POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Planning and trajectory generation methods for aggressive maneuvering of UAVs in the presence of obstacles

Laurea Magistrale in Aeronautical Engineering - Ingegneria Aeronautica

Author: Marta Manzoni

Advisor: Prof. Davide Invernizzi

Co-advisor: Eng. Roberto Rubinacci

Academic year: 2023-2024

---

1. Introduction

Unmanned Aerial Vehicles (UAVs) showcase innovative engineering, enabling autonomous flight without human pilots. Their versatility manifests in autonomous navigation and task execution, spanning applications across civilian and military domains. As UAVs' roles evolve, navigating complex environments with agility becomes crucial. Consequently, path planning in obstacle-cluttered environments has attracted attention in recent years. While kinematic motion planning ensures collision-free paths, their execution by the actual system may not be feasible. This is especially evident in agile vehicles, where additional limitations from dynamics or nonholonomic constraints must be considered. Kinodynamic motion planning addresses this issue, considering vehicle dynamics, input constraints, and state constraints.

2. Quadrotor model

Following [2], the quadcopter is modeled as a rigid body with six degrees of freedom, including linear translation along the inertial axes and rotation of the body frame relative to the inertial frame, defined by the proper orthogonal matrix $\boldsymbol{R}$. Euler angles $\Phi = [\phi, \theta, \psi]$ are employed to define the roll, pitch, and yaw angles. ω denotes the angular velocity of the body frame with respect to the inertial frame. Thus, the quadrotor state vector x comprises the center of mass position and velocity, the Euler angles, and the angular velocity:

$$x = [p, v, \Phi, \omega]^T.$$

The control input is taken as $u = [f, m]^T$, where f is the mass-normalized net body force, and m represents the three body torques. The differential equations governing the flight are

$$\ddot{x} = \boldsymbol{R} e_3 f + g,$$
$$\dot{\boldsymbol{R}} = \boldsymbol{R}[\![\omega\times]\!],$$

where $g = [0, 0, -g]^T$ is the gravity acceleration, $e_3 = [0, 0, 1]^T$ and $[\![\omega\times]\!]$ is the skew-symmetric matrix form of the vector cross product.
The quadrotor is a differentially flat system. State x and input u can be expressed as functions of the flat output $\sigma = [p, \psi]^T$, and its derivatives. Differential flatness is a valuable property that can greatly simplify the planning problem.

3. Kinodynamic motion planning

Kinodynamic motion planning optimizes vehicle trajectories considering dynamics, ensuring collision avoidance, feasibility, and adherence to input constraints.

3.1. Problem formulation

The kinodynamic motion planning problem can be formulated as an optimization problem with the objective of finding input functions $u(t)$ and state trajectories $x(t)$ that minimize a cost function while adhering to state and input constraints:

$$\begin{aligned} \min_{x(t),u(t)} \; & J = \int_0^T g(x(t),u(t))dt \\ \text{s.t. } & \dot{x} = f(x(t),u(t)), \quad t \in [0,T] \\ & x(t) \in \mathcal{S}_{free}, \quad t \in [0,T] \\ & u(t) \in \mathcal{U}, \quad t \in [0,T] \\ & x(0) = x_i, \; x(T) = x_f. \end{aligned} \tag{1}$$

Let $\mathcal{S}_{free} \subset \mathcal{S}$ denote the free region of the state space, including obstacle-free configurations $\mathcal{C}_{free}$ and dynamical constraints $\mathcal{D}_{free}$. Thus, $\mathcal{S}_{free} = \mathcal{C}_{free} \times \mathcal{D}_{free}$.
This thesis aims to compute a trajectory balancing total time and trajectory effort, represented by the cost function:

$$J = J_e + J_T = \int_0^T \|u(t)\|^2 \, dt + \rho T, \tag{2}$$

where ρ is the weight governing the trade-off between effort and time.
For differentially flat systems, system dynamics need not be enforced as constraints. Moreover, the objective function can be expressed as a function of the flat outputs $J(\sigma, \dot{\sigma}, \ddot{\sigma}, \ldots)$ and the inputs are represented in terms of the q^{th} derivative of the position. Therefore, the control effort term J_e in Eq.(2) is reformulated to minimize the q^{th} derivative of the first three entries of the flat output:

$$J_e = \int_0^T (x^{(q)})^2 \, dt + \int_0^T (y^{(q)})^2 \, dt + \int_0^T (z^{(q)})^2 \, dt.$$

Different q values yield different trajectories: for $q = 1$, minimum velocity trajectory; for $q = 2$, acceleration; for $q = 3$, jerk; and for $q = 4$, snap.

