Time-optimal trajectory planning for landing onto moving platforms

Botao Hu
Graduate Research Assistant
Rensselaer Polytechnic Institute
Troy, NY, USA

Sandipan Mishra
Associate Professor
Rensselaer Polytechnic Institute
Troy, NY, USA

ABSTRACT

In this paper, an algorithm for time-optimal trajectory generation is developed for landing a 6 degree-of-freedom (DOF) quadrotor onto a moving platform (with tilt, heave and pitch). The overall control architecture has a standard guidance-and-tracking control inner-outer loop structure. The outer loop (guidance control) solves the time-optimal trajectory generation problem. Instead of directly solving the time-optimal control problem, the proposed method reformulates this into a nonlinear programming problem that transforms the constraints on the original system dynamics and inputs onto constraints on the system states. This transformation is based on the differential flatness property of the quadrotor dynamics. The proposed method is computationally efficient and can also incorporate the collision avoidance constraints. We further demonstrate that this time-optimal problem can be resolved at periodic intervals (if disturbances and unmodeled dynamics deviate the quadrotor from the optimal trajectory). For the inner loop, a trajectory tracking controller is also designed that can deal with system uncertainties and external disturbances that may affect the quadrotor's dynamics. Simulation and experimental results show the effectiveness of the proposed method.

INTRODUCTION

There has been growing interest in Vertical Taking Off and Landing (VTOL) autonomous aerial vehicles (AAVs) for a variety of applications, including aerial imaging, surveillance, and law enforcement. One important capability for such AAVs is autonomous landing onto either a stationary or moving platform (Refs. 1–4). However, autonomous landing onto a platform (such as a ship deck on sea or a moving target) is often difficult because of the stringent constraints on safety, collision avoidance, soft contact requirements, and limited landing time. Therefore, a proper landing trajectory that minimizes the total time-to-land and satisfies the safety constraints should be generated in real-time prior to executing such a maneuver.

In our earlier work, (Refs. 5,6), we demonstrated the design of such time-optimal landing paths when the platform was horizontal and only translated in the vertical direction. (Ref. 7) presented an algorithm for time-optimal trajectory design that exploited decoupled dynamics in the x and z directions. However, if the platform is tilting in addition to movement in the horizontal and vertical directions, the trajectory generation problem is more challenging because the horizontal and vertical dynamics are coupled and thus the decoupled trajectory generation approach (as in (Ref. 7)) is not applicable.

In this paper, we address the more general trajectory generation problem to generate a time-optimal trajectory for a 6-DOF quadrotor landing onto a 3-DOF platform (with the heave, pitch and roll, which are selected because they are typically more important than the lateral motion and the yaw motion).

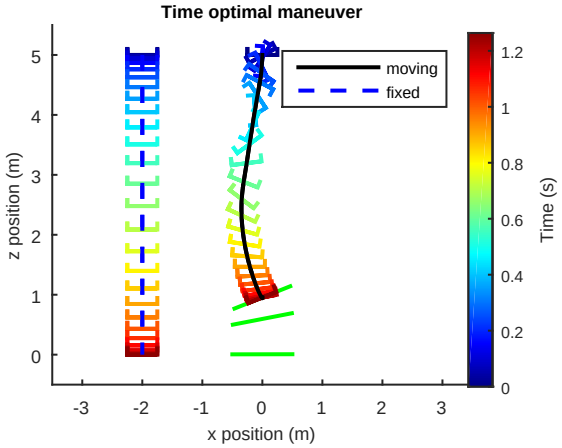


Fig. 1: Trajectories for time-optimal landing onto stationary and moving platforms. The trajectory on the left illustrates landing onto a fixed platform. The trajectory on the right illustrates landing onto an oscillating and tilting platform. The green bar represents the platform position.

In general, an optimal control problem must be solved to obtain the time-optimal landing trajectory. One approach towards this goal is to directly solve the optimal control problem. This approach, as demonstrated in (Refs. 8, 9), is time-

Presented at the AHS International 73rd Annual Forum & Technology Display, Fort Worth, Texas, USA, May 9–11, 2017.

consuming because the quadrotor dynamics are nonlinear and furthermore it is difficult to incorporate collision avoidance constraints into the optimal control problem. Another approach using model predictive control was proposed by Ngo (Ref. 3). The authors demonstrated the performance of the algorithm through simulation of a full-scale helicopter model landing onto a ship deck. The algorithm used a surrogate cost to penalize the time-to-land, instead of directly penalizing time-to-land. The time-optimality of the algorithm was thus not verifiable.

In contrast, (Ref. 10) used the differential flatness property of the quadrotor dynamics to transform the original optimal control problem into a nonlinear programming problem and generate a point-to-point time-optimal maneuver trajectory. This trajectory generation method was shown to be computationally efficient and could incorporate the obstacle avoidance constraints. This approach provides an elegant mechanism for transforming the constraints on the system input and dynamics onto the system states, which ultimately decreases the computational complexity of the problem through suitable parameterization. However, this algorithm did not account for *time-varying platform motion* as a terminal state constraint and thus is not directly applicable for landing onto a moving platform.

However, inspired by (Ref. 10), this paper develops a time-optimal landing trajectory generation method for a 6-DOF quadrotor landing onto a heaving, rolling and pitching platform. The general idea behind the proposed method is to formulate a nonlinear programming problem that uses the *system states* as the optimization variables (instead of the input) using differential flatness. The advantages of the proposed the trajectory generation method are that it can directly incorporate the time-varying motion of the platform and collision avoidance constraints during the landing maneuver. This paper is an extension of the method presented in our recent submission (Ref. 11), where a simplified quadrotor model (3-DOF) was used to generate a time-optimal landing trajectory (Fig. 1 shows the time-optimal trajectories for landing onto a stationary platform and a heaving and pitching 2-DOF platform based on method presented in (Ref. 11)). We further demonstrate that the proposed method can be used in real-time (and iteratively be solved at each time-step in a model predictive manner since the computation times are sufficiently small).

We first present the 6-DOF quadrotor dynamics model and formulate the time-optimal trajectory generation problem. We then describe a mechanism to reformulate this problem using differential flatness into a computationally efficient structure. A tracking (inner loop) controller is then proposed for tracking this time-optimal trajectory. Simulation and experimental results are then presented to validate the proposed algorithms. Finally, we conclude with challenges, future directions, and extensions for this work.

