Path Planning for Motion Dependent State Estimation on Micro Aerial Vehicles

Markus W. Achtelik, Stephan Weiss, Margarita Chli and Roland Siegwart

***Abstract*— With navigation algorithms reaching a certain maturity in the field of mobile robots, the community now focuses on more advanced tasks like path planning towards increased autonomy. While the goal is to efficiently compute a path to a target destination, the uncertainty in the robot's perception cannot be ignored if a realistic path is to be computed. With most state of the art navigation systems providing the uncertainty in motion estimation, here we propose to exploit this information. This leads to a system that can plan safe avoidance of obstacles, and more importantly, it can actively aid navigation by choosing a path that minimizes the uncertainty in the monitored states. Our proposed approach is applicable to systems requiring certain excitations in order to render all their states observable, such as a MAV with visual-inertial based localization. In this work, we propose an approach which takes into account this necessary motion during path planning: by employing Rapidly exploring Random Belief Trees (RRBT), the proposed approach chooses a path to a goal which allows for best estimation of the robot's states, while inherently avoiding motion in unobservable modes. We discuss our findings within the scenario of vision-based aerial navigation as one of the most challenging navigation problem, requiring sufficient excitation to reach full observability.**

I. Introduction

Driven by the growing demand for more autonomous mobile robots, the research in system control and path planning has been advancing rapidly over the last few years. Before either of these problems can be successfully addressed, it is crucial that the states of the robotic platform are accurately estimated during the whole mission. How can a Micro Aerial Vehicle (MAV) compute a *realistic* path through the disaster area [1] if its estimated position is very uncertain? In the past, the focus of accurate state estimation has been on the controlled states such as 6DoF (Degrees of Freedom) pose and 3DoF velocity of the vehicle. However, with the recent emergence of self-calibrating power-on-and-go-systems ([2], [3], [4], [1]), it becomes more and more important to ensure fast state convergence at the beginning of the vehicle's task (or after re-initialization during the mission). Moreover, for self-calibrating systems it is crucial to continuously excite the system in such a way that the calibrating states (IMU biases, visual scale or the transformation between sensors) are accurately estimated throughout the whole mission.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7) under grant agreement n.266470 (myCopter). Markus Achtelik is currently PhD student, Margarita Chli is senior researcher and Roland Siegwart is full professor at the ETH Zurich and head of the Autonomous Systems Lab. Stephan Weiss is Technologist II in the Computer Vision Group of the NASA Jet Propulsion Laboratory. (email: {markus.achtelik, margarita.chli}@mavt.ethz.ch, stephan.m.weiss@jpl.nasa.gov, r.siegwart@ieee.org).

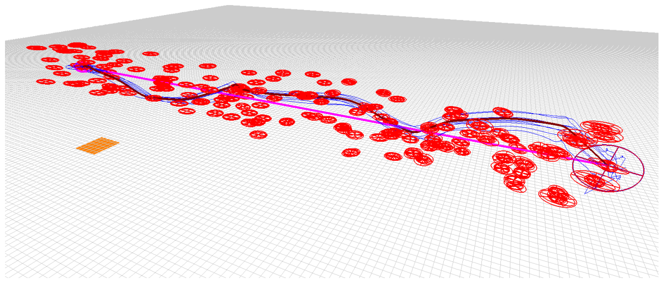

Fig. 1: The path computed by our proposed method for a distance of 10 m from start (on the right) to the goal (left). With the goal of minimizing the covariance of the whole system (for simplicity, visualized only for the x, y components of the position as 2D ellipses), the planner studies a multitude of candidate intermediate positions along the path before arriving to the final path promising the biggest reduction in uncertainty. Interestingly, flying on a direct path instead, would not reach the goal with the desired confidence in the state estimates, as not all states would experience enough excitation.

