2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Reactive Receding Horizon Planning and Control for Quadrotors with Limited On-Board Sensing

Indrajeet Yadav and Herbert G. Tanner

***Abstract*— The paper presents a receding horizon planning strategy for a quadrotor-type MAV to navigate through an unknown cluttered environment at high speed. Utilizing a lightweight on-board short-range sensor that generates point-clouds within a narrow Field of View (FOV), the reported approach generates safe and dynamically feasible trajectories within the FOV of the sensor, which the MAV uses to navigate without relying on any global planner or prior information about the environment. The effectiveness of this planner-controller combination is demonstrated in both indoor and outdoor tests featuring speeds of up to of 3.5 m/s. With minor adjustments, the local motion planner can be utilized for interception and tracking of a moving target; evidence to this effect are provided in the form of numerical (Gazebo) simulations. Given the absence of any global information about the robot's workspace, the extent to which the local planner can provide convergence guarantees is limited; when complemented by a global planner and/or target tracker, the reported lower-level, sensor-driven reactive motion control strategy completes the autonomous MAV navigation stack, enabling navigation in dynamic, uncertain, and partially-known environments with guaranteed convergence to any static or dynamic target.**

I. Introduction

The agility and flexibility in terms of size and payload capacity makes quadrotor-type micro aerial vehicles (MAV) attractive platforms in many application areas such as surveillance, aerial photography and mapping, precision agriculture, construction and defense. Although MAV of various degrees of autonomy have been deployed in these areas, the prevailing assumption has been that the environment is at least partially known, so that a motion plan can be generated a priori and then used for steering the vehicle to a desired goal. For unknown environments, recent MAV literature addresses the problem of building (or updating) the local map of the environment using an onboard perception stack [6], [17].

The ability to safely navigate through a set of waypoint poses (keyframes) using a reference trajectory composed piece-wise of polynomials in time [13], [18], together with the development of differential-geometric quadrotor controllers [11], have enabled safe navigation and aggressive maneuvering as long as constraints on vehicle dynamics are respected. While platform-specific constraints on vehicle dynamics can be identified by experimental testing, *ensuring* safety during operation requires either complete prior knowledge of the environment, or some way of completing the missing information through acquisition by means of onboard sensing; both options involve several nontrivial and open research questions [1]. Early work on online obstacle avoidance focused on building a new, or updating a prior environment map. A continuous time trajectory optimization using octomap [6] has been utilized [16], in which one uses a local planner to re-generate a safe trajectory taking collision costs from a computationally expensive Euclidean signed distance field (ESDF) map.

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

This work has been supported by DTRA under grant #HDTRA1-16-1-0039. Special thanks to Kleio Baxevani for her help in experimental studies.

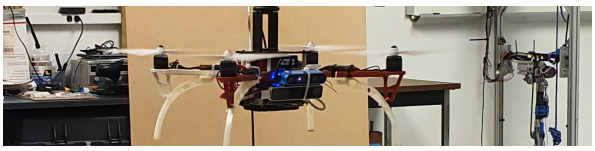


Fig. 1: The custom-built quadrotor MAV utilized for experimental studies, featuring on-board control, estimation and motion planning capabilities.

In unknown environments, a navigation approach utilizing online planning [4] constructs a point-cloud map of the environment using a Velodyne 3D LiDAR to find safe (spherical) corridors through which the MAV plans it motion. To navigate to destinations outside the sensor range, a sequence of predefined waypoints is needed. Another approach uses knowledge of obstacle location and geometry, to locally decompose the available free space into convex polyhedra and generate the safe path for the vehicle [10]. Variants of such approaches [12] consider the workspace represented as 3D grid map with uniform voxels, through which they create a convex safe flight corridor (SFC).

All aforementioned approaches either require some type of prior information and a global planner that generates a sequence of waypoints, and/or rely on (payload-taxing) high-range sensors. These methods are particularly effective for static destinations; if, however, that goal location is time-varying or if the generation of a prior map is not possible, one cannot a priori guarantee the existence of feasible and safe path through (predefined) waypoints or ensure that the local destinations will always be within sensor range.