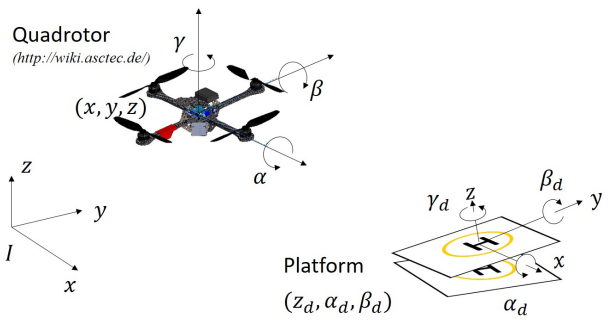


Fig. 2: Schematic of the quadrotor and the platform prior to landing. The quadrotor has six degrees of freedom and the platform has three degrees of freedom.

DYNAMICAL MODEL AND PROBLEM FORMULATION

In this section, we first present the 6-DOF system dynamics model for a quadrotor and then formulate the time-optimal landing problem as a constrained optimization problem.

6-DOF Quadrotor model

We model the quadrotor as a 6-DOF rigid body, as shown in Fig. 2. Denote the inertial frame as $\mathbb{I}$, the body frame as $\mathbb{B}$; and the system state as $\boldsymbol{x} = (\boldsymbol{p}^T, \dot{\boldsymbol{p}}^T, \boldsymbol{\theta}^T, \boldsymbol{\Omega}^T)^T$, where the vector $\boldsymbol{p} = (x,y,z)^T$ is the position and $\dot{\boldsymbol{p}} = (\dot{x},\dot{y},\dot{z})^T$ is the velocity of the quadrotor in the inertial frame. The vector $\boldsymbol{\theta} = (\alpha,\beta,\gamma)^T$ contains the roll, pitch and yaw Euler angles. The vector $\boldsymbol{\Omega} = (p,q,r)^T$ is the angular velocity about the body frame bases. The system input is denoted as $\boldsymbol{u} = (T,\tau_x,\tau_y,\tau_z)^T$ where T is the thrust and τ_x,τ_y,τ_z are the torques around the body frame bases. The dynamics of the system can be described by:

$$\begin{aligned} m\ddot{\boldsymbol{p}} &= R(\alpha,\beta,\gamma)\begin{bmatrix}0\\0\\T\end{bmatrix} - \begin{bmatrix}0\\0\\mg\end{bmatrix} \\ I\dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega}\times I\boldsymbol{\Omega} + \begin{bmatrix}\tau_x\\\tau_y\\\tau_z\end{bmatrix}, \end{aligned} \tag{1}$$

where m is the mass of the quadrotor, g is the gravitational acceleration and I is the inertia matrix. $R(\alpha,\beta,\gamma)$, the rotation matrix corresponding to the body frame $\mathbb{B}$ with respect to the inertial frame $\mathbb{I}$, is a function of the Euler angles. The rotation is described on a Z-Y-X ordering. The operator '$\times$' denotes the cross product. The angular velocity $\boldsymbol{\Omega} = (p,q,r)^T$ can be represented as a function of the first order derivatives of the Euler angles $\dot{\boldsymbol{\theta}} = (\dot{\alpha},\dot{\beta},\dot{\gamma})^T$. The rotation matrix R and the angular velocity $\boldsymbol{\Omega}$ can be represented as:

$$\begin{aligned} R &= \begin{bmatrix} c\beta c\gamma & c\gamma s\alpha s\beta - c\alpha s\gamma & s\alpha s\gamma + c\alpha c\gamma s\beta \\ c\beta s\gamma & c\alpha c\gamma + s\alpha s\beta s\gamma & c\alpha s\beta s\gamma - c\gamma s\alpha \\ -s\beta & c\beta s\alpha & c\alpha c\beta \end{bmatrix} \\ \boldsymbol{\Omega} &= \begin{bmatrix} 1 & 0 & -s\beta \\ 0 & c\alpha & c\beta s\alpha \\ 0 & -s\alpha & c\alpha c\beta \end{bmatrix}\dot{\boldsymbol{\theta}}. \end{aligned} \tag{2}$$

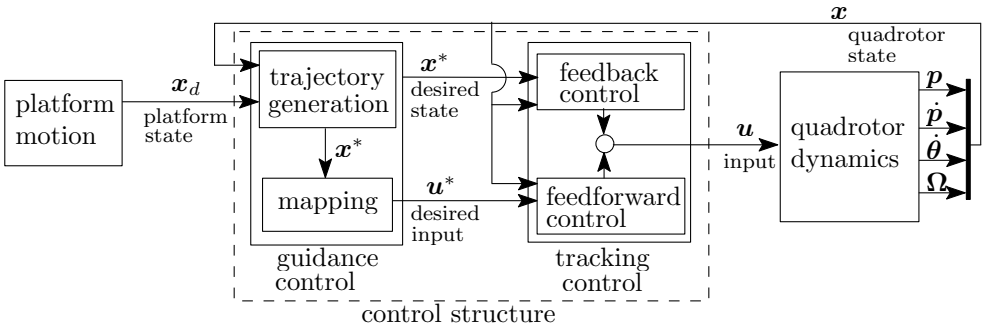


Fig. 3: Schematic of the proposed control architecture. The control architecture consists of two modules in the dashed box. The guidance control module outputs the reference trajectory and the tracking control module generates the thrust command for the quadrotor.

'c' and 's' are shorthand forms of cosine and sine functions. All the Euler angles are constrained to a feasible set. The constraints on the Euler angles can be represented as:

$$\begin{bmatrix} \alpha_{\min} \\ \beta_{\min} \\ \gamma_{\min} \end{bmatrix} \leq \boldsymbol{\theta} \leq \begin{bmatrix} \alpha_{\max} \\ \beta_{\max} \\ \gamma_{\max} \end{bmatrix} . \tag{3}$$

Control architecture

Fig. 3 shows the cascade two-loop structure of the proposed algorithm. The outer loop is the guidance control where the time-optimal landing trajectory $\boldsymbol{x}^*$ is first generated based on the platform state $\boldsymbol{x}_d$ and the quadrotor state $\boldsymbol{x}$. Then the generated optimal reference trajectory is used to obtain a reference input $\boldsymbol{u}^*$. The inner loop is a standard Model Reference Adaptive Controller (MRAC) (Ref. 12), which takes in the reference trajectory $\boldsymbol{x}^*$ and the state of the quadrotor $\boldsymbol{x}$ and generates a control command $\boldsymbol{u}$ that achieves robust trajectory tracking performance.

