2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

AeriaLPiPS: A Local Planner for Aerial Vehicles with Geometric Collision Checking

Justin S. Smith[1] and Patricio Vela[1]

***Abstract*— Real-time navigation in non-trivial environments by micro aerial vehicles (MAVs) predominantly relies on modelling the MAV with idealized geometry, such as a sphere. Simplified, conservative representations increase the likelihood of a planner failing to identify valid paths. That likelihood increases the more a robot's geometry differs from the idealized version. Few current approaches consider these situations; we are unaware of any that do so using perception space representations. This work introduces the *egocan*, a perception space obstacle representation using line-of-sight free space estimates, and *3DGap*, a perception space approach to gap finding for identifying goal-directed, collision-free directions of travel through 3D space. Both are integrated, with real-time considerations in mind, to define a local planner module of a hierarchical navigation system. The result is *Aerial Local Planning in Perception Space (AeriaLPiPS)*. AeriaLPiPS is shown to be capable of safely navigating a MAV with non-idealized geometry through various environments, including those impassable by traditional real-time approaches. The open source implementation of this work is available at *github.com/ivaROS/AeriaLPiPS*.**

I. Introduction

Most traditional MAV navigation planners use simplified robot geometry models due to computational constraints. The most common idealized model used, a sphere, is simple and computationally efficient because collision checking reduces to point queries (in inflated environments). The underlying approaches then rely on point-based, organized data structures such as k-d trees [1], Octomaps [2], [3], and linear octrees [4]. Collision checking is efficient but comes with increased up front compute costs to build the data structures. The costs do not scale well with sensor resolution [5]. Additionally, the process of extracting collision-free directions of travel (gaps) from 3D point-based representations in real time is not trivial. Using perception space to encode the local geometry [5] provides better computational scaling and has real-time performance properties. Efficient collision checking for MAV trajectory planning has been shown using perception spaces such as depth images [6] and egocylindrical disparity images [7], though these approaches still model the robot as a point.

When the robot volume lies strictly within the larger idealized volume, collision checking will be conservative due to false positives: feasible trajectories are wrongly assessed to be in collision, as in Fig. 1. Ellipsoids [8], [9] or tight bounding boxes [10] reduce volumetric mismatch but do not remove it. Considering the planar projected footprint of the robot during planning [11] improves the ability of a robot with non-ideal geometry to find a feasible, safe path through cluttered environments. The trade-off is slower planning rates or slower local map update rates. Similarly, a flight corridor defined by a convex polyhedron series allows geometry such as tilted cuboids to be used [12] but requires a fully known environment and significant processing to generate.

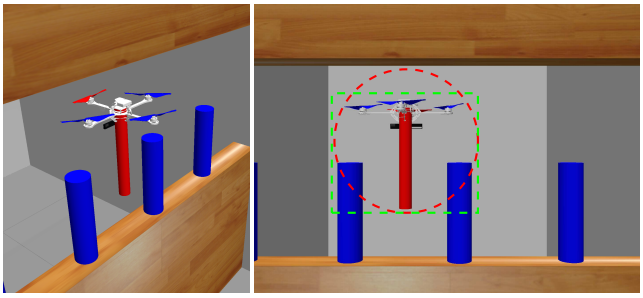


Fig. 1. This representative MAV with non-idealized geometry can easily pass through the shown obstacles. However, common idealizations of its geometry as a sphere (shown in red) or a bounding box (green) do not. A navigational planner using overly conservative idealized geometry would be unable to plot a trajectory through these obstacles.

Prior work [5] found the egocylindrical perception space representation [7] to exhibit more efficient collision checking with accurate robot geometry than traditional approaches up to a certain quantity of checks. Efficiency gains were due to the reduced pre-processing requirements of perception space and the linear scaling of collision checking. Though planning was for $SE(2)$, the same prior work [5] demonstrated the value of using fast-but-liberal representations along with slow-but-accurate representations. Additional perception space research [13], [14] established that gap-based processing was critical to efficient, safe planning in $SE(2)$. *This work extends these concepts to $SE(3)$ for efficient, non-conservative navigation of robots with non-ideal geometry in cluttered 3D environments.*

One benefit of operating in perception space is that processing occurs in a representation of lower dimension than the geometric model. 2D image representations for collision checking in 3D space can leverage fast image processing methods to interpret the local free-space. Not only does the same idea apply here, but this paper shows that gap estimation also benefits from image-based processing. Gaps, as robot-environment navigation affordances, are critical to rapidly establishing candidate local navigation goals in (line-of-sight) free-space. Local trajectory processing in a 2D representation saves time for dimension-sensitive algorithms,

*This work supported in part by NSF Award #1849333.

