Optimization-based Trajectory Tracking Approach for Multi-rotor Aerial Vehicles in Unknown Environments

Geesara Kulathunga[1], Hany Hamed[2], Dmitry Devitt[2], Alexandr Klimchik[2]

arXiv:2202.06056v1 [cs.RO] 12 Feb 2022

Abstract—The goal of this paper is to develop a continuous optimization-based refinement of the reference trajectory to 'push it out' of the obstacle-occupied space in the global phase for Multi-rotor Aerial Vehicles in unknown environments. Our proposed approach comprises two planners: a global planner and a local planner. The global planner refines the initial reference trajectory when the trajectory goes either through an obstacle or near an obstacle and lets the local planner calculate a near-optimal control policy. The global planner comprises two convex programming approaches: the first one helps to refine the reference trajectory, and the second one helps to recover the reference trajectory if the first approach fails to refine. The global planner mainly focuses on real-time performance and obstacles avoidance, whereas the proposed formulation of the constrained nonlinear model predictive control-based local planner ensures safety, dynamic feasibility, and the reference trajectory tracking accuracy for low-speed maneuvers, provided that local and global planners have mean computation times 0.06s (15Hz) and 0.05s (20Hz), respectively, on an NVIDIA Jetson Xavier NX computer. The results of our experiment confirmed that, in cluttered environments, the proposed approach outperformed three other approaches: sampling-based pathfinding followed by trajectory generation, a local planner, and graph-based pathfinding followed by trajectory generation.

Index Terms—Constrained Motion Planning, Planning under Uncertainty, Collision Avoidance.

I. INTRODUCTION

T HE reference trajectory tracking for multi-rotor aerial vehicles (MAVs) is used in various domains, e.g., cinematography, or landing on a moving platform. Even though many approaches have been proposed for tracking specified reference trajectories [1]–[3], it remains an open research problem due to several reasons: achieving real-time performance, avoiding close-in obstacles, adhering to different weather conditions, etc. Subsequently, generating a near-optimal control policy for maneuvering through a cluttered unknown environment is a rather challenging task when enforcing the dynamic

[1] Artificial Intelligence Research Center, Innopolis University, Russia ggeesara@gmail.com

[2] Centre for Robotics and Mechatronics Components, Innopolis University, Russia h.hamed@innopolis.university, d.devitt@innopolis.ru, a.klimchik@innopolis.ru

Fig. 1. Experiment with the proposed trajectory tracker to showcase how the proposed approach works in real conditions. In this experiment, the total distance of R was about 54m, and the length of P_{t_n} was set to 10

feasibility and safety constraints in real-time. Model Predictive Control (MPC) is one of the promising techniques to address such challenging tasks. However, due to the computational aspects of MPC, it is difficult to achieve real-time performance in most situations using limited available resources [4]. Such aspects are mainly because of the way the problem is formulated, e.g., as an NMPC (Nonlinear MPC), as an LMPC (Linear MPC), and the way constrains, e.g., obstacles and inputs, are handled. Moreover, the accuracy of the near-optimal policy generation of MPC depends on the sensing capabilities, e.g., FoV (Field of View), sensing distance, and the way surrounded free space and obstacles are represented. For instance, free space can be formed as a set of convex polyhedrons along the refined reference trajectory. Afterwards, incorporating a linear motion model, the problem can be formulated as convex rather than non-convex.

This paper proposes an optimization-based approach that solves two trajectories simultaneously, when the first one tries to refine the initial reference trajectory pushing the reference trajectory away from the known obstacles, while the second one generates a near-optimal control policy at every planning step incorporating the refined trajectory (Fig.1). Thus, the **contributions** of this work are as follows:

1) Developing a framework for reference trajectory tracking, ensuing safety and dynamic feasibility in which the global planner refines the reference trajectory allowing the local planner to generate a near-optimal control policy quickly at every planning iteration

2) Proposing a fast approach, formulated as a convex problem, for pushing the reference trajectory away from obstacle zones, where we implemented a parallel version of Convex Decomposition [5] (Algorithm 1, line 9) and a simplified approach, as compared to conservative

$\mathbf{x}_{t_n} = \{x_{t_n}^x, x_{t_n}^y, x_{t_n}^z, \alpha_{t_n}^z\}, \mathbf{u}_{t_n} = \{v_{t_n}^x, v_{t_n}^y, v_{t_n}^z, \omega_{t_n}^z\}$

Fig. 2. The high-level system architecture of the proposed trajectory tracker. The global and local planners run in parallel as two separate threads while sharing the reference trajectory. Each of the components, i.e., Global, Local, Mapper, average frequencies, are estimated on an NVIDIA Jetson Xavier NX computer (see Fig. 10), which is utilized for real-world experiments

approaches in prior work, for time allocation

3) Real-world and simulated experiments that showcase the agile flights in various unknown cluttered environments and a new dataset we used for benchmarking our approach with the three other approaches

II. RELATED WORK

Most of the recent trajectory tracking approaches are formulated as **optimization problems** where all the constraints are incorporated in close loop manner. Moreover, such constraints can be embodied as a part of the objective or as a part of the constraints, which can be either soft or hard constraints [6]–[8]. Such problem formulation, i.e., employing both objective and constraints, can belong to one of the types: convex [9] or non-convex [4] (non-linear). Non-convex problem formulation is usually computationally expensive. In the recent work of Liu et al. [10], a successive convex decomposition-based free-space representation, i.e., a series of overlapping polyhedra, was proposed. Such a free-space representation helps to keep MAV within the free space for the given interval. Specific interval allocation can be calculated in several different ways: let the solver allocate intervals [11] or define several intervals prior to solving. Once intervals are allocated, different methods can be used for time allocation (fixed or adaptive) [5].

