Exact Reactive Receding Horizon Motion Planning for Aerial Vehicles

Indrajeet Yadav and Herbert G. Tanner

Abstract— **This article presents a motion planning strategy that integrates a reactive receding horizon local trajectory generation methodology with a navigation function based global planner to guarantee that a quadrotor type MAV navigating in partially-known cluttered environment can reach its goal. The environment is partially known in the sense that there can be obstacles which are not a priori known to be included in the navigation function construction and used by the global planner. These obstacles are avoided reactively at the level of the local planner, which reacts to sensing information within the Field of View (FOV) of the onboard sensors and keeps generating dynamically feasible safe trajectories along the segments generated by the global planner. The proposed motion planning strategy combines the advantages of global and reactive aerial navigation, producing locally optimal (in a minimum-jerk sense) vehicle motion. The effectiveness of the planner is demonstrated through realistic ROS-Gazebo simulations.**

I. Introduction

Research advances over the last decade have enabled the use of quadrotor-type micro aerial vehicless (MAVs) in various application areas where accessibility of terrain or complexity of the environment pose challenges to the kind of device that can be successfully deployed. The agility, affordability and currently available onboard computation capability make modern quadrotors an attractive choice in applications ranging from surveillance, mapping, precision agriculture, environmental monitoring, and security. The industry standard at this time in terms of autonomous MAV navigation capabilities is the capacity to track a set of predefined trajectory waypoints without, however, any particular attention to the surrounding workspace. A priori information about the vehicle's workspace can be used to make a map and thus help the vehicle navigate in constrained environments. Environment topology information, however, is often incomplete. Recent relevant research literature focuses on building (or updating) a local environment map of this workspace using an onboard perception stack [7], [17].

A reactive local trajectory generation, integrated within a receding horizon scheme, has been shown to permit safe MAV navigation in unknown environments [24]. State of the art trajectory generation and control algorithms that can be used for this purpose are based on the idea of generating piece-wise polynomial in time trajectories [14], [18], which can then be faithfully followed using a differential-geometric trajectory tracking control design [10]. Still, there are several implementation issues which can compromise navigation mission success, and *ensuring* both convergence and safety requires addressing several open research questions [2].

Motion planners for MAVs, particularly those applicable to deployments in unknown environments, often compute using a grid-based representation of the environment [7], [16]. Such planners attempt to compensate for the incomplete environment information through onboard sensors like a 3D LiDAR, to generate a safe corridors through the environment and force the MAV to stay within that corridor [6]. One way to navigate to configurations outside the MAV sensor range, is to use a sequence of *predefined* waypoints. One such approach uses knowledge of obstacle location and geometry, and locally decomposes the available free space into convex polyhedra, amongst which a safe path is drawn [9]. Variants of such approaches [12] consider workspace representations in the form of a 3D grid map with uniform voxels, and create a convex safe flight corridor (SFC). Recent *reactive* motion planning algorithms include lightweight egospace-based techniques extended to capture the MAV configuration dynamics [4], and approaches that reactively sample safe trajectories in the FOV of the MAV, thus decoupling local obstacle avoidance from global guidance [20].

Concerns related to most grid or sampling-based motion planning algorithms is that no guarantees for convergence to the navigation goal can be provided, and that on-line map building does not scale well computationally as the size of the workspace increases. Receding horizon planning schemes circumvent the latter limitation by building only a local map. Still, however, it is not clear if convergence can be guaranteed at all using purely local and reactive planning. Some algorithms empirically demonstrate convergence [24], without, however, any formal guarantee.

Navigation functions [19] present an closed-loop motion planning and control strategy which offers formal and deterministic guarantees of convergence and safety. The methodology has been extended to cover cases of moving destinations [11] and obstacles [3]. Parallel notable extensions along a reactive navigation direction include methodologies for online construction and update of the navigation function based on sensory input [1], [22], including approaches utilizing deep neural networks for the identification of previously unknown obstacles. Such approaches have been successfully tested on fully actuated or differential drive robots at relatively low speeds; their applicability to and fast moving MAVs, which have underactuated and high order dynamics, remains to be demonstrated. There is some early and preliminary work on MAV motion planning and flight control using navigation functions [23] but safe autonomous flight in partially known environments remains an open.

Yadav and Tanner are with the Department of Mechanical Engineering at the University of Delaware {indragt,btanner}@udel.edu

This work has been supported in part by DTRA under grant #HDTRA1-16-1-0039, and in part by ARL under #W911NF-20-2-0098.

