Motion Planning

Alexandre Boeuf, Juan Cortés, Thierry Simeon

Chapter 5.2
Motion Planning

Alexandre Boeuf, Juan Cortés, Thierry Siméon

Abstract This chapter presents a kinodynamic motion planner for computing agile motions of quad-rotor-like aerial robots in constrained environments. Based on a simple dynamic model of the UAV, a computationally-efficient local planner is proposed to generate flyable trajectories of minimal time. This local planner is then integrated as a key component for global motion planning using different approaches. The good performance of the proposed methods is illustrated with results in simulation, as well as a preliminary experimentation with a real quad-rotor.

1 Introduction

When planning motions for Unmanned Aerial Vehicles (UAVs) such as quadrotors, it is important to consider the dynamic model of the system since not every geometrically valid path corresponds to a feasible motion. For example, because of its dynamic behavior, flying upside down, even for a relatively short period of time, is hardly manageable for a fixed-pitch quadrotor. Aiming to avoid the difficulties and the high computational cost involving kinodynamic motion planning [14], the problem is usually treated in two stages. The first stage applies a basic path planning approach, disregarding dynamic considerations. For this, sampling-based path planning algorithms [13], such as the Rapidly-exploring Random Tree (RRT) or the Probabilistic Roadmap (PRM), can be used to produce a collision-free path for the center of mass of the robot. Indeed, since the robot orientation depends on dynamic aspects, collisions cannot be tested for the robot itself but for its smallest bounding-sphere. In a second stage, this path, usually consisting of a sequence of straight-line segments in R^3, has to be transformed into a dynamic trajectory. Trajectory generation methods, such as [20, 16, 8], can be applied in this stage to each portion of the path. The overall trajectory then consists of a sequence of movements from one hovering position to another, which leads to severe sub-optimality in terms of execution time. However, such an unsuitable trajectory can be subsequently optimized using several types of algorithms.

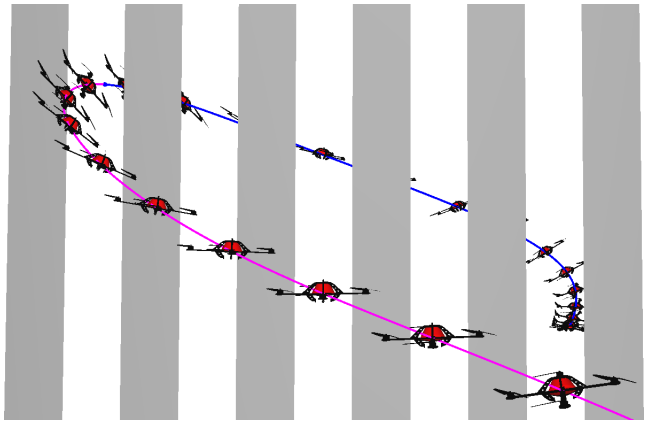

Fig. 1 Solution to the *slot problem* obtained with a basic RRT algorithm using the proposed local planner as a steering method.

The aforementioned *decoupled approach* is computationally efficient, and can be successfully applied to solve many motion planning problems for UAVs (see for example [10, 5, 18]). However, several classes of problems cannot be treated using this approach, because the robot orientation cannot be properly considered at the geometric stage. One of such problems, which we refer to as the *slot problem*, is illustrated in Fig. 1. In this problem, the robot has to go through a narrow slot-shaped passage whose width is smaller that the diameter of the smallest bounding-sphere of the robot. Since a collision-free path does not exist for this sphere, the decoupled approach will fail to find a solution.

Another type of problem that cannot be treated using a decoupled approach is the transportation of a large object (at medium or high velocity). When carrying such an object, it is not suitable anymore to consider the minimum bounding-sphere of the system because it does not fit its actual shape, thus leading to invalidity of far too many possible solutions. In these situations, collisions have to be tested for the actual shape of the system, and hence, non-hovering states have to be sampled and linked by collision-free flyable trajectories. According to [16], such trajectories must be smooth in the space of the flat outputs of the system with bounds on their derivatives. Therefore, motion planning in this context requires an efficient trajectory generation method able to interpolate two kinodynamic states (position, velocity and acceleration). Unfortunately, available trajectory generation methods are either computationally expensive, or unlikely for the present application, since they do not guarantee the respect of limits in velocity and acceleration, or because they require flying time as an input parameter [17].

This chapter presents an efficient kinodynamic motion planner for computing agile motions of quad-rotor-like aerial robots in constrained environments like the *slot* problem of Fig. 1. The approach first builds on a local trajectory planner (see Sec. 2) that uses fourth-order splines and a simplified model of the UAV to generate flyable trajectories of minimal time with respect to the closed form solution. We next explain in Sec. 3 how these local feasible trajectories can be used as part of

Table 1 Acronyms and symbols.

Symbol	Definition
$\mathbf{x}$	Kynodynamic state
$\mathscr{X}$	State space
$\mathscr{P}$	Admissible values for the positions
$\mathscr{V}$	Admissible values for the velocity
$\mathscr{A}$	Admissible values for the acceleration
$\mathscr{J}$	Admissible values for the jerk
$\mathscr{S}$	Admissible values for the snap
S	Trajectory in the state of the flat outputs
T	Total time of the motion
T_i	Total time of the motion for each component
$s(t)$	Time parameterized fourth order spline
q	Configuration (notation used in the explanations of sampling-based planners)
$\mathscr{T}_S$ and $\mathscr{T}_G$	Trees rooted at start and goal configurations, respectively
$\mathscr{G}$	Graph in the configuration/state space
G_i	A connected component in $\mathscr{G}$
$A_{\mathscr{G}}$	Adjacency matrix associated with $\mathscr{G}$

an optimization method in a decoupled approach, or applied directly as a steering method within a sampling-based motion planner for solving more constrained problems. This section also presents two improvements: an efficient quasi-metric and an incremental sampling strategy. Both techniques enhance the performance of kinodynamic planning, as illustrated by the simulation results reported in Sec. 4. Finally, preliminary experimental results with a real quadrotor are presented in Sec. 5.