Trajectory tracking problem can be solved during two different planning stages: local or/and global, in which local and global planners can be formulated as two separate or one combined optimization problem. As far as MAVs are concerned, most of the approaches exploit differential flatness property, where the smoothness and dynamic feasibility are estimated by minimizing the L2 norm of velocity difference over the trajectory [12]. However, problem formulation can be complex when enforcing different constraints [13], e.g., obstacle, time, and input constraints. Thus, in the literature, various approaches have been proposed to handle the said constraints. The most primitive paradigm is to use path planning followed by trajectory generation [14], [15], which can be considered as an open-loop problem. However, such approaches fail due to the high computation time, as well as when the environment

is highly dynamic and cluttered. To reduce the computation time and have fast reaction time, motion primitive-based local trajectory planning [16]–[18] were proposed. Such approaches are often trapped in local minima. Hence, the objectives of **global planer and local planner** can differ mainly due to the expected nature (or characteristics) of problem formulation as follows:

The functions and characteristics of a local planner are: it plans for a local near-optimal trajectory based on the currently perceived information within the close vicinity with higher accuracy [3], [19]; it generates near-optimal control policy in an online fashion in every iteration; it performs trajectory smoothing and feasibility checking to ensure the differential/dynamic constraints; the trajectories are planned consecutively, depending on the way they handle the next set of information that comes in, e.g., how they react to dynamic and static obstacles and how they decide whether to incorporate previous information [1], [2]; long horizon-based trajectory planning may generate wasteful unnecessary long trajectories [4].

The functions and characteristics of a global planner are: it tries to plan the global near-optimum trajectories [20]; it constructs the map of the way, either memory-less or fusion-based [17], (a memory-less map does not consider any previous information, but rather considers only current map; a fusion-based map building does not discard stale data, i.e., previous information, which might be problematic for dynamic obstacles, i.e., some free-known space could be considered as occupied-known space [14]); with the known mapping, global planner generates the obstacle-free kinematically feasible trajectory that often ensures the differential/dynamic constraints; it minimizes backtracking and generates efficient trajectories in cluttered environments [17] in which maintaining a clear picture of the environment is a heavy burden on computational perspective; the computational complexity depends on how much information is incorporated and the way a problem is formulated, e.g., constrained/unconstrained, linear/non-linear optimization problem [8], [9], [20].

Therefore, safety, appropriate maneuver, dynamic feasibil-

ity, and real-time performance are the main objectives that trajectory tracker must have. The safety and feasible trajectory generation of local or global planners depend on the manner free-known and free-unknown spaces are incorporated. Hence, planning space can be defined to lie within the sensor' FOV or outside of it [21]–[23], provided that a series of sensor data has been incorporated for constructing the environment map. When the environment is cluttered, local planners perform poorly due to the uncertainty of instance sensing data. Prediction horizon-based planning is commonly used for such environments, in which local planners can be more conservative compared to global planners. Such a local planner can be formed in multiple ways, e.g., Linear Model Predictive Control (MPC) [24], **Nonlinear Model Predictive Control (NMPC)** [4] , and Corridor-based Model Predictive Contouring Control(CMPCC) [25], based on the necessity and the requirements. The global planner must be less conservative compared to the local planner, e.g., when defined as an unconstrained function minimizer. In the proposed approach, the local planner is formed as an NMPC, whereas the global planner is formed as a box-constrained function minimizer.

Accurate **environment mapping** is important to perform robust planning. Out of many, memory-less and fusion-based are the main methods that are used for mapping the environment [9]. Memory-less methods rely on the instantaneous data, i.e., most of the time only on the last sensor reading, whereas fusion-based methods - on the stacking sensor readings in a specific form, e.g., Octomap [26], Voxblox [27], as a map. Such methods may have considerable estimation error and high computation time that depend on the hardware and sensors capabilities. However, estimation error that emerged due to drift and poor sensing measurement can be overcome by resetting the fusion from time to time. Therefore, the latter methods are preferred over the memory-less methods specially for reasoning of cluttered environments due to several reasons, such as limited FOV and the lack of prior information about previous sensing data. Once a map is constructed, Euclidean distance transform mapping (EDTM) [28] can be utilized to estimate the free distance from a considered position. For the mapping, we built instantaneous EDTM on top of Octomap.