This paper attempts to close this gap. While receding horizon based local planners [24] do not furnish any provable guarantee of the convergence or the completeness (or even exactness) and are prone to get trapped in local minima due to lack of knowledge of the environment, a navigation function-based reactive planning [23] can compensate despite not being designed for higher dimensional systems such as those of quadrotors. Under the assumption that some key, high-level, features of the environment topology which can typically allow a local planner to fail are known, the paper, as its key contribution, presents a cascaded MAV motion planning approach in which a navigation function-based global planner repeatedly generates a high level motion plan, which is then executed at the low-level by a model predictive control (MPC)-type local planner that fully incorporates the nonlinear vehicle dynamics [24]. The local planner utilizes a 3D point-cloud generated by an onboard RGB-D camera to select a (probabilistically optimal) safe path within the field of view and then fits a minimum jerk trajectory to it. The key advantage of this approach is that it combines the best of reactive and deliberate motion planning; the navigation function-based global planner closes the planning loop and guarantees convergence as long as unknown obstacles represent local and isolated environmental features, while the local planner fully incorporates the nonlinear dynamics and reactively accounts for local collision avoidance. The algorithm have been tested in realistic ROS-GAZEBO simulations using RotorS package [5].

II. Overview of the Approach

Figure 1 illustrates the architecture of the motion planning and control system. The arrows indicate the direction of information flow. Point-cloud data from the RGB-D sensor are utilized to frame the obstacle-free portion of the workspace and encode it as a set of rays cast from the RGB-D sensor's focal point to the edge of its FOV.

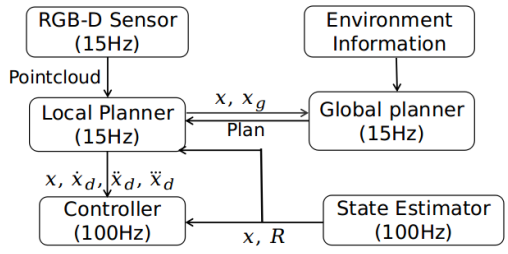


Fig. 1: Block diagram of the motion planning and control architecture.

The computation steps involved in the reported cascaded planning approach are (i) A priori information about the MAV's workspace is used to construct a navigation function; (ii) the local planner operating at 15Hz takes the pointcloud information from RGB-D sensor and shares the MAV's current location and desired global goal with the global planner; (iii) the global planner receives this information and uses the gradient of the navigation function to construct a *segment* of a feasible path connecting current MAV configuration to the global goal (assuming no unknown obstacles are on that path); (iv) the local planner receives the path segment and generates a dynamically feasible minimum-jerk trajectory that fits on it tightly; and finally, (v) a small *initial component* of that minimum-jerk reference trajectory is faithfully tracked by a provably stable differential-geometric tracking controller operating at 100Hz, which starts steering the quadrotor along its way to the path segment in the FOV. Once this process is completed, the cycle closes and repeats from step (ii).

In a typical receding horizon fashion, before the end of that initial segment is reached, the global planner uses updated quadrotor position to construct a new path segment and the local planner collects the updated point-cloud information to complete the generation of new trajectory, and transitions smoothly between reference trajectory segments. The re-planning and trajectory tracking process repeats until the final static destination is reached. On board state estimation runs independently of the planners and supplies the position information to the MAV at 100Hz.

III. Quadrotor Dynamics and Control

The quadrotor is modeled as a rigid body evolving in $\mathsf{SE}(3)$. Let m and $\mathbf{J}$ denote its mass and moment of inertia, respectively, and denote $\mathsf{x} = (x, y, z)^\intercal$ and $\mathsf{v} = \dot{\mathsf{x}}$ its 3d position and velocity in the inertial frame. Let $\mathbf{R}$ be the rotation matrix from the body-fixed frame to the inertial frame, and Ω be the MAV's angular velocity in the body-fixed frame. We denote $\hat{\cdot}$ the skew symmetry operation, and write the gravitational acceleration vector and the unit vector along the inertial vertical as $\mathsf{g} = (0, 0, g)^\intercal$, and $\mathsf{e}_3 = (0, 0, 1)^\intercal$, respectively. The control inputs to the quadrotor are the (scalar) thrust f and the (vector) moment M, both of which are bounded (cf. [15]). the constraint on the latter, can be shown to translate to a bound on angular velocity:

$$f_{\min} \leq f \leq f_{\max} \tag{1a}$$
$$\|\Omega\| \leq \Omega_{\max} \tag{1b}$$

With these in place, the dynamics of the MAV are

$$\dot{\mathsf{x}} = \mathsf{v}, \qquad m\,\dot{\mathsf{v}} = f\,\mathbf{R}\,\mathsf{e}_3 - m\,\mathsf{g} \tag{2a}$$
$$\dot{\mathbf{R}} = \mathbf{R}\,\hat{\Omega}, \qquad \mathbf{J}\,\dot{\Omega} + \Omega \times \mathbf{J}\,\Omega = \mathsf{M} \tag{2b}$$