The studies in [1], [5], [6] on MAV navigation provide approaches which allow 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 avialability of landmarks has given rise to special planning algorithms favoring areas with more reliable landmarks between start and goal position [7], [8], [9], [10]. However, power-on-and-go systems usually have two additional requirements. Firstly, as shown in [2], [3], these systems need excitation in linear acceleration and angular velocity before all states become observable. This is particularly true for systems that estimate their inter-sensor calibration, in addition to the pure vehicle pose used for control. In hovering mode or while flying on a straight path, MAVs tend to be in an unobservable mode preventing correct estimation of all their system states. Secondly, for single-camera systems, if a visual map-loss occurs (which cannot be ruled out during a real mission) re-initialization of the system has to be performed. After such a re-initialization procedure, the metric scale has to be re-estimated quickly to allow continuous and robust vehicle control[1]. With the aim of providing robustness against such cases where typical planners would fail to produce a feasible path, we seek to find not only a short path to the destination, but a path where the states monitored in our system are best observable at all times. Fig. 1 illustrates an example path generated by our method, which aims at reducing the position uncertainty of the vehicle at the goal location, and where we reach the target within some given confidence area.

[1]monocular systems measure the 3D position only up to an arbitrary scale.

In this work, we essentially study the problem of not just acquiring *any* measurement, but in fact aquiring an *informative* measurement, such that the vehicle always remains in a fully observable mode. In order to accomplish this task we employ the Rapidly Exploring Random Belief Tree (RRBT) algorithm, developed by Bry and Roy [11]. This system can cope with non-holonomic constraints of a vehicle, while it plans a collision-free path (avoiding static obstacles) incorporating the uncertainty of the state estimate and the vehicle's controller dynamics. In this paper we incorporate our MAV navigation framework within RRBT, as one of the most challenging navigation scenarios given the high agility of the MAV, and the significance of prompt and sound state estimates in order to avoid crashes. This navigation framework has been discussed theoretically in [3] and evaluated quantitatively in [4]. Exploring the power of the RRBT framework in this scenario, we demonstrate how effective path planning can improve not only the error of the state estimates, but also allow for faster convergence of badly initialized states, steering the MAV to the goal position within a confidence region. 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: In Section II, we introduce our state estimation system and the dynamic model of the MAV used in this work. In Section III, we briefly sketch the RRBT algorithm and describe how we realized all its subtasks for this complex system. Section IV finally shows the results of our proposed approach.

II. System Description

In this section we describe the properties of our systems applied to the path planning algorithm. Care has to be taken about the term *state*. We have three different kinds of state:

- *filter state* x_f: refers to the state used in the Extended Kalman Filter (EKF) based estimation described in Section II-A. It contains the vehicle pose used for control and the (inter-) sensor calibration-parameters. One of the contributions of this work is to find the shortest path to a goal, while avoiding unobservable modes for all states in x_f at any time.
- *system state* x_d: refers to the states used to describe the system dynamics. There, the system's pose and its derivatives are of interest in order to grant smooth motion. (c.f. Section II-B)
- *sampling state* x_s: state in the space which we sample from when generating paths to new state vertices in the RRBT graph structure (cf. Section III-C).

A. 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 only sensors. For the system state estimation we use an Extended Kalman Filter (EKF) approach according to our previous work in [3], [4]. For completeness we briefly summarize the essentials.

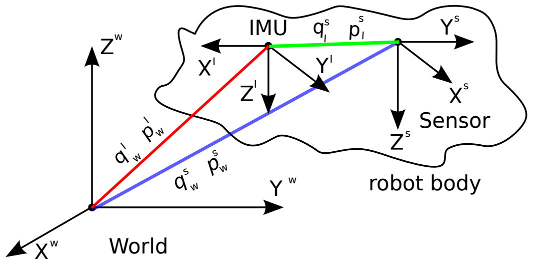

Fig. 2: Setup depicting the robot body with its sensors w.r.t. a world reference frame. The system's state is $x_f = \{p_w^i\ v_w^i\ q_w^i\ b_\omega\ b_a\ \lambda\ p_i^s\ q_i^s\}$ whereas p_w^s and q_w^s denote the robot's sensor measurements in (a possibly scaled) position and attitude respectively in a world frame.

