Trajectory tracking for quadrotors: An optimization-based planning followed by controlling approach

Geesara Kulathunga | Dmitry Devitt | Alexandr Klimchik

Center for Technologies in Robotics and Mechatronics Components, Innopolis University, Innopolis, Russia

Correspondence
Geesara Kulathunga, Center for Technologies in Robotics and Mechatronics Components, Innopolis University, Innopolis, Russia.
Email: ggeesara@gmail.com

Abstract
We present an optimization-based reference trajectory tracking method for quadrotor robots for slow-speed maneuvers. The proposed method uses planning followed by the controlling paradigm. The basic concept of the proposed method is an analogy with linear quadratic Gaussian in which nonlinear model predictive control (NMPC) is employed for predicting optimal control policy in each iteration. Multiple-shooting is suggested over direct-collocation for imposing constraints when modeling the NMPC. Incremental Euclidean distance transformation map is constructed for obtaining the closest free distances relative to the predicted trajectory; these distances are considered obstacle constraints. The reference trajectory is generated ensuring dynamic feasibility. The objective is to minimize the error between the quadrotor’s current pose and the desired reference trajectory pose in each iteration. Finally, we compared the proposed method with two other approaches and showed that the proposed method outperforms the said approaches in terms of reaching the goal without any collision. Additionally, we published a new data set that can be used for evaluating the performance of trajectory tracking algorithms.

KEYWORDS
Euclidean distance transformation map, linear quadratic Gaussian, motion planning, multiple shooting, nonlinear model predictive control, obstacle avoidance

1 | INTRODUCTION

In recent years, quadrotor-related research, particularly motion planning, expanding the range for venturing out computationally expensive techniques in various disciplines due to the increased availability of light-weight embedded devices. Yet path following and trajectory tracking are still open research fields due to their complexity. Path and trajectory are referred to as two different concepts: path is not represented by any temporal law (e.g., time, speed) and trajectory is defined by a time-parameterized path. Thus, the reference path is described with a time-free parameterization in the path following. Consequently, collision-free, geometrically feasible path generation is referred to as the path planning. Trajectory tracking involves time parameterization with space references. In addition to geometrical feasibility, dynamic and kinematic constraints are considered in trajectory tracking. In a nutshell, trajectory tracking involves finding optimal control policy to maneuver the quadrotor, ensuring dynamic and obstacle constraints.

Model Predictive Control (MPC) is one of the robust methods for determining an optimal control policy imposing constraints seamlessly. In each iteration, MPC solves an optimal control problem for a given prediction horizon. Consequently, the first portion of the optimal policy is applied to the system. Since the optimal policy calculation procedure has to compute in each step, MPC is computationally expensive. Hence, we proposed planning followed by a controlling strategy (see Figure 1) to overcome computational demands with short prediction horizon, in which a simplified motion model (4-DoF) is proposed at planning stage while keeping the real 6-DoF quadrotor model at controlling stage. However, the inability to use the real motion model and prediction horizon truncation reduce the trajectory tracker stability. The terminal constraints
and yaw, without considering pitch and roll changes in the planning stage, since this study focuses on slow speed maneuvers (predictably such motion model will lead to system instability). LQG-based planning followed by the controlling approach is proposed to overcome these drawbacks. Furthermore, nonlinear model predictive control (NMPC) is designed based on two different parameterization techniques: multiple shooting (MS) and direct collocation (DC). Hence, the comparative study was carried out for those two parameterization techniques.

Our contributions:
- Proposing a simplified motion model, which consists of only four states, that is, position \( \in \mathbb{R}^3 \) and yaw, without considering pitch and roll, to avoid the obstacles zone while minimizing the error between current pose and desired reference trajectory pose.
- A terminal constraints set (Morari & Lee, 1999) has to be introduced to preserve stability. However, adding a terminal constraints set increases computational demand at each step. Thus, we proposed a LQG-based approach (Yaghmaie et al., 2022) to reduce the disturbance and increase system stability.
- Proposing a simplified motion model, which consists of only four states, that is, position \( \in \mathbb{R}^3 \) and yaw, without considering pitch and roll changes in the planning stage, since this study focuses on slow speed maneuvers (predictably such motion model will lead to system instability). LQG-based planning followed by the controlling approach is proposed to overcome those drawbacks. Furthermore, nonlinear model predictive control (NMPC) is designed based on two different parameterization techniques: multiple shooting (MS) and direct collocation (DC). Hence, the comparative study was carried out for those two parameterization techniques.
- Constructing obstacles constraints directly from EDTM for identifying closest obstacles poses along the NMPC prediction horizon.
- Releasing a new data set, consisting of a hundred trajectories across cluttered environments; this data set can be used for evaluating the performance of reference trajectory tracking approaches.

2 | RELATED WORKS

2.1 | Map building and motion model selection

The proposed trajectory tracker can avoid obstacles while tracking the reference trajectory. Considering that a fast and accurate real-time map-building technique is required. Among the contributions toward real-time incremental map building, OctoMap (Hornung et al., 2013) is one of the popular choices among the researchers over the last decade. Moreover, Voxblox (Oleynikova et al., 2016), which is based on Truncated Signed Distance Fields (TSDFs), can be used for map building as well while keeping the map precision at high level. In this study, we use OctoMap for map building, but there is no restriction to use Voxblox instead.

