Fast and Agile Vision-Based Flight with Teleoperation and Collision Avoidance on a Multirotor

Alex Spitzer*, Xuning Yang*, John Yao, Aditya Dhawale, Kshitij Goel, Mosam Dabhi, Matt Collins, Curtis Boirum, and Nathan Michael

Abstract We present a multirotor architecture capable of aggressive autonomous flight and collision-free teleoperation in unstructured, GPS-denied environments. The proposed system enables aggressive and safe autonomous flight around clutter by integrating recent advancements in visual-inertial state estimation and teleoperation. Our teleoperation framework maps user inputs onto smooth and dynamically feasible motion primitives. Collision-free trajectories are ensured by querying a locally consistent map that is incrementally constructed from forward-facing depth observations. Our system enables a non-expert operator to safely navigate a multirotor around obstacles at speeds of 10 m/s. We achieve autonomous flights at speeds exceeding 12 m/s and accelerations exceeding 12 m/s^2 in a series of outdoor field experiments that validate our approach.

1 Introduction

Autonomous aerial vehicles with onboard vision-based sensing and planning have been shown to be capable of performing fast and agile maneuvers. However, long-horizon planning required to achieve a task has proven to be a challenge, particularly with limited onboard compute. We propose a fully integrated vision-based multirotor platform that allows minimally trained operators to teleoperate the vehicle at high speeds with agility, while remaining safe around obstacles in unstructured outdoor environments. At high speeds, the environment around the vehicle changes quickly, and is subject to dynamic obstacles and lighting conditions. Our multirotor architecture integrates the following to achieve agile and collision-free flight in these scenarios: 1) an extension of motion primitive-based teleoperation [12] to generate jerk-continuous local trajectories, a crucial component to prevent instability in agile flights, and 2) efficient local mapping for collision avoidance purposes using a KD-tree.

Many prior works in high speed flight exploit the field-of-view (FOV) of stereo cameras for fast collision checking for autonomous flights, including [4, 5], where

Authors are with the Robotics Institute at Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh PA, 15213. {aspitzer,xuningy,johnyao,adityand,kgoel1,mdabhi,mcollin1,cboirum,nmichael}@andrew.cmu.edu

* These authors contributed equally.

trajectories are constrained to be inside the FOV of the depth sensor with a max range of 10 m. In [8], trajectories generated by RRT* are checked for collisions directly in the disparity space. Lopez and How [7] presents aggressive flight on a 1.2 kg MAV, achieving a velocity of 3 m/s. These methods achieve fast collision checking by circumventing the need to construct a local map and checking for collisions in the sensor's FOV. This however, limits the range of motions the vehicle can perform. Approaches with local map generation using a laser range finder [1] and a monocular RGB camera [3] have been shown to achieve maximum velocities of 1.8 m/s and 1.5 m/s respectively, but data processing limits the update rate of the local maps. In our approach, we give a minimally trained operator full control of the vehicle and show that fast and agile flights can be achieved with a human-in-the-loop while maintaining safety.

We perform a series of high speed collision avoidance trials in both indoor and outdoor environments with untrained operators. In our experiments, our hexarotor attains speeds exceeding 12 m/s and accelerations exceeding 12 m/s^2. We are able to safely avoid obstacles at speeds up to 10 m/s and accelerations of 8 m/s^2, while retaining a local map.

2 Technical Approach

2.1 Smooth Motion Primitive-Based Teleoperation

Aggressive multirotor flights demand large angular velocities and large angular accelerations, which are directly related to the jerk and snap of the reference position [9]. Thus to avoid incurring large tracking error due to discontinuous trajectories, we extend *forward-arc motion primitives* [12] to generate trajectories that retain differentiability up to jerk and continuity up to snap. From the resulting trajectories, we can calculate desired vehicle attitude, angular velocity, and angular acceleration for use as feedforward terms in the controller.

