2023 International Conference on Unmanned Aircraft Systems (ICUAS)
June 6-9, 2023 | Lazarski University, Warsaw, Poland

2023 International Conference on Unmanned Aircraft Systems (ICUAS) | 979-8-3503-1037-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICUAS57906.2023.10156337

An Integrated Real-time UAV Trajectory Optimization and Potential Field Approach for Dynamic Collision Avoidance

D. M. K. K. Venkateswara Rao, Hamed Habibi, Jose Luis Sanchez-Lopez, and Holger Voos

***Abstract*— This paper presents an integrated approach that combines trajectory optimization and Artificial Potential Field (APF) method for real-time optimal Unmanned Aerial Vehicle (UAV) trajectory planning and dynamic collision avoidance. A minimum-time trajectory optimization problem is formulated with initial and final positions as boundary conditions and collision avoidance as constraints. It is transcribed into a nonlinear programming problem using Chebyshev pseudospectral method. The state and control histories are approximated by using Lagrange polynomials and the collocation points are used to satisfy constraints. A novel sigmoid-type collision avoidance constraint is proposed to overcome the drawbacks of Lagrange polynomial approximation in pseudospectral methods that only guarantees inequality constraint satisfaction only at nodal points. Automatic differentiation of cost function and constraints is used to quickly determine their gradient and Jacobian, respectively. An APF method is used to update the optimal control inputs for guaranteeing collision avoidance. The trajectory optimization and APF method are implemented in a closed-loop fashion continuously, but in parallel at moderate and high frequencies, respectively. The initial guess for the optimization is provided based on the previous solution. The proposed approach is tested and validated through indoor experiments.**

Experiment video link: https://youtu.be/swSspfvYjJs

Artificial Potential Field, Dynamic Collision Avoidance, Pseudospectral Method, Trajectory Optimization, UAV.

I. Introduction

Trajectory planning is one of the most important features for autonomous Unmanned Aerial Vehicle (UAV) in cluttered environments, defined as a time-parameterized motion reference, i.e., geometric values of position, heading, derivatives associated with time law, passing through the waypoints, considering geometrical feasibility, collision avoidance, kinematics and dynamics [1]. One of the main objectives of the trajectory planner is collision avoidance with obstacles, humans, and other robots. These, generally, can be titled as static and dynamic obstacles [2]. So, the information on the obstacles is to be implemented into the proposed solution. Collision avoidance is considered as the geometrical feasibility of the planning. Most of the planning approaches are constructed as an on/offline optimization problem, obtaining a trajectory with the optimized feature [3]. Mostly, the time that the UAV takes between two given points is minimized, aiming at the agile maneuver. More importantly, the real-time implementation is more desirable for navigation in dynamic environments [4], [5]. This implies the need for an algorithm that converges to a feasible solution with a low computational burden. In terms of technical development, the representation of the optimization problem with the considered constraints usually determines the reliability of the solution in practice.

Trajectory optimization has been widely investigated in the last two decades [6]. Hard-constrained methods are pioneered, in which piecewise polynomial trajectories are generated through quadratic programming [7]. Free space can be represented by a sequence of cubes [8], spheres [3] or polyhedrons [9]. Improper time allocation of polynomials, e.g., naive heuristics, leads to unsatisfying results. Fast marching [10] and kinodynamic [11], [10] are used to find a feasible initial guess, ensuring global optimality by the convex formulation. However, distance to obstacles in the free space is ignored, which often results in trajectories being close to obstacles. Moreover, kinodynamic constraints are conservative, making the trajectory's speed deficient for fast flight [12]. A feasible solution can only be obtained by iteratively adding more constraints and solving the quadratic programming problem, which is undesirable for real-time applications.

Soft-constrained methods translate the problem to a nonlinear optimization problem with smoothness and safety [13]. Since the time parameterization is continuous, it avoids numeric differentiation errors, with more accuracy to represent motions. However, it suffers from a low success rate. As a resolution, in [14] a feasible initial path is found using an informed sampling-based path-searching method. In [15], uniform B-spline is used for the trajectory parameterization, as B-spline is continuous by nature, the continuity constraint is avoided. Moreover, it is beneficial for local re-planning,

D. M. K. K. Venkateswara Rao, H. Habibi, J. L. Sanchez-Lopez, and H. Voos are with Automation and Robotics Research Group, Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg. H. Voos is also with Faculty of Science, Technology and Medicine (FSTM), Department of Engineering, University of Luxembourg, {mohan.dasari,hamed.habibi,holger.voos, joseluis.sanchezlopez}@uni.lu

*This research was partially supported by the European Union's Horizon 2020 project Secure and Safe Multi-Robot Systems (SESAME) under the grant agreement no. 101017258 and by the Department of Media, Telecommunications and Digital Policy (SMC) of the Government of the Gran Duchy of Luxembourg under the project reference SMC/CFP-2019/010/IRANATA, "Interference and RAdiation in Network PlAnning of 5G AcTive Antenna Systems," and Fonds National de la Recherche of Luxembourg (FNR), under the projects C19/IS/13713801/5G-Sky.

due to its locality property. Soft-constrained methods might stick in local minima and cannot guarantee of success rate and kinodynamic feasibility.