Recent *Reactive* motion planning algorithms include lightweight Egospace-based algorithms extended to a quadrotor's configuration dynamics [2] or reactively sampling safe trajectories in the field of view of the quadrotor, decoupling local obstacle avoidance and global guidance using a global planner [19]. An impressive *receding horizon*-based approach to (local) planning involves only limited onboard sensing and utilizes a local uniform resolution volumetric occupancy grid map and a cost map to find and navigate to safe frontier points (local goals that are closest to the global goal) [20]. A more recent approach utilizes a CNN to generate safe paths in the environment [9]. Comparative studies and

details can be found in Section IV.

This paper contributes to the literature on *real-time obstacle avoidance and navigation* with a reactive motion planning and control methodology that is applicable to cases involving both static as well as moving destinations. The method does not rely on grid-like environment representations, which are known to scale poorly with workspace size and resolution. This new methodology takes the form of a model predictive control (MPC)-type motion planner that fully incorporates the nonlinear vehicle dynamics (cf. [22]). With this planner, the vehicle—in this case, an MAV—utilizes the point-cloud generated by an onboard RGB-D camera to select a (probabilistically optimal) safe path within the field of view and fit a minimum jerk trajectory along it. The algorithm relies on low-cost, low-range commercial off the shelf (COTS) sensors and average computational capabilities to achieve the speeds up to 3.5 m/s in cluttered indoor as well as outdoor environments. In the experiments reported in this paper, the local planner is solely responsible for enabling the MAV to converge to static or dynamic destinations (validated through simulations). In cases where the local information is insufficient to construct a feasible trajectory to the goal, the planner is designed to safely stop the vehicle.

II. Overview of the Approach

Figure 2 provides a block diagram that illustrates the whole 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 casted from the focal point of the RGB-D sensor in its field of view (See representative Fig. 3).

The receding horizon planner then (i) selects an intermediate point that corresponds to the spatial point in the FOV closest to the target, (ii) assigns a cost to each ray as a weighted sum of the robot's proximity to the obstacle and the proximity to the intermediate point, (iii) selects the safest path to the intermediate point, and finally (iv) constructs the feasible minimum-jerk trajectory along this path. An *initial segment* of that minimum-jerk reference trajectory is then fed into a differential-geometric tracking controller, which starts steering the quadrotor along its way to the intermediate (local goal) point in the FOV. In a typical receding horizon fashion, before the end of that initial segment is reached, the vehicle uses 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 and while the moving target is being tracked.

III. Reactive Receding Horizon Planning

A. Problem Statement

Let $\mathcal{V} \in \mathbb{R}^3$ be the visible space within the FOV of the RGB-D sensor, $\mathcal{P} \subset \mathcal{V}$ a set of isolated points identified by the sensor's point-cloud, and $\mathcal{F} \subseteq \mathcal{V}$ be the representation of the obstacle-free space in which the quadrotor can navigate.

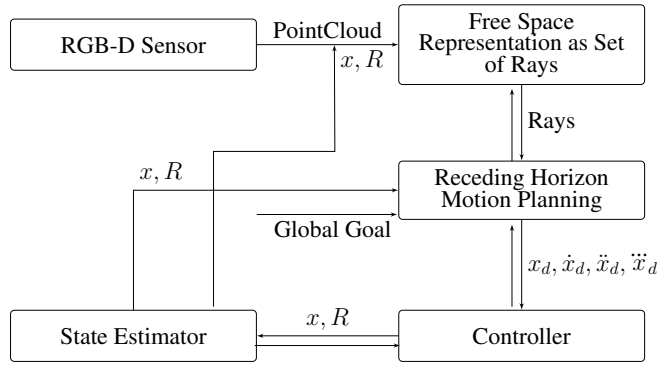


Fig. 2: Block diagram of the motion planning and control architecture.

