Trajectory Tracking For Quadrotors: An Optimization-Based Planning Followed By Controlling Approach

Geesara Kulathunga (g.mudiyanselage@innopolis.ru)
Innopolis University
Dmitry Devitt
Innopolis University
Alexandr Klimchik
Innopolis University

Research Article

Keywords: trajectory, method, quadrotor

Posted Date: October 18th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-963714/v1

Trajectory Tracking for Quadrotors: an Optimization-based Planning Followed by Controlling Approach

Geesara Kulathunga[1,*], Dmitry Devitt[1], and Alexandr Klimchik[1]

[1]Center for Technologies in Robotics and Mechatronics Components, Innopolis University, Russia
[*]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 to Linear Quadratic Gaussian (LQG) in which Nonlinear Model Predictive Control (NMPC) is employed for predicting optimal control policy in each iteration. Multiple-shooting (MS) is suggested over Direct-collocation (DC) for imposing constraints when modelling the NMPC. Incremental Euclidean Distance Transformation Map (EDTM) 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 evaluated the proposed method with two other approaches and showed that our proposal is better than those two in terms of reaching the goal without any collision. Additionally, we published a new dataset, which can be used for evaluating the performance of trajectory tracking algorithms.

Introduction

In recent years, quadrotor related research, particularly motion planning, widens the range for venturing out computationally expensive techniques in various disciplines due to the high growth rate 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 (i.e., 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. Subsequently, collision-free, geometrically feasible path generation is referred to as the path planning. Trajectory tracking involves time parameterization with space references. In addition to geometrically 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.

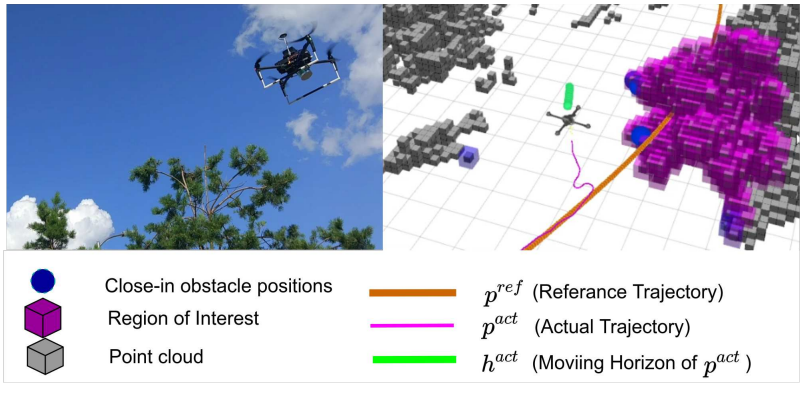


Figure 1. A field experiment result shows how the proposed trajectory tracker operates in real conditions. In this scenario, the reference trajectory (p_{ref}) passes through an obstacles zone; the trajectory tracker is trying to avoid the obstacles zone while keep minimizing the error between current pose and desired reference trajectory pose. The complete experiment is available at[1]

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 Fig.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 12-DoF quadtotor 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 set[2] has to be introduced to preserve stability. However, adding a terminal constraints set increases computational demand at each step. Thus, we proposed a LQG[3] based approach to reduce the disturbance and increase system stability.

Our Contributions:

1. Proposing a simplified motion model, which consists of only four states, i.e., position $\in \mathbf{R}^3$ and yaw, without considering pitch and roll changes in the planning stage since this research focuses on slow speed maneuvers. On the contrary, such motion model will lead to system instability. LQG-based planning followed by the controlling approach is proposed to overcome those drawbacks. Furthermore, NMPC is designed based on two different parametrization techniques: MS and DC. Hence, the comparative study was carried out for those two parametrization techniques

2. Constructing obstacles constraints directly from EDTM for identifying closest obstacles poses along the NMPC prediction horizon

3. Releasing a new dataset, consisting of a hundred trajectories across cluttered environments; this dataset can be used for evaluating the performance of reference trajectory tracking approaches

Related Works

Map Building and Motion Model Selection

The proposed trajectory tracker can avoid obstacles while tracking the reference trajectory. Considering that, the fast and accurate real-time map building technique is required. Within the literature, among the contributions towards real-time incremental map building, OctoMap[4] is one of the popular choices among the researchers over the last decade. Moreover, Voxblox[5] 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 work, we use OctoMap for map building.

In general, the quadrotor dynamics is described by 12-DOF. However, planning followed by controlling approaches does not require to define an actual motion model for planing since the controller consists of a fully-fledged motion model of the quadrotor. In[6], the motion model of the system is expressed as 12-DOF. Other than the exact model, a 6-DOF motion model is proposed for governing a quadrotor in a distributed setup[7]. Later, it was reduced to 4-DOF motion model[8].

Trajectory Generation

Trajectory generation can be carried out in different ways, including path planning followed by smoothing and trajectory planning. The authors of[9] proposed a technique for path planning using RRT*[10] and further smoothing the path by B-spline for a quadrotor. In[11], continuous curvature-based path-smoothing technique is proposed for fixed-wind aerial vehicles. Luqi et al.[12] proposed a spline-based optimization in which kinodynamic A* is used to find a feasible collision-free path for a quadrotor. In[13], researchers first generated topological paths and then utilized a guided gradient optimization-based technique for trajectory generation. The Minimum-snap[14] one of the successful techniques was proposed for trajectory generation in which differential flatness[15] property was used to navigate the quadrotor. Richter[16] proposed an optimization-based trajectory planning approach in which a set of polynomials is used to define the trajectory.