3.2. Kinodynamic planning with motion primitives

Explicitly solving Problem 1 is hard due to states and input constraints, as well as the non-convex nature of obstacle avoidance. However, testing feasibility for a candidate solution is straightforward. This paradigm shift is the basis for many motion planning algorithms. The building blocks of these candidate solutions are called motion primitives.

4. Search-based planning with online motion primitives

This section introduces a quadrotor-tailored search-based planner using motion primitives computed via a forward propagation method, inspired by the work proposed in [1].

4.1. Motion primitives generation

Leveraging differential flatness, position can be expressed as a polynomial:

$$p(t) = c_k \frac{t^k}{k!} + \ldots + c_1 t + c_0 \in \mathbb{R}^3, \tag{3}$$

where $C = [c_0, \ldots, c_k] \in \mathbb{R}^{3\times(k+1)}$. Polynomial trajectories of the form Eq.(3) can be generated by considering a linear time-invariant system:

$$\begin{aligned} \dot{x} &= Ax + Bu, \\ \dot{x} &= \begin{bmatrix} 0 & I_3 & 0 & \ldots & 0 \\ 0 & 0 & I_3 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & \ldots & 0 & I_3 \\ 0 & \ldots & \ldots & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ I_3 \end{bmatrix} u. \end{aligned} \tag{4}$$

Following [1], a discretization $\mathcal{U}_\mathcal{L} = \{u_1, \ldots, u_L\}$ of the control input set $\mathcal{U} = [-u_{max}, u_{max}]$ is considered, where each control $u_l \in \mathbb{R}^3$ defines a short-duration motion. Then, a motion primitive for system in Eq.(4) is generated by applying a constant control input $u_l \in \mathcal{U}_\mathcal{L}$ to an initial state $x_i = [p_i^T, v_i^T, a_i^T, \ldots]^T$, for a duration τ. Integration of the control u_l from an initial condition x_i results in:

$$p(t) = u_l \frac{t^n}{n!} + \ldots + a_i \frac{t^2}{2} + v_i t + p_i.$$

Equivalently, the resulting trajectory of the linear time-invariant system in Eq.(4) is:

$$x(t) = e^{At}x_i + \left[\int_0^t e^{A(t-\beta)}B\,d\beta\right]u_l$$
$$= F(t)x_i + G(t)u_l.$$

Motion primitives generation occurs independently for each of the three spatial axes. Input and state constraints are excluded initially; compliance must be assessed after the motion primitive computation. The total cost of the motion primitive is $J = (\|u_l\|^2 + \rho)\tau$.
Fig. 1 shows an example of motion primitives generated through this approach.

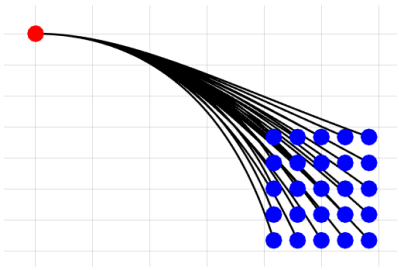


Figure 1: 2D motion primitives for a jerk-controlled system from an initial state (red dot).

4.2. Graph construction

Motion primitives discretize the state space, enabling the construction of a graph representation $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Here, $\mathcal{V}$ is the set of states and $\mathcal{E}$ is the set of edges, each defined by a motion primitive. The states in $\mathcal{V}$ are generated by iterative application of each control input $u(t) \in \mathcal{U}_L$ for a duration τ to each state.