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 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 $\boldsymbol{b}_a$, as well as the scale factor λ of the visual measurements. The calibration states are the rotation from the IMU frame to the camera sensor frame 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 Fig. 2. This yields a 24-element filter state vector x_f:

$$x_f = \left[\boldsymbol{p}_w^{i^T}\ \boldsymbol{v}_w^{i^T}\ q_w^{i^T}\ \boldsymbol{b}_\omega^T\ \boldsymbol{b}_a^T\ \lambda\ \boldsymbol{p}_i^s\ q_i^s\right]^T \tag{1}$$

The following differential equations govern the state:

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

$$\dot{\boldsymbol{v}}_w^i = C_{(q_w^i)}^T(\boldsymbol{a}_m - \boldsymbol{b}_a - \boldsymbol{n}_a) - \boldsymbol{g} \tag{3}$$

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

$$\dot{\boldsymbol{b}}_\omega = \boldsymbol{n}_{b_\omega} \quad \dot{\boldsymbol{b}}_a = \boldsymbol{n}_{b_a} \quad \dot{\lambda} = 0 \quad \dot{\boldsymbol{p}}_i^s = 0 \quad \dot{q}_i^s = 0 \tag{5}$$

With $\boldsymbol{g}$ as the gravity vector in the world frame and $\Omega(\omega)$ as the quaternion multiplication matrix of $\boldsymbol{\omega}$. 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 algorithm, we have the following measurement model with $C_{(q_w^i)}$ as the IMU's attitude in the world frame.

$$\boldsymbol{z}_p = \boldsymbol{p}_w^s = (\boldsymbol{p}_w^i + C_{(q_w^i)}^T \boldsymbol{p}_i^s)\lambda + \boldsymbol{n}_p \tag{6}$$

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

$$\boldsymbol{z}_q = q_w^s = q_i^s \otimes q_w^i \tag{7}$$

A non-linear observability analysis, as suggested in [12] and done in [2], reveals that all states are observable including the inter-sensor calibration states p_i^s (distance from the IMU to the sensor) and 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 proved in [2], [13].

B. Helicopter Model

For our goal of motion planning for a MAV, while ensuring that all states of the aforementioned state estimation filter stay observable, we will have to execute this filter along candidates of optimized paths. 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 [4]. Therefore we define the thrust in terms of acceleration in the z axis of the helicopter, and the body fixed angular velocity $\boldsymbol{\omega}_B$ 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 $\boldsymbol{\omega}_B$, we require a demanded control input to be continuous and differentiable.

For the aim of path planning, we need paths that can be followed by the helicopter given the aforementioned constraints. The findings by Mellinger and Kumar [14], about the differentially flat outputs $[x\ y\ z\ \psi]$ (i.e. position and yaw) for a quadrotor, can be applied to that problem: given a function for the position of the center of mass of the helicopter and its orientation (yaw), that are sufficiently differentiable, we can always compute the remaining states we need for the helicopter (attitude i.e. roll/pitch), and the required control inputs angular velocity $\boldsymbol{\omega}_B$ and thrust $|\boldsymbol{t}|$. The latter is simply a function of the attitude and rotor speeds. According to [14], the full attitude q can be computed as follows:

$$\boldsymbol{z}_B = \frac{\boldsymbol{t}}{|\boldsymbol{t}|};\ \boldsymbol{t} = \boldsymbol{a} + \begin{bmatrix}0\ 0\ g\end{bmatrix}^T \tag{8}$$

$$\boldsymbol{y}_B = \frac{\boldsymbol{z}_B \times \boldsymbol{x}_C}{|\boldsymbol{z}_B \times \boldsymbol{x}_C|};\ \boldsymbol{x}_C = \begin{bmatrix}\cos(\psi)\ \sin(\psi)\ 0\end{bmatrix}^T \tag{9}$$

$$\boldsymbol{x}_B = \boldsymbol{y}_B \times \boldsymbol{z}_B \tag{10}$$

$$\Rightarrow q = q({}^W R_B);\ {}^W R_B = \begin{bmatrix}\boldsymbol{x}_B\ \boldsymbol{y}_B\ \boldsymbol{z}_B\end{bmatrix} \tag{11}$$