[1]J.S. Smith and P.A. Vela are with the School of Electrical and Computer Engineering and the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30308, USA. {jssmith, pvela}@gatech.edu

much as processing for planar paths in a 1D perception-space representation is faster than the 2D equivalent [13].

Organization. Section II introduces the egocan perception space representation along with a fast gap discovery approach for identifying candidate local waypoints. Section III describes the process of collision checking non-idealized geometry in egocan space and how *collision filtering* leverages liberally and conservatively inflated models to accelerate the process. Section IV presents a navigation system utilizing the proposed strategies along with simulated experiments. Finally, Section V summarizes the findings and discusses future research directions.

II. Gap-based Waypoint Sampling in Perception Space

Typical point-based local planners exploit fast collision queries to sample many trajectories or poses per measurement in hopes of densely sampling free-space for a feasible trajectory. Given that more accurate geometric collision checkers cannot do so, the alternative is to focus on rapidly identifying a sparse set of candidate terminal points and testing synthesized trajectories going to them. Gaps have proven to be an effective means for doing so due to their favorable topological properties [13], [14]. This section introduces the egocan 2D free-space representation and shows how it can be used to identify gaps in the local environment from which to sample candidate waypoints.

A. Defining the Egocan

The egocan extends the range-based *egocylinder* [5] with upper and lower planes to create a closed surface around the MAV for encoding the local map. It is a perception space obstacle representation suitable for aerial robots.

An egocylinder [7] is a virtual cylinder with unit radius and height α centered at the origin and aligned with the y-axis since it uses camera frame geometry (z-axis ahead, x-axis right). Conceptually, it is equivalent to a volumetric LIDAR scan in that points are projected along rays to the cylinder. The projective nature is important since it maintains a 1-1, line-of-sight relationship between the camera origin and the nearest occlusion point in the world around the robot (up to the domain of the cylinder). For data storage, the surface of the cylinder is a range image I_c with k_v rows, k_h columns, and circular symmetry for the columns.

Let P be an obstacle point in the egocylinder frame. The range $\rho = \|P\|$ value will be placed at pixel coordinates p in I_c, where

$$p = \begin{bmatrix} C_h & 0 \\ 0 & C_v \end{bmatrix} \begin{bmatrix} \text{atan2}(P^1, P^3) \\ P^2/\rho \end{bmatrix} + \begin{bmatrix} k_h/2 \\ k_v/2 \end{bmatrix}, \quad (1)$$

$C_h = k_h/(2*\pi)$, $C_v = k_v/\alpha$, and α is related to the vertical field of view. The reverse operation, mapping the pixel data p in I_c out into the world is

$$P = \rho \begin{bmatrix} \sin\theta \\ (p^2 - k_v/2)/C_h \\ \cos\theta \end{bmatrix} \quad \text{for } \theta = \frac{p^1 - k_h/2}{C_h}, \rho = I_c[p]. \quad (2)$$

As an egocentric representation, the values in I_c must update as the robot moves. Propogation involves transforming the egocylinder data forward by the cylinder frame's relative motion $g^{t_i}_{t_{i+1}}$ from time t_i to t_{i+1}. Doing so involves equations (2) and (1). Updated points that no longer project onto the cylinder are lost. As such, the egocylinder has blind spots above and below, a serious limitation for a flying robot. Fig. 2(b) depicts the egocylinder. Based on its positioning, the robot's egocylinder does not capture the hydrant. The egocan removes the egocylinder's blind spots by closing the surface of the cylinder through the addition of caps, or *egocaps*, to the ends of the cylinder. The upper and lower egocaps surfaces map to the k_c x k_c depth images I_u and I_l.

Consider a point P with the depth value $d = |P_y|$. Let I_s refer to I_u if $P^2 < 0$ and I_l otherwise. The d value will be placed at pixel coordinates $\bar{p}$ in I_s, where

$$\bar{p} = C_c \begin{bmatrix} P^1/d \\ P^3/d \end{bmatrix} + \begin{bmatrix} k_c/2 \\ k_c/2 \end{bmatrix} \quad (3)$$

and $C_c = \alpha k_c/4$. The inverse mapping follows from known depth d and pixel coordinates $\bar{p}$ via the image inverse projection map. Propagation of egocap data follows a similar process as for the egocylinder but uses the egocap projection equations and their inverses. Points mapping outside of the egocylinder domain now map to one of the egocaps, thereby retaining memory of any obstacle information. Fig. 2(c) depicts the egocan reconstruction, which retains the hydrant.