In general, the quadrotor dynamics is described by 6-DOF. However, planning followed by controlling does not require to define an actual motion model for planing since the controller consists of a fully-fledged motion model of the quadrotor (van den Berg, 2016) presented 6-DOF exact motion model of a quadrotor. Other than the exact model, a 6-DOF motion model is proposed for governing a quadrotor in a distributed setup (Trawny et al., 2010). Later, it was reduced to 4-DOF motion model (Wanasinghe et al., 2015).

2.2 | Trajectory generation

Trajectory generation can be carried out in different ways, including path planning followed by smoothing and trajectory planning. Kulathunga et al. (2020) proposed a technique for path planning using Rapidly-Exploring-Random-Trees (RRT*) (Karaman & Frazzoli, 2011) and further smoothing the path by B-spline for a quadrotor. Yang and Sukkarieh (2010) proposed a continuous curvature-based path-smoothing technique is proposed for fixed-wing aerial vehicles. Zhou et al. (2019) proposed a spline-based optimization in which kinodynamic A* is used to find a collision-free path for a quadrotor. The researchers first generated topological paths and then utilized a guided gradient optimization-based technique for trajectory generation (Zhou et al., 2019). The Minimum-snap (Mellinger & Kumar, 2011) is one of the successful techniques proposed for trajectory generation in which differential flatness (Van Nieuwstadt & Murray, 1998) property was used to navigate the quadrotor. Richter et al. (2016) proposed an optimization-based trajectory planning approach in which a set of polynomials is used to define the trajectory.

2.3 | Trajectory tracking

Trajectory generation and trajectory tracking are interconnected. MPC, especially NMPC (He et al., 2022) is a promising technique in which trajectory generation and tracking are combined and solved as one problem. Nonlinear and linear, that is, linear quadratic regulator (LQR) (Rodriguez-Guevara et al., 2022), Yu et al. (2021a), and Yu et al. (2021b) approaches are the main approaches to controlling a system. With the recent development of deep reinforcement learning, learning-based approaches (Fan et al., 2020) could be added as the third approach. Recently, the scope of NMPC applicability has been increasing due to its capabilities, for example, dealing with multiple inputs and outputs, handling equality and inequality constraints, which could be either convex or non-convex. Islam et al. (2019) formulated the quadrotor dynamics based on quaternion orientation. Afterwards, they proposed an MPC-based trajectory tracker in which optimal control policy is generated by minimizing the quaternion error. In contrast to the this approach, Linear MPC trajectory tracker is proposed (Chipofya et al., 2015) in which a system is designed as an orthonormal function, for example, Laguerre.

MPC can be formed in different ways, for example, Stochastic MPC and Tube MPC. Lopez et al. (2019) formulated trajectory tracker as a
Tube MPC in which a terminal set was introduced to ensure the asymptotic convergence. The sparsity of the problem structure does matter for the fast convergence of MPC. Unlike the aforementioned approach, MS-based approach was proposed by Gros et al. (2012) to maneuver the quadrotor in a cluttered environment. They claim that MS helps to improve the sparsity of the optimal control problem (OCP). However, this technique was validated only in a simulated environment due to on-line computational demands. Li et al. (2020) proposed to obtain an optimal control policy using State-dependant Distance Metric (SDDM). They modeled the system dynamics as a linear time-invariant. Later, a reference governor (Garone & Nicotra, 2015) was introduced to maintain safety and stability since linear motion model was considered.

Ji et al. (2020) proposed intermediate controller, that is, Corridor-based Model Predictive Contouring Control (CMPCC) in between local planner and the main controller. CMPCC was based on SDDM in which flight corridors were set as hard constraints. Thus, CMPCC can capture local changes in real time; this yields the way system can address unmeasured disturbances. Once the reference trajectory was provided, trajectory tracking error was defined by the distance between the current pose of a quadrotor and the moving reference position. Initially, a set of hyperplanes was constructed within the prediction horizon as a polyhedron. Since MPC does calculate control inputs, that is, velocities, a flight corridor was generated intersecting the orthogonal projection of the velocity with corresponding polyhedral. Further, those projections are considered as the set of safe constrains that are considered as hard constraints when solving the MPC.

3 | METHODOLOGY

The proposed trajectory tracker was designed based on planning followed by controlling. There are several ways to control the quadrotor, including position (control x, y, and z position), velocity (control velocity on x, y, z direction and yaw rate), and attitude control (control the orientation with respect to an inertial frame). We decided to use velocity control in which velocity and yaw rate are required to control the robot. Moreover, velocity control helps bypass the generated high jerk, which tends to excite mechanical resonances causing vibrations. Hence, accurate control policy, that is, velocity and yaw rate, must be generated at the planning state in each iteration of NMPC. Yet, reducing the computation time of NMPC is also necessary for undergoing smooth trajectory tracking. Hence, having a simplified motion model rather than a real motion model is more appropriate at the planning stage. Since this study focuses on reference trajectory tracking for slow speed maneuvers, it is assumed that aggressive maneuvers are not necessary for such a task at the controlling stage. In a 6-DOF motion model, NMPC becomes diverged for a long prediction horizon and computationally expensive in the planning stage. Thus, a 4-DOF motion model was considered instead of incorporating the actual quadrotor model whose dynamics is described by 6-DOF. Mehrez et al. (2017) proposed similar simplified motion model for distributed MPC setup. However, the simplified motion model leads to an unstable control policy generation. Generating accurate control policies and estimating the quadrotor pose are required for smooth trajectory tracking while reducing the instability at the planning stage. As a result, we designed an LQG-based approach (Figure 2). The following sections explain each component of the proposed trajectory tracker.