Trajectory Tracking

Trajectory generation and trajectory tracking are interconnected. MPC especially, NMPC[17] is a promising technique in which trajectory generation and tracking are combined and solved as one problem. Nonlinear and Linear, i.e., Linear Quadratic Regulator (LQR)[18] approaches are the main approaches to controlling a system. With the recent development of deep reinforcement learning, learning-based approaches[19], could be added as the third approach. Recently, NMPC has been employed in more and more applications due to its capabilities, e.g., dealing with multiple input and output, handling equality and inequality constraints, which could be either convex or non-convex. The authors of[20] formulated the quadrotor dynamic 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 by Mapopa et al.[21] in which a system is designed as an orthonormal function, e.g., Laguerre.

MPC can be formed in different ways, e.g., Stochastic MPC and Tube MPC. In[22], trajectory tracker was formulated 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 above mentioned approached, MS-based approach was proposed by Gros et al.[23] 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.[24] proposed to obtain an optimal control policy using State-dependant Distance Metric (SDDM). They modelled the system dynamics as a linear, time-invarian. Furthermore, a reference governor[25] was introduced to maintain safely and stability since linear motion model was considered.

In the work of[26], they proposed intermediate controller, i.e., Corridor-based Model Predictive Contouring Control (CMPCC) in between local planner and the main controller. CMPCC was based on SDDM in which flight corridors ware set as hard constraints. Thus, CMPCC can capture local changes in real-time; this yields how the 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, i.e., velocities, flight corridor was generated intersecting velocity's orthogonal projection with corresponding polyhedral. Further, those projections are considered as the safe constrains set, which are considered as hard constraints when solving the MPC.

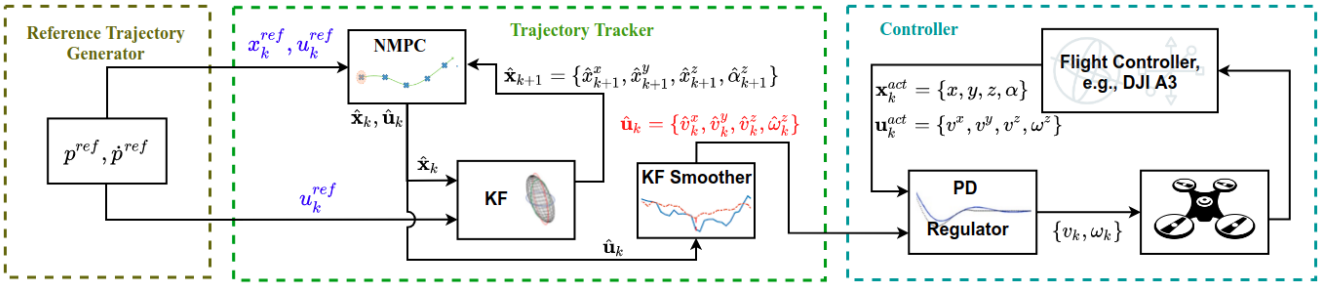


Figure 2. The block schema of the proposed trajectory tracker. Trajectory tracker and controller work as two separate closed-loop systems, provided that smoothed control command ($\hat{\mathbf{u}}_k$) is sent to the controller. Quadrotor has its onboard low level controller, namely DJI A3 flight controller[27], 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 (u_k^{act}) from DJI A3. x_k^{ref} and u_k^{ref} are estimated by using p^{ref} and $\dot{p}^{ref}$ at each time index as proposed in (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 generated high jerk, which tends to excite mechanical resonances, causing vibrations. Hence, accurate control policy, i.e., 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 work focuses on reference trajectory tracking for slow speed maneuvers, assumed that aggressive maneuvers are not necessary for such a task at the controlling stage. If we consider a 12-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 12-DOF. A similar simplified motion model was proposed in[28] for distributed MPC setup. On the contrary, 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 (Fig. 2). The following sections explain each component of the proposed trajectory tracker.

Simplified Motion Model

The system states and control inputs are described by $\mathbf{x}_k = [p_k^x, p_k^y, p_k^z, \alpha_k^z]^T \in \mathbb{R}^{n_x}$ and $\mathbf{u}_k = [v_k^x, v_k^y, v_k^z, \omega_k^z]^T \in \mathbb{R}^{n_u}$, respectively. p_k^i and $v_k^i, i \in \{x, y, z\}$ denote the quadrotor center position(m) and velocity (m/s) on each direction, i.e., x,y,z, at time t = k

in the world coordinate frame. α_k^z and ω_k^z denote the yaw angle (rad) and yaw rate (rad/s) around z axis, respectively. The simplified motion model is expressed by $\dot{\mathbf{x}}_k = \mathbf{f_c}(\mathbf{x}_k, \mathbf{u}_k)$

$$\dot{\mathbf{x}}_k = \begin{bmatrix} \dot{p}_k^x \\ \dot{p}_k^y \\ \dot{p}_k^z \\ \dot{\alpha}_k^z \end{bmatrix} = \begin{bmatrix} v_k^x cos(\alpha_k^z) - v_k^y sin(\alpha_k^z) \\ v_k^x sin(\alpha_k^z) + v_k^y cos(\alpha_k^z) \\ v_k^z \\ \omega_k^z \end{bmatrix}, \tag{1}$$

where $\mathbf{f_c}(\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$ and $n_x = n_u = 4$. The proposed tracking problem is formulated as a discrete dynamical model. Forward Euler discretization, $\mathbf{x}_{k+1} = \mathbf{f_d}(\mathbf{x}_k, \mathbf{u}_k)$ is introduced for a given sampling period in second, $\delta \in \mathbb{R} > 0$, we set $\delta = 0.05s$ in this experiment.