Table 1 summarizes the main symbols used in this chapter.

2 Steering Method

This section briefly recalls some notions about the state space of a quadrotor and gives an overview of the steering method presented in [3][1]. The steering method uses fourth-order splines to generate smooth trajectories between two arbitrary states of a quadrotor, ensuring continuity up to the jerk (and therefore, up to the angular velocity), which is important to facilitate controllability along the planned trajectories. These trajectories minimize flying time (with respect to the proposed closed-form solution) while respecting kinodynamic constraints.

Quadrotor model : A quadrotor is a free-flying object in $\mathbb{R}^3$. It has three degrees of freedom in translation (position of the center of mass: $[x,y,z] \in \mathbb{R}^3$) and three in rotation (roll, pitch and yaw angles: $[\varphi,\phi,\psi]$). However, due to under-actuation (six degrees of freedom but only four actuators), these degrees of freedom (considering their derivatives) are not independent. Indeed, the φ and ϕ angles are determined

[1] The code (C and MATLAB versions) is available upon request.

from the thrust vector, which in turn depends on the velocity and the acceleration. Considering the standard dynamic model described in Chapter **??**, the system is differentially flat for the outputs $[x, y, z, \psi]$ (see [16]). This implies that the full configuration can be determined from these four variables, and that any trajectory in the state of the flat outputs is flyable if their derivatives are correctly bounded.

State space : We define a kynodynamic state as a vector:

$$\mathbf{x} = [x, y, z, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\psi}, \ddot{x}, \ddot{y}, \ddot{z}, \ddot{\psi}] \in \mathscr{X}$$

with $\mathscr{X}$ the state space, we have $\mathscr{X} = \mathscr{P} \times \mathscr{V} \times \mathscr{A}$, where $\mathscr{P}$, $\mathscr{V}$ and $\mathscr{A}$ are intervals of $\mathbb{R}^4$. Note that although there are no actual limits (if friction is neglected) on linear velocity of the center of mass of a quadrotor, for safety reasons, it might be interesting to take such limits into consideration for trajectory planning.

Steering method : The steering method we consider provides a solution (S, T) to the problem:

$$\begin{cases} [S(0)\ \dot{S}(0)\ \ddot{S}(0)] = \mathbf{x}_0 \in \mathscr{X} \\ [S(T)\ \dot{S}(T)\ \ddot{S}(T)] = \mathbf{x}_T \in \mathscr{X} \\ \forall t \in [0,T] \begin{cases} \dot{S}(t) \in \mathscr{V} \\ \ddot{S}(t) \in \mathscr{A} \\ \dddot{S}(t) \in \mathscr{J} \\ \ddddot{S}(t) \in \mathscr{S} \end{cases} \end{cases} \tag{1}$$

where S is the trajectory in the state of the flat outputs, T is the total time of the motion and $\mathscr{V}$, $\mathscr{A}$, $\mathscr{J}$ and $\mathscr{S}$ are zero centered intervals of $\mathbb{R}^4$. Obtaining the time-optimal trajectory is difficult and computationally expensive. This is not well suited for integration in sampling-based motion planners since they make extensive use of the steering method (usually thousands of calls to solve constrained planning problems). Therefore, our solution is to compute a trajectory that approaches the optimal one with a simpler (imposed) shape enabling a rapid, analytical solution. We provide below a short overview of the steering method described in [3]. From a control point of view, it can be seen as an on-off snap based command, which means that for each coordinate the snap has always maximum, minimum, or zero values. This leads to piecewise polynomial solutions (splines) of degree four. The main idea is to find for each coordinate a set of snap commutations that minimize the overall time.

The method works in two stages. First a solution is computed independently for each output x, y, z and ψ. Each solution is a fourth order spline $s(t)$ with a bang-null snap profile (i.e. a trapezoidal jerk profile) such that $[s(0)\ \dot{s}(0)\ \ddot{s}(0)] = [x_{0i}\ v_{0i}\ a_{0i}]$ and $[s(T_i)\ \dot{s}(T_i)\ \ddot{s}(T_i)] = [x_{Ti}\ v_{Ti}\ a_{Ti}]$ with $T_i \in \mathbb{R}^+$ (different for each output of index i). These splines are divided into seven main temporal phases (see Fig. 2). Four of them (noted A, C, E and H) correspond to acceleration variations. Phases B and G are constant acceleration phases. During phase D velocity $v(t) = \dot{s}(t)$ is constant. Durations of all phases are functions of this constant velocity value v_D. During the first phase A, acceleration $a(t) = \ddot{s}(t)$ is getting from its initial value a_0 to an intermediate value noted a_B (also function of v_D). Similarly, during the last phase H,