4.3. Kinodynamic motion planning

For quadrotors, trajectory effort is expressed as:

$$J(C) = \int_0^T \|u(t)\|^2\,dt = \int_0^T \|p^{(q)}(t)\|^2\,dt.$$

Thus, the motion planning problem arises from the reformulation of Problem 1 and the incorporation of the dynamics in Eq.(4):

$$\min_{C,T}\ J(C) + \rho T$$
$$\text{s.t.}\quad \dot{x}(t) = Ax(t) + Bu(t), \quad t \in [0,T] \qquad (5)$$
$$x(t) \in \mathcal{S}_{free},\ u(t) \in \mathcal{U}, \quad t \in [0,T]$$
$$x(0) = x_i,\ x(\tau) \in \mathcal{S}_{goal}.$$

Here, the objective is to find a polynomial trajectory parametrization D and a time T.
Motion primitives allow to convert Problem 5 to a graph-search problem. This can be done by treating the control as a piecewise constant over intervals of duration τ: $u(t) = \sum_{k=0}^{N-1} u_k$. Thus, the graph-search problem is formulated as:

$$\min_{u_k} \left(\sum_{k=0}^{N-1} \|u_k\|^2 + \rho N\right)\tau$$
$$\text{s.t.}\ \ x_{k+1}(t) = F(t)x_k(t) + G(t)u_k, \quad \forall\, k,\ t \in [0,\tau]$$
$$x_{k+1}(t) \in \mathcal{S}_{free}, \quad \forall\, k,\ t \in [0,\tau]$$
$$u_k \in \mathcal{U}_L, \quad \forall\, k$$
$$x_{k+1}(0) = x_k(\tau), \quad \forall\, k$$
$$x_0(0) = x_i,\ x_N(\tau) \in \mathcal{S}_{goal},$$

where $k = 0, \ldots, N-1$.
The A* algorithm is used to solve the graph search problem. A* employs the `GetSuccessors` procedure in Algorithm 1 to explore the free state space and construct the graph. In essence, for each control $u_l \in \mathcal{U}_L$, a motion primitive is computed by applying u_l to the current node v for a duration τ. The dynamic feasibility of the trajectory is assessed and, if feasible, the successor node is added to the list along with the cost.

Algorithm 1 Given node v and the discretized control set $\mathcal{U}_L$, find the successors of v Succ(v) and their cost Cost(v).

function *GetSuccessors*(v, $\mathcal{U}_L$,τ)
Succ(v) $\leftarrow \emptyset$, Cost(v) $\leftarrow \emptyset$;
for all $u_l \in \mathcal{U}_L$ **do**
  pr $\leftarrow$ *GeneratePrimitive*(v,u_l,τ);
  if *isDynamicallyFeasible*(pr) **then**
    $s_f \leftarrow$ pr(τ);
    Succ(v) $\leftarrow$ Succ(v) $\cup$ $\{s_f\}$;
    Cost(v) $\leftarrow$ Cost(v) $\cup$ $\{(\|u_l\|^2 + \rho)\tau\}$;
  end if
end for

4.4. Numerical example

Consider a double integrator-modeled quadrotor tasked with navigating from the initial state $(x, y, v_x, v_y) = (2, 1, 0, 0)$ to the final state $(x, y, v_x, v_y) = (38, 7, 0, 0)$ in a 2D $40m \times 10m$ virtual environment. Velocity is constrained to $v_x, v_y \in [-2, 2]m/s$, while the set of control inputs is discretized with nine options within the allowable range of $u_x, u_y \in [-2, 2]m/s^2$.
In Fig. 2, the red trajectory represents the optimal collision-free trajectory guiding the vehicle to the target state. Fig. 3 shows the velocity

profile for the optimal trajectory, demonstrating adherence to velocity constraints.

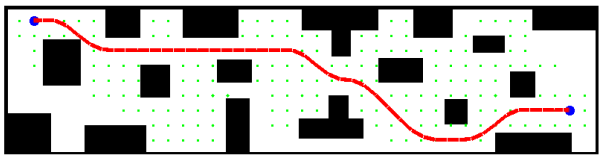


Figure 2: Simulation for the acceleration-controlled system. Blue dots: initial and final positions. Red curve: optimal trajectory. Green dots: expanded nodes.

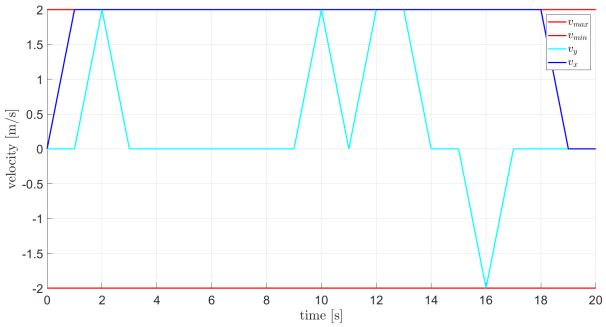


Figure 3: Velocity profile for the optimal trajectory in Fig.2.

