robotics

MDPI

Article

A Backstepping Approach to Nonlinear Model Predictive Horizon for Optimal Trajectory Planning

Younes Al Younes and Martin Barczyk *

Department of Mechanical Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada
* Correspondence: mbarczyk@ualberta.ca

Abstract: This paper presents a novel trajectory planning approach for nonlinear dynamical systems; a multi-rotor drone, built on an optimization-based framework proposed by the authors named the Nonlinear Model Predictive Horizon. In the present work, this method is integrated with a Backstepping Control technique. The goal is to remove the non-convexity of the optimization problem in order to provide real-time computation of reference trajectories for the vehicle which respects its dynamics while avoiding sensed static and dynamic obstacles in the environment. Our method is applied to two models of multi-rotor drones to demonstrate its flexibility. Several simulation and hardware flight experiments are presented to validate the proposed design and demonstrate its performance improvement over earlier work.

Keywords: trajectory generation and planning; nonlinear model predictive approach; backstepping control; dynamic obstacle avoidance; unmanned aerial vehicles

Citation: Al Younes, Y.; Barczyk, M. A Backstepping Approach to Nonlinear Model Predictive Horizon for Optimal Trajectory Planning. *Robotics* **2022**, *11*, 87. https://doi.org/10.3390/robotics11050087

Academic Editor: Sunan Huang

Received: 26 July 2022
Accepted: 29 August 2022
Published: 31 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

Planning collision-free trajectories for autonomous unmanned vehicles operating within unknown, dynamic, 3D, geometrically complex, and GPS-denied environments is a challenging and exciting research problem for both academia and industry. For agile drone systems, generating efficient trajectories in real-time requires using trajectory planning methods which respect the vehicle's dynamics and input constraints as part of the prediction process. Researchers are now studying trajectory planning approaches which can take these considerations into account, for instance receding horizon-based methods [1–3].

Trajectory planning provides a parameterized path from a starting configuration to a terminal setpoint while avoiding obstacles. It is sometimes called motion planning and mistakenly referred to as path planning, since trajectory planning is a superset of path planning by generating reference kinematics over the entire path instead of geometric paths only [4]. Planning algorithms have received much attention from robotics researchers, where most of the published algorithms fall under one of the following categories: search-based, sampling-based, artificial potential field, artificial intelligence and optimization-based methods.

Search-based methods, a.k.a. grid-based methods, use search algorithms to find the best possible collision-free path in a discretized graph of grids. Some of the widely used graph search algorithms include Dijkstra algorithm [5], A* [6], Lifelong Planning A* (LPA*) [7], D* Lite [8], Anytime Repairing A* (ARA*) [9], and Hybrid-state A* [10]. The sampling-based approaches randomly sample a space of possible robot configurations distributed in space and use search algorithms to form a directed graph of collision-free motions. Probabilistic Road-Map (PRM) [11], Rapidly-Exploring Random Tree (RRT) [12], RRT*, and Rapidly-Exploring Random Graph (RRG) methods [13] are some examples of this thoroughly studied approach in the literature. The artificial potential field method developed in [14] plans a path to a goal that avoids collisions by assigning an attractive force to the desired goal and repulsive forces to obstacles [15]. A variety of artificial

intelligence-based algorithms for path planning have been proposed in the literature, for instance, Artificial Neural Network (ANN) [16], Genetic Algorithm (GA) [17], Ant Colony Optimization (ACO) [18], Particle Swarm Optimization (PSO) [19], and Simulated Annealing (SA) [20] algorithms. Reference [21] presents a comprehensive review of artificial intelligence-based methods for path planning up to 2018. A recent line of research involves trajectory generation for flexible space robots based on deterministic artificial intelligence, which aims to use the dynamic equations of the system to autonomously determine an optimal reference trajectory (c.f. [22] and the references therein).

However, the above-mentioned motion planning algorithms have analytical and practical limitations, such as:

- Absence or limited consideration of the internal and external constraints imposed by the system and dynamic environment;
- Lack of repeatability in generating trajectories between a starting and ending configuration for a fixed initial vehicle state and environment scenario;
- Computational inefficiency in regenerating paths while moving between the start and terminal point;
- For some approaches, generating non-smooth paths that lead to jerky motions and create inefficiency in the vehicle's power draw [23].

Recently, optimization-based motion planning methods have gained researchers' attention due to their ability to resolve some or all of the above-mentioned limitations. The best-known optimization-based methods for motion planning are Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [24] and Stochastic Trajectory Optimization for Motion Planning (STOMP) [25]. The former uses gradient-based optimization while the latter uses stochastic optimization, and both produce collision-free trajectories that satisfy given constraints, but are computationally expensive. In addition, CHOMP is prone to getting trapped in local minima, where it returns infeasible or sub-optimal solutions. A very recent publication on optimization-based trajectory generation is [26], which uses Pontryagin's maximum principle to efficiently generate slew trajectories for a spacecraft.

Our proposed approach, called the Nonlinear Model Predictive Horizon (NMPH) [27], uses optimization for planning smooth, optimal, consistent, and computationally efficient trajectories which respect the internal and external constraints of the vehicle. This approach compensates for the nonlinearities of the trajectory planning dynamics and hence reduces or removes the non-convexity of the optimization problem. This is performed by combining the nonlinear plant model with the Backstepping Control (BSC) method within the optimization problem. It also accounts for static and moving obstacles as constraints in the optimization problem.

In ref. [27], we proposed an NMPH formulation which used Feedback Linearization (FBL) within its dynamics to compensate for nonlinearities. In that work, the state augmentation process required to make the system state feedback linearizable created numerical difficulties due to the need to obtain the second-order time derivative of the total thrust of the drone. This difficulty with using the FBL design steered us to find an alternative feedback design methodology to be used within the NMPH design, namely BSC. This new NMPH-BSC formulation compensates for the system nonlinearities and guarantees global asymptotic stability of the closed-loop system within the optimization problem used to predict the reference trajectories for the drone vehicle.

The backstepping control algorithm was originally developed by [28] to design adaptive controllers for a special class of nonlinear dynamical systems. Backstepping is a recursive Lyapunov-based design technique applied to a nonlinear strict-feedback system with a series of cascaded subsystems, in which the choice of the Lyapunov functions for the subsystems guarantees the global asymptotic stability of the overall controlled system [29].

One of the most important advantages of the BSC method is the systematic procedure that the designer can follow to construct a feedback control law based on an appropriate

choice of Lyapunov functions [30]. This step-by-step procedure uses the system's dynamics model, which accounts for the system nonlinearities.

The flexibility of the BSC methodology and its ability to guarantee global asymptotic stability [31,32] make it an excellent candidate to be used inside the NMPH approach, as this integration compensates for the drone's nonlinearities and leads to a less non-convex optimization problem.

To demonstrate the versatility of our NMPH-BSC approach over the earlier NMPH-FBL [27], a more detailed model of the drone vehicle dynamics, which includes linear and angular velocity drag forces and rotor gyroscopic effects, is employed. The optimization problem is also used to predict both the reference trajectory as well as its rates of change, which provides higher-quality tracking performance.

The research contributions of this paper are:

- Implementing the BSC method within NMPH to compensate for nonlinearities in order to reduce the non-convexity of the trajectory generation optimization problem;
- Demonstrating the versatility of the NMPH-BSC approach by using both a simplified and a higher-fidelity dynamics model of the drone;
- Using the NMPH optimization problem to predict both the reference trajectory as well as its rates of change for the onboard flight trajectory controller;
- Validating and evaluating the performance of the proposed approach in both simulation and hardware drone flight experiments.

The remainder of this paper is arranged as follows: the design of the NMPH-BSC for two variants of drone dynamics models is presented in Section 2. Section 3 evaluates the proposed designs in simulation and hardware flight tests. Section 4 summarizes the paper and proposes future work.

2. Problem Formulation

The Nonlinear Model Predictive Horizon (NMPH) is a recent approach to path planning proposed by the authors in [27]. In simple terms, NMPH integrates a linearizing feedback law to reduce the non-convexity of the optimization problem handled by a Nonlinear Model Predictive Control (NMPC) algorithm. NMPH generates optimal reference trajectories for the closed-loop system which steers an aerial drone's 3D position, heading angle, and their rates of change. The original NMPH [27] relied on state feedback linearization [33], which required the addition of integral states to satisfy the conditions of the method, translating to the need for numerical differentiation of measured outputs. In the present work, we replace feedback linearization with Backstepping Control (BSC) [28] to compensate for the system's nonlinearities. The recursive structure of BSC provides stable response of a dynamic system and makes it more robust to parameter uncertainties.

In this section, an overview of the NMPH framework is presented, both high-fidelity and simplified drone dynamics models are presented, and finally a backstepping design for each model is derived and integrated into the NMPH.

2.1. Nonlinear Model Predictive Horizon

NMPH is an optimization-based method whose objective is to generate optimal reference trajectories for closed-loop nonlinear systems. The optimization problem of NMPH considers the nonlinear plant model, nonlinear control law, and user-specified constraints to continuously solve for reference trajectories which are fed to the closed-loop system and which respect its dynamics. An overview of the NMPH architecture is shown in Figure 1.

Embedding the nonlinear control law in the optimization problem is performed to decrease the nonlinearity of the closed-loop system dynamics. Consequently, this will reduce the non-convexity of the optimization problem and enhance convergence and computational time of the solution, enabling real-time trajectory generation for purposes such as motion planning and environmental exploration.

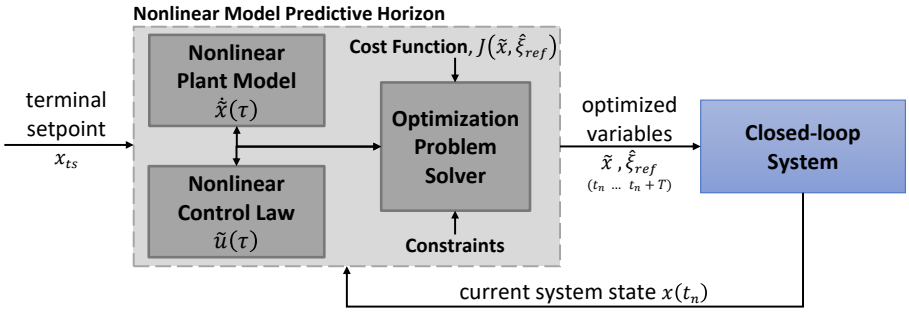


Figure 1. NMPH Architecture. The NMPH optimization process (gray box) considers models of the nonlinear plant dynamics and nonlinear control law, as well as constraints representing performance limits and environmental obstacles. The actual closed-loop system (blue box) tracks the optimized reference trajectory calculated by NMPH.

