2016 International Conference on
Unmanned Aircraft Systems (ICUAS)
June 7-10, 2016. Arlington, VA USA

Cushioned Extended-Periphery Avoidance: a Reactive Obstacle Avoidance Plugin

James Jackson[1], David Wheeler[2], Tim McLain[3]

***Abstract*— While collision avoidance and flight stability are generally a micro air vehicle's (MAVs) highest priority, many map-based path planning algorithms focus on path optimality, often assuming a static, known environment. For many MAV applications a robust navigation solution requires responding quickly to obstacles in dynamic, tight environments with non-negligible disturbances. This article first outlines the Reactive Obstacle Avoidance Plugin framework as a method for leveraging map-based algorithms while providing low-latency, high-bandwidth response to obstacles. Further, we propose and demonstrate the effectiveness of the Cushioned Extended-Periphery Avoidance (CEPA) algorithm. By representing recent laser scans in the current body-fixed polar coordinate frame, a 360° lower-bound understanding of the environment is available. With this extended field of view, motion assumptions common in other reactive planners can be relaxed and emergency control effort can be applied in any direction. CEPA is validated in simulation and on hardware in a GPS-denied environment using strictly onboard computation and sensing.**

I. Introduction

As technological advancements push to meet the size, weight, and power (SWAP) constraints imposed by micro air vehicles (MAVs), exciting applications become possible. Unfortunately the sophistication of estimation and control laws do not yet meet the safety, reliability, and robustness required for full integration into society. One open field of research is autonomous multirotor flight in unknown, dynamic, tightly confined, and cluttered environments.

As illustrated in Figure 1, path planning and obstacle avoidance algorithms generally address three objectives: avoiding collisions, facilitating stable flight, and accomplishing a mission or goal. This field of research is well developed, particularly in the context of ground robots. Because a ground robot can generally pause as needed, often the literature assumes a static, known environment. Further, due to the slow, stable dynamics of ground vehicles, disturbances, like wind will rarely induce collisions. These factors, in conjunction with less restrictive weight and computational power constraints, motivate the literature's primary emphasis on the optimal, or at times suboptimal, accomplishment of goals with respect to some specific cost function (item 3 in Figure 1).

Generally a global map, represented in a Cartesian coordinate frame, is provided to the path planner. This map comes from a priori data or from fusing sensor information using Simultaneous Localization and Mapping (SLAM) techniques. For example a 2-D obstacle map can be created as a series of body-fixed, polar laser scans are transformed into a global, Cartesian coordinate frame and fused based on sensor and state uncertainty estimates [1].

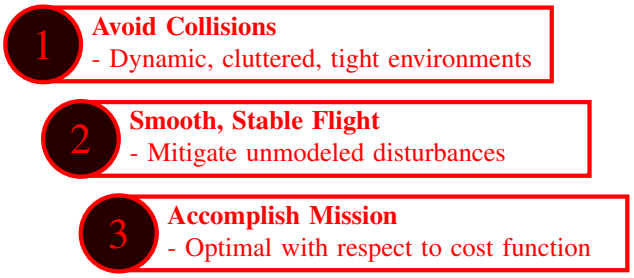

Fig. 1. MAV priorities in general. Avoiding collisions, even when they violate environment assumptions, is of paramount importance. Of secondary importance is smooth, stable flight, mitigating destabilizing disturbances. Accomplishing the desired mission should generally not come at the expense of items 1 and 2.

Given a map, obstacle-free paths are found through the environment using one of several methods. Potential field methods create artificial forces away from obstacles and towards goals [2]. These methods are generally simple and quick to calculate, but suffer from local minima and cannot guarantee obstacle avoidance. The Probability Road Map (PRM) can be used to randomly generate waypoints connecting the agent with the goal in a manner to avoid obstacles [3], but is designed for use by holonomic agents. Rapidly-Exploring Random Trees (RRT), a modification of PRM uses a similar obstacle-free waypoint path planning technique, while taking into account kinematic constraints of the vehicle. More robust algorithms such as D* Lite [4], can be used to heuristically find the shortest path to the goal through the environment.

While derivatives of these approaches have proven to be effective at fusing sensor measurements and calculating safe paths through the environment, they can incur significant computational, memory, and sensing requirements, and often assume the agent is unaffected by disturbances while safe paths are calculated. While these assumptions may be valid for ground robots and MAVs flying in spacious environments, this problem can become difficult to solve quickly enough to effectively react to large disturbances and errors in environment estimation during autonomous flight in tight quarters.