The quadrotor itself is modeled as a rigid body whose configuration is an element of $\mathsf{SE}(3)$. Denote m and $\mathbf{J}$ its mass and moment of inertia, and x and v its position and velocity in the inertial frame. Let $\mathbf{R}$ be the rotation matrix from body-fixed frame to inertial frame, and Ω the MAV's angular velocity in the body-fixed frame. With the convention that $\hat{\cdot}$ is the skew symmetry operator, $\mathsf{g} = [0, 0, g]^\intercal$ is the gravitational acceleration vector, and $\mathsf{e}_3 = [0, 0, 1]^\intercal$, the (scalar) thrust f and moment (vector) M used as control inputs to stabilize the quadrotor satisfy

$$\dot{\mathsf{x}} = \mathsf{v}, \qquad m\,\dot{\mathsf{v}} = f\,\mathbf{R}\,\mathsf{e}_3 - m\,\mathsf{g} \tag{1a}$$

$$\dot{\mathbf{R}} = \mathbf{R}\,\hat{\Omega}, \qquad \mathbf{J}\,\Omega + \Omega \times \mathbf{J}\,\Omega = \mathsf{M} \tag{1b}$$

The dynamics (1) enjoys differential flatness properties [13], which ensure that all inputs and states can be written as functions of four (flat) outputs associated with the vehicle's 3D position and yaw. These flat outputs can be brought together in a vector $\mathsf{x} = [x, y, z, \psi]^\intercal$. Since ψ is a flat output which can be selected independently, consider (smooth) reference trajectories on position coordinates at planning cycles N and $N+1$, represented as

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

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

The objective is to generate trajectories X_N and X_{N+1} that always remain within $\mathcal{F}$ and satisfy the smoothness condition $X_N(t+\delta t) = X_{N+1}(t+\delta t)$, while being dynamically feasible, i.e. $f_{\min} \leq f \leq f_{\max}$ and $\|\Omega\| \leq \Omega_{\max}$.

B. Representing the Free Space

The key idea behind generating a locally reachable free space is to present the FOV of the RGB-D sensor in the form of a finite, discrete set of points with fixed resolution, from which an *open cover* of the point-cloud measurements has been removed along with all the occluded points. The FOV of the RGB-D sensor $\mathcal{V}$ is assumed to take the shape of a rectangular pyramid that has its apex at the base frame attached to the sensor, its depth direction is aligned with the x (heading) frame axis of the quadrotor, and the sides of the pyramid are determined by the fixed sensor's viewing angles in y and z directions.

Let $R_{\max}$ be the sensor range, $R_{\min}$ be user specified minimum range for planning, and denote the angles of the

pyramid at its apex along the y and z directions ϕ_y and ϕ_z, respectively. A 3D FOV grid of resolution δ_x, δ_y, δ_z (in x, y and z directions as desired by user) within $\mathcal{V}$ with apex at coordinates $(0,0,0)$ relative to the base frame of the sensor, is generated by discretizing the x (depth) coordinate first:

$$X : \{x_i = i \cdot \delta_x : \ \ i \in \mathbb{N}, \ \ R_{\min} \leq x_i \leq R_{\max}\} \ .$$

For each $x_i \in X$, now define the fixed parameters

$$y_i^{\max} = x_i \tan\left(\frac{\phi_y}{2}\right) \qquad z_i^{\max} = x_i \tan\left(\frac{\phi_z}{2}\right)$$

Calculating $n_i^y = \left\lceil \frac{2y_i^{\max}}{\delta_y} \right\rceil$ and $n_i^z = \left\lceil \frac{2z_i^{\max}}{\delta_z} \right\rceil$, for a fixed $x_i \in X$ range-to-sensor, the grid points on the y–z plane are members of the sets

$$Y_i \triangleq \left\{ y_i = -y_i^{\max} + n \cdot \frac{2y_i^{\max}}{n_i^y}, \ \ n \in [0, \ldots, n_i^y] \right\}$$
$$Z_i \triangleq \left\{ z_i = -z_i^{\max} + n \cdot \frac{2z_i^{\max}}{n_i^z}, \ \ n \in [0, \ldots, n_i^z] \right\}$$

