Reactive Control and Metric-Topological Planning for Exploration

Michael T. Ohradzansky[1*], Andrew B. Mills[1*], Eugene R. Rush[2], Danny G. Riley[3],
Eric W. Frew[1], J. Sean Humbert [2]

***Abstract*—Autonomous navigation an unknown environments with the intent of exploring all traversable areas is a significant challenge for robotic platforms. In this paper, a simple yet reliable method for exploring unknown environments is presented based on bio-inspired reactive control and metric-topological planning. The reactive control algorithm is modeled after the spatial decomposition of wide and small-field patterns of optic flow in the insect visuomotor system. Centering behaviour and small obstacle detection and avoidance are achieved through wide-field integration and Fourier residual analysis of instantaneous measured nearness respectively. A topological graph is estimated using image processing techniques on a continuous occupancy grid image. Node paths are rapidly generated to navigate to the nearest unexplored edge in the graph. It is shown through rigorous field-testing that the proposed control and planning method is robust, reliable, and computationally efficient.**

***Index Terms*—exploration, centering, control, mapping**

I. Introduction

Successfully exploring and navigating unknown environments with robots is invaluable for providing situational awareness in a number of application spaces. Fire fighting in forested and urban areas, rescue operations in natural disasters, cleaning operations after oil spills or building collapses, underwater and space exploration/mapping, as well as surveillance for security and maintenance in remote areas all present possible application spaces for autonomous robot exploration. Autonomous robots can be deployed in place of humans to enter a structure and assess it for potential hazards, keeping humans out of harms way.

Exploration of an unknown environment is an established robotics problem. Most approaches use a continuous frontier-based method over a metric map. These frontiers may be iterated over to determine the lowest cost destination to explore [1]. Other approaches utilize clustering of frontier or consider traversability and reachability [2], [3]. Other continuous methods utilize information-theoretic approaches, which consider the expected information gain over a probabilistic map [4]. Experiments have shown improved path planning and efficient map building by reducing the entropy of a probabilistic map representation [5], [6], [7], [8]. Inference-based predictions of expected features in the environment can also lead to performance gains [9]. These approaches often help when large areas of the environment need to be explored quickly, but they can be complex and rely heavily on explicit path planning which often scales poorly with environment size and requires fast and accurate pose and metric map estimates.

Some exploration approaches use a topological graph to represent the map [10]. Most graph estimation techniques utilize Voronoi representations of the environment to identify critical points (points of local minimum distance from the nearest obstacle) and create a graph between them as in [11]. These approaches produce two edge nodes which are topologically uninteresting and lead to cluttered graphs. An extension of graph exploration is segmentation, where known areas are separated into segments for exploration [12]. Graph-based exploration has been found to be useful in tunnels [13], indoor rooms [14], and 3D maze-like indoor environments [15]. Once a graph is built, the graph can be used to efficiently navigate from one area of the map to another, to facilitate exploration, or other tasks [16]. Graph-based planning is very efficient and eliminates the need for explicit path planning over a metric map, but it is often expensive and difficult to accurately estimate a useful graph from a metric map.

This work departs from previous work in that it uses an accurate and computationally efficient graph estimation method utilizing image convolutions on a noisy but consistent metric map. A graph representation eliminates the complex overhead of a path planning system. However, path planning and obstacle avoidance are typically both integral to achieving robust and safe autonomous navigation in complex environments [17]. In order to traverse these sorts of environments, we look to reactive sensor-based obstacle avoidance to provide fast and robust navigation.

Bio-inspired approaches to robot control are a source of computationally efficient control solutions for simple navigation strategies, such as centering or speed regulation. Flying insects have developed elegant solutions to the challenges of perception and navigation using very limited onboard sensing and compute [18]. Insects rely on *optic flow*, which are the characteristic patterns of visual motion that form on their retinas as they move [19]. Many of the optic flow based navigation methods are inspired by the work of Srinivasan [20] and [21], who, through observation of honeybees traversing a corridor, developed a well-known heuristic, the *centering* response. This approach was tested in [22] and [23] where the centering response was achieved by balancing the average optic flow signal on both sides of the vehicle.

Further analysis of the insect visuomotor system by Humbert et al. reveals how specialized tangential-cell processing facilitates extraction of information critical to local navigation, and an analogous method of wide-field integration (WFI) is

[1]Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, CO, Michael.Ohradzansky@colorado.edu, Andrew.B.Mills@colorado.edu, Eric.Frew@colorado.edu

[2]Department of Mechanical Engineering, University of Colorado Boulder, CO, Eugene.Rush@colorado.edu, Sean.Humbert@colorado.edu

[3]Department of Computer Science, University of Colorado Boulder, CO, Dan.Riley@colorado.edu

*Both authors contributed equally to this work

This work was supported through the DARPA Subterranean Challenge, cooperative agreement number HR0011-18-2-0043

Manuscript 2024 submitted to 2020 International Conference on Robotics and Automation (ICRA). Received September 15, 2019.