III. Methodology

The proposed approach uses a parallel architecture, which consists of a local and a global planner, and where the global planner pushes the initial reference trajectory away from obstacle zones O_m, whereas the local planner generates an optimal control policy for tracking the modified reference trajectory by global planner. The high-level view of the proposed reference trajectory tracker is given in Algorithm 1. A pictorial visualization of the notion that is used throughout the paper is shown in Fig. 3. The known obstacles and unknown obstacles are defined as $O_m \in \widehat{M}$ and $O_u \notin \widehat{M}$, respectively. Similarly, $F_m \in \widehat{M}$ and $F_u \notin \widehat{M}$ denote the known free space and unknown free space. Hence, $F_m \cup O_m \subseteq \widehat{M}$ and all the unknown region becomes $\mathbb{R}^3 \backslash \widehat{M} \subseteq (O_u \cup F_u)$. Initial reference trajectory, namely R consists of a set of control points: $c_i, i = 0, ..., N_c$, where the number of control points

Fig. 3. Notion used for defining the reference trajectory tracker and different spaces to the current pose Q_p of MAV

is given by N_c. For generating R, we used the approach proposed in [4], which is based on uniform bspline. The initialization time of the trajectory planning and current time, where the desired pose on the reference trajectory should lie, are denoted by t_0 and t_n, respectively. For a given time t_n, starting and finishing control points are retrieved with respect to $c_s \subseteq [0, N_c)$ and $c_e \subseteq (c_s, N_c - 1]$ indices. The time difference between two consecutive control points, i.e., c_i and c_{i+1}, is defined as δ_{t_d}, which was set to 0.05s in our experimental setup. The actual discretization time interval of continuous system dynamics δ_{t_c} was set to 0.05. Besides, δ_{t_c} and δ_{t_d}, which both can be the same or slightly different from each other, can be configured. The number of control points within the R_{t_n}, namely, refining horizon, is denoted by N_r. The avoidance distance, which is the minimum free distance D_z allowed in between MAV and the closest obstacle to MAV.

Algorithm 1 Reference trajectory tracker

 Inputs: at time t_n, R_{t_n}: reference trajectory to be refined, Q_p: current pose of MAV, P_{t_n}: trajectory to be tracked , M_{t_n}: EDT map of the environment
 Outputs: R_{t_n}: refined reference trajectory; $P_{t_n} \in R_{t_n}$, v_x, v_y, v_z, ω_z : control command to maneuver MAV;

---

 procedure Global Planner
  $R_{t_n} \leftarrow\ <Q_p, R_{t_n}>$
  $S_o \leftarrow$ CheckingOccupiedSegments(R_{t_n}, M_{t_n})
  if $S_o > 0$ **then**
   for $i \leftarrow S_o$ **do**
    $A_i, b_i \leftarrow$ ParallelConvexDecomposition(S_o^i)
    $S_*^i \leftarrow$ FindPushingDirections(S_o^i, A_i, b_i)
    $R_{t_n} \leftarrow$ CalculateGradients(S_*^i, R_{t_n})
  return $R_{t_n} \leftarrow$ ApplyBoxConstraintOptimization(R_{t_n})

---

 procedure Local Planner
  $P_{t_n} \leftarrow\ <Q_p, P_{t_n}>$
  $C_o \leftarrow$ GetCloseInObstacles(P_{t_n}, M_{t_n})
  return $< v_x, v_y, v_z, \omega_z >\leftarrow$ ApplyNMPC(P_{t_n}, C_o)

A. Global Planner

The proposed approach consists of two planning stages: local and global. The local planner is designed as a constraint

nonlinear optimization problem (NLP), specifically an NMPC. The computation time of NMPC increases when the number of constraints increases, i.e., reference trajectory lies closer or within the obstacles. Such behaviour can lead to local minima and cannot find near-optimal control policy to avoid close-in obstacles. Hence, the global planner does refine the initial reference trajectory in parallel with the local planner to push the reference trajectory away from the obstacle zones. Hence, the proposed global planner is formulated as follows:

$$J = \lambda_{smooth} J_{smooth} + \lambda_{obs} J_{obs} + \lambda_{feasibility} J_{feasibility}, \tag{1}$$

where $\lambda_*, * \in \{smooth, obs, feasibility\}$ are weight parameters were set as 0.2, 0.6, and 0.2, respectively. λ_{smooth} and $\lambda_{feasibility}$ were set to low values mainly due to relax smoothness and feasibility adjustment compared to obstacle avoidance adjustment by putting high penalty weight λ_{obs}. In the following sub-sections, formulation of global planner components is explained adhering to Algorithm. 1.