For the closed-loop system, which is depicted by a blue box in Figure 1, assume that a nonlinear plant is controlled by a nonlinear feedback law as follows:

$$\dot{x} = f(x, u) \tag{1a}$$
$$\xi = h(x) \tag{1b}$$
$$u = g(x, \xi_{ref}), \tag{1c}$$

where $x \in X \subseteq \mathbb{R}^{n_x}$, $u \in U \subseteq \mathbb{R}^{n_u}$, and $\xi \in \Xi \subseteq \mathbb{R}^{n_\xi}$ are the system states, inputs, and outputs, respectively. The system outputs considered in this work are the vehicle's 3D position and heading angle and their rates, such that $\Xi \subseteq X$. The reference trajectory $\xi_{ref} \in \Xi$ is generated by our proposed algorithm, as will be presented below in (2). The system dynamics are represented by a smooth map $f : X \times U \to X$, while the feedback control law is the smooth map $g : X \times \Xi \to U$ which makes the system output to follow a reference trajectory ξ_{ref}. In this work the control law is designed using the Backstepping Control technique.

For the NMPH, which is represented by the gray box in Figure 1, a copy of (1) plus assigned constraints are used in the optimization problem to compute two predictions over a finite time horizon, from the current state x to a terminal stabilization setpoint x_{ss}. The first prediction is the predicted system state trajectory $\tilde{x}$, which includes the predicted output trajectory $\tilde{\xi}$ as a subset. The second one is the estimated reference trajectory $\hat{\xi}_{ref}$, which is used as a (continuously updated) reference trajectory for the actual closed-loop system.

The online NMPH optimization problem for a stabilization setpoint x_{ss} is shown in Equation (2) [34]. Let t_n, $n = 0, 1, 2, \cdots$ represent successive sampling times. At every sampling time, the optimization problem solves for $\tilde{x}$ and $\hat{\xi}_{ref}$ as long as $\|x_{ss} - x(t_n)\| \geq \Delta$:

$$\min_{\tilde{x}, \hat{\xi}_{ref}} \left(J(\tilde{x}, \hat{\xi}_{ref}) = E(\tilde{x}(t_n + T)) + \int_{t_n}^{t_n+T} L(\tilde{x}(\tau), \hat{\xi}_{ref}(\tau))\, d\tau \right) \tag{2}$$

$$\begin{aligned}
\text{subject to} \quad & \tilde{x}(t_n) = x(t_n)\,, && \text{(2a)}\\
& \dot{\tilde{x}}(\tau) = f(\tilde{x}(\tau), \tilde{u}(\tau)), && \text{(2b)}\\
& \tilde{u}(\tau) = g(\tilde{x}(\tau), \hat{\xi}_{ref}(\tau)), && \text{(2c)}\\
& \tilde{x}(\tau) \in \mathcal{X},\ \ \tilde{u}(\tau) \in \mathcal{U},\ \ \hat{\xi}_{ref}(\tau) \in \mathcal{Z}, && \text{(2d)}\\
& \mathcal{O}_i(\tilde{x}) \leq 0\,, \quad i = 1,\ 2,\ \ldots,\ p, && \text{(2e)}\\
\text{for} \quad & \tau \in [t_n, t_n + T].
\end{aligned}$$

The stage L and terminal cost E functions in (2) can be selected as the terms (3a,b), which allows the use of the Gauss–Newton method for quickly finding a good approximation of the Hessian matrix within the optimization problem:

$$L\big(\tilde{x}(\tau), \hat{\xi}_{ref}(\tau)\big) = \|\tilde{x}(\tau) - x_{ss}\|^2_{W_x} + \|\tilde{\xi}(\tau) - \hat{\xi}_{ref}(\tau)\|^2_{W_\xi} \tag{3a}$$

$$E\big(\tilde{x}(t_n + T)\big) = \|\tilde{x}(t_n + T) - x_{ss}\|^2_{W_T}, \tag{3b}$$

where the constraint sets for the state, control and output trajectory are $\mathcal{X} \subseteq X$, $\mathcal{U} \subseteq U$, and $\mathcal{Z} \subseteq X$, respectively. A total of p static and dynamic obstacles within the environment are represented by the inequality constraint $\mathcal{O}_i(\tilde{x}) \leq 0$ in (2e). The deviation between the predicted system state trajectory $\tilde{x}$ and the stabilization setpoint x_{ss} and between the predicted output trajectory $\tilde{\xi}$ and the estimated reference trajectory $\hat{\xi}_{ref}$ are penalized within the stage cost function L (3a) by the weighting matrices $W_x \in \mathbb{R}^{n_x \times n_x}$ and $W_\xi \in \mathbb{R}^{n_\xi \times n_\xi}$, respectively. The terminal cost function E (3b), which represents the cost of steady-state error, penalizes the deviation between the end value of the system state prediction $\tilde{x}(t_n + T)$ and the terminal setpoint x_{ss} by the weighting matrix W_T.

In words, the optimization process (2) runs as follows:

1. Measure or estimate the current state $x(t_n)$ of the actual closed-loop system;
2. Predict $\tilde{x}$ and $\hat{\xi}_{ref}$ by minimizing the cost function $J\big(\tilde{x}, \hat{\xi}_{ref}\big)$ over the prediction horizon $[t_n, t_n + T]$ subject to the system dynamics (2b), control law (2c), and assigned constraints (2e);
3. Use the estimated reference trajectory $\hat{\xi}_{ref}$ (or the predicted output trajectory $\tilde{\xi}$, as both converge to each other) as the reference trajectory of the actual closed-loop system;
4. Repeat Steps 1–3 until the drone vehicle approaches the terminal setpoint x_{ss} or the optimization problem produces an infeasible solution.

A comprehensive study about NMPH, including discrete-time representation, use of constraints, and software implementation of the optimization problem can be found in our recent work [27].

2.2. Drone Dynamics

The dynamics of a multi-rotor drone vehicle can be modeled using the Newton–Euler equations [35], governing six degrees of freedom rigid-body motion, augmented with force and torque generation models of the individual rotors. The model can either assume static hover conditions for simplicity, or include linear and angular velocity drag forces and rotor gyroscopic effects to yield a more complicated but higher-fidelity model.

To model rigid-body dynamics, two reference frames are used: a stationary ground-fixed navigation frame $\mathcal{N}$, and a moving body-fixed frame $\mathcal{B}$. The origin of the latter is placed at the drone's center of gravity, as shown in Figure 2. The frame bases follow the East, North, and Up (ENU) convention, with orthonormal basis vectors $\{n_1, n_2, n_3\}$ and $\{b_1, b_2, b_3\}$ for the navigation and body frames, respectively.

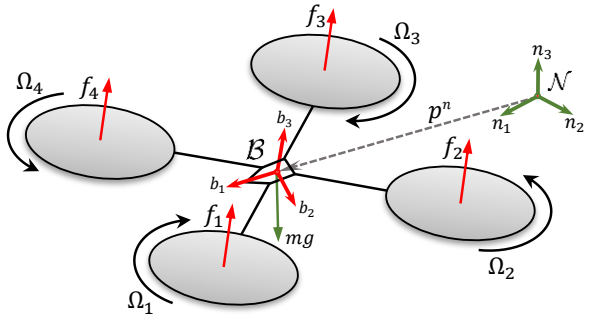


Figure 2. Reference frames used for our quadrotor vehicle.

Rigid-body pose in space can be described as a member of the Special Euclidean group $\mathrm{SE}(3) = \mathrm{SO}(3) \times \mathbb{R}^3$, the product space of the orientation and position (R_{nb}, p^n) where $R_{nb} \in \mathrm{SO}(3)$ is the rotation matrix of the body frame with respect to the navigation frame, $p^n = [x \quad y \quad z]^T \in \mathbb{R}^3$ is the position vector of the vehicle's body frame with

respect to the navigation frame, and SO(3) is the Special Orthogonal group defined as $\mathrm{SO}(3) = \{R \in \mathbb{R}^{3\times 3} \mid RR^T = R^TR = I,\ \det(R) = +1\}$. The roll-pitch-yaw Euler angles $\eta = [\phi \quad \theta \quad \psi]^T$ are employed to parametrize the rotation matrix as:

$$R_{nb} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}, \tag{4}$$

where $s_{(\cdot)} = \sin(\cdot)$ and $c_{(\cdot)} = \cos(\cdot)$. Similarly, $t_{(\cdot)} = \tan(\cdot)$ which will be seen later.

Remark 1. *The Euler angle parameterization exhibits singularities at $\theta = \pi/2 + k\pi$, $k \in \mathbb{Z}$. One solution is to maintain $-\pi/2 < \theta < \pi/2$ by adding constraints within the NMPH optimization problem under (2d).*

Conversion between translational and rotational velocity vectors can be performed using the transformations [36]:

$$\dot{p}^n = v^n = R_{nb} v^b \tag{5a}$$

$$\omega^b = W\dot{\eta}, \tag{5b}$$

where $v^n, v^b \in \mathbb{R}^3$ are the translational velocity vectors in frame $\mathcal{N}$ and $\mathcal{B}$ coordinates, respectively, $\dot{\eta} = [\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}]^T$ is the vector of Euler angle rates, and $\omega^b \in \mathbb{R}^3$ is the angular velocity vector in frame $\mathcal{B}$ coordinates. The rotational velocity transformation matrix W is given by:

$$W = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & s_\phi c_\theta \\ 0 & -s_\phi & c_\phi c_\theta. \end{bmatrix} \tag{6}$$

The time derivative of the rotation matrix is $\dot{R}_{nb} = R_{nb}\mathcal{S}(\omega^b)$, where $\mathcal{S}(\cdot) : \mathbb{R}^3 \to \mathrm{so}(3)$ maps a vector to a skew-symmetric matrix such that $\mathcal{S}(a)b = a \times b$ for $a, b \in \mathbb{R}^3$. Taking the time derivatives of (5a,b),

$$\dot{v}^n = R_{nb}\dot{v}^b + R_{nb}\mathcal{S}(\omega^b)v^b = R_{nb}\left(\dot{v}^b + \omega^b \times v^b\right) \tag{7a}$$

$$\dot{\omega}^b = \dot{W}\dot{\eta} + W\ddot{\eta}, \tag{7b}$$

where $\dot{W} = \dot{\phi}(\partial W/\partial\phi) + \dot{\theta}(\partial W/\partial\theta)$.

The Newton–Euler equations for a multi-rotor drone read [36]

$$m\dot{v}^b + \omega^b \times mv^b = \bar{u} - K_t v^b - mR_{nb}^T \bar{g} \tag{8a}$$

$$J\dot{\omega}^b + \omega^b \times J\omega^b = \bar{\tau} - K_r\omega^b - \sum_{i=1}^{4}\left(\mathcal{S}(\omega^b)J_r\, q_i\right), \tag{8b}$$

where m is the drone's mass, $\bar{u} = [0 \quad 0 \quad u]^T$ is the thrust vector with $u = \sum_{i=1}^4 f_i$ the total thrust generated in the direction of b_3, $\bar{\tau} = [\tau^{b_1} \quad \tau^{b_2} \quad \tau^{b_3}]^T$ is the vector of torques about the b_1, b_2 and b_3 frame axes, $\bar{g} = [0 \quad 0 \quad g]^T$ is the gravitational acceleration vector where $g = 9.81\ \mathrm{m/s^2}$, $J = \mathrm{diag}(J_x, J_y, J_z)$ is the drone's mass moment of inertia matrix which is assumed to be diagonal, the scalar J_r is the rotor's inertia, $q_i = [0 \quad 0 \quad (-1)^{i+1}\omega_i]^T$ where ω_i is the angular speed of the ith propeller, and $K_t = \mathrm{diag}(k_{t1}, k_{t2}, k_{t3})$, $K_r = \mathrm{diag}(k_{r1}, k_{r2}, k_{r3})$ represent the translational and rotational drag coefficient matrices of the drone, respectively.