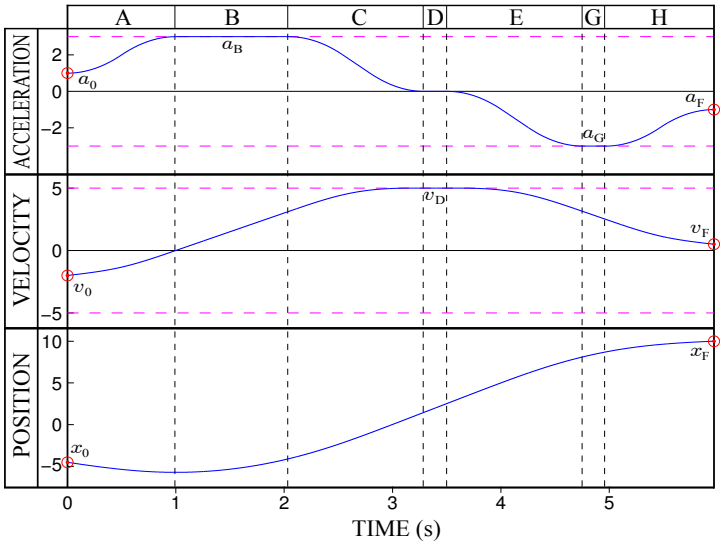

Fig. 2 Illustration of the seven main temporal phases of the spline for one output. Red circled dots are the values to be interpolated. Pink dashed lines are the boundary values for acceleration and velocity.

acceleration goes from a_G to its final value a_F. The goal is to maximize acceleration variations in order to reach full speed as fast as possible and maintain it as long as possible. Once the commutation times of all coordinates have been calculated, final times will not match. Solutions are then synchronized to form the local trajectory whose total duration is $T = \max_{i=1..4} T_i$. This is done by reducing the velocity reached during phase D (see Fig. 2) for the three fastest coordinates, using a simple dichotomous search in order to match the slowest one

The interesting feature of our approximate steering method lies in its computational speed at the price of a moderate sub-optimality. The statistical analysis reported in [2], comparing the quality of the computed trajectories against the solutions provided by a full-fledged numerical optimization (with the ACADO solver), shows that the sub-optimality of our approximate solution is really low (3%) and well justified by a high gain in computation time (10^3). Solutions are computed with an average running time of less than a millisecond, thus motivating the use of the proposed approximate 'minimum-time' steering method inside sampling-based planners. Finally, note that the method can be generalized to various types of systems [2]. For a serial manipulator, for instance, each degree of freedom would be treated as a flat output.

[2] The steering method has been has been implemented as a general-purpose and standalone C++ library named KDTP (for KinoDynamic Trajectory Planner) and a git repository is available at `git://git.openrobots.org/robots/libkdtp.git`

3 Kinodynamic Motion Planning

We discuss in this section how the steering method can be used as part of a local optimization method in a decoupled planning approach or applied directly as a steering method within a sampling-based motion planner.

3.1 Decoupled Approach

This planning approach consists of two stages : 1) planning a geometrically valid path in $\mathbb{R}^3$ for the minimum bounding sphere of the quadrotor 2) and then transforming the computed path into a trajectory in $\mathscr{X}$. In this decoupled approach, the piecewise linear path computed using a standard PRM or RRT sampling-based planner is first transformed into a trajectory by calling the steering method on every local-path, positioning velocity and acceleration to zero at end-points. As a result, velocity and acceleration will be correctly set along the local-paths without modifying its geometry. The resulting trajectory remains therefore collision-free since it follows the initial geometric path, but it is far from being time optimal because of the imposed stops at the endpoints of local paths. We then apply a random shortcut technique [7] in order to optimize the trajectory, using the steering method to iteratively improve paths portions between pairs of randomly selected states along the trajectory.

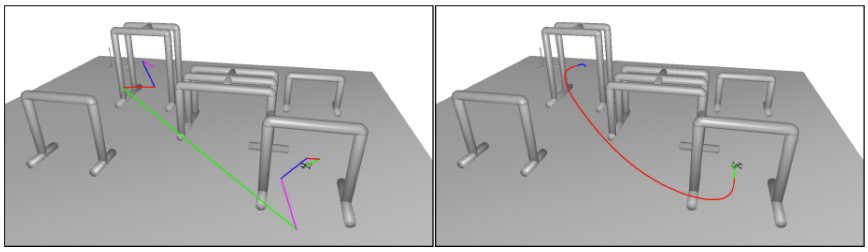

Fig. 3 Decoupled planner : geometric path obtained in the first stage (left) and smoothed trajectory considering the kinodynamic constraints (right)

This simple approach is computationally efficient and it gives good quality solutions for medium-complexity problems. The example illustrated by Fig. 3 was solved in only 0.4 seconds of planning plus 2 seconds of smoothing using the iterative shortcut technique. However this type of approach may fail to solve more challenging problems and is therefore incomplete in the sense that it is unable to find solutions to some problems involving aggressive maneuvers, such as the constrained slot problem illustrated in Fig. 1. The approach is also unsuitable to solve problems involving the transportation of a rigidly-attached large object, whose orientation is defined by that of the quadrotor, and is therefore dependent on its velocity and acceleration

3.2 Direct Kinodynamic Planning Approach

In this section we focus on direct sampling-based approaches to kinodynamic motion planning for quadrotors. We first address the problem of the metric and then present an efficient sampling strategy in the state-space that increases the probability of generating connectable states, and thus the performance of the planner. Then we describe directed variants of the well known RRT and PRM algorithms that need to be adapted to handle the non-symmetry of the steering method and of the associated quasi-metric.

Efficient state-space quasi-metric : The efficiency of the state-space exploration using randomized kinodynamic motion planning algorithms relies on a good distance metric. However, as discussed in [12], computing the actual distance between two states is as hard (and thus as costly) as solving the corresponding optimal control problem. Our steering method provides a deterministic sub-optimal solution to such a control problem. Therefore, it defines a quasi-metric[3] $M^*_{SM} : (\mathbf{x}_0, \mathbf{x}_T) \mapsto T$ on the state space. Because of the dynamics of the system, a trajectory in the state space from $\mathbf{x}_0$ to $\mathbf{x}_T$ is indeed necessarily different from a trajectory from $\mathbf{x}_T$ to $\mathbf{x}_0$, and thus M^*_{SM} is not symmetric. Although this steering method is computationally fast, it is still too costly to be used for nearest neighbor search inside a sampling-based planner. This section presents a method to approximate the quasi-metric M^*_{SM} at a very low computational cost, and presents results that show its relevance.