$\boldsymbol{a}$ denotes the acceleration and ψ 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 $\begin{bmatrix}\boldsymbol{x}_B\ \boldsymbol{y}_B\ \boldsymbol{z}_B\end{bmatrix}$. This can also be seen as the rotation matrix ${}^W R_B$ that describes the orientation of the helicopter w.r.t. the world coordinate system. For computing ω_B, we need to compute an intermediate vector $\boldsymbol{h}_\omega$:

$$\boldsymbol{h}_\omega = \|\boldsymbol{t}\|^{-1} \cdot (\boldsymbol{j} - (\boldsymbol{z}_B \cdot \boldsymbol{j}) \cdot \boldsymbol{z}_B) \tag{12}$$

$$\omega_{B,x} = -\boldsymbol{h}_\omega^T \cdot \boldsymbol{y}_B \tag{13}$$

$$\omega_{B,y} = \boldsymbol{h}_\omega^T \cdot \boldsymbol{x}_B \tag{14}$$

$$\omega_{B,z} = \dot{\psi} \cdot \boldsymbol{z}_W^T \cdot \boldsymbol{z}_B \tag{15}$$

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 $\dot{\psi}$ (i.e. snap $\boldsymbol{s}$ and $\ddot{\psi}$) are continuous. We define the vehicle state x_v as follows:

$$x_v = \begin{bmatrix}\boldsymbol{p}^T & \boldsymbol{v}^T & \boldsymbol{a}^T & \boldsymbol{j}^T & \boldsymbol{s}^T & \psi & \omega_\psi & \dot{\omega}_\psi\end{bmatrix}^T \tag{16}$$

This state is needed for the local path planning in Section III-B. To summarize, $\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{a}, \boldsymbol{j}, \boldsymbol{s} \in \mathbb{R}^{3\times1}$ denote the position and its 4 derivatives velocity, acceleration, jerk and snap respectively. $\psi, \omega_\psi, \dot{\omega}_\psi \in \mathbb{R}$ denote the yaw angle, the angular rate and angular acceleration.

III. Path Planning Towards Optimized State Estimation

In the following we discuss the main properties of the RRBT algorithm as presented in [11], and describe how it can be employed in our challenging visual-inertial MAV state estimation framework.

A. 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 Λ estimates, and a cost c. Furthermore, it has pointers to its owning vertex v and to a parent belief node: From a current vertex v_c, following each of its belief nodes to their respective preceding parent belief nodes and their owning state vertices, describes unique paths through the graph from v_{start} to v_c with the properties Σ, Λ and 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 similarly 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 II-A, 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 covariance along that edge into account. On success, exact connections are made forth and back to a set of near vertices within a certain radius [11], [15], which also get propagated. After successful propagation, a new belief node is added to the corresponding vertex, if it is not dominated by a 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 not only allows us to set 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

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

region, where it can *localize* itself while providing *sufficient excitation* to keep the system's states observable.

B. Local Path Planning

In order 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 like Rapidly Exploring Random Trees (RRT), an *approximate* connection from v_{nearest} to v_new is sufficient[3], we need to make *exact* connections between v_{new} and V_{near}, and back from v_new to v_{nearest}. The main difficulty is that multicopters are under-actuated systems, and we require the fourth derivative of the position (snap) to be continuous (c.f. Section II-B). Furthermore, we want to be able to include actuator constraints (snap), but also constraints as maximum velocity and acceleration. During graph construction, many of those connections need to be created, thus we cannot afford sophisticated optimization techniques.

A simple and fast solution was proposed by Webb and Berg [16], but it requires linearization of the dynamics around the hovering point, and 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 collision. The result would be "no connection" and the next sample would be taken. This may work for simpler approaches as RRT and just results in more samples, but for RRG or RRBT this would lead to missed exact connections between existing state vertices which are actually collision free.