The control law for this system is designed based on a (differential) geometric method [10] with minor modifications. To examine the control strategy in more detail, consider a smooth position reference trajectory $\mathsf{x}_d(t) \in \mathbb{R}^3$. From here one can construct [10] the desired rotation matrix $\mathbf{R_d}$ and angular velocity Ω_d; here, however, we select for the yaw angle ψ the direction $\mathsf{b}_{1d} = (\cos\psi, \sin\psi, 0)^\intercal$, perpendicular to the stabilizing direction. The tracking errors in position e_x, velocity e_v, orientation e_R, and angular rate e_Ω are

$$\mathsf{e}_\mathsf{x} = \mathsf{x} - \mathsf{x}_d \qquad \mathsf{e}_\mathsf{v} = \dot{\mathsf{x}} - \dot{\mathsf{x}}_d$$
$$\mathsf{e}_R = \frac{1}{2}(\mathbf{R}_d^\intercal\mathbf{R} - \mathbf{R}^\intercal\mathbf{R}_d) \qquad \mathsf{e}_\Omega = \Omega - \mathbf{R}^\intercal\mathbf{R}_d\Omega_d$$

Picking positive control gains k_x, k_v, k_R and k_Ω, the control inputs can be constructed as

$$f = -k_x\mathsf{e}_\mathsf{x} - k_v\mathsf{e}_\mathsf{v} + m\mathsf{g} + m\ddot{\mathsf{x}}_\mathsf{d},$$
$$M = k_R\mathsf{e}_\mathsf{R} + k_\Omega\mathsf{e}_\Omega + \Omega \times \mathbf{J}\Omega$$

allowing one to achieve exponentially stable tracking behavior for initial attitude error less than $\frac{\pi}{2}$, and almost global exponential stability otherwise [10].

IV. The Global Planner

A navigation function [19] is a real-valued map $\mathcal{V} : \mathcal{F} \to \mathbb{R}$, constructed on the MAV's free configuration space $\mathcal{F}$ that when tuned appropriately has a unique minimum at the desired goal configuration and is uniformly maximal over the boundary of $\mathcal{F}$. The global planner used in this paper utilizes a navigation function where known obstacles are represented as rounded parallelepipeds (aka squircles) [11]. Since in the context of this method, the MAV is to be represented by a point $\mathbf{x}$ in $\mathcal{F}$, the volume of the MAV is taken into account in planning by inflating the known obstacles by an appropriate safety margin.

Let $\mathbf{x}(t)$ denote the position of the MAV in the map's inertial frame and $\mathbf{x}_g(t)$ that of the (possibly moving) target at time t, both assumed in the interior of the free space $\mathcal{F}$. For the purposes of constructing the navigation function, $\mathbf{x}(t)$ and $\mathbf{x}_g(t)$ are assumed known; the MAV configuration $\mathbf{x}(t)$ is made available through an onboard state estimation algorithm, while the target configuration $\mathbf{x}_g(t)$ is assumed at this time to be supplied explicitly.

In this variant of navigation functions [11] the vehicle is not supposed to collide or overlap with its target, but rather to track it from a selected desired distance r. In this light, we take r to be the radius of the spherical bubble around the target $\mathbf{x}_g$ and define the goal *region* for navigation as the area minimizing the time varying goal function

$$J\big(\mathbf{x}, \mathbf{x}_g\big) = \|\mathbf{x} - \mathbf{x}_g\|^2 - r^2 \tag{3}$$

It has been shown [11], [21] that for a suitably selected obstacle function $\beta(\mathbf{x})$ and a suitably large parameter $\lambda \in \mathbb{R}_+$, there exists a positive number N such that $\forall \kappa \geq N$,

$$\tilde{\varphi}(\,\mathbf{x}, \mathbf{x}_g) = \frac{J(\mathbf{x}, \mathbf{x}_g)}{\big[J(\mathbf{x}, \mathbf{x}_g)^\kappa + \lambda\, \beta(\mathbf{x})\big]^{1/\kappa}} \tag{4}$$

has navigation function properties—with the only difference being that destination configurations are degenerate—in sphere worlds [21] as well as sphere worlds [11]. The parameter $\lambda \neq 1$ has been utilized [21] to facilitate tuning.

Assume now that the (known) obstacle collection in the configuration space of the MAV is a forest of squircles. Then a diffeomorphism $h_{\lambda_{\text{sq}}} : \mathcal{F} \to \mathcal{S}$ parameterized by a suitably chosen positive parameter $\lambda_{\text{sq}} \in \mathbb{R}_+$, allows the composition $\varphi = \tilde{\varphi} \circ h_{\lambda_{\text{sq}}}(\mathbf{x}, \mathbf{x}_g)$ to become a navigation function on $\mathcal{F}$ [11]: for any position of the target satisfying some reasonable topological conditions, all (unstable) critical points outside the destination manifold are nondegenerate saddles with attraction regions of measure zero.

