Minimum-Time Trajectory Planning Under Intermittent Measurements

Bryan Penin[1], Paolo Robuffo Giordano[2], and François Chaumette[1]

***Abstract*—This paper focuses on finding robust paths for a robotic system by taking into account the state uncertainty and the probability of collision. We are interested in dealing with intermittent exteroceptive measurements (e.g., collected from vision). We assume that these cues provide reliable measurements that will update a state estimation algorithm wherever they are available. The planner has to manage two tasks: reaching the goal in a minimum time and collecting sufficient measurements to reach the goal state with a given confidence level. We present a robust perception-aware bi-directional A* planner for *differentially flat* systems such as a unicycle and a quadrotor UAV and use a derivative-free Kalman filter to approximate the belief dynamics in the flat space. We also propose an efficient way of ensuring continuity and feasibility by exploiting the *convex-hull* property of B-spline curves.**

***Index Terms*—Reactive and Sensor-Based Planning, Motion and Path Planning, Aerial Systems: Perception and Autonomy**

I. Introduction

UNCERTAINTY-AWARE planning, also called *belief-space* planning, has received considerable attention in recent years. Indeed, model, sensors and environment uncertainties are inherent to many robotic applications and may lead to a failure of the task or impair the possibility to accurately follow a path if disregarded at the planning stage. A class of control techniques that operate over the belief space, known as partially-observable Markov decision processes (POMDPs) [1] has been derived to address the above problem. Another class of works exploits local optimal control policies assuming a linear quadratic Gaussian (LQG) control strategy. However, these approaches suffer from the "curse of dimensionality". In particular POMDPs are notorious for their computational complexity that may prohibit their application for navigation in complex or uncertain environments in high dimensional state spaces. In [2] a more scalable LQG variant is proposed and applied to environments with discontinuous sensing regions. An approximate solution to POMDPs is given in [3] but with the use of considerable pre-processing. To deal with more complex objectives, deterministic planners such as RRT* and A* are now very popular since they benefit from asymptotic optimality and explore the whole configuration space efficiently. In [4] a graph-search based on A* is proposed by discretizing the environment into cells for finding a safe route for a unicycle vehicle. Active perception with a quadrotor has been addressed in [5] for determining the path with minimal state uncertainty, and in [6] for maximizing coverage in presence of obstacles, localization and sensing uncertainty. Recently, [7] proposed an approximate POMDP control policy based on an initial guess trajectory returned by a RRT planner in a discretized environment.

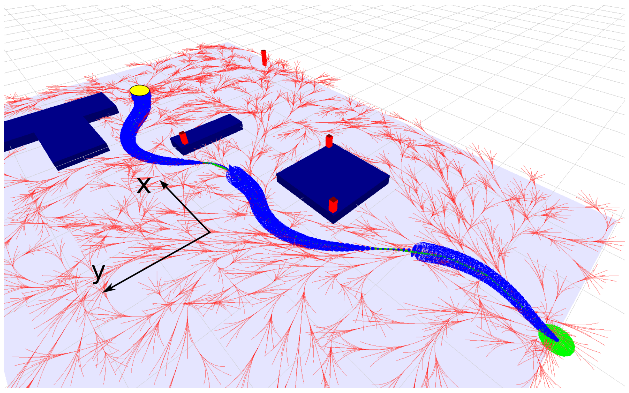


Fig. 1: Simulation environment for our framework. An optimal collision-free trajectory considering a unicycle connects the initial state (green dot) and a final state (yellow dot) in presence of obstacles (blue boxes). The pose uncertainty is represented by the blue ellipsoids whose size is reduced as soon as a landmark (red bars) is close enough to the robot and enters the field of view of the camera attached to the robot. We assume the landmarks are not occluded by the obstacles. The propagated edges of the two graphs are rendered as the red curves.

In this work, we aim at planning a trajectory from an initial to a final state in presence of obstacles and input constraints for non-trivial robotic systems (like a quadrotor). We assume that the state is not available (especially the position) but on-board sensors (including a camera) are used to reconstruct the state with some estimation algorithm fusing position measurements reconstructed from vision. Note that these measurements may be temporarily unavailable because of limited field of view, maximum range and so on. We want that the path guarantees some desired level of uncertainty in the reconstructed state despite the possible non-availability of the measurements. More precisely, the goal state has to be reached with a bounded state uncertainty to guarantee some confidence level

Manuscript received: September, 10, 2018; Revised November, 21, 2018; Accepted November, 8, 2018.

This work was supported by the French Direction Générale de l'Armement (DGA).

[1]B. Penin and F. Chaumette are with Inria, Univ Rennes, CNRS, IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France bryan.penin@inria.fr, francois.chaumette@inria.fr

[2]P. Robuffo Giordano is with CNRS, Univ Rennes, Inria, IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France prg@irisa.fr

Digital Object Identifier (DOI): see top of this page.

on the robot location. Therefore, the system has to collect sufficient information from visual landmarks sparsely placed in the environment to satisfy this final constraint (see Fig. 1 considering a unicycle equipped with a front-looking camera with reference to Fig. 2).

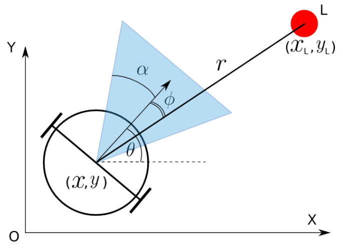


Fig. 2: A unicycle equipped with a fixed camera receives position measurements updates whenever a visual landmark (red dot) is close enough and enters the field of view.

Literature in perception-aware planning has generally focused on maximizing observability (or minimizing the state uncertainty) based on some criteria e.g., the trace or the smallest eigenvalue of the covariance matrix. This strategy definitely helps in finding a path that tries to collect as much information as possible for preventing the state uncertainty to increase too much. However, the path itself may be severely suboptimal in terms of length and duration (e.g., [8] for optimal self-calibration of UAVs). Indeed, the path length is generally not constrained and can be excessively long, especially if the robot needs to pass by all the regions/landmarks with richest information.

In this work, we propose a minimum-time planning algorithm for dynamic systems returning feasible and robust trajectories that do not guarantee minimal state uncertainty along the trajectory but a bounded state uncertainty with given bounds at the goal, which we consider as a more practical application.

II. Related Works

A. Planning in the belief space of dynamical systems

An admissible position uncertainty is defined in [9] as a goal area of a given size that a nonholonomic robot has to enter. A RRT variant is implemented where uncertain states are modeled as boxes. Our work is mostly based on the recent work of [10], [11] that propose an efficient A* planner in the flat space of a quadrotor. It is applied to aggressive manoeuvres and precise collision avoidance that are function of the robot attitude. A variant of RRT for flat systems is detailed in [12] to produce smooth dynamically feasible motion plans in real-time and in [13] for online navigation in dynamic environments with a quadrotor. Both works exploit differential flatness to build a look-up table of pre-computed feasible motion primitives.

