Online Optimisation Based Backstepping Control Design with Application to Quadrotor

Hao Lu
School of Instrumentation Science and Opto-Electronics Engineering,

Beihang University, Beijing, 100191, People's Republic of China

Cunjia Liu
Department of Aeronautical and Automotive Engineering,

Loughborough University, Loughborough, LE11 3TU,

United Kingdom, E-mail: c.liu5@lboro.ac.uk.

Matthew Coombes
Department of Aeronautical and Automotive Engineering,

Loughborough University, Loughborough, LE11 3TU, United Kingdom

Lei Guo
National Key Laboratory on Aircraft Control Technology,

Beihang University, Beijing, 100191, People's Republic of China

Wen-Hua Chen
Department of Aeronautical and Automotive Engineering,

Loughborough University, Loughborough, LE11 3TU, United Kingdom

In backstepping implementation, the derivatives of virtual control signals are required at each step. This study provides a novel way to solve this problem by combining online optimisation with backstepping design in an outer and inner loop manner. The properties of differential flatness and the B-spline polynomial function are exploited to transform the optimal control problem into a computationally efficient form. The optimisation process generates not only the optimised states but also their finite order derivatives which can be used to analytically calculate the derivatives of virtual control signal required in backstepping design. In addition, the online optimisation repeatedly

performed in a receding horizon fashion can also realise local motion planning for obstacle avoidance. The stability of the receding horizon control scheme is analysed via Lyapunov method which is guaranteed by adding a parametrised terminal condition in the online optimisation. Numerical simulations and flight experiments of a quadrotor unmanned air vehicle are given to demonstrate the effectiveness of the proposed composite control method.

I. Introduction

Backstepping is a recursive nonlinear control method where the feedback control law and the associated Lyapunov functions can be designed following a step-by-step procedure [1]. Therefore, this method has been widely used, for systems such as, aircraft, mobile robots, and manipulators, because their complicated but cascaded dynamic structures can be exploited in this recursive control design [2–6]. The key feature in backstepping algorithms is to construct a virtual control signal in each step to guide the state of the subsequent subsystem so that a stabilizing control law can be found through "step back". Although the principle of backstepping is effective and easy to follow, the conventional design procedure suffers from the complicated analytic derivations of the derivatives of virtual control signal at each step. Especially for high-order systems or multivariable systems like aircraft dynamics, the computation of analytic derivatives may become quite tedious or even infeasible in practice. To alleviate this drawback, the most widely used solution is to pass the desired virtual control signal through a low-pass filter to obtain the filtered virtual control signal and its corresponding derivative [7]. The magnitude and rate constraints can also be imposed on the virtual control signal by using a second-order filter known as command filter in [3, 8]. However, in order to achieve good control performance, the bandwidth and the constraints of each filter need to be carefully tuned, which is not straightforward [6].

On the other hand, in many application domains the tracking control requirement may be coupled with path planning problems, for example, to control an unmanned vehicle or robotic manipulator to avoid obstacles in complex operation environment. The traditional backstepping

control only focuses on the stability or the tacking performance of the system but has no means to incorporate path planning. This necessitates the development of a local motion planning function to adopt newly updated information and repeatedly replan a short term motion profile that satisfies both system dynamics and obstacle-free requirement. A practical control scheme should incorporate both motion planning and tracking control in a hierarchical structure. Specifically, the local motion planner first generates a local obstacle-free reference trajectory online. Then, the tracking controller governs the system dynamics to achieve the replanned motion. The motion planning or trajectory generation has been comprehensively studied in [9–16], which is normally formulated as optimisation problems in a receding horizon fashion. However, the sequential tracking control is usually designed in a simplified linear region. For example, a linear quadratic regulator in [12], a PID controller with feed-forward compensation for anticipation time in [17], and a linear model predictive control technique in [14] are used to cooperate with the local motion planner.

In this paper, instead of dealing the local motion planning and the tracking control separately, we propose an online optimisation based backstepping method to tackle both the local motion planning and the cascaded tracking control problem. A quadrotor is adopted as a representative to simulationally and experimentally demonstrate the effectiveness of the proposed hybrid control strategy. The differential flatness property of the quadrotor and the polynomial parametrisation are used to transfer the formulated optimisation problem into a nonlinear programming (NLP) problem. In this way, not only the optimal local trajectory can be effectively solved online, but also the optimised attitude angles and their derivatives can be generated simultaneously for backstepping. To fulfil the replanned motion, an inner-loop backstepping controller is designed to track the optimised attitude angles, which is more appropriate to deal with the inherent nonlinear dynamics than linear controllers. The main contributions of the proposed control framework include three aspects. Firstly, it can provide the derivatives of the virtual control signals required in backstepping design. Different from approximating the derivatives of the virtual control signals through command filters, these derivatives can be directly calculated by using the optimised states and their derivatives in a polynomial form. This will reduce the phase lag caused by the filters and the complexity in implementation and system tuning. Secondly, a parametrised terminal condition is proposed to guarantee

the stability of the online optimisation and the overall stability of the system under the composite control strategy has also been established. Thirdly, compared to the single-functional backstepping tracking methods, it has the capability to achieve obstacle avoidance. Although demonstrated through a case study on trajectory tracking control of a quadrotor, the proposed composite control strategy is readily applied to other differential flat systems, such as manipulators, land vehicles, maglev systems, cranes, etc., after necessary modifications.

The remainder of this paper is organised as follows. Section II introduces the mathematical model of the quadrotor. In Section III, some preliminaries about differential flatness property, B-spline polynomial function, and obstacle potential function are presented, respectively. The detailed description of the online optimisation for local motion planning is demonstrated in Section IV. Section V is devoted to the design of backstepping tracking controller. Section VI provides the results of numerical simulation and flight experiment to demonstrate the effectiveness of the proposed approach, followed by conclusions in Section VII.

II. Quadrotor modelling

Quadrotor is a kind of vertical take-off and landing (VTOL) unmanned aerial vehicles (UAVs) whose dynamic model has been extensively studied in literatures (e.g. [18–20]). The configuration of the quadrotor and the coordinate systems employed in this paper are briefly shown in Fig. 1. Let $\boldsymbol{S}_{\mathcal{B}} = \{\ O_{\mathcal{B}}\ \ \overrightarrow{\boldsymbol{X}_{\mathcal{B}}}\ \ \overrightarrow{\boldsymbol{Y}_{\mathcal{B}}}\ \ \overrightarrow{\boldsymbol{Z}_{\mathcal{B}}}\ \}$ denote the body-fixed frame with origin at the centre of gravity of the quadrotor and the North-East-Down (NED) inertial frame $\boldsymbol{S}_{\mathcal{I}} = \{\ O_{\mathcal{I}}\ \ \overrightarrow{\boldsymbol{X}_{\mathcal{I}}}\ \ \overrightarrow{\boldsymbol{Y}_{\mathcal{I}}}\ \ \overrightarrow{\boldsymbol{Z}_{\mathcal{I}}}\ \}$ is used to characterise the translational motion. The equations of motion for a rigid body driven by external force $\boldsymbol{F} \in \mathbb{R}^3$ and torque $\boldsymbol{M} \in \mathbb{R}^3$ can be derived from Newton-Euler equations and expressed as follows

$$\begin{cases} m\ddot{\boldsymbol{\zeta}} = mg\boldsymbol{Z}_{\mathcal{I}} + \boldsymbol{R}(\boldsymbol{\eta})\boldsymbol{F} & \text{(1a)} \\ \boldsymbol{J}\dot{\boldsymbol{\Omega}} = -\boldsymbol{\Omega} \times \boldsymbol{J}\boldsymbol{\Omega} + \boldsymbol{M} & \text{(1b)} \end{cases}$$

where $\boldsymbol{\zeta} = [\ x\ \ y\ \ z\ ]^T$, $\boldsymbol{\eta} = [\ \phi\ \ \theta\ \ \psi\ ]^T$, and $\boldsymbol{\Omega} = [\ p\ \ q\ \ r\ ]^T$ denote the quadrotor's inertial positions, attitude angles, and angular rates, respectively. $\boldsymbol{J}$ is the diagonal moment of inertia tensor. The

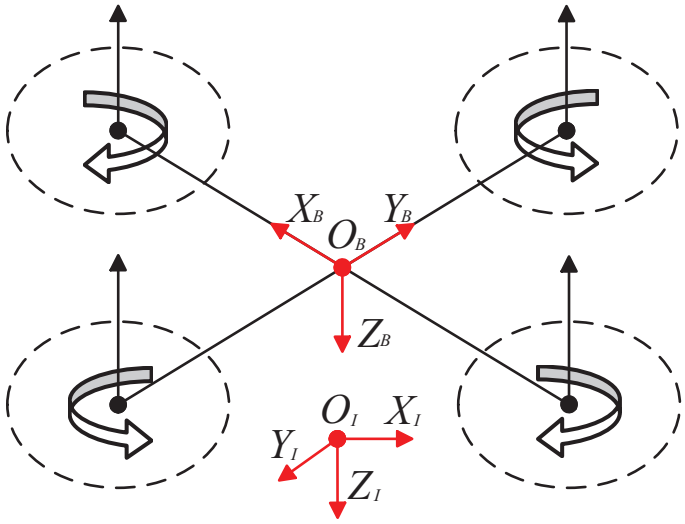


Fig. 1 Quadrotor configuration

transformation matrix from body-fixed frame to inertial frame is given by

$$\boldsymbol{R}(\boldsymbol{\eta}) = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}$$

where the compact notation c denotes cos and s for sin.