5. Search-based planning with motion primitives library

In Section 4, motion primitives result from discretization of the control input. Here, a state space discretization is exploited to create a motion primitive library, as proposed in [3].

5.1. Motion primitives library

Given an initial state x_i and a final state x_f, a motion primitive is the solution to Two-Point-Boundary-Value-Problem (TPBVP) in Eq.(1). Solving this problem for a nonlinear system of form $\dot{x}(t) = f(x(t), u(t))$ is computationally challenging, requiring nonlinear solvers. To reduce the online computational load, a motion primitives library is introduced, moving computational challenges to the offline phase.
Following [3], the database of motion primitives is generated offline by solving Problem 1 for a suitable number of boundary conditions $(x_i^k, x_f^k), k = 1, \dots, N$, obtained by uniformly gridding the continuous state space. The resulting optimal trajectories $x_k^*(t)$, control inputs $u_k^*(t)$, durations τ_k^*, and costs C_k^* are stored in a Look-Up Table (LUT). Subsequently, the database is repeatedly used online by the planner. When the planner requires a trajectory connecting two nodes, it selects a suitable motion primitive from the library.
Motion primitives create a finite lattice discretization within the state space, representing precomputed feasible maneuvers for the system. To optimize planning, the planner's search space is uniformly gridded as the region where motion primitives are built.

Invariance properties

Dynamic systems, featuring translation invariance, maintain consistent behavior under coordinate translations. This implies that optimal primitives remain the same regardless of the coordinate frame.
Certain systems, like quadrotors, exhibit robust invariance properties, being invariant to horizontal plane translations and rotations about the vertical axis. Additionally, quadrotor motion primitives are symmetric with respect to both the x- and y-axes. Leveraging these properties enables the reduction of the memory required to store the database.

Numerical example

Consider a simplified quadrotor model that includes position and velocity along with a 2D actuation space $(a_x, a_y) = (u_x, u_y)$ representing linear acceleration. For this system, a third-order polynomial representation is considered. The library of motion primitives is computed by solving the following TPBVP for every combination of initial and final states.

$$
\begin{aligned}
&\underset{C_i, B_i, i=1:4}{\text{minimize}} \int_0^\tau \|u(t)\|^2 \, dt + \rho\tau \\
\text{s.t. } & x(t) = \frac{C_1}{6}t^3 + \frac{C_2}{2}t^2 + C_3 t + C_4, \\
& y(t) = \frac{B_1}{6}t^3 + \frac{B_2}{2}t^2 + B_3 t + B_4, \\
& \|v(t)\| \in [-1.5\sqrt{2}, 1.5\sqrt{2}], \, t \in [0, \tau] \\
& \|a(t)\| \in [-4.5\sqrt{2}, 4.5\sqrt{2}], \, t \in [0, \tau] \\
& x(0) = x_i, \; y(0) = y_i, \\
& v_x(0) = v_{xi}, \; v_y(0) = v_{yi}, \\
& x(\tau) = x_f, \; y(\tau) = y_f, \\
& v_x(\tau) = v_{xf}, \; v_y(\tau) = v_{yf}.
\end{aligned} \tag{6}
$$

Here, total velocity and total acceleration (control input) are defined as $\|v\| = \sqrt{v_x^2 + v_y^2}$ and $\|u\| = \|a\| = \sqrt{a_x^2 + a_y^2}$, respectively. Velocities (v_x, v_y) and accelerations (a_x, a_y) along each axis are obtained as the first and second derivatives of the positions (x, y). Finally, Problem 6 is solved using MATLAB's `fmincon` function.