B. Contributions

In a previous work [14] we considered hard visibility constraints that may become too restrictive for minimum-time planning. In this work, we propose to relax these constraints by allowing temporary visual cues losses for performing faster trajectories in larger and more complex environments. We implement a bi-directional A* algorithm that grows two graphs, one from the initial state and one from the final state. A solution trajectory is built by connecting the two graphs. The work presented in this paper blends the following features within a graph-search algorithm: (i) incorporation of model and sensor uncertainty in collision avoidance and perception, (ii) generation of minimum-time and feasible trajectories for flat dynamic systems, (iii) incorporation of discontinuous visual measurements, (iv) efficient graph connection using the convex-hull property of B-spline curves.

In contrast to [11] we include perception constraints and state uncertainty and directly minimize the time. To the best of our knowledge, this is the first time minimum-time trajectories are generated in a graph-search planner while accounting for uncertainty in the visual perception which is affected by the system state.

The rest of this paper is organized as follows. Sect. III introduces differential flatness and the modelling of a quadrotor. Sect. IV presents the uncertainty-aware planner formulated as a graph-search problem. How the graph is built is described in Sect. V. The graphs rewiring is detailed in Sect. VI. In Sect. VII simulation and experimental results are presented for a quadrotor with an onboard camera. Finally we draw some conclusions and future directions in Sect. VIII.

III. Preliminaries

A. Differential flatness

Differentially flat systems are systems whose state $\boldsymbol{\chi}$ and inputs $\boldsymbol{u}$ can be expressed as algebraic functions of *flat outputs* derivatives up to some suitable order [15]. Differential flatness is often used for planning purposes: i) the problem size is reduced, ii) any smooth enough curve in the flat space is feasible by the real system, iii) the system dynamics are linear in the flat space. The proposed algorithm is applicable to systems represented as d independent chains of integrators of a given order r of the form

$$\boldsymbol{\eta}^{(r)} = \boldsymbol{\nu} \tag{1}$$

where $\boldsymbol{\nu} \in \mathbb{R}^d$ denotes the new inputs and $\boldsymbol{\eta} \in \mathbb{R}^d$ are the flat outputs. Let us define the system state in the flat space as

$$\boldsymbol{s} = \left(\boldsymbol{\eta}, ..., \boldsymbol{\eta}^{(r-1)}\right) \in \mathbb{R}^{d(r-1)} \tag{2}$$

In the rest of the paper we consider the system position as the flat outputs, e.g., $\boldsymbol{\eta} = (x, y, z)$ in the three-dimensional space.

B. Application to a quadrotor UAV

With reference to Fig. 3 the quadrotor state is defined as $\boldsymbol{\chi} = (\boldsymbol{p}, \boldsymbol{R}, \boldsymbol{v}, \boldsymbol{\omega}) \in SE(3) \times \mathbb{R}^6$ where $\boldsymbol{p} \in \mathbb{R}^3$ is the position of the robot COM in the world frame $\mathcal{W} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$ (being $\boldsymbol{e}_i$ the i-th column of the identity matrix), $\boldsymbol{R} \in SO(3)$ is the rotation matrix from $\mathcal{W}$ to the body frame $\mathcal{B} = \{\boldsymbol{x}_{\mathcal{B}}, \boldsymbol{y}_{\mathcal{B}}, \boldsymbol{z}_{\mathcal{B}}\}$, $\boldsymbol{v}$ the COM linear velocity expressed in $\mathcal{W}$ and $\boldsymbol{\omega}$ the angular velocity expressed in $\mathcal{B}$. The quadrotor is known to be flat with

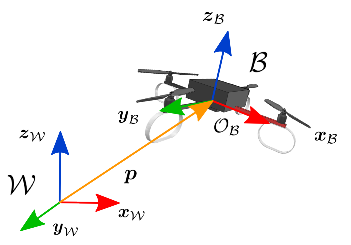


Fig. 3: Quadrotor model

flat outputs (x, y, z, ψ) [16]. The quadrotor dynamics can be decoupled into four linear subsystems of the form

$$x^{(4)} = u_1,\ y^{(4)} = u_2,\ z^{(4)} = u_3,\ \psi^{(2)} = u_4 \tag{3}$$

where $\boldsymbol{\nu} = (u_1, u_2, u_3, u_4)$ defines the new control inputs in the flat space. For the sake of simplicity, we do not plan over the yaw angle ψ that is assumed to be constant at zero. Moreover, we consider the quadrotor as three triple-integrators controlled in jerk along axes X, Y and Z (i.e., $\boldsymbol{\eta} = (x, y, z) \in \mathbb{R}^3$). With the above simplifications we seek to alleviate the planner whose complexity grows exponentially with the state dimension. Finally, we consider the state vector $\boldsymbol{s} = (\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}, \ddot{\boldsymbol{\eta}})$ and inputs $\boldsymbol{\nu} = \boldsymbol{\eta}^{(3)}$. We consider that the quadrotor is equipped with a fixed downward-looking camera capable of providing reliable position measurements when fixed landmarks on the ground enter the limited field of view.

IV. Problem formulation

We aim at solving an optimal control problem connecting an initial state $\boldsymbol{s}_{init}$ and a final state $\boldsymbol{s}_{goal}$ in a minimum time T. Let us define the following optimal control problem.

Problem 1. *Find the input $\boldsymbol{\nu}$ and time T such that:*

$$\min_{\boldsymbol{\nu},T} \quad T \tag{4a}$$

$$\text{s.t.} \quad \boldsymbol{s}(0) = \boldsymbol{s}_{init}, \quad \boldsymbol{s}(T) = \boldsymbol{s}_{goal}, \tag{4b}$$

$$\max\{\text{eig}(\boldsymbol{P}^{\boldsymbol{\eta}}(T))\} \leqslant \bar{\lambda}, \tag{4c}$$

$$(\dot{\boldsymbol{\eta}}(\tau), \ddot{\boldsymbol{\eta}}(\tau), \boldsymbol{\eta}^{(3)}(\tau)) \in \mathcal{X}^{free} \quad \forall \tau \in [0, T] \tag{4d}$$

where $\text{eig}(\boldsymbol{P}^{\boldsymbol{\eta}}(T)) \in \mathbb{R}^d$ contains the eigenvalues of the position covariance matrix $\boldsymbol{P}^{\boldsymbol{\eta}}$ at the goal state $\boldsymbol{s}_{goal}$ and $\mathcal{X}^{free} := [-\bar{v}, \bar{v}]^3 \times [-\bar{a}, \bar{a}]^3 \times [-\bar{j}, \bar{j}]^3$ denotes the hypercube space of the admissible velocities, accelerations and jerks. The desired bound on the position uncertainty is defined by $\bar{\lambda} > 0$. For nonlinear systems such as a quadrotor, the Extended Kalman Filter (EKF) is often used for approximating the belief dynamics. The EKF is based on a linearization of the system dynamics which results in cumulative errors due to the local linearization assumption. In this paper, since we plan directly in the flat space we can use a so-called *derivative-free* Kalman filter without the need for derivatives and Jacobians calculations. Moreover, the state estimation accuracy of a derivative-free Kalman filter can be improved w.r.t. a standard EKF, especially for nonlinear systems [17]. Considering the linear equivalent system in the flat space one defines the process model. When a landmark is visible we have