A properly tuned navigation function ensures that a gradient descent using its negated gradient converges to the goal from almost everywhere in the free workspace—less the attraction regions of the saddles, which are sets of measure zero. This provides the basis for a robust navigation feedback strategy that can reasonably account for estimation and model inaccuracies. Aggressive maneuvering at user-defined maximum speeds v_{max} is achieved by prescribing a desired MAV velocity vector field as

$$\mathbf{v}_d(\mathbf{x}) \triangleq -\frac{\nabla_{\mathbf{x}}\varphi}{\|\nabla_{\mathbf{x}}\varphi\|} \cdot v_{\mathsf{max}} \tag{5}$$

It should be noted at this point that (5) does not take into account the (high-dimensional and nonlinear) dynamics of the MAV. Nonetheless, through forward integration (5) can produce safe reference trajectories which the MAV can closely track using appropriate nonlinear control loops. The closure of such control loops, in a way that actually guards against collisions with unknown obstacles, is described in the next section.

V. The Local Planner

The local planner utilizes a receding horizon planning strategy in which the quadrotor continuously takes input from the onboard RGB-D sensor to generate feasible, safe, and locally optimal trajectories. In this way, the MAV reactively avoids previously unknown obstacles within its FOV while making progress toward the navigation goal specified by the global planner [24]. This section presents a brief description of the approach followed.

The MAV dynamics are differentially flat [14], at least to some extent, and that allows one to express system inputs and states as a function of the four (flat) outputs associated with the vehicle's 3D position and yaw. We can bundle these flat outputs into a vector $\mathsf{x} = [x, y, z, \psi]^\intercal$, and noting that the yaw angle ψ is kinematically decoupled from x and y, we may restrict our analysis on constructing smooth reference trajectories in x and y, knowing that ψ can be set independently afterwards. Local planner trajectories which satisfy (2) and (1) will be referred to as *dynamically feasible*. The local planner iterates cycles of computation, updating each time the MAV's reference trajectory which attempts to track the navigation function's currently generated path segment. The end point of this path segment is referred to as the *intermediate point*.

Suppose that at time t the planner is at cycle N, generating a reference trajectory segment of duration δt, and the following cycle $N+1$ produces another segment ranging from time instant $t+\delta t$ to $t+2\delta t$. Such reference trajectory segments are represented as extended vectors of desired position, velocity, acceleration and jerk; for cycles N and $N+1$ will look like:

$$X_N(t, t+\delta t) = \begin{bmatrix}\mathsf{x}_N^\intercal & \dot{\mathsf{x}}_N^\intercal & \ddot{\mathsf{x}}_N^\intercal & \dddot{\mathsf{x}}_N^\intercal\end{bmatrix}^\intercal$$

$$X_{N+1}(t+\delta t, t+2\delta t) = \begin{bmatrix}\mathsf{x}_{N+1}^\intercal & \dot{\mathsf{x}}_{N+1}^\intercal & \ddot{\mathsf{x}}_{N+1}^\intercal & \dddot{\mathsf{x}}_{N+1}^\intercal\end{bmatrix}^\intercal$$

The objective of the planner is to produce dynamically feasible reference trajectories that stay in $\mathcal{F}$ and are compatible in the sense that for any two consecutive cycles N and $N+1$ it holds that $X_N(t+\delta t) = X_{N+1}(t+\delta t)$.

We assume that the FOV of the RGB-D sensor is contained in a rectangular pyramid, having its apex at the base frame attached to the sensor, its depth direction is aligned with

the local x MAV axis. The height of this FOV pyramid if determined by the sensor's range. We discretize this volume to reduce it to a grid of fixed resolution. A ray is then cast from the apex to each point on the grid. Any ray, including its end point, that passes sufficiently close to a point within the sensor's RGB-D point cloud is discarded, on the basis that this line of sight is likely obscured by some (unknown) obstacle. These type of discrete query and selection operations can be implemented efficiently using KD-tree data structures [24].

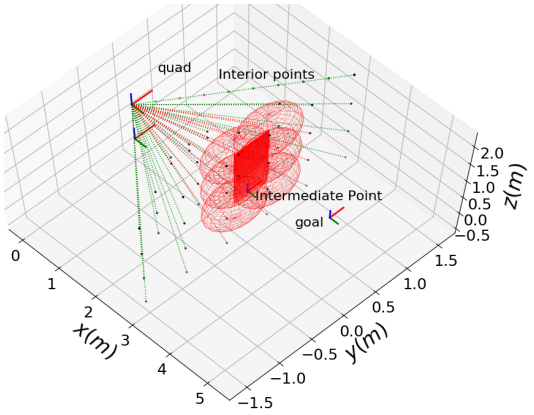


Fig. 2: Checking for collisions. Triads mark the MAV's initial configuration, its current center of gravity (COG), its local goal in its FOV, and its global goal. Black dots mark representative discretization points in the MAV's FOV. Rays in the FOV intersecting with obstacles reflected in point cloud data (red) are marked red and considered discarded. Green rays represent collision-free motion directions.