As an alternative to map-based planning, some simple

[1]James Jackson is with the Department of Mechanical Engineering, Brigham Young University jamesjackson@byu.edu
[2]David Wheeler is with the Department of Electrical Engineering, Brigham Young University david.wheeler@byu.edu
[3]Tim McLain is with the faculty of Mechanical Engineering, Brigham Young University mclain@byu.edu

978-1-4673-9333-1/16/$31.00 ©2016 IEEE

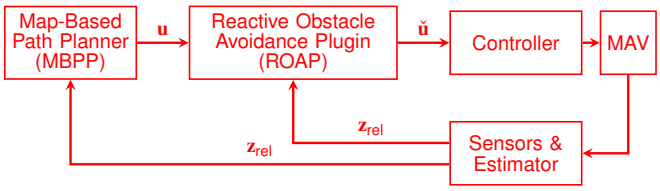

Fig. 2. Block diagram illustrating how ROAP supplements an existing path planner by modifying commands. The inner control loop rate matches the sensor rate with minimal latency, thereby improving robustness in dynamic, cluttered, and tight environments with non-negligible unmodeled disturbances.

and efficient algorithms use the concept of optical flow to demonstrate effective corridor-centering [5] and obstacle avoidance [6]. Other, more sophisticated methods use this type of data combined with other monocular features to train agents to avoid obstacles based on input data generated by an expert pilot [7]. These methods have also been demonstrated to be effective in avoiding obstacles during MAV operation but require consistent forward motion to generate meaningful features required by the controller.

In response, we outline the Reactive Obstacle Avoidance Plugin (ROAP) framework in Section II and propose a new reactive algorithm, Cushioned Extended-Periphery Avoidance (CEPA) in Section III as a specific implementation of this framework. We present simulation and hardware results of CEPA and the ROAP architecture in Section IV and conclude in Section V.

II. ROAP Motivation

In the ROAP framework, a high-level planner uses any map-based approach to plan smooth paths through a known environment while a reactive obstacle avoidance algorithm is implemented underneath to recover from disturbances or estimation errors, as illustrated in Figure 2. In this way, an efficient reactive obstacle avoidance algorithm can match the rate of the sensor with minimal latency, improving robustness in dynamic, cluttered, and tight environments with non-negligible disturbances. This provides the high-level path planner the time to account for changes in the environment, such as a recently closed door or moved obstacle, and plan an alternative feasible path. While a reactive obstacle avoidance plugin may cause the path to become suboptimal in a precarious environment, it requires much less in terms of computational and sensor capabilities, and is effective in real-life testing [8]–[11].

Clearly, in this configuration, a reactive obstacle avoidance may take action that prevents the completion of a global mission but ensures that the MAV does not damage itself or the environment. This concept parallels the MAVs priorities, illustrated in Figure 1, where in general, avoiding collisions and maintaining stable flight is of paramount importance. This is particularly relevant in environments when sensors perform poorly, such as during GPS-degradation or in featureless scenes, and in the presence of disturbances, such as wind or ground and wall effect.

For a ROAP implementation to be robust, the algorithm must exhibit the following properties:

1) Fast response, i.e. low latency, high bandwidth.
2) Independent of a priori or outdated information.
3) Limited memory/computation requirements.
4) No motion assumptions (e.g., constant motion, only forward motion).
5) Safe commands despite erroneous, outdated, or absent high-level goals.

Scherer et al. were first to propose a ROAP algorithm in their paper *Flying Fast and Low Among Obstacles* (FFLAO) [8] and demonstrated impressive hardware results using a laser scanner. While accounting for the first three properties by responding quickly to the most recent obstacle information, FFLAO constrains the MAV to move only in the direction of the sensor, limiting the MAV to forward and yawing motion alone. While this assumption works under ideal conditions, we have found that this assumption makes safe navigation difficult in tight environments or in the presence of infeasible goals where hovering, reversing and lateral motion are often necessary.

