2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Safety-Critical Rapid Aerial Exploration of Unknown Environments

Andrew Singletary, Thomas Gurriet, Petter Nilsson, and Aaron D. Ames

***Abstract*—This paper details a novel approach to collision avoidance for aerial vehicles that enables high-speed flight in uncertain environments. This framework is applied at the controller level and provides safety regardless of the planner that is used. The method is shown to be robust to state uncertainty and disturbances, and is computed entirely online utilizing the full nonlinear system dynamics. The effectiveness of this method is shown in a high-fidelity simulation of a quadrotor with onboard sensors rapidly and safely exploring a cave environment utilizing a simple planner.**

I. INTRODUCTION

The most prevalent uses for drones, including deliveries, exploration, environmental monitoring, and more, involve navigating through unknown or uncertain environments. Due to the altitude of the vehicles and their often exposed propellers, collisions are catastrophic for the drone and might also be dangerous for its surroundings. For this reason, collision avoidance techniques are crucial to further the use of these systems in everyday life.

In typical drone flight, collision avoidance is the process of creating and tracking trajectories that take the drone through the surrounding free space and avoid occupied or uncertain space. While this approach to collision avoidance can be effective in practice, as evidenced in [1]–[3], it is typically quite conservative and leads to slow mobility, or lacks guarantees of collision-free paths. The conservative aspect of these planners stems from two major hurdles: the computational complexity of the planners that necessitate simplified abstractions of the model and obstacles, and uncertainty in the mapped environment.

Planning in uncertain environments requires frequent updates to the planned trajectories as new information is gained. It is intractable to plan feasible trajectories for the true dynamics of these aerial vehicles in such short time, so traditionally, a global planner with no regard for the system dynamics creates a rough path to follow, and the local planner uses this as a guide to create more realistic, shorter trajectories that can be tracked. Even the trajectories generated by the local planner do not account for the full nonlinear dynamics, as this would require solving a large nonlinear constrained optimization problem, but are instead generated with other assumptions that approximate dynamically feasible paths to various degrees, such as triple integrator models with jerk-limited trajectories [4], [5] or linearizations [6].

While there exist convincing results of collision avoidance at the planning level, it is not the most effective layer in which to enforce safety. Planners update are too infrequent, and there is too much uncertainty stemming from the aforementioned hurdles to be able to provide rigorous guarantees of safety with such a method. Instead, the authors propose a rigorous approach to collision avoidance enforced at the control level that relies solely on the local coordinates and sensor information. This approach is robust to errors or discontinuities in localization and mapping, and can be applied in conjunction with any planning algorithm, or even a human operator. Importantly, this method allows for more primitive planning algorithms that are more aggressive, since they do not need to guarantee collision avoidance or dynamic feasibility.

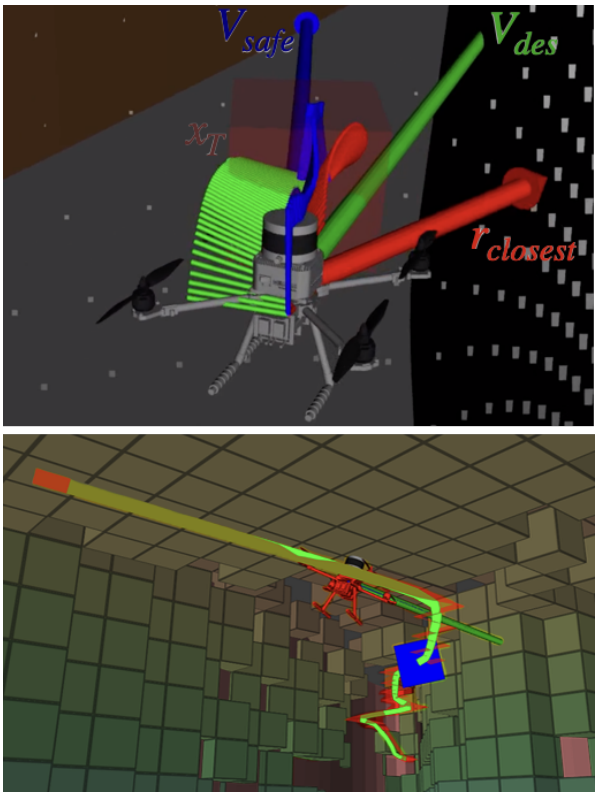

Fig. 1. Simulation environment. The top shows the desired and filtered velocity commands based on the closest point in the point cloud. The bottom shows the drone navigating through the cave.