The complexity of the problem (1) defined in Section 2 is mainly due to its order (four) and to the inequality constraints on the derivatives. We propose to solve a simpler time optimal control problem for the third order (i.e. by considering the jerk as the control input), in one dimension and without constraints other than the bounds on the control input. The problem is then to find for each output of index i the couple (S_i, T_i) such that:

$$\begin{cases} \min T_i \in \mathbb{R}^+ \text{ s.t.} \\ [S_i(0)\ \dot{S}_i(0)\ \ddot{S}_i(0)] = [x_0\ v_0\ a_0] \in \mathbb{R}^3 \\ [S_i(T_i)\ \dot{S}_i(T_i)\ \ddot{S}_i(T_i)] = [x_{T_i}\ v_{T_i}\ a_{T_i}] \in \mathbb{R}^3 \\ \forall t \in [0, T_i],\ |\dddot{S}_i(t)| \leq j_{max} \in \mathbb{R}^+ \end{cases} \tag{2}$$

For this simple integrator of the third order without constraints on the state, Pontryagin maximum principle (see for example [1]) says that the optimal control is necessarily saturated, i.e.:

$$\forall t \in [0, T_i],\ \dddot{S}_i(t) \in \{-j_{max}, j_{max}\}$$

with at most two control commutations. Solving (2) implies to find T_i and these (at most) two commutation times, which requires to solve polynomial equations of maximum degree four. Further details on how to derive the solution of this problem can be found in [2] The proposed quasi-metric is then defined as:

[3] A quasi-metric has all the properties of a metric, symmetry excepted.

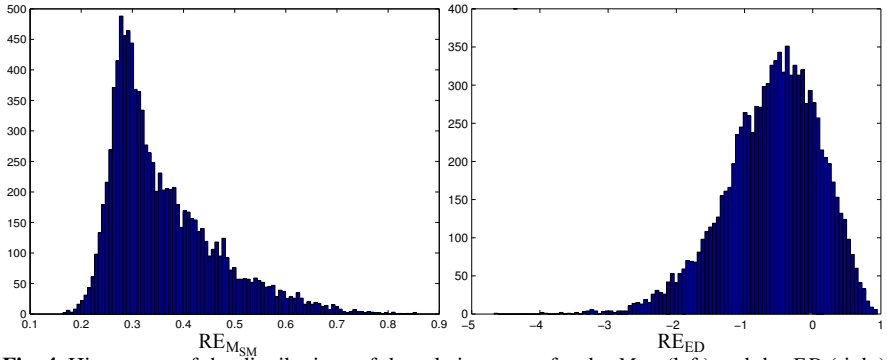

Fig. 4 Histograms of the distributions of the relative errors for the M_{SM} (left) and the ED (right) metrics

$$M_{SM} : (\mathbf{x}_0, \mathbf{x}_T) \mapsto \max_{i=1..4} T_i$$

Here we present results of an experimental test to validate the proposed approximate quasi-metric. 10^4 pairs of kinodynamic states were randomly sampled in $\mathscr{X} = [-5,5]^3 \times [-5,5]^3 \times [-10,10]^3$, considering $\mathscr{J} = [-20,20]^3$ and $\mathscr{S} = [-50,50]^3$. Note that, without loss of generality and for simplification purposes, we consider here a constant yaw. For each pair $(\mathbf{x}_1, \mathbf{x}_2)$, we computed the value $M^*_{SM}(\mathbf{x}_1, \mathbf{x}_2)$ of the quasi-metric induced by our steering method, the value $M_{SM}(\mathbf{x}_1, \mathbf{x}_2)$ given by the proposed approximation, and the value $ED(\mathbf{x}_1, \mathbf{x}_2)$ of the euclidean distance in $\mathbb{R}^3$ considering only the position of the center of mass. We study the distribution of the relative error between M^*_{SM} and M_{SM}, i.e. the quantity:

$$RE_{M_{SM}}(\mathbf{x}_1, \mathbf{x}_2) = 1 - \frac{M_{SM}(\mathbf{x}_1, \mathbf{x}_2)}{M^*_{SM}(\mathbf{x}_1, \mathbf{x}_2)}$$

For comparison, we also provide the relative error RE_{ED} between M^*_{SM} and ED. Fig. 4 shows histograms of the distributions of these errors, The low standard deviation of the distribution of the relative error for the proposed quasi-metric is a measure of the quality of the approximation. These results also provide empirical evidence that M_{SM} and M^*_{SM} are equivalent since for all pairs $(\mathbf{x}_1, \mathbf{x}_2)$, we have $0.16396 \leq RE_{M_{SM}}(\mathbf{x}_1, \mathbf{x}_2) \leq 0.85540$ which therefore implies that $\frac{1}{10}.M^*_{SM}(\mathbf{x}_1, \mathbf{x}_2) < M_{SM}(\mathbf{x}_1, \mathbf{x}_2) < 10.M^*_{SM}(\mathbf{x}_1, \mathbf{x}_2)$ This means that M_{SM} and M^*_{SM} are inducing the same topology on $\mathscr{X}$, and thus that the cost-to-go defined by our steering method is correctly evaluated by M_{SM}. This is clearly not the case for the euclidian metric ED. While M_{SM} represents a good approximation of the real cost-to-go, it is also computationally much faster (20 times) than M^*_{SM}, which motivates its use as distance metric in order to increase the performance of the planning algorithms, as shown by the simulation results of Section 4