B. Finding gaps with perception space

Marr's visual hierarchy [15] suggests a mechanism to identify gaps in the local free-space. Perception-space resides in Marr's 2.5D representation layer, which is posited to encode depth layering. Consider a spherical ray-based range representation of a robot's freespace. The range at each ray corresponds to the distance the robot can travel in the direction of the ray without colliding. Thresholding the ranges to based on a given lookahead distance d_{la} segments the rays into colliding and non-colliding distances. Gaps manifest as connected regions of false (non-colliding) values.

a) Collision Maps: Precisely computing a collision map over many distance thresholds is prohibitively expensive. A series of simplifications does generate an approximate collision map at lower cost: (1): Reduce the number of ranges that are collision checked. At minimum, the set of evaluated ranges R_{eval} consists of a *far* value (the threshold distance from above) and a *near* value (where $0 < near < far$). (2): Assume negligible pitch and roll. This reduces the state space to be evaluated but requires limiting velocities and accelerations. (3): Assume the robot's projective geometry does not vary dramatically with relative height. This allows our perception space collision checking approach (described in III-A) to use a single projection of the robot's geometry at range r_h for all collision checks at r_h.

Using the first simplification, define a set of two offset ranges for future robot poses, near and far. Threshold the egocan at these two offset distances to establish two binary collision maps (near and far). Each pixel in the maps

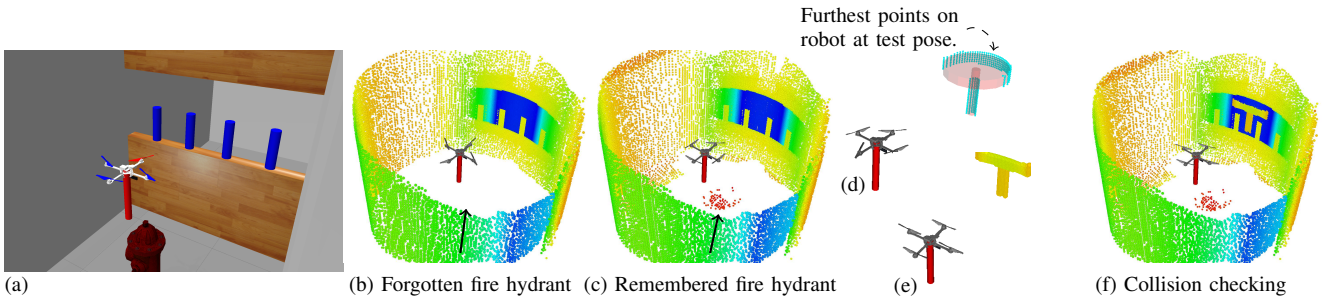


Fig. 2. Depiction of how an environment (a) is represented by the egocylinder (b) and egocan (c). Color maps from near (red) to far (blue), with unknown points in white. The fire hydrant is missing in the egocylinder representation. Continuing, there is a visualization of the detailed robot geometry at a pose with cyan points marking the far surface (d), which are then mapped to the egocan representation (e) giving a hallucinated robot measurement at the future pose, that is then overlaid (f) onto the stored egocan data from (c) for collision checking. The pose is not in collision.