We decided to adapt the *minimum snap trajectory* approach by Mellinger and Kumar [14] to our needs. This approach uses N^{th} order polynomials with more degrees of freedom than initial- and final (equality) constraints, and 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 (c.f. Section II-B), we need at least 10 parameters (9^{th} order polynomials) plus some degrees of freedom for the optimizer. Unlike in [14], we chose to optimize over the integral of the squared norm of the acceleration instead of snap. The motivation of this choice was to make the helicopter as energy efficient as possible. 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:

$$f(t) = \boldsymbol{t} \cdot \boldsymbol{c}\,;\ \boldsymbol{t} = \begin{bmatrix} 1 & t & t^2 \dots t^{N-1} \end{bmatrix} \quad (17)$$

$$\min \int_{t_0}^{T} \left\| \frac{d^2 f(t)}{dt^2} \right\| \text{ s.t.} \quad (18)$$

$$\frac{d^n f(t_0)}{dt^n} = \frac{d^n (v_{\text{start}}.x)}{dt^n};\ n = 0 \dots 4 \quad (19)$$

$$f(T) = (v_{\text{end}}.x.p) \quad (20)$$

$$\frac{d^n f(T)}{dt^n} = \frac{d^n (v_{\text{end}}.x)}{dt^n} \text{ or free};\ n = 1 \dots 4 \quad (21)$$

Where $f(t)$ is a N^{th} order polynomial with coefficients $c_0 \dots c_N$ of the position, and p_{start}, p_{end} 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 when to keep the constraints in Eq. (21) fixed or leave them free, see Section III-C.

A nice property of the above optimization problem is that there are no 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. Nonlinear optimization with the time as additional parameter and inequality constraints are costly (and numerically problematic), therefore we were seeking for a simple and fast solution. We solve the aforementioned problem with a conservative estimate for T, compare this to pre-defined maximum values and then scale the path time by the constraint that is closest to its motion constraint, i.e.:

$$c(n) = \text{abs}\big((\frac{d^n f(t)}{dt^n})/c_{n,max}\big);\ n = 1 \dots 4 \quad (22)$$

$$T_{new} = \max(c(n)) \cdot T^{(1/\text{argmax}(c(n)))} \quad (23)$$

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 [14] does not work in our case, since it also scales the boundary conditions which are non-zero in our case.

C. Sampling of State Vertices

We decided to sample in the space of the differentially flat outputs position and yaw angle as defined in [14]. The sampling state becomes:

$$x_s = \begin{bmatrix} x\ y\ z\ \psi \end{bmatrix} \quad (24)$$

This choice is first 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 $+x$ direction w.r.t. 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 III-B while leaving the derivatives of position in the final condition from Eq. (21) free. After optimization of the local path, these free variables result from the optimization and thus define the full state at v_{new}. Since we optimize over the acceleration, the optimizer will not decelerate the vehicle towards the end of a local path, thus we implicitly sample the velocity and its derivatives in the direction of motion.

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

D. Covariance Comparison

From the ordering of partial paths represented by belief nodes, a belief node n_a dominates (is "better") a node n_b, as described in [11], if:

$$n_a < n_b \Leftrightarrow n_a.c < n_b.c \wedge n_a.\Sigma < n_b.\Sigma \wedge n_a.\Lambda < n_b.\Lambda \tag{25}$$

where $n.c$, $n.\Sigma$ and $n.\Lambda$ are the properties of a belief node (Section III-A). 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 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 being added to $v.N$, another check is performed if n_b dominates any existing belief node at v, and 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 2 or 3 dimensions in position, it becomes difficult for our system with 24 states: Not only because of the dimensionality, but also because the states are "incompatible": To give an example, how could an improvement of covariance in position be compared with the gyro biases? A a very conservative measure for $n_a.\Sigma$ dominating $n_b.\Sigma$ would be:

$$n_a.\Sigma < n_b.\Sigma \Leftrightarrow \min(\text{eig}(n_b.\Sigma - n_a.\Sigma)) > 0 \tag{26}$$

This in return means that any new belief node n_b, whose covariance is better in only one dimension, would be added to $v.N$. This results in many belief nodes being added, having to be propagated and 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 the volume of the covariance ellipsoid, or the trace. However, both cannot make a distinction between ellipsoids with high or low ratio of their axes. Furthermore, since our states are incompatible, 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 divergence [17] 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 computes as:

$$D_{KL} = \frac{1}{2}\left(\text{trace}(\Sigma_{ref}^{-1} \cdot n.\Sigma) - \ln\left(\frac{\det(n.\Sigma)}{\det(\Sigma_{ref})}\right) - N\right) \tag{27}$$

Where N is the dimension of our filter (error) state, which is 22 in our case (see Section II-A). 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 from simulation. 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 high axes ratio. Another advantage of this method is that the KL-divergence can be computed at creation time of its belief nodes rather than in Eq. (26) at every belief comparison, which occur a lot during each RRBT iteration.