An optimal among all possible collision-free rays should strike an acceptable balance between safety and aggression while converging to the global goal. The latter is assessed based on how close to the navigation function-generated path segment. To strike such a balance we weigh each of the collision-free rays based on a linear combination of two cost factors: (a) a collision cost factor that penalizes a ray depending on how close it brings the MAV to any of the obstacles in the FOV, and (b) a convergence cost factor that penalizes a ray based on the distance between its end point and the intermediate point. (Given the receding horizon nature of the control scheme, we found that this singular end-point comparison was adequate for implementation purposes.)

Denote p the total number of collision-free rays, and d_i the Euclidean distance between the end point of the i^{th} ray and the intermediate point. Set $d_{\max} \triangleq \max_i d_i$, and let $\hat{r} \geq r$ be an additional safety margin (added to the radius around detected obstacles). Letting ρ_i be the minimum distance to the nearest obstacle out of all interior points on ray i, the collision cost for ray i is measured as

$$c_{\text{coll}_i} = \begin{cases} \frac{1+\hat{r}^4}{\hat{r}^4} \cdot \frac{[(\rho_i - r)^2 - \hat{r}^2]^2}{1+[(\rho_i - r)^2 - \hat{r}^2]^2} & \text{if } \rho_i - r \leq \hat{r} \\ 0 & \text{otherwise} \end{cases}$$

Then for the positive scalar weights $k_1 \in (0,1) \ni k_2$, the total cost associated with ray $i \in [0, \ldots, p]$ is (see Fig. 3)

$$c_i = k_1 \frac{d_i}{d_{\max}} + k_2\, c_{\text{coll}_i}$$

The end point of the ray that has the lowest total cost is named the *local goal* (because it is within the FOV) of the local planner. With the local goal in the FOV selected, the problem now reduces to generating a dynamically feasible and locally optimal reference trajectory from the MAV current 3D position to the local goal, tracing the optimal ray. The reference trajectory thus involves the first three flat outputs of the dynamics, namely the Cartesian position coordinate vector $\mathsf{x} = [x, y, z]^\intercal$.

The duration of the segment of the reference trajectory that runs from the MAV's location segment from the starting state (position, velocity, acceleration, jerk) $\mathsf{x}_0 = \left[\mathsf{x}_0^\intercal, \dot{\mathsf{x}}_0^\intercal, \ddot{\mathsf{x}}_0^\intercal, \dddot{\mathsf{x}}_0^\intercal\right]^\intercal$ to the local goal state $\mathsf{x}_\intercal = \left[\mathsf{x}_T^\intercal, \dot{\mathsf{x}}_T^\intercal, \ddot{\mathsf{x}}_T^\intercal, \dddot{\mathsf{x}}_T^\intercal\right]^\intercal$ is set to a predetermined *planning horizon* denoted T. The optimal ray is now divided into n_p equal segments, the endpoints of which now define way points for the reference trajectory to be generated. Let Δt_j denoting the time difference between two successive waypoints on the reference trajectory, $(x_{j-1}, y_{j-1}, z_{j-1})^\intercal$ and $(x_j, y_j, z_j)^\intercal$; naturally, $T = \sum_{i=1}^{n_p} \Delta t_i$. The reference trajectory for planning cycle N can now be defined as a solution of the following minimum jerk optimization problem over the planning horizon [14], [18],

$$\begin{cases} \arg\min_{\mathsf{x}_i} \sum_{i=0}^{n_p} \int_0^{\Delta t_i} \left\| \frac{d^3 \mathsf{x}_i}{dt^3} \right\|^2 \mathrm{d}t \\ \text{subject to} \\ \frac{\mathrm{d}^k \mathsf{x}_i}{\mathrm{d}t^k}\,|_{\Delta t_i} = \frac{\mathrm{d}^k \mathsf{x}_{i+1}}{\mathrm{d}t^k}\,|_0 \quad k = 0, \ldots, 3 \\ \mathsf{x}(0) = \mathsf{x}_0, \;\; \mathsf{x}(T) = \mathsf{x}_\intercal \end{cases} \tag{6}$$

Problem (6) is converted to a quadratic program (QP) and efficiently solved using standard solvers (see [8], [18]). Once a reference trajectory is generated, it is checked to ensure that the nominal thrust and angular velocity conforms to the dynamic constraints (1). In case they do not, the trajectory completion time is dilated thus reducing the average speed of the MAV. Once the output reference trajectory is finalized, an initial portion of it of time duration $T_c \ll T$ (called the *control horizon*), is then tracked by the MAV utilizing a variation [24] of a differential-geometric motion controller (cf. [10]). An illustrative example of this receding horizon planning method for the relatively simple test scenario of Fig. 2, is presented in Fig. 3.