3.1 | Simplified motion model

The system states and control inputs are described by $x_k = [p_x^k, p_y^k, p_z^k, \alpha_z^k]^T \in \mathbb{R}^n$ and $u_k = [v_x^k, v_y^k, v_z^k, \omega_z^k]^T \in \mathbb{R}^n$, respectively. In this formula, $p_x^k$ and $v_x^k$ denote the quadrotor center position (m) and velocity (m/s) in each direction, that is, x, y, z, at time $t = k$ in the world coordinate frame; $\alpha_z^k$ and $\omega_z^k$ denote the yaw angle (rad) and yaw rate (rad/s) around z axis, respectively. The simplified motion model is expressed by $\dot{x}_k = f(x_k, u_k)$

$$
\dot{x}_k = \begin{pmatrix}
\dot{p}_x^k \\
\dot{p}_y^k \\
\dot{p}_z^k \\
\dot{\alpha}_z^k
\end{pmatrix} = \begin{pmatrix}
v_x^k \cos(\alpha_z^k) - v_y^k \sin(\alpha_z^k) \\
v_x^k \sin(\alpha_z^k) + v_y^k \cos(\alpha_z^k) \\
v_z^k \\
\omega_z^k
\end{pmatrix}.
$$

FIGURE 2 The block scheme of the proposed trajectory tracker. Trajectory tracker and controller work as two separate closed-loop systems, provided that smoothed control command ($\tilde{u}_k$) is sent to the controller. Quadrotor has its onboard low level controller, namely DJI A3 flight controller DJIA3 (2019), which acts as a velocity controller that accepts velocity and yaw rate as input to control the quadrotor. PD regulator is employed for minimizing the error between trajectory tracker and actual controls ($\tilde{u}_k^{\text{ctrl}}$) from DJI A3. $x_t^{\text{ref}}$ and $\tilde{u}_k^{\text{ref}}$ are estimated by using $p_t^{\text{ref}}$ and $\tilde{p}_t^{\text{ref}}$ at each time index as proposed in (3). $x_t^{\text{ref}}$ and $\tilde{u}_k^{\text{ctrl}}$ of the real system (DJI M100) is calculated by its attached sensors: fusion of GPS and IMU.
where \( f_\varepsilon : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n \) and \( n_k = n_o = 4 \). The proposed tracking problem is formulated as a discrete dynamical model. Forward Euler discretization, \( x_{k+1} = f_\varepsilon(x_k, u_k) \) is introduced for a given sampling period in seconds, \( \delta \in \mathbb{R} > 0 \), we set \( \delta = 0.1s \) in this experiment.

\[
x_{k+1} = \begin{bmatrix}
    p^o_x^k \\
p^o_y^k \\
p^o_z^k \\
\alpha^k
\end{bmatrix} + \delta \begin{bmatrix}
    v^e_x \cos(\alpha^k) - v^e_y \sin(\alpha^k) \\
v^e_x \sin(\alpha^k) + v^e_y \cos(\alpha^k) \\
v^e_y \\
\omega^e_z
\end{bmatrix},
\]

where \( f_\varepsilon : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n \).

### 3.2 Close-in obstacles identification

Close-in obstacle pose identification relative to the pose of the quadrotor is crucial for tracking a trajectory safely. In this study, OctoMap was used to build the EDTM of the environment incrementally. A 16-channel LiDAR was used for reasoning the environment. EDTM was used to estimate the obstacle-free distance and the corresponding closest pose (close-in obstacle pose) from a specified pose. NMPC predicts a sequence of steps ahead of the forecasted trajectory. Thus, close-in obstacle poses are calculated with respect to each position along the NMPC forecasted trajectory. The number of steps depends on the horizon length of NMPC.

Figure 3 shows an instance view of a close-in obstacle detection with respect to quadrotor pose. Each red dotted line depicts the distance between the quadrotor and an obstacle position. Initially, close-in obstacles poses are estimated relative to the initial position of the quadrotor. After that, the previous prediction horizon steps are used to estimate the close-in obstacle poses. This will result in generating a set of obstacle constraints to guarantee a collision-free trajectory after discarding the poses of same obstacles. Tracking problem synthesis subsection in this paper describes how obstacle constraints are incorporated into trajectory tracker.

### 3.3 Reference trajectory generation

Tracking a reference trajectory while avoiding obstacles is quite a challenging task due to several reasons. For example, the robot could be trapped in a local minimum or have difficulty finding the appropriate return position back to the reference trajectory after avoiding the obstacles. Nonetheless, MPC-based trajectory tracking handles these problems adequately. In this study, the third-order \((d = 3)\) uniform B-Spline was used to generate the reference trajectories. In general, \(d\)th order B-Spline can be defined for a given knot sequence, that is, \( p^{\text{ref}} = \{ p_0, p_1, ..., p_n \} \), and control points, that is, \( p^{\text{ref}} = \{ p_0, p_1, ..., p_n, p_{n+d+1} \} \). The knot vector \( \tau = \{ t_0, t_1, ..., t_n \} \) and \( n_k = n_o + d + 1 \). For a given time index \( k \), corresponding position \( c^{\text{ref}}(k) \) can be fully determined using the DeBoorCox formula as given in Algorithm 1, which was initially proposed by de Boor (1971):

\[
c^{\text{ref}}(k) = \text{DeBoorCox}(k, p^{\text{ref}}), c^{\text{ref}}(k) \in \mathbb{R}^3.
\]