To express the drone's dynamics with respect to the navigation frame, Equation (8) are combined with (5) and (7) to yield:

$$mR_{nb}^{T}\dot{v}^{n} = \bar{u} - K_t R_{nb}^{T} v^{n} - mR_{nb}^{T}\bar{g} \tag{9a}$$

$$JW\ddot{\eta} + J\dot{W}\dot{\eta} + \mathcal{S}(W\dot{\eta})JW\dot{\eta} = \bar{\tau} - K_r W\dot{\eta} - \mathcal{S}(W\dot{\eta})\sum_{i=1}^{4} J_r\, q_i. \tag{9b}$$

This leads to:

$$\dot{v}^{n} = -\frac{1}{m}R_{nb}K_t R_{nb}^{T} v^{n} - \bar{g} + \frac{1}{m}R_{nb}\bar{u} \tag{10a}$$

$$\ddot{\eta} = -(JW)^{-1}\big(J\dot{W}\dot{\eta} + K_r W\dot{\eta} + \mathcal{S}(W\dot{\eta})\big(JW\dot{\eta} + \sum_{i=1}^{4} J_r\, q_i\big) - \bar{\tau}\big). \tag{10b}$$

Remark 2. *Each multi-rotor configuration (quadrotor, hexarotor, octorotor, and so on) has a different expression for the net body-frame thrust and torque vectors $\bar{u}$ and $\bar{\tau}$. These expressions are algebraic and can be readily calculated. The dynamics presented in (10) thus model any multi-rotor drone as long as the correct $\bar{u}$ and $\bar{\tau}$ expressions are used.*

The development of the proposed NMPH with a backstepping control design will be based on the dynamics model presented in (10). We will also present a design based on a simplified version of (10) to illustrate the ease of adapting the proposed approach to different model representations. This is in contrast to the formulation of NMPH with feedback linearization presented in our recent work [27] where this adaptation requires a fundamental re-derivation of the expressions involved.

In the simplified version of (10), body and propeller gyroscopic effects are dropped from the model, and the translational and rotational drags are neglected as well. The simplified model can then be written as:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \frac{J_y - J_z}{J_x}\dot{\theta}\dot{\psi} + \frac{1}{J_x}\tau^{b_1} \\ \frac{J_z - J_x}{J_y}\dot{\phi}\dot{\psi} + \frac{1}{J_y}\tau^{b_2} \\ \frac{J_x - J_y}{J_z}\dot{\phi}\dot{\theta} + \frac{1}{J_z}\tau^{b_3} \\ (c_\phi s_\theta c_\psi + s_\phi s_\psi)\frac{u}{m} \\ (c_\phi s_\theta s_\psi - s_\phi c_\psi)\frac{u}{m} \\ -g + (c_\phi c_\theta)\frac{u}{m} \end{bmatrix}. \tag{11}$$

2.3. Backstepping Control Design

Because a drone's dynamics are nonlinear, solving the NMPC optimization problem is challenging because of its non-convexity. Introducing backstepping control into the optimization problem (within the framework of NMPH) will either remove or reduce the nonlinearity of the overall system, and consequently the non-convexity of the optimization problem. This will make it possible to find an optimal solution more quickly.

In this section, a backstepping control law is derived for the drone dynamics, which will be used within the NMPH framework to enhance the performance of the reference trajectory prediction. To demonstrate the versatility of this methodology, both the simplified (11) and high-fidelity (10) models of the drone dynamics will be considered.

Our backstepping control design consists of a coupling of inner and outer control loops [37]. The inner loop controls the rotational dynamics of the drone and tracks desired values provided by the outer loop, which controls the translational dynamics. In the literature, many studies of backstepping control applied to multi-rotor drones considered applying the design steps to each system output separately [36,38–40], but in our work the method is first applied to the rotational dynamics subsystem by itself, then to the translational dynamics subsystem. This approach will facilitate the integration of BSC within the NMPH framework as discussed later in Section 2.4.

First, recall the terms:

$$\eta = [\phi \quad \theta \quad \psi]^T, \quad \dot{\eta} = [\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}]^T, \quad \eta_d = [\phi_d \quad \theta_d \quad \psi_d]^T, \tag{12}$$

where $\eta_d \in \mathbb{R}^3$ are the desired Euler angles, to be provided by the outer loop design. Now, define the tracking error vector $\delta_1 \in \mathbb{R}^3$ as:

$$\delta_1 = \eta_d - \eta, \tag{13}$$

and choose a positive semi-definite Lyapunov function candidate $V_1 \geq 0 \in \mathbb{R}$, such that

$$V_1 = \frac{1}{2}\delta_1^T \delta_1. \tag{14}$$

The time derivative of (14) is:

$$\dot{V}_1 = \delta_1^T \dot{\delta}_1 = \delta_1^T (\dot{\eta}_d - \dot{\eta}). \tag{15}$$

Next, define the virtual tracking error rate $\delta_2 \in \mathbb{R}^3$ and the first virtual control $v_1 = [v_\phi \quad v_\theta \quad v_\psi]^T \in \mathbb{R}^3$ as:

$$\delta_2 = v_1 - \dot{\eta} \tag{16}$$

$$v_1 = \dot{\eta}_d + \Lambda_1 \delta_1, \tag{17}$$

where $\Lambda_1 = \text{diag}(\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}^{3\times 3}$ is a diagonal gain matrix that contains positive entries such that Λ_1 is positive definite or $\Lambda_1 > 0$. Using (16) and (17), the derivative of the Lyapunov function candidate (15) can be written as:

$$\dot{V}_1 = \delta_1^T (v_1 - \Lambda_1 \delta_1 - \dot{\eta}) = \delta_1^T (\delta_2 - \Lambda_1 \delta_1) = \delta_1^T \delta_2 - \delta_1^T \Lambda_1 \delta_1, \tag{18}$$

which by inspection may or may not be negative semi-definite. Therefore, a recursive step must be performed. Note that the time derivatives of (13) and (16) are:

$$\dot{\delta}_1 = \dot{\eta}_d - \dot{\eta} = v_1 - \Lambda_1 \delta_1 + \delta_2 - v_1 = -\Lambda_1 \delta_1 + \delta_2 \tag{19}$$

$$\dot{\delta}_2 = \dot{v}_1 - \ddot{\eta} = \ddot{\eta}_d + \Lambda_1 \dot{\delta}_1 - \ddot{\eta}. \tag{20}$$

Now define the new positive semi-definite Lyapunov function candidate $V_2 \geq 0 \in \mathbb{R}$ as:

$$V_2 = \frac{1}{2}\delta_1^T \delta_1 + \frac{1}{2}\delta_2^T \delta_2, \tag{21}$$

such that

$$\begin{aligned} \dot{V}_2 &= \delta_1^T \dot{\delta}_1 + \delta_2^T \dot{\delta}_2 \\ &= -\delta_1^T \Lambda_1 \delta_1 + \delta_1^T \delta_2 + \delta_2^T \left(\ddot{\eta}_d + \Lambda_1(-\Lambda_1 \delta_1 + \delta_2) - \ddot{\eta}\right) \\ &= -\delta_1^T \Lambda_1 \delta_1 + \delta_2^T \left(\delta_1 + \ddot{\eta}_d - \Lambda_1^2 \delta_1 + \Lambda_1 \delta_2 - \ddot{\eta}\right), \end{aligned} \tag{22}$$

where $\delta_1^T \delta_2 = \delta_2^T \delta_1$. To stabilize the tracking errors δ_1 and δ_2, the backstepping control formulation will introduce a second virtual control $v_2 \in \mathbb{R}^3$. We will define v_2 based on the system dynamics, and then recursively design it within the backstepping control structure.

As mentioned in Section 2.2, we will apply the backstepping technique to both the full and simplified system dynamics presented in (10) and (11), respectively. For the full model, the attitude dynamics in (10b) can be written as:

$$\ddot{\eta} = \bar{f}_1(x) + \bar{g}_1(x, \bar{\tau}), \tag{23}$$

where

$$\bar{f}_1(x) = -(JW)^{-1}\left(J\dot{W}\dot{\eta} + K_r W\dot{\eta} + \mathcal{S}(W\dot{\eta})\left(JW\dot{\eta} + \sum_{i=1}^{4} J_r q_i\right)\right) \tag{24}$$

$$\bar{g}_1(x,\bar{\tau}) = (JW)^{-1}\bar{\tau} = \begin{bmatrix} \frac{1}{J_x}\tau^{b_1} + \frac{1}{J_y}s_\phi t_\theta \tau^{b_2} + \frac{1}{J_z}c_\phi t_\theta \tau^{b_3} \\ \frac{1}{J_y}c_\phi \tau^{b_2} - \frac{1}{J_z}s_\phi \tau^{b_3} \\ \frac{1}{J_y}\frac{s_\phi}{c_\theta}\tau^{b_2} + \frac{1}{J_z}\frac{c_\phi}{c_\theta}\tau^{b_3} \end{bmatrix} := \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}. \tag{25}$$

The second virtual control is defined as $v_2 = \bar{g}_1(x,\bar{\tau}) = [\tau_\phi \quad \tau_\theta \quad \tau_\psi]^T$, where τ_ϕ, τ_θ and τ_ψ are the virtual inputs of the rotational subsystem.