$$\dot{\boldsymbol{s}} = A\boldsymbol{s} + B\boldsymbol{u} + \boldsymbol{\zeta}, \quad \boldsymbol{y} = C\boldsymbol{s} + \boldsymbol{\epsilon} \tag{5}$$

where $\boldsymbol{\zeta}$ is the process noise and $\boldsymbol{\epsilon}$ is the measurement noise. For a flat system controlled in acceleration (i.e., $r = 2$), assuming the velocity is estimated through filtering of position measurements, the matrices A, B, and C are given by

$$A = \begin{pmatrix} 0&0&1&0\\ 0&0&0&1\\ 0&0&0&0\\ 0&0&0&0 \end{pmatrix}, B = \begin{pmatrix} 0&0\\ 0&0\\ 1&0\\ 0&1 \end{pmatrix}, C = \begin{pmatrix} 1&0&0&0\\ 0&1&0&0 \end{pmatrix} \tag{6}$$

In the next sections we will show how Problem 1 can be transformed from an infinite dimensional optimal control problem to a finite dimensional bi-directional graph-search problem. We choose to extend two graphs to improve the search, to increase the rate of convergence and the chance to find a solution especially in complex and cluttered environments. Moreover, the use of two graphs generally propagates fewer vertices than with a single graph.

A. Motion primitives

As in [10] we use polynomials to parameterize the flat state components and generate motion primitives to explore the flat space in a discrete way (see (1)). More precisely, by applying a number of sampled constant inputs $\boldsymbol{\nu}_k \in [-\bar{j}, ..., \bar{j}]^3$ along each axis for a duration $\tau > 0$ one can iteratively build a graph $\mathcal{G}(V, E)$ rooted in state $\boldsymbol{s}_0$. Here, V defines the set of discrete states denoted as the *vertices* $\boldsymbol{s}$ in the graph representation that are connected with a motion primitive referred as an *edge* in the set E (e.g., see Fig. 1). For a flat system controlled in jerk (i.e,. $r = 3$) a motion primitive represents the state $\boldsymbol{s}(t)$ starting at state $\boldsymbol{s}_0$ for $t \in [0, \tau]$ with a curve defined as

$$\boldsymbol{s}(t) = M(\boldsymbol{\nu}_k, \boldsymbol{s}_0, t) := \begin{bmatrix} \boldsymbol{\nu}_k \frac{t^3}{6} + \ddot{\boldsymbol{\eta}}_0 \frac{t^2}{2} + \dot{\boldsymbol{\eta}}_0 t + \boldsymbol{\eta}_0 \\ \boldsymbol{\nu}_k \frac{t^2}{2} + \ddot{\boldsymbol{\eta}}_0 t + \dot{\boldsymbol{\eta}}_0 \\ \boldsymbol{\nu}_k t + \ddot{\boldsymbol{\eta}}_0 \end{bmatrix} \tag{7}$$

These trajectories reflect the system dynamics thanks to differential flatness and provide the minimum jerk between the states $\boldsymbol{s}_0$ and $\boldsymbol{s}(\tau)$ [18]. The free flat space will be explored with a propagation of these motion primitives further detailed in Sect. V. Naturally, changing the admissible bounds for $\boldsymbol{\nu}_k$ and duration τ will affect the free space coverage.

Problem 1 can be reformulated as Problem 2 in the graph representation where we seek the trajectory connecting the initial and goal states with the optimal control sequence $\boldsymbol{\nu}_k^*$ and the minimal number N^* of motion primitives.

Problem 2. *Find the sequence $\boldsymbol{\nu}_k$ and N such that:*

$$\min_{\boldsymbol{\nu}_k,N} \quad N \tag{8a}$$

$$\text{s.t.} \quad \boldsymbol{s}_0 = \boldsymbol{s}_{init}, \quad \boldsymbol{s}_N = \boldsymbol{s}_{goal}, \tag{8b}$$

$$\max\{\text{eig}(\boldsymbol{P}_N^{\boldsymbol{\eta}})\} \leqslant \bar{\lambda}, \tag{8c}$$

$$(\dot{\boldsymbol{\eta}}_k, \ddot{\boldsymbol{\eta}}_k, \boldsymbol{\eta}_k^{(3)}) \in \mathcal{X}^{free} \quad \forall k \in [\![0, N]\!] \tag{8d}$$

where $\boldsymbol{P}_N^{\boldsymbol{\eta}}$ is the covariance matrix on the position at the goal vertex. The resulting trajectory will have a total time $N^*\tau$. Finally, collisions are avoided by considering the robot shape as representative of the position uncertainty ellipse (or ellipsoid in 3D) whose estimation is detailed in the next section. Motion primitives that violate the collision constraints are not added to the graph.

In the next section we will show how state uncertainty is included in visual perception and in collision avoidance to guarantee perception of visual measurements and safe navigation to a given level of confidence.

B. State estimation uncertainty

Let σ be the major axis of the uncertainty ellipse $\boldsymbol{P}^{\boldsymbol{\eta}}$ at a given state. Then, for a 99% confidence level one has $\sigma_{99\%} = \sqrt{9.21}\sqrt{\lambda}$ (from the Chi-Square probabilities) where λ is the largest eigenvalue of $\boldsymbol{P}^{\boldsymbol{\eta}}$. This confidence ellipse defines the region that contains 99% of all samples that can be drawn from the Gaussian distribution. We take a sphere with radius $\sigma_{99\%}$ as representative of the robot occupancy. It will vary with the pose uncertainty and will be included in the planner for ensuring robust collision-free paths. Now, let us consider state uncertainty in the visual perception. With reference to Fig. 4, in order to check that a landmark at position $\boldsymbol{r}_L$ is visible along an edge, we impose conditions on angles β_1 and β_2 on both planes X-Z and Y-Z. On a given plane one has

$$\beta_i = \arccos\left(\frac{(\boldsymbol{p}-\boldsymbol{r}_L).\boldsymbol{l}_i}{\|\boldsymbol{p}-\boldsymbol{r}_L\|}\right), \quad i = 1,2 \tag{9}$$

where $\boldsymbol{l}_i$ are given by

$$\boldsymbol{l}_1 = \begin{pmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{pmatrix}\boldsymbol{t},\, \boldsymbol{l}_2 = \begin{pmatrix}\cos\alpha & \sin\alpha\\ -\sin\alpha & \cos\alpha\end{pmatrix}\boldsymbol{t} \tag{10}$$