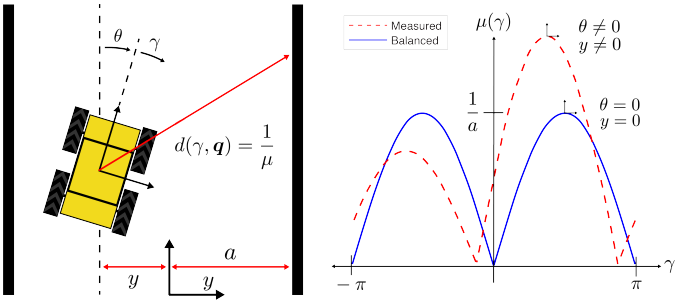

Fig. 1. Planar tunnel geometry. a) Notation and vehicle state definitions. b) Corresponding measured nearness signal shown in dashed red, along with the balanced nearness signal in solid blue.

introduced in [24]. It is demonstrated that centering behavior can be asymptotically stabilized in a corridor using feedback generated by the appropriate choice of weighting patterns. Optic flow can also be used for small obstacle avoidance, as demonstrated in [25] and [26].

In many of the previously stated methods, optic flow is extracted from camera image streams, which requires additional steps of processing to achieve. Similar to optic flow, depth scans also encode information about a vehicle's orientation and lateral displacement relative to the center-line of a corridor. One benefit of using depth scans is that they can be measured directly using any number of COTS scanning lidar devices, such as the RPLIDAR A3 used in this work. This paper departs from [24], [25], and [26] by applying the same WFI method to the nearness function directly. Nearness is a component of the tangential optic flow model present, and can be computed easily from depth scans.

We present an approach with fast and accurate graph estimation and simple topological planning which interfaces in a robust manner with a reactive corridor-centering controller. This method alleviates the need for explicit path planning and the bandwidth requirements of position-based path-following controllers. Other approaches may require numerous sensing modalities to map, estimate vehicle state, and avoid obstacles. Our approach only requires a depth sensor and is improved by IMU and odometry estimates. Section III details the nearness-based controller, section IV explains the graph estimation and planning scheme, and section V details our experimental setup and section VI is a discussion of results.

II. Nearness-Based Control

In this section, a feedback controller is developed based on wide field integration (WFI) of horizontal depth scans for navigating unknown, corridor-like environments. The method of WFI is an analog to tangential cell processing present in the insect visuomotor system. It is demonstrated that centering behavior can be asymptotically stabilized in a corridor-like environment using feedback generated by the appropriate choice of weighting patterns. In addition to centering, a method for small obstacle detection and avoidance is developed based on Fourier residual analysis of measured nearness.

A. Wide-field Centering Response

In [24], a spatially continuous representation of planar optic flow is presented. One of the components of optic flow, nearness, is also spatially continuous and is defined as

$$\mu(\gamma, \boldsymbol{q}) = \frac{1}{d(\gamma, \boldsymbol{q})} \tag{1}$$

where d is the azimuthal distance to the nearest object in the environment, γ is the body-frame-referred viewing angle, and the vehicle's pose $\boldsymbol{q} = (x, y, \theta)$. The nearness function can be written in closed form for a planar tunnel geometry as a function of γ , the tunnel half-width a, lateral position y, and body-frame orientation θ:

$$\mu(\gamma, \boldsymbol{q}) = \begin{cases} -\dfrac{\sin(\gamma+\theta)}{a+y} & -\pi \leq \gamma+\theta < 0 \\ \dfrac{\sin(\gamma+\theta)}{a-y} & 0 \leq \gamma+\theta < \pi \end{cases} \tag{2}$$

By projecting the nearness function onto different weighting patterns, simple representations of relative proximity and orientation to the surrounding environment can be extracted.

The insect visuomotor tangential cells are modeled as weighting patterns $F_i(\gamma)$, which represent sensitivity to particular nearness patterns in the azimuthal plane. The wide field integration operation is modeled as a spatial inner product with the nearness function, which maps sensitivity patterns to controlled inputs:

$$u_i(\boldsymbol{x}) = \frac{1}{\pi} \int_{-\pi}^{\pi} \mu(\gamma, \boldsymbol{q}) \cdot F_i(\gamma)\, d\gamma \tag{3}$$

where $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})$. The inner product has been multiplied by a factor of $\frac{1}{\pi}$ to simplify later calculations and notation.