and the grid consists of points $x_i \times (Y_i \times Z_i)$ with x_i taking values on parallel planes in the sensor FOV depth direction. Casting a ray from the sensor's focal point to each of these grid points gives a set of possible paths within the sensor's FOV that can be used to traverse if free from collision. Now select a ball of fixed radius r around each point $p \in \mathcal{P}$ of the point-cloud and remove the rays that intersect with any of these balls. This is efficiently done using a KD-Tree data structure. The obstacle pointcloud $\mathcal{P}$ is converted into a KD-Tree and each ray is discretized and represented in the form of a finite set of n interior points (green points in Fig. 3) on it. All these finite sets constitute another KD-Tree that now represents all ray discretizations. These two pointclouds are then queried for nearest neighbour pairs, i.e. find nearest obstacle for each interior point on the ray within radius r–colliding rays (with an obstacle within r) are removed.

The size of the ball with radius r is chosen so that it can fully enclose the quadrotor, with a suitable—based on how conservative with respect to the practical risk collision due to uncertainty or disturbances—safety margin. An illustration of the process is found in Fig. 3; the associated computational requirements are discussed in Section IV. Henceforth it is assumed that the sensor focal point (and FOV apex) is aligned to the center of gravity (COG) of the vehicle, for which the motion planner generates the reference trajectory. Using the (constant) transformation between COG and sensor frame (at the FOV apex), the coordinates of all the points and rays can be expressed relative to body-fixed COG frame of the vehicle.

C. Local Goal and Collision Costs

The optimal path among all possible free (non-colliding) rays should strike an acceptable balance between safety against collisions and speed of convergence to the goal point. There is no global planner here to sketch a complete path from initial to final configuration; in lieu of that, our local planner sets an *intermediate point*, where the straight line between the quadrotor and its final goal intersects with the edge of its FOV, then assigns a cost to each ray, in the form of a linear weighted sum of two cost components: the first is the distance of each ray end-point (black points in Fig. 4(a))

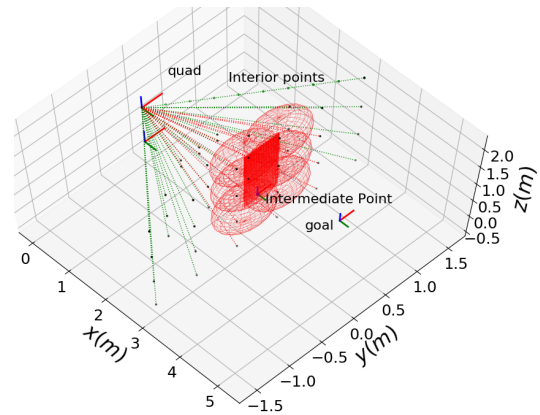


Fig. 3: Checking for collisions. Triads on the plot mark the COG, local goal, global goal and the initial vehicle configuration frames, respectively. Rays colliding with obstacles (depicted by red points) are similarly colored red, while green indicates that the rays are collision free. Black dots represents the sampled points. Interior points used in nearest neighbour search are shown as green dots on only one ray.

to the intermediate point, normalized over the maximum distance; the second is normalized collision cost for the ray.

Denote p the total number of collision-free rays, and d_i the Euclidean distance between the end point of the ith ray and the intermediate point. Set $d_{\max} \triangleq \max_i d_i$, and let $\hat{r} \geq r$ be an additional safety margin (on 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 constructed as