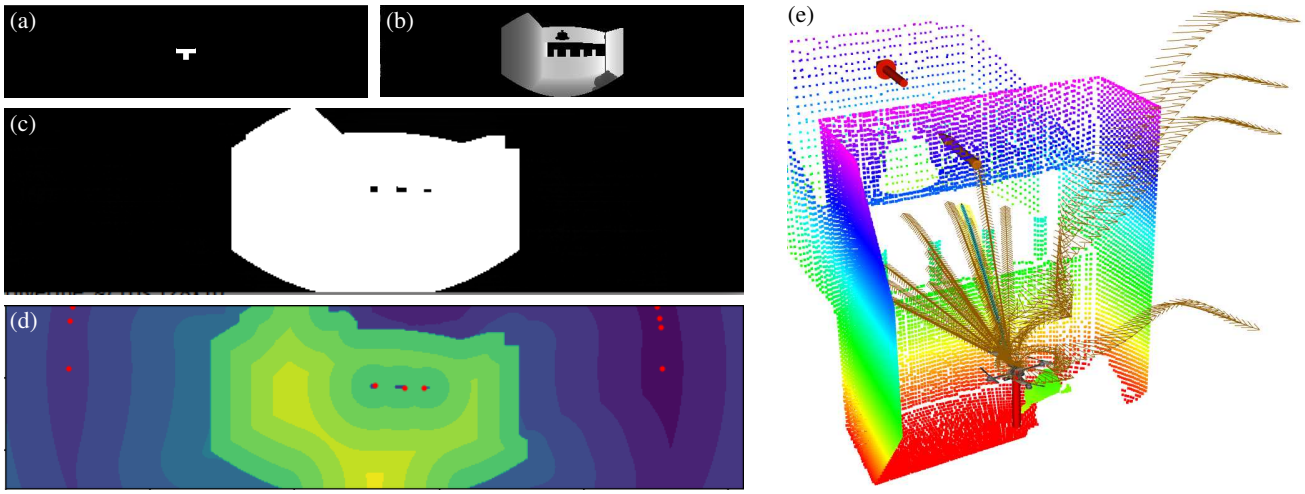


Fig. 3. Depiction of the steps to identify candidate gaps points from the egocan for the scenario depicted in earlier figures. (a) Robot map from hallucinating a robot at an offset pose; (b) The egocylinder image; (c) Binary union of convolved binary collision maps with hallucinated robot maps; (d) Costmap with cost coloring going from low (purple) to high (yellow). Local minima are marked by red dots. The right-most figure (e) is an example of local planning from the gap points, plus an additional set of exploratory points. Considered trajectories are shown as tan arrows radiating outward from the robot. The selected trajectory is indicated by yellow arrows and blue base points. The egocan information is visualized as a point cloud with color indicating height, from low (red) to high (purple).

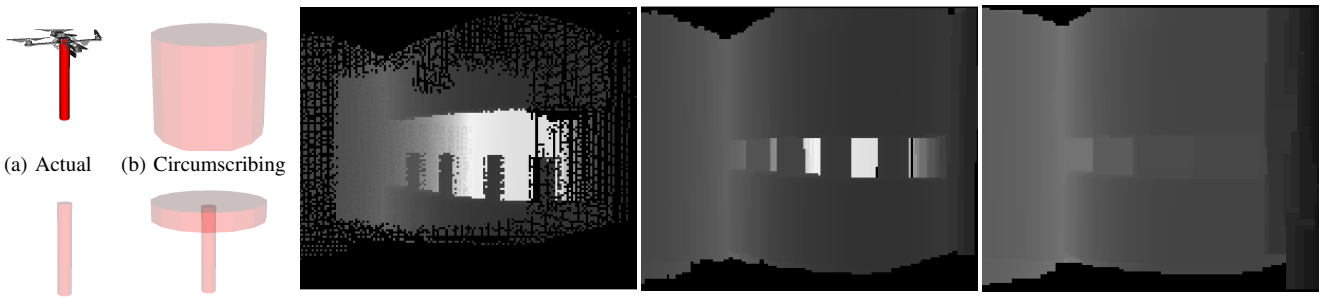
represents the expected collision outcome for a point mass robot following the line of sight path in the direction of the pixel's ray. Paths terminating at false pixels are collision-free for a point robot (up to the offset distance).

b) Collision Checking: The latter simplifications permit volumetric collision checking of the robot. Place the robot at two offset ranges in egocylinder coordinates and render its silhouettes to generate collision masks. The rendering should use the furthest-surface points. The offset range chosen should have the furthest of all surface points at the specified near/far distance. Convolving each binary mask with its associated collision map, then binarizing, defines candidate collision-free poses that respect the robot's geometry. Joining them with OR operations leads to a single perception-space collision map. Candidate terminal points are in non-colliding (false) regions.

Figure 3 depicts the process, starting with (a) the convolution kernel of the robot for a given offset distance, followed by (b) visualization of egocylinder part of the egocan, and (c) a final collision map generated based on merging individual collision maps. The collision map in Fig. 3(c) exhibits three gap regions in the center, plus one large connected gap region composed of the left and right *false* regions (rearward movement). These gap regions lead to topologically distinct trajectory sets, such that trajectories going through one gap cannot be warped to pass through another gap without passing through collision space. Each gap region represents a set of possible trajectories that reflect navigation opportunities for local goal attainment.