1) Finding Pushing Direction: Some of the control points in R_{t_n} can occur within the obstacles zones. Hence, control points that lie within the obstacle zone O_m must be pushed towards an obstacle-free zone F_m. Let $S_o^i \in R_{t_n}$ be the ith segment within R_{t_n} to be modified. S_o^i consists of a set of control points $\mathbf{c}_j \in S_o^i, j = 0, ..., N_i^{seg}$, where N_i^{seg} is the number of control points in S_o^i. Thus, pushing direction of each control point is determined by solving the following convex problem for each segment:

$$\min_{\mathbf{p}_0,...,\mathbf{p}_n} \lambda_1 t_1 + \lambda_2 t_2 + \lambda_3 t_3$$
$$\text{s.t.} \quad A\mathbf{p}_j \leq b,$$
$$\|\mathbf{p}_0 - \mathbf{c}_0\|_2 \leq t_1,$$
$$\|\mathbf{p}_n - \mathbf{c}_n\|_2 \leq t_2, \tag{2}$$
$$\sum_{k=1}^{n-1} \|\mathbf{p}_{k+1} - \mathbf{p}_k\|_2 \leq t_3,$$

where A and b represents the free space F_m as a convex polyhedron from $\mathbf{c}_0$ to $\mathbf{c}_n$ in S_o^i (Sec.III-A2). For the ith segment, $n = N_i^{seg}$. The regularization parameters: $\lambda_1 = 0.8, \lambda_2 = 0.8$, and $\lambda_3 = 0.6$, were set in a way to provide more bias on start and end control points compared to middle control points. $\mathbf{p}_0, ..., \mathbf{p}_{N^{seg}}$ construct the updated segment S_*^i corresponding to S_o^i that will be used for finding each control point's gradient direction (Sec. III-A3).

2) Parallel Convex Decomposition: To further reduce the computation time, we have implemented the parallel version of Convex Decomposition [5] (Algorithm 1, line 9). Once desired control points are identified, i.e., $\mathbf{c}_j \in S_o^i, j = 0, ..., N_i^{seg}$ (Sec.III-A1), check for intermediate control points that are in F_m. Convex decomposition is applied to such successive control points in parallel that result in the free space in the form of H-rep $Ax \leq b$ for each S_o^i.

3) Calculating Gradients: The objective of J_{obs} is to push each $S_o^i, i = 0, ..., N^{seg}$ segment towards the obstacle-free zone. N^{seg} is the number of segments that are within the obstacle zone for the considered refine trajectory segment R_{t_n}, at time t_n. For the ith segment, by knowing S_o^i, S_*^i can be

determined (III-A1). To find each gradient direction vector that crosses the $\mathbf{c}_j \perp S_*^i$, $j = 0, .., N_i^{seg}$, let $\mathbf{v}_1 = \mathbf{c}_{j+1} - \mathbf{c}_{j-1}$ be the approximated direction vector along $\mathbf{c}_j$, and $\mathbf{p}_k \in S_*^i$ be the control point that intersects $\mathbf{v}_1$ (Fig. 4). Then, the corresponding direction vector $\mathbf{v}_2$ can be defined as $\mathbf{p}_k - \mathbf{c}_j$. By calculating angle $\theta = cos^{-1}(\mathbf{v}_1 \cdot \mathbf{v}_2 / \|\mathbf{v}_1\|_2 \|\mathbf{v}_2\|_2)$ between $\mathbf{v}_1$ and $\mathbf{v}_2$, the optimal value of k can be determined as provided in Algorithm 2. Thus, the gradient vector that corresponds to c_j can be fully determined as:

$$\mathbf{c}_j^{grad} = (\mathbf{c}_j^* - \mathbf{c}_j)/ \|\mathbf{c}_j^* - \mathbf{c}_j\|_2,$$
$$\mathbf{c}_j^* = \mathbf{p}_k + \frac{(\mathbf{p}_k - \mathbf{p}_{k-1})(\mathbf{v}_1 \cdot (\mathbf{c}_j - \mathbf{p}_k)) \cdot}{\mathbf{v}_1 \cdot (\mathbf{p}_k - \mathbf{p}_{k-1})} \tag{3}$$

Algorithm 2 Estimation of direction vectors pushing the control points towards the free space.

Inputs: at time t_n, R_{t_n}: reference trajectory to be refined, N^{seg}: segments indices to be refined within R_{t_n}
Outputs: R_{t_n}: after adding, gradient vector corresponds to each control point

---

procedure GRADIENTESTIMATION
  for $i \leftarrow 0$ to N^{seg} **do**
    for $j \leftarrow 1$ to N_i^{seg} **do**
      $k \leftarrow N_i^{seg}/2$
      $\mathbf{v}_1 = \mathbf{c}_{j+1} - \mathbf{c}_{j-1}, \ \mathbf{v}_2 = \mathbf{p}_k - \mathbf{c}_j$
      $\mathbf{c}_j \in S_o^i, \ \mathbf{p}_k \in S_*^i$
      $val = previous\_val \leftarrow \mathbf{v}_1 \cdot \mathbf{v}_2$
      while $k \geq 0$ and $k < N_i^{seg}$ **do**
        $k \leftarrow \begin{cases} k--, & if \ val \leq 0 \\ k++, & otherwise \end{cases}$
        $val \leftarrow \mathbf{v}_1 \cdot \mathbf{v}_2$
        if $val \cdot previous\_val \leq 0$ **then**
          $\mathbf{c}_j^* \leftarrow \mathbf{p}_k + \frac{(\mathbf{p}_k - \mathbf{p}_{k-1})(\mathbf{v}_1 \cdot (\mathbf{c}_j - \mathbf{p}_k))}{\mathbf{v}_1 \cdot (\mathbf{p}_k - \mathbf{p}_{k-1})}$
          $\delta d \leftarrow \|\mathbf{c}_j^* - \mathbf{c}_j\|_2$
          $\mathbf{c}_j^{grad} = \frac{\mathbf{c}_j^* - \mathbf{c}_j}{\|\mathbf{c}_j^* - \mathbf{c}_j\|_2}$
  return R_{t_n}