As stated in [11], applying Eq. (25) 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 to use a small tolerance factor ϵ to block these useless paths:

$$\begin{aligned} n_a \lesssim n_b \;\Leftrightarrow\; &(n_a.c < n_b.c) \wedge \\ &(n_a.\Sigma + \epsilon I < n_b.\Sigma) \wedge (n_a.\Lambda < n_b.\Lambda + \epsilon I) \end{aligned} \tag{28}$$

E. 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 (Eq. (24)) 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 III-C. The result of this approximate connection defines the full state (Eq. (16)) at v_{new}. This corresponds to the CONNECT($v_{\text{nearest}}.x$, $x_{\text{v,rand}}$) function in [11].

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 II-A on the measurements, and system inputs (c.f. Section II-B) that were generated while creating e_{new}. The initial state $x_{\text{f,init}}$ of the filter is set to:

$$\begin{aligned} x_{\text{f, init}} = [&v_{\text{nearest}}.x.\boldsymbol{p}^T,\; v_{\text{nearest}}.x.\boldsymbol{v}^T, \ldots \\ &\ldots q(v_{\text{nearest}}.\boldsymbol{a},\; v_{\text{nearest}}.\psi), \boldsymbol{b}_\omega^T,\; \boldsymbol{b}_a^T,\; \lambda,\; \boldsymbol{p}_i^s,\; q_i^s]^T \end{aligned} \tag{29}$$

Position and velocity can be directly obtained from $v_{\text{nearest}}.x$ while the attitude quaternion q is a function from the acceleration and yaw angle at v_{nearest}, as explained in Section II-B. 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 [11].

We define the cost for flying along an edge e as $\int \|\boldsymbol{a}\|$, minimizing the energy necessary to reach the goal (c.f. Section III-B). This seems to contradict with the need of excitation of the vehicle for its states to stay observable. However, this is a trade-off between excitation reducing the

[4]This would be states like inter sensor calibration or sensor biases and do not change with the vehicle's dynamic

[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 in [18]. For the sake of simplicity, we neglect this issue in this paper.

uncertainty of the vehicle's states and energy efficiency. That is, the vehicle gets just as much excited as necessary to reach the goal within the defined uncertainty region.

On 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. (21). Whenever an outgoing edge is added to an existing vertex, all its belief nodes are added to a search queue which is then exhaustively searched. Its APPENDBELIEF() function (c.f. [11]) uses the methods discussed in Section III-D to decide if a belief is added to a state vertex (Eq. (28)) and if it dominates and thus can prune existing beliefs (Eq. (25)).

Similarly to the approach in [16], 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. Therefore, as long as there exist no edges to or from the goal state, we explicitly add the goal state to the set of near vertices V_{near} at each iteration of the RRBT algorithm.

We explicitly tried 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 [11], [15], where n is the number of state vertices and d the dimension of the sampled state x_s. ϵ in Eq. (28) 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 (c.f. Section III-D).

IV. RESULTS

In the following, we pick a few representative states of our state estimation filter and show how our proposed method improves its estimates in simulation. The setup is the following: we want to fly the MAV from a starting location along the x-axis to a goal in 10 m distance. This direct path and the resulting path from our method can be seen in Fig. 3.