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

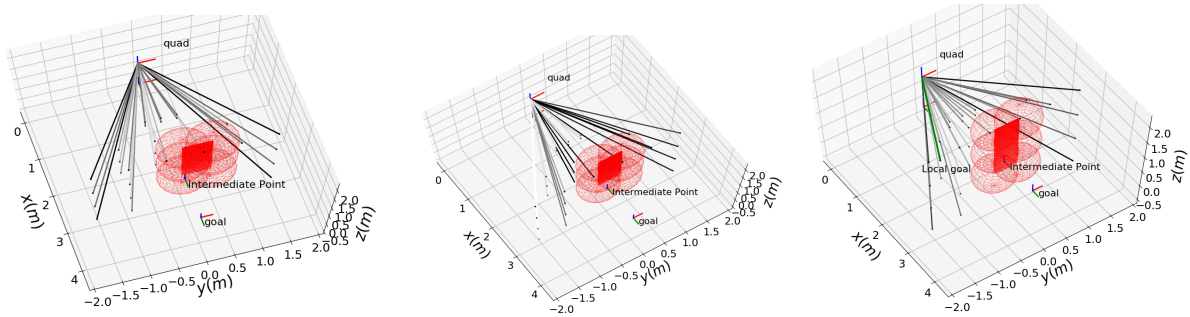
Then for the positive scalar weights $k_1 \in (0,1) \ni k_2$, the total cost associated with ray i is expressed for $i \in [0, \ldots, p]$ as (Fig. 4(c))

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

Once again, a KD-Tree over the obstacle point cloud is utilized to query the distance to nearest neighbour of each interior point of a non-colliding ray (green points in Fig. 3), and compute their minimum to get ρ_i for i^{th} ray. The collision cost function then normalizes the cost of each ray into the $[0,1]$ interval. Thus any ray that touches the ball around its nearest obstacle is assigned a collision cost of 1, while any ray that lies at least $\hat{r}$-away from every obstacle incurs zero cost [21]. All other rays are assigned costs within the $(0,1)$ interval. The end point of the ray with the lowest total cost becomes the *local goal* (i.e., within the FOV) for the planner. The procedure is illustrated in Fig. 4.

D. Receding Horizon Trajectory Generation

With the definition of a local goal within the FOV, the problem now reduces to generating a dynamically feasible optimal reference trajectory from the quadrotor's current location to that local goal that traces the ray that was selected. Since this trajectory will be updated periodically in the spirit of receding horizon control, care must be taken to ensure the continuity of these reference trajectories between update cycles. The reference trajectory segment from the starting location to the local goal is the *planning horizon*.



(a) (b) (c)

Fig. 4: Assigning Costs to Non-colliding rays. (a) Cost of distance of ray end points to the intermediate point, the rays ending father away from the intermediate point incur higher cost, shown as darker lines. (b) Collision cost, the rays closer to the obstacles have high cost (dark) while white rays have zero cost. (c) Total Cost, minimum cost ray and local goal for replanning. The green ray incur minimum cost and ensures safety as well as convergence.

The ray with the least cost that is selected is now divided into n_p equal segments, the endpoints of which now define waypoints for the reference trajectory to be generated. The reference trajectory is produced as a solution of the following minimum jerk optimization problem over the planning horizon, that involves the first three flat outputs of the quadrotor dynamics, namely its Cartesian position coordinate vector $\mathsf{x} = [x, y, z]^\intercal$.

With Δt_j denoting the time interval between two successive waypoints, $(x_{j-1}, y_{j-1}, z_{j-1})^\intercal$, $(x_j, y_j, z_j)^\intercal$, on the reference trajectory, and $T \triangleq \sum_{i=1}^{n_p} \Delta t_i$ the time duration of the whole reference trajectory (the planning horizon), the minimum jerk trajectory O^N between starting waypoint $\mathsf{x}_0 = \left[\mathsf{x}_0^\intercal, \dot{\mathsf{x}}_0^\intercal, \ddot{\mathsf{x}}_0^\intercal, \dddot{\mathsf{x}}_0^\intercal\right]^\intercal$ and end point (local goal) $\mathsf{x}_\mathsf{T} = \left[\mathsf{x}_T^\intercal, \dot{\mathsf{x}}_T^\intercal, \ddot{\mathsf{x}}_T^\intercal, \dddot{\mathsf{x}}_T^\intercal\right]^\intercal$ can be obtained [13], [18] as the solution of the optimization problem

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

Problem (2) is converted to a quadratic program (QP) and efficiently solved using standard solvers (see [7], [18]).