$$\mathbf{x}_{k+1} = \begin{bmatrix} p_k^x \\ p_k^y \\ p_k^z \\ \alpha_k^z \end{bmatrix} + \delta \begin{bmatrix} v_k^x cos(\alpha_k^z) - v_k^y sin(\alpha_k^z) \\ v_k^x sin(\alpha_k^z) + v_k^y cos(\alpha_k^z) \\ v_k^z \\ \omega_k^z \end{bmatrix}, \tag{2}$$

where $\mathbf{f_d}(.) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$.

Close-in Obstacles Identification

Close-in obstacle pose identification relative to the quadrotor is crucial for tracking a trajectory safely. In this work, OctoMap was utilized 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. Fig. 3 shows an instance view of close-in obstacle detection with respect to quadrotor pose. Each red dotted line depicts the distance between quadrotor and an obstacle position. Initially, close-in obstacles are estimated relative to the initial position of the quadrotor. After that, it uses the previous prediction horizon steps 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 same obstacle poses. Tracking problem synthesis subsection describes how obstacle constraints are incorporated into trajectory tracker.

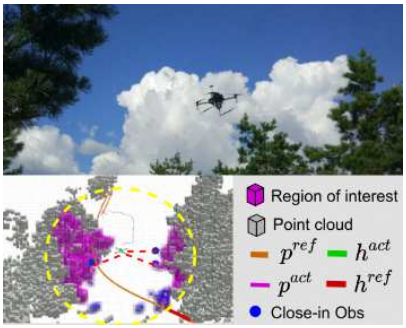


Figure 3. Illustration of close-in obstacle detection. Close-in obstacle pose estimation in respect of current prediction horizon (h_{act}) within the region of interest using Euclidean Distance Transformation Map (EDTM)

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 the these problems adequately. In this work, 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, i.e., $p^{knot} = \{t_0, t_1, ..., t_{n_k}\}$, and control points, i.e., $p^{ref} = \{p_0, p_1, ..., p_{n_p}\}$, where $t_* \in \mathbb{R}$, $p_* \in \mathbb{R}^d$ and $n_k = n_p + d + 1$; $*$ denotes indexing of p^{ref} and p^{knot}. Since d was set to 3, each p_i^{ref} depicts position, i.e., $p_i^{ref} =< x_i, y_i, z_i >$ in $\mathbb{R}^3$ where $i = 0, ..., n_p$. For a given time index, i.e., k corresponding position, $c^{ref}(k)$ can be fully

determined using DeBoorCox formula as given in Algorithm 1, which was initially proposed by de Boor[29]:

$$c^{ref}(k) = DeBoorCox(k, p^{ref}), \quad c^{ref}(k) \in \mathbb{R}^3. \tag{3}$$

Here $c^{ref}(k)$ does not exactly equal to p_i^{ref}. $c^{ref}(k)$ is continuous and it may or may not passing through control points, e.g., p_i^{ref} due to B-spline interpolation. We estimate reference velocity by taking the first derivative of p^{ref} and evaluating B-spline derivative using $\dot{c}^{ref}(k) = DeBoorCox(k, \dot{p}^{ref})$ since the derivative of B-spline is also a B-spline. More information about the way uniform B-spline trajectory are generated[12].

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

procedure DEBOORCOX(k, p)
  $t_k = \begin{cases} p^{knot}[d], & if\ k < p^{knot}[d] \\ p^{knot}[n_p], & if\ k > p^{knot}[n_p] \\ k, & otherwise \end{cases}$
  d_k = d
  while *true* **do**
    if $p^{knot}[d_k + 1] \geq t_k$ **then**
      break
    d_k++
  p_e[d]
  for $i \leftarrow 0 \quad to \quad d$ **do**
    $p_e[i] \leftarrow p[d_k - d + i]$
  for $r \leftarrow 1 \quad to \quad d$ **do**
    for $i \leftarrow d \quad to \quad r$ **do**
      $\beta \leftarrow \frac{t_k - p^{knot}[i+d_k-d]}{p^{knot}[i+1+d_k-r] - p^{knot}[i+d_k-d]}$
      $p_e[i] \leftarrow (1-\beta)p_e[i-1] + \beta p_e[i]$
  return $p_e[d]$

Tracking Problem Synthesis

The proposed trajectory tracker was initially defined as an OCP problem and then transformed into an NLP problem. A problem is always solved better in nonlinear manner as opposed to linearizing motion model at every time instance since the motion model is nonlinear. The above OCP problem can transform into NLP in various ways including, MS and DC. We implemented both MS and DC methods. The intuition of MS is to split system integration into small time intervals incorporating state constraints into the decision variables. NMPC can handle the constraints, which do not need to be convex. Hence, we selected NMPC as a closed-loop controller which handles a constrained non-linear system implicitly to solve the optimal control policy for a fixed prediction horizon $N_e \in \mathbb{N}$. The sections below explain the formulation of the proposed trajectory tracker with MS and DC.