In the generic vehicle dynamic model (1), the formulation of the external force and torque determines the type of aircraft. For a quadrotor, the external force and torque can be commonly characterised by thrust force $\boldsymbol{F} = [\ 0 \ \ 0 \ \ -u\ ]^T$ and three control torques $\boldsymbol{M} = [\ u_\phi \ \ u_\theta \ \ u_\psi\ ]^T$. The relations between the four control inputs and the angular speed of the four rotors can be represented as

$$\begin{bmatrix} u \\ u_\phi \\ u_\theta \\ u_\psi \end{bmatrix} = \begin{bmatrix} \rho & \rho & \rho & \rho \\ 0 & -\rho l & 0 & \rho l \\ -\rho l & 0 & \rho l & 0 \\ \kappa & -\kappa & \kappa & -\kappa \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix}$$

where l is the distance from the rotor to the centre of cross frame; ρ and κ are the propeller-to-force and propeller-to-torque scaling factors, respectively. The propellers are driven by DC motors and $w_i, i = 1, 2, 3, 4$ are the motor velocities. The dynamic model of motor can be approximately

described by the following first-order equation

$$\dot{w}_i = k_m(w_i^{des} - w_i)$$

where k_m is the motor time constant and w_i^{des} is the desired angular velocities.

The rotational kinematic relationship between the attitude angle and the angular rate can be derived as follows

$$\dot{\boldsymbol{\eta}} = \boldsymbol{\Phi}(\boldsymbol{\eta})\boldsymbol{\Omega} \tag{2}$$

where

$$\boldsymbol{\Phi}(\boldsymbol{\eta}) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}$$

and it also implies $\boldsymbol{\Omega} = \boldsymbol{\Psi}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}}$, $\boldsymbol{\Psi}(\boldsymbol{\eta}) = \boldsymbol{\Phi}^{-1}(\boldsymbol{\eta})$. Differentiating Eq. (2) and invoking the rotational dynamics in terms of $\boldsymbol{\Omega}$ in (1b), the dynamic model that describes the quadrotor's rotational movement can be rewritten in terms of $\boldsymbol{\eta}$ as

$$\mathbb{J}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + \boldsymbol{C}(\dot{\boldsymbol{\eta}}, \boldsymbol{\eta})\dot{\boldsymbol{\eta}} = \boldsymbol{M}$$

where $\mathbb{J}(\boldsymbol{\eta}) = \boldsymbol{J}\boldsymbol{\Psi}(\boldsymbol{\eta})$ is defined as a pseudoinertia matrix and $\boldsymbol{C}(\dot{\boldsymbol{\eta}}, \boldsymbol{\eta}) = \dot{\mathbb{J}}(\boldsymbol{\eta}) + \boldsymbol{\Psi}(\boldsymbol{\eta}) \times \mathbb{J}\dot{\boldsymbol{\eta}}$ is the Coriolis term [18]. By defining a new pseudocontrol toques $\tilde{\boldsymbol{M}} = [\ \tilde{u}_\phi \ \ \tilde{u}_\theta \ \ \tilde{u}_\psi\ ]^T$ as

$$\tilde{\boldsymbol{M}} = \mathbb{J}^{-1}(\boldsymbol{\eta})(\boldsymbol{M} - \boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}})$$

we obtain

$$\ddot{\boldsymbol{\eta}} = \tilde{\boldsymbol{M}} \tag{3}$$

Through this coordinate transformation, the rotational dynamics is further simplified. Using Eqs.

(1a) and (3), the quadrotor model can be rewritten as follows

$$\begin{cases} \ddot{x} = -\dfrac{1}{m}u(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)) & \text{(4a)} \\ \ddot{y} = -\dfrac{1}{m}u(\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi)) & \text{(4b)} \\ \ddot{z} = -\dfrac{1}{m}u\cos(\phi)\cos(\theta) + g & \text{(4c)} \\ \ddot{\phi} = \tilde{u}_\phi & \text{(4d)} \\ \ddot{\theta} = \tilde{u}_\theta & \text{(4e)} \\ \ddot{\psi} = \tilde{u}_\psi & \text{(4f)} \end{cases}$$

where the system states are $\boldsymbol{x} = [\ x\ \ \dot{x}\ \ y\ \ \dot{y}\ \ z\ \ \dot{z}\ \ \phi\ \ \dot{\phi}\ \ \theta\ \ \dot{\theta}\ \ \psi\ \ \dot{\psi}\ ]^T$, the control inputs are $\boldsymbol{u} = [\ u\ \ \tilde{u}_\phi\ \ \tilde{u}_\theta\ \ \tilde{u}_\psi\ ]^T$, and the outputs $\boldsymbol{y} = [\ x\ \ y\ \ z\ \ \psi\ ]^T$.

III. Preliminaries

A. Differential flatness

Differential flatness is a property of some nonlinear dynamic systems, for which all the system variables can be expressed in terms of a set of specific variables, namely the flat outputs, and their derivatives up to some finite orders [21]. The concept of differential flatness has been exploited to design feedforward control schemes for nonlinear systems [22–24], which normally form a specific combination of a nominal feedforward input and a local feedback controller. In this paper, the differential flatness property is adopted in the receding horizon framework to facilitate the online optimisation. Consider a general nonlinear system with state $\boldsymbol{x}(t) \in \mathbb{R}^n$ and input $\boldsymbol{u}(t) \in \mathbb{R}^m$

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t))$$

This system is differential flat if there exists such a flat output vector $\boldsymbol{z}(t) \in \mathbb{R}^m$ that has the following properties [21–23]:

1) the elements of flat output vector $\boldsymbol{z}$ are differentially independent

2) the flat output vector $\boldsymbol{z}$ can be expressed by a combination of system variables

$$\boldsymbol{z}(t) = \lambda(\boldsymbol{x}(t))$$

3) all the system states and inputs can be expressed in dependence of $\boldsymbol{z}$ and its derivatives up

to first r-th order

$$\begin{aligned} \boldsymbol{x}(t) &= \Upsilon(\boldsymbol{z}(t), \dot{\boldsymbol{z}}(t), \ddot{\boldsymbol{z}}(t), \cdots, \boldsymbol{z}^{(r-1)}(t)) \\ \boldsymbol{u}(t) &= \Gamma(\boldsymbol{z}(t), \dot{\boldsymbol{z}}(t), \ddot{\boldsymbol{z}}(t), \cdots, \boldsymbol{z}^{(r)}(t)) \end{aligned} \tag{5}$$

As for the quadrotor dynamics (4), the system outputs $\boldsymbol{y} = [\ x\ \ y\ \ z\ \ \psi\ ]^T$ are chosen as the flat output vector $\boldsymbol{z}$. The atitude angles and control inputs can be expressed in terms of the flat output components as [14]

$$\begin{cases} \phi = \sin^{-1}(\dfrac{m}{u}(-\ddot{x}\sin\psi + \ddot{y}\cos\psi)) & \text{(6a)} \\ \theta = \tan^{-1}(-\dfrac{\ddot{x}\cos\psi + \ddot{y}\sin\psi}{g - \ddot{z}}) & \text{(6b)} \\ \psi = \psi & \text{(6c)} \end{cases}$$

$$\begin{cases} u = m\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} - g)^2} & \text{(7a)} \\ \tilde{u}_\phi = \ddot{\phi} & \text{(7b)} \\ \tilde{u}_\theta = \ddot{\theta} & \text{(7c)} \\ \tilde{u}_\psi = \ddot{\psi} & \text{(7d)} \end{cases}$$

A complete parametrisation of all system variables also needs the higher derivatives of rotational states $\boldsymbol{\eta}$ and control input u. They can be derived by continuously differentiating Eqs. (6) and (7a). Usually the MATLAB Symbolic Math Toolbox is used to facilitate such kind of derivative calculation. Singularities appear when $g = \ddot{z}$ in Eqs. (6a) and (6b), which means the quadrotor is in free fall. This can be avoided by restricting the input $u > 0$ and pitch and roll angle $-90° < \theta < 90°$ and $-90° < \phi < 90°$ according to Eq. (4c) [12].

B. B-spline parametrisation

A p-th degree B-spline curve $C(\epsilon)$ is a piecewise polynomial function represented by [25]

$$C(\epsilon) = \sum_{i=1}^{n} N_{i,p}(\epsilon) P_i, \quad 0 \le \epsilon \le 1 \tag{8}$$

where P_i, $i = 1, \cdots, n$ are the control points and $N_{i,p}(\epsilon)$ are the basis piecewise polynomial functions defined on a non-decreasing knot sequence

$$\boldsymbol{U} \triangleq \Big[\underbrace{\epsilon_1 \ \cdots \ \epsilon_1}_{p+1} \ \epsilon_{p+2} \ \cdots \ \epsilon_{m-p} \ \underbrace{\epsilon_m \ \cdots \ \epsilon_m}_{p+1} \Big] \tag{9}$$

where ϵ_k, $k = 1, \cdots, m$ are called knots, $\epsilon_1 = 0$, and $\epsilon_m = 1$. The degree of basis functions p, the number of control points n, and the number of knots m are related by

$$m = n + p + 1$$

The i-th B-spline basis function of p-degree is defined as

$$N_{i,0}(\epsilon) = \begin{cases} 1 & \epsilon_i < \epsilon < \epsilon_{i+1} \\ 0 & otherwise \end{cases}$$

$$N_{i,p}(\epsilon) = \frac{\epsilon - \epsilon_i}{\epsilon_{i+p} - \epsilon_i} N_{i,p-1}(\epsilon) + \frac{\epsilon_{i+1,p-1} - \epsilon}{\epsilon_{i+p+1} - \epsilon_{i+1}} N_{i+1,p-1}(\epsilon)$$

and the r-th time derivatives of the basis functions $N_{i,p}(\epsilon)$ is given by

$$N_{i,p}^{(r)}(\epsilon) = p\left(\frac{N_{i,p-1}^{(r-1)}(\epsilon)}{\epsilon_{i+p} - \epsilon_i} - \frac{N_{i+1,p-1}^{(r-1)}(\epsilon)}{\epsilon_{i+p+1} - \epsilon_{i+1}}\right)$$

It should be noted that the derivative order r cannot exceed the selected B-spline curve degree p, otherwise all higher derivatives would be zero. When the denominators involving knot differences become zero, the quotient is defined to be zero.

