A time-optimal trajectory generation algorithm for quadrotor landing onto a moving platform

Botao Hu[1] and Sandipan Mishra[1]

Abstract— **In this paper, we propose a time-optimal trajectory generation algorithm for landing a 3 DOF (x-z and tilt) quadrotor model onto a moving platform. The algorithm exploits the differential flatness of the quadrotor dynamics model and formulates a nonlinear programming problem, which is then solved to obtain the time-optimal landing trajectory. The advantages of the proposed algorithm over state-of-the-art solution techniques for time optimal trajectory design include computational efficiency and the ability to incorporate dynamics and state-constraints (such as collision avoidance from an obstacle) into the optimization problem. Simulation results and comparisons with a benchmark algorithm show the effectiveness of the proposed method.**

I. INTRODUCTION

Of late, small unmanned aerial vehicles (UAV's) such as quadrotors are being used for many applications such as aerial imaging, package delivery, and surveillance. As such, there is now a growing body of literature on a variety of autonomy problems related to UAV's. [1] [2] present recent surveys on advances in algorithm design for guidance and control of UAVs. Designing time optimal trajectories for maneuvers are important for several applications, including package delivery, search-and-rescue problems, and perching/landing as presented in [3] [4]. Landing, in particular, poses unique challenges for time optimal trajectory design because of the stringent safety requirements. Furthermore, landing on to a moving deck or surface further compounds this problem. In this paper, we focus on design of time-optimal trajectory design for landing quadrotors onto moving platforms.

Given a known trajectory or set of fixed way points, the time optimal maneuver problem may be cast as a *time-optimal trajectory following problem*, which has seen substantial research interest. The time-optimal trajectory following problem for different applications, including robotic arms and quadrotors, has been studied in [5] [6] [7] [8].

However, for most UAV (quadrotor) maneuvers, the trajectory (or waypoints) is not typically given *a priori*. Usually, the initial state of the quadrotor is known and the final state is prescribed but may be time varying. Examples of such problems include time optimal landing onto a vertically oscillating platform, as presented in [9] and the time optimal maneuver problem presented in [3]. Fig. 1 illustrates vertical landing of a quadrotor onto a fixed platform and a vertically oscillating and tilting platform. A key challenge for the latter case is the fact that the time-optimal trajectory generation

[1]Botao Hu and Sandipan Mishra are with the department of Mechanical, Aerospace and Nuclear Engineering of Rensselaer Polytechnic Institute, 110 8th street, Troy, NY, 12180 {hub2,mishs2}@rpi.edu

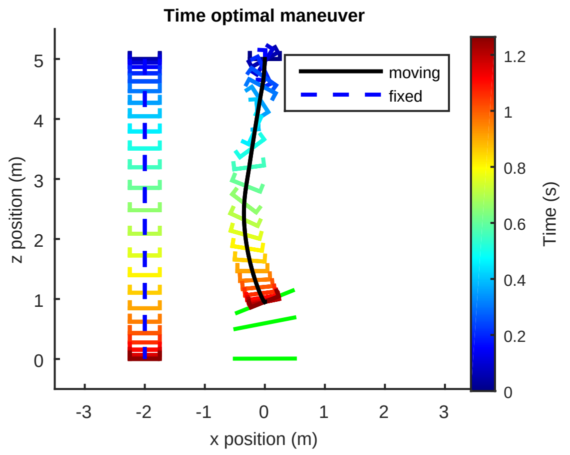

Fig. 1. Trajectory comparison for landing onto stationary and moving platforms. The trajectory on the left illustrates landing onto a fixed platform. The trajectory on the right illustrates landing onto an oscillating and tilting platform. The green bar is the platform position.

algorithm must account for the terminal state being time-varying (and thus dependent on the minimum final time). In this paper, we focus on the problem of time optimal trajectory generation for a quadrotor landing onto a moving and tilting platform.

For this class of problems, a time-optimal trajectory must be generated by solving a non-convex optimization problem. Nonlinear programming algorithms such as Genetic Algorithm [10] and the Switching Time Optimization Algorithm [3] have been proposed for solving this problem.

In addition to directly solving the nonlinear programming problem, an interesting approach that uses differential flatness to address the point-to-point maneuver problem was presented in [11] [4] [12], [13]. Although this approach does not directly deal with the problem of trajectory generation for landing onto a *moving* (translating and tilting) platform, it presents a mechanism to transform the constraints on the system input onto the constraints on the trajectory by exploiting the differential flatness of the quadrotor dynamics. This approach can significantly improve computational efficiency of solving the time optimal point-to-point maneuver as well as collision avoidance problems [4].

Inspired by this body of previous work, this paper presents a new trajectory generation algorithm that can generate a time optimal trajectory to land a quadrotor onto a translating and tilting platform. The general idea is to formulate an optimization problem that exploits the differential flatness property as in [4]. Unlike the algorithm proposed in [4], the formulation of the problem and the selection of the optimization variables presented in this paper expands the scope of problems that may be solved, as explained below.

Specifically, one major advantage of the proposed algorithm is that it allows for the final desired state to be time varying, as long as the final desired state can be expressed as a known explicit function of time (i.e. $\mathbf{x}_d(t) = f(t)$). We demonstrate the applicability of this algorithm for a quadrotor landing/perching onto a translationally moving and tilting platform. Furthermore, this algorithm can also easily incorporate the obstacle avoidance constraints into the trajectory generation. Even though the problem formulation is not convex, by suitable choice of the initial guesses of the optimization variables, the problem can be solved efficiently.

This paper is organized as follows. Section II presents the system dynamics and the time-optimal landing problem formulation. Section III develops the differentially flat time-optimal trajectory generation algorithm, while section IV presents the results that illustrate the performance of the algorithm. Section V concludes the paper.

II. 3 DOF Quadrotor Dynamics Model and Problem Formulation

We first present the 3 DOF system dynamics model for a quadrotor and formulate the time-optimal landing trajectory design as a constrained optimization problem.