With multiple-shooting

As detailed in reference trajectory generation subsection, $c^{ref}(t)$ and $\dot{c}^{ref}(t)$ give only position and velocity in the $\mathbb{R}^3$. Yaw and yaw rate were not incorporated for tracking because keeping the reference yaw does not have any meaningful impact when avoiding obstacles due to changed heading angle. Thus, we relaxed yaw and yaw rate from the tracking problem. $\mathbf{x}_k^{ref} = c^{ref}(k)$ and $\mathbf{u}_k^{ref} = \dot{c}^{ref}(k)$ are the reference state vector and control inputs (3). $\mathbf{x}_k$ and $\mathbf{u}_k$ are the state vector and control inputs (2). For simplicity, we used the same notation $\mathbf{x}_k$ and $\mathbf{u}_k$ when formulating NMPC, but NMPC skips the 4th entry in both state and control inputs, i.e., yaw and yaw rate. MS can be described as:

$$\begin{aligned} J_{N_e}(\mathbf{x}_k, \mathbf{u}_k) &= \sum_{h=0}^{N_e} \left\| \mathbf{x}_{k+h} - \mathbf{x}_{k+h}^{ref} \right\|_Q^2 + \left\| \mathbf{u}_{k+h} - \mathbf{u}_{k+h}^{ref} \right\|_R^2 \\ \min_{\mathbf{w}} \quad & J_{N_e}(\mathbf{x}_k, \mathbf{u}_k) \\ \text{s.t.} \quad & g_1(\mathbf{w}) = 0, \quad g_2(w) \leq 0 \\ & p^{lower} \leq \mathbf{x}_{k+h} \leq p^{upper} \quad \forall 0 \leq h \leq N_e \\ & u^{lower} \leq \mathbf{u}_{k+h} \leq u^{upper} \quad \forall 0 \leq h \leq N_e - 1, \end{aligned} \tag{4}$$

where $\mathbf{w} = [\mathbf{u}_k, \ldots, \mathbf{u}_{k+N_e-1}, \mathbf{x}_k, \ldots, \mathbf{x}_{k+N_e}]$ denotes the decision variables set to be minimized. $Q \in \mathbb{R}^{n_x \times n_x}$ and $R \in \mathbb{R}^{n_u \times n_u}$ 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. p^{upper} and p^{lower} depict the upper and lower bound of x_k, respectively. These bounds are enforced based on the map constraints where quadrotor is allowed to fly. u^{lower} and u^{upper} define the minimum and maximum linear and angular velocities allowed for the quarotor maneuver. $g_1(\mathbf{w})$ depicts the constraints that system dynamics imposes as follows:

$$g_1(\mathbf{w}) = \begin{bmatrix} \bar{\mathbf{x}}_k - \mathbf{x}_k \\ \vdots \\ f_d(\mathbf{x}_{k+h}, \mathbf{u}_{k+h}) - \mathbf{x}_{k+h+1} \\ \vdots \\ f_d(\mathbf{x}_{k+N_e-1}, \mathbf{u}_{k+N_e-1}) - \mathbf{x}_{k+N_e} \end{bmatrix}. \tag{5}$$

$g_2(\mathbf{w})$ 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. Subsequently, reconstructing $g_2(w)$ is useful to integrate the dynamic environment changes into the trajectory tracker as described in the equation below:

$$g_2(\mathbf{w}) = \begin{bmatrix} dis(\mathbf{x}_j^o, \mathbf{x}_k) \\ \vdots \\ dis(\mathbf{x}_j^o, \mathbf{x}_{k+h}) \\ \vdots \\ dis(\mathbf{x}_j^o, \mathbf{x}_{k+N_e}) \end{bmatrix}, \; j = 1, ..., N_o, \tag{6}$$

where $\bar{\mathbf{x}}_k$ is the initial position and N_o is the number of obstacles and $dis(\mathbf{x}_j^o, \mathbf{x}_{k+h})$ can be calculated as follows:

$$-\sqrt{(x_j^o - x_{k+h})^2 + (y_j^o - y_{k+h})^2 + (z_j^o - z_{k+h})^2} + d^o$$

where d^o is the safe zone distance between a quadrotor and close-in obstacles.

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. The first step is to define the collocation points with respect to a chosen polynomial when formulating DC. In this experiment, we chose Lagrangian 3rd order (N_d) polynomial to select collocation points. However, other polynomials such as B-spline or Bézier can be utilized as well. We fixed the time interval (δ) when defining g_1 in MS. Nonetheless, DC has more freedom to determine how points are distributed between two consecutive time interval.