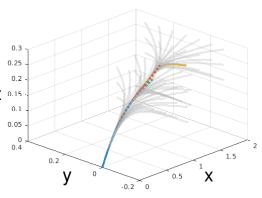
The motion primitives are parameterized as follows. We define a *local frame* $\mathcal{L}$ to be a fixed z-axis aligned frame, taken at a snapshot in time. The motion primitive definition will be provided in the local frame at the time at which an input is issued, and can be freely transformed into a fixed global frame or body frame for control purposes. An action $\mathbf{a}$ specified by a continuous external input, such as a joystick, generates a single motion primitive. For a multirotor, we define an action as $\mathbf{a} = \{v_x, v_z, \omega\}$ in the local frame at which the input is issued, where v_x is the x-velocity (i.e., the forward velocity of the vehicle), v_z is the z-velocity, and ω is the yaw rate. Let $\mathbf{x}$ denote a vector containing the position and yaw of the vehicle, i.e. $\mathbf{x} = [x,\ y,\ z,\ \theta]$. Then, the endpoint velocities are defined by the unicycle model [10]. The unicycle model dynamics are given by $\dot{\mathbf{x}}(\mathbf{a}, \tau) = [v_x \cos(\omega\tau) \quad v_x \sin(\omega\tau) \quad v_z \quad \omega]^\top$, where $\tau \in [0, T]$. The position endpoints are unconstrained and we enforce all higher order derivatives above velocity to be zero.

A regeneration step k occurs when a new input is received from the joystick, or the previous trajectory $\boldsymbol{\gamma}(\mathbf{a}_{k-1}, T)$ finishes executing. Alternatively, a fixed regeneration rate can be chosen in order to accommodate changes in the environment for collision avoidance. Suppose regeneration step k occurs at time t_k. Then, a library of dynamically feasible motion primitives is generated in the local frame $\mathcal{L}_{t_k}$ specified by the reference state at time t_k, i.e. $\mathbf{x}_{\text{ref}}(t_k) = \boldsymbol{\gamma}(\mathbf{a}_{k-1}, t_k - t_{k-1})$, given a set of discretized actions $\mathbf{a}_k$. Each motion primitive $\boldsymbol{\gamma}$ is a vector of four 8th order polynomials that specify the trajectory along the position coordinates x, y, z and yaw coordinate θ. Given an action $\mathbf{a}_k$ at regeneration step k, each motion in the motion primitive library is generated in frame $\mathcal{L}_{t_k}$ according to

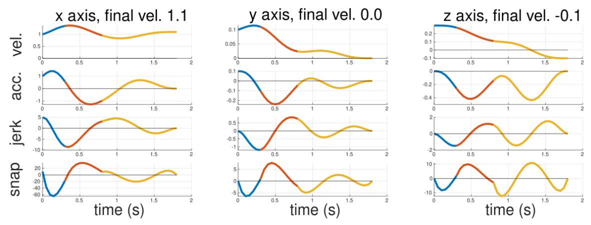
$$\begin{aligned} \boldsymbol{\gamma}(\mathbf{a}_k, \tau) &= \sum_{i=0}^{8} \mathbf{c}_i \tau^i \\ \text{s.t. } \boldsymbol{\gamma}^{(j)}(\mathbf{a}_k, 0) &= \mathbf{x}_{\text{ref}}^{(j)}(t_k) \;\; \text{for } j = 0,1,2,3,4 \\ \dot{\boldsymbol{\gamma}}(\mathbf{a}_k, T) &= \dot{\mathbf{x}}(\mathbf{a}_k, T) \\ \boldsymbol{\gamma}^{(j)}(\mathbf{a}_k, T) &= 0 \;\; \text{for } j = 2,3,4 \end{aligned} \tag{1}$$

where $\{\cdot\}^{(j)}$ specifies the j^{th} time derivative. Note that all constraints are appropriately transformed into $\mathcal{L}_{t_k}$. The duration of the trajectory T and the maximum x, z, and yaw velocities are specified according to the desired operational range. We further allow the operator to freely rotate the motion primitive library with respect to $\mathcal{L}_{t_k}$'s z-axis to allow for sideway slalom motions.