We now design v_2 to make the time derivative of the Lyapunov candidate Function (22) negative semi-definite. Let

$$\begin{aligned} v_2 &= \dot{v}_1 + \delta_1 - \bar{f}_1(x) + \Lambda_2\delta_2 \\ &= \ddot{\eta}_d + \Lambda_1\dot{\delta}_1 + \delta_1 - \bar{f}_1(x) + \Lambda_2\delta_2 \\ &= \ddot{\eta}_d + \Lambda_1(-\Lambda_1\delta_1 + \delta_2) + \delta_1 - \bar{f}_1(x) + \Lambda_2\delta_2, \end{aligned} \tag{26}$$

where $\Lambda_2 = \mathrm{diag}(\lambda_4, \lambda_5, \lambda_6) \in \mathbb{R}^{3\times 3}$ is the second diagonal gain matrix with positive entries such that $\Lambda_2 > 0$. By substituting (23) and (26) into (22) we obtain:

$$\begin{aligned} \dot{V}_2 &= -\delta_1^T\Lambda_1\delta_1 + \delta_2^T\left(\delta_1 + \ddot{\eta}_d - \Lambda_1^2\delta_1 + \Lambda_1\delta_2 - \bar{f}_1(x) - v_2\right) \\ &= -\delta_1^T\Lambda_1\delta_1 - \delta_2^T\Lambda_2\delta_2 \leq 0. \end{aligned} \tag{27}$$

By (21) and (27), we can thus conclude the asymptotic stability of the error terms δ_1 and δ_2, and thus the rotational subsystem. Consequently, the physical control law for the rotational subsystem can be found by returning to (26),

$$v_2 = [\tau_\phi \quad \tau_\theta \quad \tau_\psi]^T = \ddot{\eta}_d + (I - \Lambda_1^2)\delta_1 + (\Lambda_1 + \Lambda_2)\delta_2 - \bar{f}_1(x), \tag{28}$$

then solving (25) to obtain the physical control inputs as:

$$\begin{bmatrix} \tau^{b_1} \\ \tau^{b_2} \\ \tau^{b_3} \end{bmatrix} = \begin{bmatrix} J_x\left(\tau_\phi - s_\theta\tau_\psi\right) \\ J_y\left(c_\phi\tau_\theta + s_\phi c_\theta\tau_\psi\right) \\ J_z\left(-s_\phi\tau_\theta + c_\phi c_\theta\tau_\psi\right) \end{bmatrix}. \tag{29}$$

We now perform the backstepping design for the translational dynamics of the drone. The actual and desired position vectors with respect to the navigation frame are written as:

$$\chi = p^n = [x \quad y \quad z]^T, \quad \dot{\chi} = v^n = [\dot{x} \quad \dot{y} \quad \dot{z}]^T, \quad \chi_d = p_d^n = [x_d \quad y_d \quad z_d]^T. \tag{30}$$

For the first step of the backstepping design, the position tracking error vector and its time derivative are defined as:

$$\delta_3 = \chi_d - \chi\,, \quad \dot{\delta}_3 = \dot{\chi}_d - \dot{\chi} \in \mathbb{R}^3. \tag{31}$$

Consider the Lyapunov candidate function $V_3 \in \mathbb{R}$ and its time derivative,

$$V_3 = \frac{1}{2}\delta_3^T\delta_3 \tag{32}$$

$$\dot{V}_3 = \delta_3^T\dot{\delta}_3 = \delta_3^T(\dot{\chi}_d - \dot{\chi}). \tag{33}$$

Define the virtual tracking error rate and the first virtual control for the translational subsystem as:

$$\delta_4 = v_3 - \dot{\chi} \tag{34}$$
$$v_3 = \dot{\chi}_d + \Lambda_3 \delta_3, \tag{35}$$

where $\Lambda_3 = \text{diag}(\lambda_7, \lambda_8, \lambda_9) \in \mathbb{R}^{3\times3}$ is a diagonal gain matrix with positive entries, and $v_3 = [v_x \quad v_y \quad v_z]^T \in \mathbb{R}^3$ is the virtual control vector. Substituting (34) and (35) into the Lyapunov candidate function rate (33) yields:

$$\begin{aligned} \dot{V}_3 &= \delta_3^T (v_3 - \Lambda_3 \delta_3 + \delta_4 - v_3) \\ &= -\delta_3^T \Lambda_3 \delta_3 + \delta_3^T \delta_4, \end{aligned} \tag{36}$$

which cannot be guaranteed to be negative semi-definite. Therefore, a new Lyapunov candidate function $V_4 \in \mathbb{R}$ is defined as:

$$V_4 = \frac{1}{2}\delta_3^T \delta_3 + \frac{1}{2}\delta_4^T \delta_4 \tag{37}$$
$$\begin{aligned} \dot{V}_4 &= \delta_3^T \dot{\delta}_3 + \delta_4^T \dot{\delta}_4 \\ &= -\delta_3^T \Lambda_3 \delta_3 + \delta_3^T \delta_4 + \delta_4^T (\ddot{\chi}_d + \Lambda_3(\dot{\chi}_d - \dot{\chi}) - \ddot{\chi}) \\ &= -\delta_3^T \Lambda_3 \delta_3 + \delta_4^T \delta_3 + \delta_4^T (\ddot{\chi}_d + \Lambda_3(v_3 - \Lambda_3 \delta_3 + \delta_4 - v_3) - \ddot{\chi}) \\ &= -\delta_3^T \Lambda_3 \delta_3 + \delta_4^T (\delta_3 + \ddot{\chi}_d - \Lambda_3^2 \delta_3 + \Lambda_3 \delta_4 - \ddot{\chi}). \end{aligned} \tag{38}$$

The translational dynamics of the full model (10a) can be written as:

$$\ddot{\chi} = \bar{f}_2(x) + \bar{g}_2(x, u), \tag{39}$$

where

$$\bar{f}_2(x) = -\frac{1}{m} R_{nb} K_t R_{nb}^T v^n - \bar{g} \tag{40}$$
$$\bar{g}_2(x, u) = \frac{1}{m} \begin{bmatrix} (c_\phi s_\theta c_\psi + s_\phi s_\psi) u \\ (c_\phi s_\theta s_\psi - s_\phi c_\psi) u \\ (c_\phi c_\theta) u \end{bmatrix} := \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}, \tag{41}$$

and where $u \in \mathbb{R}$ is the total thrust of the propellers, a physical input.

The next step of the backstepping design introduces the second virtual control for the translational system $v_4 = \bar{g}_2(x, u) = [u_x \quad u_y \quad u_z]^T$. Assign this control as:

$$\begin{aligned} v_4 &= \dot{v}_3 + \delta_3 - \bar{f}_2(x) + \Lambda_4 \delta_4 \\ &= \ddot{\chi}_d + \Lambda_3 (\dot{\chi}_d - \dot{\chi}) + \delta_3 - \bar{f}_2(x) + \Lambda_4 \delta_4 \\ &= \ddot{\chi}_d + \Lambda_3 (v_3 - \Lambda_3 \delta_3 + \delta_4 - v_3) + \delta_3 - \bar{f}_2(x) + \Lambda_4 \delta_4 \\ &= \ddot{\chi}_d + \delta_3 - \Lambda_3^2 \delta_3 + \Lambda_3 \delta_4 + \Lambda_4 \delta_4 - \bar{f}_2(x), \end{aligned} \tag{42}$$

where $\Lambda_4 = \text{diag}(\lambda_{10}, \lambda_{11}, \lambda_{12}) \in \mathbb{R}^{3\times3}$ contains positive entries. Substituting (39) and (42) into (38) yields:

$$\begin{aligned} \dot{V}_4 &= -\delta_3^T \Lambda_3 \delta_3 + \delta_4^T (v_4 + \bar{f}_2(x) - \Lambda_4 \delta_4 - \bar{f}_2(x) - v_4) \\ &= -\delta_3^T \Lambda_3 \delta_3 - \delta_4^T \Lambda_4 \delta_4, \end{aligned} \tag{43}$$

such that $\dot{V}_4 \leq 0$, meaning the error terms δ_3, δ_4 are asymptotically stable, and thus the translational subsystem dynamics.

For our cascaded control design, the desired roll and pitch angles for the inner loop system are extracted from (41) after computing (42). Assume ψ is a measured state of the system. Then, the desired roll and pitch angles ϕ_d, θ_d and thrust u are obtained by solving (41), which gives:

$$\begin{aligned} \theta_d &= \tan^{-1}\left(\frac{c_\psi u_x + s_\psi u_y}{u_z}\right) \\ \phi_d &= \tan^{-1}\left(\frac{s_\psi u_x - c_\psi u_y}{u_z}\cos\theta_d\right) \\ u &= \frac{m u_z}{\cos\phi_d \cos\theta_d}. \end{aligned} \tag{44}$$

Remark 3. *In addition to the Euler angles limitations mentioned in* Remark 1*, the outer loop control law provides solutions if and only if the total thrust is a non-zero positive value, $u > 0$. This condition must be included within the constraints of the optimization problem in (2d) to avoid solution infeasibility.*

We can also perform the backstepping control design using the simplified system model (11). The second time derivative of the Euler angles vector η is written as:

$$\ddot{\eta} = \bar{f}_3(x) + \bar{g}_3(\bar{\tau}), \tag{45}$$

where

$$\bar{f}_3(x) = \begin{bmatrix} \frac{J_y - J_z}{J_x}\dot{\theta}\dot{\psi} \\ \frac{J_z - J_x}{J_y}\dot{\phi}\dot{\psi} \\ \frac{J_x - J_y}{J_z}\dot{\phi}\dot{\theta} \end{bmatrix}, \quad \bar{g}_3(\bar{\tau}) = \begin{bmatrix} \frac{1}{J_x}\tau^{b_1} \\ \frac{1}{J_y}\tau^{b_2} \\ \frac{1}{J_z}\tau^{b_3} \end{bmatrix}. \tag{46}$$

The terms δ_1, δ_2 and virtual input v_1 are defined exactly as in the full model backstepping design. Analogously to (28), the second virtual input v_2 for the (simplified) rotational dynamics is now assigned as:

$$v_2 = \ddot{\eta}_d + (I - \Lambda_1^2)\delta_1 + (\Lambda_1 + \Lambda_2)\delta_2 - \bar{f}_3(x), \tag{47}$$

and since $v_2 = [\tau_\phi \quad \tau_\theta \quad \tau_\psi]^T = \bar{g}_3(\bar{\tau})$, the physical inputs are obtained from (46) as:

$$\bar{\tau} = \begin{bmatrix} \tau^{b_1} \\ \tau^{b_2} \\ \tau^{b_3} \end{bmatrix} = \begin{bmatrix} J_x\tau_\phi \\ J_y\tau_\theta \\ J_z\tau_\psi \end{bmatrix} = \mathrm{diag}(J_x, J_y, J_z)\, v_2. \tag{48}$$

For the translational dynamics, define the position vector χ, whose second time derivative is written as $\ddot{\chi} = \bar{f}_4(x) + \bar{g}_4(x, u)$, where:

$$\bar{f}_4(x) = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{49}$$

$$\bar{g}_4(x, u) = \begin{bmatrix} (c_\phi s_\theta c_\psi + s_\phi s_\psi)\frac{u}{m} \\ (c_\phi s_\theta s_\psi - s_\phi c_\psi)\frac{u}{m} \\ (c_\phi c_\theta)\frac{u}{m} \end{bmatrix}. \tag{50}$$

The definitions of δ_3, δ_4 and virtual input v_3 remain identical to the full model case. Assign the virtual control v_4 as in (42):

$$v_4 = \ddot{\chi}_d + (I - \Lambda_3^2)\delta_3 + (\Lambda_3 + \Lambda_4)\delta_4 - \bar{f}_4(x), \tag{51}$$

where $\bar{f}_4(x)$ is given in (49). Since $v_4 = [u_x \quad u_y \quad u_z]^T = \bar{g}_4(x, u)$, assuming the yaw angle ψ is known, we solve this expression for the desired roll and pitch angles ϕ_d, θ_d and the total thrust u using Equation (44).