Similar methods include usage of Hamilton-Jacobi reachability analysis to provide trajectory tracking error bounds [7], which enables collision avoidance for any planner by inflating surrounding obstacles by an appropriate margin. This works very well for systems of low dimensionality (less than ~ 5), but becomes intractable with in higher dimensions, requiring abstractions or simplifications for systems of nontrivial state

Andrew Singletary, Thomas Gurriet, Petter Nilsson, and Aaron D. Ames are with Department of Mechanical Engineering, California Institute of Technology, Pasadena CA 91125, U.S.A. Email addresses: {asinglet, tgurriet, pettni, ames}@caltech.edu

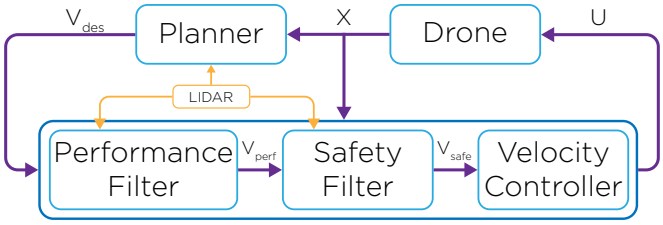

Fig. 2. Structure of the safety filter. U is the control input driving the physical system, X is the state, and V represents the desired velocity commands.

dimension. Another promising direction is to leverage a pre-computed funnel library to chain together robust trajectories at run time [8]. This method has displayed provably safe control in the presence of uncertainty, but is somewhat limited by the finite number of maneuvers in the library that restricts the possible motions.

The novel method presented uses a multi-step filtering approach: a rigorous regulation of the control system that provides guarantees of safety at the level of dynamics, as well as a filter on the desired high-level inputs given by the planner that allows for high-speed travel. This approach requires no offline reachability analysis or trajectory generation, and is applicable to any aerial vehicle that is capable of reaching a stable hover or landing reasonably quickly. A robustness result is also presented, which extends the provable safety to systems with state uncertainty and disturbances, making the method more practical for real robots. Another strength of this method is its ability to maintain safety using only the point cloud data, removing the need to overapproximate obstacles geometrically or through gridding.

The efficacy of this method is demonstrated by showcasing safe, high-speed exploration of an uncertain environment using a relatively simple planner and onboard sensing in a high-fidelity simulation environment with measurement uncertainty and unsensed disturbances.

The layout of the paper is as follows: Section II details the method and how it is used to provide safety guarantees for the system. Section III then outlines how state uncertainty and external disturbances can be accounted for in the framework. Section IV discusses details of the implementation, including the development of the planner, and showcases simulation results. Finally, Section V concludes the work and details future research directions pertaining to this approach.

II. Collision Avoidance Framework

The collision avoidance strategy presented in this work occurs in three stages as illustrated in Fig. 2. The safety guarantees of this method come from the safety filter, which regulates the control inputs in a minimally invasive way such that the system never collides with its surrounding environment. In order to improve the input regulation, the system is first augmented with a velocity controller, making linear velocities the control inputs to the safety filter, rather than the PWM of the motors. To further improve the overall performance of the system, another component, dubbed the performance filter, is used to regulate the velocity inputs sent from the planner in order to minimize the interventions from

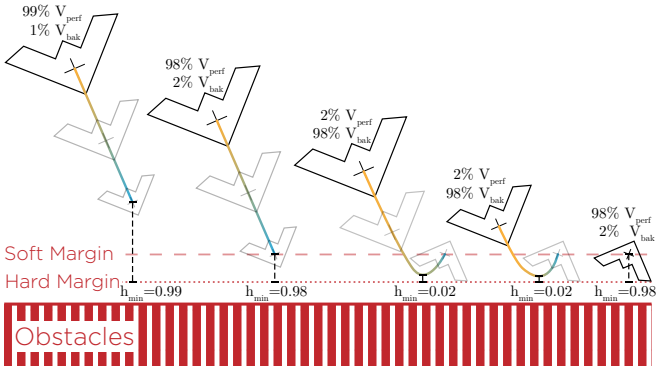

Fig. 3. Illustration of the safety filter. From left to right is a depiction of the evolution of the drone as it gets close to the obstacle. In yellow to blue are the backup trajectories. In grey are the hypothetical positions of the drone if it were to follow the backup trajectory. The size of the drone icon corresponds to its velocity.

the safety filter. All these components then work together to provide a filter with guaranteed collision avoidance, while minimizing conservativeness.