A. 3 DOF Quadrotor model

The model used in this paper is a two-dimensional three degrees of freedom (3 DOF) quadrotor model, as presented in [3] [4] [1]. The state of the system includes the horizontal position $x(t)$, the vertical height $z(t)$ and the pitch angle $\theta(t)$. $x_d(t), z_d(t), \theta_d(t)$ denote the horizontal position, vertical position and pitch angle of a platform that the quadrotor must land on. Note that the platform state can be *time-varying* but must be explicitly written as a function of time t. Fig. 2 shows the quadrotor and platform models. The state of the quadrotor $\mathbf{x}$ and the state of the platform $\mathbf{x}_d$ can therefore be defined as:

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} = \begin{bmatrix} x & \dot{x} & z & \dot{z} & \theta \end{bmatrix} \in \mathbb{R}^5 \\ \mathbf{x}_d &= \begin{bmatrix} x_d & \dot{x}_d & z_d & \dot{z}_d & \theta_d \end{bmatrix} \in \mathbb{R}^5. \end{aligned} \tag{1}$$

The thrust is F_t and the mass of the quadrotor is m. The rotation speed is ω. Define the system input $\mathbf{u}$ as:

$$\mathbf{u}(t) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \frac{F_T}{m} \\ \omega \end{bmatrix} \in \mathbb{R}^2. \tag{2}$$

The dynamics are thus described by the following:

$$\begin{bmatrix} \ddot{x} \\ \ddot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} \tag{3}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{bmatrix} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} x_2 \\ u_1 \sin(x_5) \\ x_4 \\ u_1 \cos(x_5) - g \\ u_2 \end{bmatrix}, \tag{4}$$

[1]This model is chosen so as to use the results presented in [3] as a benchmark comparison.

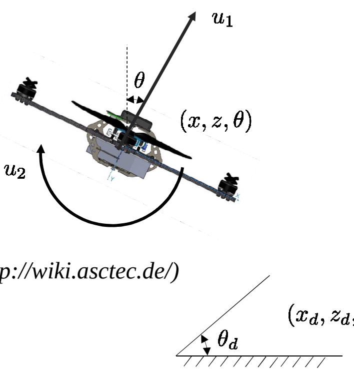

Fig. 2. Schematic of the quadrotor and the landing platform. The quadrotor is modeled with position and pitch angle (x, z, θ) and has two inputs u_1 and u_2. The platform is modeled with $x - z$ position and pitch angle (x_d, z_d, θ_d).

where g is the gravitational acceleration. The input u_1 is the nominal thrust, while the input u_2 is the angular rate, both of which are constrained to a feasible set denoted by $\mathbf{U}$. The actuator saturation constraints on u_1 and u_2 can be written as:

$$\begin{aligned} &\mathbf{u}(t) \in \mathbf{U} = \\ &= \left\{ (u_1, u_2) | \begin{array}{c} u_{1\min} \le u_1 \le u_{1\max} \\ \le u_{2\min} \le u_2 \le u_{2\max} \\ U_{1\min} \le u_{1\min} \le u_{1\max} \le U_{1\max} \\ U_{2\min} \le u_{2\min} \le u_{2\max} \le U_{2\max} \end{array} \right\}. \end{aligned} \tag{5}$$

The actuation range for u_1 is $[U_{1\min}, U_{1\max}]$, while the actuation range for u_2 is $[U_{2\min}, U_{2\max}]$. In order to conservatively design the inputs, we use a smaller range $[u_{i\min}, u_{i\max}]$ for each input. This is in order to account for unknown dynamics and disturbances during operation. Thus, for the trajectory generation, constraints as $u_1 \in [u_{1\min}, u_{1\max}], u_2 \in [u_{2\min}, u_{2\max}]$ are used.

B. Problem formulation

The objective of the trajectory generation algorithm is to achieve land the quadrotor onto a moving and tilting platform within the smallest possible time. Further, during the maneuver, a set of safety constraints must be satisfied. These constraints include but are not limited to input saturation constraints and collision avoidance constraints. Finally, at the point of landing, the quadrotor states should be within a small neighborhood of the platform states to guarantee a smooth and minimum-impact landing. These states include the position (x, z), velocity $(\dot{x}, \dot{z})$ and pitch angle θ.

Let $\mathbf{g}(\mathbf{x})$ characterize the safety measurables, such as the input or the distance from obstacle, etc. The safety constraints are enforced by staying within a set $\mathbf{G}$, i.e.,:

$$\mathbf{g}(\mathbf{x}) \in \mathbf{G} = \{\mathbf{g} | \mathbf{g}_{\min} \le \mathbf{g} \le \mathbf{g}_{\max}\}. \tag{6}$$

Let t_f be the final (as yet unknown) time, $\mathbf{x}_0$ as the initial state. Denote $\mathbf{x}_d(t_f)$ as the state of the platform at the final time. The time optimal quadrotor landing problem can be formulated as an optimal control problem. The optimization variables are the system input (over the landing time horizon)

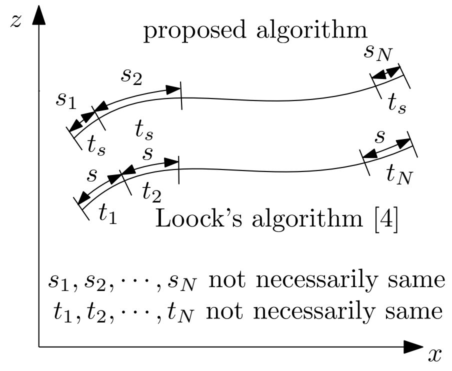

Fig. 3. This is the comparison of trajectory discretization between proposed method and the Loock's algorithm proposed in [4]. The duration time for the proposed algorithm is same for each discretized step, while the amount of distance for each step is different. The Loock's algorithm adopts an opposite discretize approach.