2.4. Backstepping Control Law Integration within NMPH

A copy of the full system model in (10) is used within the NMPH optimization problem as $\dot{\tilde{x}} = [\tilde{p}^n \quad \tilde{v}^n \quad \tilde{\eta} \quad \dot{\tilde{\eta}}]^T$. Let $\tilde{\chi} = \tilde{p}^n$ and $\dot{\tilde{\chi}} = \tilde{v}^n$ below. The backstepping control design representing $\tilde{u}(\tau) = g\big(\tilde{x}(\tau), \hat{\tilde{\xi}}_{ref}(\tau)\big)$ within the NMPH (2c) is implemented in two stages. The first stage is the outer loop, which takes the desired position vector of the drone, and computes the thrust plus the desired roll and pitch angles. The second stage is the inner loop, which takes the computed roll and pitch angles plus a desired yaw angle, and computes the torque inputs. The details of the two-stage process (implemented as input constraints within the NMPH) are as follows:

1. For the first NMPH input constraint, define the virtual input for the translational dynamics (42) as:

$$\tilde{v}_4 = \ddot{\tilde{\chi}}_d + (I - \Lambda_3^2)\tilde{\delta}_3 + (\Lambda_3 + \Lambda_4)\dot{\tilde{\delta}}_4 - \bar{f}_2(\tilde{x}), \tag{52}$$

   where $\tilde{\delta}_3 = \tilde{\chi}_d - \tilde{\chi}$ from (31) and $\dot{\tilde{\delta}}_4 = \dot{\tilde{\chi}}_d + \Lambda_3\tilde{\delta}_3 - \tilde{\chi}$ from (34)
2. Associate the total thrust u and the desired roll and pitch angles ϕ_d, θ_d from (44) with $[u_x \quad u_y \quad u_z]^T = \tilde{v}_4$ and $\psi = \tilde{\eta}(3)$
3. For the second NMPH input constraint, define the virtual input for the rotational dynamics (28):

$$\tilde{v}_2 = \ddot{\tilde{\eta}}_d + (I - \Lambda_1^2)\tilde{\delta}_1 + (\Lambda_1 + \Lambda_2)\dot{\tilde{\delta}}_2 - \bar{f}_1(\tilde{x}), \tag{53}$$

   where $\tilde{\delta}_1 = \tilde{\eta}_d - \tilde{\eta}$ from (13) and $\dot{\tilde{\delta}}_2 = \dot{\tilde{\eta}}_d + \Lambda_1\tilde{\delta}_1 - \tilde{\eta}$ from (16).
4. Associate the input torques τ^{b_1}, τ^{b_2} and τ^{b_3} from (29) with $[\tau_\phi \quad \tau_\theta \quad \tau_\psi] = \tilde{v}_2$ and $\phi = \tilde{\eta}(1), \theta = \tilde{\eta}(2)$.
5. Let $[u \quad \tau^{b_1} \quad \tau^{b_2} \quad \tau^{b_3}]$ be $\tilde{u}(\tau)$ (2c) in the NMPH optimization problem, a function of the NMPH states $\tilde{x}$ and the estimated reference trajectories $\hat{\tilde{\xi}}_{ref}$, where

$$\hat{\tilde{\xi}}_{ref} = [\tilde{\chi}_d \quad \dot{\tilde{\chi}}_d \quad \ddot{\tilde{\chi}}_d \quad \tilde{\eta}_d(3) \quad \dot{\tilde{\eta}}_d(3) \quad \ddot{\tilde{\eta}}_d(3)]^T. \tag{54}$$

6. Solve the optimization problem (2), which leads to the prediction of the system states $\tilde{x}$ and the estimated reference trajectories $\hat{\tilde{\xi}}_{ref}$. The latter is used as the reference trajectory for the actual closed-loop system.

3. Evaluation of NMPH-BSC

For testing and validation, the NMPH-BSC approach was implemented and tested in simulated and hardware flight tests on quadcopter and hexacopter drone vehicles, respectively.

The algorithms are implemented within the Robot Operating System (ROS) [41], a Linux-based software environment that handles communications between the vehicle's onboard computer and its hardware subsystems. The ACADO Toolkit [42] is used to solve the optimization problem. For implementation, the overall NMPH problem (2) was coded in C++, then converted into highly efficient C code by ACADO to be able to run the calculations in real-time.

The set of continuous-time equations in (2) is a Nonlinear Programming (NLP) optimization problem, which can be discretized using the direct multiple-shooting method. NLP solves optimization problems, which include nonlinear functions and/or nonlinear constraints using Sequential Quadratic Programming (SQP) [43], and in our case the qpOASES solver is used to solve SQP numerically [44].

3.1. Simulation Environment

The proposed approach was first implemented in a simulation on a quadcopter drone using the AirSim simulator [45]. AirSim is an open-source package which provides photo-realistic rendered environments and a physics engine to enable performing lifelike simulations of drone vehicles. Moreover, we used the PX4 autopilot [46] running onboard

the drone for software-in-the-loop operation to make the simulated drone's characteristics more closely resemble the hardware unit.

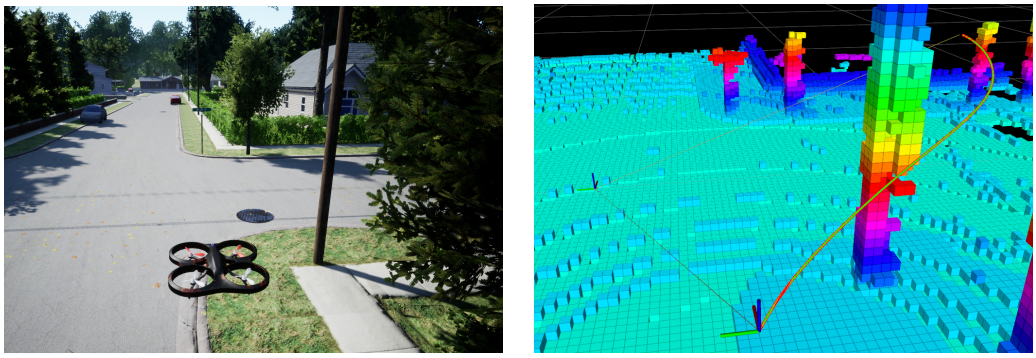
In our work, an incremental volumetric mapping technique named Voxblox [47] was used. Voxblox represents the environment volumetrically using a signed distance field and classification into unknown, free, or occupied spaces. The drone then uses this generated map to continuously build dynamic obstacle constraints that are used by the NMPH optimization problem to generate collision-free trajectories [34].

A desktop computer equipped with an Intel Core i7-10750H CPU and an Nvidia GeForce RTX 2080 Super GPU was used to run the optimization calculations and simulation environment in conjunction with ROS. The prediction horizon of the optimization problem was set to 8.0 s and discretized into 40 samples, which was found to be sufficient for motion planning purposes. The cost function weights were empirically tuned to provide good trajectory planning performance.

The drone's pose and environmental sensor readings were obtained from the AirSim simulator and communicated to our NMPH-based global motion planning system. The resulting output was used as a reference trajectory for the drone vehicle's flight controller. This design enables the drone to explore an unmapped environment, as will be discussed in Section 3.1.2.

3.1.1. Trajectory Planning

In this simulation, an optimal reference trajectory was planned and tracked within the AirSim simulator as shown in Figure 3a. The quadcopter vehicle started at $p^n = [-9, -3.5, 2]^T$ m, $\psi = 0^\circ$ and the NMPH-BSC algorithm was used to generate an optimal trajectory to the desired setpoint $p_d^n = [-5, -8, 5]^T$ m, $\psi_d = 90^\circ$ while avoiding an obstacle as shown in Figure 3b. The optimization problem within NMPH-BSC provided an estimate of the reference output trajectory $\hat{\xi}_{ref}$ and a prediction of the system state trajectory $\tilde{x}$, which included the predicted output trajectory $\tilde{\xi}$ as a subset.



(**a**) (**b**)

Figure 3. Trajectory Planning using the NMPH-BSC approach. (**a**) AirSim simulation environment. (**b**) Trajectory generation while avoiding static obstacle.

Figure 4 shows that the estimated reference trajectories of the vehicle's position and velocities $\hat{\xi}_{ref} = [\hat{\xi}_{x,ref}, \hat{\xi}_{y,ref}, \hat{\xi}_{z,ref}, \hat{\xi}_{\psi,ref}, \hat{\xi}_{\dot{x},ref}, \hat{\xi}_{\dot{y},ref}, \hat{\xi}_{\dot{z},ref}, \hat{\xi}_{\dot{\psi},ref}]$ perfectly match their corresponding predicted reference trajectories $\tilde{\xi} = [\tilde{x}_{x,ref}, \tilde{x}_{y,ref}, \tilde{x}_{z,ref}, \tilde{x}_{\psi,ref}, \tilde{x}_{\dot{x},ref}, \tilde{x}_{\dot{y},ref}, \tilde{x}_{\dot{z},ref}, \tilde{x}_{\dot{\psi},ref}]$. This confirms that using the stage cost function in (3a) minimizes the deviation between the estimated and predicted reference trajectories, and thus ensures their convergence towards each other. This validates the statement made in Section 2.1, that either the estimated or the predicted trajectory can be used as the reference trajectory for the closed-loop system.

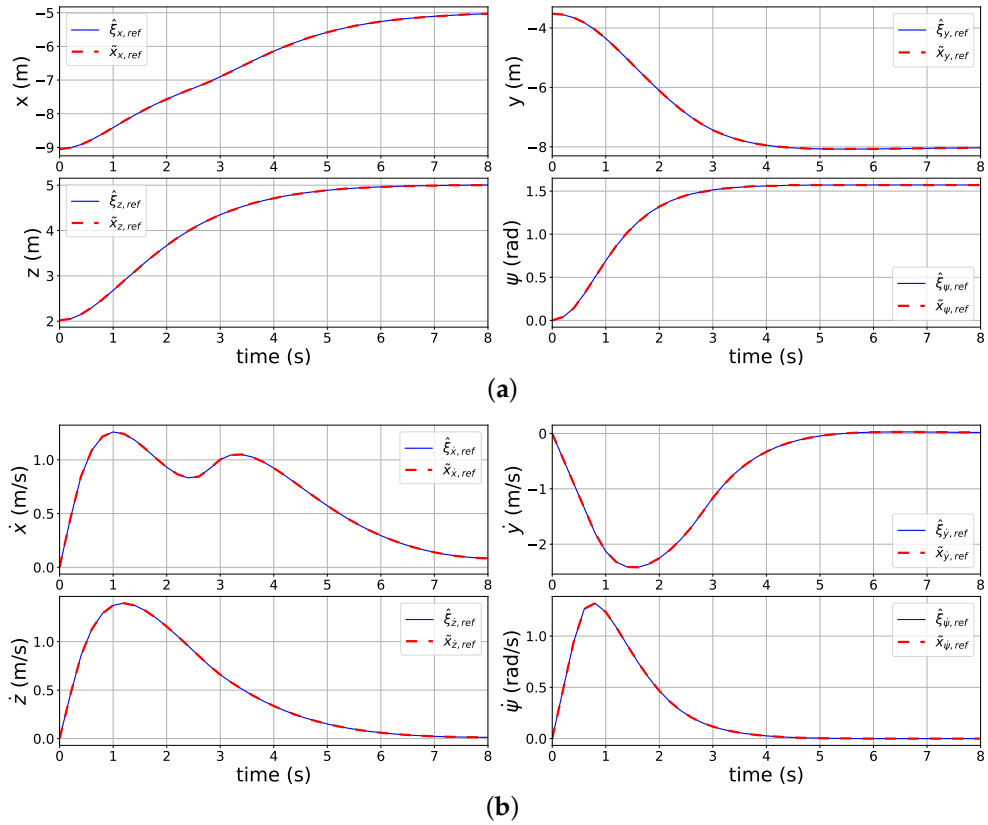


Figure 4. NMPH reference trajectory generation. The estimated and predicted reference position trajectories are depicted in (**a**), and the estimated reference velocity trajectories are shown in (**b**).