Problem formulation

We first focus on the design of the trajectory generation block (outer loop). The objective of this trajectory generation block is to create an optimal trajectory for the quadrotor to land onto a translating and tilting platform (with a known but time-dependent trajectory) within the smallest possible time, t_f.

Denote $\boldsymbol{x}_d(\cdot) = (\boldsymbol{p}_d^T, \dot{\boldsymbol{p}}_d^T, \boldsymbol{\theta}_d^T, \boldsymbol{\Omega}_d^T)^T \in \Re^{12}$ as the state of the platform. $\boldsymbol{p}_d = (x_d, y_d, z_d)^T$ is the platform position, $\boldsymbol{\theta}_d = (\alpha_d, \beta_d, \gamma_d)^T$ as the Euler angles of the platform and $\boldsymbol{\Omega}_d = (p_d, q_d, r_d)^T$ as the angular velocity of the platform. Let t_f be the final (as yet unknown) time, $\boldsymbol{x}_0$ as the initial state for the quadrotor. Denote $\boldsymbol{x}_d(t_f)$ as the state of the platform at the final time. The time-optimal quadrotor landing problem can be formulated as an optimal control problem as (4):

$$\begin{aligned} \underset{\boldsymbol{u}(t), t_f}{\arg\min} \quad & J = t_f \\ s.t \quad & \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}), \quad \boldsymbol{x} \in \Re^6 \\ & \boldsymbol{x}(t=0) = \boldsymbol{x}_0 \\ & \boldsymbol{x}(t=t_f) = \boldsymbol{x}_d(t_f) \in \Re^6 \\ & \boldsymbol{g}(\boldsymbol{x}(t)) \in \boldsymbol{G}, \quad \forall t \in [0, t_f] \\ & \boldsymbol{u}_{\min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{\max}, \quad \forall t \in [0, t_f] \end{aligned} \tag{4}$$

$\boldsymbol{g}(\boldsymbol{x})$ captures the (safety and feasibility) constraints including the Euler angle limits, obstacle avoidance constraints and the collision avoidance constraints. The exact definitions of the constraints will be introduced in the following section.

The problem described above is a standard time-optimal control formulation. Direct solution of this problem is typically time-consuming, as was pointed out in (Ref. 7). Therefore, we reformulate the problem into a nonlinear programming problem that is computationally tractable for real-time implementation, as follows.

TIME-OPTIMAL TRAJECTORY GENERATION ALGORITHM

In this section, we describe the time-optimal trajectory generation algorithm and a force/torque mapping scheme that generates the ideal (feedforward) input, based on the differential flatness property of the quadrotor dynamics.

Differentially flat systems

We briefly recall the differential flatness property of a dynamical system. A detailed explanation of differentially flat systems can be found in (Ref. 13).

Definition: Given a system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$, a flat output $\boldsymbol{y}$ is defined as a function of the derivatives of the state and system input:

$$\boldsymbol{y} = \phi(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, \cdots, \boldsymbol{u}^{(i)}). \tag{5}$$

The system is called *differentially flat* if the system state and system input can be represented by the derivatives of the system flat output, i.e.,

$$\begin{aligned} \boldsymbol{x} &= \phi_x(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}, \cdots, \boldsymbol{y}^{(j)}) \\ \boldsymbol{u} &= \phi_u(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}, \cdots, \boldsymbol{y}^{(j-1)}). \end{aligned} \tag{6}$$

It is straightforward to show that the dynamics for this 6-DOF quadrotor model represented in (1) is indeed differentially flat (Ref. 14) . Choosing the flat output as $\boldsymbol{y} = (\boldsymbol{p}^T, \boldsymbol{\Omega}^T)^T$, the input for this system can be represented as a function of derivatives of the flat output as shown below.

Denote $\boldsymbol{x}_a$ as the *augmented* system state and it is defined as:

$$\boldsymbol{x}_a = (\boldsymbol{p}^T, \dot{\boldsymbol{p}}^T, \boldsymbol{\theta}^T, \dot{\boldsymbol{\theta}}^T, \ddot{\boldsymbol{p}}^T, \boldsymbol{\Omega}^T, \dot{\boldsymbol{\Omega}}^T)^T \tag{7}$$

$\boldsymbol{x}_a$ is a vector of flat output and derivatives. The inputs for the system can be represented by function $\psi_1(\boldsymbol{x})$ and $\psi_2(\boldsymbol{x})$ as:

$$\begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} = \psi_1(\boldsymbol{x}_a) = R^{-1}(m\ddot{\boldsymbol{p}} + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}) \\ \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \psi_2(\boldsymbol{x}_a) = I\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times I\boldsymbol{\Omega} \tag{8}$$

The inputs, namely $\psi_1(\boldsymbol{x}_a)$, $\psi_2(\boldsymbol{x}_a)$ are constrained to their feasible ranges respectively. The constraints on the input $\boldsymbol{u}$ can thus be represented as :

$$\begin{bmatrix} 0 \\ 0 \\ T_{\min} \end{bmatrix} \leq \psi_1(\boldsymbol{x}_a) \leq \begin{bmatrix} 0 \\ 0 \\ T_{\max} \end{bmatrix} \\ \begin{bmatrix} \tau_{x\min} \\ \tau_{y\min} \\ \tau_{z\min} \end{bmatrix} \leq \psi_2(\boldsymbol{x}_a) \leq \begin{bmatrix} \tau_{x\max} \\ \tau_{y\max} \\ \tau_{z\max} \end{bmatrix} \tag{9}$$

Remark: The key benefit of using the differentially flat property is that we can avoid the integration of the nonlinear dynamics, as mentioned in (Ref. 10).

Trajectory generation

Based on the new constraints (9) extracted above, we now describe time-optimal trajectory generation algorithm. The idea is to reformulate the original optimal control problem described in (4) into a nonlinear programming problem that is more amenable to numerical solution in real time on-the-fly.

Cost function For time-optimallanding, the objective function to be minimized is the total time to land. The cost function is thus defined same as defined in (4) as: min $J = t_f$.