and the final landing time. The optimal control problem can be written as (7):

$$\begin{aligned}
\arg\min_{\mathbf{u}(t),t_f} \quad & J = t_f \\
s.t \quad & \dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u}) \\
& \mathbf{x}(t=0) = \mathbf{x}_0 \\
& \mathbf{x}(t=t_f) = \mathbf{x}_d(t_f) \\
& \mathbf{g}(\mathbf{x}(t)) \in \mathbf{G}, \forall t \in [0,t_f] \\
& \mathbf{u}(t) \in \mathbf{U}, \forall t \in [0,t_f].
\end{aligned} \tag{7}$$

Next, we propose a trajectory generation algorithm to solve the problem described in (7). We reformulate (7) by using the differential flatness of the quadrotor system dynamics. The proposed algorithm discretizes the trajectory into a predetermined number of equal time-segments (N). The distance along the trajectory, s_i, covered in the time-segment $t_i = \Delta t = t_s$ for each step may vary. This is the key difference from the Loock's algorithm proposed in [4], where for each segment (i) the distance along the trajectory, $s_i = s$, is same, while the length of the time-segment t_i varies. Fig. 3 illustrates this difference in the discretization approaches of these two algorithms.

III. Time-optimal trajectory generation algorithm using differential flatness

In this section, we develop a time-optimal trajectory generation algorithm using differential flatness. The trajectory generation algorithm consists of two key steps: determining the flat output and reformulating the problem using the flat output and its derivatives. The idea behind this is to transform the constraints on the system input into constraints on the system states. The optimization variables are therefore the system states, from which the optimal control input can be reconstructed, again using the differential flatness property. We first briefly recall the concept of differential flatness and flat output and then present the reformulation of the time-optimal problem.

A. *Differential flatness and flat output*

We here briefly recall the differentially flat property of the quadrotor dynamics model. A detailed explanation of differentially flat systems is presented in [14].

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

$$\mathbf{y} = \phi(\mathbf{x},\mathbf{u},\dot{\mathbf{u}},\cdots,\mathbf{u}^{(i)}). \tag{8}$$

The system is differentially flat if the system state and system input can be represented by the derivatives of the system flat output

$$\begin{aligned}
\mathbf{x} &= \phi_x(\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots,\mathbf{y}^{(j)}) \\
\mathbf{u} &= \phi_u(\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots,\mathbf{y}^{(j-1)}).
\end{aligned} \tag{9}$$

Based on the definition above, the system dynamics described by (3) is differentially flat (A formal proof can be found in [4]). The flat output $\mathbf{y}$ is (x,z,θ). The control input u_1 and u_2 at time t can be represented in terms the state from the dynamics equation (3):

$$\begin{aligned}
\begin{bmatrix} 0 \\ u_1(t) \\ u_2(t) \end{bmatrix} &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \ddot{x} \\ \ddot{z}+g \\ \dot{\theta} \end{bmatrix} \\
&= \begin{bmatrix} \ddot{x}(t)\cos(\theta(t)) + \sin(\theta(t))(\ddot{z}+g) \\ \ddot{x}(t)\sin(\theta(t)) + (\ddot{z}(t)+g)\cos(\theta(t)) \\ \dot{\theta}(t) \end{bmatrix}.
\end{aligned} \tag{10}$$

Remark: Choosing (x,z,θ) as the flat output simplifies the constraint on u_2 in the optimization problem. The constraint on u_2 can be transformed into a linear constraint of the state $\dot{\theta}$, details of which are presented in the following subsection.

B. *Optimization problem formulation*

1) Cost function: For the optimization problem, the objective function to be minimized is the total time. The cost function is thus defined same as defined in (7):

$$\min \quad J = t_f. \tag{11}$$

2) Optimization variables: Although the original system dynamics for optimization are continuous, for computational tractability, the system dynamics are discretized into N steps. The number N is a given constant fixed *a priori* . A proper selection of N is a result of trade off between the accuracy and the computational time necessary for the numerical solution of the problem. The sampling time t_s for each step can be calculated from the total time:

$$t_s = \frac{t_f}{N-1}. \tag{12}$$

A discretized state variable $X(k)$ will be used to represent the state variables at k-th sampling time. The elements in $X(k)$ are denoted as $x_k, \dot{x}_k, \cdots, \dot{\theta}_k$:

$$X(k) = \begin{bmatrix} x_k & \dot{x}_k & z_k & \dot{z}_k & \theta_k & \ddot{x}_k & \ddot{z}_k & \dot{\theta}_k \end{bmatrix}^T \in \mathbb{R}^{8\times 1}. \tag{13}$$

The optimization variables thus include the discretized state variable $X(k)$ for all N steps and the final time t_f. The overall size of the optimization variable is a vector belongs to $\mathbb{R}^{(8N+1)\times 1}$ by summing up all the state variables and final time t_f.

3) Optimization constraints: The optimization constraints may vary for different cases, i.e., collision avoidance, actuator limits, state bounds, etc. Here, we present formulations of some basic classes of constraints.

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

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ z_{k+1} \\ \dot{z}_{k+1} \\ \theta_{k+1} \end{bmatrix} - \begin{bmatrix} x_k + \dot{x}_k t_s + \frac{\ddot{x}_k t_s^2}{2} \\ \dot{x}_k + \ddot{x}_k t_s \\ z_k + \dot{z}_k t_s + \frac{\ddot{z}_k t_s^2}{2} \\ \dot{z}_k + \ddot{z}_k t_s \\ \theta_k + \dot{\theta}_k t_s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (14)$$
$$\forall k = 1, 2, \cdots, N-1,$$

2) System input constraints: based on the differential flatness (10), the constraints on u_1 can be represented as:

$$\begin{aligned} & u_{1\min} \leq \sin(\theta_k) + (\ddot{z}_k + g)\cos(\theta_k) \leq u_{1\max} \\ & \ddot{x}_k \cos(\theta_k) - (\ddot{z}_k + g)\sin(\theta_k) = 0 \\ & \forall k = 1, 2, \cdots, N. \end{aligned} \quad (15)$$