A. Collision Avoidance on Point Clouds

Before detailing the method, it is important to discuss obstacle representation in planning and collision avoidance algorithms. The vast majority of collision avoidance algorithms represent obstacles in one of two ways: either by occupied grids in a map that is periodically updated from point cloud data, or by geometrically representing obstacles (e.g. each obstacle is a polytope or ellipsoid) by clustering point cloud data. While the proposed method is able to work with either of these representations, the authors believe that there are notable advantages to operating directly on the point cloud. Doing this allows for collision avoidance to be done entirely locally, making it free from errors in global localization and mapping. Operating on point clouds also eliminates the inherit conservativeness of approximating obstacles geometrically or by grids, while also benefiting from the fastest update rates.

Because of these reasons, the authors will represent obstacles simply as points in the point cloud, and a collision-free state is considered to be one that maintains positive distance to all points in the point cloud. In between point cloud updates, the point cloud will be fixed, and localization of the drone will be with respect to the fixed point cloud. This representation also showcases the scalability of the method with respect to obstacle resolution and the number of obstacles, thus any other practical representation of obstacles could be utilized instead.

B. Safety filter

Now, the rigorous safety filter will be presented. To prove the safety of this system, some notation and concepts must first be introduced.

Consider continuous-time affine control systems of the form:

$$\dot{x} = f(x) + g(x)u. \tag{1}$$

The functions f and g defined on a compact set $X \subset \mathbb{R}^n$ are continuously differentiable. The controller is restricted to be a function $u : \mathbb{R}^+ \times X \longrightarrow \mathbb{R}^n$ that is locally Lipschitz continuous in state over X and piecewise continuous in time over $\mathbb{R}^+$. The set $U \subset \mathbb{R}^m$ is a compact and convex set of admissible inputs for this system. The final assumption is that system (1) has a unique solution for all time, and for any initial condition $x(0) \in \text{Int}(X)$.

The goal here is to keep a system in some subset of its state-space that is considered safe, $S \subset \mathbb{R}^n$. In order to guarantee safety for dynamical systems, one must find a **control invariant** subset of S that satisfies the following property:

$$\forall x(t_0) \in S,\ \exists u(\cdot) \in U \text{ s.t. } \forall t > t_0, x(t) \in S. \tag{2}$$

If a set is control invariant, then there exists techniques in nonlinear control theory to enforce safety [9], [10]. This can be done by writing the set as the super-level set of a continuously differentiable function $h(x)$,

$$\begin{aligned} S &= \{x \in \mathbb{R}^n \ \mid \ h(x) \geq 0\} \\ \partial S &= \{x \in S \ \mid \ h(x) = 0\}, \end{aligned} \tag{3}$$

and enforcing the condition

$$\frac{\partial h}{\partial x}\left(f(x) + g(x)u(x)\right) + \alpha(h(x)) \geq 0, \tag{4}$$

on the input u for some class-κ function $\alpha(\cdot)$. In this context, $h(x)$ is known as a control barrier function, and for all $x(0) \in S$, the solution to (1) remains in S for all time. Finding an explicit representation of control invariant sets, however, is difficult even for simple systems, and intractable in the general case. To avoid these complexities, the work presented in [11] proposes to use sets that are implicitly defined as a function of the flow of the system under a backup policy.

A backup policy is defined as a controller $u_b(x)$ that desires to bring the system to some forward invariant set, S_b. Note that verifying the invariance of a closed-loop system is far simpler than computing a control invariant set. For many aerial vehicles, this set S_b can be defined as a small tube around a stable hover with near-zero angular and linear velocities, as these vehicles can remain in such a set until they run out of battery. Thus, a suitable backup policy would be coming to a stop as quickly as possible, and thereafter hold the position.

With the assumptions placed on the control system, it holds that for all continuously differentiable u_b there exists a solution to (1) for all time. Therefore, one can define $\phi_t^{u_b} : [0,\infty) \times X \to \mathbb{R}^n$ to be the *flow* of (1) under the control law u_b.