With $\boldsymbol{g}$ as the (constant) gravity acceleration and $\boldsymbol{a}$ as the robot acceleration in the world frame one has $\boldsymbol{t} = (\boldsymbol{a}+\boldsymbol{g})/\|\boldsymbol{a}+\boldsymbol{g}\|$. Since the β_i are function of the position and the acceleration, the bearing uncertainty $\Delta\beta \in \mathbb{R}$ on a given plane can be computed as

$$\Delta\beta = \begin{pmatrix}\frac{\partial\beta}{\partial\boldsymbol{\eta}} & \frac{\partial\beta}{\partial\ddot{\boldsymbol{\eta}}}\end{pmatrix}^T \begin{pmatrix}\boldsymbol{P}^{\boldsymbol{\eta}} & 0\\ 0 & \boldsymbol{P}^{\ddot{\boldsymbol{\eta}}}\end{pmatrix} \begin{pmatrix}\frac{\partial\beta}{\partial\boldsymbol{\eta}} & \frac{\partial\beta}{\partial\ddot{\boldsymbol{\eta}}}\end{pmatrix} \tag{11}$$

where $\boldsymbol{P}^{\ddot{\boldsymbol{\eta}}}$ is the acceleration covariance matrix. The state covariance matrices are evaluated with the Kalman filter along each valid discretized motion primitive. This way one can ensure a (theoretical) 99% confidence on the perception if the following upper bound conditions are satisfied on both planes X-Z and Y-Z at a given state

$$|\beta_i| + \sqrt{9.21}\Delta\beta < 2\alpha,\ i = 1,2 \tag{12}$$

These conditions allow an exact and fast evaluation of the visibility and only rely on the flat state. Moreover, it allows us to consider a realistic pyramid-shaped field of view (since we do not plan over the yaw). The update step of the Kalman filter is therefore applied with the simulated measurements whenever conditions (12) are met along the propagated motion primitives.

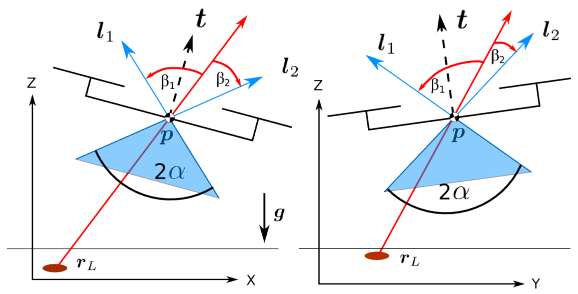


Fig. 4: The quadrotor in the vertical planes X-Z and Y-Z with $\psi = 0$. Having uncertainties on the state affects the perception. To evaluate if a landmark (red blob) is visible under the state uncertainty we check that condition (12) is satisfied on both planes.

V. BUILDING THE GRAPH

In this section we show how to exploit some vertices to efficiently explore the free space with the design of an heuristic function in order to build the graph. Traditionally, distance-based heuristics are used but they are not very relevant for dynamic or nonholonomic systems that cannot change their velocity, acceleration or orientation instantaneously. That is why a heuristic function more appropriate for second- or higher-order systems has been proposed in [10], by taking smoothness into account. As well know, A* algorithms rely on two functions: the heuristic function $h(\boldsymbol{s}, \boldsymbol{s}')$ that encodes an (optimistic) approximation of the *cost-to-go* from a vertex $\boldsymbol{s}$ to a goal vertex $\boldsymbol{s}'$ and the function $g(\boldsymbol{s})$ which represents the cost of vertex $\boldsymbol{s}$. Without a heuristic, A* is equivalent to a Dijkstra search, but encoding some theoretic information into the heuristic function can greatly reduce the number of expansions in favouring exploration toward promising directions/areas. We use the heuristic function proposed in [10] that originates from the resolution of Pontryagin's minimum principle and invite the reader to refer to the latter paper for more information. This function now encodes the "effort" required to connect two states given the considered control input (e.g., velocity, acceleration or jerk) and is used to select vertices leading to the exploration toward regions with minimal energy in order to encourage smooth trajectories. The cost of an edge itself is $g(\boldsymbol{s}) = \tau$ because we want to minimize the time. We propose a bi-directional A* algorithm that builds two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$. $\mathcal{G}_1$ starts at the initial state $\boldsymbol{s}_0$ and $\mathcal{G}_2$ starts at the final state $\boldsymbol{s}_N$. Both graphs will be propagated and connected to return full trajectories from $\boldsymbol{s}_{init}$ to $\boldsymbol{s}_{goal}$. The planner is detailed in the following section.

Designing an efficient space exploration is tedious when complex tasks are involved and should not rely on a too strong *a priori*. Namely, in our case, the search should not be biased towards the goal since it may not collect sufficient visual cues from the landmarks to satisfy (8c). When multiple goals are present one may bias the search towards these goals, here, the landmarks. However, ensuring convergence to the final goal is not straightforward, especially for dynamic systems with perception goals. In the end, we choose to not rely on any exploration a priori to be able to deal with any environment and landmark configurations (provided a solution exists). We will rely instead on random-based exploration by smoothly propagating vertices toward states sampled randomly in the free space. Algorithm 1 runs for a given number of iterations I and is detailed below. Note that the uncertainty is only propagated on graph $\mathcal{G}_1$ with the Kalman filter since graph $\mathcal{G}_2$ is grown *backwards* (i.e., from the final goal $\boldsymbol{s}_{goal}$ towards the initial state $\boldsymbol{s}_{init}$). With reference to Fig. 5, the algorithm