Since FFLAO, Oleynikova et al. has presented a compelling ROAP implementation using stereo vision [9], stressing the importance of low computation requirements. Schopferer et al. has presented a novel decoupled iterative planning method [12] that achieves near-optimal reactive avoidance under computational limitations by considering the kinematic feasibility of planned trajectories. Hrabar presented a method that blurs the line between reactive and map-based obstacle avoidance [11] by keeping a local memory of the environment in the form of a 3D voxel grid and searching for a feasible path using PRM. While the ability to hover is added in this method, it focuses primarily on extending the field-of-view of the sensor, rather than extending the possible maneuvers of the MAV to include lateral and reverse motion. While these methods are all accompanied by impressive results, they are subject to most or all of the same motion constraints found in FFLAO. To address this concern, we present the Cushioned Extended-Periphery Avoidance (CEPA) algorithm, which extends these previous methods to allow for safe operation of MAVs in tightly constrained environments in the presence of infeasible goals and non-negligible disturbances.

III. CEPA Algorithm Description

The algorithm addresses two main issues related to safe autonomous MAV operation:

1) Guide the MAV around obstacles towards waypoints chosen by the high-level planner.
2) Apply additional control in emergency situations if the MAV comes too close to an obstacle.

Typical path planning approaches use a Cartesian coordinate or graph-based system, either iterating through each coordinate or node to form a cost map [13], [14]. CEPA, like FFLAO, performs planning in the polar, body-fixed, sensor frame of the laser scanner. Further, CEPA analytically inflates

the proposed path in polar coordinates. As a result, the path can be verified for obstacles by a simple differencing in the polar domain. These two features reduce computational load and algorithm latency.

To remove limiting motion assumptions, CEPA efficiently fuses recent laser scans to create a lower-bound, 360° sensor view. Like [11], this approach blurs the line between a purely reactive avoidance method and a map-based method, which could potentially reduce the reactive nature of the algorithm. However, without a 360° sensor or some level of local memory, necessary lateral or reverse movement cannot be executed safely. A small amount of local memory provides some of the environmental awareness of a map-based planner while maintaining the responsiveness of a reactive planner. CEPA expects velocity commands from a high-level planner and then outputs modified velocity commands, as needed, given input from the most recent laser scans, as shown in Figure 2. With this architecture, CEPA can be paired with any high-level path planner which outputs body-fixed velocity commands without modification.

CEPA is derived in two dimensions primarily due to the sensing capabilities of traditional laser scanners. This assumes relatively planar motion in a structured environment, which is often the case for indoor operation of MAVs. To extend CEPA to 3D operations, CEPA could either be layered in cylindrical coordinates or performed entirely in spherical coordinates. Because CEPA leverages the computational benefit of operating directly in the sensor frame, the choice of 3D coordinates should likely mimic the coordinates of the 3D sensor.

A. Steering Algorithm

The steering algorithm is designed to choose commands that are most like the commands provided by the high-level path planner, but that also safely avoids obstacles. To accomplish this, CEPA computes a cost function which balances modification of an incoming command with proximity to observed obstacles.

First, a suitable path must be in approximately the same direction and approximately the same size as the incoming command when feasible. This can formulated by maximizing the weighted sum of the inner product and the relative size of the goal vector $\mathbf{v}$ and the outgoing command $\check{\mathbf{v}}$, expressed by

$$k_1\left(\mathbf{v}^\top\check{\mathbf{v}}\right)+k_2\frac{\|\check{\mathbf{v}}\|}{\|\mathbf{v}\|}. \tag{1}$$

Secondly, the degree of interference for the proposed command is calculated by projecting two elongated safety cushions onto the polar map, with fixed look-ahead time T. As illustrated in Figure 3, a lower-bound safety cushion of radius r_{LB} defines the minimum required separation distance for a feasible path. An upper-bound safety cushion of radius r_{UB} defines where obstacles begin to influence commands. A safety cushion for a given radius r at specified bearing angle