3) Initial and final state constraints: the initial state and the final state of the quadrotor should be same as the given initial state $\mathbf{x}_0$ and the platform final state $\mathbf{x}_d(t_f)$ respectively. The constraints that enforce these are represented as:

$$\begin{aligned} \begin{bmatrix} x_1 & \dot{x}_1 & z_1 & \dot{z}_1 & \theta_1 \end{bmatrix} &= \mathbf{x}_0^T \\ \begin{bmatrix} x_N & \dot{x}_N & z_N & \dot{z}_N & \theta_N \end{bmatrix} &= \mathbf{x}_d(t_f)^T \end{aligned} \quad (16)$$

It should be noted that the platform final state $\mathbf{x}_d(t_f)$ 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: for the optimization variable $X(k)$, some elements may be bounded (to enforce state bounds or input limits, etc.). These bounds on $X(k)$ are typically linear constraints. Among such bounds: (1) the total time t_f should be positive, (2) the angle of the quadrotor θ_k should be bounded by $[-2\pi, 2\pi]$ and (3) the input $u_2(k) = \dot{\theta}_k$ should be bounded by $[u_{2\min}, u_{2\max}]$ for all $k = 1, 2, \cdots, N$. These constraints are written as

$$\begin{aligned} 0 \leq & \quad t_f & , \\ -2\pi \leq & \quad \theta_k & \leq 2\pi, \\ u_{2\min} \leq & \quad \dot{\theta}_k & \leq u_{2\max}, \\ & \quad \forall k & = 1, 2, \cdots, N. \end{aligned} \quad (17)$$

4) Formulate the optimization problem: With the definition for the optimization variables, constraints and cost functions, a new formulation of the optimization problem (7) may now be constructed. By stacking the constraints described previously and denote the constraint function as $g(X(k))$, the optimization problem (7) can now be written as

$$\begin{aligned} \underset{X(k), t_f}{\arg\min} \quad & J = t_f \\ s.t \quad & g_{\min} \leq g(X(k), t_f) \leq g_{\max} \\ & \forall k = 1, 2, \cdots, N. \end{aligned} \quad (18)$$

Remark: The optimization problem is modeled using CasADi [15] and solved with Ipopt [16].

C. Determining the time-optimal control input signals and switching times

In this subsection, we first use solution from the trajectory generation algorithm above, $X^*(k)$ to find the optimum inputs $u_{1,2}^*(k)$ for each step and the switching times for each input. It was shown in [3] that for the quadrotor point to point maneuver problem, the time-optimal input u_1 is always bang-bang. Further the input u_2 is bang-bang bang-singular, i.e., may include singular arcs.

Definition: The *switching time(s)* is (are) defined as the time(s) when the input switches from one state (minimum/maximum/singular) to another state (minimum/maximum/singular). With the switching times, we can apply the optimality verification algorithm proposed in this paper to verify the optimality of the solution.

The solution of the problem (18) yields the optimal solution $X^*(k)$, which is the optimal state trajectory and the optimal landing time t_f^*. From the state $X^*(k)$, it is straightforward to obtain the inputs ($u_{1,2}^*$) at k-th step and the switching time for each input. Denote $\mathbf{T}_T$ as the set of switching time for u_1 and $\mathbf{T}_R$ as the set of switching time for u_2, then $\mathbf{T}_T$ and $\mathbf{T}_R$ can be obtained as follows:

1) The switch time set $\mathbf{T}_T$: according to the constraint (10), the k-th step control input u_{1k}^* can be calculated as:

$$\begin{gathered} u_{1k}^* = \ddot{x}_k^* \sin(\theta_k^*) + (\ddot{z}_k^* + g)\cos(\theta_k^*) \\ \forall k = 1, 2, \cdots, N. \end{gathered} \quad (19)$$

The switching time for u_1 can be obtained by comparing the u_1 at the k-th step and the $k+1$-th step. The time corresponds to k-th step is $(k-1)t_s$. The input u_1^* switches at $t = (k-1)t_s$ if the following condition is true:

$$\begin{aligned} & (k-1)t_s \in \mathbf{T}_T \\ & \text{if} \begin{cases} u_{1k}^* = u_{1\max}, u_{1(k+1)}^* \neq u_{1k} \\ u_{1k}^* = u_{1\min}, u_{1(k+1)}^* \neq u_{1k} \end{cases} \\ & \forall k = 1, 2, \cdots, N-1. \end{aligned} \quad (20)$$

2) The switching time set $\mathbf{T}_R$: according to the constraint, the k-th step control input u_{2k}^* can be calculated as

$$u_{2k}^* = \dot{\theta}_k \quad (21)$$

The switching time for u_2 can be obtained by comparing the u_2 at the k-th step with the $k+1$-th step and the $k-1$-th step. The input u_2^* switches at $(k-1)t_s$ if the following condition is true:

$$\begin{aligned} & (k-1)t_s \in \mathbf{T}_R \\ & \text{if} \begin{cases} u_{2k}^* = u_{2\max}, u_{2(k+1)}^* \neq u_{2k} \\ u_{2k}^* = u_{2\max}, u_{2(k-1)}^* \neq u_{2k} \\ u_{2k}^* = u_{2\min}, u_{2(k+1)}^* \neq u_{2k} \\ u_{2k}^* = u_{2\min}, u_{2(k-1)}^* \neq u_{2k} \end{cases} \\ & \forall k = 2, 3, \cdots, N-1 \end{aligned} \quad (22)$$

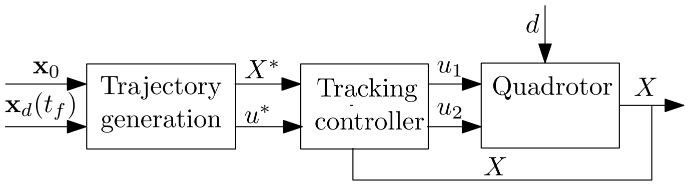

Fig. 4. Block diagram of the overall feedback control structure. In the trajectory generation phase, the initial state and final state is used to generate a reference trajectory $X(k)^*$. In the tracking control phase, a tracking controller is used to generate control input u_1 and u_2 to track the given reference trajectory. d is the disturbance.