Algorithm 1 Bi-A*

```
1:  $g(\boldsymbol{s}^1_{best}) \leftarrow \infty,\ g(\boldsymbol{s}^2_{best}) \leftarrow \infty$
2:  for $i \leftarrow 1$ to $I$ do
3:      $\boldsymbol{s}_{rand} \leftarrow Sample()$
4:      $(X^1_{near}, X^2_{near}) \leftarrow NearVertices(\boldsymbol{s}_{rand}, \mathcal{G}_1, \mathcal{G}_2, \rho)$
5:      $(X^1_c, X^2_c) \leftarrow NearVertices(\boldsymbol{s}_{rand}, \mathcal{G}_1, \mathcal{G}_2, \gamma_c)$
6:      if $X^1_c \neq \emptyset$ and $X^2_c \neq \emptyset$ then
7:          $(\boldsymbol{s}^1_{new}, \boldsymbol{s}^2_{new}) \leftarrow ConnectG(X^1_c, X^2_c)$
8:          $(\boldsymbol{P}^{\boldsymbol{\eta}}_N, coll) \leftarrow BackProp(\boldsymbol{s}^2_{new}, P_{\boldsymbol{s}^1_{new}}, \mathcal{G}_2)$
9:          if $\max\{\mathrm{eig}(\boldsymbol{P}^{\boldsymbol{\eta}}_N)\} \leqslant \bar{\lambda}$ and $!coll$
10:             if $g(\boldsymbol{s}^1_{new}) + g(\boldsymbol{s}^2_{new}) < g(\boldsymbol{s}^1_{best}) + g(\boldsymbol{s}^2_{best})$
11:                 $\boldsymbol{s}^1_{best} = \boldsymbol{s}^1_{new},\ \boldsymbol{s}^2_{best} = \boldsymbol{s}^2_{new}$
12:     if $X^1_{near} \neq \emptyset$ then
13:         $L_s \leftarrow GetSortedList(X^1_{near})$
14:         $\boldsymbol{s}^*_1 \leftarrow ChooseBestParent(L_s, \boldsymbol{s}_N)$
15:         $L^c_1 \leftarrow ExtendVertex(\boldsymbol{s}^*_1)$
16:         $G_1 \leftarrow InsertVertices(L^c_1)$
17:     if $X^2_{near} \neq \emptyset$ then
18:         $L_s \leftarrow GetSortedList(X^2_{near})$
19:         $\boldsymbol{s}^*_2 \leftarrow ChooseBestParent(L_s, \boldsymbol{s}_0)$
20:         $L^c_2 \leftarrow ExtendVertex(\boldsymbol{s}^*_2)$
21:         $G_2 \leftarrow InsertVertices(L^c_2)$
22:     if $X^1_{near} = \emptyset$ and $X^2_{near} = \emptyset$ then
23:         $(\boldsymbol{s}^*_1, \boldsymbol{s}^*_2) \leftarrow BestVertices(\mathcal{G}_1, \mathcal{G}_2)$
24:         $L^c_1 \leftarrow ExtendVertex(\boldsymbol{s}^*_1)$
25:         $G_1 \leftarrow InsertVertices(L^c_1)$
26:         $L^c_2 \leftarrow ExtendVertex(\boldsymbol{s}^*_2)$
27:         $G_2 \leftarrow InsertVertices(L^c_2)$
    return $\boldsymbol{s}^1_{best}, \boldsymbol{s}^2_{best}$
```

procedures are detailed below:

Sample: returns an independent and uniformly distributed random sample vertex $\boldsymbol{s}_{rand}$ in the free space.

NearVertices: given a sample vertex $\boldsymbol{s}_{rand}$, a graph $\mathcal{G} = (V, E)$ and a ball region $\mathfrak{B}_r$ of a given radius ρ, the set of near vertices is defined as $Near(\boldsymbol{s}, \mathcal{G}, \rho) = \{\boldsymbol{s} \in V : \mathrm{d}(\boldsymbol{s}, \boldsymbol{s}_{rand}) \leqslant \rho\}$ where d is the Euclidean distance and $\rho = \gamma(\log(K)/K)^{1/q}$ is the radius for expansion with K is the number of vertices and q is the space dimension. The ball radius helps capturing vertices when the graph is hollow and shrinks with the number of vertices to reduce the computation time. We use a constant radius γ_c for finding connections candidates (see procedure *ConnectG*).

GetSortedList: given a list of vertices V and a goal $\boldsymbol{s}'$, this function returns a list L_s of the sorted vertices $\boldsymbol{s} \in V$ in increasing heuristic cost $h(\boldsymbol{s}, \boldsymbol{s}')$.

ChooseBestParent: the vertex with lowest h cost from a list of vertices is chosen for expansion. We seek to find the parent vertex that will expand vertices towards the given goal with the lowest energy (highest smoothness).

BestVertices: when no near vertices are found in $\mathfrak{B}_r$, this function finds the vertex $\boldsymbol{s}$ in graph $\mathcal{G}_1$ with lowest cost $h(\boldsymbol{s}, \boldsymbol{s}_N)$ and analogously for $\mathcal{G}_2$ with $h(\boldsymbol{s}, \boldsymbol{s}_0)$.

ExtendVertex: propagates a motion primitive from a given parent vertex. This function includes the belief state propagation with the Kalman filter and collision and feasibility tests.

InsertVertices: valid vertices/edges are added to the graph and marked as *children* from their parent vertex.

InsertVertex: this function inserts a single vertex/edge pair.

ConnectG: this procedure is triggered whenever vertices from both graphs are found in the procedure *NearVertices* within a second ball region of constant radius γ_c centered on $\boldsymbol{s}_{rand}$. Indeed, we seek pairs of vertices in a vicinity region to perform connection tests (see Algorithm 2) using the function *solveQP* presented in Sect. VI. Note that γ_c can for instance, be chosen as the "spatial resolution" of motion primitives or larger to find more connections.

BackProp: given a vertex $\boldsymbol{s}_0$ with a covariance matrix $\boldsymbol{P}_0$ from graph $\mathcal{G}_1$, once a connection is found we *back-propagate* the state uncertainty through $\mathcal{G}_2$ by considering the sampled states between $\boldsymbol{s}_0$ and the goal $\boldsymbol{s}_N$ (see Fig. 5).

Algorithm 1 aims at finding the most direct trajectory towards the goal, especially in case of low process noise and tries to mimic a couple of nice properties of classic graph-search planners, namely: i) expansion towards unexplored regions; ii) probabilistic *completeness* due to a uniform *random walk*; iii) asymptotic optimality.

Algorithm 2 ConnectG

```
Input: $X^1_c, X^2_c$
1: $success$ = false
2: for $\boldsymbol{s}_1$ in $X^1_c$ do
3:     for $\boldsymbol{s}_2$ in $X^2_c$ do
4:         if $h(\boldsymbol{s}_1, \boldsymbol{s}_2) < \bar{h}$ then
5:             $success \leftarrow solveQP(\boldsymbol{s}_1, \boldsymbol{s}_2)$
6:             if $success$ = true then
7:                 return $(\boldsymbol{s}_1, \boldsymbol{s}_2)$
8: return 0
```

Algorithm 2 performs connection trials on the vertices in X^1_c, X^2_c if their heuristic cost is lower than a given value $\bar{h}$. This value can be chosen off-line to skip connections that may require "too much" energy. Usually, graph-search planners for dynamic systems involve two steps. First, an optimal path is found ignoring the system dynamics. Then a refining step is performed by optimizing over a selection of state keyframes along the path. The resulting trajectory is smoothed and more adapted to dynamic systems (see e.g., [10], [19]). However, the shape of this trajectory may strongly differ from the original path (e.g., in [19]). In our context, visual perception cannot be guaranteed with such a technique and it does not take into account the uncertainty in collision avoidance. A key role of the bi-directional planner used in this work is that if a connection is made, the initial and final states are exactly connected, which is generally not the case in the graph-search planners literature. For instance, [10] stops the search when a vertex becomes close enough to the goal state $\boldsymbol{s}_{goal}$, a condition that may not be met if not properly tuned. In this work we aim instead at finding the optimal trajectory that will be directly tracked by the real system without additional refining steps.