The result of having snap-continuous trajectories (see Fig. 1) ensures that we have smoothness in error dynamics, thus minimizing instabilities and tracking error.



(a)



(b)

Fig. 1: (a) A trajectory composed of 3-segments of motion primitives that switches to a new motion primitive at arbitrary points along the trajectory that have non-zero higher-order-derivative terms. The discarded trajectory is shown in dotted lines. (b) Higher-order time derivatives (velocity, acceleration, jerk, and snap) of the three segment trajectory, showing that the trajectories are differentiable up to jerk and continuous in snap at the switching points. At the end of the trajectory, all higher order derivatives are zero except for the user specified velocity.

2.2 *Pruning & Trajectory Selection.*

At every time step, a family of motion primitives, called the motion primitive library, is created. The motion primitive library is constructed by discretizing the continuous input along each action dimension, such that each action $\mathbf{a}_i \in A$ is selected from a convex set $A := \{\mathbf{a} \in [\mathbf{a}_{\min}, \mathbf{a}_{\max}]\}$ with size $N_1 \times N_2 \times \cdots \times N_n$ where N_i is the

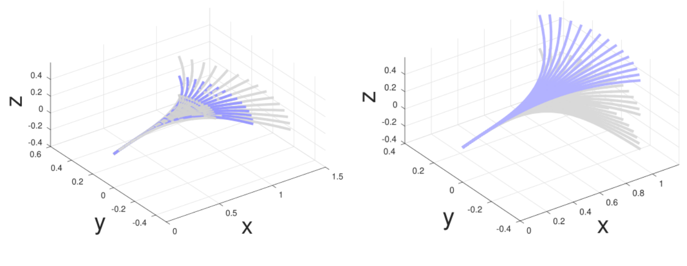


Fig. 2: Motion primitive library with (a) variation in linear velocity and with (b) variation in z velocity

dimension of the space of each input. An example of the discretization is shown in Fig. 2.

At every time step, the operator input is mapped to the closest input in the action space, as defined by the Euclidean norm. A priority queue that minimizes input distance from the selected input $\mathbf{a}_{\text{joystick}}$ to each input in the action space $\mathbf{a}_i \in A$ is used to iterate through the action space until a feasible, collision-free trajectory is found. This results in having the operator input mapped to the feasible motion primitive in the library that is parameterized by the closest discretized action, i.e. $\boldsymbol{\gamma}(\mathbf{a}_i) = \operatorname{argmin} \|\mathbf{a}_{\text{joystick}} - \mathbf{a}_i\|$. A trajectory is deemed collision-free if the minimum distance between any point along the trajectory and the surrounding environment is above the sum of the vehicle size and an operator specified collision radius. Algorithm 1 describes teleoperation with reactive collision avoidance in detail.

The effect of this pruning algorithm is that the vehicle exhibits natural behavior in the presence of obstacles. If a pillar is in front of the vehicle, then the vehicle chooses a motion primitive some angle away and avoids the obstacle. If a wall is present, then the vehicle will choose linear velocities that gradually decrease until the vehicle is stopped.

2.3 Local Map Representation using KD-Trees

We present a local mapping framework that generates a spatially consistent local map of the robot surroundings represented as voxel grids. The local map is generated by retaining only the depth sensor measurements obtained at poses that lie in the vicinity of the vehicle's current pose. This enables trajectories to span in the space observed by all of the retained sensor measurements. Sequential sensor measurements obtained using a stereo imaging sensor often contain redundant information about the surroundings of the robot. The novelty of information in an incoming sensor measurement at the resolution of the voxel grids is dictated by the rotation and translation displacement of the robot between the current frame and the previous frames. To enforce spatial locality, we dynamically select anchor frames and transform subsequent sensor measurements that contain novel information about the surroundings in to the anchor frames. In order to efficiently incorporate only novel information