Sensitivity patterns of $\cos(n\gamma)$ and $\sin(n\gamma)$ have been chosen and the projections correspond to the spatial Fourier coefficients of the nearness function. Table II-A contains the analytical values of the linearized projections about $(y = 0, x = 0)$ up to the first two spatial harmonics.

Mode	Projection $y_j(\boldsymbol{x})$	Linearization
a_0	$\frac{2a(\cos\theta+1)}{\pi(a^2-y^2)}$	0
a_1	$\frac{\sin\theta(a\theta+\pi y)}{\pi(a^2-y^2)}$	0
b_1	$\frac{-a\sin\theta+a\theta\cos\theta+\pi y\cos\theta}{\pi(a^2-y^2)}$	$\frac{1}{a^2}y$
a_2	$\frac{(y-a)\cos\theta-(3a+y)\cos 2\theta}{3\pi(a^2-y^2)}$	0
b_2	$\frac{4a(2\sin\theta-\cos\theta)}{3\pi(a^2-y^2)}$	$\frac{8}{3a}\theta$

One can see that after WFI and linearization, simple representations of the vehicle's lateral displacement and orientation are generated. Extracting the first and second sine harmonics (b_1 and b_2) yields the observation equation $\boldsymbol{y} = C\boldsymbol{x}$, where $\boldsymbol{x} = (y, \theta)$, and C is given by:

$$C = \begin{pmatrix} \frac{1}{a^2} & 0 \\ 0 & \frac{8}{3a} \end{pmatrix} \tag{4}$$

Manuscript 2024 submitted to 2020 International Conference on Robotics and Automation (ICRA). Received September 15, 2019.

For the purpose of this paper, the unicycle model is used as the dynamics model for the vehicle. This model assumes control over the vehicle's forward speed, v, and turning rate, $\dot{\theta}$. The lateral equations of motion are as follows:

$$\begin{aligned} \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= u_\theta \end{aligned} \tag{5}$$

where v is the vehicle's forward speed and u_θ is the steering rate command. The control objective is to keep the vehicle centered in any corridor. Given an instantaneous nearness pattern μ, the corresponding weighting function for centering behaviour F_θ generates a compensatory steering response $u_\theta = \langle \mu, F_\theta \rangle$.

For design of the steering controller, a weighting pattern, or sum of weighting patterns, F_θ is chosen to achieve centering behaviour about the center-line of a corridor. To stabilize the vehicle to the center of the tunnel, the control input u_θ should contain both lateral displacement y and rotational θ stiffness terms. The sine harmonics, b_1 and b_2 contain these terms, and the corresponding feedback controller is

$$u_\theta = k_1 b_1 + k_2 b_2 \tag{6}$$

In this case, the weighting function $F_\theta = k_1 \sin(\gamma) + k_2 \sin(2\gamma)$. Combining Equations 9 and 10 with the analytic values for the sine harmonic terms leads to the final linearized closed-loop dynamics:

$$\begin{pmatrix} \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} 0 & v \\ \frac{8k_2}{3a} & \frac{k_1}{a^2} \end{pmatrix} \begin{pmatrix} y \\ \theta \end{pmatrix} \tag{7}$$

Equation 7 has eigenvalues:

$$s_{1,2} = \frac{k_1}{2a^2} \pm \frac{1}{2}\sqrt{\frac{k_1^2}{a^4} - \frac{32vk_2}{3a}} \tag{8}$$

With a choice of $k_1 < 0$, the centering control law is clearly stable, leaving k_2 to be tuned to satisfy additional performance criteria. It is interesting to note that the eigenvalues of the characteristic equation scale with $1/a^2$, so the closed-loop system responds faster in narrower corridors.

B. Small-field Obstacle Avoidance

In addition to the centering behaviour, the vehicle must also be able to avoid small obstacles in the environment. As described in [25], small obstacles show up as high-frequency perturbations in the measured optic flow. Similarly, small obstacles also show up as high-frequency perturbations to measured nearness μ, and can be extracted using a reconstruction of the wide-field nearness signal μ_{wf}. To reconstruct the wide-field signal, the first N harmonics of the Fourier series are used.

$$\begin{aligned} \mu_{sf}(\gamma, \boldsymbol{x}) &= \mu(\gamma, \boldsymbol{x}) - \mu_{wf}(\gamma, \boldsymbol{x}) \\ \mu_{wf}(\gamma, \boldsymbol{x}) &\approx \frac{a_0}{2} + \sum_{n=1}^{N} (a_n \cos n\gamma + b_n \sin n\gamma) \end{aligned} \tag{9}$$

A dynamic threshold of three times the standard deviation is applied to the current small-field signal. If a peak in the SF signal crosses the threshold, a small-field steering command is generated based on the following control law:

$$u_{sf} = k_0 \cdot \text{sign}(\gamma_0) \cdot e^{-k_\psi |\gamma_0|} \cdot e^{\frac{-k_d}{|d_0|}} \tag{10}$$

where γ_0 is the azimuthal location of the peak, d_0 is the proximity of the peak, and k_0, k_ψ, and k_d are tuning parameters. The small-field steering command u_{sf} is simply added to the wide-field steering command u_{wf} to generate the final steering command u_θ. Adding the two steering commands enables the vehicle to avoid small obstacles that it detects in the sensor plane while maintaining proper centering behaviour. The vehicle obeys the nearness controller until it reaches a predefined radius of a node, at which point it switches steering and forward speed control over to the controller described in section III.