Here, \( c^{\text{ref}}(k) \) does not exactly equal to \( p^{\text{ref}} \); \( c^{\text{ref}}(k) \) is continuous and it may or may not pass through control points, for example, \( p^{\text{ref}} \) due to B-spline interpolation. We estimate reference velocity by taking the first derivative of \( p^{\text{ref}} \) and evaluating B-spline derivative using \( c^{\text{ref}}(k) = \text{DeBoorCox}(k, \dot{p}^{\text{ref}}) \) since the derivative of B-spline is also a B-spline. More information about the way

---

**Algorithm 1** Reference position \( c^{\text{ref}}(k) \) or velocity \( c^{\text{ref}}(k) \) estimation for a given time \( k \). \( p \) can be either \( p^{\text{ref}} \) (position) or \( \dot{p}^{\text{ref}} \) (velocity), and \( k \) is the desired time index.

1: procedure \text{DeBoorCox}(k, p) 
2: \( t_k = \begin{cases} 
    p^{\text{knot}}[d], & \text{if } k < p^{\text{knot}}[d] \\
    p^{\text{knot}}[n_p], & \text{if } k > p^{\text{knot}}[n_p] \\
    k, & \text{otherwise}
\end{cases} 
3: d_k = d 
4: while true do 
5: \text{if } p^{\text{knot}}[d_k + 1] \geq t_k \text{ then} 
6: \quad \text{break} 
7: \quad d_k++ 
8: \quad p_e[d] 
9: for i \leftarrow 0 \text{ to } d \text{ do} 
10: \quad p_e[i] \leftarrow p[d_k - d + i] 
11: for r \leftarrow 1 \text{ to } d \text{ do} 
12: \quad for i \leftarrow d \text{ to } r \text{ do} 
13: \quad \quad \beta \leftarrow \frac{p^{\text{knot}}[i + 1 + d_k - d] - p^{\text{knot}}[i] - \tau}{d_k - d} 
14: \quad \quad p_e[i] \leftarrow (1 - \beta)p_e[i - 1] + \beta p_e[i] 
15: \quad return p_e[d]
The sections below explain the constraints imposed by obstacles, as described in the reference trajectory generation subsection. Calculations \( g_{\text{ref}}(x_k, u_k) \) can be calculated as follows:

\[
\delta = \sum_{t=0}^{N_c} \left[ x_{k+1}^{\text{ref}} - x_{k+1} \right]_t + \left[ u_{k+1}^{\text{ref}} - u_{k+1} \right]_t^2, \quad \text{s.t. } \delta \leq \delta_{\text{ref}}.
\]

where \( \delta_{\text{ref}} \) denotes the decision variables set to be minimized. \( Q \in \mathbb{R}^{n\times n} \) and \( R \in \mathbb{R}^{n\times n} \) are symmetric. Furthermore, \( Q \) and \( R \) should be positive definite and positive semi-definite, respectively. In the experiment, both \( Q \) and \( R \) were set as identity matrices for giving the same weights for each state; \( p^\text{lower} \) and \( p^\text{upper} \) depict the upper and lower bound of \( x_k \), respectively. These bounds are enforced based on the map constraints where the quadrotor is allowed to fly. Notations \( u^\text{lower} \) and \( u^\text{upper} \) define the minimum and maximum linear and angular velocities allowed for the quasirotor maneuver. Term \( g_1(w) \) depicts the constraints that system dynamics imposes as follows:

\[
g_1(w) = \begin{bmatrix}
\dot{x}_k - x_k \\
\vdots \\
\dot{x}_k - x_k \\
\end{bmatrix}, \quad f_d(x_k, u_k) - x_{k+1} \\
\vdots \\
f_d(x_{k+N_c}, u_{k+N_c}) - x_{k+N_c}.
\]

Term \( g_2(w) \) describes the constraints imposed by obstacles, which is reconstructed in each iteration due to changes in the number of obstacles relative to the quadrotor pose. Consequently, reconstructing \( g_2(w) \) is useful to integrate the dynamic environment changes into the trajectory tracker as described in the equation below:

\[
g_2(w) = \begin{bmatrix}
d_{\text{dis}}(x_j, x_k) \\
\vdots \\
d_{\text{dis}}(x_j, x_k) \\
\end{bmatrix}, \quad j = 1, ..., N_o, \tag{6}
\]

where \( x_k \) is the initial position and \( N_o \) is the number of obstacles, and \( d_{\text{dis}}(x_j, x_k) \) can be calculated as follows:

\[
d_{\text{dis}}(x_j, x_k) = \sqrt{(x_j - x_k)^2 + (y_j - y_k)^2 + (z_j - z_k)^2 + d^2},
\]

where \( d^2 \) is the safe zone distance between a quadrotor and close-in obstacles.

### 3.4.2 Formulation with direct-collocation

In MS, we traded nonlinearity with a sparsity structure to reduce the nonlinearity. DC adds more degrees of freedom. Hence, DC exploits the sparsity even more. On the contrary, computation power is increased dramatically, though DC gives higher accuracy than MS.
When formulating DC, the first step is to define the collocation points with respect to a chosen polynomial. In this experiment, we chose Lagrangian 3rd order (Nₖ) polynomial to select collocation points. However, other polynomials, such as B-spline or Bézier, can be utilized as well. We fixed the time interval (8) when defining gᵢ in MS. Nonetheless, DC has more freedom to determine how points are distributed between two consecutive time intervals.