To formulate DC, we kept the same discretization as in the MS, i.e., $\mathbf{u}_{k+i}, \mathbf{u}_{k+i+1}$, for $i \in [t_k, t_{k+1}]$, $\quad i = 0, .., N_d - 1$. Let η be the coefficient for 3rd order (N_d) Legendre points, $\eta = [0, 0.112, 0.500, 0.888]$. Subsequently, each consecutive time interval (t_{k+i} and t_{k+i+1}) was divided into small sub-intervals as follows:

$$t_{k,j} := t_k + h_k \eta_j, \quad k = 0, ..., N_e - 1, \; j = 0, 1, ..., N_d, \tag{7}$$

where $h_k = t_{k+1} - t_k$ and corresponding state vector at $t_{k,j}$ is denoted by $\mathbf{x}_{k,j}$. In each control interval, Langrangian polynomial is defined as

$$L_j(\eta) = \prod_{r=0, r \neq j}^{N_d} \frac{\eta - \eta_r}{\eta_j - \eta_r} \quad L_j(\eta) = \begin{cases} 1, & if \; j = r \\ 0, & otherwise \end{cases}. \tag{8}$$

State trajectory approximation ($\bar{\mathbf{x}}_k$) and intermediate derivatives, i.e., $\nabla \bar{\mathbf{x}}_{k,j}$, can be derived by using Langrangian functions (8) as $\bar{\mathbf{x}}_k = \Sigma_{r=0}^{N_d} L_r(\frac{t - t_k}{h_k}) \cdot \mathbf{x}_{k,r}$ and $\nabla \bar{\mathbf{x}}_{k,j} = \frac{1}{h_k} \Sigma_{r=0}^{N_d} \dot{L}_r(\eta_j) \cdot \mathbf{x}_{k,r}$, where $\dot{L}_r(\eta_j)$ depicts the derivative of $L_r(\eta_j)$. Plugging $\nabla \bar{\mathbf{x}}_{k,j}$ into (2), a set of collocation equations is constructed at each collocation point as follows:

$$h_k \cdot \mathbf{f_d}(\mathbf{x}_{k,j}, \mathbf{u}_k) - \Sigma_{r=0}^{N_d} \dot{L}_r(\eta_j) \cdot \mathbf{x}_{k,r} = 0, \tag{9}$$

where $k = 0, ..., N_e - 1$ and $j = 0, ..., N_d$. As we formulated NMPC in multiple-shooting subsection, for DC, only $g_1(\mathbf{w})$ is changed, i.e., (9), and the rest will remain the same.

NMPC output is post-processed to improve the smoothness using a KF smoother before sending to the controller as shown in Fig. 2. KF smoother is needed to reduce the noise of NMPC, which occurs when number of obstacles constraints are increased. Sub-optimal KF[30] 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), $\hat{\mathbf{x}}_{k+1} = A\hat{\mathbf{x}}_k + B\mathbf{u}_k = \mathbf{f_d}(\hat{\mathbf{x}}_k, \mathbf{u}_k)$. In KF smoother, $B = 0$ and $\hat{\mathbf{u}}_{k+1} = A\hat{\mathbf{u}}_k$ where A is an identity matrix in $\mathbb{R}^{n_x \times n_x}$ and B is given by

$$\delta \begin{bmatrix} cos(\theta(k)) & -sin(\theta(k)) & 0 & 0 \\ sin(\theta(k)) & cos(\theta(k)) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$\hat{\mathbf{x}}_k = < \underbrace{\hat{x}_k^x, \hat{x}_k^y, \hat{x}_k^z}_{\hat{\mathbf{p}}_k}, \hat{\alpha}_k^z >$ and $\hat{\mathbf{u}}_k = < \underbrace{\hat{v}_k^x, \hat{v}_k^y, \hat{v}_k^z}_{\hat{\mathbf{v}}_k}, \hat{\omega}_k^z >$ denote the expected values after applying KF and KF smoother, respectively. As explained in multiple-shooting subsection, yaw rate, i.e., α_k^z, was excluded from the tracking problem. Hence, yaw rate is estimated as follows:

$$\hat{\omega}_k^z = \begin{cases} atan2(\hat{v}_k^y, \hat{v}_k^x) & \hat{v}_k^y \geq 0 \\ atan2(\hat{v}_k^y, \hat{v}_k^x) + 2\pi & otherwise \end{cases}. \tag{11}$$

Estimated control and pose, $\hat{\mathbf{u}}_k$ and $\hat{\mathbf{x}}_\mathbf{k}$, respectively, and actual control and pose (see Fig. 2), $\mathbf{u}_k^{act} = < \mathbf{v}_k^{act}, \omega_k^{act} >$ and $\mathbf{x}_k^{act} = < \mathbf{p}_k^{act}, \alpha_k^{act} >$, respectively, are the inputs for the PD regulator that is formulated as follows for a given time index k:

$$v_k = k_p(\hat{\mathbf{p}}_k - \mathbf{p}_k^{act}) - k_d \mathbf{v_k}^{act}, \quad \omega_k = k_a(\hat{\alpha}_k^z - \alpha_k^{act}) - k_v \omega_k^{act}, \tag{12}$$

where constant terms ($k_p = 0.8, k_d = 0.27, k_a = 2.5$, and $k_v = 0.5$) were estimated empirically.

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. Firstly, trajectory tracker accuracy is analysed in an obstacle-free zone to show reference trajectory tracking accuracy. Afterwards, we compared the proposed trajectory tracker and the two other approaches. The second part of experiment lays out the qualitative analysis of the proposed trajectory tracker. The proposed trajectory tracker was implemented in C++11 with -O2 (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 on-board computer on the quadrotor as shown in Fig. 4.