Algorithm 1 Snap-continuous Motion Primitives based Teleoperation and Collision Checking with KD-Tree Local Map

```
 1: Given KD-Tree local map $L$, collision radius $r$, vehicle radius $r_v$
 2: Receive input $\mathbf{a}_{\text{joystick}}$
 3: Generate the minimum input distance queue according to $d_i = \|\mathbf{a}_{\text{joystick}} - \mathbf{a}_i\|$
 4: while $\mathbf{a}_i$ is infeasible do
 5:     Pop the top action element off of the minimum input distance queue $\mathbf{a}_i$
 6:     Generate $\boldsymbol{\gamma}_i = \boldsymbol{\gamma}(\mathbf{a}_i)$ according to (1)
 7:     for $\tau = 0 : T$ discretized at some $\triangle t$ do
 8:         Query $L$ for the closest point $p$ to $\boldsymbol{\gamma}_i(\mathbf{a}, \tau)$
 9:         if $\|p - \boldsymbol{\gamma}_i(\mathbf{a}, \tau)\| \geq r + r_v$ then
10:             Set $\mathbf{a}_i$ to feasible
11:             Set $\boldsymbol{\gamma} = \boldsymbol{\gamma}_i$
12:         end if
13:     end for
14: end while
```

in the local map, we classify each incoming sensor measurement into one of the following categories: a KeyFrame (*KF*), a SubFrame (*SF*), or a BufferFrame (*BF*) (see Table 1). The local map (*L*) is updated in a locally consistent coordinate frame according to the type of sensor frame ($F \in \{KF, SF, BF\}$).

A sensor measurement that is more than α_k meters away from the current *KF* is classified as a *KF*. Sensor measurements that are not new *KF*s, but are either more than α_s meters in position or β_s degrees in heading away from the previous *SF*, are classified as *SFs*. Sensor measurements that are neither *KF*s nor *SF*s are classified as *BF*s, which do not contain sufficient novel information about the surroundings, but are registered to *L* to account for dynamic changes in the scene. *BF*s are in *L* only for the iterations in which they are observed.

Table 1: Frame Classification

Frame Type	**Definition**	**Initialization Condition for Frame *F***
KF (Anchor)	Sensor frames to which subsequent *SFs* and *BFs* are registered	$T\,(KF_{i-1}, F) > \alpha_k$
SF	Sensor frames that provide novel information about the vehicle surroundings	$T\,(SF_{i-1}, F) > \alpha_s \|\|$ $H\,(SF_{i-1}, F) > \beta_s$
BF	Sensor frames that are registered for one time step to accommodate the dynamic changes in the surroundings when the vehicle has not been displaced by a sufficient distance	$F \neq (KF, SF)$

$T\,(a,b)$ is the translational distance between a and b; $H\,(a,b)$ is the heading rotational distance between a and b.

A new local map *L* is spawned every time a new *KF* is detected. *L* consists of all the *SF*s registered to the current *KF*, the current *BF* and all the *SF*s registered to the previous *KF*, resulting in a voxel grid map representing the occupied space centered around the vehicle. This allows us to only update the KD-Tree that contains the voxel centers of the *KF* to which the incoming sensor scan is registered. Because a high fidelity map is not essential for collision avoidance, each sensor measurement is downsampled into an occupancy grid with a fixed voxel size before registering it to L. Voxel centers in the map are then arranged in a KD-Tree to minimize the time complexity of nearest neighbor queries.

The algorithmic efficiency and independence of the definition of the local map from the sensor model enables our approach to incorporate measurements from multiple depth sensors with widely separate FOVs.

2.4 State Estimation

We use VINS-Mono [11], a tightly-coupled visual-inertial odometry framework that has been shown to perform favorably when compared to other state of the art open source state estimation algorithms [2]. VINS-Mono jointly optimizes vehicle motion, feature locations, camera-IMU extrinsics, and IMU biases over a sliding window of monocular images and preintegrated IMU measurements. Because our local planning strategy does not require a globally consistent map, we disable the loop closure functionality of VINS-Mono to reduce its computational footprint. We run an auxiliary state estimator during takeoff in order to provide smooth state estimates when the vehicle has not yet experienced sufficient motion excitation for VINS-Mono to initialize. The auxiliary state estimator is an unscented Kalman filter that fuses downward rangefinder altitude observations, downward optical flow, and IMU measurements to estimate vehicle velocity, position, and attitude.