To formulate DC, we kept the same discretization as in the MS, that is, uₖ+i, uₖ+i+1, for i ∈ {tₖ, tₖ+1}, i = 0, ..., Nₖ - 1. Let η be the coefficient for 3rd order (Nₖ) Legendre points, η = [0, 0.112, 0.500, 0.888]. Subsequently, each consecutive time interval (tₖ+i and tₖ+i+1) was divided into small sub-intervals as follows:

\[ t_{kj} := t_k + h_k \eta_j, \quad k = 0, ..., N_k - 1, \quad j = 0, 1, ..., N_j, \]  (7)

where \( h_k = t_{k+1} - t_k \) and corresponding state vector at \( t_{kj} \) is denoted by \( x_{kj} \). In each control interval, Langrangian polynomial is defined as

\[ L_j(\eta) = \prod_{r=0, r \neq j}^{N_k} \frac{\eta - \eta_r}{\eta_j - \eta_r}, \quad L_j(\eta) = \begin{cases} 1, & \text{if } j = r \\ 0, & \text{otherwise} \end{cases} \]  (8)

State trajectory approximation (\( \hat{x}_k \)) and intermediate derivatives, that is, \( \nabla \hat{x}_k \), can be derived by using Langrangian functions (8) as

\[ \hat{x}_k = \sum_{r=0}^{N_k} L_r(\eta) \cdot \hat{x}_{kr}, \quad \nabla \hat{x}_k = \frac{1}{h_k} \sum_{r=0}^{N_k} L_r(\eta) \cdot \nabla \hat{x}_{kr}, \]  (9)

where \( k = 0, ..., N_k - 1 \) and \( j = 0, ..., N_j \). As we formulated NMPC in the Multiple-shooting subsection, for DC, only \( g_1(\omega) \) is changed, that is, (9), and the rest will remain the same.

NMPC output is post-processed to improve the smoothness using a KF smoother before sending the estimated control policy of trajectory planner to the controller as shown in Figure 2. KF smoother is needed to reduce the noise of NMPC, which occurs when the number of obstacles constraints is increased. Suboptimal KF (Van Der Merwe & Wan, 2001) was utilized over optimal KF to have better numerical stability. When defining the relationship between the current state and the next state of KF, the same motion model is utilized (2). \( \hat{x}_{k+1} = A \hat{x}_k + B u_k = f_k(\hat{x}_k, u_k) \). In KF smoother, \( B = 0 \) and \( \hat{x}_{k+1} = A \hat{x}_k \), where \( A \) is an identity matrix in \( \mathbb{R}^{n_{x+\omega} \times n_{x+\omega}} \) and \( B \) is given by

\[
\begin{bmatrix}
\cos(\theta(k)) & -\sin(\theta(k)) & 0 & 0 \\
\sin(\theta(k)) & \cos(\theta(k)) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}.
\]  (10)

Expressions \( \hat{x}_k = <\hat{x}_x, \hat{x}_y, \hat{x}_z, \hat{\omega}_z> \) and \( \hat{u}_k = <\hat{\omega}_{\omega x}, \hat{\omega}_{\omega y}, \hat{\omega}_{\omega z}, \hat{\omega}_{\omega z}> \) denote the expected values after applying KF and KF smoother, respectively. As explained in the Multiple-shooting subsection, yaw rate, that is, \( \alpha_i^v \), was excluded from the tracking problem. Hence, yaw rate is estimated as follows:

\[ \hat{\omega}_z^v = \begin{cases} \text{atan2}(\hat{\psi}_y^v, \hat{\psi}_x^v), & \hat{\psi}_y^v \geq 0 \\ \text{atan2}(\hat{\psi}_y^v, \hat{\psi}_x^v) + 2\pi, & \text{otherwise} \end{cases} \]  (11)

Estimated control and pose, \( \hat{u}_k \) and \( \hat{x}_k \), respectively, and actual control and pose (see Figure 2), \( u_k^{act} = <\nu_{\omega x}^{act}, \nu_{\omega y}^{act}, \nu_{\omega z}^{act}> \) and

\[ \begin{align*}
\text{FIGURE 4} & \quad \text{The hardware setup of the quadrotor. TFMini Plus is for estimating the altitude, whereas Velodyne Lidar-16 is for reasoning the environment.} \\
\text{FIGURE 5} & \quad \text{Control inputs are estimated based on LQG (Linear Quadratic Gaussian). Kalman filter is used as the estimator in LQG, where smoothness can be by adjusting the uncertainty matrix. (a) The simulated experiment that used to test the proposed LQG approach. (b) The predicted control inputs: } \nu_{\omega x}^{hat} \text{, } \nu_{\omega y}^{hat} \text{, } \nu_{\omega z}^{hat} \text{, all } x, y, z \text{ are generated by using the proposed NMPC approach and estimated them using LQG, respectively.}
\end{align*} \]
\( x_k^{\text{act}} = \langle p_k^{\text{act}}, \alpha_k^{\text{act}} \rangle \), respectively, are the inputs for the PD regulator that is formulated as follows for a given time index \( k \):

\[
v_k = k_p (p_k - p_v^*) - k_v v_k^{\text{act}}, \quad \omega_k = k_\omega (\alpha_k \omega_k^{\text{act}})
\]

where constant terms \((k_p = 0.8, k_v = 0.27, \omega = 2.5, \text{ and } k_\omega = 0.5)\) were estimated empirically.