Once gradient vectors are estimated, J_{obs} is determined (4), which is defined as a continuously differentiable exact penalty function.

$$J_{obs} = \Sigma_{i=d}^{N_r-d} J_{obs_i},$$
$$J_{obs_i} = \mathbf{v}_i \cdot dis_e^3, \quad \frac{\partial J_{obs_i}}{\partial \mathbf{c}_i} = -3 \cdot dis_e^2 \cdot \mathbf{c}_i^{grad}, \tag{4}$$

where $dis_e = D_z - (\mathbf{c}_i - \mathbf{c}_i^*) \cdot \mathbf{c}_i^{grad}$ and $\mathbf{v}_i = \mathbf{c}_{i+1} - \mathbf{c}_i$, and avoidance distance D_z was set to 0.8m (distance must be higher than the radius of the MAV) in our study.

4) Dead Zone Recovery: The map construction is not precise when the depth sensor has a small FoV. Moreover, EDTM building takes a considerable amount of time when the environment is cluttered. Therefore, the local planner may generate control commands that lead to quadrotor Q_p maneuvers into the D_z (Fig. 3) zone. In such situations, free space segmentation (Sec.III-A2) does not provide correct constraints set A, b. Hence, the proposed approach (2)

Fig. 4. Pushing control points that are within the obstacle zone, towards the free space. Projected control points segment (S_*^i) is obtained as explained in Sec. III-A1 utilizing S_o^i for i^{th} segment. c_j^* depicts gradient vector corresponding to c_j

fails to estimate control points $\mathbf{p}_0, ..., \mathbf{p}_{N_i^{seg}}$ appropriately, i.e., estimated control points may lie in the extreme ends $(\mathbf{c}_0, \mathbf{c}_{N_i^{sec}})$ of the provided trajectory S_o^i, provided that R_{t_n} is not dynamically feasible. Hence, the objective is to consider whole R_{t_n} rather than each segment separately, followed by the free space segmentation. Thus, the following recovery mechanism is proposed to push the R_{t_n} away from O_m. The recovery mechanism is executed only when the (2) is failed.

$$\min_{\mathbf{p}_1, ..., \mathbf{p}_{N_r}} \sum_{l=1}^{N_r} q_l$$
$$\text{s.t.} \quad A_r(\mathbf{p}_l + \mathbf{c}_l) \leq b_r, \ \|\mathbf{p}_l\|_2 \leq q_l, \ l = 1, ..., N_r, \quad (5)$$

where $c_l, l = 1, ..., N_r$ are the control points to be pushed. The recovered control points are determined by $\mathbf{p}_l + \mathbf{c}_l$, N_r is the number of control points at time t_n in R_{t_n}, and A_r and b_r are obtained by giving R_{t_n} to Algorithm 1, line 9.

5) Smoothing: We have employed a velocity controller since the proposed trajectory tracker targets low-speed maneuvers. Hence, higher-order components, i.e, acceleration, jerk, snap, should be minimized, which causes effects such as vibrations. However, we decided to minimize only acceleration components without considering higher-order components, e.g., jerk, snap. We have formulated J_{smooth} minimizing both acceleration and jerk components as well as only considering acceleration components. However, adding jerk did not affect J_{smooth} considerably. Thus, J_{smooth} was formulated only as minimizing the acceleration components:

$$J_{smooth_i} = \mathbf{a}_i^\top \mathbf{a}_i, \quad \frac{\partial J_{smooth_i}}{\partial \mathbf{c}_i} = 2 \frac{\partial \mathbf{a}_i}{\partial \mathbf{c}_i}, \quad (6)$$

where $\partial \mathbf{a}_i / \partial \mathbf{c}_i = 1$, $\partial \mathbf{a}_i / \partial \mathbf{c}_{i+1} = -2$, $\partial \mathbf{a}_i / \partial \mathbf{c}_{i+2} = 1$. $\mathbf{a}_i, \mathbf{v}_i, \mathbf{c}_i \in \mathbb{R}^3$ are respectively acceleration ($\mathbf{a}_i = \mathbf{c}_{i+2} - 2\mathbf{c}_{i+1} + \mathbf{c}_i$), velocity ($\mathbf{v}_i = \mathbf{c}_{i+1} - \mathbf{c}_i$), and control point at i^{th} index of R_{t_n}.