The quadrotor utilizes an in-house customized version of an onboard differential-geometric motion controller (cf. [11]) to track an initial portion of this flat output reference trajectory for a time interval we refer to as the *control horizon*, and is a fraction of the planning horizon T. The length of the control horizon is dependent on the vehicle's speed, its sensor update rate, and its computational capabilities. Specifically, the control horizon should be longer than the *sensor update horizon* (the period between sensor updates) plus some safety margin. Thus, upon receiving new sensor data, the quadrotor generates a new trajectory and appends the newly computed reference trajectory segment to the end of the trajectory segment it is currently implementing. When the quadrotor reaches the end of the control horizon of the reference trajectory segment currently being implemented, it smoothly switches to the next trajectory. The process between sensor updates constitutes a *replanning cycle*. An illustrative example of this method for the relatively simple test scenario of Fig. 3, is presented in Fig. 5.

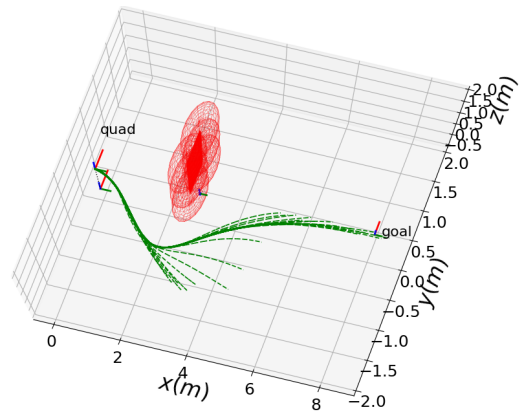


Fig. 5: Trajectory Generation. Dashed green lines shows generated trajectories wile solid green is concatenated trajectory.

The selection of segment time intervals Δt_i (temporal waypoint separation) affects the performance of the trajectory generation algorithm (2). Here, two sigmoid functions involving time and distance are used to define a velocity profile that regulates time-allocation for the entire trajectory from start to goal. Denoting t the time elapsed since the start of the whole planned maneuver, d the vehicle's remaining distance to its goal, and v_{av} the desired average MAV speed, the reference velocity used for time stamping of reference trajectory waypoints during any replanning cycle is generated by the expression

$$v = \operatorname{erf}(k_t \cdot t) \cdot \operatorname{erf}(k_d \cdot d) \cdot v_{\text{av}} \;\; , \tag{3}$$

where k_t and k_d, are positive tuning parameters.

Compared to alternative trapezoid velocity profiles [13], the difference here is that the velocity profile of (3) produced for the entire remaining quadrotor trajectory is effective also in scenarios involving moving target interception, in which the vehicle needs to adjust its speed to match that of its target while at the vicinity of the latter.

This work utilizes checks on thrust and angular velocities proposed in [15] to ensure dynamic feasibility of the reference trajectory. Any failed trajectory is regenerated at reduced v. Smoothness in the yaw angle ψ, on the other hand, is ensured by fitting a third order time polynomial on

yaw angles for the planned trajectory of the form $\psi(t) = a_1 + a_2 t + a_3 t^2 + a_4 t^3$ with $\psi(0) = \psi_0$, $\dot{\psi}(0) = \dot{\psi}_0$, $\psi(T) = \psi_T$, and $\dot{\psi}(T) = \dot{\psi}_T$. Angular rate $\dot{\psi}(T)$ is always kept zero, ψ_T is set so that the quadrotor's camera faces the local goal at time T, while ψ_0 and $\dot{\psi}(0)$ are simply set by the preceding replanning cycle. A number of waypoints n_p along the trajectory keeps the vehicle close to its generating ray, and together with safety margins (Section III-C) and short control horizon, ensures that the final trajectory would be collision free.