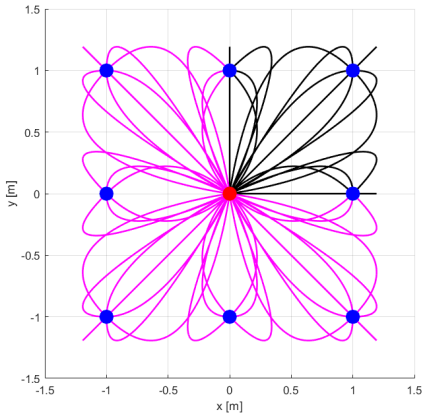
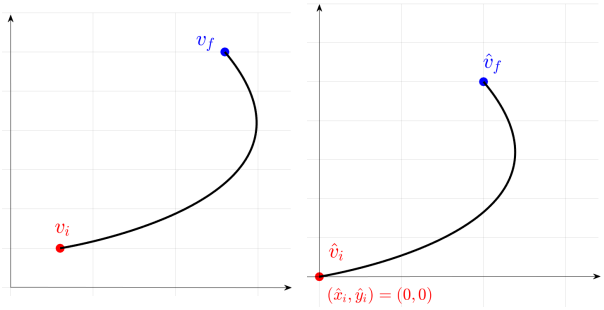


Figure 4: Motion primitives subset. Red dot: initial state. Blue dots: final states. Black lines: trajectories for various final velocities.

Fig. 4 shows a subset of these motion primitives originating from the steady state $x_i = y_i = v_{xi} = v_{yi} = 0$. Due to symmetry property, only trajectories in the first quadrant (black curves) are stored in the database; others (pink curves) can be generated through mirroring. Furthermore, translation invariance allows maintaining a small database while covering the entire vehicle operating space. In practice, the initial position can be set as $(\hat{x}_i, \hat{y}_i) = (0, 0)$ during the database construction. Subsequently, the motion primitives can be translated to match any other initial position (x_i, y_i), as shown in Fig. 5.



(a) original nodes. (b) translated nodes.

Figure 5: Steps involved in the translation.

5.2. Kinodynamic motion planning

The A* algorithm is used for planning, employing the `GetSuccessorsLibrary` procedure in Algorithm 2 to explore the state space and build the graph. When provided with an initial state and a final state, the `GetSuccessorsLibrary` procedure is employed to query the database of motion primitives. This process involves a sequence of translations. Moreover, the chosen primitives must ensure continuity of all state variables at each node.

Algorithm 2 Given node v and database $\mathcal{L}$, including initial states s_i, final states s_f, primitives pr, and effort costs c, find the set of successors Succ(v) and their cost Cost(v).

function *GetSuccessorsLibrary*$(v, \mathcal{L})$
Succ$(v) \leftarrow \emptyset$, Cost$(v) \leftarrow \emptyset$;
for all $s_i, s_f \in \mathcal{L}$ **do**
  $\mathcal{L}$.*isContinuous*$(v, s_i\ \mathcal{L})$;
  $\mathcal{L}$.*Translate*$(v, s_i, s_f\ \mathcal{L})$;
end for
for all pr $\in \mathcal{L}$ **do**
  if *isCollisionFree*(pr) **then**
    Succ$(v) \leftarrow$ Succ$(v) \cup \{s_f\}$;
    Cost$(v) \leftarrow$ Cost$(v) \cup \{c + \rho\tau\}$;
  end if
end for

6. Experimental results

This section demonstrates the effectiveness of the approach introduced in Section 5 for navigating real-world cluttered environments. Experiments are carried out with the ANT-X quadrotor in the Aerospace Systems and Control Laboratory at Politecnico di Milano (Fig.6).

Figure 6: Quadrotor and indoor environment used for the experiments.

In the experiment, a quadrotor operating in a $10m \times 4m$ environment featuring two obstacles (Fig.6) is considered. The starting position is set at $(-3.5, 0.5, 1)$, and the goal re-

gion is a square with a side length of $0.5m$, centered at $(3, -0.5, 1)$. The simplified quadrotor model considered includes position, velocity and acceleration along with a 2D actuation space $(j_x, j_y) = (u_x, u_y)$ representing linear jerk. A fifth-order polynomial representation is employed for this system, and the database is computed by solving a TPBVP derived from Problem 6 considering such parametrization. Total velocity, acceleration, and jerk limits are set at $v_{max} = 1.5\sqrt{2}m/s$, $a_{max} = 4.5\sqrt{2}m/s^2$ and $j_{max} = 15\sqrt{2}m/s^3$. Database construction starts at $(x_i, y_i) = (0, 0)$ and is based on a uniform square grid, with final state coordinates in $(x_f, y_f) = [-2, 0) \cup (0, 2] \times [-2, 0) \cup (0, 2]$. Initial and final velocities and accelerations are chosen from sets $\{-1.5, 0, 1.5\}m/s$ and $\{-4.5, 0, 4.5\}m/s$, respectively.
Fig. 7 shows the planned trajectory, while Fig. 8 shows the position and velocity profiles with their setpoints, demonstrating precise tracking.