6) Feasibility: To ensure the refined trajectory, namely, R_{t_n}, which is dynamically feasible for the maneuver, objective function penalizes the velocity and acceleration components only when their limits exceed the min and max, as follows:

$$J_{feasibility_i} = (\mathbf{v}_i \oplus \mathbf{v}_{max})^\top (\mathbf{v}_i \oplus \mathbf{v}_{max}) \cdot \frac{1}{\delta^2}$$
$$+ (\mathbf{a}_i \oplus \mathbf{a}_{max})^\top (\mathbf{a}_i \oplus \mathbf{a}_{max}) \quad (7)$$

where the operator $\oplus$ is defined as

$$\oplus = \begin{cases} - & if \ \mathbf{v}_i > \mathbf{v}_{max} \parallel \mathbf{a}_i > \mathbf{a}_{max} \\ + & if \ \mathbf{v}_i < -\mathbf{v}_{max} \parallel \mathbf{a}_i < -\mathbf{a}_{max} \ , \\ not \ considering & otherwise \end{cases}$$
$$(8)$$

where allowed maximum velocity and acceleration components are given by $\mathbf{v}_{max} \in \mathbb{R}^3$ and $\mathbf{a}_{max} \in \mathbb{R}^3$, respectively. When the velocity and acceleration components are within the allowed range, there will be no added cost. Once objective function J (1) was formed, we have used L-BFGS-B (Limited-memory Broyden Fletcher Goldfarb Shanno Box-constrained algorithm) [29] for solving J. Subsequently, Mosek solver [30] was employed to solve (2) and (5).

B. Local Planner

At time t_n, $P_{t_n} = [\mathbf{Q}_p, \mathbf{c}_{t_n}, \mathbf{c}_{t_n+1}, ..., \mathbf{c}_{t_n+N_p}] \subseteq R_{t_n}$ forms the reference trajectory for the given prediction horizon, N_p. The local planner generates the optimal control to maneuver the quadrotor considering close-in obstacles $g_2(\mathbf{w})$ and system dynamics $g_1(\mathbf{w})$ where $\mathbf{w} = [\mathbf{u}_{t_n}, ..., \mathbf{u}_{t_n+N_p-1}, \mathbf{x}_{t_n}, ..., \mathbf{x}_{t_n+N_p}]$. Hence, the objective of local planner is to optimize both control inputs and states simultaneously. Such an objective can be designed using multiple shooting technique as follows:

$$J_P(\mathbf{x}, \mathbf{u})_{t_n} = \sum_{l=0}^{N_p} \|\mathbf{x}_{t_n+l} - \mathbf{c}_{t_n+l}\|_Q^2 + \left\|\mathbf{u}_{t_n+l} - \mathbf{v}_{t_n+l}^{ref}\right\|_R^2$$
$$\min_{\mathbf{w}} \quad J_P(\mathbf{x}, \mathbf{u})_{t_n}$$
$$\text{s.t.} \quad g_1(\mathbf{w}) = 0, \quad g_2(\mathbf{w}) \leq 0$$
$$\mathbf{x}_{min} \leq \mathbf{x}_{t_n+l} \leq \mathbf{x}_{max} \quad \forall 0 \leq l \leq N_p$$
$$-\mathbf{v}_{max} \leq \mathbf{u}_{t_n+l} \leq \mathbf{v}_{max} \quad \forall 0 \leq l \leq N_p - 1, \quad (9)$$

At every planning cycle, local planner gets $Q_p, \mathbf{x}_{t_n}$, and $\mathbf{u}_{t_n}$ as an input and estimates the optimal control policy, i.e., $\hat{\mathbf{u}}_{t_n} = \{\hat{v}_{t_n}^x, \hat{v}_{t_n}^y, \hat{v}_{t_n}^z, \hat{\omega}_{t_n}^z\}$, where $\hat{v}_{t_n}^\mu, \mu \in x, y, z$ denotes velocity on each μ direction, and yaw angle around z axis is given by $\hat{\omega}_{t_n}^z$. The local planner is adopted from our previous work, where the explanation of $g1$, $g2$ and $\hat{\mathbf{u}}_{t_n}$ is detailed [4].

To conclude, as summarized in Algorithm. 1, this section explained how the proposed trajectory tracker is formulated. In the following section, the qualitative and quantitative analysis of the proposed approach is provided.

IV. EXPERIMENTAL PROCEDURE AND RESULTS

The experiment prototype of MAV DJI M100 (Fig. 5) is equipped with the following components: Velodyne Lite 16 lidar for reasoning the MAV's surrounding environment;

Nvidia Jetson NX computer for online computations. Offline computations and the simulated experiments were carried out on an Intel i9-9900K (16) @ 5 GHz computer. Hence, the timing breakdowns were measured for both real-world and simulated experiments by those two computers. The simulated experiments were performed in a Gazebo environment. For the simulated and real-world experiments, PX4 [31] and DJI A3 controllers were employed, respectively.

Fig. 5. The experimental prototype MAV (DJI M100) was used in real-world experiments. The proposed approach runs on the on-board computer (Nvidia Jetson NX) and sends control commands to A3 controller

The first experiment[2] was aimed to estimate reference trajectory tracking error without refining. Such trajectory tracking can be directly used in, for example, cinematography. Moreover, this is a way to check the local planner (9) is able to track the reference trajectory that the proposed global planner provides. As shown in Fig. 6, we estimated position estimation error $|p - p^{ref}|_2$ between the tracked trajectory p and the reference trajectory p^{ref}. The fusion of GPS position, IMU data, and vehicle velocity is used to estimate p, whereas p^{ref} is the output of local planner. The mean estimation error $\overline{|p - p^{ref}|_2}$ was less than 1m during the whole flight in which velocity varied in between -1.2 m/s to 1.2 m/s.