C. Forward Speed Regulation

Control of the vehicle's forward speed is also handled using feedback on Fourier harmonics according to:

$$u_v = v_{max}(1 - k_v * |b_2|) \tag{11}$$

Here, feedback from the second sine harmonic slows the vehicle down when it has orientation error. This slowing response can be tuned by adjusting the forward speed controller gain k_v. The final forward speed command u_v is saturated by parameters v_{min} and v_{max}.

III. Planning

This section details a topological planning strategy. This planner directs the vehicle when the local environment no longer resembles a corridor, i.e. junctions and dead-ends. It requires a system capable of solving the simultaneous localization and mapping (SLAM) problem to generate a map that is consistent and estimate the vehicle's position. The planner first estimates a graph structure from a discrete map of the environment and then generates a path of edges on that graph to the nearest unexplored edge.

A. Graph Estimation

In order to navigate and plan through the environment, we reduce our metric representation of the environment, an occupancy grid, into a sparse topological representation, a graph. Our estimate of a topological representation of the environment is a three-step process: skeletonization, node extraction, and edge extraction.

1) Skeletonization: First, the map image is binarized by selecting an occupancy probability threshold, if using an occupancy grid. Then, the image is smoothed with a Gaussian blur. We use a Gaussian blur convolution because local variations in corridor widths and uneven wall textures may cause our next step, thinning, to place nodes and edges where there are none. The kernel for our Gaussian image convolution is a square Gaussian noise matrix with zero mean, variance σ^2, and size:

$$s = 2\lceil c\sigma \rceil + 1 \tag{12}$$

This kernel size is always odd, at least 3×3 and increases in size as the variance increases which yields a wider blur. c is a constant parameter that determines the discrete spread of the blur as a function of the variance; $c = 2$ was chosen for our problem as it gave the most consistent thinning results. Finally, the smoothed image is thinned to extract the medial axis or skeleton of the binary image. This work uses a fast approximation of the Zhang-Suen thinning algorithm presented in [27]. An example skeletonization and its resulting graph is given in Figure 2.

2) Node Extraction: Given that the vehicle can always navigate through a corridor-like environment, we are interested in only extracting nodes with degree 1 or greater than 2. Degree 2 nodes correspond to sections of the environment where the vehicle does not require additional control to smoothly traverse. Thus, we are interested in constructing a graph that is as sparse as possible.

Nodes with degree 1 or greater than 2 will be located at skeleton pixels that are adjacent to 1 or more than 2 skeleton pixels. Nodes are extracted from the skeletonized image by performing a second image convolution with the following kernel:

$$K_{node} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 9 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{13}$$

From the resulting convolution, pixels with a value of 10 are node locations with a degree of 1, or end nodes, and pixels with a value greater than 11 are node locations with at least three edges.

Often, there are multiple adjacent pixels with a value greater than 11. In this case, we group all pixels within a 3×3 neighborhood into one node location which is closest to the centroid of the group.

3) Edge Extraction: Edges are extracted by examining the previous image convolution in a neighborhood around the node pixels. This extraction routine is detailed in Algorithm 1.

Algorithm 1 takes the node array, N, and the resultant image convolution from Node Extraction, I, and outputs the graph, G. The edges of the graph, E, are a tuple of the parent node, child node, and the pixel path between them. The pixel path is used to compute the cost to travel over each edge and may be used to assess the traversability of that edge. Neighborhood(a, b) returns the $(2b{+}1){\times}(2b{+}1)$ neighborhood of pixel indices centered on pixel location a. $p(n)$ is the pixel location of node n. Finally, FindSkeleton() returns the pixel locations of all skeleton pixels in the list.