C. Sampling Candidate Waypoints from Gaps

The next step is to identify non-colliding points in the collision map to serve as candidate local waypoints for advancing towards the defined goal. Our solution, denoted *3DGap*, constructs a perception space costmap based on the Euclidean distance transform of the collision map and the distance between each pixel and the projected local goal. The resulting costmap is a level-set function, such as depicted in Fig. 3(d). Identifying local minima (with non-minima suppression) will generate candidate gap points to target for synthesizing collision-free trajectories. In Fig. 3(d) there are three minima for the three central holes (i.e., free-space



(a) Actual (b) Circumscribing (c) Inscribing (d) Detailed (e) Egocylinder range image (f) Liberally inflated (g) Conservatively inflated

Fig. 4. Collision checking of the actual robot (a) uses an idealized conservative model (b), idealized liberal model (c), and, when needed, a detailed non-idealized model. The actual tests for a given egocan involve a collision filtering process. The process conditionally queries the actual data, such as the egocylinder image (e), the liberal equivalent (f), and the conservative equivalent (g).

gaps), plus several vertical near the left and right edges. These 10 points define local goal candidates to score and collision check. Sparsification of the minima is due to non-maximal suppression and to a lesser degree quantization artifacts in the distance transform.

More generally, up to k_{nms} points are selected from the perception-space costmap. Once the points have been identified, the next step is to map them to 3D navigation waypoints. Due to the line-of-sight visibility properties of the egocan, it is desirable to plan trajectories that end at the target gap rather than passing through it since not much may be known about the environment on the other side of the obstacles. As a result, the distance to backproject each selected local minima pixel ray is set to the distance of the nearest collision point in the collision map (based on tracing along the characteristics of the distance transform). Some local minima do not point towards the goal, such as the left-most and right-most sets of points in Fig. 3(d).

Due to the depth quantization induced by selecting only two offset distances for collision checking, the waypoints generated from the costmap local minima may be unnecessarily close to obstacles. When possible, the waypoint is shifted away from nearby obstacles by checking for collisions in a 5x5 grid (centered on the waypoint and orthogonal to the vector between the robot and waypoint) and identifying the center of the largest continuous collision free region. An example waypoint set can be seen in Fig. 3(e), as the terminal points for the trajectories marked by brown or yellow vectors.

III. Collision Checking in Perception Space

Perception space collision checking requires mapping future robot poses into perception space [16]. It involves hallucinating a perception space representation of the robot model at the specified pose and comparing it to the current local map representation. This section outlines the process of hallucinating an egocan representation of a robot and accelerated collision checking through *collision filtering*.

A. Collision checks with non-idealized robot geometry

Let $\hat{I}_c$, $\hat{I}_u$, and $\hat{I}_l$ represent a hallucinated egocan, which is equivalent to the z-buffer of a rendered image but with furthest surface selection. Each pixel in $\hat{I}_c$ is assigned the range of the farthest point projecting to it (pixels with no projected points have 0 depth). Similarly, model geometry is projected into $\hat{I}_u$ and $\hat{I}_l$ using farthest point depth values. Figure 2 provides an example of this hallucinating process. Figure 2(d) depicts the actual robot and the robot pose to test for collision. Figure 2(b) depicts the equivalent egocan hallucination, and Fig. 2(e) depicts the hallucinated data overlaid on the stored egocan data. The robot is not in collision as noted by the fact that the original egocan data for the area, Fig. 2(f), is blue and hence further than the robot data, which is yellow to green. Formally, collision checking involves testing that $\hat{I}_\bullet < I_\bullet$ for the three egocan surfaces.

B. Collision Filtering with Inflated Models

While collision checking with the egocan is fast relative to other methods that test against the complete geometry, it is slower than point-robot models with organized data structures (and constant or logarithmic collision check costs). Egocans inflated by the circumscribing (Fig. 4.b) and inscribing (Fig. 4.c) geometry of the robot permit point-model collision checking but can return false positives and false negatives, respectively. These mutually exclusive false positive/negative properties can be exploited as part of a multi-pass collision checking framework to improve computational efficiency without compromising accuracy.

A more detailed, non-idealized model (Fig. 4.d) that covers the collision volume of the actual model (Fig. 4.a) without including too much additional volume will ideally minimize both false positives and false negatives at the cost of higher computational complexity. The detailed collision checker can address situations where the results of the idealized models are not sufficient to declare a pose as either colliding or collision free. Consequently, our multipass collision checking approach operates as follows:

1) Test the conservatively inflated model:
   Pass if no collision, continue if collision.
2) Test the liberally inflated model:
   Continue if no collision, fail if collision.
3) Test the non-ideal model:
   Pass if no collision, fail if collision.