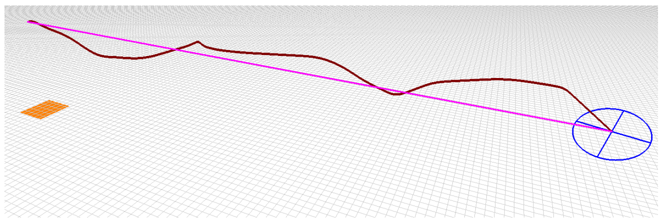

Fig. 3: The direct path and the path computed by our proposed method for a distance of 10 m from the start with the position uncertainty visualized (blue) to the goal region (orange)

We set an initial state covariance that we obtained during real experiments with our framework described in [4]. We simulate the system inputs (acceleration, angular velocity) and measurements of our state estimation framework (Section II-A) with the values computed in Section II-B along the partial paths. Measurements in 3DoF position and 3DoF attitude are simulated, assuming a constant measurement uncertainty here for simplicity. Since the presented approach is a sampling based technique, different measurement uncertainties could be incorporated.

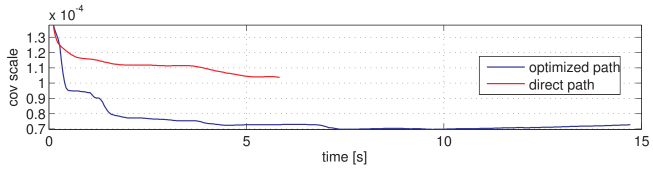

Fig. 4: Comparison of the evolution of uncertainty of the visual scale over time for the direct path (red) and the optimized path (blue).

Fig. 4 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 (c.f. Eq. (6)). As a result, even if the position was measured correct, and without any drift, the position uncertainty grows with increasing distance if the visual scale is uncertain. This is shown in Fig. 5: in the top left, the uncertainty of the position in the x-axis for the optimized path (blue) p_x decreases until $t = 5\,\text{s}$, which corresponds to the improvement of the visual scale uncertainty in Fig. 4 due to excitation of the system. After $t = 5\,\text{s}$, the visual scale has converged and since the system moves away from the origin (Fig. 5,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 behaviour happens notably faster. Since the trajectories for p_y (bottom right) stay close to 0 on the right side of Fig. 5, both the uncertainty (top right) for the direct and the optimized path decrease, while the optimized path is performing remarkably better.

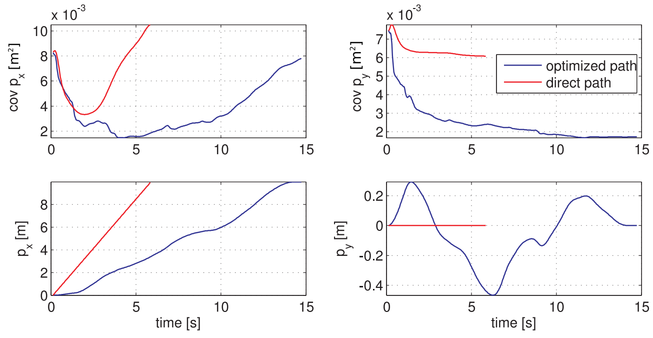

Fig. 5: Behaviour of the uncertainty for the direct path (red) and the optimized path (blue) in position for the x-axis (left) and the y-axis (right). Since the visual scale error is multiplicative, the uncertainty of the position grows after the initialization phase with increasing distance to the origin. The trajectory in the y axis stays close to zero, why uncertainty decreases.

The inter sensor calibration state q_i^s plays another important role. 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.

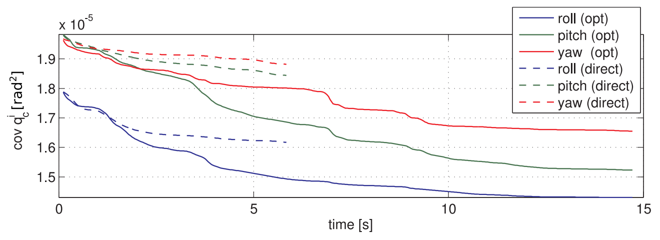

Fig. 6: Comparison of the inter sensor calibration state q_i^s. The optimized path also outperforms the direct path in terms of uncertainty and convergence speed. This state is of major importance since it rotates pose measurements into the IMU reference frame

Fig. 7 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 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 ($\approx 0.6\,\mathrm{m}$). In fact, the direct path would have never reached the goal with its covariance ellipsoid within the uncertainty region around the goal.