To facilitate the online optimisation problem, each element of the flat outputs $\boldsymbol{z}(\tau) = [\ x \ \ y \ \ z \ \ \psi\ ]^T$ can be parametrised in terms of the B-spline basis functions as

$$z_j(\tau) = \boldsymbol{N}(\epsilon)\boldsymbol{P}_j, \quad t \leq \tau \leq t + T \tag{10}$$

where $j = 1, 2, 3, 4$ denotes the jth element in $\boldsymbol{z}(t)$, $\boldsymbol{N}(\epsilon) \triangleq [\ N_{1,p}(\epsilon) \ \ N_{2,p}(\epsilon) \ \cdots \ N_{n,p}(\epsilon)\ ]$ is the vector of B-spline basis functions, and $\boldsymbol{P}_j \triangleq [\ P_{1,j} \ \ P_{2,j} \ \cdots \ P_{n,j}\ ]^T$ is the set of control points that are treated as the decision variables in the online optimisation process. For a specified time horizon T, the knot $\epsilon \in [0, 1]$ represents the normalised time index such that the conversion relationship between current time τ and knot ϵ is

$$\tau = t + \epsilon T \tag{11}$$

where t is the sampling time instant. Therefore, the first-order time derivatives of the basis functions are calculated as

$$\frac{dN_{i,p}(\epsilon)}{d\tau} = \frac{dN_{i,p}(\epsilon)}{d\epsilon}\frac{d\epsilon}{d\tau} = \frac{1}{T}\frac{dN_{i,p}(\epsilon)}{d\epsilon} \tag{12}$$

By using Eqs. (10) and (12), the r-th time derivatives of the flat outputs $z_j(t)$, $j = 1, 2, 3, 4$ are derived as

$$z_j^{(r)}(t) = \frac{1}{T^r}\boldsymbol{N}^{(r)}(\epsilon)\boldsymbol{P}_j \tag{13}$$

C. Obstacle potential function

In this paper, obstacle potential function is used to describe the obstacle influence. The obstacle potential function J_{obs} may consist of several contributors, namely $J_{obs} = \sum_{i=1}^{n} J_{obs}^i$ where n is the number of the obstacles being considered. Each contributor can be described by the elliptical-potential-function as [26]

$$J_{obs}^i = \begin{cases} \frac{A}{K}e^{-\alpha K}, & K \geq 1 \\ Ae^{-\alpha K^{1+\frac{1}{\alpha}}}, & 1 > K \geq 0 \end{cases} \tag{14}$$

where α determines how rapidly the potential rises near the obstacle surface and falls off away from the obstacle, A acts as an overall scale factor for the potential, and K is defined as a pseudo-distance in two dimensions from the obstacle:

$$K = \left[\left(\frac{x}{a}\right)^{2n} + \left(\frac{b}{a}\right)^{2}\left(\frac{y}{b}\right)^{2n}\right]^{\frac{1}{2n}} - 1$$

where the semi-major axis a and the semi-minor axis b are determined by the n-ellipse that surrounds the obstacle. The n-ellipse is defined as

$$\left(\frac{x}{a}\right)^{2n} + \left(\frac{b}{a}\right)^{2}\left(\frac{y}{b}\right)^{2n} = 1$$

From Eq. (14), it can be known that the potential function $J_{obs}^i > 0$ is always satisfied. The elliptical formulation is viable for a large class of object shapes by varying the parameter n.

IV. Online optimisation for local motion planning

A. Problem formulation and simplification

A typical objective of autonomous flight control is to regulate the quadrotor to track a predefined reference trajectory $\boldsymbol{y}_r = [\ x_r \ \ y_r \ \ z_r \ \ \psi_r\ ]^T$. However, since the unexpected obstacles might be encountered during the mission, it is essential to replan a real-time local motion profile online to deal with this kind of pop-up threats. In addition, some hard constraints should also be taken into account. Intuitively, receding horizon control (RHC) is considered as a suitable framework to solve the constrained local motion planning problem [9, 10]. At each sampling time, the RHC is a finite horizon open-loop optimal control that can handle constraints explicitly. It is solved online repeatedly, which predicts an optimal control profile based on current states and system model, and applies the first control action in this profile to the system [27]. To generate the obstacle-free local motion profile at time instant t, the open-loop optimal control problem can be formulated into the following general form:

$$\min_{\boldsymbol{x},\,\boldsymbol{u}} J(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{15}$$

Subject to:

$$\dot{\boldsymbol{x}}(\tau) = f(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)), \tag{16a}$$

$$\boldsymbol{lb}_0 \leq \boldsymbol{c}_0(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq \boldsymbol{ub}_0 \tag{16b}$$

$$\boldsymbol{lb}_f \leq \boldsymbol{c}_f(\boldsymbol{x}(t+T), \boldsymbol{u}(t+T)) \leq \boldsymbol{ub}_f \tag{16c}$$

$$\boldsymbol{lb}_\tau \leq \boldsymbol{c}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)) \leq \boldsymbol{ub}_\tau \tag{16d}$$

where $J(\boldsymbol{x}(t), \boldsymbol{u}(t))$ is the cost function to be minimised, $\boldsymbol{x}(\tau)$ is the state trajectory of system dynamics (16a) driven by control input $\boldsymbol{u}(\tau)$ for the time period $\tau \in [t,\ t+T]$, Eqs. (16b) and (16c) are the initial and terminal constraints, respectively, and inequality constraints on trajectory and actuators are expressed through Eq. (16d).

By using the differential flatness property introduced in Section III A, the system states and inputs in Eq. (16a) can be expressed in flat output space as shown in Eq. (5). With this transfor-

mation the optimal control problem is rewritten as

$$\min_{\bar{\boldsymbol{z}}} J(\bar{\boldsymbol{z}}(t)) \tag{17}$$

Subject to:

$$\begin{aligned} \boldsymbol{lb}_0 &\leq \boldsymbol{c}_0(\bar{\boldsymbol{z}}(t)) \leq \boldsymbol{ub}_0 \\ \boldsymbol{lb}_f &\leq \boldsymbol{c}_f(\bar{\boldsymbol{z}}(t+T)) \leq \boldsymbol{ub}_f \\ \boldsymbol{lb}_\tau &\leq \boldsymbol{c}(\bar{\boldsymbol{z}}(\tau)) \leq \boldsymbol{ub}_\tau \end{aligned} \tag{18}$$

where $\bar{\boldsymbol{z}} = (\boldsymbol{z}(t), \dot{\boldsymbol{z}}(t), \ddot{\boldsymbol{z}}(t), \cdots, \boldsymbol{z}^{(r)}(t))$. Since the flat outputs can characterise system dynamics inherently, the dynamic constraint Eq. (16a) in the original OC problem has been removed.

As the quadrotor output $\boldsymbol{y}$ is chosen as the flat output $\boldsymbol{z}$, the trajectory tracking error is defined as $\boldsymbol{z}_e(t) = \boldsymbol{z}(t) - \boldsymbol{z}_r(t)$. To minimise the tracking errors, the cost function J is given as follows

$$J(\boldsymbol{z}_e(t)) = g(\boldsymbol{z}_e(t+T)) + \int_t^{t+T} L(\boldsymbol{z}_e(\tau))d\tau \tag{19}$$

where $g(\boldsymbol{z}_e(t+T)) = \frac{1}{2}\boldsymbol{z}_e(t+T)^T\boldsymbol{R}\boldsymbol{z}_e(t+T)$ is the terminal cost to guarantee the stability by penalising the tracking error at the end of prediction horizon, $L(\boldsymbol{z}_e(\tau)) = \boldsymbol{z}_e(\tau)^T\boldsymbol{Q}\boldsymbol{z}_e(\tau) + J_{obs}(\tau)$ is the running cost, J_{obs} is the obstacle potential function introduced in Section III C, and $\boldsymbol{Q}$ and $\boldsymbol{R}$ are positive definite weighting matrices.

Furthermore, to transform the optimal control problem from an infinite-dimensional space to a finite one, a suitable parametrisation of the flat outputs is also required. By using the B-spline polynomial introduced in Section III B, the flat output vector can be parametrised as

$$\begin{aligned} \boldsymbol{z}(\tau) &= [\ z_1(\tau)\ \ z_2(\tau)\ \ z_3(\tau)\ \ z_4(\tau)\ ]^T \\ &= [\ \boldsymbol{N}(\epsilon)\boldsymbol{P}_1\ \ \boldsymbol{N}(\epsilon)\boldsymbol{P}_2\ \ \boldsymbol{N}(\epsilon)\boldsymbol{P}_3\ \ \boldsymbol{N}(\epsilon)\boldsymbol{P}_4\ ]^T \\ &= \boldsymbol{\Lambda}(\epsilon)\bar{\boldsymbol{P}} \end{aligned} \tag{20}$$

where $\boldsymbol{\Lambda}(\epsilon) = diag(\boldsymbol{N}(\epsilon), \boldsymbol{N}(\epsilon), \boldsymbol{N}(\epsilon), \boldsymbol{N}(\epsilon))$ is the matrix of B-spline basis functions and $\bar{\boldsymbol{P}} = [\ \boldsymbol{P}_1^T\ \ \boldsymbol{P}_2^T\ \ \boldsymbol{P}_3^T\ \ \boldsymbol{P}_4^T\ ]^T$ is the control point vector. Then the running cost and terminal cost terms

in Eq. (19) can be written as

$$\begin{aligned} & L(\boldsymbol{z}_e(\tau)) \\ =&(\boldsymbol{z}(\tau)-\boldsymbol{z}_r(\tau))^T\boldsymbol{Q}(\boldsymbol{z}(\tau)-\boldsymbol{z}_r(\tau))+J_{obs}(\tau) \\ =&\bar{\boldsymbol{P}}^T\boldsymbol{\Lambda}(\epsilon)^T\boldsymbol{Q}\boldsymbol{\Lambda}(\epsilon)\bar{\boldsymbol{P}}-2\bar{\boldsymbol{P}}^T\boldsymbol{\Lambda}(\epsilon)^T\boldsymbol{Q}\boldsymbol{z}_r(\tau)+\boldsymbol{z}_r(\tau)^T\boldsymbol{Q}\boldsymbol{z}_r(\tau)+J_{obs}(\tau) \end{aligned} \tag{21}$$