In the second simulation, tracking performance is assessed by having the vehicle move from the initial position $p^n = [0, 0, 2]^T$ m to the desired terminal point $p_d = [6, -4, 2]^T$ m using the optimized reference trajectory provided by the NMPH-BSC. The tracking performance is plotted in Figure 5, showing the vehicle satisfactorily tracks the time-varying reference trajectory generated by the NMPH-BSC algorithm. The small variation between the desired and actual outputs is because the former are obtained from numerical integration of the drone dynamics (10), while the latter are obtained from the physics engine of the simulation environment, which likely uses more complicated aerodynamic force and torque models than our design. This modeling mismatch can also be expected for real-world hardware testing, which will be covered in Section 3.2.

3.1.2. Exploration of Unknown Environment

In this simulation test, the drone explored an unknown environment by using a modular global motion planner, as described by the authors of [34]. This graph-based motion planner generated terminal setpoints within unexplored areas of an incrementally built-up volumetric map of the environment [47,48], and the NMPH-BSC algorithm was used to calculate optimal trajectories from the vehicle's current pose to these terminal setpoints. To achieve a smooth integration between the graph-based planner and the NMPH-BSC trajectory planning approach, computationally efficient algorithms for obstacle mapping and avoidance plus robust path guidance algorithms were used. Further details of this methodology can be found in [34].

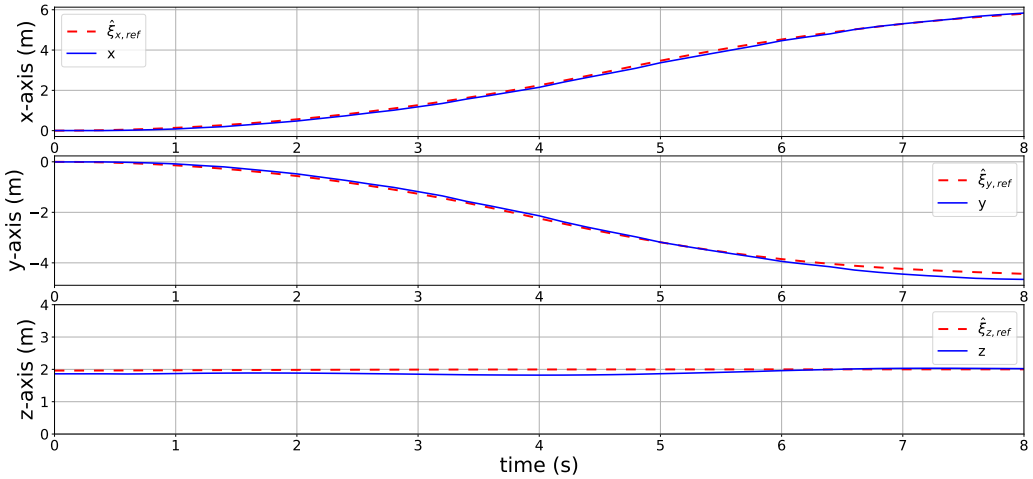


Figure 5. Vehicle's trajectory tracking performance between the start point (0, 0, 2) m and the terminal setpoint (5.8, −4.5, 2) m.

Figure 6 depicts a part of the exploration mission performed by the drone. The vehicle explored an unknown environment using the global motion planner in conjunction with the NMPH-BSC for local trajectory planning, which led to smoother flight trajectories than the stop-and-go patterns obtained from the motion planner alone. The data were collected over an exploration time of 1002 s. Table 1 offers a comparison between three approaches: graph-based planner only [48], graph-based planner plus NMPH-FBL [27], and graph-based planner plus NMPH-BSC (this paper). Based on this comparison, it can be seen that both NMPH algorithms have the effect of reducing the distance traveled, which reduces the mission time and consequently the energy consumption of the vehicle. Using the proposed NMPH-BSC leads to an improved performance over the NMPH-FBL. This is in addition to NMPH-BSC's advantages of avoiding numerical differentiation and its ability to extend to more complicated plant models as compared to NMPH-FBL.

Table 1. Comparison between graph-based and graph-based-plus-NMPH approaches to motion planning.

	Total Length of the Generated Paths	**Average Path Length (between Terminal Points)**	**Exploration Time**	**Continuous Path Generation**
Graph-based	1252 m	8.03 m	1322 s	No
Graph-based & NMPH-FBL	977 m (22.0% improvement)	6.25 m (22.2% improvement)	1032 s (21.9% improvement)	Yes
Graph-based & NMPH-BSC	949 m (24.2% improvement)	6.08 m (24.3% improvement)	1002 s (24.2% improvement)	Yes

3.2. Hardware Flight Experiments

For hardware experiments, a custom DJI FlameWheel F550 hexacopter was built and instrumented to explore unknown environments using our proposed NMPH-BSC approach in conjunction with a global motion planner. The drone was equipped with a Pixhawk 2.1 flight controller running the PX4 autopilot system [46], plus an Nvidia Jetson Xavier NX system-on-module running ROS Melodic Morenia [41] under Ubuntu 18.04. A Velodyne Puck LITE LiDAR sensor and an Intel RealSense T265 stereo camera were mounted on the drone to provide 360° point cloud and estimated pose data, respectively. The drone system used for testing is depicted in Figure 7.

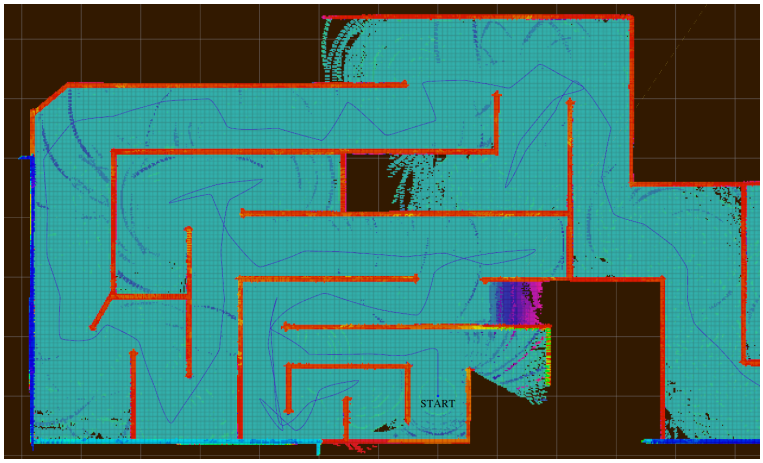


Figure 6. Exploration of unknown environment using global motion planner using NMPH-BSC for local trajectory planning.

Figure 7. DJI FlameWheel F550 hexacopter vehicle equipped with onboard sensors and computing systems.

Trajectory generation and tracking were evaluated by running the NMPH-BSC algorithm onboard the vehicle. Figure 8a shows the planned trajectory between two setpoints avoiding a sensed obstacle generated by the NMPH-BSC algorithm, while Figure 8b depicts the flight trajectory achieved by the drone using its flight controller to track the planned trajectory. The NMPH-BSC solver provides continuous updates of the estimated and predicted reference trajectories as the vehicle moves towards its endpoint. In this way, the system can handle uncertainties and disturbances such as dynamic environments and moving obstacles. The regeneration rate was set to 5 Hz, although this can be set as high as 100 Hz with the presented hardware.

Trajectory planning using NMPH-BSC in the presence of dynamic obstacles was evaluated experimentally as shown in Figure 9. In this Figure, it can be seen that the drone was able to regenerate trajectories to avoid a moving obstacle while flying through a constrained indoor environment. It is worth pointing out that the continuous trajectory regeneration process of NMPH-BSC provided smooth transitions between the generated trajectories, leading to smooth flight around the moving obstacle.

In a second flight test experiment, the navigation capabilities of the system were tested within a confined indoor environment as illustrated in Figure 10. During this test, the motion planner generated five terminal setpoints and the NMPH-BSC algorithm provided continuously regenerating local reference trajectories between the successive setpoints to ensure smooth flight.

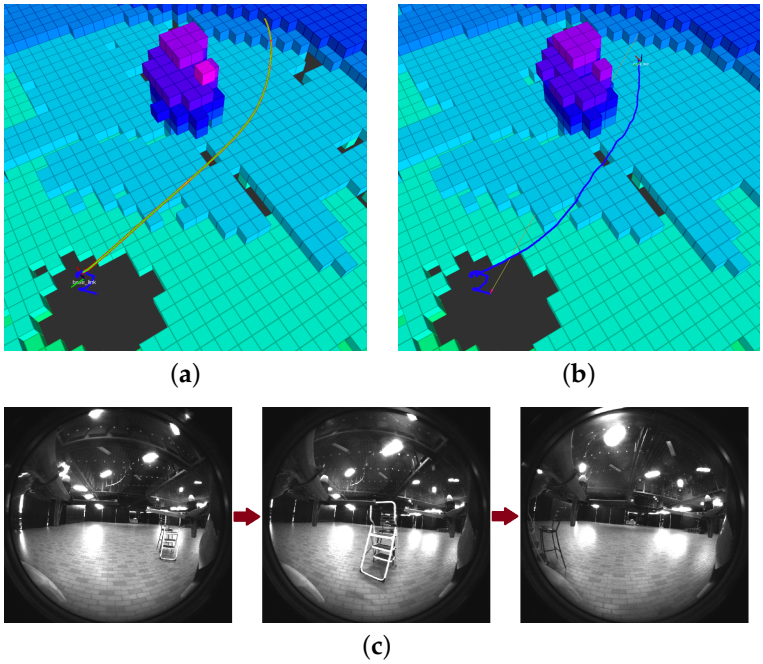


Figure 8. Trajectory planning with obstacle avoidance flight test using the NMPH-BSC algorithm. (**a**) Trajectory generation. (**b**) Trajectory tracking. (**c**) The mapped obstacle as seen from the left fisheye lens of the onboard camera.