Algorithm 1 $\mathcal{G} = (N, E) \leftarrow$EdgeExtract$(N, I)$

```
G ← InitializeGraph(N);
for n in N \ N_end do
    for p in Neighborhood(p(n), r_neighbor) do
        if IsSkeleton(p) and Neighbors(p, p(n)) then
            p_prev ← p(n)
            while IsSkeleton(p) do
                path.append(p)
                p ← (FindSkeleton(Neighborhood(p, 1)\p_prev))
            end while
            E ← E.append(n, ClosestNode(p), path)
        end if
    end for
end for
return G
```

B. Exploration

Exploration of the environment can be accomplished by constantly traveling to the frontier [1]. Similarly, given a graph, G, the graph may be explored by traversing unexplored edges. The only edges that may be unexplored are pendant edges (edges that connect to nodes of degree 1). Frontier pixels are labeled in the map image and then clustered to filter out small groups caused by sensor noise and small occluded geometry. Pendant edges are labeled unexplored if they have not been traversed and there is a minimum number of frontier pixels in close proximity to the corresponding pendant node. By using frontiers to label edges explored/unexplored, we ensure that the robot does not spend time traversing edges that do not expand the map.

The general exploration strategy is to navigate from the robot's current position to the nearest unexplored edge. "Nearness" to each node with an unexplored edge can easily be evaluated with Dijkstra's algorithm, A*, or the reader's choice of graph-based planning strategy. The cost to traverse each edge is the length of that edge's pixel path. More efficient exploration strategies are available for exploring an unknown graph with one or many robots, but those are not the subject of this work [16].

C. Control Integration

In order for the robot to follow the commanded series of edges to the closest unseen edge, the robot makes a decision at each node in the commanded edge path. The robot switches from nearness control to a position-based controller once it is within a certain radius of the next node. A proportional controller:

$$\boldsymbol{u} = \begin{bmatrix} u_v \\ u_\theta \end{bmatrix} = \begin{bmatrix} k_p ||\boldsymbol{p_n} - \boldsymbol{p}||_2 \\ k_\theta (\text{atan2}\,(\Delta y, \Delta x) - \theta) \end{bmatrix} \tag{14}$$

closed around the robot's radial distance error, $(||\boldsymbol{p_n} - \boldsymbol{p}||_2)$, and inertial heading error, $(\text{atan2}\,(\Delta y, \Delta x) - \theta)$. This controller commands the robot until the rate of change of its radial distance error is below a certain value. Finally, the robot turns in place to face the inertial heading of the next edge in the commanded edge path, using proportional control around θ, and resumes wide-field control. Figure 4 shows the interaction between these two controllers.

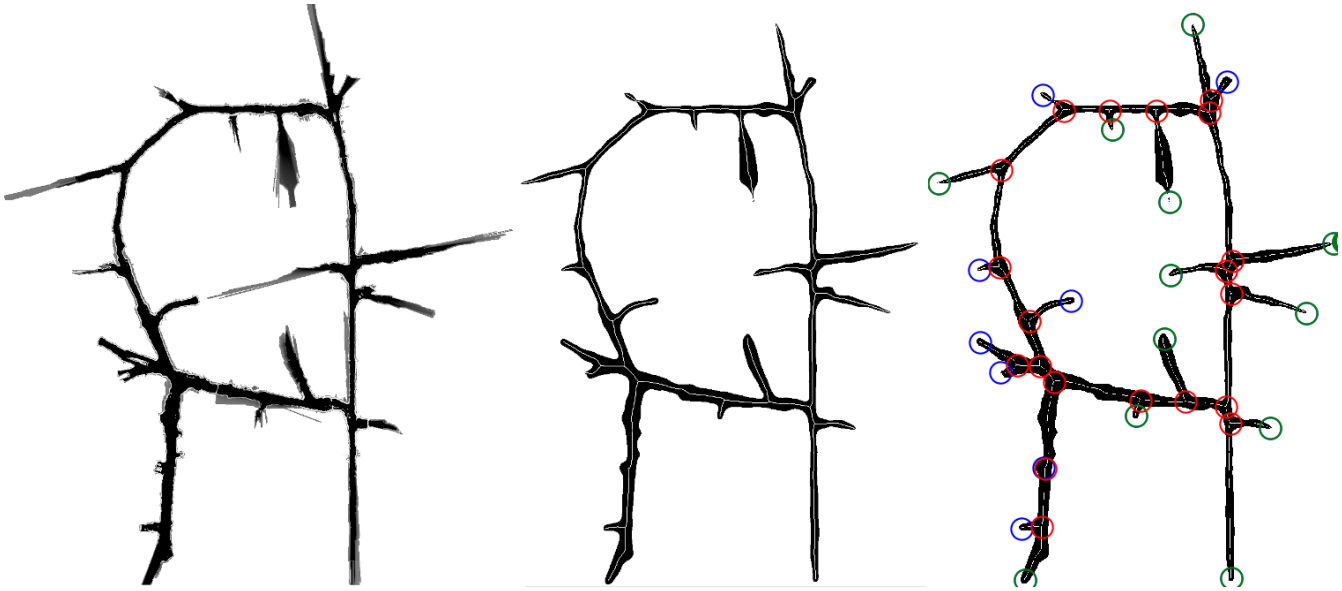

Fig. 2. Online graph estimate on map data from the Edgar Experimental Mine in Idaho Springs, CO. The leftmost image is a continuous occupancy grid map of the environment. The middle image is the medial axis result following the Gaussian blur, thresholding, and thinning. The rightmost image is the resulting graph. Red markers are nodes of degree ≥ 3, green markers are unexplored pendant nodes, and blue markers are explored.