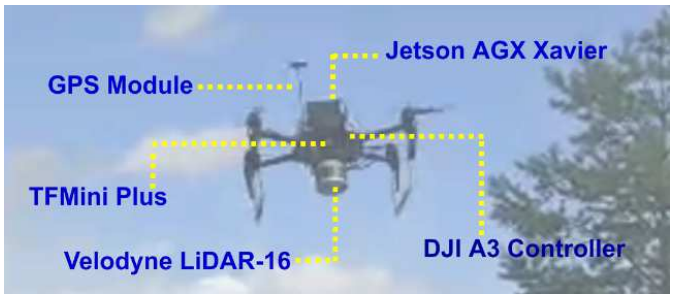


Figure 4. The hardware setup of the quadrotor. TFMini Plus is for estimating the altitude whereas Velodyne LiDAR-16 is for reasoning the environment

Checking the trajectory tracking accuracy is required for an obstacle-free zone since this is the initial proposal. Fig. 5 shows the reference trajectory employed for this experiment, and the complete test is available at[31]. Fig. 6b clearly shows that the quadrotor almost follows the reference trajectory adhering to the imposed constraints, e.g., max velocity (0.4m/s) (Fig. 6a).

The next experiment is devoted to comparing the proposed trajectory tracker with two other approaches. We used the same validation technique, which Oleynikova et al[32] utilized for checking the behaviour in different environments. We generated

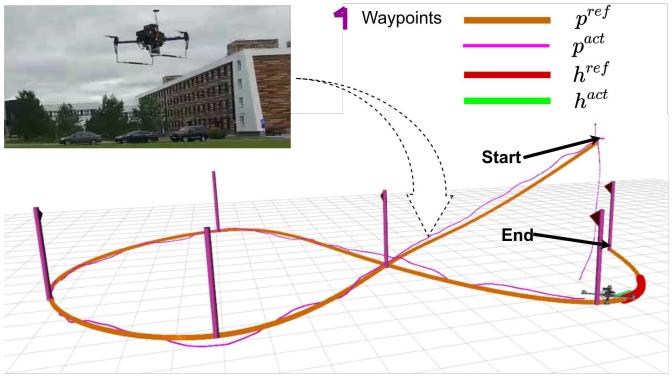
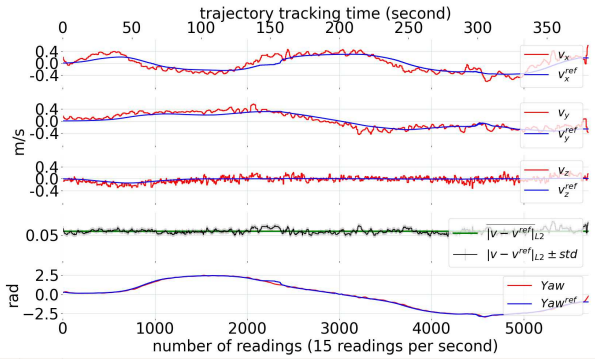
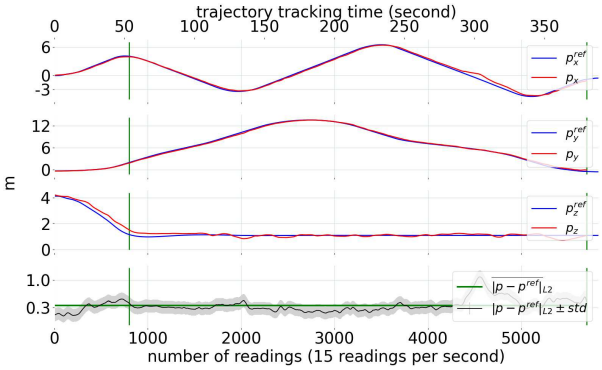


Figure 5. An example reference trajectory (p_{ref}) that was generated as proposed in the reference trajectory generation subsection. p_{act} is the actual trajectory along which the quadrotor flew using the proposed trajectory tracker



(a) Velocity and yaw angle profile; as shown in Subplot 4, average velocity error, $0.07 \pm 0.02 m/s$. To keep the low error rate, Kalman filter was employed for estimating NMPC output in which small fluctuations are eliminated



(b) Positional changes over time. Compared to x and y directions, along the z-axis has higher error; this is due to low accuracy reading of TFMini Plus. Average position error remains around $0.3 \pm 0.04 m$

Figure 6. The proposed trajectory tracker was evaluated using a reference trajectory (i.e., Fig. 5) in a place where no presence of close-in obstacles

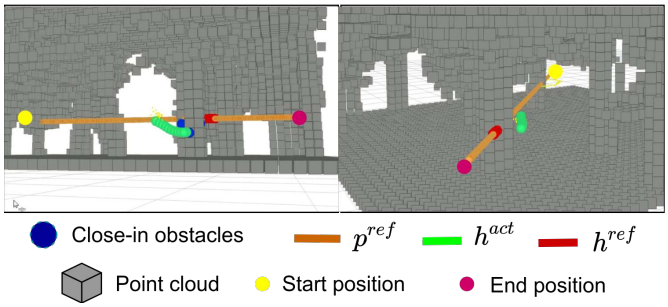


Figure 7. The result of the proposed trajectory tracker using a Poisson forest[33] 10x10x10 meters, of 0.2 trees/m^2. The start and goal positions were generated randomly keeping at least 4m apart

10 Poisson forests[33] with 10x10x10 of densities in between 0.2 trees/m and 0.8 trees/m. Since Usenko[34] also used the same validation test, we compared the proposed trajectory tracker with theirs as given in Table. 1; an example scenario is shown in Fig. 7. We kept all the other configuration parameters constant only the maximum velocity (V_{max}) was varied.