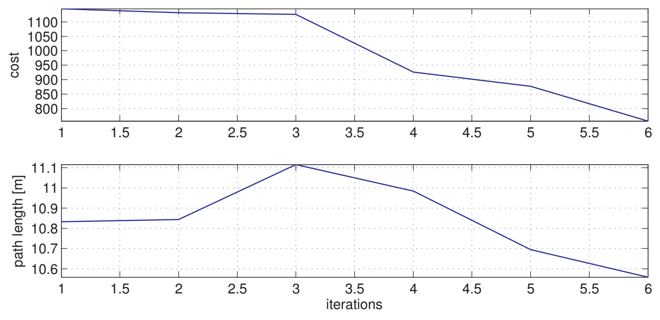

Fig. 7: Cost and path length over the number of optimized path updates. During the simulation, 150 state vertices were sampled, resulting in 750 belief nodes. Finally, a path slightly longer than the direct path yields remarkably better results for our state estimation filter.

Computational complexity with respect to the number of belief nodes is discussed in [11]. The main cost in our approach is due to the propagation of the covariances. Future work will focus on effective model reduction and an efficient search through the belief nodes, reducing complexity for real time operation and deployment on real flights.

V. 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. As such power-on-and-go systems need excitation to render all their states observable, we demonstrated how effective path planning can improve not only the error of the state estimates, but also allow for faster convergence after (re-)initialization. Our approach can cope with non-holonomic constraints of a vehicle, while it plans a collision-free path (avoiding static obstacles) incorporating the uncertainty of the state estimate and the vehicle's dynamics. We showed in simulations that our approach significantly reduces state convergence time, while avoiding navigation in unobservable modes. The cost of which is only a slightly longer path than the most direct path between start and goal.

VI. Acknowledgments

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

References

[1] S. Weiss, "Vision based navigation for micro helicopters," Ph.D. dissertation, ETH Zurich, 2012.

[2] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *International Journal of Robotics Research (IJRR)*, vol. 30, no. 1, pp. 56–79, 2011.

[3] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[4] S. Weiss, M. Achtelik, M. Chli, and R. Siegwart, "Versatile distributed pose estimation and sensor self-calibration for an autonomous mav," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[5] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM based navigation for autonomous micro helicopters in GPS-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011. [Online]. Available: http://dx.doi.org/10.1002/rob.20412

[6] M. W. Achtelik, M. C. Achtelik, S. Weiss, and R. Siegwart, "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)*, 2011.

[7] R. He, S. Prentice, and N. Roy, "Planning in Information Space for a Quadrotor Helicopter in a GPS-denied Environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2008)*, Los Angeles, CA, 2008, pp. 1814–1820.

[8] S. Prentice and N. Roy, "The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance," *International Journal of Robotics Research*, vol. 8, no. 11-12, pp. 1448–1465, December 2009.

[9] M. Bryson, M. Johnson-Roberson, and S. Sukkarieh, "Airborne smoothing and mapping using vision and inertial sensors," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[10] N. Roy and S. Thrun, "Coastal navigation with mobile robots," in *Advances in Neural Information Processing Systems (NIPS '99)*, 1999.

[11] A. Bry and N. Roy, "Rapidly-exploring Random Belief Trees for Motion Planning Under Uncertainty," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[12] R. Hermann and A. Krener, "Nonlinear controllability and observability," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, 1977.

[13] F. Mirzaei and S. Roumeliotis, "A Kalman Filter-Based Algorithm for IMU-Camera Calibration: Observability Analysis and Performance Evaluation," *IEEE Transactions on Robotics and Automation*, vol. 24, no. 5, pp. 1143 –1156, 2008.

[14] D. Mellinger and V. Kumar, " Minimum Snap Trajectory Generation and Control for Quadrotors ," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[15] S. Karaman and E. Frazzoli, "Incremental Sampling-based Algorithms for Optimal Motion Planning," in *Proceedings of Robotics: Science and Systems (RSS)*, Zaragoza, Spain, June 2010.

[16] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Optimal Motion Planning for Systems with Linear Differential Constraints," *CoRR*, vol. abs/1205.5088, 2012.

[17] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[18] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "An observability-constrained sliding-window filter for slam," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, Sept 2011, pp. 65–72.