Definition 1. The **safe exploration space** of the system under the backup control policy is defined to be the set:

$$\Omega_T^{u_b} \triangleq \left\{x \in X \mid \left(\phi_T^{u_b}(x) \in S_b\right) \wedge \left(\forall t \in [0,T],\ \phi_t^{u_b}(x) \in S\right)\right\}. \tag{5}$$

It will be proven below that if the set $\Omega_T^{u_b}$ is nonempty, and $x_0 \in \Omega_T^{u_b}$, then under the backup policy $u_b(x)$, the system is forward invariant, and thus remains in the free space for all time. Notably, defining:

$$S \triangleq \{x \in X \mid h(x) \geq 0\}, \tag{6}$$

$$S_b \triangleq \{x \in X \mid h_b(x) \geq 0\}, \tag{7}$$

with h and h_b continuously differentiable, allows one to express the safe exploration space as

$$\Omega_T^{u_b} = \left\{x \in S \mid \min_{t\in[0,T]} h \circ \phi_t^{u_b}(x) \geq 0 \wedge h_b \circ \phi_T^{u_b}(x) \geq 0\right\}. \tag{8}$$

Therefore:

$$h_T^{\Omega}(x) \triangleq \min_{t\in[0,T]} \left\{h \circ \phi_t^{u_b}(x),\ h_b \circ \phi_T^{u_b}(x)\right\} \tag{9}$$

is a control barrier function and one can define a filtering policy that guarantees that if $x(0) \in \Omega_T^{u_b}$ the system will remain in $\Omega_T^{u_b}$, and thus in S.

In previous works [11], [12], it is shown that by integrating sensitivity functions of the dynamics and the backup controller, the barrier condition (4) can be enforced by solving a convex optimization problem at each step, resulting in an optimal control input that keeps the system safe. This algorithm, while convex, is quite computationally expensive for the types of processors that are used for small aerial vehicles, whose computational resources are also needed for sensing, mapping, planning, etc. Thus, a different strategy to enforce the invariance of the system is presented below, that requires no integration of sensitivity functions or optimization problems.

Proposition 1. *Given a smooth function $\alpha : X \times \mathbb{R} \to U$, a control law defined by*

$$u(x) = \alpha\left(x, h_T^{\Omega}(x)\right), \tag{10}$$

regulates solutions of (1) *that stay in $\Omega_T^{u_b} \subseteq S$ for all time if for all $x \in \Omega_T^{u_b}$:*

$$\alpha(x,0) = u_b(x). \tag{11}$$

Proof. From (5), we have that for all $x \in \Omega_T^{u_b}$, the flow $\phi_T^{u_b}(x) \in S_b$ and $\forall t \in [0,T], \phi_t^{u_b}(x) \in S$. Since the system is forward invariant under the backup control law, we have that, under the backup strategy $u_b(x)$, the system stays in $\Omega_T^{u_b} \subseteq S$ for all time from any initial condition inside the set.

From the assumptions on the dynamics and backup controller, we have that $h_T^{\Omega}(x)$ is continuous, and thus know that $h_T^{\Omega}(x)$ will not become negative without passing through 0. Thus, before leaving $\Omega_T^{u_b}$, we have that $h_T^{\Omega}(x) = 0$. If $\alpha(x,0) = u_b(x)$, then the backup strategy will be employed before leaving $\Omega_T^{u_b}$, and thus x is safe for all time. □

Finally, to be able to evaluate the policy online, one only has to be able to evaluate the flow of the system $\phi_t^{u_b}$ for all $t \in [0,T]$. Even though this cannot be done numerically for all $t \in [0,T]$, it can be approximated by numerically integrating the dynamics and evaluating the flow on a finite set of points in $[0,T]$ (see [11]–[13]).

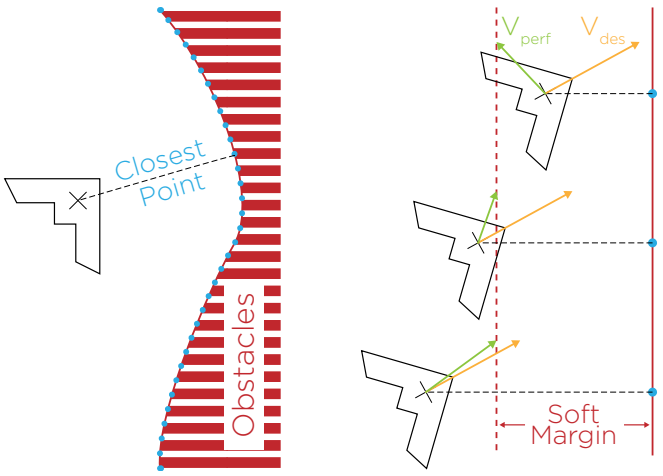

Fig. 4. Illustration of the performance filter.

C. Velocity controller

The velocity controller is a mapping from desired linear velocities to the control input to the robot. Note that, for some aerial vehicles, other desired outputs from the planner can be added here. In the quadrotor example, the yaw rate is also part of the desired velocity vector.