Much like a physical multi-layer filter excludes objects of different sizes, this approach does the same for the different

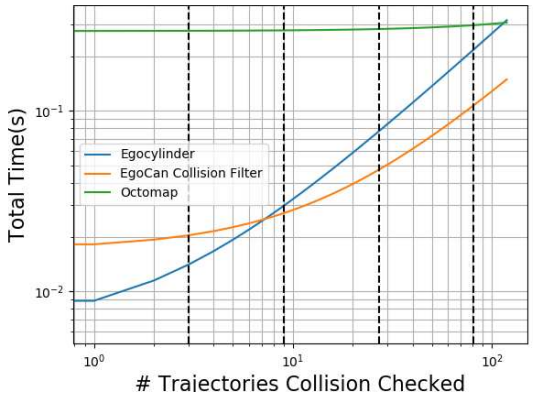


Fig. 5. Benchmarking the total time to update a representation and evaluate N trajectories for Egocylinder, Egocan Collision Filter, and Octomap. Octomap resolution is 0.1m; all other parameters are given in Table I.

TABLE I

EGOCAN & PLANNING PARAMETERS

Egocylinder Size ($k_h * k_v$)	Egocap Size ($k_c * k_c$)	Depth Image Resolution	Depth Image Rate
512x128	128x128	320x240	10Hz

Max Replan Period (T_{replan})	Min Time to Collision ($T_{collision}$)	Min Time to Traj. End ($T_{endpoint}$)
1s	5s	5s

Freespace threshold (T_{fs})	Max # Points to Sample (k_{nms})	Min Distance btwn Samples (d_{nms})	Hallucination Distances (near, far)
10	10	5	1m, 2m

robot models (too large, too small, just right) as needed. We call the process *collision filtering*. If the navigation scenario is benign and there is ample space in the local environment, then the conservative model will be sufficient. In tight scenarios, the additional passes will be required.

Also depicted in Fig. 4 are the original perception space egocan data, the liberally inflated version, and the conservatively inflated version. In this case, the conservative egocan indicates that the center region of the image is not passable. The liberal egocan indicates that it is. The detailed model check is required to establish the proper outcomes for a given pose, as depicted earlier in Fig. 2(f).

C. Benchmarking Egocan Computational Costs

We perform a benchmarking analysis of the computational costs of the egocan collision filter approach, the standard egocylinder approach, and an octomap implementation. Time statistics include both the time to assimilate the new data into the collision-space representation plus the time cost of evaluating n trajectories for collisions. Benchmarking uses a dataset of prerecorded depth images and pregenerated trajectories. Collecting the timing trends results in Fig. 5.

Due to the higher setup costs of organized data structures, perception space approaches have an initial budget advantage. However, higher collision checking costs due to using geometric models reduce the processing time advantage as more trajectories are checked. Collision filtering lowers the slope so that the cross-over point occurs later.

IV. EXPERIMENTS

This section describes the integration of these concepts as part of a local planner in a hierarchical navigation system and the experiments performed to evaluate it.

A. AeriaLPiPS: A Perception Space Navigation Pipeline

At the top level, a global planner uses an incomplete 2D prior map of the environment to generate an estimated path to the goal. This path is not updated during navigation, increasing reliance on the local planner for maintaining collision avoiding movement within the environment. Sensor measurements for collision avoidance come from a forward mounted depth camera.

Trajectories are generated from the robot's current pose to each of the candidate waypoints provided by the approach described in II-C and to a local goal on the global path. The reference paths for these trajectories are designed to reach the target heights early in order to approach target poses straight-on and provide a compatible camera perspective of the region beyond the gap. To maximize the ability to avoid obstacles, trajectories are synthesized such that the MAV always faces in the direction of motion. This is performed using a modified trajectory generation approach [16] to stabilize a trajectory in [x, y, z, θ] along the provided 3D reference path.

Each trajectory is analyzed and assigned a cost based on the estimated cost-to-go; trajectories that go too high or too low are assigned a fatal cost. As in [5], trajectories are sorted by ascending cost and collision checked one after the other until a collision-free trajectory is found (if any). If a collision-free trajectory is found it is executed by a Lee position controller [17]; otherwise, the controller holds the MAV at its present pose.

Replanning is performed if the current trajectory collides in less than $T_{collision}$ secs, if less than $T_{endpoint}$ secs remains (unless the trajectory ends at the global goal), or if more than T_{replan} secs have elapsed since last planning. Navigation ends on arriving within D_{thresh} meters of the specified goal.