IV. Experimental Setup

The proposed planning and control system was implimented on a Husky A200, a medium-sized unmanned ground vehicle (UGV), provided by Clearpath Robotics. The primary sensor used on the vehicle is the Slamtec RPLIDAR A3, which is a 360 degree laser range scanner. The A3 has a max sensing distance of 25m, a min sensing distance of 0.1m, and a sample rate of 10Hz. It is important to use a sensor with a minimum sensing distance that is less than the width of the vehicle so that it can center in narrow corridors. In addition to the RPLIDAR, a downward facing Realsense D435 is used for rough terrain detection and avoidance. An Ouster OS1-64 and Landmark Gladiator 3DM-GX5-15 IMU are used with Google Cartographer for generating local position and global map estimates required for graph estimation [28]. Scans from the RPLIDAR can also be used with Google Cartographer in lieu of the Ouster pointclouds, reducing the required sensor suite for planning and navigation to a single IMU and 360 degree depth scanner.

V. Validation

A. Results

The performance of the whole system is best characterized by the size of the environments explored, linear distance traveled, and average speed of traversal.

This navigation strategy was deployed on three identical vehicles at the DARPA Subterranean Challenge Circuit Event in the Pennsylvania-based Bruceton Experimental Mine. During one of the runs, one vehicle traveled a distance of 1,595m in 62 minutes while another traveled 953m in 56 minutes, averaging .43 m/s and .28 m/s respectively. Each of these runs was terminated by DARPA staff after the 60 minute time limit had been reached. During another run on a different course, one vehicle traveled 840m in 30 minutes, an average of .46 m/s.

Prior to the DARPA challenge, this system was deployed at the Edgar Experimental Mine in Idaho Springs, Colorado whose tunnels span over 135,000 m^2. In Figure 2, a graph of the mine constructed during a single test is shown. During this test, the vehicle covered 1,092m in 45 minutes, an average of .4 m/s.

Figure 3 presents four examples that demonstrate the performance of our centering solution. Example (a) shows this controller's response to a local, but significant, deviation in the environment from a corridor. As the robot navigates this corridor, the robot reacts to this deviation like a disturbance and quickly re-centers within the corridor over a few meters in response. Example (b) is actually three separate paths that the robot took to navigate one section of the environment.

Example (c) contains both a constrained passage (≈ 1.2 m wide compared to a ≈ 0.85 m wide Husky) and a 90 deg bend. Although this local environment bends and varies in width, our robot traverses it with ease because it is topologically linear. Finally, example (d) serves to characterize the controller's behavior to an initial angle error. The robot starts from the left with an initial heading error of ≈ 45 deg and exhibits an oscillatory response that decays over the next 10 meters until another heading error in the opposite direction presents itself and elicits a similar response. Examples (c) and (d) show a fairly traditional linear control response to a state error over a wider domain than a simple constant-width corridor.

B. Discussion

Our original efforts at finding a reliable open-source navigation solution for exploring unknown environments did not

Manuscript 2024 submitted to 2020 International Conference on Robotics and Automation (ICRA). Received September 15, 2019.