Optimization variables Although the original system dynamics (1) for optimization are continuous, for numerical tractability, the system dynamics are discretized into N segments. The number of segments, N, is chosen and fixed *a priori*. A proper selection of N is the result of the trade-off between the accuracy and the computational time necessary for the numerical solution of the problem. In the *RESULTS* section of this paper, a simulation result that shows the trade-off between the accuracy and the computational time is presented and a proper number of segments are selected. The sampling time t_s for each segment can be calculated from the total time:

$$t_s = \frac{t_f}{N-1} \tag{10}$$

The discretized state variable X_k represents the state variables at k-th sampling time. The elements in X_k are:

$$X_k = (\boldsymbol{p}_k^T, \dot{\boldsymbol{p}}_k^T, \ddot{\boldsymbol{p}}_k^T, \boldsymbol{\theta}_k^T, \dot{\boldsymbol{\theta}}_k^T, \boldsymbol{\Omega}_k^T, \dot{\boldsymbol{\Omega}}_k^T)^T \in \Re^{21} \tag{11}$$

Optimization constraints The constraints in general may be classified into two categories, namely, the discretized kinematics constraints and system-level constraints. The latter set of constraints may vary for different cases and typically include collision avoidance, actuator limits, state bounds, etc. Here, we present formulations of typical constraints.

1. Discretized kinematics constraints: based on the system discretization, the discretized state variable $X(k)$ must satisfy the following constraints:

$$\begin{bmatrix} \boldsymbol{p}_{k+1} \\ \dot{\boldsymbol{p}}_{k+1} \\ \boldsymbol{\theta}_{k+1} \\ \boldsymbol{\Omega}_{k+1} \end{bmatrix} - \begin{bmatrix} \boldsymbol{p}_k + \dot{\boldsymbol{p}}_k t_s + \ddot{\boldsymbol{p}}_k \frac{t_s^2}{2} \\ \dot{\boldsymbol{p}}_k + \ddot{\boldsymbol{p}}_k t_s \\ \boldsymbol{\theta}_k + \dot{\boldsymbol{\theta}}_k t_s \\ \boldsymbol{\Omega}_k + \dot{\boldsymbol{\Omega}}_k t_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_3 \\ \boldsymbol{0}_3 \\ \boldsymbol{0}_3 \\ \boldsymbol{0}_3 \end{bmatrix} \\ \forall k = 1, 2, \cdots, N-1 \tag{12}$$

2. System input constraints: based on (9), the constraints on $\boldsymbol{u}$ can be expressed as constraints on the system states:

$$\begin{bmatrix} 0 \\ 0 \\ T_{\min} \end{bmatrix} \leq \psi_1(X_k) \leq \begin{bmatrix} 0 \\ 0 \\ T_{\max} \end{bmatrix} \\ \begin{bmatrix} \tau_{x\min} \\ \tau_{y\min} \\ \tau_{z\min} \end{bmatrix} \leq \psi_2(X_k) \leq \begin{bmatrix} \tau_{x\max} \\ \tau_{y\max} \\ \tau_{z\max} \end{bmatrix} \\ \forall k = 1, 2, \cdots, N \tag{13}$$

3. Initial and final state constraints: the initial state and the final state of the quadrotor must match the given initial state $\boldsymbol{x}_0$ and the platform final state $\boldsymbol{x}_d(t_f)$ respectively. The constraints that enforce these are:

$$\begin{bmatrix} \boldsymbol{p}_1^T & \dot{\boldsymbol{p}}_1^T & \boldsymbol{\theta}_1^T & \dot{\boldsymbol{\theta}}_1^T & \boldsymbol{\Omega}_1^T \end{bmatrix}^T - \boldsymbol{x}_0 = \boldsymbol{0}_{15} \\ \begin{bmatrix} \boldsymbol{p}_N^T & \dot{\boldsymbol{p}}_N^T & \boldsymbol{\theta}_N^T & \dot{\boldsymbol{\theta}}_N^T & \boldsymbol{\Omega}_N^T \end{bmatrix}^T - \boldsymbol{x}_d(t_f) = \boldsymbol{0}_{15} \tag{14}$$

It should be noted that the platform final state $\boldsymbol{x}_d(t_f)$ can be time varying if the platform motion can be explicitly expressed as a function of total time t_f.

4. Bounds on the state variables: some elements of the optimization variable X_k may be bounded to enforce state bounds or input limits, etc. These bounds on X_k are typically linear constraints. Examples include (1) the total time t_f is positive, (2) the Euler angles should be small enough to justify the small angle assumption from (3) for all $k = 1, 2, \cdots, N$. These constraints are written as

$$0 \leq t_f, \\ \begin{bmatrix} \alpha_{\min} \\ \beta_{\min} \\ \gamma_{\min} \end{bmatrix} \leq \boldsymbol{\theta}_k \leq \begin{bmatrix} \alpha_{\max} \\ \beta_{\max} \\ \gamma_{\max} \end{bmatrix}, \\ \forall k = 1, 2, \cdots, N \tag{15}$$

Formulation of the optimization problem With the definition of the optimization variables, constraints and cost functions, a new formulation of the optimization problem (4) may

now be constructed. By stacking the constraints described previously and denote the constraint function as $g(X_k)$, the optimization problem (4) can now be written as

$$\begin{aligned} \underset{X_k,t_f}{\arg\min} \quad & J = t_f \\ s.t \quad & g_{\min} \leq g(X_k,t_f) \leq g_{\max}, \forall k = 1,2,\cdots,N. \end{aligned} \tag{16}$$

Remark: The optimization problem can then be modeled (using CasADi (Ref. 15)) and solved (with Ipopt (Ref. 16)) using standard NLP solvers.

Choice of initial guess for optimization In the previous subsection, a nonlinear programming problem was formulated. Usually an accurate initial feasible guess is necessary to solve such a (typically nonconvex) nonlinear programming problem. Here, we have used a randomized approach to generate a set of initial guesses.