On board state estimation is implemented through a visual-inertial MSCKF navigation stack (Open-VINS [5]). The complete implementation for planning, control and state estimation is open-source.[1]

IV. Results

A. Numerical Testing

The reactive planning and control framework was tested in Poisson forest like environment with obstacle densities of 18 and 36 obstacles within a 100 m^2 area, using the RotorS simulation package [3]. Figure 6 shows the probability of success of the mission without hitting any obstacle at different obstacle densities and MAV velocities, indicating an increased likelihood of collision at higher vehicle speeds (cf. [8]). The MAV is more likely to collide with the obstacles at higher velocities or in obstacle-rich environments.

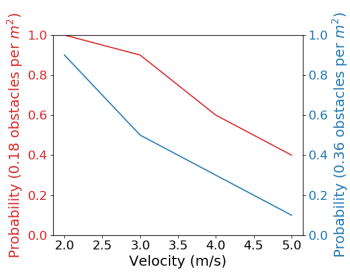


Fig. 6: Mission success probability in Poisson forest environment as a function of obstacle density and MAV velocity. Adjusted to quadrotor size, these densities represents dense obstacle environment, see attached video submission.

B. Experimental Testing

The quadrotor used for experimental testing (Fig. 1) is a custom-build platform, based on a DJI Flamewheel F450 frame. The computational infrastructure includes an onboard Intel NUC Core i7-8650U CPU@1.9GHz×8 and a Pixhawk flight controller. The obstacle point cloud is provided by an Intel Realsense-D435 depth RGB-D camera (640×480 pixel, 30 Hz) while the Realsense-T265 VI-sensor (2 848×800 pixel 30 Hz cameras with a 62 Hz IMU) is used for state estimation. This package provides reliable depth information for up to 3 meters. A voxel filter reduces the density of the generated pointcloud to a uniform size of 0.1 m, which is considered sufficient for typical obstacle avoidance purposes.

On this hardware setup the reactive receding horizon planner and controller can achieve a robust speed of 15 Hz, while Open-VINS runs at 30 Hz, state estimated are fast-propagated and sent to the Pixhawk flight controller at 62Hz. Over five different runs each of overall trajectory length of 25 m in both indoor and outdoor environments (Fig. 7) the 75% quartile is shown to be well below 0.033 seconds; however, to take into account extreme cases and robustify the planning and control loop, sensor update frequency is adjusted at 15Hz (with control horizon of 75ms).

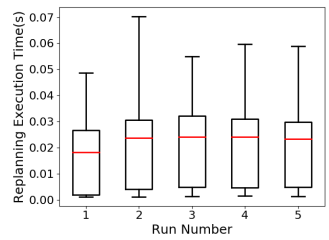


Fig. 7: Replanning execution time. Median is marked in red. 196 rays each having 10 interior points were used during the experiments.

In this configuration, the MAV flew safely among moderately dense obstacles at 3-3.5 m/s in both indoor as well as outdoor environments. In one experiment (see https://youtu.be/CZqLhNsOGHU or the attached video) the MAV flew along a 25 m trajectory around two trees between its starting and the final positions, and demonstrated collision avoidance (with the second tree) at 3.2 m/s. Indoor experiments demonstrated avoidance of collisions with randomly placed chairs and tables at a speed of 2.5 m/s. These speeds surpass those reported in recent literature [2], [4], [20] owing to faster replanning rate.

C. Discussion

Recent literature reports experimental quadrotor navigation results at impressively high speeds [12], [14], [19]; yet most of the systems either employed high-end and expensive sensors with extended range—e.g., Velodyne VLP-16 or Hokuyo UST-20LX LiIDAR mounted on a gimbal to provide 270° FOV [12], [14], compared to a 69.4°× 42.5° FOV cone in this work—or involved a global planner [19]. In addition, the top speeds reported in [19] were achieved at free areas while the average flight speed was reported at 2.4 m/s. In the absence if a global planner, however, a planner with myopic vision cannot arbitrarily increase the speed in anticipation of an unseen obstacle. In contrast to deep learning based approaches (see [9]), the method reported here does not rely on training data and achieves better performance without depending on the training environment. A unique feature of the reported approach is its ability to coordinate MAV motion in pursuit of a moving target (see video supplement). In this case, instead of generating the final trajectory and stopping at the static navigation goal, the MAV intercepts the given target as before but then continue replanning with the target in the FOV as the local goal.