D. Trajectory tracking feedback controller design

In the previous subsections, we have designed a time optimal trajectory. Furthermore, the (feed-forward) control input $u^*_{1,2}$ is also obtained from the algorithm as described in the previous subsection. This control input is designed such that the quadrotor tracks the time-optimal trajectory. However, if there is uncertainty in the system dynamics, then directly using the control input from the trajectory will not be effective without a feedback controller.

Thus, in this subsection, a feedback controller is proposed to track the generated time optimal reference trajectory, in conjunction with the feed-forward control input $u^*_{1,2}$ obtained from.

Denote the actual state for the quadrotor as $X(k)$ and the reference trajectory as $X(k)^*$ at k-th step:

$$\begin{aligned} X(k) &= \begin{bmatrix} x & \dot{x} & z & \dot{z} & \theta & \ddot{x} & \ddot{z} & \dot{\theta} \end{bmatrix} \\ X(k)^* &= \begin{bmatrix} x^* & \dot{x}^* & z^* & \dot{z}^* & \theta^* & \ddot{x}^* & \ddot{z}^* & \dot{\theta}^* \end{bmatrix}. \end{aligned} \quad (23)$$

The goal is to design a feedback controller $\mathbf{u}_{fb}$ to track $X^*(k)$ obtained in the previous subsection. In this paper, a feedback linearization controller is proposed inspired a controller for a wheeled mobile robot in [17].

Define three linearized controls u_x, u_z, u_θ and use u_x, u_z, u_θ as system input to represent u_1 and u_2. The system dynamics can be written as

$$\begin{aligned} \ddot{x} &= u_1 \sin(\theta) = u_x \\ \ddot{z} &= u_1 \cos(\theta) - g = u_z \\ \dot{\theta} &= u_2 = u_\theta \end{aligned} \quad (24)$$

The system with u_x,u_z,u_θ as input is linear. A state feedback controller is designed for u_x, u_z, u_θ to track the reference trajectory:

$$\begin{aligned} u_x &= \ddot{x}^* + k_{p1}(x^* - x) + k_{p2}(\dot{x}^* - \dot{x}) \\ u_z &= \ddot{z}^* + k_{p3}(z^* - z) + k_{p4}(\dot{z}^* - \dot{z}) \\ u_\theta &= \dot{\theta}^* + k_{p5}(\theta^* - \theta) \end{aligned} \quad (25)$$

$k_{p1}, k_{p2}, k_{p3}, k_{p4}$ and k_{p4} are positive feedback gains. The system input u_1 can be therefore calculated from u_x, u_z based on (24). Therefore, the controller input can be obtained as:

$$\begin{aligned} u_1 &= u_x \sin(\theta) + (u_z + g)\cos(\theta) \\ u_2 &= u_\theta \\ u_1 &\in [U_{1\min}, U_{1\max}], u_2 \in [U_{2\min}, U_{2\max}] \end{aligned} \quad (26)$$

By proper selection of the gains, it is possible to achieve relatively good tracking performance.

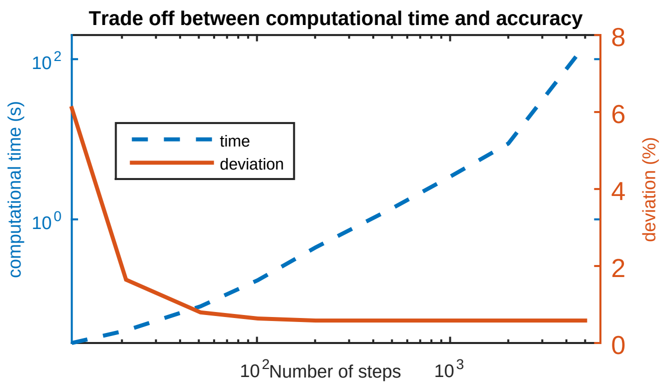

Fig. 5. Plot illustrating the trade off between the accuracy and the computational time. As the step number (N) becomes larger, the deviation from the optimal solution obtained in [3] decreases but the computational time increases.

IV. Simulation Results

We now present simulation results of trajectory generation for a quadrotor onto a translationally moving and tilting platform. We also present a benchmark comparison with the algorithm described in [3] for several stationary point-to-point maneuvers. For all the simulation results, the input constraints on u_1, u_2 are assumed to be as same as those enforced in the benchmark algorithm [3]:

$$\begin{aligned} u_{1\min} = 1m/s^2 \quad &, \quad u_{1\max} = 20m/s^2 \\ u_{2\min} = -10rad/s \quad &, \quad u_{2\max} = 10rad/s \end{aligned} \quad (27)$$

For the controller design, the bounds are set at

$$\begin{aligned} U_{1\min} = 0.5m/s^2 \quad &, \quad U_{1\max} = 25m/s^2 \\ U_{2\min} = -15rad/s \quad &, \quad U_{2\max} = 15rad/s. \end{aligned} \quad (28)$$

The quadrotor position in this simulation include the $x-y$ coordinate and the tilting angle. All quadrotor length coordinates mentioned are in meters and tilting angles are in radians. All the simulations are performed on an Ubuntu computer with an Intel i7 processor and 16 GB RAM. The algorithm codes are written in Python for the benchmark comparison (for comparison of computational time). For the rest of the simulations, the code is written in Matlab.