By augmenting the system with this velocity controller in the formulation of the dynamics for the safety filter, it becomes easier to generate the function α presented in the previous section. This is because finding heuristics that result in well-behaved obstacle avoidance is much easier at the velocity level.

D. Performance filter

The performance filter is an advancement of a previous work on a geofencing for civilian UAVs [14]. The idea presented there is to remove the component of the velocity vector in the direction of the nearest obstacle when the drone is in close proximity, as illustrated in Fig. 4. As the drone gets closer to the soft margin that defines how far away from an obstacle one wishes to stay, more and more of the velocity vector in that direction is removed. If the drone reaches the inside of the soft margin, the velocity vector is altered to push the drone back out.

Readers should note that, with perfect velocity control, this method alone would guarantee safety of the system. However, since this is not the case for physical systems, the above safety filter is needed.

III. ROBUSTNESS

A drawback of the method showcased in previous works [11] [12] is that, while the method was scalable to higher dimensional systems, the computational complexity of the method did not allow for any robustness guarantees to be added. However, by eliminating the difficult gradient computations and sensitivity integration, it is now possible to provide guarantees on the safety of the system in the presence of state uncertainty and disturbances.

This is important because, while the above framework provides safety guarantees for the ideal case, if the initial state of the vehicle is uncertain, or there are external forces that impact the system, the guarantees no longer hold. For safety to be established in a more realistic context, it must be done for all possible backup trajectories resulting from all possible initial conditions and disturbances.

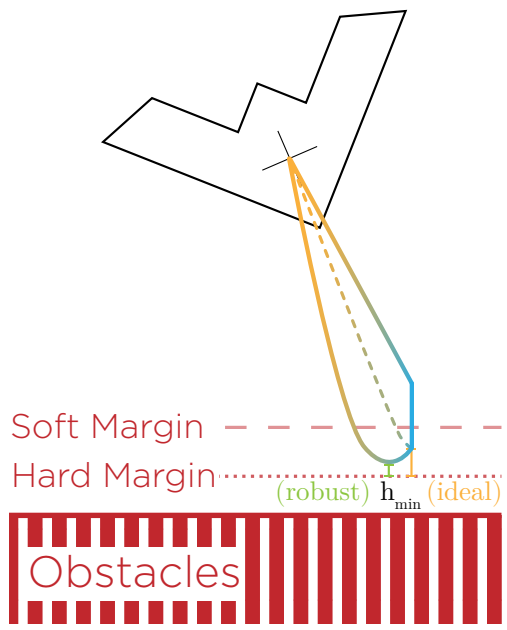

Fig. 5. Illustration of a robust backup trajectory.

Consider a vehicle governed by the closed loop dynamics

$$\dot{x} = f(x) + g(x)u(x) + d, \tag{12}$$

where $d \in D$ represents a bounded additive disturbance term. For such a system, the flow is a set valued map $\phi_t^{u_b} : X \rightrightarrows X$, and the backup trajectory is a tube. If the dynamics are linear and the disturbance term additive, computing this tube can be solved with little computational overhead using techniques from linear reachability analysis [15]. Also, if a nonlinear system is monotonic, reachability for additive disturbances can be done almost trivially [16], however these types of systems are rare in robotics. For more general systems, several reachability analysis tools exist [17], [18], however these tools are not well suited for online computation at the update rates required for stable flight. Sampling-based methods would be more appropriate for online computation, but proving high-confidence safety with these methods requires much care and knowledge of how the dynamics evolve.

A suitable alternative to reachability algorithms for this application is the use of interval arithmetic [19]. Since the sets of possible initial conditions and disturbances are bounded, we can represent them as intervals whose uncertainty propagates through our evaluation of the dynamics and subsequent integration. This results in the dynamics and thus the flow of the system being set-valued functions whose range of values grows over time.

One important caveat to consider while propagating uncertainty is the effect that the uncertainty has on the controller. As the set of reachable states for the system grows, the set of possible input values grows as well for any sort of

feedback controller. When these intervals enters the inputs, the uncertainty of the system can blow up rapidly. In this case, during integration, the nominal trajectory should be used for the state feedback given to the controller during the integration of the backup trajectory, i.e. the trajectory with zero external disturbance starting from center of the initial state interval.

Fig. 5 illustrates the concept of checking the safety condition over the entire set of possible trajectories. The dashed line shows one of the possible trajectories, which is encompassed in the integration of the intervals of bounded uncertainty. The size of the position uncertainty at the end of the backup trajectory for a maximum disturbance of 1 N in all directions is shown in Fig. 1 as x_T.