Gradient-based method is the mainstream approach [4], [5], as a non-linear optimization, taking into account smoothness, feasibility, and safety using various parameterization methods, including polynomial and B-spline. The computation time can be reduced by compact environment representation [16]. Furthermore, it is effective for local replanning, which is useful for high-speed flight in unknown environments. However, the local minima are still an issue. The stochastic sampling strategy partially overcomes this issue with the cost of computationally intensive [17]. The quality of the initial guess can improve the success rate [14]. [18] used an iterative post-process to improve the success rate of [19].

Topological path planning method utilises the idea of topologically distinct paths for planning, in which paths belonging to different homotopy are used to escape local minima. Trajectory planning is presented in [20] in distinctive topologies using Voronoi and sampling-based front-ends and Timed-Elastic-Bands local planner [21] as back-ends. Based on [22], in [4] a real-time topological planning is proposed by efficient topology equivalence checking. In most of the similar works, collision avoidance in dynamic environments is considered, considering the velocity of obstacles [23], decentralized Nonlinear Model Predictive Control (NMPC) [24] and sequential NMPC [25]. Uncertainties can be handled by enlarging bounding volumes [26], which might lead to conservative or infeasible solutions. Moreover, the chance-constrained approaches are computationally intensive and thus not eligible for real-time collision avoidance [27].

Motivated by the above-mentioned considerations, in this paper we design an integrated solution for trajectory optimization with the Artificial Potential Field (APF), satisfying real-time minimum-time UAV trajectory planning and dynamic collision avoidance. The problem is transcribed into a nonlinear programming problem using Chebyshev pseudospectral methods. The state and control histories are approximated by using Lagrange polynomials and collocation is used to satisfy the dynamics constraints. The obstacle avoidance constraint is modelled by a novel sigmoid function to overcome the drawbacks of Lagrange polynomial approximation in pseudospectral methods that only guarantees inequality constraint satisfaction only at nodal points. It also guarantees the feasibility of the optimization problem. Furthermore, to reduce the computational complexity, automatic differentiation is used to obtain the gradient of cost function and Jacobian of constraints. An APF method is used to update the optimal control inputs for guaranteeing collision avoidance. The proposed framework structure can overcome convergence issues, by having the components run in parallel but at moderate and high frequencies.

The rest of the paper is organized as follows. The UAV dynamics model, optimization formulation, and NLP transcription are covered in Sec. II. Sec. III comprises the novel constraint formulation for collision avoidance, the APF method for guaranteed collision avoidance, and computational implementation. The proposed architecture is also presented. Indoor lab setup, and experimental results are presented in IV. Finally, the paper ends with conclusions in Sec. V.

II. Problem Formulation and Preliminaries

A. Dynamics Model

Consider an approximate dynamics model for the UAV as a first-order dynamics system in terms of the position vector $\underline{x}(t) = [x(t), y(t), z(t)]^T \in \mathbb{R}^3$ and commanded position vector $\underline{u}_{cmd}(t) = [x_{cmd}(t), y_{cmd}(t), z_{cmd}(t)]^T \in \mathbb{R}^3$, both presented in the inertial frame of reference, as

$$\dot{\underline{x}}(t) = diag(K_x, K_y, K_z)\,(\underline{u}_{cmd}(t) - \underline{x}(t))\,, \tag{1}$$

where K_x, K_y, and K_z are constant dynamics gains.

The model (1) assumes that the position of the UAV converges to the commanded position asymptotically [1]. The low-level controller in the autopilot rejects any unknown disturbances, like wind, and computes the required motor inputs based on the error between current and commanded positions, with desirable performance.

B. Optimization Problem Formulation

The trajectory planning problem is to determine the control inputs required to transfer the UAV from the given initial to final positions in minimum time. The trajectory planning problem is formulated to determine optimal position vector $\underline{x}_{opt}(t) = [x_{opt}(t), y_{opt}(t), z_{opt}(t)]^T \in \mathbb{R}^3$ and corresponding commanded position vector $\underline{u}_{cmd}(t)$, such that to steer $\underline{x}(t)$ towards $\underline{x}_{opt}(t)$, in minimum time, starting from the initial position $\underline{x}(t_0) = \underline{x}_0 \in \mathbb{R}^3$ to the final position $\underline{x}(t_f) = \underline{x}_f \in \mathbb{R}^3$, where t_f is the unknown terminal time. Also, it is aimed to avoid n_o moving obstacles with the position vector $\underline{x}_{obs,k}(t) = [x_{obs,k}(t), y_{obs,k}(t), z_{obs,k}(t)]^T \in \mathbb{R}^3$, for $k \in \{1, ..., n_o\}$, by keeping the UAV outside of safety spheres, considered around the obstacles. Furthermore, there are n_w waypoints that UAV is supposed to pass through before reaching $\underline{x}_f$. This optimization problem is mathematically formulated as

$$[\underline{x}_{opt}(t), \underline{u}_{cmd}(t)] = \underset{\underline{u}_{cmd}(t)}{\operatorname{argmin}} J\left(\underline{x}(t), \underline{u}_{cmd}(t), \underline{x}_0, \underline{x}_f\right), \tag{2}$$