and

$$\begin{aligned} & g(\boldsymbol{z}_e(t+T)) \\ =&\frac{1}{2}(\boldsymbol{z}(t+T)-\boldsymbol{z}_r(t+T))^T\boldsymbol{R}(\boldsymbol{z}(t+T)-\boldsymbol{z}_r(t+T)) \\ =&\frac{1}{2}\bar{\boldsymbol{P}}^T\boldsymbol{\Lambda}(1)^T\boldsymbol{R}\boldsymbol{\Lambda}(1)\bar{\boldsymbol{P}}-\bar{\boldsymbol{P}}^T\boldsymbol{\Lambda}(1)^T\boldsymbol{R}\boldsymbol{z}_r(t+T)+\frac{1}{2}\boldsymbol{z}_r^T(t+T)\boldsymbol{R}\boldsymbol{z}_r(t+T) \end{aligned} \tag{22}$$

where $\epsilon = 1$ in Eq. (22) is obtained from the conversion relationship (11). Specifically, choosing $\epsilon = 1$ obtains $\tau = t + T$ from $\tau = t + \epsilon T$. Furthermore, defining

$$\boldsymbol{Q}_t = \int_t^{t+T} \boldsymbol{\Lambda}(\epsilon)^T\boldsymbol{Q}\boldsymbol{\Lambda}(\epsilon)d\tau$$

$$\boldsymbol{G}_t = -2\int_t^{t+T} \boldsymbol{\Lambda}(\epsilon)^T\boldsymbol{Q}\boldsymbol{z}_r(\tau)d\tau$$

$$\boldsymbol{Q}_1 = \frac{1}{2}\boldsymbol{\Lambda}(1)^T\boldsymbol{R}\boldsymbol{\Lambda}(1)$$

$$\boldsymbol{G}_1 = -\boldsymbol{\Lambda}(1)^T\boldsymbol{R}\boldsymbol{z}_r(t+T)$$

$$C = \frac{1}{2}\boldsymbol{z}_r^T(t+T)\boldsymbol{R}\boldsymbol{z}_r(t+T)+\int_t^{t+T}\boldsymbol{z}_r^T(\tau)\boldsymbol{Q}\boldsymbol{z}_r(\tau)+J_{obs}(\tau)d\tau$$

the cost function (19) is expressed in the following compact form

$$J(\bar{\boldsymbol{P}}) = \bar{\boldsymbol{P}}^T(\boldsymbol{Q}_t+\boldsymbol{Q}_1)\bar{\boldsymbol{P}}+\bar{\boldsymbol{P}}^T(\boldsymbol{G}_t+\boldsymbol{G}_1)+C \tag{23}$$

where the control point vector $\bar{\boldsymbol{P}}$ is the only underlying variable that needs to be optimised. Thus, the optimal control problem (17) is further simplified as

$$\min_{\bar{\boldsymbol{P}}} J(\bar{\boldsymbol{P}}) \tag{24}$$

As to the inequality constraints, such as velocity constraints, acceleration constraints, and altitude angle constraints, they can also be parametrised in terms of $\bar{\boldsymbol{P}}$ and incorporated into one constraint

$$\boldsymbol{lb} \leq \mathbf{c}(\bar{\boldsymbol{P}}) \leq \boldsymbol{ub} \tag{25}$$

The final cost function (23) and the corresponding constraint (25) have been highly simplified compared to the original ones because they are expressed in terms of control point vector $\bar{\boldsymbol{P}}$. This will significantly relieve the computational burden of solving the optimal control problem online.

Moreover, the initial boundary conditions should be satisfied to ensure that the replanned motion can start smoothly from the current vehicle states. According to the derivatives of the B-spline curve at the endpoint, the first three control points $P_{1,j}$, $P_{2,j}$, and $P_{3,j}$ of each flat output element can be determined as:

$$\begin{aligned} P_{1,j} &= z_j(0) \\ P_{2,j} &= \tfrac{\dot{z}_j(0)\epsilon_{p+2}}{p}T + P_{1,j} \\ P_{3,j} &= \tfrac{\ddot{z}_j(0)\epsilon_{p+2}\epsilon_{p+3}}{(p-1)p}T^2 + \tfrac{\epsilon_{p+2}+\epsilon_{p+3}}{\epsilon_{p+2}}P_{2,j} - \tfrac{\epsilon_{p+3}}{\epsilon_{p+2}}P_{1,j} \end{aligned}$$

where $z_j(0)$, $\dot{z}_j(0)$ and $\ddot{z}_j(0)$ are the current position, velocity and acceleration provided by corresponding sensors and the parameters ϵ_{p+2} and ϵ_{p+3} are the knots defined in (9). Since this relationship can further scale down the dimension of the control point vector by three, the actual number of variables to be optimised in the online optimisation is $j(n-3)$. Through all these transformations, the original optimal control problem has been formulated into an NLP problem which can be effectively solved by using the MATLAB *fmincon* function. On the other hand, if the obstacle avoidance is not required, i.e. there is no the nonlinear obstacle potential function $J_{obs}(t)$, the NLP problem is indeed a quadratic programming (QP) problem, for which many faster and more convenient solvers are available.

B. Stability analysis and implementation

Solving the optimisation problem (17) at time instant t can obtain an open-loop optimal solution $\boldsymbol{z}_e^*(\tau)$ for the time period $\tau \in [t,\ t+T]$. Since in the moving horizon fashion the optimal motion profile is only implemented in the time interval $\xi \in [t,\ t+\delta)$ where δ is the sampling time, we have

$$\boldsymbol{z}_e(\xi) = \boldsymbol{z}_e^*(\xi), \quad \xi \in [t,\ t+\delta) \tag{26}$$

At time $t+\delta$, the receding horizon control needs to take the current flat output error $\boldsymbol{z}_e(t+\delta)$ and solve the optimisation problem again.

The stability of the receding horizon control in terms of states and inputs has been extensively investigated [27–30]. However, in this paper the receding horizon control problem is parametrised in the flat output space with B-spline polynomials. To analyse its stability, we will first introduce the terminal condition in the flat output space in the following Lemma.

Lemma. Suppose that the online optimisation problem (17) in differential flatness algorithm is feasible at the time $t \geq 0$. Then the local motion planning problem under the receding horizon control is the asymptotically stable if the following terminal condition is satisfied:

$$\dot{g}(\boldsymbol{z}_e^*(\xi+T)) + L(\boldsymbol{z}_e^*(\xi+T)) \leq 0, \;\; \xi \in [t,\; t+\delta) \tag{27}$$

Proof. Choose the cost function (19) as a Lyapunov function candidate

$$V(\boldsymbol{z}_e(t)) = J(\boldsymbol{z}_e(t))$$

For a time period $\xi \in [t,\; t+\delta)$, the Lyapunov function $V(\boldsymbol{z}_e(\xi))$ is nonincreasing due to the following equation:

$$\begin{aligned} V(\boldsymbol{z}_e(\xi)) =& V(\boldsymbol{z}_e^*(\xi)) \\ =& V(\boldsymbol{z}_e^*(t)) - \int_t^{\xi} \boldsymbol{z}_e^*(\tau) d\tau \\ =& V(\boldsymbol{z}_e(t)) - \int_t^{\xi} \boldsymbol{z}_e(\tau) d\tau \\ \leq& V(\boldsymbol{z}_e(t)) \end{aligned}$$

The difference of the Lyapunov function for two time instants t and $t+\delta$ is given by

$$\begin{aligned} & V(\boldsymbol{z}_e^*(t+\delta)) - V(\boldsymbol{z}_e^*(t)) \\ =& g(\boldsymbol{z}_e^*(t+T+\delta)) + \int_{t+\delta}^{t+T+\delta} L(\boldsymbol{z}_e^*(\tau)) d\tau - g(\boldsymbol{z}_e^*(t+T)) - \int_t^{t+T} L(\boldsymbol{z}_e^*(\tau)) d\tau \\ =& g(\boldsymbol{z}_e^*(t+T+\delta)) - g(\boldsymbol{z}_e^*(t+T)) - \int_t^{t+\delta} L(\boldsymbol{z}_e^*(\tau)) d\tau + \int_{t+T}^{t+T+\delta} L(\boldsymbol{z}_e^*(\tau)) d\tau \end{aligned} \tag{28}$$

Integrating Eq. (27) over the time period $[0,\; \delta]$ gives

$$g(\boldsymbol{z}_e^*(t+T+\delta)) - g(\boldsymbol{z}_e^*(t+T)) + \int_{t+T}^{t+T+\delta} L(\boldsymbol{z}_e^*(\tau)) d\tau \leq 0 \tag{29}$$

Substituting Eq. (29) into Eq. (28) and using Eq. (26) yield

$$V(\boldsymbol{z}_e(t+\delta)) - V(\boldsymbol{z}_e(t)) \leq -\int_t^{t+\delta} L(\boldsymbol{z}_e(\tau)) d\tau < 0 \;\; (\boldsymbol{z}_e(t) \neq 0) \tag{30}$$

It can be seen that $V(\boldsymbol{z}_e(t))$ is monotonically nonincreasing and bounded below by zero. So the following result can be obtained by using the inequality (30) repeatedly:

$$\begin{aligned} V(\boldsymbol{z}_e(t)) - V(\boldsymbol{z}_e(0)) \leq & -\int_0^t L(\boldsymbol{z}_e(\tau))d\tau \\ = & -\int_0^t \boldsymbol{z}_e(\tau)^T \boldsymbol{Q} \boldsymbol{z}_e(\tau) + J_{obs}(\tau) d\tau \\ < & -\int_0^t \boldsymbol{z}_e(\tau)^T \boldsymbol{Q} \boldsymbol{z}_e(\tau) d\tau \end{aligned}$$

where the integral term on the right side of the inequality converges and is low bounded. According to the proofs in [29–31], it follows that $\boldsymbol{z}_e(t) \to 0$ as $t \to \infty$. Hence, the actual system output $\boldsymbol{y}$ under the receding horizon control will converge to the reference trajectory $\boldsymbol{y}_r$ asymptotically. □