B. Navigation Tasks

The navigation system is tested in 2 simulated environments using Gazebo with the RotorS MAV simulation plugin/framework [17], [18]. The robot platform utilized is a modified Hummingbird with a 40x5cm vertical pole rigidly fixed to its underside. Each of the test environments (depicted in Fig. 6) is designed to demonstrate a particular capability of the navigation system:

Obstacle course: go over/under/around obstacles and through openings.

Barrier: go through tight openings much too small for traditional approximations (see Fig 1).

The geometric representations used for the conservative, liberal, and detailed egocan collision checkers are shown in Fig 4(b)-(d). Table I gives the main egocan and planning-related parameters used. The prior maps provided to the

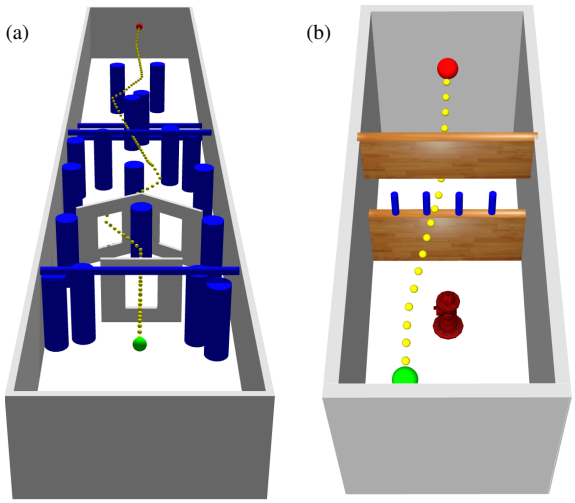


Fig. 6. Overhead views of simulated worlds used in experiments: (a): Obstacle course scenario, (b): Non-idealized barrier scenario. Starting locations are marked by green spheres, goal locations by red spheres, and example paths taken by smaller yellow spheres.

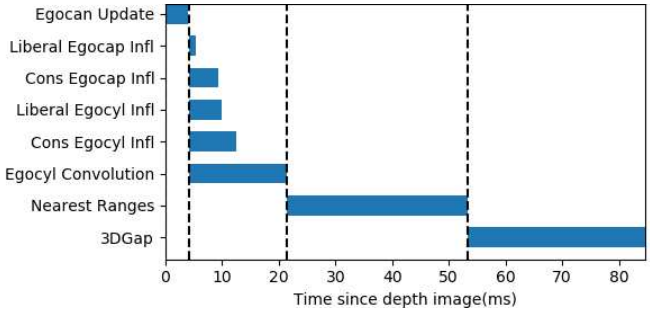


Fig. 7. Timeline depicting the various stages of the perceptual pipeline. The duration of each stage represents the average over all successful experiments.

planner contain only the exterior walls. The expermental conditions for each environment are held constant through the many repeated runs. Desired forward velocity is set to 0.25m/s to satisfy the upright pose assumption.

C. Results

As shown in Fig. 6, the proposed navigation system succeeded in finding collision-free paths through each environment, including through a barrier that would have thwarted planners relying on idealized geometries. Success rates for each world, along with other statistics, are provided in Table II. The most common cause of failure was not selecting a trajectory through the first opening in time.

The timeline in Fig. 7 illustrates the parallel nature of the perception and waypoint sampling pipeline. Vertical dashed lines indicate dependencies between stages. Since planning makes use of the most recent waypoints and inflated images available, it is not limited by the update rate of the entire pipeline and can run at a faster rate; there is little value to running significantly faster since new information will not be available. The nearest obstacle analysis and the image-based collision costmap are the processing bottlenecks for the current implementation. The current versions have not been

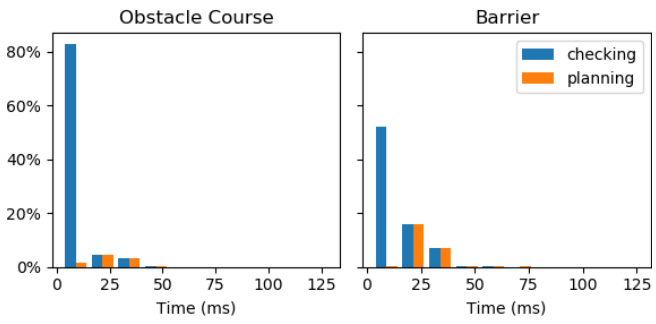


Fig. 8. Timing distributions for each scenario, categorized by whether the planner only checked the current trajectory (*checking*) or also considered and evaluated additional trajectories (*planning*).