where

$$J(\cdot) = t_f\left(\underline{x}(t), \underline{u}_{cmd}(t), \underline{x}_f\right) - t_0, \tag{3}$$

subject to dynamics (1) and boundary conditions

$$\underline{x}(t_0) = \underline{x}_0, \tag{4a}$$
$$\underline{x}(t_f) = \underline{x}_f, \tag{4b}$$
$$\underline{x}(t_j) = w_j, \tag{4c}$$

bound constraints on position and speed

$$\underline{x}_L \leq \underline{x}(t) \leq \underline{x}_U \tag{5a}$$
$$\underline{u}_L \leq \underline{u}_{cmd}(t) \leq \underline{u}_U \tag{5b}$$
$$\dot{\underline{x}}_L \leq \dot{\underline{x}}(t) \leq \dot{\underline{x}}_U \tag{5c}$$

and obstacle avoidance constraints

$$R_k \leq \left\|\underline{x}(t) - \underline{x}_{obs,k}(t)\right\|, \tag{6}$$

for $j \in \{1,...,n_w\}$ and $k \in \{1,...,n_o\}$, where, $t_0 \leq t_j \leq t_f$, are increasing time sequence, and $w_j = [x_j(t), y_j(t), z_j(t)]^T \in \mathbb{R}^3$ is the position of the j^{th} waypoint. X_L and X_U denote the element-wise lower and upper bound vectors on the variable vector $X(t)$. Finally, R_k for $k \in \{1,...,n_o\}$, represents safety radius of k^{th} obstacle. In fact, $R_k \leq \left\|x(t) - x_{obs,k}(t)\right\|$ imposes the constraint to keep the position of the UAV outside of the safety sphere around the obstacle.

C. NLP Transcription:

The continuous-time trajectory optimization problem (2) is transcribed into a nonlinear programming (NLP) problem using Chebyshev pseudospectral method [28]. In this method, the state and control histories between the initial and final waypoints are approximated by Lagrange polynomials in non-dimensionalized time, as illustrated in Figure 1

$$\underline{x}(\tau) \approx \sum_{i=1}^{n} \underline{x}_i \phi_i(\tau), \tag{7a}$$
$$\underline{u}_{cmd}(\tau) \approx \sum_{i=1}^{n} \underline{u}_i \phi_i(\tau), \tag{7b}$$

where $\tau \in [0,1]$ is the non-dimensionalized time; n is the order of the polynomial; $\underline{x}_i$ and $\underline{u}_i$ are state and control approximation vectors at nodes; i denotes node number; and $\phi_i(\tau)$ is the basis function given by

$$\phi_i(\tau) = \prod_{j=1, j\neq i}^{n} \frac{\tau - \tau_j}{\tau_i - \tau_j}, \tag{8}$$

where τ_i and τ_j are roots of shifted Chebyshev polynomials obtained by

$$\tau_i = \frac{1 + \tilde{\tau}_i}{2}. \tag{9}$$

The roots of the Chebyshev polynomial are determined as

$$\tilde{\tau}_i = \cos(\pi k/n), \quad k = 0,1,...,n \tag{10}$$

Let Δt be the maneuver time. The actual and non-dimensionalized times are related as

$$\tau = \frac{t}{\Delta t}, \tag{11}$$
$$d\tau = \frac{dt}{\Delta t}. \tag{12}$$

The Lagrange polynomials are collocated at the interpolation points to result in a set of algebraic equations

$$\underline{x}_i \dot{\phi}_i(\tau) = \dot{\underline{x}}(\tau). \tag{13}$$

This can be rewritten as

$$\sum_{i=1}^{n} \underline{x}_i \dot{\phi}_i(\tau) = \Delta t\, f(\underline{x}_i, \underline{u}_i). \tag{14}$$

Then, the cost function J, in (2) reduces to

$$J = \Delta t. \tag{15}$$

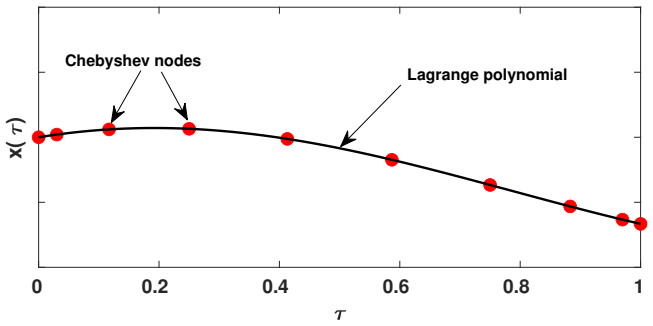

Fig. 1. Illustration of trajectory approximation using Chebyshev pseudospectral method.

The problem (2) with cost function (15) with boundary conditions (4), bound constraints (5), and obstacle avoidance constraints (6), constitute the NLP problem. This NLP problem is solved using interior point algorithm [29] implemented in IPOPT solver [30] interfaced with Matlab. Moreover, CasADi [31] is used for the automatic differentiation to determine the gradient and Jacobian of cost function and constraints, respectively, for speeding up the computation of the solution, which is vital for real-time implementation.

III. Collision Avoidance

In this section the modifications applied to the optimization problem (15), are addressed that guarantee the feasibility of the problem and the fast convergence of the solver, aiming at the real-time re-optimization. The overall computational architecture is presented in Figure 2.