IV. IMPLEMENTATION

In this section, we will detail the steps taken to implement the collision avoidance algorithm on a high-fideltity quadrotor simulator, and showcase the method working with a simple planner to explore a large 240m by 460m cave system.

A video detailing this work and showcasing the exploration can be found here [20].

A. Dynamics

The model used for the demonstration is a 16-dimensional, nonlinear model of a quadrotor with voltage inputs, to be as realistic to the actual system as possible.

A standard 12-dimensional model for a quadrotor is first obtained from force-balance equations in a rotating reference frame (e.g. [21], [22]). Let the 12-dimensional state be $\mathbf{x} = [\mathbf{r}, \mathbf{v}, \xi, \omega]^\intercal$, where $\mathbf{r}$ and $\mathbf{v}$ are position and velocity in $\mathbb{R}^3$, $\xi = [\phi, \theta, \psi]^\intercal$ are roll, pitch and yaw angles, and $\omega \in \mathbb{R}^3$ are angular velocities in the quadrotor body frame. Then

$$M\frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathbf{r} = \begin{bmatrix} 0 \\ 0 \\ -Mg \end{bmatrix} + F_z R(\xi) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{13a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\xi = T(\xi)\omega, \tag{13b}$$

$$J\frac{\mathrm{d}}{\mathrm{d}t}\omega = \tau - (\omega \times (J\omega)). \tag{13c}$$

Here $R(\xi)$ is the x-y-z rotation matrix from a body-fixed frame to the world frame, and T the resulting mapping between angular velocities:

$$\begin{aligned} R(\xi) &= R_z(\psi)R_y(\theta)R_x(\phi), \\ T(\xi) &= \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sec(\theta)\sin(\phi) & \cos(\phi)\sec(\theta) \end{bmatrix}. \end{aligned} \tag{14}$$

The vertical force and angular torques acting on the body are obtained from motor angular velocities

$$F_z = k_f\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right), \tag{15a}$$

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} -lk_f & -lk_f & lk_f & lk_f \\ -lk_f & lk_f & lk_f & -lk_f \\ -k_t & k_t & -k_t & k_t \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}, \tag{15b}$$

with $l = \frac{D}{2\sqrt{2}}$ and D the frame diameter. Since the angular velocities of the propellers cannot be controlled directly, their response to a voltage input V_i must be modeled. The equation of motion for the angular velocity Ω_i of each motor is

$$(J_{rot} + J_{prop})\dot{\Omega}_i = \frac{1}{K_v R}\left(V_i - \frac{\Omega_i}{K_v}\right) - k_t\Omega_i^2. \tag{16}$$

Disturbances enter the system through external forces acting on the center of mass of the vehicle.

B. Attitude and Velocity Controller

The attitude controller is simple cascade PD controller on the angles and angular rates of the quadrotor. The motor voltages at the output of the velocity controller are given by

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} u_z - u_\phi - u_\theta - u_\psi \\ u_z - u_\phi + u_\theta + u_\psi \\ u_z - u_\phi + u_\theta - u_\psi \\ u_z - u_\phi - u_\theta + u_\psi \end{bmatrix} \tag{17}$$

where

$$u_z = K_{vz}(v_{z,\mathrm{des}} - v_z) + u_{\mathrm{hover}}(\phi, \theta) \tag{18}$$

$$u_\phi = K_\phi(K_v(v_{y,\mathrm{des}} - v_y) - \phi) - k_\phi\omega_x \tag{19}$$

$$u_\theta = K_\theta(K_v(v_{x,\mathrm{des}} - v_x) - \theta) - k_\theta\omega_y \tag{20}$$

$$u_\psi = -k_\psi(\omega_{z,\mathrm{des}} - \omega_z). \tag{21}$$

Note that the upper-case K gains are the proportional gains, while the lower-case k gains are the derivative gains. Also, $u_{\mathrm{hover}}(\phi, \theta)$ is the input required to maintain a constant height.

C. Backup Controller and Safety Filter

The backup controller is simply the velocity controller fed with a desired velocity of zero.

The filtering law α is chosen to be

$$v_{\mathrm{safe}} = \lambda(x)v_{\mathrm{perf}} + (1 - \lambda(x))\, v_b, \tag{22}$$

with

$$\lambda(x) = 1 - e^{-3h_T^\Omega(x)/\Delta_m}, \tag{23}$$

where Δ_m is the difference between the soft and hard margin. The backup policy v_b being the same control law as the one used in the performance filter.

