was developed, which takes MAV dynamics into account. Instead of sampling in the high-dimensional

Table I. Timings for selected experiments that we performed.

Experiment	Time to first path (s)	Total time (s)	iterations
Uncertainty-aware planning, w/o gap (Figure 17 left)	1	24	152
Uncertainty-aware planning, w/ gap (Figure 17 right)	51	51	633
Uncertainty-aware planning around (Figure 18)	25	328	413
Full navigation scenario (Figure 20)	19	825	1,141

space describing the dynamics of the MAV (up to fourth derivative of position), sampling of state vertices was reduced to position and yaw only, while the remaining states were optimized by the local planner. Comparison of high-dimensional covariance ellipsoids turned out to be complicated due to incompatibility of states, such as, for instance, position and gyro biases. A solution was proposed that provides a good tradeoff between completeness and computational complexity (in terms of the number of belief nodes), based on a reference ellipsoid found in experiments.

We put the proposed methodology to the test via a diverse set of experiments, demonstrating that we can successfully plan paths in a medium-sized outdoor environment, taking into account the quality of localization and the full vehicle dynamics, in order to render all motion-dependent states observable. We first presented simulation experiments under simplified conditions, showing that a direct path may not be the best path, given the observability constraints from the state estimation framework. These experiments also showed that accurate estimation of the scaling factor of monocular vision-based localization—which needs excitation—is essential for reducing position uncertainty when flying further away from the origin, highlighting the necessity of a planning approach such as that proposed herein. Experiments in a real scenario were conducted in which a map of the environment was created by a flyover at a safe altitude. Insight was given on how this map provides the planner with uncertainty and static obstacle information. The experiments show that safe and feasible paths were planned while keeping state estimation uncertainty within the desired bounds. If necessary, obstacles were circumnavigated, and featureless areas were avoided by either flying around or by increasing altitude for more features to become visible in the field of view of the camera. Closed-loop experiments running the entire navigation pipeline confirmed that the covariance computed by the planner matches the covariance reported by the state estimation framework during the experiments.

Computational complexity in the described setup is certainly an issue. However, in the current state, we are close to real-time and we believe that full real-time operation will be possible for the presented scenario. At its current state, the system is intended for offline planning due to computational complexity, which also excludes obstacle avoidance for dynamic objects. We do not expect the planning area to be largely extended. In contrast, by reasonably reducing the planning area—experiments showed that optimized paths are close to straight line connections—and optimizations such as computing "one-step transfer functions" (Prentice & Roy, 2009) as well as a more efficient search through belief-nodes, we believe that the proposed method is real-time-capable and can thus be extended toward online mapping and replanning. The mentioned optimizations are on the road map for future enhancements, but they do not change the outcome of the methodology presented here.

In terms of scalability and in a broader context, we believe our approach is very suitable for local to medium-sized planning problems. We envision tasks such as approach, takeoff, and landing, as well as inspection tasks in confined spaces, for which we believe our approach scales very well. For larger scenarios, paths on a coarser scale could, for instance, be planned with probabilistic road maps, using straight-line connections. The proposed method can then be employed to plan paths along these straight connections while taking into account the full dynamics of the vehicle and the motion requirements from the state estimator. This would still be close to optimality, since our experiments showed that an optimized path does not deviate much from a straight connection.

In conclusion, we provided insight, discussion, and experiments on how to address the challenges arising with autonomous MAV operation based on visual localization during motion planning under uncertainty, resulting from observability requiring motion and bad visual feature configurations. Groups working on visual localization within the control loop are very familiar with the issues we addressed, therefore we see this article as a valuable contribution to the MAV research community.

ACKNOWLEDGMENTS

The authors gratefully acknowledge Adam Bry and Nick Roy for providing their generic RRBT framework and many helpful discussions, which led to the results of this work.

REFERENCES

Achtelik, M. W., Achtelik, M. C., Weiss, S., & Siegwart, R. (2011). Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China.

Achtelik, M. W., Lynen, S., Chli, M., & Siegwart, R. (2013a). Inversion based direct position control and trajectory following of micro aerial vehicles. In Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan.

Achtelik, M. W., Lynen, S., Weiss, S., Kneip, L., Chli, M., & Siegwart, R. (2012). Visual-inertial SLAM for a small helicopter in large outdoor environments. In Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS). Video: http://markus.achtelik.net/videos/IROS2012video.mp4.

Achtelik, M. W., Weiss, S., Chli, M., & Siegwart, R. (2013b). Path planning for motion dependent state estimation on micro aerial vehicles. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany.

Bry, A., & Roy, N. (2011). Rapidly-exploring random belief trees for motion planning under uncertainty. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China.