A. Constraints Formulation

To apply the obstacle avoidance constraint (6) to the collocation points, the common condition to be checked is the distance of the points from i^{th} obstacle to be greater than the safety radius R_i for $i \in \{1,...,n_o\}$. However, this might lead to the safety violation as illustrated in Figure 3. Evidently, the collocation points $C_{j,1}$ and $C_{j,2}$ are outside of the safety sphere, i.e., $r_{i,j,1} > R_i$ and $r_{i,j,2} > R_i$. However, the line segment $\overline{C_{j,1}C_{j,2}}$ is still passing through the safety sphere. The trivial solution is to increase the number of collocation points and apply the distance constraint. However, this increases the computational burden. More importantly, it is still not geometrically guaranteed that connecting lines are moved outside.

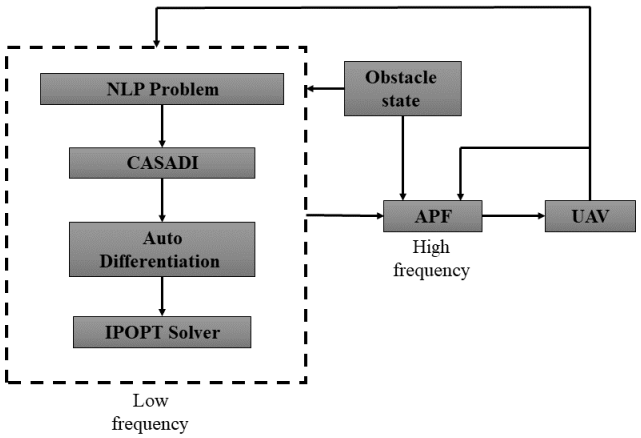

Fig. 2. Computational architecture.

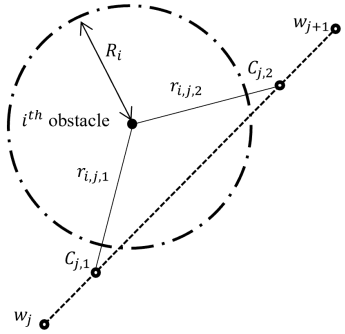

Fig. 3. Violation of safety region for the case the waypoints and collocation points are still outside.

A resolution is to consider the perpendicular distance for each line segment from the obstacle, i.e., $d_{i,j,k}$, for $k = 0,\ldots,N$, where N represents the number of collocation points, $C_{j,0} = w_j$ and $C_{j,N+1} = w_{j+1}$, as illustrated in Figure 4, where, $d_{i,j,k}$, for $k = 0,\ldots,4$, are all same for all the segments, as they are on the same straight line. Considering only perpendicular distance as the constraint $d_{i,j,k} > R_i$ leads to a very conservative approach, as it moves all points $C_{j,1}$ and $C_{j,4}$ to satisfy the constraint, as shown in Figure 5. This takes a lot of workspace and might lead to an infeasible optimization problem after applying the other constraints. Therefore, we only need the constraint to be applied to $C_{j,2}$ and $C_{j,3}$ in Figure 5, as they lie within the safety sphere. The solution we propose is to first check if the projection of i^{th} obstacle position on the line segment $\overline{C_{j,k}C_{j,k+1}}$ lies on the segment or its extension. To do so, we compute

$$t_{i,j,k} = \frac{\left(x_{obs,i} - C_{j,k}\right).\left(C_{j,k+1} - C_{j,k}\right)}{\left(C_{j,k+1} - C_{j,k}\right).\left(C_{j,k+1} - C_{j,k}\right)}, \qquad (16)$$

where $\cdot$ represents the inner product. If $0 < t_{i,j,k} < 1$, then the projection point is on the line segment $\overline{C_{j,k}C_{j,k+1}}$. Otherwise, it is on its extension. Accordingly, the collision avoidance constraint, illustrated in Figure 7, is as follows.

if $0 < t_{i,j,k} < 1$ & $d_{i,j,k} < R_i$ **then**

Move the $C_{j,k}$, for $k = 1,\ldots,N$, such that $\overline{C_{j,l}C_{j,l+1}}$, for $l = 0,\ldots,N$ is outside of the safety sphere.

end if

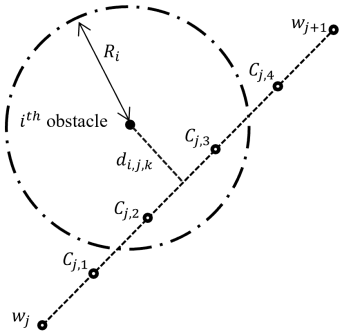

Fig. 4. Perpendicular distance check.

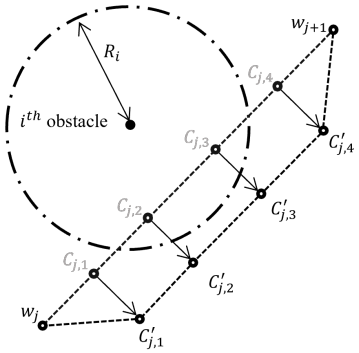

Fig. 5. Applying the perpendicular distance check to all the points. The points with superscript ($'$) represent the new position of the points in grey.