D. Planner

The planner is designed to work in tandem with the Octomap mapping library [23]. Octomap uses point cloud data to construct a map that is stored in an octree. When a point cloud ray passes through a voxel for the first time it is added to the map, and afterwards an estimated occupancy probability p is maintained. The map therefore segments the space into free (with probability $1 - p$), occupied (with probability p), and unknown.

In order to explore an unknown environment, one must define frontier points as possible goals for the global planner. For this implementation, a frontier is defined as a cell in free space that is adjacent to unknown space. Due to the large number of frontiers that are generated during exploration

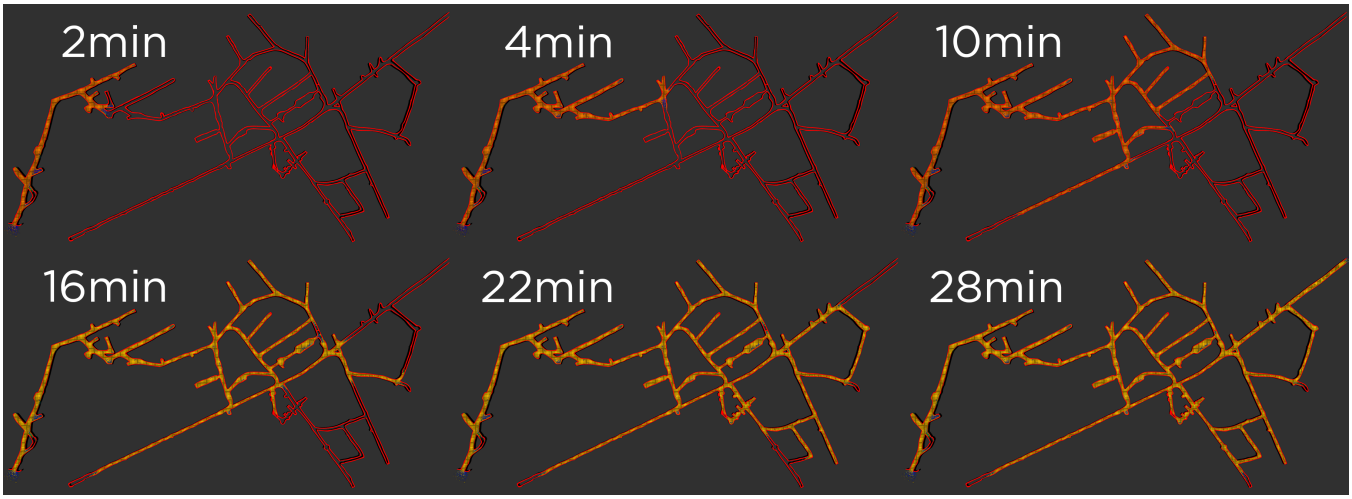

Fig. 6. Pictures of the cave (in red) and the octomap (in yellow) being built throughout the 28 minutes it takes for the drone to completely explore the cave.

and the uncertainty inherent in mapping, nearby frontiers are clustered together, and the planner uses cluster size as a heuristic for target selection.

The global planner is a basic implementation of the A* algorithm that searches for a path to a nearby frontier cluster. The search algorithm runs directly on the octomap, which is possible via implementation of algorithms that enumerate neighbors in a 3D octree [24]. The heuristic encourages the planner to visit large clusters that are close to the current position of the drone. After a global plan is achieved, the local planner creates a spline to form a smoother version of the global plan, as show in Fig. 1. In order to smoothly track the spline, the velocity controller is given a desired waypoint 2m in front of the closest point to the drone along the spline. The position of the closest point along the spline is estimated by discretizing the spline and searching for the minimum distance over this countable set of spline points. The closest point is also constrained to never cause the drone to reverse along the spline. A path is recomputed using the A* algorithm when the drone gets close to the end of the spline, or if the last path is more than 10 seconds old.

E. Interval Arithemtic for Robustness

The Boost interval library was selected for this implementation, with the rounding policy set to `save_state<rounded_transc_exact<double>>` to allow the use of transcendental functions. The data types of the states were changed from `double` to `interval<double>` to enable interval arithmetic.

In order to check the point cloud over the entirety of the trajectory, the distance metric was changed to be the minimum euclidean distance to the bounding box of possible drone positions.

F. Simulation Environment and Results

The simulation environment is a ROS-based C++ environment. The point cloud data is obtained from a Velodyne LIDAR sensor inside of the Gazebo simulator at a frequency of 10 hz. The simulation, including visualization in Gazebo and RVIZ, was able to run at a frequency of 500 hz on a modern laptop computer with an Intel i9 CPU.