3 Implementation

Hardware. We experimentally evaluate our proposed approach on a 3.8 kg hexarotor that fits within a 20 cm $\times$ 60 cm $\times$ 80 cm volume (Fig. 3a). The hexarotor has an average flight time of 7 min and a power to weight ratio of 3. Two Intel RealSense D435 depth cameras are used for mapping: one facing forwards and one facing upwards at a 45 degree angle, which aids obstacle avoidance while accelerating forwards. A downward facing Matrix Vision mvBlueFOX-MLC200w is used as the RGB camera input for VINS-Mono and the NVIDIA Tegra TX2 is used for computation. The hexarotor uses a cascaded PD control architecture as in [9] with jerk and snap references used to compute feedforward angular velocities and accelerations.

Teleoperation. Motion primitives are generated with an angular velocity bound of 2 rad/s. There are 25 discretizations for the v_x action, 11 for ω, and 5 for v_z for a total of 1375 motion primitives generated per trajectory iteration. Trajectories are generated at 25 Hz, and the local map is updated at 30 Hz.

We follow [6] and choose the maximum velocity of the motion primitives to be such that the vehicle can always safely stop given a known constant maximum acceleration, known sensor range and known sensor and mapping rates. Since our vehicle has a power to weight ratio of 3, we assume a conservative maximum acceleration at 6 m/s^2, and assume a worst case sensor and mapping rate of 10 Hz. With a sensor range of 10 meters, this allows for a maximum motion primitive velocity of 10.37 m/s.

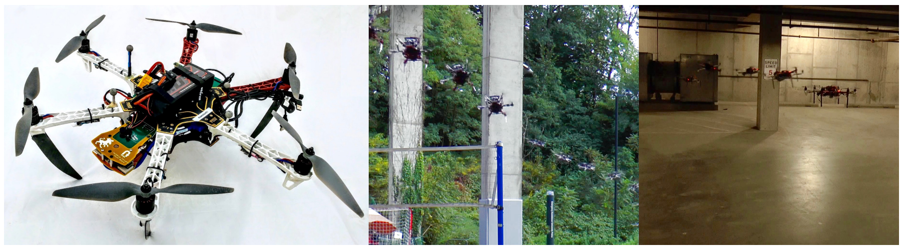


Fig. 3: (a) The hexarotor vehicle used for aggressive flights in (b) outdoor environments and (c) a dimly lit garage. The overlays of the vehicle positions over time depict the trajectories the vehicle took to avoid pillars in its way.

4 Experiments and Results

4.1 Experiments in Aggressive High Speed Flights

We conduct seven aggressive flight experiments in order to test our framework, including four outdoor experiments over an area of 40 m $\times$ 20 m with obstacles over grass (Fig. 3b), and two indoor experiments in a dimly lit garage (Fig. 3c). The experiments are described in Table 2. The local map used for all experiments uses $\alpha_k = 2$ m, $\alpha_s = 0.2$ m, $\beta_s = 0.1$ rad and a voxel size of 0.2 m.

Table 2: Experiment descriptions and parameters.