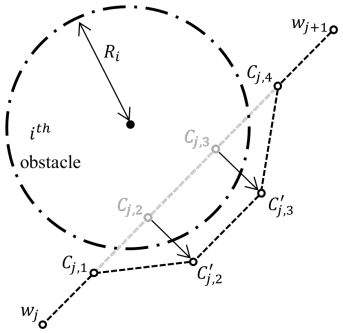

Fig. 6. Applying the projection point check.

This approach theoretically solves the mentioned problem, however, it imposes a practical implementation issue. This stems from the presence of the *if* condition in the construction of constraints which is a common approach for obstacle avoidance constraints [5]. Using *if* condition changes the structure and size of the constraints at every iteration. This is due to the fact that by moving collocation points, the constraint might be satisfied and it is removed

from the constraints matrix. More importantly, *if* condition is not a smooth condition which might make a problem with the convergence of the solver. To resolve this problem, we propose a smooth constraint avoiding the use of *if* condition, as

$$f_{i,j,k} := \frac{1}{2}\Gamma_{i,j,k}\left(R_i - d_{i,j,k}\right) \leq 0, \tag{17}$$

where $\Gamma_{i,j,k} = \tanh\left(\frac{t_{i,j,k}}{\delta}\right) - \tanh\left(\frac{t_{i,j,k}-1}{\delta}\right)$ and $\delta > 0$ is a small scalar. It is readily shown that for $0 < t_{i,j,k} < 1$, $\Gamma_{i,j,k} \approx 1$. Therefore, $f_{i,j,k} = R_i - d_{i,j}$. Then the constraint $f_{i,j,k} \leq 0$ is equivalent to $R_i - d_{i,j,k} \leq 0$. On the other hand, for $t_{i,j,k} \leq 0$ or $1 \leq t_{i,j,k}$, the term $\Gamma_{i,j,k} = 0$ and, hence, $f_{i,j,k} = 0$. Moreover, the condition $0 \leq 0$ is already satisfied in numerical solvers and it does not apply the distance constraint on the points $C_{j,k}$ and $C_{j,k+1}$. So, the constraint (17) encapsulates both conditions for the line segment on which the projection point lies, without using *if* condition. More importantly, this constraint is smooth at points $t_{i,j,k} = 0$ and $t_{i,j,k} = 1$. $f_{i,j,k}$ is illustrated in Figure for $\delta = 0.05$ and $R_i = 1.5$.

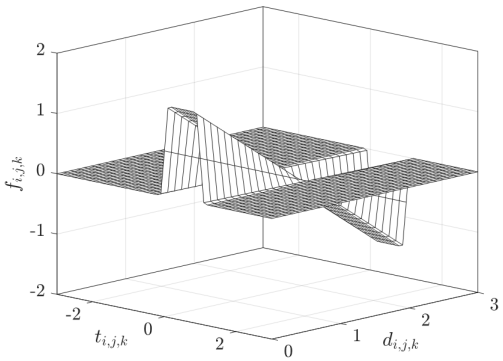

Fig. 7. Smooth obstacle avoidance constraint.

Another improvement to guarantee real-time feasibility with fast convergence is to make sure, the initial guess does not go through the safety sphere. To address this, we first radially move the collocation points of the initial guess out of the sphere. For each point, we check if it is inside of the sphere, if so, that point is moved in the direction of the carrying radius, i.e., connecting the centre to the point. However, this does not necessarily guarantee that the connecting line does not pass through the sphere. Therefore, we can consider the perpendicular distance of the centre to each line. However, just moving all the points according to the perpendicular distance might make lead to a conservative initial guess which is away from the optimal solution. So, we only move the lines, on which the projection point lies, not on their extension, in the perpendicular direction. This improvement might make the initial guess non-smooth and longer than the direct connecting line. So, finally, we shorten the guess, i.e., we check if the direct line between the collocation points pas through the sphere. if not, the direct line is replaced.

B. Artificial Potential Field

An APF method using sigmoid function [32] is used to make minor corrections to the optimal control inputs to improve safety and guarantee collision avoidance in case of any unexpected delays in the computation of the optimization solution. In the case of any unexpected delay in communication or convergence issue, this ensures obstacle avoidance and safety. APF method, as illustrated in Figure 8, creates a virtual field around the obstacle that increases or decreases in strength as the distance to it decreases or increases, respectively. In this paper, the sigmoid function is used to design the APF, as

$$F_k(t) = \frac{1}{2}\left(1 + tanh(\alpha R_k - \|r_k(t)\|)\right) - \eta \tag{18}$$

for $k \in \{1, ..., n_o\}$, where $F_k(t)$ is the repelling force, α and η are tuning parameters, and $r_k(t) = \underline{x}(t) - \underline{x}_{obs,k}(t)$. By solving the optimization problem (2) with cost function (15) with boundary conditions (4), bound constraints (5), and obstacle avoidance constraints (17), the optimal control inputs, i.e. $\underline{u}_{opt}(t)$, is obtained. Then $\underline{u}_{cmd}(t)$ is obtained by modifying $\underline{u}_{opt}(t)$ with the repelling force from APF, as

$$\underline{u}_{cmd}(t) = \underline{u}_{opt}(t) + \sum_{k=1}^{n_o} F_k(t)\frac{r_k(t)}{\|r_k(t)\|}. \tag{19}$$