Generating the reference trajectory The solution of the optimization problem above generates the optimal state trajectory, $\boldsymbol{X}^* = \begin{bmatrix} X_1^* & X_2^* & \cdots & X_k^* \cdots & X_N^* \end{bmatrix} \in \Re^{21N}$, which is the state variable at each sampling time. For the $k-$th sample, the optimal solution is X_k^*

$$\begin{aligned} X_k^* &= \begin{bmatrix} \boldsymbol{p}_k^{*t} & \dot{\boldsymbol{p}}_k^{*T} & \ddot{\boldsymbol{p}}_k^{*T} & \boldsymbol{\theta}_k^{*T} & \dot{\boldsymbol{\theta}}_k^{*T} & \boldsymbol{\Omega}_k^{*T} & \dot{\boldsymbol{\Omega}}_k^{*T} \end{bmatrix} \\ k &= 1,2,\cdots,N \end{aligned} \tag{17}$$

X_k^* contains the optimal reference state trajectory $\boldsymbol{x}^*$. The reference state at $k-$th sampling time is denoted as

$$\boldsymbol{x}^* = \begin{bmatrix} \boldsymbol{p}_k^{*T} & \dot{\boldsymbol{p}}_k^{*T} & \boldsymbol{\theta}_k^{*T} & \boldsymbol{\Omega}_k^{*T} \end{bmatrix} \in \Re^{12} \tag{18}$$

Generating the optimal input (force/torque) In the previous subsection, the *state* reference trajectory was generated based on the solution of the optimization problem $\boldsymbol{X}^*$. Based on the differential flatness property of the quadrotor dynamics, the ideal reference input $\boldsymbol{u}^*$ can then be obtained. The mapping from the state to the force and torque input is given by (8). By replacing the state with the solution from the previous subsection, the optimal input $\boldsymbol{u}^*$ can be calculated:

$$\begin{aligned} T^* &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \psi_1(\boldsymbol{x}_{X_k^*}) \\ \begin{bmatrix} \tau_x^* \\ \tau_y^* \\ \tau_z^* \end{bmatrix} &= \psi_2(\boldsymbol{x}_{X_k^*}) \\ \boldsymbol{u}^* &= \begin{bmatrix} T^* & \tau_x^* & \tau_y^* & \tau_z^* \end{bmatrix}^T \end{aligned} \tag{19}$$

It should be noted that $\boldsymbol{u}^*$ is the *ideal* input. If there are external disturbances or system uncertainties, directly applying the ideal input $\boldsymbol{u}^*$ without any feedback stabilization or disturbance rejection will not guarantee good tracking performance.

TRACKING CONTROL (INNER LOOP) DESIGN

Thus far we have generated a suitable time-optimal reference trajectory $\boldsymbol{x}^*$ and the corresponding input $\boldsymbol{u}^*$. In this section, a tracking control algorithm is described that generates the control input $\boldsymbol{u}$ based on the reference input to track the reference trajectory. The proposed trajectory tracking controller consists of a feedback and a feedforward controller, described below.

Feedback controller design The feedback controller is a standard linear quadratic regulator (LQR). It is based on linearized system dynamics, as in (Ref. 17). The linear dynamics can be written as

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u} \\ A &\in \Re^{12\times12}, B \in \Re^{12\times4} \end{aligned} \tag{20}$$

The feedback controller is of the form

$$u_{fb} = K(\boldsymbol{x}^* - \boldsymbol{x}), \tag{21}$$

where K is obtained from a standard LQR design. The reference state is as obtained in the previous section.

Feedforward controller design A model reference adaptive controller (MRAC) similar to (Ref. 12) is used as the feedforward inner loop controller, to account for uncertainty in the system model. The control law is

$$\begin{aligned} u_{ff} &= \boldsymbol{\delta}\boldsymbol{\eta} = K_x\boldsymbol{x} + K_r\boldsymbol{u}^* \\ \boldsymbol{\delta} &= \begin{bmatrix} K_x & K_r \end{bmatrix}, \eta = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{u}^* \end{bmatrix} \\ K_x &\in \Re^{4\times12}, K_r \in \Re^{4\times4} \end{aligned} \tag{22}$$

The matrix $\boldsymbol{\delta}$ is a matrix of time-varying adaptive parameters. The vector $\boldsymbol{\eta}$ is the vector of system state $\boldsymbol{x}$ and system reference input $\boldsymbol{u}^*$. The classical adaptive law is given by

$$\begin{aligned} \dot{K}_x &= \Gamma_x B^T P e \boldsymbol{x}^T \\ \dot{K}_r &= \Gamma_r B^T P e \boldsymbol{u}^{*T} \end{aligned} \tag{23}$$

where Γ_x and Γ_r are diagonal, positive definite matrices of adaptive gains, $e = \boldsymbol{x}^* - \boldsymbol{x}$ is the model tracking error. P is the positive definite solution of the Lyapunov equation $A^T P + PA = -Q$ where Q any positive definite matrix of compatible dimension.

RESULTS

Experimental setup

We now briefly describe the experimental setup shown in Fig. 4. The set up consists of a central computer, a quadrotor, a platform and a motion capture system. A Hummingbird quadrotor from the Ascending Technology is used for the experimental validation. The platform has heave, roll, and pitch degrees of freedom. The heave motion is generated by a step

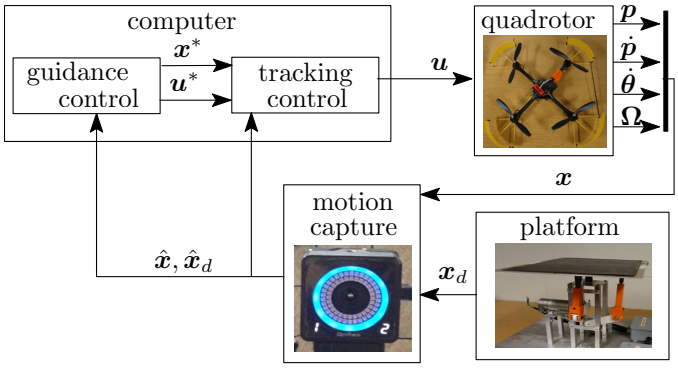


Fig. 4: Schematic of the setup for quadrotor landing experiments. In this setup, the computer sends the control commands to the quadrotor. The position and orientation of the quadrotor and the platform are captured by the motion capture system and fed back to the computer at 100 Hz.