TABLE I
Cost function (time to perform maneuver, t_f) comparisons with the benchmark algorithm in [3]

desired terminal position (m, m, rad)	benchmark result (s)	proposed result (s)	deviation (%)	computation time (s)
(0,1,0)	0.650	0.645	-0.78%	0.864
(0,3,0)	1.083	1.084	0.09%	0.722
(0,5,0)	1.300	1.346	3.54%	0.903
(3,0,0)	0.890	0.898	0.85%	0.319
(6,0,0)	1.223	1.231	0.65%	0.378
(9,0,0)	1.478	1.488	0.67%	0.342
(12,0,0)	1.694	1.705	0.64%	0.385
(15,0,0)	1.885	1.896	0.58%	0.481
(5,5,0)	1.400	1.4097	0.69%	0.573

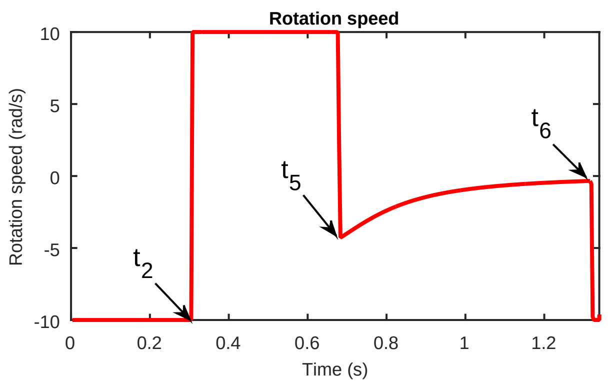

Fig. 6. Plot of the input u_2 for the maneuver shown in Fig. 8 .u_2 has a singular arc between t_5 and t_6. t_2,t_5 and t_6 are the switching times.

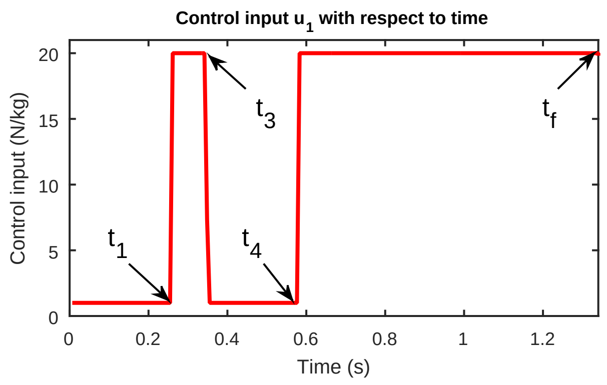

Fig. 7. Plot of the input u_1 for the maneuver shown in Fig. 8. The input u_1 is a bang-bang type control. The switching times are t_1,t_3 and t_4. The final time (end of landing maneuver) is t_f.

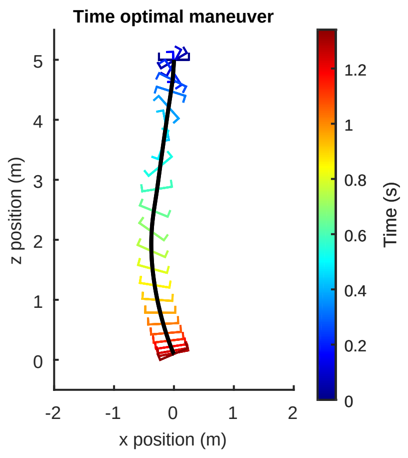

Fig. 8. Trajectory of the quadrotor landing onto a platform. The platform is oscillating sinusoidally in vertical direction and tilting at a constant speed of 0.3 rad/s. The black line is the trajectory of the center of the quadrotor and the colored line shows the quadrotor position and orientation every 50 milliseconds.

A. Benchmark comparison for the trajectory generation algorithm

In this subsection we will present the time-optimal trajectory generation solutions for a set of different horizontal and vertical maneuvers. In [3], the authors listed nine cases of different final positions (stationary). Since the authors in [3] solve the problem based on Pontryagin's minimum principle, the results are considered to be optimal. We first perform a series of simulations to pick the best value for N, i.e., the number of trajectory segments. For every simulation, the final state is set to be (15,0,0) and different values for N are picked from the following set $\mathbf{N}$:

$$\mathbf{N} = [11, 21, 51, 101, 201, 501, 1001, 2001, 5001]. \tag{29}$$

Each solution is then compared with the corresponding result from [3] to determine the (sub) optimality or deviation from t_f. Fig.5 shows the trade-off between the number of steps and the accuracy of the solution obtained. It can be seen that with the increasing resolution of discretization (i.e. N increasing), the computational time increases, while the accuracy of the solution increases. Based on this trade-off, we pick $N = 201$ for all the following simulations.

Next, we use the proposed differentially flat time-optimal trajectory generation algorithm to generate the time optimal trajectory for the 9 different cases and compare them with the benchmark results. Table. I compares the cost function (i.e., time to land) results from the proposed method and the results from the benchmark [3]. For all the cases, the initial values for the optimization variables are all set to be 1, which is an arbitrary picked value. The results from the proposed algorithm are within $\pm 1\%$ of the benchmark result except for the vertical climb to (0,3,0). The computation time for all cases are ≤ 1 second. Thus, we claim that this algorithm is computationally efficient and can maintain relatively high accuracy.

B. Landing onto a tilting and oscillating platform

In this subsection, we demonstrate time optimal landing onto a platform that is vertically oscillating as a sinusoid and pitching (tilting) at constant angular velocity. The initial position for the quadrotor is $(0,5,0)$. The platform motion $[x_d, \dot{x}_d, z_d, \dot{z}_d, \theta_d]$ is defined as:

$$\begin{aligned} &\begin{bmatrix} x_d(t) & \dot{x}_d(t) & z_d(t) & \dot{z}_d(t) & \theta(t)_d \end{bmatrix} \\ = &\begin{bmatrix} 0 & 0 & 0.1\sin(t) & 0.1\cos(t) & -0.3t \end{bmatrix} \end{aligned} \tag{30}$$

Fig. 8 shows the landing trajectory obtained from the proposed trajectory generation algorithm. The time-to-land is 1.340 seconds. Fig.7 and Fig.6 show the control input of u_1 and u_2. It can be seen that there are six switches in total for the control input u_1 and u_2. The optimal solution that satisfies Pontryagin's minimum principle is 1.339 seconds and the deviation of the proposed solution from the optimum is thus 0.04%.