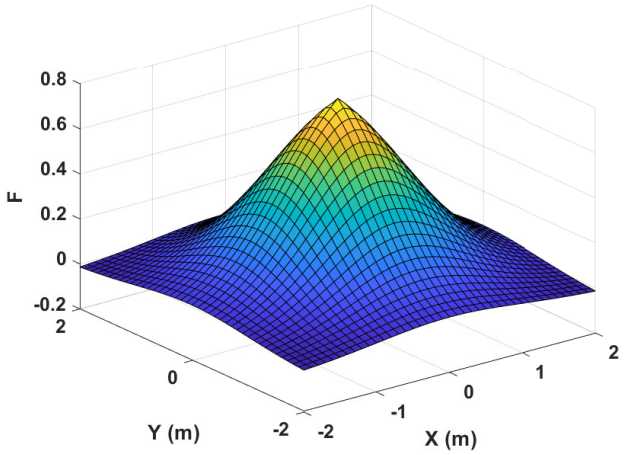

Fig. 8. Illustration of APF for $\alpha = 1.875$ and $\eta = 0.029$.

C. Parallelization

APF (18) can be implemented at a very high frequency due to its simplicity. However, solving the optimization problem (2) is numerically expensive. It should be noted that it is not always required to solve the optimization problem at a high rate, and the trajectory can be slightly modified by APF. Therefore, we propose to implement these two components to run simultaneously in parallel, with different frequencies. In fact, the higher frequency of APF always guarntees the

obstacle avoidance and, hence, the safety of the solution. The proposed parallelization is shown in a pseudocode in Table I.

At first, the optimization problem is solved once by defining the initial guess and boundary conditions based on the initial and final UAV positions and obstacle location. Then, parallel implementation of the two methods begins in a while loop. If no optimization code is running, the control inputs at initial time $(t = 0)$ are obtained from the solution and are treated as inputs to be commanded. The optimization solution along with the co-state multipliers are used to make a new initial guess. The initial boundary condition is updated based on the new UAV position and the optimization solver begins running in the background until a new solution is computed. The APF method is used to determine the necessary repulsive corrections to the initial control inputs, obtained from the optimization solution, and it is commanded to the UAV. The process is repeated until the UAV reaches its desired position.

TABLE I

PESUDOCODE FOR OPTIMIZATION AND APF PARALLELIZATION.

	Initialization
1.	$\underline{x}(0) \leftarrow \underline{x}_0$ Initial condition
2.	$\underline{x}(t_f) \leftarrow \underline{x}_f$ Final condition
3.	Solve an initial NLP
4.	$d(t) \leftarrow \|\underline{x}_f - \underline{x}(t)\|$
	Start parallelization
5.	**while** $d(t) > Threshold$, **do**
6.	**if** NLP solved
7.	$\underline{u}_{opt} \leftarrow \underline{u}(0)$
8.	$\underline{x}(0) \leftarrow \underline{x}_{uav}$
9.	$\underline{x}(t_f) \leftarrow \underline{x}_f$
10.	Solve NLP in background
11.	**end**
12.	Compute (19)
13.	Command $\underline{u}_{cmd}(t)$ to UAV dynamics (1)
14.	$d(t) \leftarrow \|\underline{x}_f - \underline{x}(t)\|$
15.	**end**

IV. EXPERIMENTAL RESULTS AND DISCUSSION

A. Experimental Setup

The setup of the UAV used for the experiments in this paper, onboard components, indoor positioning system, and communication network is illustrated in Figure 9. The UAV is a quadrotor vehicle, equipped with Pixhwak CUAV V5 Nano autopilot running the PX4 operating system. It also has a small onboard computer, Raspberry Pi Model B, running an Ubuntu server. The obstacle is a small DJI Tello drone. Positioning of the UAV and obstacle is carried out by the Optitrack system, which comprises 12 infrared cameras fixed to the ceiling at a height of around 5 m and a high-performance desktop computer running Motive software. The Optitrack system is calibrated to detect the circular reflective markers fixed on the UAV and obstacle asymmetrically and determines their position and orientation with respect to a predefined FLU inertial coordinate system at a frequency of 120 Hz. The UAV's onboard computer runs roscore and acts as an interface for communication between the Optitrack system, autopilot, and the ground control station (GCS). It receives the position and orientation from the Motive software over WiFi and relays it to mavros node. The mavros node further sends it to the autopilot, which internally fuses it with the IMU's high-frequency acceleration, and angular-rate measurements and receives the PX4-estimated position, and orientation in the inertial frame to make it available to GCS. The GCS initiates ROS in Matlab and has access to all the ROS topics, including obstacle's pose, UAV's pose, and IMU data published by mavros. The trajectory optimization code runs in Matlab to determine the control inputs. The computed control inputs are commanded to the UAV by publishing them to relevant topics in mavros, which relays them to the autopilot. The Tello drone is controlled by a separate computer interfaced with both Tello and Optitrack system simultaneously by Tello's WiFi and ethernet connections, respectively. A custom ROS node is developed to interface with the Tello driver and gain access to its control.