To embed the terminal condition in the optimisation, it also needs to parametrise Eq. (27) in term of $\bar{\boldsymbol{P}}$. Assuming $\dot{\boldsymbol{z}}_r = 0$, the two items in Eq. (27) are separately expressed as

$$\begin{aligned} & \dot{g}(\boldsymbol{z}_e(t+T)) \\ = & (\dot{\boldsymbol{z}}(t+T))^T \boldsymbol{R}(\boldsymbol{z}(t+T) - \boldsymbol{z}_r(t+T)) \\ = & \bar{\boldsymbol{P}}^T \bar{\boldsymbol{\Lambda}}(1)^T \boldsymbol{R} \boldsymbol{\Lambda}(1) \bar{\boldsymbol{P}} - \bar{\boldsymbol{P}}^T \bar{\boldsymbol{\Lambda}}(1)^T \boldsymbol{R} \boldsymbol{z}_r(t+T) \end{aligned} \tag{31}$$

and

$$\begin{aligned} & L(\boldsymbol{z}_e(t+T)) \\ = & (\boldsymbol{z}(t+T) - \boldsymbol{z}_r(t+T))^T \boldsymbol{Q} (\boldsymbol{z}(t+T) - \boldsymbol{z}_r(t+T)) + J_{obs}(t+T) \\ = & \bar{\boldsymbol{P}}^T \boldsymbol{\Lambda}(1)^T \boldsymbol{Q} \boldsymbol{\Lambda}(1) \bar{\boldsymbol{P}} - 2\bar{\boldsymbol{P}}^T \boldsymbol{\Lambda}(1)^T \boldsymbol{Q} \boldsymbol{z}_r(t+T) + \boldsymbol{z}_r(t+T)^T \boldsymbol{Q} \boldsymbol{z}_r(t+T) + J_{obs}(t+T) \end{aligned} \tag{32}$$

where $\bar{\boldsymbol{\Lambda}}(1) = diag(\frac{1}{T}\dot{\boldsymbol{N}}(1), \frac{1}{T}\dot{\boldsymbol{N}}(1), \frac{1}{T}\dot{\boldsymbol{N}}(1), \frac{1}{T}\dot{\boldsymbol{N}}(1))$ and $\frac{1}{T}\dot{\boldsymbol{N}}(\epsilon)$ denotes the first-order derivative of the basis function defined in (12). Thus, the parametrised terminal condition is obtained by summing up Eq. (31) and Eq. (32). Consequently, by incorporating it into the integrated constraint (25), the terminal condition is enforced and satisfied in the optimisation process such that the stability of online optimisation for local motion planning is guaranteed.

Given the optimised solution $\bar{\boldsymbol{P}}^*$, the replanned obstacle-free trajectory $\boldsymbol{z}^*$ and the corresponding optimised attitude angle $\boldsymbol{\eta}^*$ can be calculated base on the flat output parametrisation (10) and (13) and the differential flatness property (6). In addition, the optimised force control input u^* obtained from Eq. (7a) can be directly used for altitude tracking control. However, the inner-loop attitude control requires much faster control rate that the online optimisation cannot satisfy at the

current stage. Therefore, it is essential to design an inner-loop controller to track the optimised attitude angle.

V. Backstepping design for tracking control

The optimisation problem is solved in the receding horizon framework and the optimised motion profile is replanned in a local region. In order to fulfil the replanned motion, an attitude controller needs to be designed to track the desired attitude angle $\boldsymbol{\eta}^*$ asymptotically. The backstepping method is adopted to deal with this nonlinear control problem.

Define the attitude tracking error as

$$\boldsymbol{e}_1 = \boldsymbol{\eta} - \boldsymbol{\eta}^* \tag{33}$$

Consider the Lyapunov function candidate as

$$V_1 = \frac{1}{2}\boldsymbol{e}_1^T\boldsymbol{e}_1 \tag{34}$$

Taking the time derivative of V_1 along the trajectory of (2) yields

$$\dot{V}_1 = \boldsymbol{e}_1^T(\boldsymbol{\Phi}(\boldsymbol{\eta})\boldsymbol{\Omega} - \dot{\boldsymbol{\eta}}^*) \tag{35}$$

where angular rate $\boldsymbol{\Omega}$ is selected as the virtual control signal for $\boldsymbol{e}_1$ subsystem. To make $\dot{V}_1$ negative, the desired value of $\boldsymbol{\Omega}$ can be constructed as

$$\boldsymbol{\Omega}_d = \boldsymbol{\Psi}(\boldsymbol{\eta})(-k_1\boldsymbol{e}_1 + \dot{\boldsymbol{\eta}}^*) \tag{36}$$

This results in

$$V_1 = -k_1\boldsymbol{e}_1^T\boldsymbol{e}_1 \tag{37}$$

For the next step, the actual angular rate $\boldsymbol{\Omega}$ is forced to track the desired angular rate $\boldsymbol{\Omega}_d$. Define the angular rate tracking error as

$$\boldsymbol{e}_2 = \boldsymbol{\Omega} - \boldsymbol{\Omega}_d \tag{38}$$

Let the Lyapunov function candidate be

$$V_2 = V_1 + \frac{1}{2}\boldsymbol{e}_2^T\boldsymbol{e}_2 \tag{39}$$

Taking the time derivative of V_2 along the vector field of (1b) and (2) yields

$$\dot{V}_2 = -k_1 \boldsymbol{e}_1^T \boldsymbol{e}_1 + \boldsymbol{e}_2^T(-\boldsymbol{J}^{-1}\boldsymbol{\Omega} \times \boldsymbol{J}\boldsymbol{\Omega} + \boldsymbol{J}^{-1}\boldsymbol{M} - \dot{\boldsymbol{\Omega}}_d) \tag{40}$$

If the actual control torque is chosen as

$$\boldsymbol{M} = \boldsymbol{J}(-k_2\boldsymbol{e}_2 + \dot{\boldsymbol{\Omega}}_d + \boldsymbol{J}^{-1}\boldsymbol{\Omega} \times \boldsymbol{J}\boldsymbol{\Omega}) \tag{41}$$

the time derivative of the Lyapunov function V_2 satisfies

$$\dot{V}_2 = -k_1 \boldsymbol{e}_1^T \boldsymbol{e}_1 - k_2 \boldsymbol{e}_2^T \boldsymbol{e}_2 < 0, \;\; (k_1, k_2 > 0 \; and \; \boldsymbol{e}_1, \boldsymbol{e}_2 \neq 0) \tag{42}$$

Thus, the equilibrium $(\boldsymbol{e}_1, \boldsymbol{e}_2) = 0$ is globally stable and the tracking errors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ converge to zero asymptotically.

The overall stability of the closed-loop system must combine both the stability analyses of online optimisation and backstepping control. Define the total Lyapunov function as

$$V_{total} = V(\boldsymbol{z}_e) + V_2(\boldsymbol{e}_1, \boldsymbol{e}_2)$$

The difference of the total Lyapunov function is given by

$$\begin{aligned} V_{total}(t+\delta) - V_{total}(t) = &\int_t^{t+\delta} \dot{V}(\boldsymbol{z}_e(\tau)) + \dot{V}_2(\boldsymbol{e}_1(\tau), \boldsymbol{e}_2(\tau))d\tau \\ =&V(\boldsymbol{z}_e(t+\delta)) - V(\boldsymbol{z}_e(t)) + \int_t^{t+\delta} \dot{V}_2(\boldsymbol{e}_1(\tau), \boldsymbol{e}_2(\tau))d\tau \end{aligned} \tag{43}$$

Substituting the stability results Eq. (30) and Eq. (42) obtains $V_{total}(t+\delta) - V_{total}(t) < 0$ for $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{z}_e \neq 0$. Recursively, it can be inferred that the total Lyapunov function candidate V_{total} monotonously decreases with respect to time such that the overall stability of the closed-loop system under the composite online optimisation based backstepping controller is guaranteed.

The time derivative of $\boldsymbol{\Omega}_d$ is used in (41) and it can be calculated by differentiating Eq. (36) as

$$\dot{\boldsymbol{\Omega}}_d = \dot{\boldsymbol{\Psi}}(\boldsymbol{\eta})(-k_1\boldsymbol{\delta}_1 + \dot{\boldsymbol{\eta}}^*) - k_1\boldsymbol{\Omega} + k_1\boldsymbol{\Psi}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}}^* + \boldsymbol{\Psi}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}}^* \tag{44}$$

As mentioned in the introduction, it is not a straightforward task to get the derivative of virtual control signal in the backstepping scheme. Taking Eq. (44) for an example, without the information of $\dot{\boldsymbol{\eta}}^*$ and $\ddot{\boldsymbol{\eta}}^*$ the so-called command filter must be introduced in the design process and such kind of design can be found in [32]. The first command filter is used to produce the filtered version of $\boldsymbol{\eta}^*$

and its derivative and the second command filter is used to filter $\boldsymbol{\Omega}_d$ to generate $\dot{\boldsymbol{\Omega}}_d$. Although the good performance under the command filtered backstepping control can be achieved, it is difficult to tune the cascaded bandwidth and the constraints of the command filters. Furthermore, to prove the stability, an additional auxiliary linear filter is also integrated to compensate for the unachieved portion of the desired virtual control signal caused by using command filter [8], which makes the design process and implementation become more complicated. In this paper, $\dot{\boldsymbol{\eta}}^*$ and $\ddot{\boldsymbol{\eta}}^*$ can be directly calculated through the optimised control point vector $\bar{\boldsymbol{P}}^*$ due to the differential flatness and B-spline parametrisation properties. An example of how to calculate $\dot{\phi}$ is shown in the following. Taking the time derivative of Eq. (6a) yields

$$\dot{\phi} = -\frac{m}{\sqrt{1-\sin(\phi)^2}}\left(\frac{\dddot{x}}{u}\sin(\phi) - \frac{\dddot{y}}{u}\cos(\phi) + (\ddot{x}\cos(\psi)+\ddot{y}\cos(\psi))\frac{\dot{\psi}}{u} - (\ddot{x}\sin(\psi)-\ddot{y}\cos(\psi))\frac{\dot{u}}{u^2}\right) \quad (45)$$

where

$$\dot{u} = \frac{m}{\sqrt{\ddot{x}^2+\ddot{y}^2+(\ddot{z}-g)^2}}(\ddot{x}\dddot{x} + \ddot{y}\dddot{y} + (\ddot{z}-g)\dddot{z}) \quad (46)$$

is derived by differentiating Eq. (7a) with respect to time. Thus, it can seen that $\dot{\phi}$ is characterized by the flat outputs $\boldsymbol{z} = [\ x \ \ y \ \ z \ \ \psi\ ]^T$ and their derivatives up to third-order. Based on the B-spline parametrisation properties (13) and (20), the derivatives of the flat outputs are expressed as

$$\boldsymbol{z}^{(n)} = \frac{1}{T^n}\boldsymbol{\Lambda}^{(n)}\bar{\boldsymbol{P}}^*, \quad n = 1, 2, 3 \quad (47)$$

By virtue of the optimised $\bar{\boldsymbol{P}}^*$, $\dot{\phi}$ can be calculated by reversing the above steps. Other required state derivatives, namely $\dot{\boldsymbol{\eta}}^*$ and $\ddot{\boldsymbol{\eta}}^*$, can also be calculated in a similar fashion. As a result, $\dot{\boldsymbol{\Omega}}_d$ is directly obtained from Eq. (44) overcoming the derivative calculation problem of virtual control signal. An overall system diagram showing the connections between the online optimisation, the backstepping controller and the dynamic system is depicted in Fig. 2.