As seen in Table 1, the proposed trajectory tracker takes higher solving time as compared to the others. Higher solving

Table 1. Comparison of trajectory planning approaches

Algorithm	Success Fraction	Mean Norm Path Length	Mean Compute time[s]
[32] S = 2 jerk	0.48	1.1079	0.0310
[32] S = 3 vel	0.47	1.1067	0.09793
[32] S = 3 jerk	0.50	**1.0996**	**0.0367**
[32] S = 3 jerk + Restart	**0.63**	1.1398	0.1724
[34] C = 2	0.47	**1.0668**	**0.0008**
[34] C = 3	0.47	1.0860	0.0011
[34] C = 5	0.51	1.1502	0.0021
[34] C = 9	**0.57**	1.3008	0.0072
Proposed V_{max} = 0.5m/s, N_e = 25	0.88	1.097	**0.360**
Proposed V_{max} = 1.0m/s, N_e = 25	0.90	**1.0009**	0.4211
Proposed V_{max} = 1.5m/s, N_e = 25	0.90	1.0060	0.4234
Proposed V_{max} = 2.5m/s, N_e = 25	**0.91**	1.0265	0.498

S: the order of trajectory segment size with respect to minimizing parameters, i.e., jerk, vel, C: control points to be taken for the optimization, N_e: NMPC prediction horizon

time due to the high computational demand of NMPC. On the other hand, the proposed trajectory tracker could solve the path all most all the given test cases which imply having high success fraction (SF = $\frac{success\,test\,cases}{total\,test\,cases}$). The obstacle cost was incorporated as constraints rather than minimizing as part of the cost; that was the key to obtain 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 high possibility to crash the quadrotor. That is why the other two approaches are having less SF despite low Mean Computation Time (MCT = $\frac{total\,execution\,time}{total\,NMPC\,iterations}$). Mean Norm Path Length (MNPL) was calculated by ratio to the total distance of the projected path to straight line distance between start and goal pose.

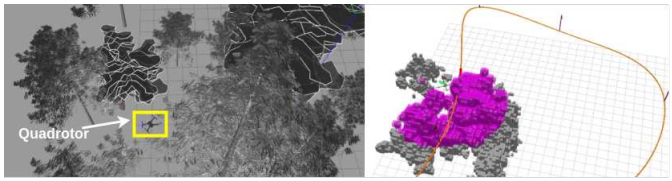
Table. 1 shows the dataset[33] we used for comparison. This dataset was not complicated enough for the proposed trajectory tracker. Additionally, this dataset was not created targeting for trajectory tracking. Thus, we proposed a new dataset that can be used for validating the performance of trajectory tracking algorithms in cluttered environments. The proposed dataset consists of 100 different reference trajectories as described in reference trajectory generation subsection. Each trajectory is provided with a set of intermediate waypoints (see Fig. 8). The number of waypoints is used to construct a reference trajectory is varying between four to sixteen. For more information about the dataset, refer to[1]. Table. 2 shows the evaluation of the proposed trajectory tracker against the proposed dataset. 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.

Table 2. Comparison of results on the proposed dataset

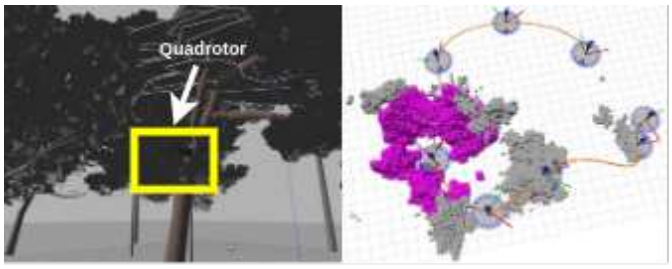
Technique	Success Fraction	Mean Norm Path Length	Mean Compute time[s]
MS, V_{max} = 0.5m/s , N_e = 25	0.8258	**0.9838**	**0.3895**
MS, V_{max} = 1.0m/s, N_e = 25	0.8431	1.0045	0.4217
MS, V_{max} = 1.5m/s , N_e = 25	0.8435	1.0974	0.4432
MS, V_{max} = 2.0m/s , N_e = 25	**0.8524**	1.0071	0.5191
DC, V_{max} = 0.5m/s , N_e = 25	0.9205	**1.528**	**0.6700**
DC, V_{max} = 1.0m/s , N_e = 25	0.9406	1.603	0.7985
DC, V_{max} = 1.5m/s , N_e = 25	0.9505	1.6019	0.7681
DC, V_{max} = 2.0m/s , N_e = 25	**0.9606**	1.623	0.9985

DC: Direct-collocation, MS: Multiple-shooting, N_e: Prediction horizon of the NMPC

CasADi[35] with Ipopt[36] 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 application,



(a) The quadrotor flying at the altitude of about 10m



(b) The quadrotor flying at low altitude

Figure 8. The sample reference trajectories that our dataset provides. Each trajectory either lays at low altitude only or fluctuates between low and high altitude, as shown in Fig. 8b

this behaviour 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 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.