Experiment	**Description**	$\mathbf{T}$ (s)	$\mathbf{v}_x^{\max}$ (m/s)	$\omega^{\max}$ (rad/s)	r (m)
Outdoor-1	High speed teleoperation outside	2.2	12.0	2.0	N/A
Outdoor-2,3	Aggressive teleoperation with collision avoidance outside	2.0	5.0	1.0	0.8
Indoor-1,2	Aggressive teleoperation with collision avoidance in a dimly lit garage	1.5	3.0	1.5	0.4
Outdoor-4	High speed, aggressive teleoperation with collision avoidance outside	1.3*	7.0	2.0	0.5
Outdoor-5	High speed, aggressive teleoperation with collision avoidance outside	1.5*	10.0	2.0	0.2

$\mathbf{T}$ denotes the duration of the motion primitive, $\mathbf{v}_x^{\max}$ is the maximum desired speed, $\omega^{\max}$ is the maximum yaw rate, and r is the collision radius.
*Motion primitive duration increased adaptively as a linear function of the desired velocity change

High Speed Flight. In **Outdoor-1**, we execute straight line trajectories that hit a maximum desired speed of 12 m/s using the teleoperation system. Figure 4 shows the speed and acceleration attained during three runs. The vehicle achieves a maximum acceleration of 13.5 m/s^2.

High Speed and Aggressive Flight with Collision Avoidance. Experiments **Outdoor-2-5**, and **Indoor-1,2** stress test our collision avoidance algorithm at high speeds while maintaining aggressiveness in the commanded trajectories. In both environments, the operator repeatedly tries to fly the vehicle at the maximum speed, as indicated in Table 2, into an obstacle. Figure 6 shows the speeds and accelerations

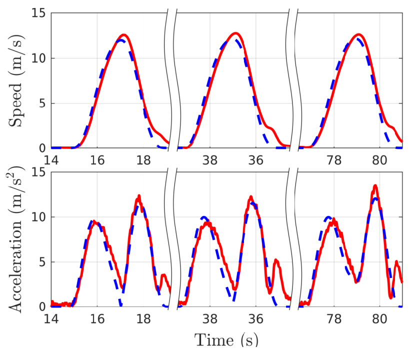


Fig. 4: Desired (dashed blue) vs. estimated (solid red) speed and acceleration achieved during high speed flight in experiment **Outdoor-1**. A maximum acceleration of 13.5 m/s^2 was attained.

attained during the six experiments. The vehicle successfully reaches speeds of 10 m/s and accelerations of 8 m/s^2 in the outdoor environment and speeds of 3 m/s and accelerations of 5 m/s^2 in the indoor environment. Figure 6 also shows regions where the operator's selected trajectory would bring the vehicle closer than r meters to an obstacle. In all such cases, the operator's trajectory is pruned and a collision-free trajectory is selected. Figure 5 shows an example instance of motion primitive pruning to avoid a collision, along with the local map generated by the vehicle.

Computational Performance Analysis. Table 3 shows that our trajectory generation and pruning time is faster than the user input rate (10 Hz) and that our map generation time is faster than the sensor input rate (30 Hz), enabling real time operation. Furthermore, Table 4 indicates that the in-flight computational footprint of the proposed system is less than 75% of the available onboard capacity.

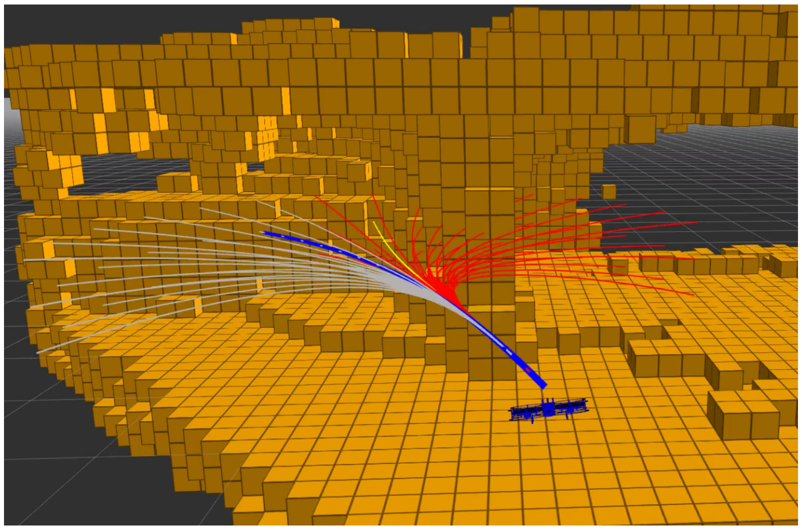


Fig. 5: A snapshot of the map and motion primitive library during an indoor experiment when the user selected trajectory (yellow) is not chosen to avoid a collision.