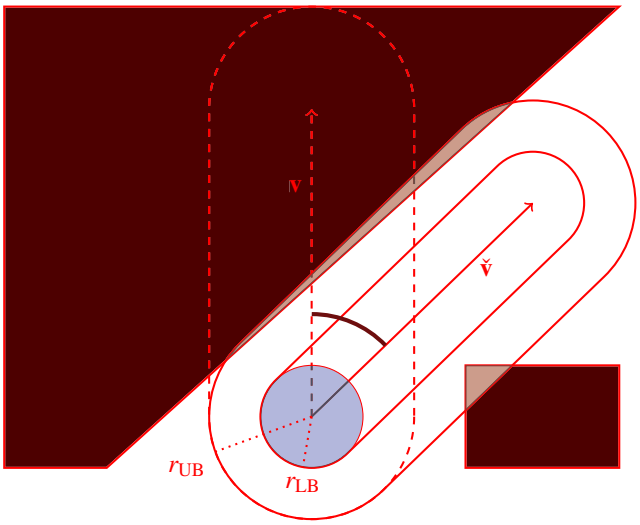

Fig. 3. An example steering configuration. $\mathbf{v}$ is the obstacle-laden goal vector supplied by the path planner. CEPA identifies $\check{\mathbf{v}}$ as the minimum-cost, collision-free command and passes it to the controller. The heading discrepancy and the obstacle intrusion into the outer safety cushion induce costs shown in red. The proposed path is deemed feasible because the inner safety cushion is not penetrated. While the figure illustrates a Cartesian representation, CEPA works in the sensor's polar coordinate frame.

ϕ is defined analytically as

$$SC_r(\phi,\check{\mathbf{v}})=\begin{cases} r\csc\phi & \phi\in[\gamma,\frac{\pi}{2}) \\ r & \phi\in[\frac{\pi}{2},\frac{3\pi}{2}] \\ -r\csc\phi & \phi\in(\frac{3\pi}{2},2\pi-\gamma) \\ d\cos\phi+\sqrt{r^2-d^2\sin^2\phi} & \phi\in[2\pi-\gamma,\gamma) \end{cases}, \tag{2}$$

where $d=\|\check{\mathbf{v}}\|T$ is the look-ahead distance and $\gamma=\mathrm{atan2}(d,r)$. Note that Equation 2 assumes $\check{\mathbf{v}}$ is directed towards $\phi=0$. Rotating the safety cushion is as simple as shifting the indices of the polar array containing the N returned range measurements.

The lower-bound safety cushion, SC_{LB}, is an estimate of the space the MAV will occupy during the execution of the command for the look-ahead time T. Any conflict with this inner cushion renders the proposed command invalid. The larger cushion, SC_{UB}, acts as a buffer region that may become occupied during the execution of a valid command, but during general operation should remain free. Like a deformable ball, the proposed path will respond to minimize intrusions, guiding the MAV away from obstacles. The extent of the intrusion is found by differencing the safety cushion and laser scan at each angle $LS(\phi_i)$, after masking the array to only regard potential conflicts. A discrete integral can then be used to model the amount of intrusion into the safety bubble for a potential command given a recent laser scan

$$\Omega\left(\check{\mathbf{v}}|LS\right)=\sum_i^N\kappa\left(\phi_i|\check{\mathbf{v}},LS\right), \tag{3}$$

where

$$\kappa(\phi_i|\check{\mathbf{v}}, LS) = \begin{cases} \infty & LS(\phi_i) \in [0, SC_{\text{LB}}(\phi_i)] \\ f(SC_{\text{UB}}(\phi_i) - LS(\phi_i)) & LS(\phi_i) \in (SC_{\text{LB}}(\phi_i), SC_{\text{UB}}(\phi_i)) \\ 0 & LS(\phi_i) \in [SC_{\text{UB}}(\phi_i), \infty) \end{cases} \tag{4}$$

and $f(x)$ is any positive definite function for $x > 0$. In our implementation, $f(x) = x^2$.

A weighted sum of Equations 1 and 3 forms a cost function whose minimum is the command which is passed to the controller. Using a polar coordinate frame simplifies the cost function sufficiently that even a brute-force method is capable of solving the optimization as fast as the incoming laser scan measurements, typically 10 to 40 Hz:

$$\check{\mathbf{v}}^* = \underset{\check{\mathbf{v}}}{\operatorname{argmin}} \left[k_3 \Omega(\check{\mathbf{v}}) - k_1 \left(\mathbf{v}^\top \check{\mathbf{v}} \right) - k_2 \frac{\|\check{\mathbf{v}}\|}{\|\mathbf{v}\|} \right]. \tag{5}$$

The relative size of gains k_1, k_2, and k_3 can be adjusted for required performance. If k_3 is chosen to be larger than k_1 and k_2, CEPA will prefer to deviate from the planned path to ensure safety. A large k_3 makes the safety cushion inelastic, responding rigidly to approaching obstacles, while a smaller value will provide a softer response. The relative size of k_1 and k_2 will determine how CEPA responds to path deviations. If k_1 is larger than k_2, then CEPA will prefer changing direction to slowing down and vice-versa.