## 4 | EXPERIMENT PROCEDURE

We conducted several types of experiments in the simulated and real-world environments to perform qualitative and quantitative analyses of the proposed trajectory tracker. First, trajectory tracker accuracy was analyzed in an obstacle-free zone to show the reference trajectory tracking accuracy. Afterwards, we compared the proposed trajectory tracker and two other approaches. The second part of the experiment laid out the qualitative analysis of the proposed trajectory tracker. The proposed trajectory tracker was implemented in C++11 with -02 (code-level optimization). We used a computer with 2.70 GHz CPU and 8GB RAM for simulated
experiments, whereas Jetson AGX Xavier was utilized as the onboard computer on the quadrotor as shown in Figure 4.

As mentioned in Section 3, generating accurate control policies and estimating the quadrotor pose is required for smooth trajectory tracking while reducing the instability at the planning stage. As a result, we designed an LQG-based approach. We have conducted a simulated experiment to showcase how the LQG-based approach affects the generation of smoothing control policies. The NMPC solver occasionally does not generate optimal solutions mainly due to the reference trajectory lying within the obstacle regions. Hence, the estimated control policy is not accurate enough for maneuvering (Figure 5). Thus, the proposed LQG-based approach helps eliminate those outliers, that is, control policies that are not feasible.

Our first experiment is devoted to checking the trajectory tracking accuracy is required for an obstacle-free zone. This experiment was conducted to ensure that the proposed approach can track the provided reference trajectory appropriately. NMPC prediction horizon $N_e$ was also adjusted for real-world conditions. When the value of $N_e$ is increased, execution time accumulates more. When the value of $N_e$ is decreased, the feasibility of the NMPC solver reduces. Thus, $N_e$ was kept at 15 after experiencing these behaviors.

Since NMPC model discretization is set at $\delta = 0.05$ s, the duration of the predicted trajectory is 1.5 s. Figure 6 shows the reference trajectory employed for this experiment, and the complete test is available at Github. Figure 7b clearly shows that the quadrotor almost follows the reference trajectory adhering to the imposed constraints, that is, max velocity ($0.4 \text{ m/s}$) (Figure 7a).

The next experiment is devoted to comparing the proposed trajectory tracker with two other approaches: presented by Oleynikova et al. (2016) and by Usenko et al. (2017), which were targeted on the slow-speed maneuvers similar to the proposed approach. We used the same validation technique that Oleynikova et al. (2016) used for checking the behavior in different environments. We generated 10 Poisson forests (Karaman & Frazzoli, 2012) with $10 \times 10 \times 10$ of densities in between 0.2 and 0.8 trees/m$^2$. Since Usenko et al. (2017) also used the same validation test, we compared the proposed trajectory tracker with theirs (Table 1); an example scenario is shown in Figure 8. We kept all the other configuration parameters constant only but the maximum velocity ($v_{\text{max}}$) was

### TABLE 1  Comparison of trajectory planning approaches

| Algorithm                        | Success fraction (SF) | Mean norm path length (MNPL) | Mean compute time (MCT) (s) |
|----------------------------------|-----------------------|-------------------------------|-----------------------------|
| Oleynikova et al. (2016) S = 2 jerk | 0.48                  | 1.1079                        | 0.0310                      |
| Oleynikova et al. (2016) S = 3 vel | 0.47                  | 1.1067                        | 0.09793                     |
| Oleynikova et al. (2016) S = 3 jerk | 0.50                  | 1.0996                        | 0.0367                      |
| Oleynikova et al. (2016) S = 3 jerk + Restart | 0.63                  | 1.1398                        | 0.1724                      |
| Usenko et al. (2017) C = 2       | 0.47                  | 1.0668                        | 0.0008                      |
| Usenko et al. (2017) C = 3       | 0.47                  | 1.0860                        | 0.0011                      |
| Usenko et al. (2017) C = 5       | 0.51                  | 1.1502                        | 0.0021                      |
| Usenko et al. (2017) C = 9       | 0.57                  | 1.3008                        | 0.0072                      |
| Proposed $v_x \leq 0.5 \text{ m/s}$, $N_e = 15$, $\delta = 0.05$ s | 0.88                  | 1.097                         | 0.108                       |
| Proposed $v_x \leq 1.0 \text{ m/s}$, $N_e = 15$, $\delta = 0.05$ s | 0.90                  | 1.0009                        | 0.167                       |
| Proposed $v_x \leq 1.5 \text{ m/s}$, $N_e = 15$, $\delta = 0.05$ s | 0.90                  | 1.0060                        | 0.213                       |
| Proposed $v_x \leq 2.0 \text{ m/s}$, $N_e = 15$, $\delta = 0.05$ s | 0.91                  | 1.0265                        | 0.248                       |

Note: Bolded values represent the best performing configurations (SF, MNPL, and MCT) for each considered approach. Abbreviations: $C$, control points to be taken for the optimization; $S$, the order of trajectory segment size with respect to minimizing parameters, that is, jerk, vel; $N_e$, NMPC prediction horizon; $\delta = 0.05$ s, forward Euler discretization step size.