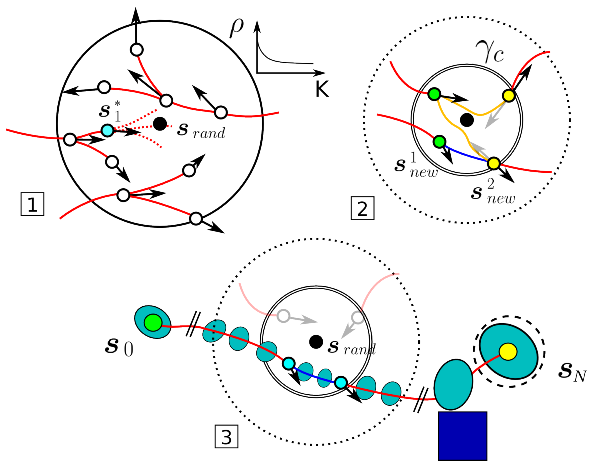


Fig. 5: 2D case: Inside a ball of radius ρ centered at $\boldsymbol{s}_{rand}$ (black dot), picture 1 shows how the vertex with lowest cost h is chosen for expansion (in cyan). The black arrows represent the vertex velocity vector. If vertices from graph $\mathcal{G}_1$ (in green) and from graph $\mathcal{G}_2$ (in yellow) are found inside a ball region of fixed radius γ_c, connections trials are performed except for connections with a high h cost (orange lines). Note that we consider the opposite velocity (and higher derivatives) vectors for vertices coming from graph $\mathcal{G}_2$. If a candidate connection is found (blue line) the uncertainty is propagated along $\mathcal{G}_2$ starting from $\boldsymbol{s}^1_{new}$ (picture 3). If no collisions are found between the obstacles (blue box) and the uncertain robot occupancy (turquoise ellipses) and if the final constraint (8c) is satisfied on $\boldsymbol{P}^{\boldsymbol{\eta}}_N$, a solution trajectory is reconstructed from the initial vertex $\boldsymbol{s}_0$ (green dot) to the goal vertex $\boldsymbol{s}_N$ (yellow dot) and its total cost is evaluated.

Next section details how the connection between the two graphs is performed in an optimal and efficient way.

VI. Connecting the Graphs

Connecting the two graphs is a critical step. One has to ensure state continuity between two candidate vertices $\boldsymbol{s}_1 \in V_1$ and $\boldsymbol{s}_2 \in V_2$ in a given time T_c. This problem is known as the Boundary Value Problem (BVP). Moreover, one wants to ensure feasibility as well and connections have to be evaluated quickly since the process may be called many times. We propose an optimal formulation to the BVP that can be solved as a single convex quadratic program. We exploit the *convex-hull* property of B-splines in order to impose constraints directly on the curve control points to alleviate the solver.

A. Solving the constrained BVP

The problem we want to solve is the following

Problem 3. Find $\boldsymbol{s}, \boldsymbol{\nu}$ such that:

$$\min_{\boldsymbol{s},\boldsymbol{\nu}} \int_0^{T_c} \|\boldsymbol{\nu}(\tau)\|^2 d\tau \tag{13a}$$

$$\text{s.t.} \quad \boldsymbol{s}(0) = \boldsymbol{s}_1, \quad \boldsymbol{s}(T_c) = \boldsymbol{s}_2, \tag{13b}$$

$$(\dot{\boldsymbol{\eta}}(\tau), ..., \boldsymbol{\eta}^{(r)}(\tau)) \in \mathcal{X}^{free} \quad \forall \tau \in [0, T_c] \tag{13c}$$

We penalize the input norm to obtain a smooth connection trajectory. For the quadrotor one minimizes the jerk norm (i.e., $r = 3$). Now, we parameterize the flat state $\boldsymbol{s}$ as B-splines to turn the infinite dimensional problem to a finite one with a limited number of coefficients that can be solved numerically. A trajectory s is parameterized as

$$s(t) = \sum_{i=0}^{i=n} B_{i,p}(\tau)\boldsymbol{P}, \quad \forall \tau \in [0, T] \tag{14}$$

where B_i is a polynomial basis of degree p (of order $k = p+1$) and $\boldsymbol{P} \in \mathbb{R}^{n+1}$ represents the set of coefficients.

B. A linear quadratic program based on B-splines

The reason we use B-splines is for their *convex-hull* property that will allow us to write linear inequality constraints directly on the B-spline control points. A similar approach has been used in [20] for manipulators. This strategy avoids discretizing the flat outputs and the constraints that may lead to a great number of constraints. In short, since on its knot vector the basis functions are non-zero and sum to 1, a point on the curve will lie in the convex-hull of the control points. By constraining the control points, we are ensured that the B-spline curve will satisfy the same constraints. However, the distance between the control points and the actual curve is positive but we will show that the conservatism introduced is very reasonable with the considered basis functions. Constraints (13c) can be mapped in the space of the control points. Let us differentiate the B-spline of degree p defined on the *clamped* knot vector of size $n + k + 1$ such that $u_i \leqslant u_{i+1}, i = 0, \ldots, n-k$

$$U = (\underbrace{0, \ldots, 0}_{p+1}, u_{p+1}, \ldots, u_{k-p-1}, \underbrace{1, \ldots, 1}_{p+1}) \tag{15}$$

The first derivative can be expressed as a function of the control points $\boldsymbol{P}$ with

$$s^{'}(u) = p \sum_{i=0}^{i=n-1} B_{i+1,p-1}(t) \frac{\boldsymbol{P}_{i+1} - \boldsymbol{P}_i}{T(u_{i+p+1} - u_{i+1})} \tag{16}$$

Let us define the vector of new coefficients

$$\boldsymbol{Q}_i = p\frac{\boldsymbol{P}_{i+1} - \boldsymbol{P}_i}{T(u_{i+p+1} - u_{i+1})}, \quad \forall i \in [\![0, n-1]\!] \tag{17}$$

For the second derivative one has

$$s^{''}(u) = \sum_{i=0}^{i=n-2} B_{i+2,p-2}(t)\boldsymbol{R}_i \tag{18}$$

where $\boldsymbol{R}_i$ are the control points of the second derivative. One has

$$\boldsymbol{R}_i = (p-1)\frac{\boldsymbol{Q}_{i+1} - \boldsymbol{Q}_i}{T(u_{i+p+1} - u_{i+2})}, \quad \forall i \in [\![0, n-2]\!] \tag{19}$$