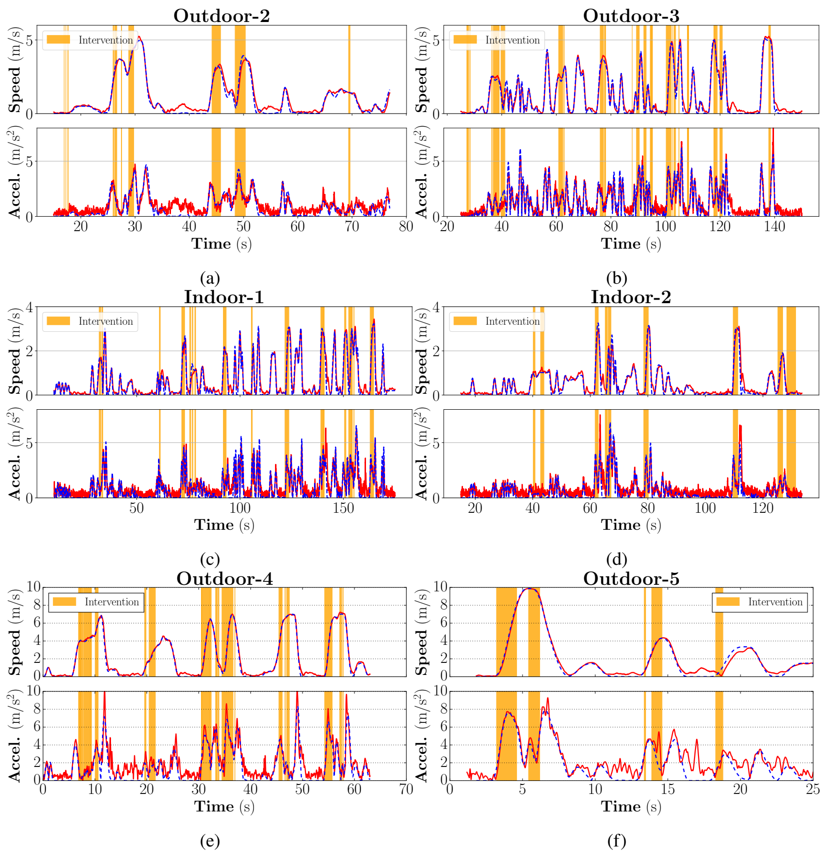


Fig. 6: Desired (dashed blue) vs. estimated (solid red) speed and acceleration achieved during our six collision avoidance experiments. During orange regions, the operator's trajectory was not selected in order to keep a minimum distance to surrounding obstacles. The system accurately tracks trajectory references and avoids obstacles while reaching speeds of up to 10 m/s, and accelerations up to 8 m/s^2.

Table 3: Execution time (ms) and std. dev. per iteration for safe teleoperation during some of our experiments

Experiment	**Trajectory Generation**	**Trajectory Pruning**	**Local Map Generation**
Outdoor-2	0.47 ± 0.072	29.37 ± 24.03	5.72 ± 5.54
Outdoor-3	0.48 ± 0.071	23.65 ± 15.42	5.84 ± 7.95
Indoor-1	0.50 ± 0.125	8.34 ± 5.19	11.67 ± 14.83
Indoor-2	0.51 ± 0.219	15.73 ± 7.23	12.71 ± 16.78

Table 4: CPU usage (%, out of a total available 600%) and std. dev. on a 6 core NVIDIA TX2