Bryson, M., Johnson-Roberson, M., & Sukkarieh, S. (2009). Airborne smoothing and mapping using vision and inertial sensors. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan.

Choudhury, S., Scherer, S., & Singh, S. (2013). RRT*-AR: Sampling-based alternate routes planning with applications to autonomous emergency landing of a helicopter. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Cover, H., Choudhury, S., Scherer, S., & Singh, S. (2013). Sparse tangential network (SPARTAN): Motion planning for micro aerial vehicles. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany.

Davison, A. J., Molton, N. D., Reid, I., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6), 1052–1067.

Fraundorfer, F., Heng, L., Honegger, D., Lee, G. H., Meier, L., Taskanen, P., & Pollefeys, M. (2012). Vision-based autonomous mapping and exploration using a quadrotor MAV. In Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS).

Hartley, R., & Zisserman, A. (2004). Multiple view geometry in computer vision, 2nd ed. Cambridge University Press.

He, R., Prentice, S., & Roy, N. (2008). Planning in information space for a quadrotor helicopter in a gps-denied environment. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Pasadena, CA.

Hermann, R., & Krener, A. (1977). Nonlinear controllability and observability. IEEE Transactions on Automatic Control, 22(5).

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots. Software available at http://octomap.github.com.

Huang, G. P., Mourikis, A. I., & Roumeliotis, S. I. (2011). An observability-constrained sliding-window filter for SLAM. In Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS) (pp. 65–72), San Francisco.

Karaman, S., & Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. In Proceedings of Robotics: Science and Systems (RSS).

Kavraki, L. E., Švestka, P., Latombe, J.-C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4), 566–580.

Kelly, J., & Sukhatme, G. S. (2011). Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. International Journal of Robotics Research (IJRR), 30(1), 56–79.

Klein, G. (2006). Visual tracking for augmented reality. Ph.D. thesis, University of Cambridge.

Klein, G., & Murray, D. W. (2007). Parallel tracking and mapping for small AR workspaces. In Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR).

Kuffner, J. J., & LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. The Annals of Mathematical Statistics, 22(1), 79–86.

Lupashin, S., Schöllig, A., Sherback, M., & D'Andrea, R. (2010). A simple learning strategy for high-speed quadrocopter multi-flips. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Lynen, S., Achtelik, M. W., Weiss, S., Chli, M., & Siegwart, R. (2013). A robust and modular multi-sensor fusion approach applied to MAV navigation. In Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan.

Mellinger, D., & Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China.

Mellinger, D., Kushleyev, A., & Kumar, V. (2012). Mixed-integer quadratic program (MIQP) trajectory generation for heterogeneous quadrotor teams. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Mirzaei, F., & Roumeliotis, S. (2008). A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation. IEEE Transactions on Robotics and Automation, 24(5), 1143–1156.

Prentice, S., & Roy, N. (2009). The belief roadmap: Efficient planning in belief space by factoring the covariance. International Journal of Robotics Research, 8(11-12), 1448–1465.

Richter, C., Bry, A., & Roy, N. (2013). Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In Proceedings of the International Symposium on Robotics Research (ISRR).

Roy, N., & Thrun, S. (1999). Coastal navigation with mobile robots. In Advances in Neural Information Processing Systems (NIPS '99).

Shen, S., Michael, N., & Kumar, V. (2011). Autonomous multi-floor indoor navigation with a computationally constrained MAV. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Turpin, M., Mohta, K., Michael, N., & Kumar, V. (2013). Goal assignment and trajectory planning for large teams of aerial robots. In Proceedings of Robotics: Science and Systems (RSS).

Vega-Brown, W., Bachrach, A., Bry, A., Kelly, J., & Roy, N. (2013). CELLO: A fast algorithm for covariance estimation. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany.

Webb, D. J., & van den Berg, J. (2012). Kinodynamic RRT*: Optimal motion planning for systems with linear differential constraints. CoRR, abs/1205.5088.

Weiss, S. (2012). Vision based navigation for micro helicopters. Ph.D. thesis, ETH Zurich.

Weiss, S., Achtelik, M., Chli, M., & Siegwart, R. (2012). Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), St. Paul, MN.

Weiss, S., Achtelik, M. W., Lynen, S., Achtelik, M. C., Kneip, L., Chli, M., & Siegwart, R. (2013). Monocular vision for long-term MAV state-estimation: A compendium. Journal of Field Robotics, 30(5), 803–831.

Weiss, S., & Siegwart, R. (2011). Real-time metric state estimation for modular vision-inertial systems. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China.

Wzorek, M., Kvarnström, J., & Doherty, P. (2010). Choosing path replanning strategies for unmanned aircraft systems. In International Conference on Automated Planning and Scheduling (ICAPS).