C. Landing onto a tilting and moving away platform

This subsection shows the landing of a quadrotor onto a platform that is pitching and moving away. The initial position of the quadrotor is $(0,5,0)$ and the initial horizontal

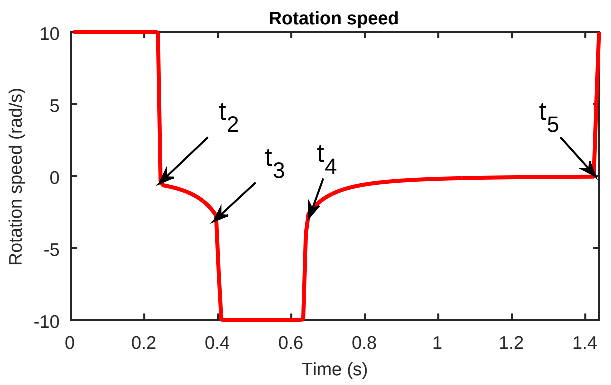

Fig. 9. Plot of the input u_2 for the maneuver shown in Fig.11. The input u_2 has two singular arcs (between t_2 and t_3, between t_4 and t_5) and the switching times are t_2, t_3, t_4 and t_5.

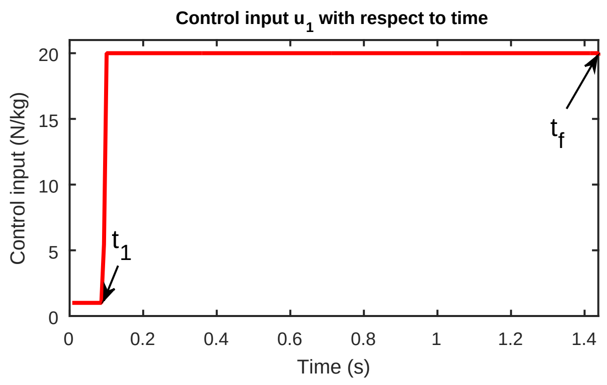

Fig. 10. Plot of the input u_1 for the maneuver shown in Fig.11. The input u_1 switches only once at t_1 and is a bang-bang control.

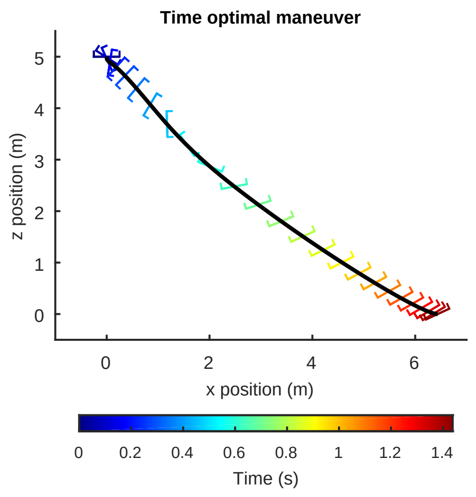

Fig. 11. Trajectory of a quadrotor landing onto a moving platform. The platform is moving away at a constant speed and tilting at a constant speed of 0.3 rad/s. The black line shows the trajectory of the center of the quadrotor and the colored line shows the position and the orientation of the quadrotor every 50 milliseconds.

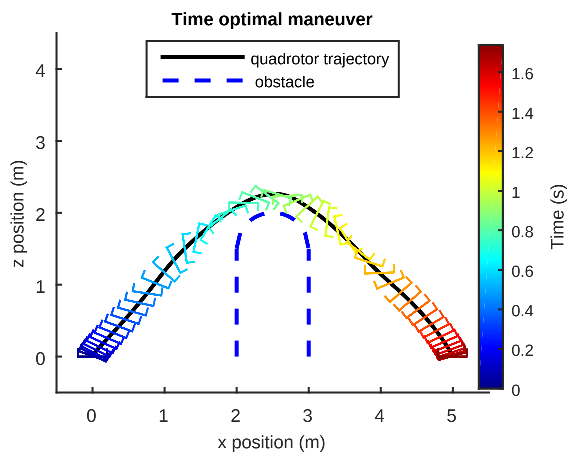

Fig. 12. Plot of quadrotor trajectory for a maneuver that avoids an obstacle, while reaching a fixed position (5,0,0) in a time-optimal manner. The black line is the trajectory of the center of the quadrotor, while the dashed blue line is the obstacle. The colored line shows the quadrotor position and orientation. The quadrotor position is plotted every 70 milliseconds.

distance is from the quadrotor to the platform is 5 meters. The platform motion is defined as:

$$\begin{aligned} & \begin{bmatrix} x_d(t) & \dot{x}_d(t) & z_d(t) & \dot{z}_d(t) & \theta_d(t) \end{bmatrix} \\ = & \begin{bmatrix} 5+t & 1 & 0 & 0 & -0.3t \end{bmatrix} \end{aligned} \tag{31}$$

Fig. 11 shows the quadrotor maneuver trajectory obtained from the trajectory generation algorithm. The optimal time to land found to be 1.437 seconds. Fig. 9 and Fig.10 shows the corresponding input u_1 and u_2 from the optimal solution. It can be seen that in this case, there is only one switch for input u_1 and the control input u_2 has two singular arcs. The optimal solution that satisfies Pontryagin's minimum principle results in a time to land of 1.435 seconds and the error in this case is 0.1%.