Notice that the reference trajectory generated as a solution of the minimum-jerk optimization problem involves only the 3D Cartesian coordinates of the MAV and their derivatives. The fourth flat output ψ, the MAV yaw angle, is kinematically decoupled and can be presecribed independently. To construct a smooth reference trajectory for the yaw angle, we fit a third order time polynomial between initial yaw state $\psi(0) = \psi_0$, $\dot{\psi}(0) = \dot{\psi}_0$ and final yaw state (at the local goal) $\psi(T) = \psi_T$, and $\dot{\psi}(T) = \dot{\psi}_T$, of the form $\psi(t) = a_1 + a_2 t + a_3 t^2 + a_4 t^3$. When realizing this, and for any planning cycle, we kept the yaw rate $\dot{\psi}(T)$ at zero, and set ψ_T so that the MAV's onboard camera always faced the local goal at time T. On the other hand, ψ_0 and $\dot{\psi}(0)$ are set by the preceding planning cycle.

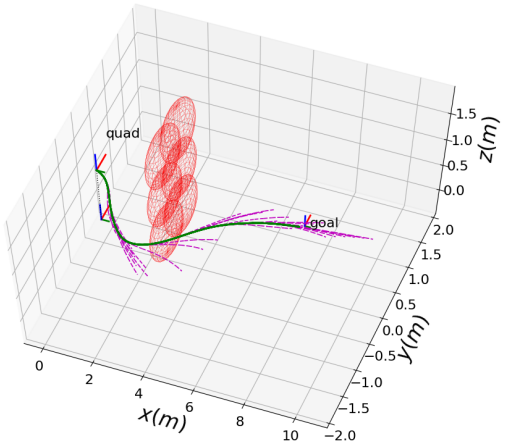


Fig. 3: Trajectory Generation. Dashed magenta lines show generated reference trajectories while solid green curve is the trajectory of the MAV as it tracked the trajectory segments consecutively.

VI. Results

This section outlines the steps toward the integration of the two planners through a 2.5D example. Consider a workspace with one nonconvex Π-shaped obstacle, designed and positioned in a way that could potentially trap the MAV (Fig. 4(a)). The workspace has a 15m × 15m horizontal footprint, and in it, the MAV is supposed to reach $(x, y, z) = (12, 0, 0)$ m, starting from $(-14, 1, 0)$ m. If no prior information is utilized about the Π-shaped obstacle, a *purely reactively* navigating MAV would encounter the interior wall and most likely be trapped inside. A navigation function incorporating this obstacle, however, can inform the MAV to safely maneuver around it (Fig. 4(b)).

The global planner utilizes the navigation function in a receding horizon manner: it regenerates a path toward the global goal and updates the navigation plan from whatever configuration the MAV finds itself in the beginning of each replanning cycle. This iterative process is reflected in the path of Fig. 4(b), in the different colors used to mark the different segments of the path to the goal: each segment is generated as part of a different planning cycle. The end point of each such segment defines the intermediate point for the local planner (Section IV). The local planner then iteratively generates dynamically feasible jerk-minimizing reference trajectories, reactively avoiding previously unknown obstacles detected in the MAV's FOV. The differential-geometric controller subsequently tracks these collision-free reference trajectories to steer the MAV toward the global goal.

While true that during trajectory tracking the MAV can deviate slightly from the navigation function path, safety is never compromised because of the reactive nature of the local planner. Even if the local planner steers the MAV away from the nominal path as it attempts to reactively avoid obstacles, the global planner regenerates a feasible path to the global goal at 15Hz (only a small selection of these segments are shown in Fig. 4(b) for clarity). As long as the unknown obstacles are isolated, local (much smaller in scale compared to the workspace topology features encoded in the navigation function), and disjoint from the navigation function's domain boundaries, the global planner is guaranteed to produce a new global navigation solution (cf [13]).

Figures 5 and 6 illustrate how the local and global planners work in tandem to account for unforeseen obstacle encounters. In Figure-5, four obstacles were placed along the path that the MAV would normally follow from its initial configuration to the final goal. The MAV initially follows the path originally generated by the global planner faithfully (black path in Fig. 6), until the unknown obstacles emerge in its FOV. At this stage, the local planner reactively generates avoidance maneuvers, which the global planner adapts to by reconstructing a high-level navigation plan to the final goal (purple path Fig. 6). GAZEBO and RVIZ simulation can be found at https://youtu.be/MCIX7E7qX28.