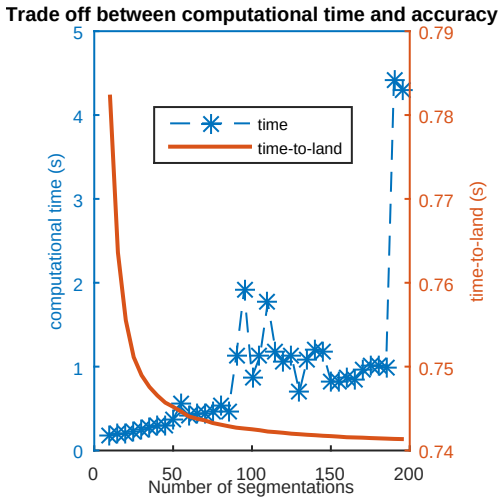


Fig. 5: Plot of trade-off between the accuracy and the total time as the number of segments N is increased. The dashed line shows the computational time required to solve the time-optimal problem and the solid time shows the total time-to-land. The total time-to-land converges to an optimal (minimum) value, and the computational time increases significantly if N is larger than 185.

motor and the roll/pitch motion are generated using two servo motors. Several infrared markers are attached to the quadrotor and the platform. The position and orientation of these markers are captured by the motion capture system and sent to the computer. The trajectory generation algorithm and the tracking control algorithm are implemented on MATLAB on this Linux computer. The generated control commands are then sent wirelessly to the quadrotor at 100Hz update rate.

For simulation results, we use the quadrotor model parameters presented in (Ref. 18). The feasible system input set is defined as follows:

$$\begin{aligned} 1\ N \leq T \leq 30\ N \\ -8\ Nm \leq \tau_x \leq 8\ Nm \\ -8\ Nm \leq \tau_y \leq 8\ Nm \\ -8\ Nm \leq \tau_z \leq 8\ Nm \end{aligned} \tag{24}$$

The roll, pitch and yaw Euler angles are all bounded within $[-\frac{\pi}{2}, \frac{\pi}{2}]\ rad$.

Choice of number of segments N

In this subsection, we first perform a series of simulations to choose the best number of segments N for the trajectory generation algorithm. Figure 5 shows the computation time and final time for various choices of the number of segments N. The initial position of the quadrotor was $(0\ m, 0\ m, 2\ m)$. The final state $\boldsymbol{x}_d(t_f)$ is defined as

$$\begin{aligned} \boldsymbol{p}_d(t_f) &= (1\ m, 1\ m, \sin(t_f)\ m) \\ \dot{\boldsymbol{p}}_d(t_f) &= (0\ m/s, 0\ m/s, \cos(t_f)\ m/s) \\ \boldsymbol{\theta}_d(t_f) &= (-0.6\sin(t_f)\ rad, 0.6\sin(t_f)\ rad, 0\ rad). \end{aligned} \tag{25}$$

With a larger N, the computational time (blue dashed line) increases, while the optimal total time-to-land (brown solid line) reduces and converges to a minimum value, which is the *true* optimal time-to-land. It is interesting to note that the computational time increases significantly after $N > 185$ except for some outliers. Therefore, we use $N = 185$ as the ideal segment number in the following simulations. Furthermore, we note that for $N = 185$, the optimization problem can be solved in less than 1 second, therefore opening up the possibility for real-time implementation.

Simulation of time-optimal landing onto a heaving-rolling-pitching platform

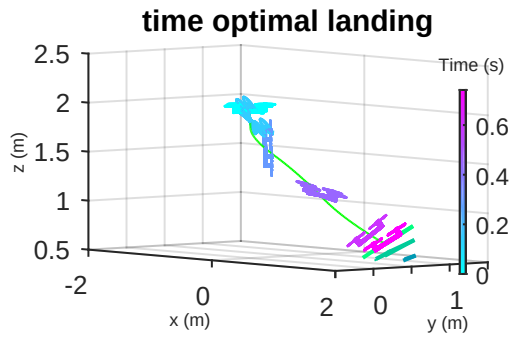
In this simulation, a time-optimal trajectory is designed for a quadrotor landing onto a heaving, rolling and pitching platform. The quadrotor initial position is $(0\ m, 0\ m, 2\ m)$. The platform motion $\boldsymbol{x}(t_f)$ is defined as:

$$\begin{aligned} \boldsymbol{p}_d(t_f) &= (1\ m, 1\ m, \sin(t_f)\ m) \\ \dot{\boldsymbol{p}}_d(t_f) &= (0\ m/s, 0\ m/s, \cos(t_f)\ m/s) \\ \boldsymbol{\theta}_d(t_f) &= (-0.6\sin(t_f)\ rad, 0.6\sin(t_f)\ rad, 0\ rad). \end{aligned} \tag{26}$$

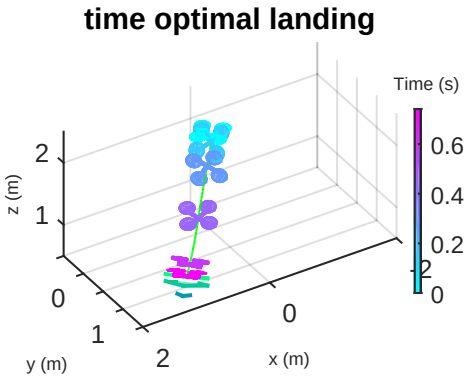
Fig. 6 illustrates a plot of the time-optimal landing maneuver, (a) is the side view while (b) shows the top view. It can be seen from both the side view and the top view figures that at the final time instant of the maneuver, the quadrotor lands on the platform with the height, roll angle and pitch angle matching that of the platform. The total time for the maneuver is 0.75 seconds. Fig. 8 shows the corresponding optimal input for this landing maneuver. It can be seen that the inputs are within the feasible bounds defined in (24).

Simulation of time-optimal landing with collision avoidance

In this simulation, in addition to landing onto a platform with same motion defined in (28), there is a spherical obstacle centered at $(0.5\ m, 0.5\ m, 1.5\ m)$ with a radius of 0.25 meter. We assume that a safety distance from the center of the quadrotor to the surface of the sphere of 0.25 meter must be maintained,

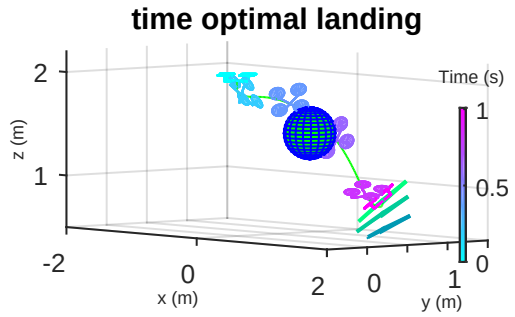


(a) Side view of quadrotor maneuver.