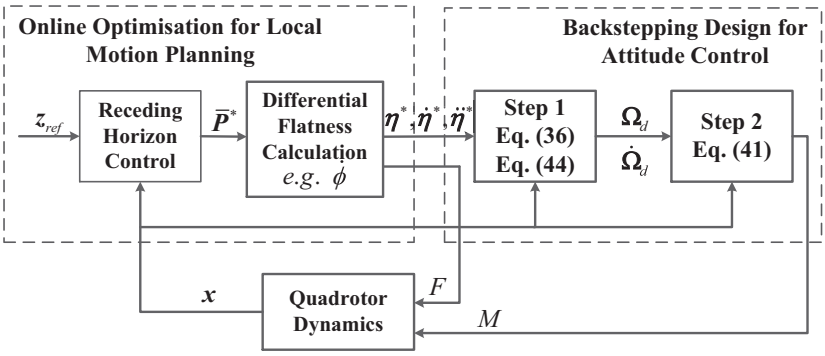


Fig. 2 Control block diagram

VI. Simulation and experiment

A. Numerical simulation

In this section, numerical simulation and flight experiment are carried out to validate the proposed online optimisation based backstepping control strategy. The implementation of the simulation is conducted by using two computers running the Simulink and MATLAB environment separately. One computer uses Simulink to execute a full dynamic model of the quadrotor together with the backstepping attitude controller and the differential flatness calculation module. The other computer solves the online optimisation problem using *fmincon* function within MATLAB where the C-MEX functions are used to improve efficiency. The two computers are connected via LAN (Local Area Network) using UDP (User Datagram Protocol). Moreover, the two parts of the simulation are synchronised and performed in real time to include the computational delay arisen from the online optimisation. The model parameters of the quadrotor used in the simulation are listed in Table 1 and the controller parameters are summarised in Table 2.

Table 1 Quadrotor parameters

Notation	Value	Notation	Value
m, kg	2	$\rho, N/rpm^2$	3×10^{-6}
$J, kg \cdot m^2$	$diag\{5 \times 10^{-3}, 5 \times 10^{-3}, 9 \times 10^{-3}\}$	$\kappa, N \cdot m/rpm^2$	1.5×10^{-7}
l, m	0.22	$k_m, 1/s$	20

The quadrotor is required to track a three-dimensional (3D) square trajectory which is clockwise,

Table 2 Controller parameters

Notation	Description	Value	Notation	Description	Value
p	Order of polynomials	4	n	Number of control points	7
m	Number of knots	12	ϵ_6, ϵ_7	Mid knots	$0.1, 0.9$
T, s	Prediction horizon	3	δ, s	Sampling time	0.05
A	Potential field scaler factor	60	α	Potential field decay rate	-5
a, b, m	Obstacle semi axes	$1, 1$	k_1, k_2	Backstepping control gains	$10, 20$

starting from and ending at the origin. For simplicity, the heading angle command ψ_r remains constant. During the route, there are two pop-up obstacles that appear and are detected by the on-board sensors. The obstacles are assumed to be cylindrical with a radius of $1\,m$ and are much taller than the operating altitude of the quadrotor such that the quadrotor must fly around, not over, the obstacles. According to the description of elliptical-potential-function in Section III C, the two-dimensional (2D) potential field about an obstacle with the centre at the origin is depicted in Fig. 3. As the quadrotor comes close to the obstacle surface, the value of potential would increase

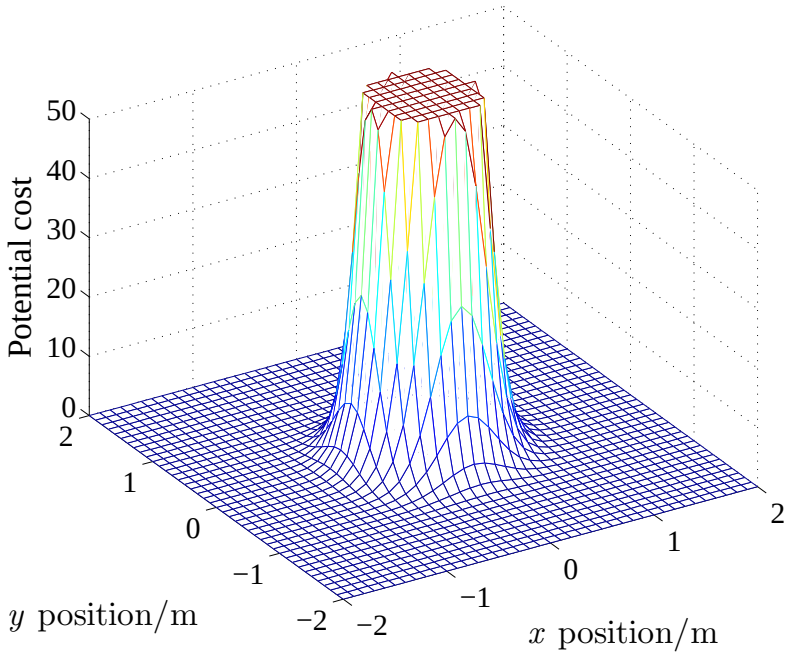


Fig. 3 Potential field

dramatically. In order to minimise the cost function, the replanned trajectory is away from the obstacles as shown in Fig. 4 in the 3D space. Obviously, adequate obstacle clearance distance is created for the quadrotor. If necessary, this safe clearance distance may be increased or decreased by adjusting the obstacle potential function J_{obs}. To be more specific, the projection of horizontal motion is shown in Fig. 5. It can be seen that the quadrotor under the proposed composite online optimisation based backstepping control law is able to successfully avoid the obstacles encountered as expected, smoothly pass all the abrupt corners, and closely track the predetermined box trajectory. The detailed simulation results are presented in Fig. 6 which demonstrates the time histories of $x-y-z$ positions, attitude angles, and control inputs. As seen in Fig. 6(b), the curves of the optimised angles and the actual angles are almost coincident, which indicates the good tracking performance of the inner-loop backstepping attitude controller.

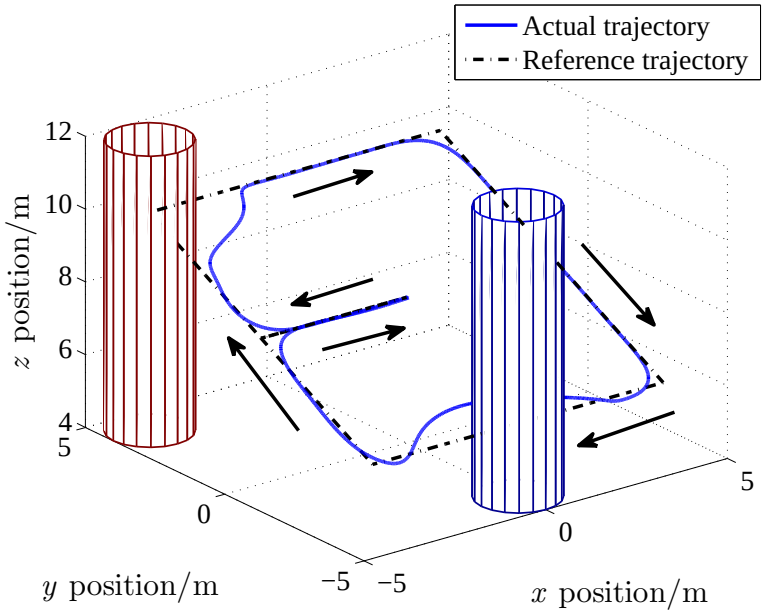


Fig. 4 Trajectory tracking results

B. Indoor flight test

To demonstrate the effectiveness of the proposed control algorithm, flight tests have been conducted in an indoor flight test environment. The test facility consists of VICON motion capture system, ground station and UAV platforms, which has been used to support different research