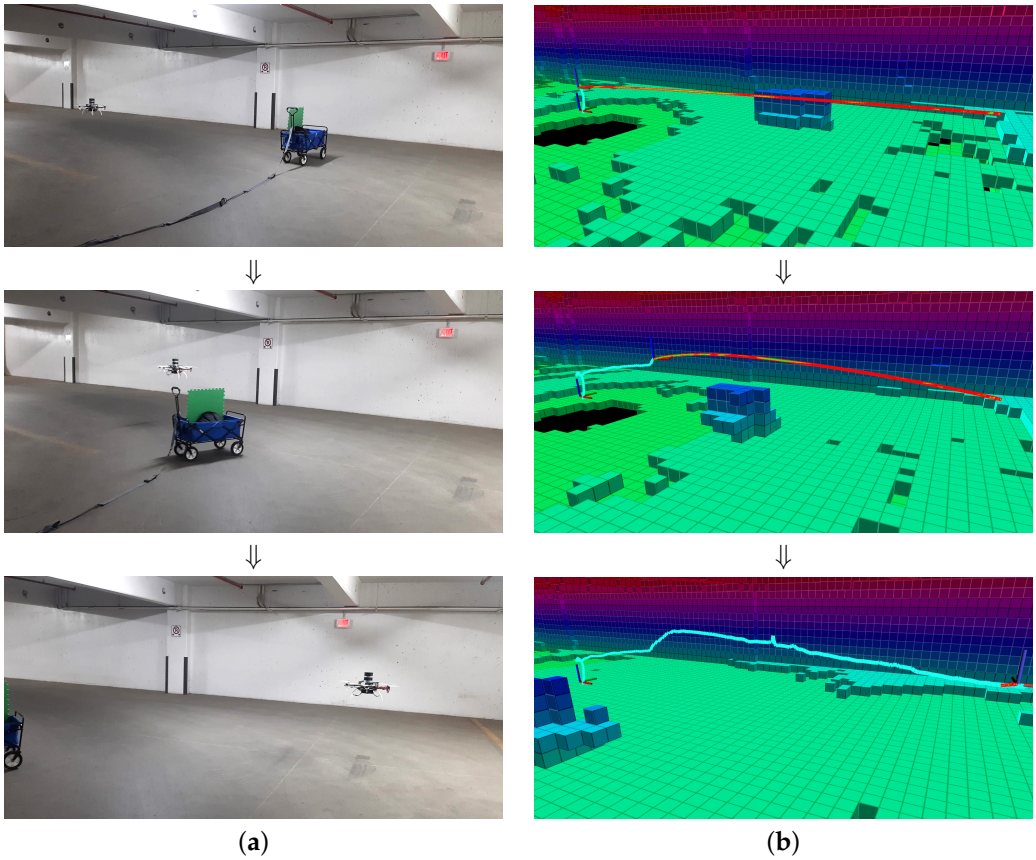


Figure 9. Hardware flight test for trajectory planning involving a dynamic obstacle using the NMPH-BSC algorithm. (**a**) Hardware drone avoiding the moving obstacle. (**b**) RViz visualization of the trajectory regeneration and flight path.

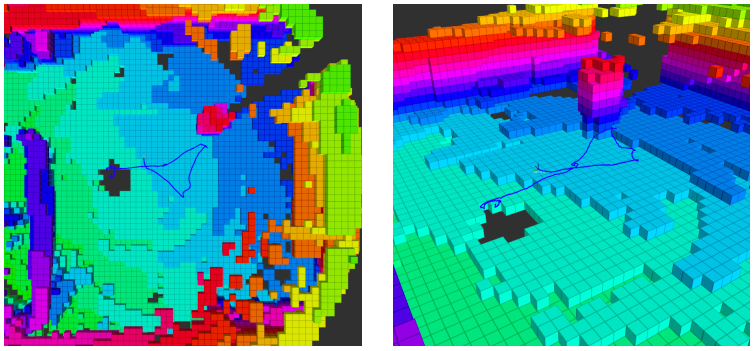


Figure 10. The hardware hexacopter vehicle navigating a confined indoor environment using a graph-based planner to generate terminal setpoints and NMPH-BSC to generate local trajectories between setpoints.

A final hardware flight test was performed outdoors in order to assess the trajectory planning performance in the presence of wind, in this case approximately 15 km/h. As before, the graph-based motion planner provided multiple terminal setpoints, while our NMPH-BSC planned smooth trajectories between them in real-time. The resulting flight trajectory can be seen in Figure 11. Despite the presence of a wind disturbance, the drone was able to smoothly navigate between generated setpoints as in the earlier tests.

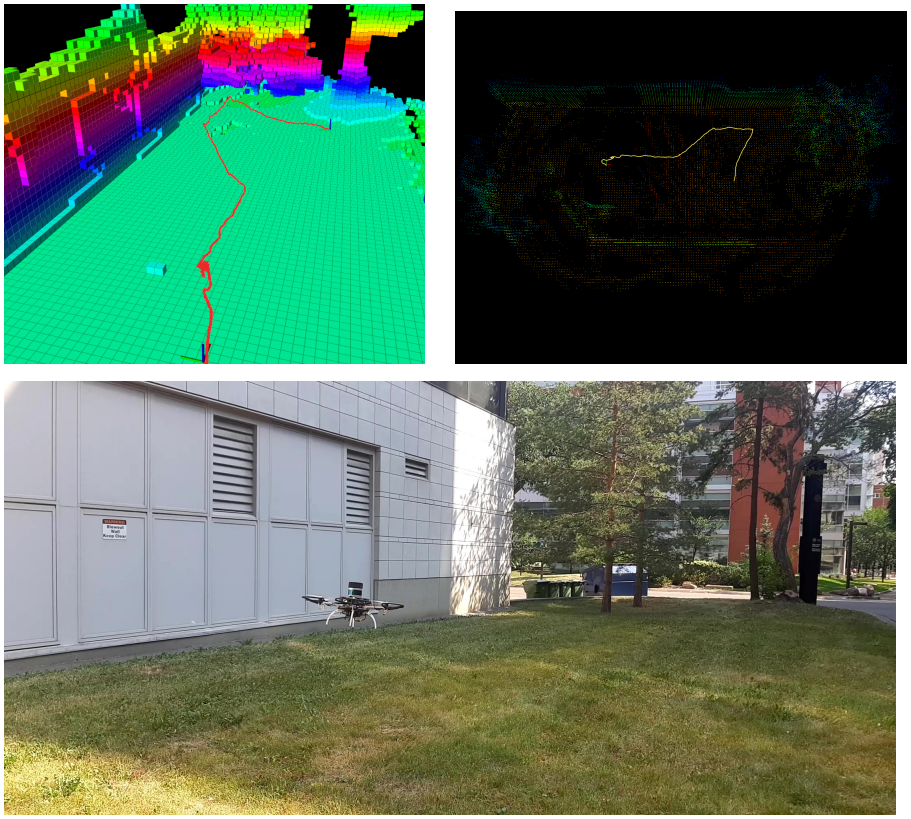


Figure 11. Flight test employing the NMPH-BSC algorithm in an outdoor environment with a 15 km/h wind speed.

4. Conclusions and Future Work

This paper proposed an optimization-based trajectory planning approach for drones operating in unknown environments. The proposed method embeds the Backstepping Control technique within our recently-proposed Nonlinear Model Predictive Horizon framework [27] for generating reference trajectories for nonlinear dynamical systems. This

integration reduces the non-convexity of the optimization problem and thus enables real-time computation of optimal trajectories which respect the nonlinear dynamics of the vehicle while avoiding static and dynamic obstacles.

The resulting NMPH-BSC design was tested in simulation and hardware flight experiments on quadcopter and hexacopter drone vehicles, respectively. The results showed an improvement in performance over a system previously proposed by the authors based on Feedback Linearization [34]. The new design was shown to offer additional implementation advantages over our earlier work including the ability to readily extend to more complicated plant models and avoid numerical differentiation.

Future work will include implementing an adaptation scheme allowing the online updating of the optimization problem weights within the NMPH, and testing the developed system in large-scale, unknown and GPS-denied environments such as subterranean mines.

Author Contributions: Conceptualization, Y.A.Y. and M.B.; methodology, Y.A.Y.; software, Y.A.Y.; validation, Y.A.Y.; formal analysis, Y.A.Y.; investigation, Y.A.Y.; resources, M.B.; data curation, Y.A.Y.; writing—original draft preparation, Y.A.Y.; writing—review and editing, M.B.; visualization, Y.A.Y.; supervision, M.B.; project administration, M.B.; funding acquisition, M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by NSERC Alliance-AI Advance Program grant number 202102595. The APC was funded by NSERC Alliance-AI Advance Program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

Variable	Meaning
x, u, ξ	System state, input and output vectors
X, U, Ξ	State, input and output spaces
$\mathbb{R}^k$	Vector with k real-valued entries
ξ_{ref}	Reference (output) trajectory
f, h, g	System dynamics, system output and feedback maps
x_{ss}	Stabilization setpoint
$\tilde{x}$, $\tilde{u}$, $\tilde{\xi}$	Predicted state, input and output trajectories
$\hat{\xi}_{ref}$	Estimated reference trajectory
t_n	Sampling time
Δ	Error tolerance
L, E	Stage and terminal cost functions
$\mathcal{X}$, $\mathcal{U}$, $\mathcal{Z}$	State, input and output constraint sets
T	Prediction horizon
W_x, $W_{\tilde{\xi}}$, W_T	State, output and terminal cost weighting matrices
$\mathcal{O}_i$	ith obstacle constraint
$\mathcal{N}$, $\mathcal{B}$	Navigation and body-fixed frame
(n_1, n_2, n_3), (b_1, b_2, b_3)	Navigation and body frame basis vectors
SE(3), SO(3)	Special Euclidean and Special Orthogonal groups
R_{nb}	Rotation matrix of $\mathcal{B}$ relative to $\mathcal{N}$
p^n	Position vector of vehicle's $\mathcal{B}$ frame origin relative to $\mathcal{N}$
η	Vector of three Euler angles
ϕ, θ, ψ	Roll, pitch and yaw Euler angles
$s_{(\cdot)}$, $c_{(\cdot)}$, $t_{(\cdot)}$	Sine, cosine and tangent of angle