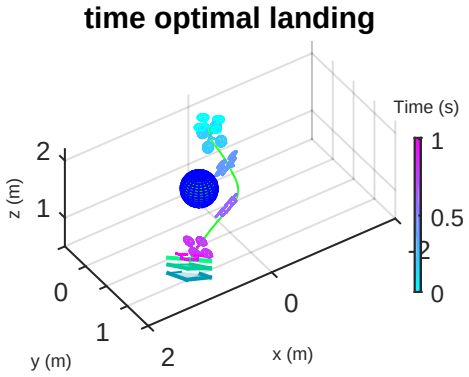


(b) Top view of quadrotor maneuver

Fig. 6: Plot of the time-optimaltrajectory for landing onto a heaving, rolling and pitching platform. At the final time, the position and inclination of the quadrotor match with those of the platform. The quadrotor and the platform position/orientation are plotted every 0.15 second.



(a) Side view of quadrotor maneuver



(b) Top view of quadrotor maneuver

Fig. 7: Plot of the time-optimaltrajectory for landing onto a heaving, rolling and pitching platform, while avoiding an obstacle. The platform motion is same as in Fig. 6. In this case, the difference is that this trajectory avoids a spherical obstacle centered at (0.5,0.5,1.5). The quadrotor and the platform states are plotted every 0.2 seconds.

i.e. the center of the quadrotor is always 0.5 meters away from the center of the sphere. Therefore, an extra constraint to avoid the obstacle is added to the optimization problem (16) as below

$$\|\boldsymbol{p}_k - \begin{bmatrix}0.5 & 0.5 & 1.5\end{bmatrix}^T\| \geq 0.5, \forall k = 1,2,\cdots,N \qquad (27)$$

Fig. 7 shows the time-optimal trajectory that avoids a sphere obstacle. It can be seen that the generated trajectory avoids collision with the obstacle and guarantees that the height, roll and pitch angle of the quadrotor match those of the platform at the final time (at contact). Fig. 9 shows the optimal input $\boldsymbol{u}^*$ corresponding to the time-optimal trajectory. The input constraints are also satisfied.

Experimental validation of time-optimal landing

In this experiment, we demonstrate real time implementation of time-optimal landing onto a stationary platform. The quadrotor hovers at $(0\ m, 0\ m, 0.5\ m)$ at the beginning of the maneuver. The final state stationary in this case is defined as:

$$\begin{aligned} \boldsymbol{p}_d(t_f) &= (0\ m, 2.5\ m, 0\ m) \\ \dot{\boldsymbol{p}}_d(t_f) &= (0\ m/s, 0\ m/s, 0\ m/s) \\ \boldsymbol{\theta}_d(t_f) &= (0\ rad, 0\ rad, 0\ rad). \end{aligned} \qquad (28)$$

It should be noticed that the reference trajectory was generated with conservative limits on the maximum angular velocity and maximum jerk (derivative of the acceleration) in the horizontal direction to make the trajectory trackable. Figure 10 shows the trajectory tracking in y and z directions. The green dashed line is the reference trajectory and the red line is the actual trajectory of the quadrotor. It can be seen that in the y direction, the tracking is good. In the z direction, there is some tracking error during the maneuver but the error at the final time is zero, which shows the effectiveness of the proposed controller.

CONCLUSIONS

This paper proposed a time-optimal trajectory generation method for landing a quadrotor with 6 degrees of freedom

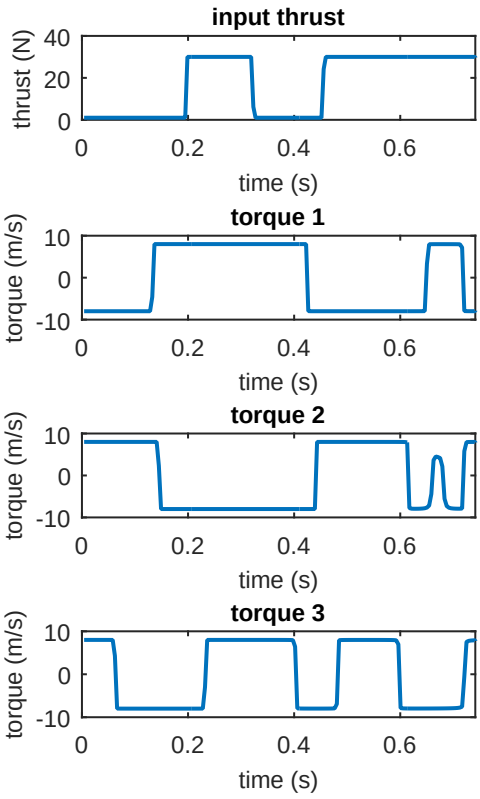


Fig. 8: Plot of input for the time-optimal trajectory shown in Fig. 6. All the inputs satisfy the constraints for feasibility.