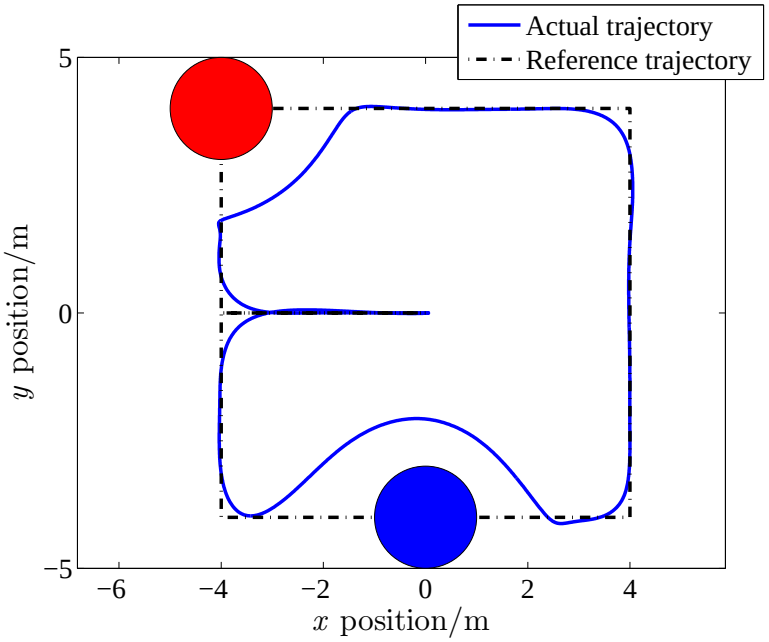
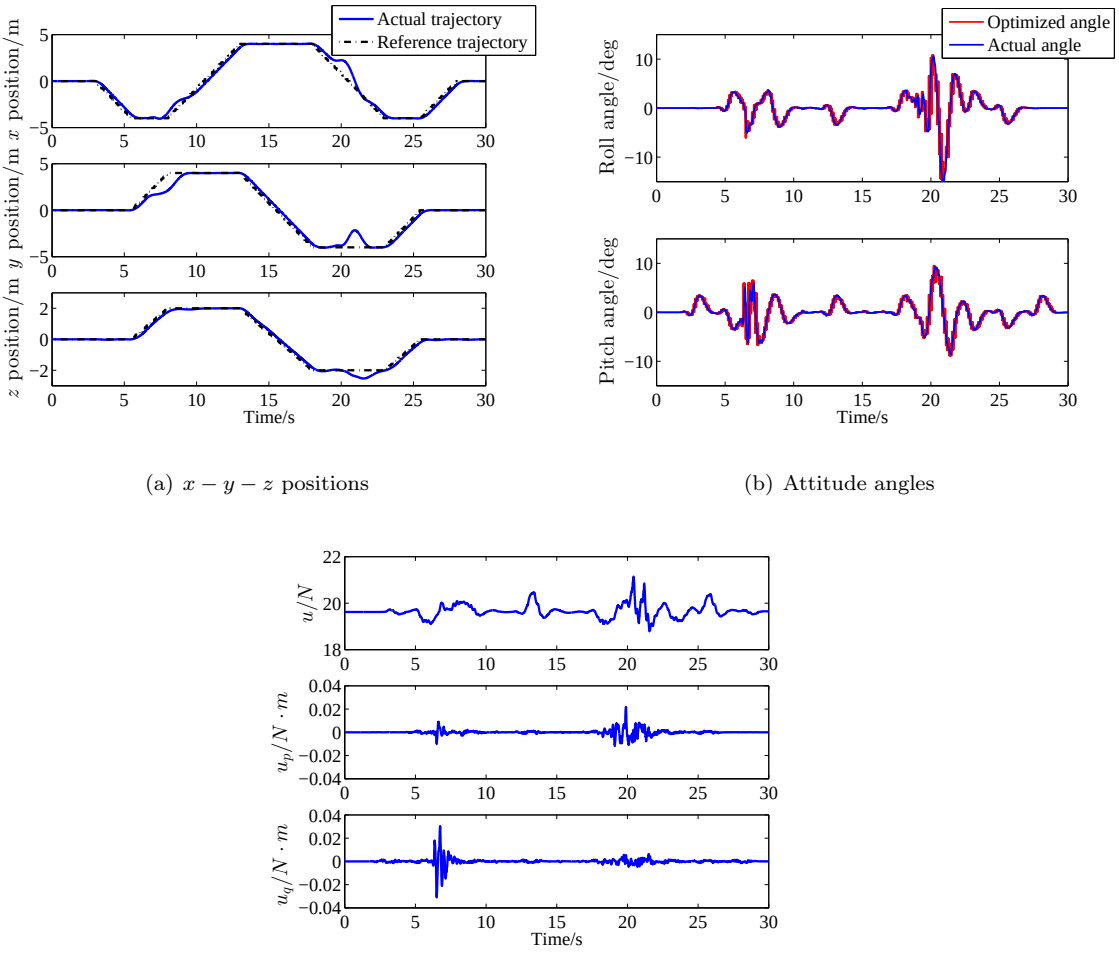


Fig. 5 Horizontal motion projection

projects (see e.g. [33, 34] for details). In this work, the quadrotor platform adopts the Asctec Pelican quadrotor, as shown in Fig. 7. It has two onboard ARM7 microprocessors, i.e. the low level processor and the high level processor. The low level possessor handles sensor data processing, data fusion as well as a basic attitude control algorithm which enables pilot controlled flight from a radio transmitter. The high level processor can be used to implement custom control algorithms which can communicate to the ground station through a wireless serial modem. MATLAB/Simulink can be used to program the high level processor of the Pelican. With the help of Asctec Simulink toolkit, the Simulink models are able to access all the sensor information, direct motor speed control, and XBee serial modem. Therefore, the Simulink based control algorithms can be translated into C-code which is then uploaded on to the Pelican's high level processor with necessary modifications [35]. Compared to the numerical simulation in subsection VI A, the backstepping attitude control algorithm is implemented on the high level processor. On the other hand, the ground station contains the online optimisation module running in MATLAB and the differential flatness calculation running in Simulink such that it sends attitude $\boldsymbol{\eta}$ and derivatives $\dot{\boldsymbol{\eta}}$, $\ddot{\boldsymbol{\eta}}$ commands to the high level processor via the Xbee serial modem. Given that GPS signals are not available when the quadrotor is operated indoor, the VICON motion capture system is employed to provide the position and ve-



(a) $x - y - z$ positions

(b) Attitude angles

(c) Control inputs

Fig. 6 Simulation results with respect to time

locity over ethernet to the ground station. In summary, an overall structure of the test environment is shown in Fig. 8.

Due to the limitation of indoor area, the box trajectory and the radius of cylindrical obstacles are smaller than those executed in the simulation. Also, the height and heading are controlled to remain constant such that the test results are 2D as shown in Fig. 9. The tracking results of $x - y$ positions and attitude angles are given in Fig. 10. It can be observed that the quadrotor can successfully follow the predefined trajectory and avoid the obstacles in the flight test. Considering the measurement errors and atmospheric disturbances existing in the actual flight, the actual pitch and roll angles do not track the command angles that generated by online optimisation as closely as in the simulated case, which also leads to the fluctuations in actual trajectory. Nevertheless, the

tracking accuracies of attitude angle and trajectory remain within a favourable range.

Fig. 7 Asctec Pelican

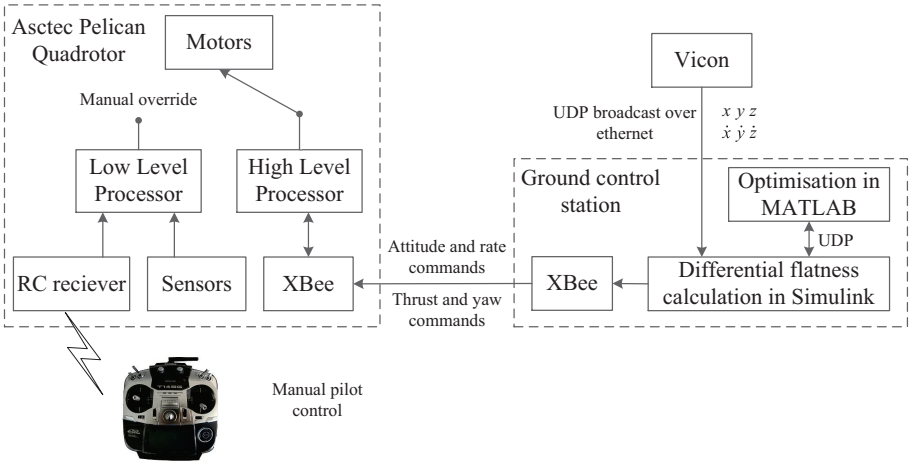


Fig. 8 Structure of the test environment using the Asctec Pelican quadrotor

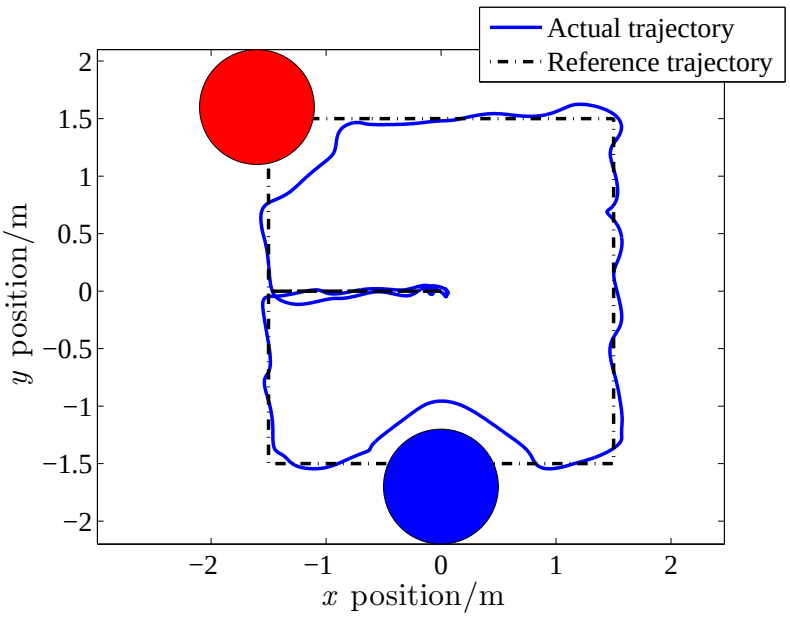
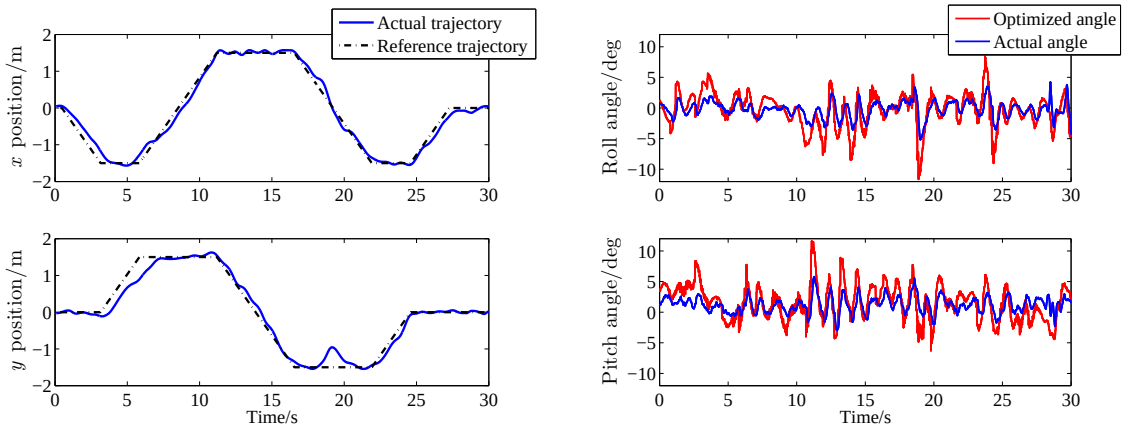