Now we can express $\boldsymbol{Q}$ and $\boldsymbol{R}$ as functions of $\boldsymbol{P}$ with

$$\boldsymbol{Q} = \boldsymbol{A_Q}\boldsymbol{P}, \quad \boldsymbol{R} = \boldsymbol{A_R}\boldsymbol{Q} = \boldsymbol{A_R}\boldsymbol{A_Q}\boldsymbol{P} \tag{20}$$

where matrix $\boldsymbol{A_Q} \in \mathbb{R}^{n\times(n+1)}$ and $\boldsymbol{A_R} \in \mathbb{R}^{(n-1)\times n}$. Similarly, control points of the third derivative are given by

$$\boldsymbol{S}_i = (p-2)\frac{\boldsymbol{R}_{i+1} - \boldsymbol{R}_i}{T(u_{i+p+1} - u_{i+3})}, \quad \forall i \in [\![0, n-3]\!] \tag{21}$$

Now, one can easily set semi-infinite bounds on the derivatives coefficients $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{S}$ that are linear in $\boldsymbol{P}$. All the constraints can be rewritten as functions of the control points $\boldsymbol{P}$. One wants to solve the following problem on each axis.

Problem 4. Find $\boldsymbol{P}$ such that:

$$\min_{\boldsymbol{P}} \boldsymbol{P}^T(\boldsymbol{B}_r^T\boldsymbol{B}_r)\boldsymbol{P} \tag{22a}$$

$$s.t. \left\langle \boldsymbol{P}, B_j(0) \right\rangle = \boldsymbol{\eta}_1^{(i)},\ \forall (i,j) \in [\![0, r-1]\!]^2 \tag{22b}$$

$$\left\langle \boldsymbol{P}, B_j(T_c) \right\rangle = \boldsymbol{\eta}_2^{(i)},\ \forall (i,j) \in [\![0, r-1]\!]^2 \tag{22c}$$

$$-\bar{v} \leqslant \boldsymbol{Q}_i \leqslant \bar{v}, \quad \forall i \in [\![0, n-1]\!] \tag{22d}$$

$$-\bar{a} \leqslant \boldsymbol{R}_i \leqslant \bar{a}, \quad \forall i \in [\![0, n-2]\!] \tag{22e}$$

$$-\bar{j} \leqslant \boldsymbol{S}_i \leqslant \bar{j} \quad \forall i \in [\![0, n-3]\!] \tag{22f}$$

where $\boldsymbol{B}_r$ is the r-th derivative of the B-spline basis function. We recall that $r = 3$ for the case of a quadrotor and we minimize the jerk. Problem 4 will be solved using qpOASES [21] that implements an online active set strategy. Note that the connection time T_c is fixed and we found that choosing $T_c = \tau$ generates a reasonable amount of successful connections.

VII. Simulation and Experimental results

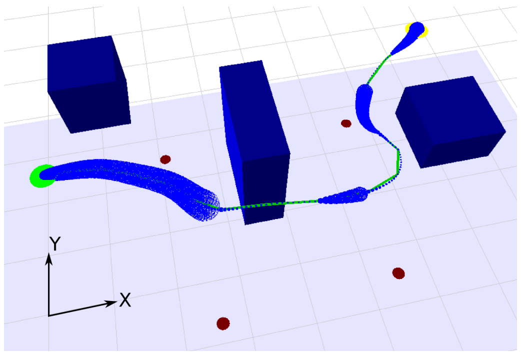
In this section we show some results from different scenarios for the 3D quadrotor. Figure 6 shows two optimal trajectories and Fig. 7 shows the constrained derivatives along a connection considering B-splines of order 4. With the given degree we can see that the curves (e.g., the jerks) are not penalized by conservatism. Finally, Fig. 8 shows the tracking performance for a simulated quadrotor in V-Rep using controller [22]. We can see that despite considering the third-integrator model approximation we can follow the optimal trajectory quite accurately regarding the attitude tracking.

For the experiment illustrated in this section we used a MK-Quadro from MiKroKopter equipped with a front-looking camera with a field of view of 45° tracking the AprilTags with ViSP [23]. The setup includs an on-board ODROID-XU4 Linux computer running ROS and the TeleKyb framework [24] for controlling the quadrotor. An optimal trajectory computed off-line using a jerk input $\bar{j} = 4m.s^{-3}$ is tracked by the system (see Fig. 9) in presence of two obstacles (blue boxes) and four landmarks. We invite the reader to refer to the attached video for more simulation and experimental results.

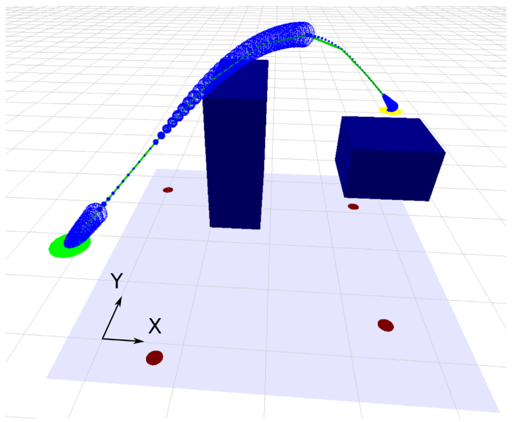
Finally, Problem 4 is solved within 5 ms for the 3D quadrotor after about 90 SQP iterations and the planner is able to find an optimal trajectory in 5 to 10 seconds. The presented algorithm has been also applied to the unicycle case by considering constraints on the real system inputs. Details and results are reported in [25].

VIII. Conclusion and future directions

In this work we proposed to incorporate perception constraints in a graph-search planner for planning minimum-time and feasible trajectories for flat dynamic systems. The optimization framework allows exact connection between a given initial and final states while ensuring collision avoidance and bounded final uncertainty at the goal by accounting for the state uncertainty at the planning stage. In this paper



(a) An optimal trajectory providing robust collision avoidance and guaranteed visual perception.



(b) In this case the quadrotor is able to increase its height to compensate for the rotation of the camera and to enlarge its field of view.

Fig. 6: Two optimal trajectories for the quadrotor with $\tau = 0.35s$ and $\bar{j} = 10m.s^{-3}$ in a 12x8x5m^3 operating region considering four visual landmarks (red dots). The initial and final states are chosen such that no landmark is visible so the quadrotor starts with some uncertainty and is able to reach the goal with a bounded uncertainty by observing the landmarks during its motion. Note that the motion primitives are not represented.