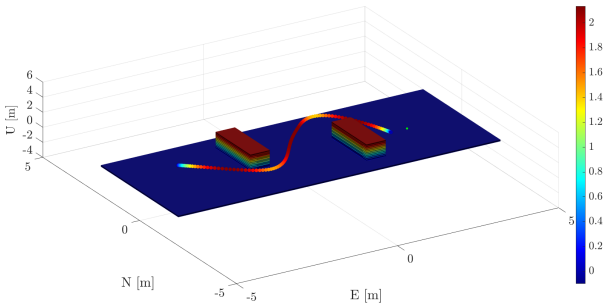


Figure 7: Planned trajectory.

7. Discussion

The first method computes motion primitives online, making it suitable only for simple systems. The second method extends motion planning to any dynamic system by introducing a database. The first method employs fixed-duration primitives, hindering minimum-time trajectory generation. In contrast, the second method optimizes the duration of the primitives. Finally, the analytical solutions used by the first method lack information on dynamic constraints, requiring a subsequent feasibility check. The second method integrates dynamic constraints directly into the database.

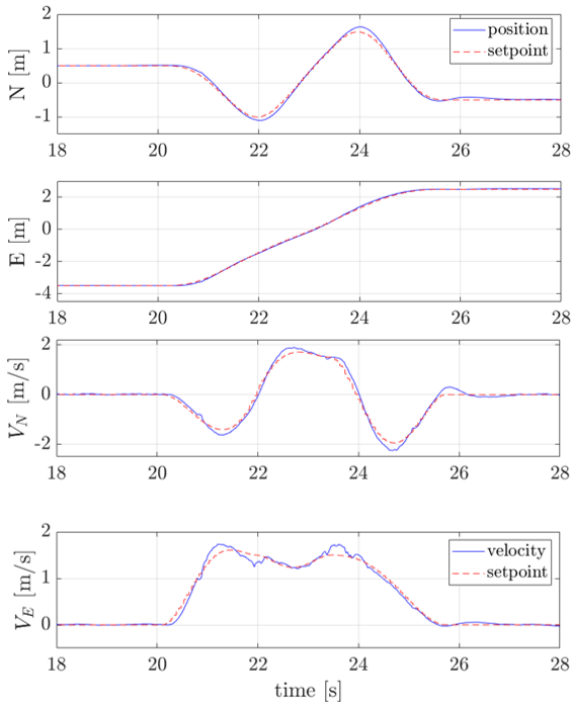


Figure 8: Position and velocity setpoints (dashed) compared to tracked profiles (solid).

8. Conclusions

This thesis addresses UAV trajectory planning in cluttered environments by implementing two methods. The first, a quadrotor-tailored search-based planner, excels in kinodynamic motion planning for quadrotors. The second method introduces a database of precomputed solutions, demonstrating applicability to arbitrary dynamical systems. The experimental results reveal the success of using this approach as the basis for safe and fast navigation. In conclusion, both methods demonstrate ability to generate complete, collision-free, resolution-optimal, and dynamically feasible trajectories. Moreover, this generic framework can be integrated with other path-planning techniques, such as sampling-based methods.

Prospective works

Future work will focus on addressing motion planning in unknown and dynamic environments. This involves dynamic re-planning using real-time data. Additionally, to address modeling errors and improve tracking, Iterative Learning Control (ILC) will be considered. This involves a learning phase in real flights to build the database.

References

[1] S. Liu, N. Atanasov, K. Mohta, and V. Kumar. Search-based motion planning for quadrotors using linear quadratic minimum time control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2872–2879, 9 2017.

[2] M. W. Mueller, M. Hehn, and R. D'Andrea. A computationally efficient motion primitive for quadrocopter trajectory generation. *IEEE Transactions on Robotics*, 31(6):1294–1310, 7 2015.

[3] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini. Sampling-based optimal kinodynamic planning with motion primitives. *Autonomous Robots*, 43(7):1715–1732, 2019.