Fig. 9 Experimental trajectory tracking results



(a) $x - y$ positions

(b) Attitude angles

Fig. 10 Experimental results with respect to time

VII. Conclusion

This paper proposes an online optimisation based backstepping control design which is a hierarchical control framework suitable for differentially flat systems and is demonstrated on a quadrotor unmanned vehicle. The main motivation is driven by the difficulties in calculating the derivatives of virtual control signals in traditional backstepping approaches. The proposed method is a straightforward and convenient way to solve this problem, because the derivatives of virtual control signals

can be directly calculated based on the optimised states, expressed in polynomial forms, which are repeatedly generated from online optimisation. The intrinsic properties of differential flatness and B-spline polynomials make the optimisation problem amenable to real-time solution and easy to include the terminal condition for assuring the closed-loop stability. Therefore, the local motion planning function is coupled with tracking control to achieve obstacle-free flight. Numerical simulation and flight test results show that the quadrotor under the proposed composite control strategy avoids the obstacles successfully and achieves high-quality control performance.

Acknowledgements

This work was supported by National Basic Research Program of China (2012CB720003), National Natural Science Foundation of China (61127007, 61121003, and 61320106010). Hao Lu would like to thank the Chinese Scholarship Council for supporting his study in the United Kingdom.

References

[1] Krstic, M., Kokotovic, P. V., Kanellakopoulos, I.: 'Nonlinear and adaptive control design' (John Wiley & Sons, Inc., New York, 1995)

[2] Fierro, R., Lewis, F. L.: 'Control of a nonholonomic mobile robot: backstepping kinematics into dynamics', Proc. 34th Conf. Decision and Control, New Orleans, LA, December 1995, pp. 3805–3810

[3] Farrell, J., Sharma, M., Polycarpou, M.: 'Backstepping-based flight control with adaptive function approximation', *J. Guid. Control Dyn.*, 2005, 28, (6), pp. 1089–1102

[4] Xia, Y., Zhu, Z., Fu, M.: 'Back-stepping sliding mode control for missile systems based on an extended state observer', *IET Control Theory Appl.*, 2011, 5, (1), pp. 93–102

[5] Park, S. H., Han S. I.: 'Robust-tracking control for robot manipulator with deadzone and friction using backstepping and RFNN controller', *IET Control Theory Appl.*, 2011, 5, (12), pp. 1397–1417

[6] Lu, H., Liu, C., Guo, L., Chen, W.-H.: 'Flight control design for small-scale helicopter using disturbance-observer-based backstepping', *J. Guid. Control Dyn.*, 2015, 38, (11), pp. 2235–2240

[7] Yip, P. P., Hedrick, J. K.: 'Adaptive dynamic surface control: a simplified algorithm for adaptive backstepping control of nonlinear systems', *Int. J. Control*, 1998, 71, (5), pp. 959–979

[8] Farrell, J., Polycarpou, M., Sharma, M.: 'Command filtered backstepping', *IEEE Trans. Autom. Control*, 2009, 54, (6), pp. 1391–1395

[9] Berry, A. J., Howitt, J., Gu, D.-W., Postlethwaite, I.: 'A continuous local motion planning framework for unmanned vehicles in complex environments', *J. Intell. Robot. Syst.*, 2011, 66, (4), pp. 477–494

[10] Mahadevan, R., Agrawal, S. K., Doyle, F. J.: 'Differential flatness based nonlinear predictive control of fed-batch bioreactors', *Control Eng. Pract.*, 2001, 9, (8), pp. 889–899

[11] Flores, M., Milam, M.: 'Trajectory generation for differentially flat systems via NURBS basis functions with obstacle avoidance', Proc. American Control Conf., Minneapolis, MN, June 2006, pp. 5769–5775

[12] Cowling, I. D., Yakimenko, O. A., Whidborne, J. F., Cooke, A. K.: 'Direct method based control system for an autonomous quadrotor', *J. Intell. Robot. Syst.*, 2010, 60, (2), pp 285–316

[13] Chamseddine, A., Zhang, Y., Rabbath, C. A., Join, C., Theilliol, D.: 'Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle', *IEEE Trans. Aerosp. Electron. Syst.*, 2012, 48, (4), pp. 2832–2848

[14] Liu, C., Chen, W.-H.: 'A practical receding horizon control framework for path planning and control of autonomous VTOL vehicles', 4th European Conf. for Aerospace Science Avionics, St. Petersburg, Russia, July 2011, pp. 1–11

[15] Suryawan, F., De Doná, J., Seron, M.: 'Splines and polynomial tools for flatness-based constrained motion planning', *Int. J. Syst. Sci.*, 2013, 43, (8), pp. 1396–1411

[16] Prodan, I., Olaru, S., Bencatel, R., Borges de Sousa, J., Stoica, C., Niculescu, S.-I.: 'Receding horizon flight control for trajectory tracking of autonomous aerial vehicles', *Control Eng. Pract.*, 2013, 21, (10), pp. 1334–1349

[17] Berry, A. J., Howitt, J., Gu, D.-W., Postlethwaite, I.: 'Continuous local motion planning & control for Micro-Air-Vehicles in complex environments', AIAA Guidance, Navigation, and Control Conf., Toronto, Canada, August 2010, pp. 1–16

[18] Kendoul, F., Lara, D., Fantoni-Coichot, I., Lozano, R.: 'Real-time nonlinear embedded control for an autonomous quadrotor helicopter', *J. Guid. Control Dyn.*, 2007, 30, (4), pp. 1049–1061

[19] Hoffmann, G. M., Huang, H., Waslander, S. L., Tomlin, C. J.: 'Quadrotor helicopter trajectory tracking control: theory and experiment', AIAA Guidance, Navigation, and Control Conf. and Exhibit, Hilton Head, South Carolina, August 2007, pp. 1–20

[20] Mellinger, D., Michael, N., Kumar, V.: 'Trajectory generation and control for precise aggressive maneuvers with quadrotors', *Int. J. Robot. Res.*, 2012, 31, (5), pp. 664–674

[21] Fliess, M., Lévine, J., Martin, P., Rouchon, P.: 'Flatness and defect of non-linear systems: introductory theory and examples', *Int. J. Control*, 1995, 61, (6), pp. 1327–1361

[22] Hagenmeyer, V., Delaleau, E.: 'Exact feedforward linearization based on differential flatness', *Int. J.*

Control, 2003, 76, (6), pp. 537–556

[23] Hagenmeyer, V., Delaleau, E.: 'Continuous-time non-linear flatness-based predictive control: an exact feedforward linearisation setting with an induction drive example', *Int. J. Control*, 2008, 81, (10), pp. 1645–1663

[24] Ferrin, J., Leishman, R., Beard, R., McLain, T.: 'Differential flatness based control of a rotorcraft for aggressive maneuvers', 2011 IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco, CA, September 2011, pp. 2688–2693

[25] Piegl, L., Tiller, W.: 'The NURBS book 2nd' (Springer Verlag, Berlin, 1997)

[26] Volpe, R., Khosla, P.: 'Artificial potentials with elliptical isopotential contours for obstacle avoidance', Proc. 26th Conf. Decision and Control, Los Angeles, CA, December 1987, pp. 180–185

[27] Mayne, D. Q., Rawlings, J. B., Rao, C. V., Scokaert, P. O. M.: 'Constrained model predictive control: stability and optimality', *Automatica*, 2000, 36, (6), pp. 789–814

[28] Chen, W.-H., Ballance, D. J., O'Reilly, J.: 'Model predictive control of nonlinear systems: computational burden and stability', *IEE Proc. Control Theory Appl.*, 2000, 147, (4), pp. 387–394

[29] Chen, H., Allgöwer, F.: 'A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability', *Automatica*, 1998, 34, (10), pp. 1205-1217

[30] Gu, D., Hu, H.: 'Receding horizon tracking control of wheeled mobile robots', *IEEE Trans. Control Syst. Technol.*, 2006, 14, (4), pp. 743–749

[31] Chen, W.-H., Gawthrop, P. J.: 'Constrained predictive pole-placement control with linear models', *Automatica*, 2006, 42, (4), pp. 613–618

[32] Zuo, Z.: 'Trajectory tracking control design with command-filtered compensation for a quadrotor', *IET Control Theory Appl.*, 2010, 4, (11), pp. 2343–2355

[33] Liu, C., Chen, W.-H., Andrews, J.: 'Tracking control of small-scale helicopters using explicit nonlinear MPC augmented with disturbance observers', *Control Eng. Pract.*, 2012, 20, (3), pp. 258–268

[34] Coombes, M., Eaton, W., McAree, O., Chen, W.-H.: 'Development of a generic network enabled autonomous vehicle system', 2014 UKACC Int. Conf. Control, Loughborough, UK, July, 2014, pp. 621–627

[35] (2016, Mar.) *AscTec Research Home* [Online], Available: http://wiki.asctec.de/display/AR/AscTec+Research+Home