In general, the MAV's speed will ultimately be limited primarily by the computational capabilities, the replanning frequency, and the safety distance (margin) required for the vehicle to stop when the planning fails and the algorithm has to abort—a possibility which cannot be eliminated in purely reactive and local planning methods. Failure may

[1] https://github.com/indsy123/Quadrotor-Navigation-using-Receding-Horizon-planning

occur, for example, if the MAV encounters a long wall or long concave type obstacle. Then, since the free space grid points are generated lines in different planes, unavailability of grid points on the farthest plane indicates that a wall like obstacle is ahead and the planner resorts to a stopping behavior to safely halt the MAV. The necessity of always being able to trigger an emergency behavior in case of such contingencies also poses a limit on the maximum allowable quadrotor speed, depending on the maximum deceleration it can achieve in order to come to a halt. Specifically, given a sensor range R, and a safety margin r set equal to the radius of a virtual sphere fully but tightly enclosing the MAV, the vehicle should be able to stop within a distance of $R - 2r$; assuming a maximum achievable deceleration $a_{\max}$, the maximum safe speed for the vehicle should thus be set at or below $\sqrt{2a_{\max}(R-2r)}$.

Since purely reactive approaches are supposed to rely *exclusively* on local information, convergence to the navigation goal cannot be guaranteed for all possible scenarios. The work reported here extends the envelope of what can be achieved with purely reactive but deliberate quadrotor navigation in cluttered environments, particularly at the low-end of the technology and sensor sophistication spectrum. This investigation naturally exposes the limits of purely reactive motion planning approaches. It is expected that knowledge of those limits, can guide the development of hybrid (local and global [23]) MAV motion planning methodologies destined for deployment in environments where uncertainty is reasonably well characterized, in order to complement each other and operate robustly in real-world scenarios.

V. Conclusions

The challenges that a completely autonomous MAV faces when tasked to navigate in a completely unknown and cluttered environment are drastically exacerbated when the vehicle's sensors are very short-ranged, and thus, the motion planner that aims at operating the robot within a reasonable safety envelop has to strike a balance between safety and aggressive maneuvering. In this context, adaptive motion planning and control strategies within the spirit of receding horizon control appear to be appropriate and effective, and this paper reports on one such method with improved and novel characteristics. The development of this method demonstrates that it is possible to realize end-to-end (incorporating perception, real-time motion planning, and platform control) purely reactive quadrotor navigation strategies, that can be surprisingly effective in a wide range of application scenarios—even if the absence of global information precludes formal completeness guarantees. A testament to the generality of application of this methodology is its usage in dynamic target interception and tracking problems, among the first reported in literature utilizing a purely reactive motion planning approach.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. Openvins: A research platform for visual-inertial estimation. In *Proc. of the IEEE International Conference on Robotics and Automation*, Paris, France, 2020.

[6] 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.

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

[8] Sertac Karaman and Emilio Frazzoli. High-speed flight in an ergodic forest. *CoRR*, abs/1202.0253, 2012.

[9] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments, 2018.

[10] 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 (ICRA)*, pages 1469–1475, May 2016.

[11] 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.

[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] 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, 2011.

[14] Kartik Mohta et al. Fast, autonomous flight in gps-denied and cluttered environments. *Journal of Field Robotics*, 35(1):101–120, 2018.

[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 *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 *2017 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*, pages 649–666. Robotics Research: The 16th International Symposium ISRR. Springer International Publishing, Cham, 2016.

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

[20] Sikang Liu, M. Watterson, S. Tang, and V. Kumar. High speed navigation for quadrotors with limited onboard sensing. In *proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1484–1491, May 2016.

[21] H. G. Tanner and A. Kumar. Towards decentralization of multi-robot navigation functions. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4132–4137, April 2005.

[22] Herbert G. Tanner and J. L. Piovesan. Randomized receding horizon navigation. *IEEE Transactions on Automatic Control*, 55(11):2640–2644, 2010.

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