Conclusion

We proposed an optimization-based trajectory tracking approach for slow-speed maneuvers and conducted qualitative and quantitative performance analysis. All in all, the proposed trajectory tracker can track a given reference trajectory while avoiding obstacles. The simplified motion model-based planner helped to reduce the computational demands when calculating optimal control policy. However, computational demand is yet to be further reduced to guarantee a fast reaction time. Moreover, fast reaction time improves safety maneuvering in cluttered environments. Forthcoming works will concentrate on further reducing the computational demands.

References

1. The proposed trajectory tracker: https://github.com/gprathap/trajectory_trackerv1.
2. Morari, M. & Lee, J. H. Model predictive control: past, present and future. *Comput. & Chem. Eng.* **23**, 667–682 (1999).
3. Stein, G. & Athans, M. The lqg/ltr procedure for multivariable feedback control design. *IEEE Transactions on Autom. Control.* **32**, 105–114 (1987).
4. Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C. & Burgard, W. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Auton. robots* **34**, 189–206 (2013).
5. Oleynikova, H., Taylor, Z., Fehr, M., Nieto, J. & Siegwart, R. Voxblox: Building 3d signed distance fields for planning. *arXiv* arXiv–1611 (2016).
6. van den Berg, J. Extended lqr: Locally-optimal feedback control for systems with non-linear dynamics and non-quadratic cost. In *Robotics Research*, 39–56 (Springer, 2016).
7. Trawny, N., Zhou, X. S., Zhou, K. & Roumeliotis, S. I. Interrobot transformations in 3-d. *IEEE Transactions on Robotics* **26**, 226–243 (2010).
8. Wanasinghe, T. R., Mann, G. K. & Gosine, R. G. Relative localization approach for combined aerial and ground robotic system. *J. Intell. & Robotic Syst.* **77**, 113–133 (2015).
9. Kulathunga, G., Fedorenko, R., Kopylov, S. & Klimehik, A. 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)*, 145–153 (2020).

10. Karaman, S. & Frazzoli, E. Sampling-based algorithms for optimal motion planning. *The international journal robotics research* **30**, 846–894 (2011).

11. Yang, K. & Sukkarieh, S. An analytical continuous-curvature path-smoothing algorithm. *IEEE Transactions on Robotics* **26**, 561–568 (2010).

12. Zhou, B., Gao, F., Wang, L., Liu, C. & Shen, S. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics Autom. Lett.* **4**, 3529–3536 (2019).

13. Zhou, B., Gao, F., Pan, J. & Shen, S. Robust real-time uav replanning using guided gradient-based optimization and topological paths. *arXiv preprint arXiv:1912.12644* (2019).

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

15. Van Nieuwstadt, M. J. & Murray, R. M. Real-time trajectory generation for differentially flat systems. *Int. J. Robust Nonlinear Control. IFAC-Affiliated J.* **8**, 995–1020 (1998).

16. Richter, C., Bry, A. & Roy, N. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research*, 649–666 (Springer, 2016).

17. Kamel, M., Stastny, T., Alexis, K. & Siegwart, R. Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. In *Robot operating system (ROS)*, 3–39 (Springer, 2017).

18. Reyes-Valeria, E., Enriquez-Caldera, R., Camacho-Lara, S. & Guichard, J. Lqr control for a quadrotor using unit quaternions: Modeling and simulation. In *CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing*, 172–178 (2013).

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

20. Islam, M., Okasha, M. & Sulaeman, E. A model predictive control (mpc) approach on unit quaternion orientation based quadrotor for trajectory tracking. *Int. J. Control. Autom. Syst.* **17**, 2819–2832 (2019).

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

22. Alessandretti, A., Aguiar, A. P. & Jones, C. N. Trajectory-tracking and path-following controllers for constrained underactuated vehicles using model predictive control. In *2013 european control conference (ecc)*, 1371–1376 (IEEE, 2013).

23. Gros, S., Quirynen, R. & Diehl, M. Aircraft control based on fast non-linear mpc & multiple-shooting. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 1142–1147 (IEEE, 2012).

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

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

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

27. Dji flight controller: https://www.dji.com/ru/a3.

28. Mehrez, M. W., Mann, G. K. & Gosine, R. G. An optimization based approach for relative localization and relative tracking control in multi-robot systems. *J. Intell. & Robotic Syst.* **85**, 385–408 (2017).

29. de Boor, C. Subroutine package for calculating with b-splines. *Los Alamos Sci. Lab. Rep. LA-4728-MS* (1971).

30. Van Der Merwe, R. & Wan, E. A. The square-root unscented kalman filter for state and parameter-estimation. In *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, vol. 6, 3461–3464 (IEEE, 2001).

31. The proposed trajectory tracker: https://www.youtube.com/watch?v=1yrybtryknq.

32. Oleynikova, H. *et al.* Continuous-time trajectory optimization for online uav replanning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5332–5339 (IEEE, 2016).

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

34. Usenko, V., von Stumberg, L., Pangercic, A. & Cremers, D. Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 215–222 (IEEE, 2017).

35. Andersson, J. A., Gillis, J., Horn, G., Rawlings, J. B. & Diehl, M. Casadi: a software framework for nonlinear optimization and optimal control. *Math. Program. Comput.* **11**, 1–36 (2019).

36. Wächter, A. & Biegler, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. programming* **106**, 25–57 (2006).

Author contributions statement

GK Proposed and implemented algorithm. DD and GK were in charge of real-world flight tests and creating a quadrotor for the tests. GK wrote the manuscript in consultation with AK. AK made critical reviewing the manuscript and final approval for publication.