D. *Time optimal maneuver with obstacle avoidance*

In this subsection, we demonstrate the performance of the algorithm when the desired terminal position is $(5,0,0)$, while avoiding an obstacle. The obstacle is defined as the union of [1] a circle centered at (2.5,1.5) with a radius of 0.5 meters and [2] a rectangle withfour corners at $(2,0),(2,1.5),(3,1.5)$ and $(3,0)$. Further, we assume that the "safe distance" from any part the quadrotor to the obstacle is 0.25 meters, i.e., the center of the quadrotor should be at least 0.75 meters away from the center of the obstacle. In this case, additional geometric constraints on the trajectory are added to the trajectory generation problem (18):

$$\begin{aligned} (x_k-2.5)^2+(z_k-1.5)^2-0.75^2 \geq 0 \\ \forall k=1,2,\cdots,N \end{aligned} \tag{32}$$

Fig.12 shows the maneuver of the quadrotor. It can be seen that the quadrotor's actual maneuver can avoid collision with the obstacle.

E. *Time optimal maneuver under disturbance*

Next, we present the performance of the overall closed loop system with the time-optimal trajectory generation as well as the feedback controller. The quadrotor is to track the

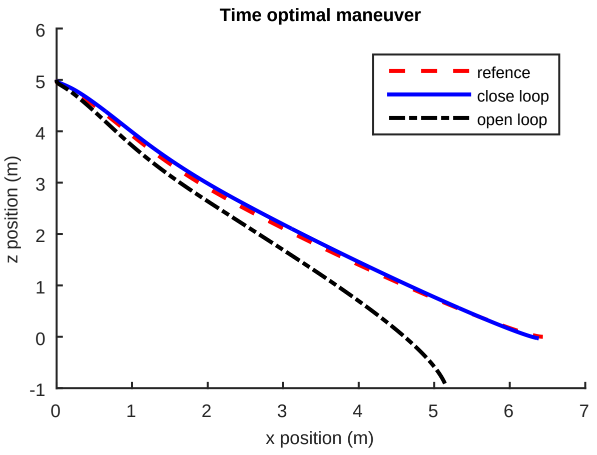

Fig. 13. Plot of the tracking performance (for a time-optimal trajectory) with a feedback controller and in open loop. The actual mass of the quadrotor is 1.25 times larger than the nominal guess, which leads to less acceleration under the same amount of thrust. The open loop controller (dashed dot line) cannot adapt to the difference and it fails to track the reference trajectory (dashed line). The closed loop controller (solid line) tracks the reference trajectory well.

trajectory shown in Fig.11. Assume the actual mass m is 1.25 times larger than the nominal mass $\tilde{m}$. The maximum bound is defined as

$$\begin{aligned} \ddot{x} &= \frac{1}{1.25}u_1\sin(\theta) \\ \ddot{z} &= \frac{1}{1.25}u_1\cos(\theta) - g \end{aligned} \tag{33}$$

Fig. 13 shows the trajectories under feedback controller and the open loop controller. From the result, the closed loop controller is able to track the reference trajectory closely while the open loop controller will deviate from it.

V. Conclusions and Future Work

In this paper, a time-optimal trajectory generation algorithm for a quadrotor is proposed. By discretizing the system dynamics with a fixed sampling time and applying the differential flatness property of the system dynamics, a standard time-optimal trajectory generation problem is transformed into a nonlinear programming problem. Simulation results and comparison with benchmark algorithm show the effectiveness of the proposed method.

In our future work, we plan to implement the trajectory generation algorithm in a real experimental set up on a Hummingbird quadrotor. This method is also promising for application to other differentially flat systems, such as car parking and robotic arm motion planning, which is the subject of our current research.

References

[1] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, 2012.

[2] J. Sanchez-Lopez, S. Saripalli, P. Campoy, J. Pestana, and C. Fu, "Toward visual autonomous ship board landing of a vtol uav," in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, May 2013, pp. 779–788.

[3] M. Hehn, R. Ritz, and R. DAndrea, "Performance benchmarking of quadrotor systems using time-optimal control," *Autonomous Robots*, vol. 33, no. 1-2, pp. 69–88, 2012.

[4] W. Van Loock, G. Pipeleers, and J. Swevers, "Time-optimal quadrotor flight," in *Proc. of the 2013 European Control Conference (ECC)*, 2013, pp. 1788–1792.

[5] K. Shin and N. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, no. 6, pp. 531–541, 1985.

[6] J. E. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985.

[7] R. Verschueren, N. Duijkeren, J. Swevers, and . M. Diehl, "Time-optimal motion planning for n-dof robot manipulators using a path-parametric system reformulation," in *American Control Conference (ACC), 2016*, July 2016, pp. 2092–2097.

[8] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[9] B. Hu, L. Lu, and S. Mishra, "Fast, safe and precise landing of a quadrotor on an oscillating platform," in *American Control Conference, 2015.*, July 2015, pp. 3836–3841.

[10] L.-C. Lai, C.-C. Yang, and C.-J. Wu, "Time-optimal control of a hovering quad-rotor helicopter," *Journal of Intelligent and Robotic Systems*, vol. 45, no. 2, pp. 115–135, 2006.

[11] A. Chamseddine, T. Li, Y. Zhang, C. A. Rabbath, and D. Theilliol, "Flatness-based trajectory planning for a quadrotor unmanned aerial vehicle test-bed considering actuator and system constraints," in *2012 American Control Conference (ACC)*, June 2012, pp. 920–925.

[12] Y. Bouktir, M. Haddad, and T. Chettibi, "Trajectory planning for a quadrotor helicopter," in *Control and Automation, 2008 16th Mediterranean Conference on*, 2008, pp. 1258–1263.

[13] M. Hehn and R. DAndrea, "Real-time trajectory generation for quadrocopters," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 877–892, 2015.

[14] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International journal of control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[15] J. Andersson, "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] A. Wachter and L. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[17] G. Oriolo, A. D. Luca, and M. Vendittelli, "Wmr control via dynamic feedback linearization: design, implementation, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 835–852, Nov 2002.