Experiment	Safe Teleoperation	VINS Mono	Total
Outdoor-2	32.92 ± 17.59	67.45 ± 10.87	395.72 ± 43.53
Outdoor-3	26.70 ± 19.98	56.19 ± 19.47	359.89 ± 60.87
Indoor-1	57.38 ± 21.25	64.21 ± 3.42	450.40 ± 43.14
Indoor-2	58.76 ± 24.09	63.82 ± 6.41	456.47 ± 36.83

5 Conclusion and Future Work

In this work, we present a system architecture designed for collision-free, agile autonomous flight through cluttered environments. Our hexarotor vehicle is capable of flying at speeds exceeding 12 m/s and with accelerations exceeding 12 m/s^2, as shown in **Outdoor-1**. The teleoperation and collision avoidance system presented has been shown to consistently avoid obstacles while traveling at speeds up to 10 m/s in an outdoor environment (**Outdoor-5**) and in a dimly lit garage (**Indoor-1**, **Indoor-2**).

Although we've attained relatively high levels of performance, there are limitations to the architecture. Currently, motion primitive durations are chosen by the operator. If the motion primitive duration is set too low, the vehicle will not be able to meet the desired acceleration and performance will suffer. Consequently, it is difficult to attain high speeds while remaining responsive at low speeds. A natural extension of the approach is to choose motion primitive durations in a principled manner to ensure that generated references are dynamically feasible. We also believe that chaining together motion primitives to generate more complex reference paths is an exciting avenue for future work. This would enable the vehicle to execute more complicated maneuvers, particularly in cluttered environments.

References

[1] Chen, J., Liu, T., Shen, S.: Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments. In: Proc. of the IEEE Intl. Conf. on Robot. and Autom. Stockholm, Sweden (2016) 2

[2] Delmerico, J., Scaramuzza, D.: A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In: Proc. of the IEEE Intl. Conf. on Robot. and Autom. Brisbane, Australia (2018) 6

[3] Dey, D., Shankar, K.S., Zeng, S., Mehta, R., Agcayazi, M.T., Eriksen, C., Daftry, S., Hebert, M., Bagnell, J.A.: Vision and learning for deliberative monocular cluttered flight. In: Field and Service Robotics, pp. 391–409. Springer (2016) 2

[4] Florence, P., Carter, J., Tedrake, R.: Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps. In: Workshop on the Algo. Found. of Robot. San Francisco, USA (2016) 1

[5] Florence, P.R., Carter, J., Ware, J., Tedrake, R.: Nanomap: Fast, uncertainty-aware proximity queries with lazy search over local 3d data. In: Proc. of the IEEE Intl. Conf. on Robot. and Autom. Brisbane, Australia (2018) 1

[6] Liu, S., Watterson, M., Tang, S., Kumar, V.: High speed navigation for quadrotors with limited onboard sensing. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1484–1491 (2016). DOI 10.1109/ICRA.2016.7487284 6

[7] Lopez, B.T., How, J.P.: Aggressive 3-d collision avoidance for high-speed navigation. In: Proc. of the IEEE Intl. Conf. on Robot. and Autom., pp. 5759–5765. Singapore (2017) 2

[8] Matthies, L., Brockers, R., Kuwata, Y., Weiss, S.: Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In: Proc. of the IEEE Intl. Conf. on Robot. and Autom., pp. 3242–3249. Hong Kong, China (2014) 2

[9] Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 2520–2525. IEEE (2011) 2, 6

[10] Pivtoraiko, M., Nesnas, I.A., Kelly, A.: Autonomous robot navigation using advanced motion primitives. In: Proc. of the IEEE Aerospace Conf., pp. 1–7. Big Sky, USA (2009) 2

[11] Qin, T., Li, P., Shen, S.: Vins-mono: A robust and versatile monocular visual-inertial state estimator. IEEE Trans. Robotics **34**(4), 1004–1020 (2018). DOI 10.1109/TRO.2018.2853729 6

[12] Yang, X., Sreenath, K., Michael, N.: Online Adaptive Teleoperation via Motion Primitives for Mobile Robots. In: Special Issue on Learning for Human-Robot Collaboration, Autonomous Robots '18. Springer (2018) 1, 2