The cave environment was a large 240m by 460m structure with one entrance and one exit. The cave height is constant at roughly 3m, but the width is constantly changing, and gets as small as 0.75m with several protruding areas. For reference, the size of the quadrotor is 0.5m in diameter.

The quadrotor was able to explore the entire 240m by 460m cave in just under 28 minutes. The maximum allowable speed from the planner was 5 m/s, which the drone reached during open areas of the cave. The average desired speed sent from the planner was 4.09 m/s, and the average speed of the drone after the safety filter was 3.28 m/s.

A positive value of the barrier function described in Section II was maintained throughout, meaning the quadrotor never went closer than the minimum allowed distance to a point in the point cloud, which was set at 0.2 meters.

V. CONCLUSION

In this paper, the authors presented a novel approach for collision avoidance at the controller level. This method uses concepts from set invariance to provide safety guarantees with respect to local point cloud data. This approach was shown to work in a high-fidelity simulation environment with a very simple planner with no offline computations.

For future work, we plan to apply this system on a quadrotor with an Intel Realsense D435 providing point cloud data. We also plan to more tightly integrate the planner and the safety filter, allowing the safety filter to inform the planner when there is a discrepency between the map and the point cloud data.

Another possible research direction is applying this technique to moving obstacles. A disadvantage to planning over maps is that they frequently update too slowly to reliably plan high-speed trajectories in dynamic environments. This could enable the proposed method to be a strong candidate for planning in dynamic environments.

REFERENCES

[1] 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.

[2] Y. Lin and S. Saripalli, "Sampling-based path planning for uav collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3179–3192, 2017.

[3] S. Hrabar, "3d path planning and stereo-based obstacle avoidance for rotorcraft uavs," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 807–814.

[4] J. Tordesillas, B. T. Lopez, and J. P. How, "FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments," *arXiv e-prints*, p. arXiv:1903.03558, Mar 2019.

[5] S. Lai, K. Wang, H. Qin, J. Q. Cui, and B. M. Chen, "A robust online path planning approach in cluttered environments for micro rotorcraft drones," *Control Theory and Technology*, vol. 14, no. 1, pp. 83–96, 2016.

[6] J. Dentler, S. Kannan, M. A. O. Mendez, and H. Voos, "A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors," in *2016 IEEE conference on control applications (CCA)*. IEEE, 2016, pp. 519–525.

[7] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: a modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1517–1522.

[8] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[9] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[10] J.-P. Aubin, *Viability theory*. Springer Science, 2009.

[11] T. Gurriet, M. Mote, A. D. Ames, and E. Feron, "An online approach to active set invariance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 3592–3599.

[12] A. Singletary, P. Nilsson, T. Gurriet, and A. D. Ames, "Online active safety for robotic manipulators," in *2019 International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, p. Final Version Submitted.

[13] T. Gurriet, M. Mote, A. Singletary, E. Feron, and A. D. Ames, "A scalable controlled set invariance framework with practical safety guarantees," in *2019 IEEE Conference on Decision and Control (CDC)*. IEEE, 2019.

[14] T. Gurriet and L. Ciarletta, "Towards a generic and modular geofencing strategy for civilian uavs," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 540–549.

[15] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2005, pp. 291–305.

[16] N. Ramdani, N. Meslem, and Y. Candau, "Computing reachable sets for uncertain nonlinear monotone systems," *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 263–278, 2010.

[17] X. Chen, E. Abraham, and S. Sankaranarayanan, "Taylor model flowpipe construction for non-linear hybrid systems," in *2012 IEEE 33rd Real-Time Systems Symposium*. IEEE, 2012, pp. 183–192.

[18] S. Kong, S. Gao, W. Chen, and E. Clarke, "dreach: δ-reachability analysis for hybrid systems," in *International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems*. Springer, 2015, pp. 200–205.

[19] R. E. Moore, *Methods and applications of interval analysis*. SIAM, 1979.

[20] "Video of the simulation." https://www.youtube.com/watch?v=MdOKY-ykCSw.

[21] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE ICRA*, 2011, pp. 2520–2525.

[22] D. Zhou and M. Schwager, "Vector field following for quadrotors using differential flatness," *Proc. IEEE ICRA*, pp. 6567–6572, 2014.

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

[24] H. Samet, "Neighbor finding in images represented by octrees," *Computer Vision, Graphics and Image Processing*, vol. 46, no. 3, pp. 367–386, 1989.