B. Map Memory

Applying a command in a direction that is not currently observed is inherently presumptuous. Previous ROAP algorithms [8]–[10] assume that it is always possible to find a viable path while maintaining forward motion. It is not uncommon, however, that a MAV needs to move in a direction in which it is not receiving measurements, such as overshooting a position goal or counteracting a disturbance propelling the vehicle forward. While it is possible to perform large yawing motions to always look in the direction of motion, the control delay makes rejecting disturbances in tight environments impossible.

As an alternative to colliding, some measure of memory must be integrated to ensure that the MAV does not move into objects that it has seen previously, but cannot currently observe with its sensor. This can be done by extending the vehicle's peripheral vision. The reactive planner should not, however, provide a full-resolution map of the explored environment due to computational constraints, but enough to ensure safe navigation.

To do this, some number of previous laser scans and the estimates of the relative transform between each, are saved as a queue in the reactive avoidance memory. In the event that backward motion is necessary, previous laser scans are transformed to be with respect to the current body frame, augmenting the current sensor measurement. If the MAV has moved forward recently, then the concatenation of even two 180 degree laser scans provide some 360° understanding of the environment, as illustrated in Figure 4. With this information, the MAV can more confidently execute commands which are not directly in the field of view.

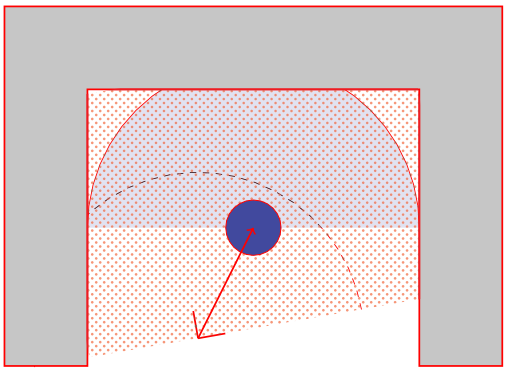

Fig. 4. A visual description of the way memory is kept in the reactive planner. Although the MAV can only observe obstacles in the direction of the current 180° laser scan (blue-solid), appending previous laser scans gives the MAV a limited 360° understanding about the entire shaded area and allows the MAV to safely move backwards

This approach does not extend the field of view of the sensor, but rather assumes, (1) an object has not recently approached the MAV from the rear, and (2) accurate transform estimates are available. For a more conservative memory estimate, the covariance of the transforms can be used to provide the $n\sigma$ worst-case transform. Further, these covariances can be set to grow with time, shrinking the assumed distances to obstacles in the rear 180 degrees. This results in more conservative navigation, but also is more taxing on the processor during memory updates.

C. Emergency Avoidance

In some cases, a disturbance may cause an obstacle to penetrate the MAVs lower-bound safety threshold r_{LB}. In keeping with the proposed priorities presented in Figure 1, the command provided by the map-based path planner is temporarily ignored as emergency action is taken.

As illustrated in Figure 5, the periphery-enhanced 360-degree obstacle map is filtered such that

$$\frac{d\rho}{d\phi} \leq K$$

where K represents the maximum-allowable slope in polar coordinates. For each obstacle detected within r_{LB} a small avoidance vector is formed pointing towards the MAV, proportional to the extent of the intrusion. The summation of these small vectors forms the final command $\check{\mathbf{v}}$. Filtering is critical to ensure that small obstacles are not overpowered by large obstacles in the map. Both small and large obstacles produce commands on similar orders of magnitude given they intrude the same amount into the cushion. In this way, the cushion models the physical response of a deformable ball. With a 360° understanding provided by the map memory, this command can be executed with some level of confidence in any direction.

IV. Experimentation and Results

CEPA was implemented in ROS [15] and tested in a Gazebo simulator, adapted from [16], and on a hexacopter platform. The simulation parameters paralleled the hardware

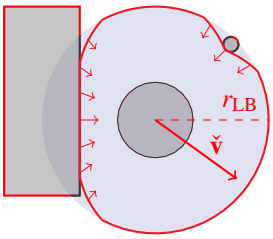

Fig. 5. Illustration of emergency avoidance. The red line represents the 360° filtered obstacle map when $K = 0.01$. The summation of the individual red avoidance vectors forms the final command $\check{\mathbf{v}}$.