Sampling connectible states : We present below an efficient state sampling strategy proposed in [4] with the aim to increase the probability of generating "connectable"

states and thus the performance of the planning algorithms. Indeed, trajectories generated by the steering method presented in Section II do not guarantee to respect bounds on the position of the robot. Such a constraint is typically violated when samples are close to the boundary of the workspace and the velocity is high, so that it is not possible to decelerate to avoid crossing this positional limit. In a similar way, bounds on velocity can also be violated. If acceleration is too high and velocity is close to the limit, produced trajectories will be invalid because velocity can not be reduced in time to meet the constraints. Note however that the imposed shape for the trajectories produced by our steering method guarantees that bounds on acceleration are respected.

We say that a kinodynamic state $\bar{\mathbf{x}} \in \mathscr{X}$ is *forward-connectible* (respectively *backward-connectible*) if and only if there is a state $\mathbf{x} \in \mathscr{X}$ such that the local path $(\bar{\mathbf{x}}, \mathbf{x})$ (respectively $(\mathbf{x}, \bar{\mathbf{x}})$) produced by the steering method lies entirely in $\mathscr{X}$ (i.e. respects bounds on position, velocity and acceleration). A state that is forward-connectible and backward-connectible is said to be connectible (non-connectible otherwise). This is illustrated on Fig. 5. In case of uniform sampling of the state space, non-connectible states can be generated, and thus, local paths computed to connect those states have to be discarded *a posteriori* by the planner. This is rather inefficient since generating and testing a local path for validity is a costly operation. The goal of the proposed sampling technique is to notably reduce the probability of generating non-connectible states, and hence to improve the performance of planning algorithms.

The sampling technique proceeds in a decoupled and incremental way. First, acceleration is uniformly sampled. The idea is then to compute a set of velocity values for which the state is known to be non-connectible. Velocity is then uniformly sampled outside this set. Finally, given this couple (velocity, acceleration) a set of position values for which the state is known to be non-connectible is computed, and the position is then uniformly sampled outside this set.

Fig. 6 illustrates how to compute given a uniformly sampled acceleration value a_s for one output the set of velocity values for which the state is known to be non-connectible, Let us denote v_{max} and a_{max} as the bounds on the absolute value of velocity and acceleration respectively. We study acceleration $a(t)$ and velocity $v_0(t)$ on a neighborhood around $t = 0$ for $a(0) = a_s$ and $v_0(0) = 0$. The idea is to apply a saturated acceleration variation and determine the extrema of $v_0(t)$ in this neighborhood. Using them, we can compute the limits on velocity v_s such that

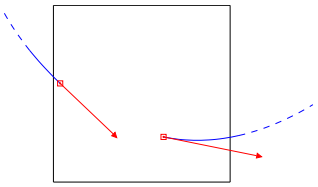

Fig. 5 Examples of non-connectible states in two dimensions. Red squares are positions and red arrows are velocity vectors. Blue curves are examples of trajectories. Bounds on position are represented in black. The state on the left is not backward-connectible. The state on the right is not forward-connectible.

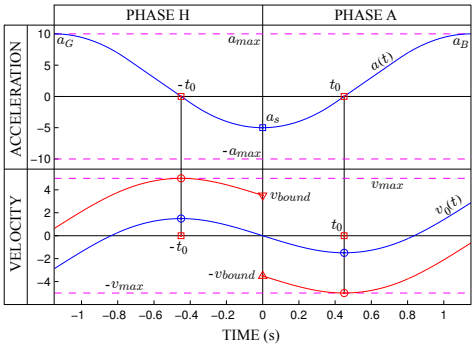

Fig. 6 Acceleration and velocity (blue curves) around $t = 0$. Saturated acceleration variation is applied in order to determine the limits on initial velocity (red triangles). Red curves are velocities for these limits.

$v(t) = v_0(t) + v_s$ lies in $[-v_{max}, v_{max}]$. We use notations defined in section 2. For $t > 0$, phase A of our steering method is applied. Phase H is applied for $t < 0$. The sampled value a_s locally imposes a direction of variation of $v_0(t)$ on phases A and H. We want to reverse this direction of variation in minimum time. This is equivalent to driving $a(t)$ to zero in minimum time. For that, we set $a_B = a_G = -\text{sign}(a_s).a_{max}$. This corresponds to the highest acceleration variation achievable by our steering method. Note that, by construction, acceleration is symmetric during phases A and H (i.e $a(-t) = a(t)$) and $v_0(t)$ is anti-symmetric (i.e $v_0(-t) = -v_0(t)$). Since $a(t)$ is a second order spline strictly monotonic on phase A, it is straightforward to compute the unique $t_0 > 0$ such that $a(t_0) = 0$. The value $v_{bound} = v_{max} - |v_0(t_0)|$ is then the upper bound on the absolute value of v_s. This means that if $|v_s| > v_{bound}$ then $v(t) = v_0(t) + v_s$ will violate the constraints on velocity. A velocity value v_s is then uniformly sampled in $[-v_{bound},\ v_{bound}]$.

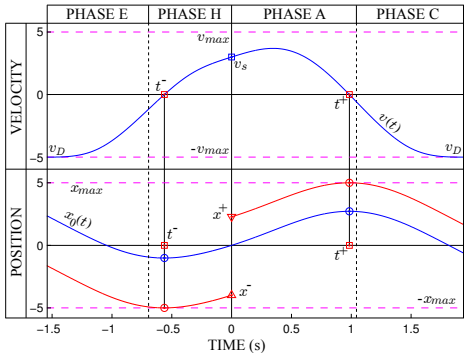

Fig. 7 Velocity and position (blue curves) around $t = 0$. Saturated velocity variation is applied in order to determine the limits on initial positions (red triangles). Red curves are positions for these limits.