Fig. 6. Experimental results for real-world tracking accuracy without considering obstacles. Estimated tracking error is less than 1m during the whole experiment

The second experiment was devoted to demonstrating the behaviour of the proposed approach in a real-world condition[3], where the initial reference trajectory passes through a cluttered environment followed by open space and back to a cluttered environment where the terminal pose was placed in an obstacle zone. Map update range ($update\_range$) was kept 4m from the center of the MAV and max speed set to 0.6m/s for the safety of MAV. Total flight time was around 150s and the distances of initial reference trajectory and traversed trajectory were 54.3m and 79.8m, respectively. Since trajectory termination pose was within the obstacles, trajectory tracker terminates early(see Fig. 8). Such a behaviour is due the

---

[2] tracking accuracy without considering obstacles: https://www.youtube.com/watch?v=pKVeGdr8crU

[3] behaviour of the proposed approach in a challenging environment https://youtu.be/g6xHvkcrYcQ

Fig. 7. The trajectory tracking error for real-world experimental results on tracking the reference trajectory shown in Fig. 6

fact that the global planner was designed as a box-constraint function minimizor, whereas local planner was designed as a constraint NLP. Hence, the local planner terminated correctly, though the global planner completely failed to refine, which is in fact true.

Fig. 8. Showcasing the behaviour of the proposed trajectory tracker in a challenging environment

In the third experiment[4], we conducted four different real-world tests to estimate the run-time breakdown (mean computation time) in the average case. Three out of four tests were performed in static environments: open area with small obstacles, open area with sizeable obstacles, and a cluttered environment) and the fourth experiment was performed in a dynamic environment (Fig.9). Reference trajectories of each of them were completely different from each other. However, we fixed the trajectory tracking duration to 90s. Afterwards, run-time breakdown (Fig.10) was estimated based on three sub-modules: NMPC solver (main force of the local planner), EDT mapper (utilize both local and global planner), and main parts of the global planner (smoothing, feasibility, calculating gradients, and finding pushing directions). The objective was to understand how the run-time of each of the listed sub-modules is affected due to environmental changes. Since those three modules were executed in parallel, local and global

---

[4] experiments used for estimating the run-time https://www.youtube.com/watch?v=jyDe5BSigm8

TABLE I
COMPARATIVE ANALYSIS OF THE PROPOSED APPROACH AND THE THREE OTHER APPROACHES FOR CHECKING GOAL-REACHING ACCURACY. ALL THE METRICS WERE OBTAINED DURING EXPERIMENTS ON 12 DIFFERENT ENVIRONMENTS WHILE KEEPING THE SAME START AND GOAL POSES

Algorithm	Success Fraction (SF)	Mean Computation Time (MCT) in seconds	Distance Estimation (m)		
			Mean	Max	Min
[15] RRT* max_allowed_iterations=10000	0.41	2.4	112.56	198.45	91.04
[4] $P_{t_n} = 20, update\_range = 5m$	0.58	0.397	93.5	154.67	68.45
[4] $P_{t_n} = 40, update\_range = 5m$	0.66	0.441	98.91	201.56	78.23
[9] $N_{whole} = N_{safe} = 6\ max\_poly = 3$	0.83	≈ 0.01	54.56	81.45	49.39
[9] $N_{whole} = N_{safe} = 12\ max\_poly = 6$	**0.83**	$\approx$ **0.01**	53.78	78.67	47.45
Proposed $P_{t_n} = 10, update\_range = 4m$	0.91	0.04 ± 0.01	56.78	68.78	49.86
Proposed $P_{t_n} = 15, update\_range = 6m$	**1.0**	**0.03 ± 0.01**	54.89	66.80	48.67

N_{whole}, N_{safe} : the number of discretization points in the whole and safe trajectory, max_poly: maximum number of polydrons to represent the free space, P_{t_n}: NMPC prediction horizon length

planners have mean computation times of approximately 0.06s (15Hz) and 0.05s (20Hz), respectively.

(a)   (b)   (c)   (d)

Fig. 9. Different scenarios for real-world experiments: static (a,b,d) and dynamic (c) were used to estimate the run-time breakdowns of the proposed trajectory tracker (Fig.10)

Fig. 10. Estimation of mean computation time (run-time) of the proposed approach, i.e., time break down of each sub components, in real-world scenarios (see Fig.9)

In the final experiment, we have generated 12 random forests, e.g., Fig. 11(a), where density (40m×40m×10m) was kept the same for all the environments. The three other methods: RRT* [15], a local planner [4], and FASTER [9] were used to validate the proposed approach. The results are provided in Table I and an example test case is shown in Fig. 11. When the environment is cluttered, FASTER failed mainly due to the inability to find a path to local goal pose using JPS [32]. In the proposed approach, the dead zone recovery technique tries to recover when the global planner fails to refine the trajectory and the local planner is also capable of planning ahead independently from the global planner. In consequence, the proposed approach has a higher success rate (number of times successfully reach the goal) compared to the other methods despite mean computation time (MCT) (ratio

of total execution time to the total number of iterations) is slightly lower than FASTER. Since we are targeting low-speed maneuver, MCT is also acceptable. In each environment, the same start and goal poses were considered and the distance between them was set to 38.6m, ensuring no obstacle presence on those poses. There is no distinctive difference in the mean distance estimation between the proposed and FASTER.