(3.81 kg, 1.0 m outer diameter). A 40 Hz Hokuyo UTM-30LX laser range finder with a 30 meter range and 180 degree field of view was used for obstacle detection and modeled in the simulator.

A PID velocity controller, using the multirotor model-inversion technique presented in [17] was used to control the system. Yaw was controlled with an under-damped proportional controller, causing the laser scanner to generally be oriented in the direction of commanded motion. The following CEPA gains were used: $k_1 = 1$, $k_2 = 1$, $k_3 = 4$, $T = 4$ s, $K = 0.01$, $r_{LB} = 0.55$ m, $r_{UB} = 1.0$ m, and $f(x) = x^2$.

During each simulation experiment, wind was modeled as a succession of applied forces with a normally distributed magnitude, $\mathcal{N}(1N, 0.5N^2)$, and uniformly distributed direction. Wind magnitude and direction were recalculated according to a Poisson process with $\frac{1}{\lambda} = 10$ seconds. These wind model parameters were selected to mimic the significant wall effect that large multirotors experience in tight environments.

FFLAO, defined in [8] was also implemented in 2D for comparison. It was implemented with gains $k_g = 10.5$, $k_o = 0.8$, $c_1 = 1.0$, $c_2 = 0.25$, $c_3 = 1.0$ and $c_4 = 1.0$. It should be noted that this algorithm has demonstrated success in more than 700 flight tests and at speeds exceeding 10 m/s, but due to motion assumptions and constraints it is not designed for operation in tightly confined environments with non-negligible disturbances. It was implemented as a comparison to motivate the relaxation of motion constraints necessary in these types of environments.

A. Simulation Results

Two tightly-constrained environments were used to validate the algorithm. The first environment, shown in Figure 6 consists of a dense grid of cylinders requiring tight maneuvering. While the high-level path planner commands the MAV directly towards the goal, each respective ROAP algorithm modifies the commands to autonomously navigate through the environment. Each algorithm was tested 1500 times. The supplied high-level command had a magnitude between 1.0 m/s and 5.0 m/s and was directed towards the goal. However, regardless of the commanded magnitude, as the multirotor entered the cluttered environment, both CEPA and FFLOA reduced the outgoing command to close to 0.8 m/s to maintain safe flight throughout the course.

TABLE I
TABLE OF SIMULATION RESULTS FOR SIMULATION SCENARIO 1

	FFLAO	CEPA
Completion Rate	0.2188	0.9863
Average Duration (s)	71.61	63.54

The collision-free success rate and average flight duration of successful flights taken for the MAV to autonomously navigate safely through the several environments and reach its goal are recorded in Table I.

As can be seen from Table I, placing a constraint on lateral velocity causes performance to suffer in our tightly-confined environment with non-negligible disturbances. This is largely because when moving through such a tightly-confined environment, forward velocity, u, must be kept low. This gives opportunity for disturbances to induce non-negligible lateral velocity which must be corrected in order to avoid collisions. With a constraint on lateral velocity, the MAV is much slower at correcting these errors because it must induce large yawing motions, and therefore is unable to fly safely. CEPA, on the other hand, is able to handle these disturbances because of its ability to move the MAV in any direction to avoid collisions.

The second environment simulates the scenario where a high-level path planner commands an infeasible goal and the obstacle avoidance must prevent the MAV from crashing until a proper goal is received. Specifically, we explored the scenario when a goal is placed on the far side of recently closed door, as shown in Figure 7. After recognizing the obstruction, the avoidance algorithm was required to correct the commands for 30 seconds until an alternative route was provided. This second scenario was tested 50 times. In each trial, the CEPA algorithm enabled the MAV to successfully pause at the door, accounting for all disturbances while waiting for an updated plan. FFLAO, however, was never able to complete the task because its imposed motion constraints disallowed backward motion. As the MAV approached the closed door, it correctly stopped forward motion, but was unable to correct for any disturbance.

The average latency of CEPA was 2.9 ms with a standard deviation of 1.6 ms. Calculations were easily available at the laser scanner's bandwidth of 40 Hz even using a brute-force optimization method.