Given a couple (v_s, a_s) for one output, Fig. 7 illustrates how the set of position values for which the state is known to be non-connectible can be computed The principle is similar to the one above for velocity sampling. Velocity $v(t)$ and position $x_0(t)$ are studied around $t = 0$ for $a(0) = a_s$, $v(0) = v_s$ and $x_0(t) = 0$. We apply a saturated velocity variation and determine the extrema of $x_0(t)$ in this neighborhood. Using them, we can compute the limits on position x_s such that $x(t) = x_0(t) + x_s$ lies in $[-x_{max}, x_{max}]$ (bounds on position). For $t > 0$, phases A to C of our steering method are applied. Phases E to H are applied for $t < 0$. We want to reverse the direction of variation of the position imposed by v_s as fast as possible. This is equivalent to driving $v(t)$ to zero in minimum time. For that, we set $v_D = -\text{sign}(v_s).v_{max}$ for both phases A to C and E to H. This corresponds to the highest velocity variation achievable by our steering method. The only difference here is that neither $v(t)$ nor $x_0(t)$ have symmetry proprieties. We compute $t^+ > 0$ such that $v(t^+) = 0$ and $t^- < 0$ such that $v(t^-) = 0$. If $v_s \geq 0$ then $x^+ = x_{max} - x_0(t^+)$ and $x^- = -x_{max} - x_0(t^-)$ else $x^+ = x_{max} - x_0(t^-)$ and $x^- = -x_{max} - x_0(t^+)$. A position value x_s is then uniformly sampled in $[x^-,\ x^+]$.

Directed bi-RRT : In the well known undirected version of the bi-RRT algorithm [11], two trees $\mathscr{T}_S$ and $\mathscr{T}_G$ are constructed in parallel. $\mathscr{T}_S$ grows from the start configuration and $\mathscr{T}_G$ from the goal configuration. Each iteration for one of the trees consists of sampling a configuration q_{rand}, finding its nearest neighbor q_{near} in the tree (according to a defined metric), and extending it toward q_{rand} (using a steering method) to create a new configuration q_{new}. Each time an expansion is successful for one of the trees, a direct connection is attempted between q_{new} and its nearest neighbor in the other tree. The algorithm ends if this local path is valid (i.e. when the trees are connected). In our directed version, both the steering method and the quasi-metric M_{SM} are non-symmetric, and thus have to be called taking care of the order of the two states. The nearest neighbors $N_S(\bar{\mathbf{x}})$ and $N_G(\bar{\mathbf{x}})$ of a state $\bar{\mathbf{x}}$ in $\mathscr{T}_S$ and $\mathscr{T}_G$ respectively are defined as such:

$$\begin{cases} N_S(\bar{\mathbf{x}}) = \underset{\mathbf{x} \in \mathscr{T}_S}{\arg\min} M_{SM}(\mathbf{x}, \bar{\mathbf{x}}) \\ N_G(\bar{\mathbf{x}}) = \underset{\mathbf{x} \in \mathscr{T}_G}{\arg\min} M_{SM}(\bar{\mathbf{x}}, \mathbf{x}) \end{cases}$$

For an expansion of $\mathscr{T}_S$, we test the local path $\left(N_S(\mathbf{x}_{rand}), \mathbf{x}_{new}\right)$ for validity. In case of success, the algorithm ends if the local path $\left(\mathbf{x}_{new}, N_G(\mathbf{x}_{new})\right)$ is valid. For an expansion of $\mathscr{T}_G$, the local path $\left(\mathbf{x}_{new}, N_G(\mathbf{x}_{rand})\right)$ is tested for validity, and the algorithm ends in case of validity of the local path $\left(N_S(\mathbf{x}_{new}), \mathbf{x}_{new}\right)$.

Directed PRM : At each iteration of the undirected version of the PRM algorithm [9], a collision free configuration q is sampled and added to the graph $\mathscr{G}$. For every connected component G_i of $\mathscr{G}$, connections are attempted between q and each node of G_i in increasing order of distance from q until one is successful. A threshold on this distance can be considered with the aim to reduce computational cost. In our directed version, we consider the strongly connected components G_i of $\mathscr{G}$. Moreover, we maintain during the execution the adjacency matrix $A_{\mathscr{G}}$ of the

transitive closure of the graph of the strongly connected components of G. This square matrix, whose dimension is the number of strongly connected components, is defined by $A_{\mathscr{G}}[i][j] = 1$ if a path in $\mathscr{G}$ exists from every node of G_i to every node of G_j and $A_{\mathscr{G}}[i][j] = 0$ otherwise. If $A_{\mathscr{G}}[i][j] = 1$ we say that G_i is connected to G_j. Note that $A_{\mathscr{G}}[i][j] = A_{\mathscr{G}}[j][i] = 1$ if and only if $i = j$. At each iteration, a valid state $\mathbf{x}$ is sampled and added to $\mathscr{G}$ (which has n strongly connected components). Its strongly connected component $G_{n+1} = \{\mathbf{x}\}$ is added to the matrix $A_{\mathscr{G}}$. For every connected component G_i of $\mathscr{G}$ ($i = 1..n$), if G_i is not connected to G_{n+1}, connections from every node $\mathbf{x}_j$ of G_i to $\mathbf{x}$ are attempted in increasing order of $M_{SM}(\mathbf{x}_j, \mathbf{x})$ until one is valid. As for the undirected version, a threshold on the value of M_{SM} can be considered here. $A_{\mathscr{G}}$ is updated if neecessary. Then, if G_{n+1} is not connected to G_i, connections from $\mathbf{x}$ to every node $\mathbf{x}_j$ of G_i are attempted in increasing order of $M_{SM}(\mathbf{x}, \mathbf{x}_j)$ until one is valid. If used in single query mode, the algorithm ends when the strongly connected component of the initial state is connected to the strongly connected component of the goal state.