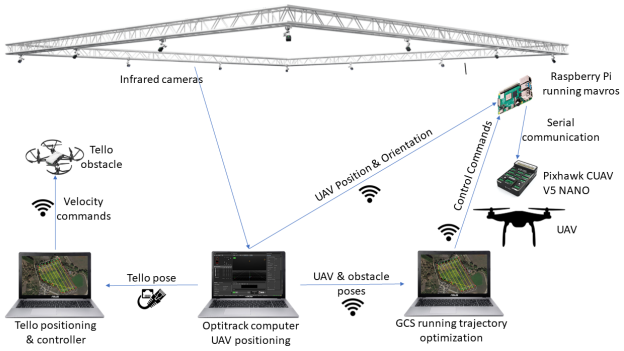

Fig. 9. Illustration of the UAV positioning and control network architecture in the lab.

B. Experimental Results

A single Tello drone is used as a dynamic obstacle and controlled to fly in a circular trajectory at an angular speed of 1.0 rad/s. The radius, height, and yaw angle were maintained to be constant at 0.5 m, 0.75 m, and zero deg, respectively. Two points $(-1.5, -1.5, 0.75)$ and $(1.5, 1.5, 0.75)$ in the FLU frame of reference are chosen as the boundary points. The values of α and η in (18) are chosen as 1.875 and 0.029, respectively.

The UAV is commanded to take off and reach the first boundary point. Once it reached, the trajectory optimization and APF method run in parallel iteratively. In the first iteration, an optimal trajectory is computed with boundary conditions and constraints defined appropriately. To prevent abrupt accelerations to UAV, the initial velocity is constrained not to have large values. The initial control input obtained from the optimization solution was corrected using APF method and commanded to the UAV. In the second iteration, a new optimization problem is created with the new UAV and obstacle positions. By using the warmstart feature of

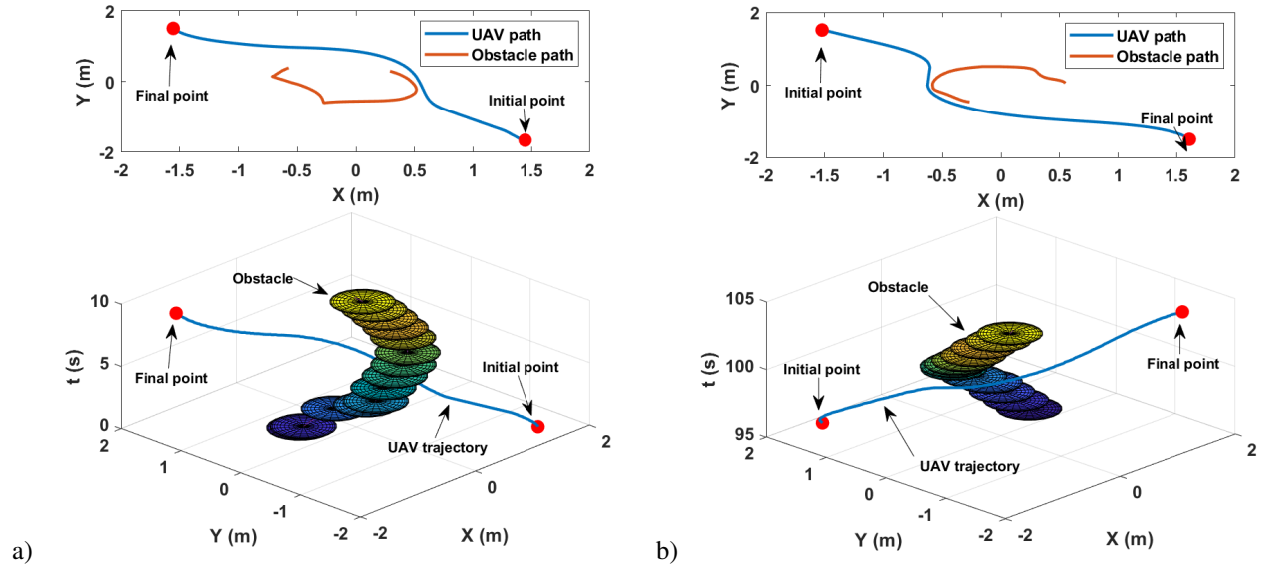

Fig. 10. Evolution of UAV trajectories with respect to moving obstacle: a) $t \in [0.3, 9.33]$ and b) $t \in [96.03, 103.86]$.

IPOPT, the optimization solution and corresponding co-state multipliers of constraints from the first iteration are specified as the initial guess. The optimization problem runs in the background using the parfeval function in Matlab. The initial control input from the first iteration is still used to make corrections using APF method and commanded. In the subsequent iterations, it was checked if the optimization code running in the background is solved or not. If so, the routine in the second iteration is repeated. Otherwise, only the APF method is implemented. Process is repeated until the UAV reached its final point.

To ensure that the proposed approach is robust and works for different obstacle positions and approaching scenarios, the UAV is continued to fly even after reaching the final point. However, the boundary points were switched. Based on the experimental analysis, the optimization solver is so robust in computing the solution. Every time the boundary points are switched, the solution from the previous iteration is an incorrect initial guess. However, the solver is able to compute a new solution without any noticeable delays.