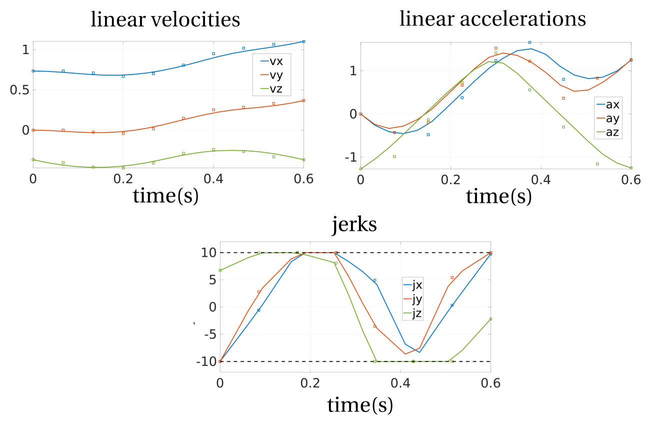


Fig. 7: Constrained velocities, accelerations and jerks along the connection trajectory with bounds $\bar{v} = 2m.s^{-1}$, $\bar{a} = 4m.s^{-2}$ and $\bar{j} = 10m.s^{-3}$. The small squares represent the B-spline control points.

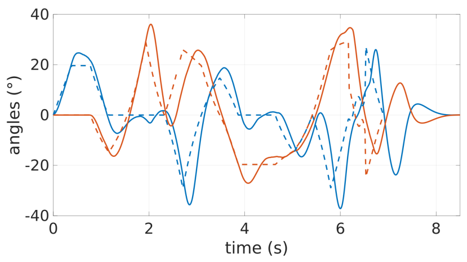


Fig. 8: Plots of the attitude tracking. The dashed lines are the command values while the solid lines show the actual robot attitude in V-Rep.

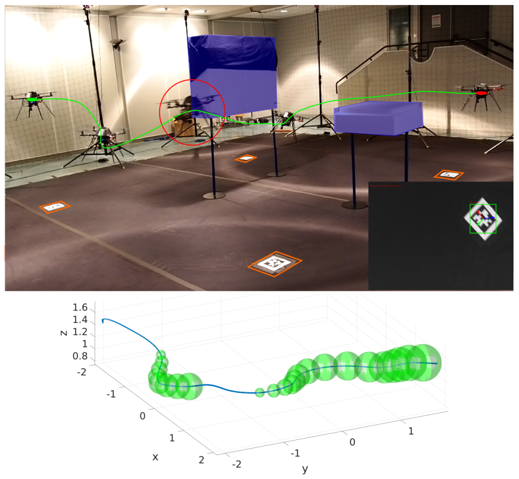


Fig. 9: Snapshots of the quadrotor during the experiment. The quadrotor is tracked with a Vicon system and follows an optimal trajectory (in green) along which landmarks (the AprilTags in orange) are visible on some portions. The lower right figure shows an Apriltag tracked using ViSP when the quadrotor is at the configuration circled in red. The evolution of the uncertainty is shown below after running the Kalman filter on the recorded data and using the AprilTags detection. The landmark at the goal is not taken into account in the planning and is was used to check that the quadrotor reaches its goal within the expected confidence region.

we consider visual measurements that are function of the attitude and propose an efficient optimal graph rewiring by exploiting the convex-hull property of B-splines. Of course, other exteroceptive sensors could be considered. The planner success rate depends on the motion primitives parameters, the ball regions radius and the maximal number of iterations. It could be possible to replan optimal trajectories during motion and even consider dynamic obstacles for the unicycle case (see [25]). Note that the derivative-free Kalman filter in the flat space drastically reduces the computation load for the state estimation compared to an EKF with the real nonlinear quadrotor dynamics. Then, we assumed the landmarks position is known but it would possible to incorporate their position uncertainty in the planner. Finally, we believe the triple-integrator approximation of the quadrotor could be modelled as an additional noise on the model. This would allow a more adequate representation of the real model.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[2] M. Rafieisakhaei, S. Chakravorty, and P. Kumar, "Belief space planning simplified: Trajectory-optimized lqg (t-lqg)," *arXiv preprint arXiv:1608.03013*, 2016.

[3] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *ICRA'11*. IEEE, 2011, pp. 723–730.

[4] J. P. Gonzalez and A. Stentz, "Planning with uncertainty in position an optimal and efficient planner," in *IROS'05*. IEEE, 2005, pp. 2435–2442.

[5] G. Costante, C. Forster, J. Delmerico, P. Valigi, and D. Scaramuzza, "Perception-aware path planning," *arXiv preprint arXiv:1605.04151*, 2016.

[6] B. Davis, I. Karamouzas, and S. J. Guy, "C-opt: Coverage-aware trajectory optimization under uncertainty," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1020–1027, 2016.

[7] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using differential dynamic programming in belief space," in *Robotics Research*. Springer, 2017, pp. 473–490.

[8] K. Hausman, J. Preiss, G. S. Sukhatme, and S. Weiss, "Observability-aware trajectory optimization for self-calibration with application to uavs," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1770–1777, 2017.

[9] R. Pepy, M. Kieffer, and E. Walter, "Reliable robust path planning with application to mobile robots," *International Journal of Applied Mathematics and Computer Science*, vol. 19, no. 3, pp. 413–424, 2009.

[10] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *IROS'17*. IEEE, 2017, pp. 2872–2879.

[11] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based Motion Planning for Aggressive Flight in SE (3)," *arXiv preprint arXiv:1710.02748*, 2017.

[12] L. Bascetta, I. M. Arrieta, and M. Prandini, "Flat-rrt*: A sampling-based optimal trajectory planner for differentially flat vehicles with constrained dynamics," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6965–6970, 2017.

[13] M. Pivtoraiko, D. Mellinger, and V. Kumar, "Incremental micro-uav motion replanning for exploring unknown environments," in *ICRA'13*. IEEE, 2013, pp. 2452–2458.

[14] B. Penin, P. Robuffo Giordano, and F. Chaumette, "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions," *IEEE Robotics and Automation Letters*, 2018.

[15] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of nonlinear systems: Introductory theory and examples," *International J. of Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[16] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2018.

[17] G. G. Rigatos, "Derivative-free kalman filtering for autonomous navigation of unmanned ground vehicles," in *ICSCS'12*. IEEE, 2012, pp. 1–6.

[18] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[19] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.

[20] W. Van Loock, G. Pipeleers, and J. Swevers, "B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction," *Mechanical Sciences*, vol. 6, no. 2, p. 163, 2015.

[21] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[22] T. Lee, M. Leokyand, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *49th IEEE Conf. on Decision and Control*, Atlanta, GA, Dec. 2010, pp. 5420–5425.

[23] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40–52, December 2005.

[24] V. Grabe, M. Riedel, H. H. Bülthoff, P. R. Giordano, and A. Franchi, "The TeleKyb Framework for a Modular and Extendible ROS-based Quadrotor Control," in *ECMR'13*. IEEE, 2013, pp. 19–25.

[25] B. Penin, "Contributions to optimal and reactive vision-based trajectory generation for a quadrotor uav," Ph.D. dissertation, Université Rennes 1, December 2018, http://rainbow-doc.irisa.fr/pdf/2018_phd_penin.pdf.