$\mathbb{Z}$	Set of integers
v^n, v^b	Translational velocity vectors in frame $\mathcal{N}$ and $\mathcal{B}$ coordinates
ω^b	Angular velocity vector in frame $\mathcal{B}$ coordinates
W	Transformation matrix (6)
$\mathcal{S}(\cdot)$	Map from 3-element vector to 3×3 skew-symmetric matrix
so(3)	Lie algebra associated to SO(3)
m	Mass of drone
f_i, u	Thrust of ith propeller, total thrust
τ^{b_k}	Torque about b_k axis
g	Acceleration due to gravity
$\bar{u}$, $\bar{\tau}$, $\bar{g}$	Thrust, torque and gravity vectors in (8)
$J = \text{diag}(J_x, J_y, J_z)$	Mass moment of inertia matrix of drone
J_r	Rotational inertia of propeller
ω_i	Angular velocity of ith propeller
$K_t = \text{diag}(k_{t1}, k_{t2}, k_{t3})$	Translational drag coefficient matrix
$K_r = \text{diag}(k_{r1}, k_{r2}, k_{r3})$	Rotational drag coefficient matrix
η_d	Desired Euler angles
$\delta_1, \delta_2, \delta_3, \delta_4$	Tracking error vectors
$\bar{f}_k$, $\bar{g}_k$	Vectors (24), (39), (45), (49)
V_1, V_2, V_3, V_4	Lyapunov function candidates
v_1, v_2	Virtual control vectors
$\tau_\phi, \tau_\theta, \tau_\psi$	Virtual inputs of the rotational subsystem
Λ_1, Λ_2, Λ_3, Λ_4	Positive-definite diagonal gain matrices
χ, χ_d	Actual and desired position vectors of drone
Acronym	**Meaning**
GPS	Global Positioning System
LPA*	Lifelong Planning A*
ARA*	Anytime Reparing A*
PRM	Probabilistic Road-Map
RRT	Rapidly-Exploring Random Tree
RRG	Rapidly-Exploring Random Graph
ANN	Artificial Neural Network
GA	Genetic Algorithm
ACO	Ant Colony Optimization
PSO	Particle Swarm Optimization
SA	Simulated Annealing
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
STOMP	Stochastic Trajectory Optimization for Motion Planning
NMPH	Nonlinear Model Predictive Horizon
BSC	Backstepping Control
FBL	Feedback Linearization
NMPC	Nonlinear Model Predictive Control
ENU	East, North, Up
ROS	Robot Operating System
NLP	Nonlinear Programming
SQP	Sequential Quadratic Programming
LiDAR	Light Detection and Ranging

References

1. Bergman, K.; Ljungqvist, O.; Glad, T.; Axehill, D. An optimization-based receding horizon trajectory planning algorithm. *IFAC-PapersOnLine* **2020**, *53*, 15550–15557. [CrossRef]
2. Manoharan, A.; Sharma, R.; Sujit, P.B. Multi-AAV Cooperative Path Planning using Nonlinear Model Predictive Control with Localization Constraints. *arXiv* **2022**, arXiv:2201.09285.
3. Wu, D.M.; Li, Y.; Du, C.Q.; Ding, H.T.; Li, Y.; Yang, X.B.; Lu, X.Y. Fast velocity trajectory planning and control algorithm of intelligent 4WD electric vehicle for energy saving using time-based MPC. *IET Intell. Transp. Syst.* **2019**, *13*, 153–159. [CrossRef]
4. Gasparetto, A.; Boscariol, P.; Lanzutti, A.; Vidoni, R. Path planning and trajectory planning algorithms: A general overview. In *Motion and Operation Planning of Robotic Systems*; Carbone, G., Gomez-Bravo, F., Eds.; Springer: Cham, Switzerland, 2015; Volume 29, pp. 293–308.

5. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
6. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
7. Koenig, S.; Likhachev, M. Fast replanning for navigation in unknown terrain. *IEEE Trans. Robot.* **2005**, *21*, 354–363. [CrossRef]
8. Al-Mutib, K.; AlSulaiman, M.; Emaduddin, M.; Ramdane, H.; Mattar, E. D* lite based real-time multi-agent path planning in dynamic environments. In Proceedings of the 2011 Third International Conference on Computational Intelligence, Modelling & Simulation, Langkawi, Malaysia, 20–22 September 2011; pp. 170–174.
9. Likhachev, M.; Gordon, G.J.; Thrun, S. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*; Thrun, S., Saul, L.K., Schölkopf, B., Eds.; MIT Press: Cambridge, MA, USA, 2004; Volume 16, pp. 767–774.
10. Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robot. Res.* **2010**, *29*, 485–501. [CrossRef]
11. Siméon, T.; Laumond, J.P.; Nissoux, C. Visibility-based probabilistic roadmaps for motion planning. *Adv. Robot.* **2000**, *14*, 477–493. [CrossRef]
12. LaValle, S.M.; Kuffner, J.J. Rapidly-Exploring Random Trees: Progress and Prospects. In *Algorithmic and Computational Robotics: New Directions*; Donald, B.R., Lynch, K.M., Rus, D., Eds.; CRC Press: Boca Raton, FL, USA, 2001; pp. 293–308.
13. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
14. Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [CrossRef]
15. Iswanto, I.; Ma'arif, A.; Wahyunggoro, O.; Cahyadi, A.I. Artificial Potential Field Algorithm Implementation for Quadrotor Path Planning. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 575–585. [CrossRef]
16. Martin, P.; del Pobil, A.P. Application Of Artificial Neural Networks To The Robot Path Planning Problem. In *Applications of Artificial Intelligence in Engineering IX*; Adey, R., Rzevski, G., Russell, D., Eds.; WIT Press: Southampton, UK, 1994; pp. 73–80.
17. Zhao, M.; Ansari, N.; Hou, E.S.H. Mobile manipulator path planning by a genetic algorithm. *J. Field Robot.* **1994**, *11*, 143–153. [CrossRef]
18. Wang, H.J.; Xiong, W. Research on global path planning based on ant colony optimization for AUV. *J. Mar. Sci. Appl.* **2009**, *8*, 58–64. [CrossRef]
19. Qiaorong, Z.; Guochang, G. Path planning based on improved binary particle swarm optimization algorithm. In Proceedings of the 2008 IEEE Conference on Robotics, Automation and Mechatronics, Chengdu, China, 21–24 September 2008; pp. 462–466.
20. Martinez-Alfaro, H.; Flugrad, D.R. Collision-free path planning for mobile robots and/or AGVs using simulated annealing. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 2–5 October 1994; Volume 1, pp. 270–275.
21. Zhang, H.Y.; Lin, W.M.; Chen, A.X. Path planning for the mobile robot: A review. *Symmetry* **2018**, *10*, 450. [CrossRef]
22. Sands, T. Flattening the Curve of Flexible Space Robotics. *Appl. Sci.* **2022**, *12*, 2992. [CrossRef]
23. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: New York, NY, USA , 2006.
24. Zucker, M.; Ratliff, N.; Dragan, A.D.; Pivtoraiko, M.; Klingensmith, M.; Dellin, C.M.; Bagnell, J.A.; Srinivasa, S.S. CHOMP: Covariant Hamiltonian optimization for motion planning. *Int. J. Robot. Res.* **2013**, *32*, 1164–1193. [CrossRef]
25. Kalakrishnan, M.; Chitta, S.; Theodorou, E.; Pastor, P.; Schaal, S. STOMP: Stochastic trajectory optimization for motion planning. In Proceedings of the 2011 IEEE international conference on robotics and automation, Shanghai, China, 9–13 May 2011; pp. 4569–4574.
26. Sandberg, A.; Sands, T. Autonomous Trajectory Generation Algorithms for Spacecraft Slew Maneuvers. *Aerospace* **2022**, *9*, 135. [CrossRef]
27. Al Younes, Y.; Barczyk, M. Nonlinear Model Predictive Horizon for Optimal Trajectory Generation. *Robotics* **2021**, *10*, 90. [CrossRef]
28. Krstic, M.; Kokotovic, P.V.; Kanellakopoulos, I. *Nonlinear and Adaptive Control Design*; John Wiley & Sons, Inc.: New York, NY, USA, 1995.
29. Vaidyanathan, S.; Azar, A.T. *Backstepping Control of Nonlinear Dynamical Systems*; Academic Press: San Diego, CA, USA, 2020.
30. Zhou, J.; Wen, C. *Adaptive Backstepping Control of Uncertain Systems: Nonsmooth Nonlinearities, Interactions or Time-Variations*; Springer: Heidelberg, Germany, 2008.
31. Fadhel, F.S.; Noaman, S.F. The Generalized Backstepping Control Method for Stabilizing and Solving Systems of Multiple Delay Differential Equations. *Al-Nahrain J. Sci.* **2018**, 150–156. [CrossRef]
32. Ye, Z.; Mohamadian, H. Nonlinear backstepping control of multibody aerodynamic systems with equational modeling. In Proceedings of the 2009 IEEE International Conference on Control and Automation, Christchurch, New Zealand, 9–11 December 2009; pp. 1775–1779.
33. Marino, R.; Tomei, P. *Nonlinear Control Design: Geometric, Adaptive, and Robust*; Prentice Hall: Hoboken, NJ, USA , 1995.
34. Al Younes, Y.; Barczyk, M. Optimal Motion Planning in GPS-Denied Environments Using Nonlinear Model Predictive Horizon. *Sensors* **2021**, *21*, 5547. [CrossRef]
35. Murray, R.M.; Li, Z.; Sastry, S.S. *A Mathematical Introduction to Robotic Manipulation*; CRC Press: Boca Raton, FL, USA, 1994.
36. Madani, T.; Benallegue, A. Control of a quadrotor mini-helicopter via full state backstepping technique. In Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA, 13–15 December 2006; pp. 1515–1520.

37. Al Younes, Y.; Drak, A.; Noura, H.; Rabhi, A.; Hajjaji, A.E. Quadrotor position Control using cascaded adaptive integral backstepping controllers. In *Applied Mechanics and Materials*; Trans Tech Publications Ltd.: Durnten-Zurich, Switzerland , 2014; Volume 565, pp. 98–106.
38. Bouabdallah, S.; Siegwart, R. Full control of a quadrotor. In Proceedings of the 2007 IEEE/RSJ international conference on intelligent robots and systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 153–158.
39. Basri, M.A.M.; Husain, A.R.; Danapalasingam, K.A. Enhanced backstepping controller design with application to autonomous quadrotor unmanned aerial vehicle. *J. Intell. Robot. Syst.* **2015**, *79*, 295–321. [CrossRef]
40. Chen, F.; Jiang, R.; Zhang, K.; Jiang, B.; Tao, G. Robust backstepping sliding-mode control and observer-based fault estimation for a quadrotor UAV. *IEEE Trans. Ind. Electron.* **2016**, *63*, 5044–5056. [CrossRef]
41. Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software in Robotics, Kobe, Japan, 12–17 May 2009 .
42. Houska, B.; Ferreau, H.; Diehl, M. ACADO Toolkit—An Open Source Framework for Automatic Control and Dynamic Optimization. *Optim. Control Appl. Methods* **2011**, *32*, 298–312. [CrossRef]
43. Boggs, P.T.; Tolle, J.W. Sequential quadratic programming. *Acta Numer.* **1995**, *4*, 1–51. [CrossRef]
44. Ferreau, H.; Kirches, C.; Potschka, A.; Bock, H.; Diehl, M. qpOASES: A parametric active-set algorithm for quadratic programming. *Math. Program. Comput.* **2014**, *6*, 327–363. [CrossRef]
45. Shah, S.; Dey, D.; Lovett, C.; Kapoor, A. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Field and Service Robotics*; Hutter, M., Siegwart, R., Eds.; Springer: Cham, Switzerland, 2018; pp. 621–635.
46. Meier, L.; Honegger, D.; Pollefeys, M. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 6235–6240.
47. Oleynikova, H.; Taylor, Z.; Fehr, M.; Siegwart, R.; Nieto, J. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1366–1373.
48. Dang, T.; Mascarich, F.; Khattak, S.; Papachristos, C.; Alexis, K. Graph-based path planning for autonomous robotic exploration in subterranean environments. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 4–8 November 2019; pp. 3105–3112.