B. Hardware Results

To definitively understand its effectiveness, CEPA was exercised in hardware. Flight test computation was performed using an onboard Intel i7 computer with a 2.4 GHz quad core processor and 16 GB of RAM. To emphasize the light-weight nature of CEPA, avoidance was restricted to use less than 1/16 of the available processing time. State estimation was performed using the Relative Multiplicative Extended Kalman Filter described in [18] provided with position measurements from an RGB-D visual odometry algorithm described in [19]. No external positioning system or off-board processing was required.

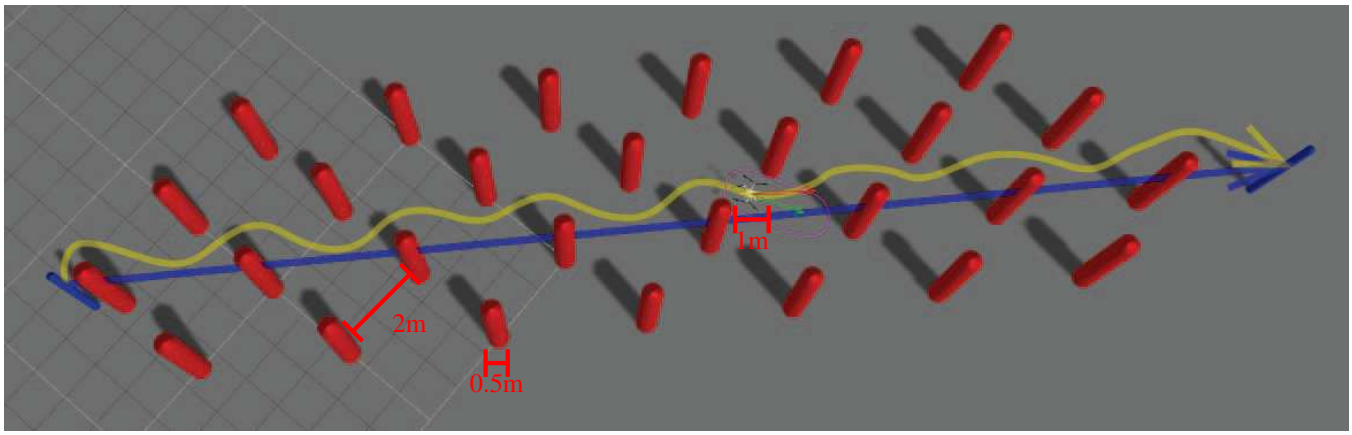

Fig. 6. Scenario 1: A grid of densely positioned cylinders obstruct the MAV's path between the start and goal positions represented as blue pillars. The high-level path planner commanded a 1m/s velocity directly towards the goal at all times during the test. The blue line is the original infeasible path planned by the high level path planner, while the yellow line is the path ultimately taken by the MAV as a result of CEPA intervention. The red arrow is the current high-level command. The green arrow is the modified CEPA command with the magenta safety cushion shown.

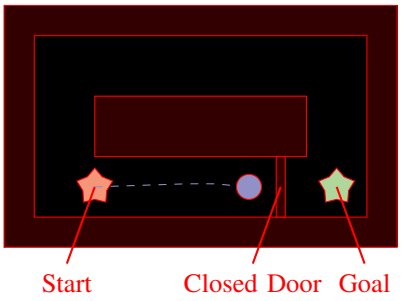

Fig. 7. Scenario 2: The high-level path planner commands an infeasible path due to a recent environment change. The ROAP block must maintain safety while a new path is planned.

The MAV was placed in scenarios which isolated three particular challenges:

1) Selecting an appropriate path around several obstacles.
2) Taking action to avoid a previously observed obstacle when is no longer in the field of view.
3) Preventing collision when provided and infeasible goal.

Challenges 1 and 2 were addressed in the first scenario, where the MAV was placed in a wide hallway with two large obstacles in the middle, as shown in Figure 8. The high-level path planner continuously provided commands at 0.8 m/s directly towards to the goal, while CEPA correctly chose a safe path around the obstacles and arrived at the goal. During this flight, after navigating around the first obstacle, estimation errors and disturbances caused the MAV to be pushed backwards towards the first obstacle. Although the MAV was oriented towards the goal, and could no longer directly see the first obstacle, it responded correctly by commanding additional control away from the unseen obstacle behind it. After avoiding the first obstacle, the MAV then navigated around the second obstacle and to the goal without further issues. During the test, the MAV maintained a distance of at least 0.1 m from any obstacle, successfully completing the task with no user input.