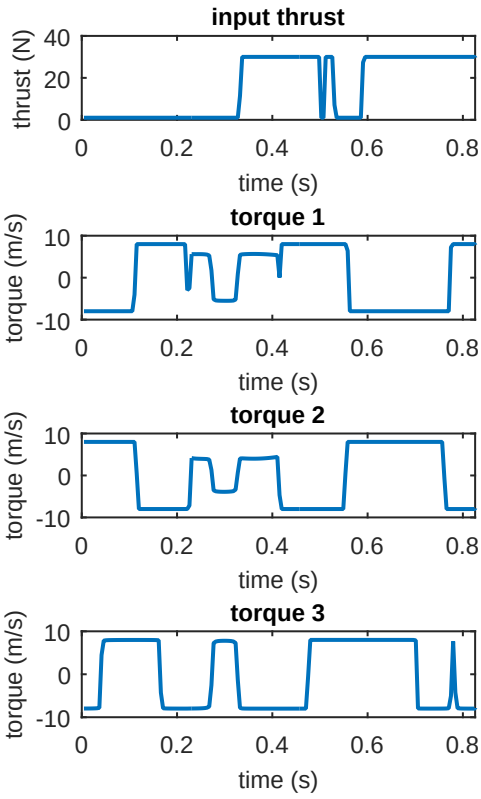
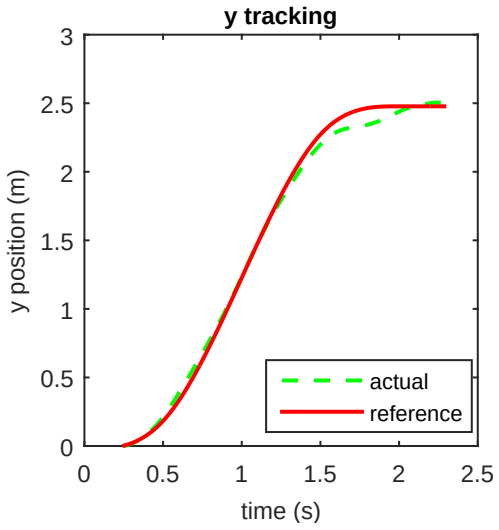
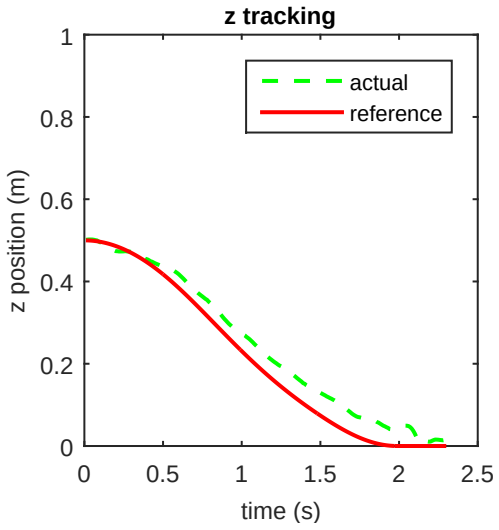


Fig. 9: Plot of input for the time-optimaltrajectory shown in Fig. 7. All the inputs satisfy the constraints for feasibility.



(a) y direction tracking



(b) z direction tracking

Fig. 10: Plot of the quadrotor trajectory landing onto a stationary ground. The dashed line is the reference trajectory and the red line is the actual tracking trajectory.

onto a heaving, rolling and pitching platform. The time-optimal control problem is transformed into a nonlinear programming problem by applying the differential flatness property of the quadrotor dynamics, which improves the computational tractability. A trajectory tracking controller based on the Model Reference Adaptive Control is designed to track the reference trajectory in a real experiment. Simulation and experimental results show the effectiveness of the proposed trajectory generation method. We demonstrate real-time implementation of the system

Author contact: Botao Hu hub2@rpi.edu; Sandipan Mishra mishrs2@rpi.edu.

ACKNOWLEDGMENT

This work was funded by Office of Naval Research under award N00014-16-1-2705.

REFERENCES

[1]Sanchez-Lopez, J. L., Saripalli, S., and Campoy, P., "Autonomous ship board landing of a VTOL UAV," American Helicopter Society 69th Forum, 2013.

[2]Horn, J. F., Tritschler, J., and He, C., "Autonomous Control Modes and Optimized Path Guidance for Shipboard Landing in High Sea States," Technical report, DTIC Document, 2014.

[3]Ngo, T. D., *Constrained control for helicopter shipboard operations and moored ocean current turbine flight control*, Ph.D. thesis, Dept of Aerospace Eng., Virginia Polytechnic Inst, Blacksburg, VA, 2016.

[4]Holmes, W. and Langelaan, J., "Autonomous ship-board landing using monocular vision," 72nd American Helicopter Society Forum, May 17 2016.

[5]Hu, B., Lu, L., and Mishra, S., "Fast, safe and precise landing of a quadrotor on an oscillating platform," American Control Conference, 2015., July 2015.

[6]Hu, B., Lu, L., and Mishra, S., "A control architecture for fast and precise autonomous landing of a VTOL UAV onto an oscillating platform," AHS 71st Auunal Forum, May 2015.

[7]Hehn, M. and DAndrea, R., "Real-time trajectory generation for quadrocopters," *IEEE Transactions on Robotics*, Vol. 31, (4), 2015, pp. 877–892.

[8]Hehn, M., Ritz, R., and DAndrea, R., "Performance benchmarking of quadrotor systems using time-optimal control," *Autonomous Robots*, Vol. 33, (1-2), 2012, pp. 69–88.

[9]Lai, L.-C., Yang, C.-C., and Wu, C.-J., "Time-Optimal Control of a Hovering Quad-Rotor Helicopter," *Journal of Intelligent and Robotic Systems*, Vol. 45, (2), 2006, pp. 115–135.

[10]Van Loock, W., Pipeleers, G., and Swevers, J., "Time-optimal quadrotor flight," Proc. of the 2013 European Control Conference (ECC), 2013.

[11]Hu, B. and Mishra, S., "A time-optimal trajectory generation algorithm for quadrotor landing onto a moving platform," American Control Conference, 2017., under review.

[12]Dydek, Z., Annaswamy, A., and Lavretsky, E., "Adaptive Control of Quadrotor UAVs: A Design Trade Study With Flight Evaluations," *Control Systems Technology, IEEE Transactions on*, Vol. 21, (4), July 2013, pp. 1400–1406.

[13]Fliess, M., Lévine, J., Martin, P., and Rouchon, P., "Flatness and defect of non-linear systems: introductory theory and examples," *International journal of control*, Vol. 61, (6), 1995, pp. 1327–1361.

[14]Loock, W. V., Pipeleers, G., Diehl, M., Schutter, J. D., and Swevers, J., "Optimal Path Following for Differentially Flat Robotic Systems Through a Geometric Problem Formulation," *IEEE Transactions on Robotics*, Vol. 30, (4), Aug 2014, pp. 980–985.
doi: 10.1109/TRO.2014.2305493

[15]Andersson, J., *A General-Purpose Software Framework for Dynamic Optimization*, PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013.

[16]Wachter, A. and Biegler, L., "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, Vol. 106, (1), 2006, pp. 25–57.

[17]Mellinger, D., Michael, N., and Kumar, V., "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, Vol. 31, (5), 2012, pp. 664–674.

[18]Debrouwere, F., Van Loock, W., Pipeleers, G., Dinh, Q. T., Diehl, M., De Schutter, J., and Swevers, J., "Time-optimal path following for robots with convex–concave constraints using sequential convex programming," *IEEE Transactions on Robotics*, Vol. 29, (6), 2013, pp. 1485–1495.