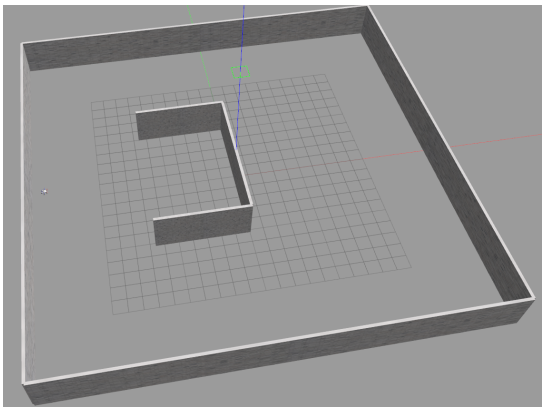
The MAV successfully avoids long wall-like obstacles (e.g. the Π-shaped obstacle in the current setup) if they are known and captured by the navigation function. If not included, such obstacles may obstruct the entire FOV of the MAV and trigger an emergency stop policy that brings the MAV to a halt (see [24] for a more detailed discussion).

Parameters controlling the density of grid points as well as incoming point cloud can be adjusted depending upon the computational resources available for real-time local planning. Our experience points to these parameters as the main contributors to computation overhead. Additional mitigation measures include the frequency of the global planner, which is the same frequency as that of the local planner by default (15Hz), but can be slightly reduced depending upon the user constraints. A detailed discussion on the computational efficiency of the local planner [24] is beyond the scope of this particular paper.

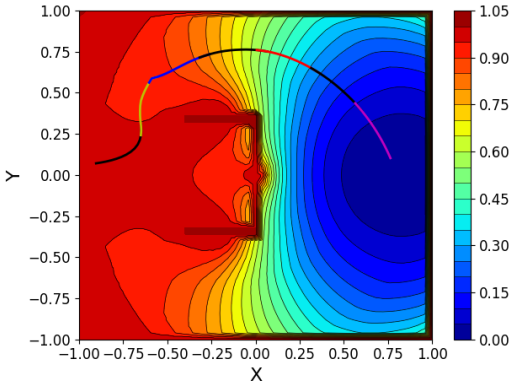
Among other issues that affect the performance of the planner is the inclusion of key environmental features (e.g. obstacles that can trap the MAV) in the global planner, and the proper tuning of the navigation function so that it is free of local minima. The latter is typically achieved with higher values for κ that lead to more direct paths to the goal and a relatively more aggressive maneuver around obstacles. In any case, the motion plans generated cannot come with any guarantees of optimality, especially since the total length of the trajectory depends on a number of unknown environmental features.

VII. Conclusions

As parallel and independent work in literature has started to demonstrate, deliberate and reactive robot navigation can be integrated in ways that combine the best aspects of both approaches. This paper reports on a new integrated end-to-end planning and control approach, in the spirit of receding horizon control, which is tailored to MAV navigation in partially known static environments. It is thus possible to steer an MAV in such environments with guarantees on safety against collisions *and* on convergence to the navigation goal, *despite* uncertainty about the existence of scattered isolated obstacles. What is particularly notable in the reported approach is that it is computationally light for autonomous



(a) MAV workspace in Gazebo



(b) Navigation Function and path from start to goal

Fig. 4: The MAV's known workspace and the global planner's path. The path from the start point to the global goal (in blue region) is generated by the navigation function's gradient. The multiple colored segments indicate the portions produced at different cycles of the global planner.

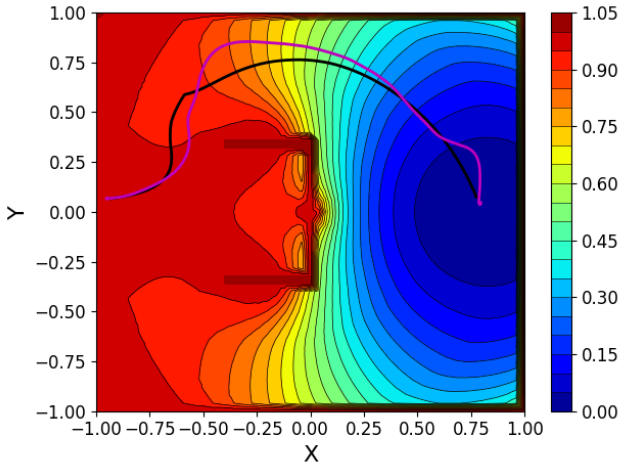


Fig. 6: Quadrotor Trajectories. Black curve shows the trajectory generated by global planner when none of the additional obstacles (pillars and tree) are known to it. Magenta curve shows the actual trajectory followed by the quadrotor.