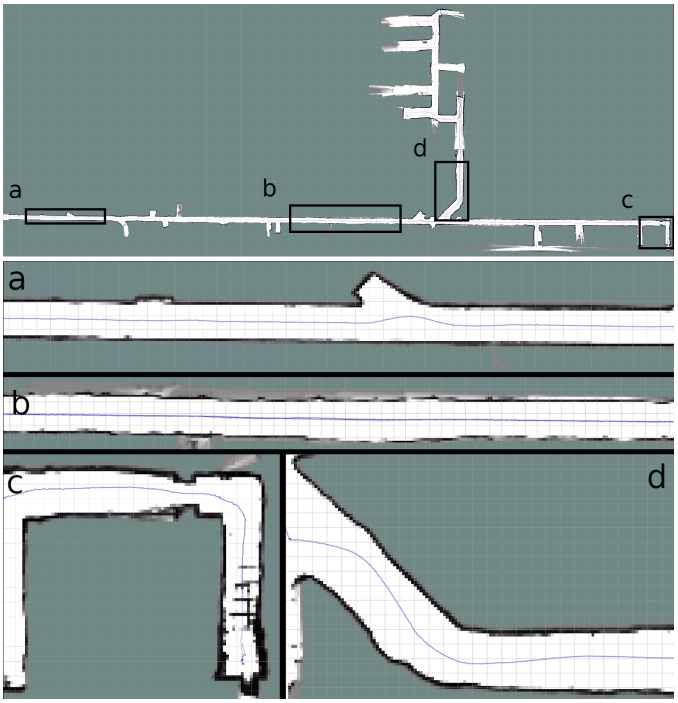

Fig. 3. Robot path histories and local occupancy grids from the Bruceton Experimental Mine. (a) demonstrates the robot's centering performance in a topologically linear environment with local perturbations. (b) shows the robot path history over the same corridor after three different traversals. This demonstrates the consistency of the wide-field control in both directions. (c) and (d) show controller performance in corners, bends, and constrained passages. Grid cell side lengths are 10 meters in the full map and one meter locally. The vehicle was commanded to travel at a speed of 0.8 m/s.

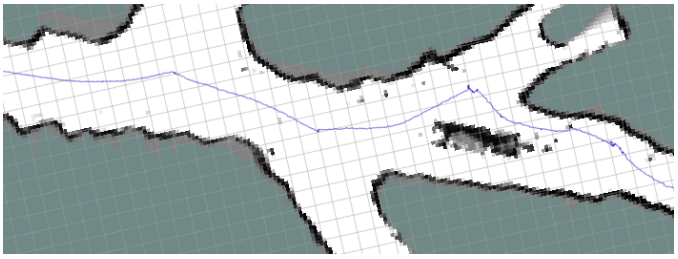

Fig. 4. Edgar Research Mine local control and topological planning performance in a cluttered topologically linear environment. The vehicle displayed smooth and consistent operation alternating between wide-field centering and local node controllers across multiple T-junctions and locally narrow corridors.

lead to anything we felt confident in deploying. Many solutions require a continuous stream of accurate state estimates of the vehicle. This is particularly hard to accomplish in sensory-deprived environments like a cave where GPS is unavailable. State estimation methods such as Visual-Inertial SLAM are often used in place of GPS, but they perform inconsistently in subterranean environments where the lighting is poor and specular highlights dominate the visual field. Finding a solution that relied on a minimal number of sensors, particularly sensors whose performance is unaffected by light, and had minimal dependence on state estimation became a priority after initial testing. In our solution only a single scanning depth sensor, the RPLIDAR A3, is needed for generating the low-level control commands for navigation between junctions or during exploration.

While the low-level steering controller does not require explicit state estimation, the planning strategy does. For this purpose, we used a LIDAR-based SLAM system with an on-board IMU. Depending on the distance between topological nodes, our system only requires a map and position update every few seconds. This is especially useful in large underground environments where walls can be long and featureless and loop closures require tens of seconds to calculate. The largest shortcoming of the planning system is its unpredictable output when presented with open environments that do not readily resemble a graph. The system may still work in these environments, but it is unlikely that the robot will follow the medial axis edge paths when using reactive control.

In addition to dependencies on state estimates, many path planning solutions are computationally expensive, and require significant compute to generate even simple, straight trajectories. The proposed low-level control method is extremely computationally simple, as the core operation of computing the Fourier series coefficients only requires a few thousand floating point operations depending on the number of points in the depth scan.

Planning a path, like when using the open source package Btraj [29], requires a choice of a start and goal point as well as an environment to plan through. Choosing intelligent goal points and generating a continuous representation of the environment can be computationally expensive processes. If the environment in between the start and goal point is too complex, the solver may be unable to generate a viable trajectory within the limits of imposed constraints. Tuning constraints on a trajectory is a nontrivial process, and a solution is not guaranteed for every environment. The proposed control law has a very simple pipeline for quickly and efficiently generating control commands using solely the instantaneous raw sensor data. Tuning the steering controller gains for the proposed controller is a simple and intuitive process.

VI. Future Work

Our work focused on ground vehicles in a relatively 2D environment. Potential follow-on work in 3D spaces, particularly those where paths traverse each other vertically, is of great interest. In addition, larger, more open environments present additional challenges to the centering controller and graph-planning techniques. Aerial vehicles such as quadrotors introduce additional dynamics such as 3D motion and less stable sensors. Finally, multi-robot cooperative control using our techniques could improve exploration efficiency.

Acknowledgment

This work was supported through the DARPA Subterranean Challenge, cooperative agreement number HR0011-18-2-0043.

REFERENCES

[1] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151, Monterey, CA, 1997.

[2] Reid Simmons, David Apfelbaum, Wolfram Burgard, and Dieter Fox. Coordination for multi-robot exploration and mapping. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 852–858, Austin, TX, 2000.