### FIGURE 8  The result of the proposed trajectory tracker using a Poisson forest Karaman and Frazzoli (2012) $10 \times 10 \times 10$ m, of 0.2 trees/m$^2$. The start and the target positions were generated randomly keeping at least 4 m apart.

1The source code and complete experiments - https://github.com/GPrathap/trajectory-tracker.git
When $V_{\text{max}}$ is higher, the coverage of prediction horizon is higher, that is, it may take a higher number of obstacles constraints, which directly affects the Mean Computation Time (MCT) of NMPC solver.

As seen in Table 1, the proposed trajectory tracker could solve the trajectory in almost all the given test cases, which implies having high success fraction ($SF = \frac{\text{success test cases}}{\text{total test cases}}$). The obstacle cost was incorporated as constraints rather than minimizing as part of the cost; that was the key to obtaining high SF. In the other two approaches, obstacle cost was incorporated as a minimizing term of the objective. The objective minimization does not imply that obstacle constraints are met every time. Thus, there is a high possibility to crash the quadrotor. That is why the other two approaches are having less SF despite low MCT. The computational demand of NMPC is considerably higher, which yields higher solving time (MCT). Higher MCT is due to problem formulation, that is, as a constraint optimization problem. Since our objective is low-speed maneuvers, approximately 0.1 s of MCT is sufficient. Mean Norm Path Length (MNPL) was calculated by a ratio to the total distance of the projected path to the straight line distance between start and goal pose.

Table 2 shows the data set (Karaman & Frazzoli, 2012) we used for comparison. This data set was not complicated enough for the proposed trajectory tracker. Additionally, this data set was not created targeting for trajectory tracking. Thus, we proposed a new data set that can be used for validating the performance of trajectory tracking algorithms in cluttered environments. The proposed data set consists of 100 different reference trajectories as described in reference trajectory generation subsection. Each trajectory is provided with a set of intermediate waypoints (see Figure 9). The number of waypoints is used to construct a reference trajectory that is changing between 4 and 16. Such trajectories varied. When $V_{\text{max}}$ is higher, the coverage of prediction horizon is higher, that is, it may take a higher number of obstacles constraints, which directly affects the Mean Computation Time (MCT) of NMPC solver.

**TABLE 2** Comparison of results on the proposed data set

| Technique | Success fraction | Mean norm path length | Mean compute time (s) |
|-----------|-----------------|-----------------------|----------------------|
| MS, $v_x \leq 0.5 \text{ m/s}$, $N_e = 15$, $\delta = 0.05 \text{ s}$ | 0.8258 | 0.9838 | 0.105 |
| MS, $v_x \leq 1.0 \text{ m/s}$, $N_e = 15$, $\delta = 0.05 \text{ s}$ | 0.8431 | 1.0045 | 0.148 |
| MS, $v_x \leq 1.5 \text{ m/s}$, $N_e = 15$, $\delta = 0.05 \text{ s}$ | 0.8435 | 1.0974 | 0.196 |
| MS, $v_x \leq 2.0 \text{ m/s}$, $N_e = 15$, $\delta = 0.05 \text{ s}$ | 0.8524 | 1.0071 | 0.256 |
| DC, $v_x \leq 0.5 \text{ m/s}$, $N_e = 15$, $\delta = 0.05 \text{ s}$ | 0.9205 | 1.528 | 0.368 |
| DC, $v_x \leq 1.0 \text{ m/s}$, $N_e = 15$, $\delta = 0.05 \text{ s}$ | 0.9406 | 1.603 | 0.431 |
| DC, $v_x \leq 1.5 \text{ m/s}$, $N_e = 15$, $\delta = 0.05 \text{ s}$ | 0.9505 | 1.6019 | 0.489 |
| DC, $v_x \leq 2.0 \text{ m/s}$, $N_e = 15$, $\delta = 0.05 \text{ s}$ | 0.9606 | 1.623 | 0.645 |

Note: Bolded values represent the best performing configurations (SF, MNPL, and MCT) for each considered approach. Abbreviations: DC, direct collocation; MS, multiple shooting; $N_e$, prediction horizon of the NMPC; $\delta = 0.05 \text{ s}$, forward Euler discretization step size.

**TABLE 3** Run-time breakdown of NMPC formulation (4)

| Avg time (ms) | nlp_f | 16.9 |
|---------------|-------|-------|
|               | nlp_g | 29.45 |
|               | nlp_grad_f | 18.9 |
|               | nlp_hess_l | 460.77 |
|               | nlp_jac_g | 429.66 |
|               | total | 977.58 |

Abbreviations: nlp_f, nlp_g, cost function and constraints function estimation at the optimal solution; nlp_grad_f, the gradient of objective; nlp_hess_l, Hessian of the Lagrangian; nlp_jac_g, Jacobian of the constraints; NMPC, nonlinear model predictive control.
are generated, ensuring these factors: geometric complexity, obstacles, duration, and altitude changes when generating trajectories. Data set was collected in Gazebo (Koenig & Howard, 2004) simulated environment by using PX4 flight controller (Meier et al., 2011). More information about the data set, refers to the github repo of the source code. Table 2 shows the evaluation of the proposed trajectory tracker against the proposed data set. In this evaluation, MNPL was defined as the ratio of the reference trajectory path length to the corresponding proposed trajectory. SF and MCT are the same as before.

The number of waypoints is used to construct a reference trajectory that is changing between 4 and 16. Such trajectories are generated, ensuring these factors: geometric complexity, obstacles, duration, and altitude changes when generating trajectories.