4 Simulation Results

Results presented below show the influence of the quasi-metric and the sampling technique on the two previously presented motion planners. Experiments have been conducted on two different environments shown in Fig. 8 and for the same quadrotor whose diameter is equal to 0.54 meters. We consider $\mathscr{V} = [-5,5]^3$, $\mathscr{A} = [-10,10]^3$, $\mathscr{J} = [-20,20]^3$, $\mathscr{S} = [-50,50]^3$ (using SI base units). Yaw is kept constant.

The first environment, referred to as *boxes*, is a cube with side length of 10 meters filled with box shaped-obstacles of different sizes. The second environment, referred to as *slots*, is also a cube with side length of 10 meters but divided in two halves by a series of aligned obstacles separated by 0.40 meters (hence smaller than the robot diameter). This problem is particularly challenging since going across these obstacles requires to find a path in a very narrow passage in the state-space. Every combination of environment, algorithm, metric and sampling strategy has been tested. Results are provided in Tab. 2 for CPU and flying times in seconds, number of nodes (and iterations for the RRT) and percentage of not connectible nodes (with

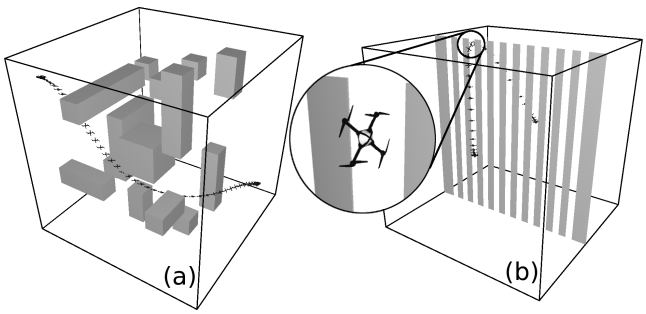

Fig. 8 Testing environments (a) *boxes* (b) *slots*

Table 2 B: boxes, S: slots, P: prm, R: rrt, M: proposed metric, E: euclidean metric, I: incremental sampling, U: uniform sampling

Experiment	B.P.M.I.	B.P.E.I.	B.P.M.U.	B.P.E.U.	S.P.M.I.	S.P.E.I.	S.P.M.U.	S.P.E.U.
CPU time (s)	0.05648	0.07884	3.284	4.409	1.505	1.220	578.5	444.2
Flying time (s)	8.180	8.772	8.000	8.126	9.074	8.979	8.615	8.387
Number of nodes	12.11	13.77	78.64	88.33	71.93	61.59	767.9	725.8
% of not connectible nodes	0	0	82.53	84.38	0	0	89.11	89.19
Experiment	B.R.M.I.	B.R.E.I.	B.R.M.U.	B.R.E.U.	S.R.M.I.	S.R.E.I.	S.R.M.U.	S.R.E.U.
CPU time (s)	0.02780	0.04088	0.04144	0.05612	2.165	2.466	558.8	512.9
Flying time (s)	9.674	10.84	9.365	10.09	25.42	34.72	33.96	55.98
Number of nodes	8.79	8.84	9.18	10.77	334.5	565.6	4429.4	8813.0
Number of iterations	26.45	45.04	45.58	65.02	982.9	2502.9	30253.7	196233.3
% of not connectible nodes	50.34	54.94	51.51	59.85	25.60	34.86	79.41	82.48

respect to the criteria defined in section 3). Each experiment is designated by an acronym whose meaning is explained in the caption. Results are averaged over 100 runs and are for an implementation in C, integrated in our motion planning software Move3D [19], and run on a single core of an Intel Xeon W3520 processor at 2.67GHz.

Results show a significant improvement of the performance of both algorithms thanks to the integrations of the proposed techniques. However, one can clearly see that the metric and the sampling technique have a more notable effect on one or the other planner. Results for the PRM algorithm shows that the sampling method has a great influence on its performance. Its integration indeed improves CPU time by two orders of magnitude for both environments. On the other hand, one can see that for the *slots* environment CPU times are slighlty worse with the use of the metric. This can be explained by the difference of computing time between our quasi-metric and the euclidean distance. This is also observed in one case for the RRT algorithm (SRMU vs. SREU). For the RRT algorithm, results show that the influence of the metric is more important. This was to be expected since RRT-based algorithms are known to be very sensitive to the metric. One can see that the number of iterations is significantly reduced, meaning that the search is better guided. The improvement produced by the sampling technique is also very significant for the *slots* environment but less noticeable for the *boxes* environment. This can be explained by the fact that, in RRT-based algorithms, the sampled states are not tested for connections but used to define a direction for extension. A new state is then generated according to that direction. One can see that, for the *boxes* environment, about half of these states are not connectible regardless of the sampling method. Finally note that flying times are given for the raw, non-smoothed trajectories, which explains the rather large difference of path quality between PRM and RRT results.