TABLE II
NAVIGATION RESULTS

World	Success Rate	Path Time (s)	Path Length (m)
Obstacle course	178/186	108.90 (0.3)	27.8 (1.3)
Barrier	181/182	20.0 (0.4)	5.0 (2.1)

optimized and may be improved to have lower time costs. The distributions of planning times for each environment are shown in Fig. 8. During trajectory tracking, only collision checking is performed, which has a low time cost. It operates at frame-rate (30-50Hz). Planning takes longer and operates at an environment-dependent rate. The result is a bimodal time distribution for the local module of the hierarchical navigation system.

V. CONCLUSION

This work considered the case of robot navigation problems whose planning approaches require accommodating non-idealized geometry for collision checking and proposed a perception space solution in the form of the egocan, with associated gap processing *3DGap*. The egocan is a local perception space model of free-space used for efficient collision checking (in terms of time from image data to first collision check). Using inflated point-robot models improves multi-query collision checking efficiency through filtered processing. *3DGap* is an egocan-based local goal identifier that identifies gap regions for recovering locally optimal terminal points with collision-free reachability. Simulated scenarios demonstrated that the local planning method, called *AeriaLPiPS*, enables successful navigation in multiple environments, including those in which traditional planners relying on idealized geometry would fail. The use of gaps for planar navigation has an associated proof of collision-free operation for idealized robot models [14]. We believe that the same should hold for navigation via gaps through space and hope to translate the existing proof to the 3D case. The high success rate here suggests that it should be possible. In future work, we also plan to explore richer trajectory spaces, agile maneuvering, and using the projection operator as a barrier function [14] for reshaping the control input to enhance safety.

REFERENCES

[1] B. T. Lopez and J. P. How, "Aggressive 3-D collision avoidance for high-speed navigation," in *IEEE International Conference on Robotics and Automation*, May 2017, pp. 5759–5765.

[2] A. Bry, C. Richter, A. Bachrach, and N. Roy, "Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 969–1002, 2015.

[3] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2016, pp. 5332–5339.

[4] L. Campos-Macías, R. Aldana-López, R. de la Guardia, J. I. Parra-Vilchis, and D. Gómez-Gutiérrez, "Autonomous navigation of mavs in unknown cluttered environments," *Journal of Field Robotics*, vol. 38, no. 2, pp. 307–326, 2021.

[5] J. Smith, S. Feng, F. Lyu, and P. Vela, "Real-time egocentric navigation using 3D sensing," in *Machine Vision and Navigation*, O. Sergiyenko, W. Flores-Fuentes, and P. Mercorelli, Eds. Springer, 2020, p. 431–484.

[6] N. Bucki, J. Lee, and M. W. Mueller, "Rectangular pyramid partitioning using integrated depth sensors (rappids): A fast planner for multicopter navigation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4626–4633, July 2020.

[7] A. T. Fragoso, "Egospace Motion Planning Representations for Micro Air Vehicles," Ph.D. dissertation, California Institute of Technology, 2018.

[8] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in SE(3)," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, July 2018.

[9] N. J. Sanket, C. D. Singh, K. Ganguly, C. Fermüller, and Y. Aloimonos, "Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2799–2806, Oct 2018.

[10] Menglu Lan, Shupeng Lai, Yingcai Bi, Hailong Qin, Jiaxin Li, Feng Lin, and B. M. Chen, "Bit*-based path planning for micro aerial vehicles," in *IECON - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Oct 2016, pp. 6079–6084.

[11] B. MacAllister, J. Butzke, A. Kushleyev, H. Pandey, and M. Likhachev, "Path planning for non-circular micro aerial vehicles in constrained environments," in *IEEE International Conference on Robotics and Automation*, May 2013, pp. 3933–3940.

[12] S. Yang, B. He, Z. Wang, C. Xu, and F. Gao, "Whole-body real-time motion planning for multicopters," in *IEEE International Conference on Robotics and Automation*, May 2021, pp. 9197–9203.

[13] J. Smith, R. Xu, and P. Vela, "egoTEB: Ego-centric, perception space navigation using timed-elastic bands," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 2703–2709.

[14] R. Xu, S. Feng, and P. Vela, "Potential gap: A gap-informed reactive policy for safe hierarchical navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8325–8332, 2021.

[15] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. New York, NY, USA: Henry Holt and Co., Inc., 1982.

[16] J. Smith and P. Vela, "PiPS: Planning in perception space," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 6204–6209.

[17] (2018) rotors_simulator. [Online]. Available: https://github.com/ethz-asl/rotors_simulator

[18] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," in *Robot Operating System (ROS): The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, 2016, pp. 595–625.