(a)   (b)   (c)

Fig. 11. An example of testing the proposed (b) and FASTER (c) algorithms on a randomly generated forest (a)

V. CONCLUSION

This work presents a reference trajectories tracking approach for low-speed agile flights ensuring safety and dynamic feasibility in completely unknown environments. The essential properties of the proposed approach are online trajectory refinement and near-optimal control policy generation in parallel in horizon-based fashion, while only reasoning the surrounding environment. The proposed approach was tested on various simulated and real-world environments, achieving long range trajectory tracking. The local and global planners have mean computation times of approximately 0.06s (15Hz) and 0.05s (20Hz), respectively, provided that tracking accuracy is less than 1m in obstacle-free zones. We expect to extend this work for high-speed maneuvers in which we are going to focus on improving the local planner. The source code and complete experiments are available at Github[4]

[4] The source code and complete experiments - https://github.com/GPrathap/trajectory-tracker.git

References

[1] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6753–6760.

[2] M.-E. Guerrero-Sánchez, O. Hernández-González, G. Valencia-Palomo, F.-R. López-Estrada, A.-E. Rodríguez-Mata, and J. Garrido, "Filtered observer-based ida-pbc control for trajectory tracking of a quadrotor," *IEEE Access*, vol. 9, pp. 114 821–114 835, 2021.

[3] O. Mechali, L. Xu, Y. Huang, M. Shi, and X. Xie, "Observer-based fixed-time continuous nonsingular terminal sliding mode control of quadrotor aircraft under uncertainties and disturbances for robust trajectory tracking: Theory and experiment," *Control Engineering Practice*, vol. 111, p. 104806, 2021.

[4] G. Kulathunga, D. Devitt, and A. Klimchik, "Trajectory tracking for quadrotors: an optimization-based planning followed by controlling approach," *doi.org/10.21203/rs.3.rs-963714/v2*.

[5] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[6] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon" next-best-view" planner for 3d exploration," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1462–1468.

[7] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5332–5339.

[8] W. Merkt, V. Ivan, and S. Vijayakumar, "Continuous-time collision avoidance for trajectory optimization in dynamic environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7248–7255.

[9] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 1934–1940.

[10] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and optimization*, vol. 56, no. 4, pp. 2712–2733, 2018.

[11] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, "Aggressive quadrotor flight through cluttered environments using mixed integer programming," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1469–1475.

[12] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2520–2525.

[13] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.

[14] G. Kulathunga, D. Devitt, R. Fedorenko, S. Savin, and A. Klimchik, "Path planning followed by kinodynamic smoothing for multirotor aerial vehicles (mavs)," in *2020 International Conference Nonlinearity, Information and Robotics (NIR)*. IEEE, 2020, pp. 1–7.

[15] G. Kulathunga, R. Fedorenko, S. Kopylov, and A. Klimehik, "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)*. IEEE, 2020, pp. 145–153.

[16] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE transactions on robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[17] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[18] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 2872–2879.

[19] R. Tallamraju, E. Price, R. Ludwig, K. Karlapalem, H. H. Bülthoff, M. J. Black, and A. Ahmad, "Active perception based formation control for multiple aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4491–4498, 2019.

[20] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "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)*. IEEE, 2017, pp. 215–222.

[21] J. Tordesillas, B. T. Lopez, J. Carter, J. Ware, and J. P. How, "Real-time planning with multi-fidelity models for agile flights in unknown environments," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 725–731.

[22] B. T. Lopez and J. P. How, "Aggressive collision avoidance with limited field-of-view sensing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1358–1365.

[23] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2020.

[24] M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 773 − 11 780, 2014, 19th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016434890

[25] J. Ji, X. Zhou, C. Xu, and F. Gao, "Cmpcc: Corridor-based model predictive contouring control for aggressive drone flight," *arXiv preprint arXiv:2007.03271*, 2020.

[26] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[27] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[28] I. Ragnemalm, "The euclidean distance transform in arbitrary dimensions," *Pattern Recognition Letters*, vol. 14, no. 11, pp. 883–888, 1993.

[29] Lbfgs++. [Online]. Available: https://lbfgspp.statr.me/

[30] Mosek. [Online]. Available: https://www.mosek.com/

[31] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 6235–6240.

[32] K. Zhou, L. Yu, Z. Long, and S. Mo, "Local path planning of driverless car navigation based on jump point search method under urban environment," *Future Internet*, vol. 9, no. 3, p. 51, 2017.