[3] S. J. Moorehead, R. Simmons, and W. L. Whittaker. Autonomous exploration using multiple sources of information. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 3098–3103, Seoul, Korea, 2001.

[4] Benjamin Charrow, Sikang Liu, Vijay Kumar, and Nathan Michael. Information-theoretic mapping using Cauchy-Schwarz Quadratic Mutual Information. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4791–4798, Seattle, Washington, 2015.

[5] E. Palazzolo and C. Stachniss. Information-driven autonomous exploration for a vision-based MAV. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,*, 4(2W3):59–66, 2017.

[6] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon path planning for 3D exploration and surface inspection. *Autonomous Robots*, 42(2):291–306, 2018.

[7] Christian Potthast and Gaurav S. Sukhatme. A probabilistic framework for next best view estimation in a cluttered environment. *Journal of Visual Communication and Image Representation*, 25(1):148–164, 2014.

[8] Henry Carrillo, Philip Dames, Vijay Kumar, and José A. Castellanos. Autonomous robotic exploration using a utility function based on Rényi's general theory of entropy. *Autonomous Robots*, 42(2):235–256, 2018.

[9] Andrew J. Smith and Geoffrey A. Hollinger. Distributed inference-based multi-robot exploration. *Autonomous Robots*, 42(8):1651–1668, 2018.

[10] Gregory Dudek, Michael Jenkin, Evangelos Milios, and David Wilkes. Robotic Exploration as Graph Construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, 1991.

[11] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[12] Kai M. Wurm, Cyrill Stachniss, and Wolfram Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1160–1165, Nice, France, 2008.

[13] Sebastian Thrun, Scott Thayer, William Whittaker, Christopher Baker, Wolfram Burgard, David Ferguson, Dirk Hahnel, Michael Montemerlo, Aaron Morris, Zachary Omohundro, Charlie Reverte, and Warren Whittaker. Autonomous exploration and mapping of abandoned mines: Software architecture of an autonomous robotic system. *IEEE Robotics and Automation Magazine*, 11(4):79–91, 2004.

[14] Richard Bormann, Florian Jordan, Wenzhe Li, Joshua Hampp, and Martin Hägele. Room segmentation: Survey, implementation, and analysis. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1019–1026, Stockholm, Sweden, 2016.

[15] Helen Oleynikova, Zachary Taylor, Roland Siegwart, and Juan Nieto. Sparse 3D Topological Graphs for Micro-Aerial Vehicle Planning. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 8478–8485, Madrid, Spain, 2018.

[16] Peter Brass, Flavio Cabrera-Mora, Andrea Gasparri, and Jizhong Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4):707–717, 2011.

[17] Michael Hoy, Alexey S. Matveev, and Andrey V. Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33(3):463497, 2015.

[18] Mark A. Frye and Michael H. Dickinson. Fly flight a model for the neural control of complex behavior. *Neuron*, 32:385–388, 2001.

[19] Borst A and Juergen Haag. Neural networks in the cockpit of the fly. *Journal of comparative physiology. A, Neuroethology, sensory, neural, and behavioral physiology*, 188:419–37, 08 2002.

[20] Mandyam V. Srinivasan, Javaan Singh Chahl, Keven Weber, Svetha Venkatesh, Martin G. Nagle, and Shao-Wu Zhang. Robot navigation inspired by principles of insect vision. *Robotics and Autonomous Systems*, 26:203–216, 1998.

[21] Mandyam V. Srinivasan and Shaowu Zhang. Visual motor computations in insects. *Annual Review of Neuroscience*, 27(1):679–696, 2004. PMID: 15217347.

[22] Jose Santos-Victor and Giulio Sandini. Embedded visual behaviors for navigation. *Robotics and Autonomous Systems*, 19:299–313, 1997.

[23] Stephen Griffiths, Jeff D. Saunders, Andrew Curtis, D. Blake Barber, Timothy W. McLain, and Randy Beard. Maximizing miniature aerial vehicles obstacle and terrain avoidance for mavs. 2006.

[24] James Sean Humbert and Andrew Maxwell Hyslop. Bioinspired visuomotor convergence. *IEEE Transactions on Robotics*, 26:121–130, 2010.

[25] Michael Ohradzansky, Hector E. Alvarez, Jishnu Keshavan, Badri Ranganathan, and J. Sean Humbert. Autonomous bio-inspired small-object detection and avoidance. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, 2018.

[26] Hector E. Alvarez, Michael Ohradzansky, Jishnu Keshavan, Badri Ranganathan, and J. Sean Humbert. Bio-inspired approaches for small-object detection and avoidance. *IEEE Transactions on Robotics*, 2019.

[27] TY Zhang and Ching Y Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.

[28] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE, 2016.

[29] Fei Gao, William Wu, Yu C. Lin, and Shaojie Shen. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 344–351, 2018.