The experiment was carried out for 150 seconds. Throughout the experiment, the UAV was able to fly back and forth between the two boundary points without having any collision with the moving obstacle. UAV trajectories and optimal trajectories computed from 135 s to the end were recorded in a video[1]. It should be noted that y and x axes in the figure correspond to F and $-L$ axes in the FLU frame of reference. This was done to have consistency between the UAV motion in the video and the plotting axes. During the experiment, the optimization solver was able to run at a frequency of around 10 Hz. The APF method was run at around 60 Hz. Despite the differences in the frequencies of their runtime, it appears that the trajectories are getting updated in real-time without any noticeable delays. The APF method can run at a much higher frequency, but most of the time was consumed reading messages from ROS topics and publishing the control commands.

The paths and trajectories of the UAV and the obstacle from 0.3 to 9.33 s and 96.03 to 103.86 s are presented in Figure 10. It can be seen that the UAV and obstacle trajectories (shown as circular disks) never intersected with each other. At the beginning of the maneuver in both cases, the obstacle was not on the line segment joining the two boundary points. Therefore, the computed optimal trajectory was a straight line. During the course of the maneuver, the obstacle started approaching the UAV and intersecting its path. The optimization solver was able to quickly compute new trajectories avoiding the obstacle. As soon as the obstacle started moving away from the UAV, the computed trajectories became almost straight again. The UAV continued reaching its final point along this straight line. Occasionally, slightly curved trajectories are computed due to the effects of the sub-optimal previous solution chosen as the initial guess. Overall, the proposed approach and the computational framework developed were shown to work effectively for real-time optimal UAV trajectory planning and dynamic collision avoidance.

V. Conclusions

In this paper, we presented an approach for real-time optimal UAV trajectory planning in the presence of dynamic obstacles, combining trajectory optimization with APF method. The optimization problem minimized the flying time between the initial and final positions. We transcribed the problem into a nonlinear programming problem using Chebyshev pseudospectral method. The state and control histories are approximated by using Lagrange polynomials

[1] https://youtu.be/swSspfvYjJs

and Chebyshev nodes. More importantly, we presented a novel sigmoid-type collision avoidance constraint. Automatic differentiation of cost function and constraints were used for fast convergence. The APF method was for guaranteeing collision avoidance. We also presented a parallel architecture running at moderate and high frequencies. Indoor experiments were conducted which confirmed the effectiveness of the proposed approach.

REFERENCES

[1] M. Castillo-Lopez, P. Ludivig, S. A. Sajadi-Alamdari, J. L. Sanchez-Lopez, M. A. Olivares-Mendez, and H. Voos, "A real-time approach for chance-constrained motion planning with dynamic obstacles," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3620–3625, 2020.

[2] J. L. Sanchez-Lopez, M. Castillo-Lopez, M. A. Olivares-Mendez, and H. Voos, "Trajectory tracking for aerial robots: an optimization-based planning and control approach," *J. Intell. Robot. Syst.*, vol. 100, no. 2, pp. 531–574, 2020.

[3] F. Gao, W. Wu, W. Gao, and S. Shen, "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments," *Journal of Field Robotics*, vol. 36, no. 4, pp. 710–733, 2019.

[4] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time uav replanning using guided gradient-based optimization and topological paths," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1208–1214, IEEE, 2020.

[5] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, "Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4101–4107, IEEE, 2021.

[6] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

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

[8] J. Chen, K. Su, and S. Shen, "Real-time safe trajectory generation for quadrotor flight in cluttered environments," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1678–1685, IEEE, 2015.

[9] 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.

[10] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 344–351, IEEE, 2018.

[11] W. Ding, W. Gao, K. Wang, and S. Shen, "An efficient b-spline-based kinodynamic replanning framework for quadrotors," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1287–1306, 2019.

[12] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "Faster: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 922–938, 2021.

[13] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[14] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 3681–3688, IEEE, 2017.

[15] 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)*, pp. 215–222, IEEE, 2017.

[16] 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.

[17] 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)*, pp. 5332–5339, IEEE, 2016.

[18] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526–1545, 2020.

[19] 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.

[20] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[21] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*, pp. 1–6, VDE, 2012.

[22] L. Jaillet and T. Siméon, "Path deformation roadmaps: Compact graphs with useful cycles for motion planning," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1175–1188, 2008.

[23] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, vol. 39, no. 1, pp. 101–121, 2015.

[24] D. H. Shim, H. J. Kim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, vol. 4, pp. 3621–3626, IEEE, 2003.

[25] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, "Real-time planning for automated multi-view drone cinematography," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–10, 2017.

[26] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha, "Chance constraint based multi agent navigation under uncertainty," in *Proceedings of the Advances in Robotics*, pp. 1–6, 2017.

[27] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams, "A probabilistic particle-control approximation of chance-constrained stochastic predictive control," *IEEE transactions on Robotics*, vol. 26, no. 3, pp. 502–517, 2010.

[28] F. Fahroo and I. M. Ross, "Direct trajectory optimization by a chebyshev pseudospectral method," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 160–166, 2002.

[29] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, 1999.

[30] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[31] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[32] J. Ren, K. A. McIsaac, R. V. Patel, and T. M. Peters, "A potential field model using generalized sigmoid functions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 477–484, 2007.