5 Experiment

We also performed some preliminary experimental validation of the planning algorithms in the Aerial Robotics Testbed from LAAS-CNRS (see Fig. 9) to show that the planned trajectories using the steering method of Section 2 can actually be

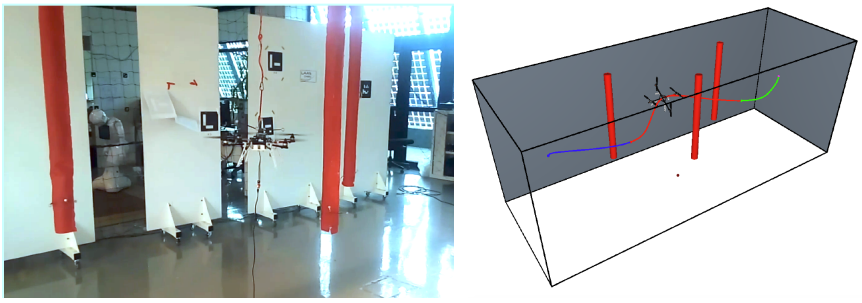

Fig. 9 Experimental set-up with the MikroKopter quadrotor and example of planned trajectory

executed by a real quadrotor. The arena consists of an indoor flight volume with a ground area of 6 by 3 meters surrounded by a safety net at a height of 4 meters and monitored by an Optitrack system. The quadrotor is a MikroKopter equipped with an accelerometer and a gyroscope. As controller, we implemented the geometric tracking control algorithm proposed by [15]. These first experiments were performed in a simple environment constrained by only 3 cylindrical obstacles (see Fig. 9). Trajectories were planned using the decoupled approach presented in Section 3. These first results show the ability of the quadrotor to follow correctly the planned trajectory, although some tracking error was observed for high speed. See [2] for more details on the experimental set-up and results.

6 Conclusion

In this chapter, we have presented a new approach for planning kino-dynamically feasible motions of quadrotor-like aerial robots. The approach relies on a computationally efficient method for generating local trajectories that interpolate two given states (position, velocity and acceleration) with a fourth-order spline respecting bounds on the derivatives up to snap. This local trajectory planner can be used in a decoupled approach, or applied directly as a steering method within sampling-based motion planners for solving more constrained problems. We also proposed two techniques to enhance the performance of kinodymanic motion planners based on directed variants of the RRT and PRM algorithms : a quasi-metric in the state space that accurately estimates the cost-to-go between two states and an incremental sampling technique that notably increases the probability of generating connectible states. Simulation results show that the integration of these techniques significantly improves the performance of the PRM-based and RRT-based planners, particularly for solving very constrained problems. While our first experiments performed with a real quadrotor indicate the ability to correctly follow the planned trajectory, further experimental work as well as improvements in the implemented techniques for localization and control remain to be done to reduce the tracking error observed at high speed and to better demonstrate the ability to execute more agile maneuvers in constrained environments. As future work, it would also be interesting to

investigate the integration of the techniques presented in this chapter within optimal sampling-based planners like T-RRT* [6] to compute minimal-time trajectories or solutions optimizing a cost function, for example in order to maximize the clearance to obstacles.

References

1. D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I. Athena Scientific, 2005.
2. A. Boeuf. Kinodynamic motion planning for quadrotor-like aerial robots. *PhD Toulouse University*, 2017.
3. A. Boeuf, J. Cortés, R. Alami, and T. Siméon. Planning agile motions for quadrotors in constrained environments. In *Proc. IEEE/RSJ IROS*, 2014.
4. A. Boeuf, J. Cortés, R. Alami, and T. Siméon. Enhancing sampling-based kinodynamic motion planning for quadrotors. In *Proc. IEEE/RSJ IROS*, 2015.
5. P. M. Bouffard and S. L. Waslander. A hybrid randomized/nonlinear programming technique for small aerial vehicle trajectory planning in 3d. In *Planning, Perception and Navigation for Intelligent Vehicles (PPNIV) 63, 2009*, 2009.
6. D. Devaurs, T. Siméon, and J. Cortés. Optimal path planning in complex cost spaces with sampling-based algorithms. *IEEE Transactions on Automation Science and Engineering, 13(2)*, pages 415–424, 2016.
7. Roland Geraerts and Mark H Overmars. Creating high-quality paths for motion planning. *The International Journal of Robotics Research*, 26(8):845–863, 2007.
8. Markus Hehn and Raffaello D'Andrea. Quadrocopter trajectory generation and control. *IFAC Proceedings Volumes*, 44(1):1485–1491, 2011.
9. L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
10. E. Koyuncu and G. Inalhan. A probabilistic b-spline motion planning algorithm for unmanned helicopters flying in dense 3d environments. In *Proc. IEEE/RSJ IROS 2008*, 2008.
11. S. LaValle and J. Kuffner. Rapidly-exploring random trees: progress and prospects. In *Algorithmic and Computational Robotics: New Directions*. A. K. Peters, Wellesley, MA, 2001.
12. S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research, 20(5)*, pages 378–400, 2001.
13. Steven LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.
14. Steven M LaValle and James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.
15. Taeyoung Lee, Melvin Leoky, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5420–5425. IEEE, 2010.
16. D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proc. IEEE ICRA*, 2011.
17. M. W. Mueller, M. Hehn, and R. D'Andrea. A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification. In *Proc. IEEE/RSJ IROS*, 2013.
18. C. Richter, A. Bry, and N. Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Proc. ISRR*, 2013.
19. Thierry Siméon, Jean-Paul Laumond, and Florent Lamiraux. Move3D: a generic platform for path planning. In *Proc. IEEE ISATP*, 2001.
20. M. Haddad Y. Bouktir and T. Chettibi. Trajectory planning for a quadrotor helicopter. In *Control and Automation, 2008, 16th Mediterranean Conference on*, 2008.