Fig. 8. Hardware validation of CEPA in a GPS-denied environment using strictly onboard computation and sensing.

In the second scenario, the high-level path planner commanded the MAV directly through a flat wall for 5 seconds, very much like the closed-door simulations performed previously. In this demonstration, however, there was no feasible way to reach the goal. During this test, the MAV reached a minimum distance of 0.1 m from the wall, and after some damped oscillatory movement, hovered stably 0.5 m from the wall. Videos of the simulation and hardware demonstrations are available at `https://youtu.be/35Og9PYwXOI`.

V. Conclusions

We have outlined the Reactive Obstacle Avoidance Plugin framework, which allows for high-bandwidth, low-latency control corrections to improve MAV robustness. This method allows SWAP constrained MAVs to robustly leverage map-based path planners, generally designed for ground robots in static, known environments, while mitigating disturbances and avoiding collisions. To demonstrate the effectiveness of this framework, we have presented the Cushioned Extended-Periphery Avoidance algorithm. CEPA relaxes motion assumptions common in other reactive path planners, allowing for more confident control in tight environments with non-negligible disturbances. By working in the laser scanner's polar coordinate frame, and by incorporating previous laser scans, safe controls can be efficiently computed despite erroneous, outdated, or even absent high-level goals.

Future work includes improving the safety cushion lookahead window by incorporating the MAV's dynamics (e.g., momentum) and allowing trajectory based inputs as well as extending CEPA to three dimensions. Developing a fast, camera-based ROAP algorithm without limiting motion assumption remains an open problem. Current work also includes more extensive hardware testing, especially in the presence of moving obstacles.

Acknowledgements

This research was supported by the NSF Center for Unmanned Aircraft Systems (C-UAS), and Brigham Young University.

References

[1] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, Feb 2007.

[2] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," 1991.

[3] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566 – 580, 1996.

[4] S. Koenig and M. Likhachev, "D* Lite," *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pp. 476–483, 2002.

[5] C. McCarthy and N. Barnes, "Performance of optical flow techniques for indoor navigation with a mobile robot," *IEEE International Conference on Robotics and Automation*, vol. 2, no. April, pp. 5093–5098, 2004.

[6] J. R. Deming and S. Bruder, "Obstacle avoidance using image flow in an RT-Linux environment in a PC-104 platform," *Machine Learning and Applications, 2004. Proceedings. 2004 International Conference on*, pp. 215–219, 2004.

[7] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive UAV control in cluttered natural environments," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1765–1772, 2013.

[8] S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli, "Flying fast and low among obstacles," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2023–2029, 2007.

[9] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive Avoidance Using Embedded Stereo Vision for MAV Flight," 2015.

[10] J. Saunders, R. Beard, and J. Byrne, "Vision-based Reactive Multiple Obstacle Avoidance for Micro Air Vehicles," *2009 American Control Conference, Vols 1-9*, pp. 5253–5258, 2009.

[11] S. Hrabar, "Reactive obstacle avoidance for rotorcraft UAVs," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4967–4974, 2011.

[12] S. Schopferer and F. M. Adolf, "Rapid trajectory time reduction for unmanned rotorcraft navigating in unknown terrain," in *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*, pp. 305–316, 2014.

[13] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[14] S. Scherer, D. Ferguson, and S. Singh, "Efficient C-space and cost function updates in 3D for unmanned aerial vehicles," *2009 IEEE International Conference on Robotics and Automation*, pp. 2049–2054, 2009.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," *ICRA*, vol. 3, no. Figure 1, p. 5, 2009.

[16] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. Cham: Springer International Publishing, 2016.

[17] J. Ferrin, R. Leishman, R. Beard, and T. McLain, "Differential flatness based control of a rotorcraft for aggressive maneuvers," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 2688–2693, Sept 2011.

[18] R. C. Leishman and T. W. McLain, "Multiplicative Extended Kalman Filter for Relative Rotorcraft Navigation," *Journal of Aerospace Information Systems*, pp. 1–17, 2014.

[19] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4973–4980, 2014.