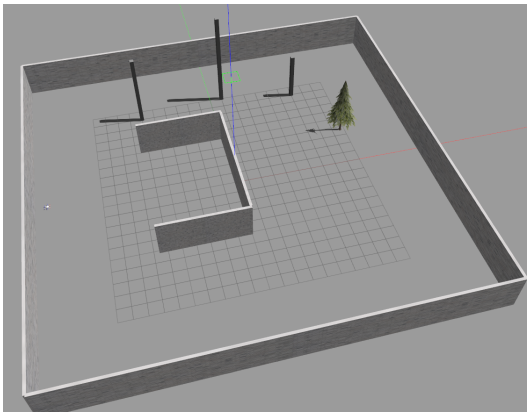


Fig. 5: Workspace with 3 pillars and one tree previously unknown to the global planner. These obstacles are avoided reactively by the local planner as they become visible in the FOV of the camera.

onboard calculations, and relies exclusively on inexpensive and lightweight sensor payloads, allowing for MAV navigation and flight at relatively high speeds. It should also be noted that although the approach is demonstrated here in 2.5D environments, it can be fully implemented in three dimensions in a straightforward way.

REFERENCES

[1] Omur Arslan and Daniel E Koditschek. Sensor-based reactive navigation in unknown convex sphere worlds. *The International Journal of Robotics Research*, 38(2-3):196–223, 2019.

[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, Dec 2016.

[3] Chuchu Chen, Caili Li, , and Herbert G. Tanner. Navigation functions with non-point destinations and moving obstacles. In *Proceedings of the American Control Conference*, pages 2532–2537, 2020.

[4] Anthony T. Fragoso, Cevahir Cigla, Roland Brockers, and Larry H. Matthies. Dynamically feasible motion planning for micro air vehicles using an egocylinder. In Marco Hutter and Roland Siegwart, editors, *Field and Service Robotics*, pages 433–447, Cham, 2018. Springer International Publishing.

[5] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016.

[6] F. Gao and S. Shen. Online quadrotor trajectory generation and autonomous navigation on point clouds. In *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 139–146, Oct 2016.

[7] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, Apr 2013.

[8] Gurobi Optimization Inc. Gurobi Optimizer Reference Manual, 2014. http://www.gurobi.com.

[9] B. Landry, R. Deits, P. R. Florence, and R. Tedrake. Aggressive quadrotor flight through cluttered environments using mixed integer programming. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1469–1475, May 2016.

[10] T. Lee, M. Leoky, and N. H. McClamroch. Geometric tracking control of a quadrotor uav on se(3). In *Proceedings of 49th IEEE Conference on Decision and Control*, pages 5420–5425, 2010.

[11] C. Li and H. G. Tanner. Navigation functions with time-varying destination manifolds in star worlds. *IEEE Transactions on Robotics*, 35(1):35–48, 2019.

[12] 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*, 2(3):1688–1695, July 2017.

[13] Savvas G. Loizou, Herbert G. Tanner, Vijay Kumar, and Kostas Kyriakopoulos. Closed loop motion planning and control for mobile robots in uncertain environment. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2926–2931, 2003.

[14] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2520–2525, May 2011.

[15] M. W. Mueller, M. Hehn, and R. D'Andrea. A computationally efficient motion primitive for quadrocopter trajectory generation. *IEEE Transactions on Robotics*, 31(6):1294–1310, Dec 2015.

[16] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran. Continuous-time trajectory optimization for online UAV replanning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5332–5339, Oct 2016.

[17] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. Voxblox: Incremental 3D euclidean signed distance fields for on-board MAV planning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1366–1373, Sep. 2017.

[18] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In Masayuki Inaba and Peter Corke, editors, *Robotics Research*, volume 114 of *Springer Tracts in Advanced Robotics*, pages 649–666. Springer International Publishing, Cham, 2016.

[19] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, Oct 1992.

[20] M. Ryll, J. Ware, J. Carter, and N. Roy. Efficient trajectory planning for high speed flight in unknown environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 732–738, May 2019.

[21] Jianxin Sun and Herbert G. Tanner. Constrained decision-making for low-count radiation detection by mobile sensors. *Autonomous Robots*, 39(4):519–536, 2015.

[22] V. Vasilopoulos, G. Pavlakos, S. L. Bowman, J. D. Caporale, K. Daniilidis, G. J. Pappas, and D. E. Koditschek. Reactive semantic planning in unexplored semantic environments using deep perceptual feedback. *IEEE Robotics and Automation Letters*, 5(3):4455–4462, 2020.

[23] I. Yadav and H. G. Tanner. Mobile radiation source interception by aerial robot swarms. In *International Symposium on Multi-Robot and Multi-Agent Systems*, pages 63–69, Aug 2019.

[24] I. Yadav and H. G. Tanner. Reactive receding horizon planning and control for quadrotors with limited on-board sensing. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7058–7063, 2020.