Ipopt (Biegler & Zavala, 2009) was used for solving the NMPC. Table 2 displays the comparative result of two parameterization techniques (DC and MS). Even DC has a higher SF, MCT of DC is almost two-wise than MS. Thus, for real-time applications, this behavior is not acceptable. With that, we can conclude that MS is the optimal choice. When it comes to computational power, MS is better than DC, whereas the accuracy of DC is somewhat higher than MS. So, we decided to use MS as the default technique though the proposed solution supports both methods; the one to be used can be configured. Run-time breakdown of the Ipopt solver for the proposed NMPC formulation (MS) is given in Table 3 for the average case.

5 | CONCLUSION

We proposed an optimization-based trajectory tracking approach for slow-speed maneuvers and conducted qualitative and quantitative performance analysis. The proposed trajectory tracker can track a given reference trajectory while avoiding obstacles for low-speed maneuvers in which changes in roll and pitch were not considered and the only change in yaw is considered. With such consideration, the simplified motion model-based was proposed, which helped to reduce the computational demands when calculating control policy. Moreover, the problem was formulated as a constraints optimization problem that ensures safety. However, computational demand is yet to be further reduced to guarantee a fast reaction time to handle the dynamic obstacles, which will improve the safety maneuvering in cluttered environments. Forthcoming works will concentrate on further reducing the computational demands.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in trajectory_trackerv1 at https://github.com/GPPrathap/trajectory_trackerv1.git.

ORCID

Geesara Kulathunga http://orcid.org/0000-0003-3378-9813
Alexandr Klimchik http://orcid.org/0000-0002-2244-1849

REFERENCES

Biegler, L.T. & Zavala, V.M. (2009) Large-scale nonlinear programming using IPOPT: an integrating framework for enterprise-wide dynamic optimization. Computers & Chemical Engineering, 33(3), 575–582.

Chipofya, M., Lee, D.J. & Chong, K.T. (2015) Trajectory tracking and stabilization of a quadrotor using model predictive control of Laguerre functions. In Abstract and Applied Analysis, Vol. 2015. Hindawi.

de Boor, C. (1971) Subroutine package for calculating with splines. Los Alamos Scient. Lab. Report LA-4728-MS.

DJI A3. Dji flight controller. (2019) Available at: https://www.dji.com/ru/a3

Fan, D.D., Agha-mohammadi, A.-a. & Theodorou, E.A. (2020) Deep learning tubes for tube MPC. arXiv preprint arXiv:2002.01587.

Garone, E. & Nicotra, M.M. (2015) Explicit reference governor for constrained nonlinear systems. IEEE Transactions on Automatic Control, 61(5), 1379–1384.

Gros, S., Quirynen, R. & Diehl, M. (2012) Aircraft control based on fast nonlinear MPC & multiple-shooting. In: 2012 IEEE 51st IEEE Conference on Decision and Control (CDC). IEEE, pp. 1142–1147.

He, D., Wang, Q., Han, P. & Du, H. (2022) Constrained nonlinear MPC for accelerated tracking piece-wise references and its applications to thermal systems. Control Theory and Technology, 20, 1–11. https://www.springer.com/journal/11768

Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C. & Burgard, W. (2013) Octomap: an efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots, 34(3), 189–206.

Islam, M., Okasha, M. & Sulaeman, E. (2019) A model predictive control (MPC) approach on unit quaternion orientation based quadrotor for trajectory tracking. International Journal of Control, Automation and Systems, 17(11), 2819–2832.

Ji, J., Zhou, X., Xu, C. & Gao, F. (2020) Cmpcc: corridor-based model predictive contouring control for aggressive drone flight. arXiv preprint arXiv:2007.03271.

Karaman, S. & Frazzoli, E. (2011) Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research, 30(7), 846–894.

Karaman, S. & Frazzoli, E. (2012) High-speed flight in an ergodic forest. In: 2012 IEEE International Conference on Robotics and Automation. IEEE, pp. 2899–2906.

Koenig, N. & Howard, A. (2004) Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (iROS)(IEEE Cat. No. 04CH37566). IEEE, Vol. 3, pp. 2149–2154.

Kulathunga, G., Fedorenko, R., Kopylov, S. & Klimehik, A. (2020) Real-time long range trajectory replanning for MAVs in the presence of dynamic obstacles. In: 2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), pp. 145–153.

Li, Z., Arslan, O. & Atanasov, N. (2020) Fast and safe path-following control using a state-dependent directional metric. arXiv preprint arXiv:2002.02038.

Lopez, B.T., Slotine, J.-J.E. & How, J.P. (2019) Dynamic tube MPC for nonlinear systems. In: 2019 American Control Conference (ACC). IEEE, pp. 1655–1662.

Mehrez, M.W., Mann, G.K. & Gosine, R.G. (2017) An optimization based approach for relative localization and relative tracking control in multi-robot systems. Journal of Intelligent & Robot Systems, 85(2), 385–408.

Meier, L., Tanskanen, P., Fraudorfer, F. & Pollefeys, M. (2011) Pixhawk: a system for autonomous flight using onboard computer vision. In: 2011 IEEE International Conference on Robotics and Automation. IEEE, pp. 2992–2997.

Mellinger, D. & Kumar, V. (2011) Minimum snap trajectory generation and control for quadrotors. In: 2011 IEEE International Conference on Robotics and Automation. IEEE, pp. 2520–2525.
