A Robust and Fast Occlusion-based Frontier Method for Autonomous Navigation in Unknown Cluttered Environments

Nicholas Mohammad and Nicola Bezzo

***Abstract*— Navigation through unknown, cluttered environments is a fundamental and challenging task for autonomous vehicles as they must deal with a myriad of obstacle configurations typically unknown a priori. Challenges arise because obstacles of unknown shapes and dimensions can create occlusions limiting sensor field of view and leading to uncertainty in motion planning. In this paper we propose to leverage such occlusions to quickly explore and cover unknown cluttered environments. Specifically, this work presents a novel occlusion-aware frontier-based approach that estimates gaps in point cloud data and shadows in the field of view to generate waypoints to navigate. Our scheme also proposes a breadcrumbing technique to save states of interest during exploration that can be exploited in future missions. For the latter aspect we focus primarily on the generation of the minimum number of breadcrumbs that will increase coverage and visibility of an explored environment. Extensive simulations and experiment results on an unmanned ground vehicle (UGV) are demonstrated to validate the proposed technique, showing improvements over traditional state of the art frontier-based exploration methods.**

I. Introduction

Autonomous exploration and mapping of unknown, cluttered environments is one of the most active areas of research in robotics with far reaching applications. Robots with these capabilities can be leveraged in inspections [1], surveillance [2], search and rescue [3], and even household applications like robotic vacuum cleaning. A critical component in each of these tasks is that maps of the environment are built either before or during exploration in order to assist with path planning and keep track of where the robot has been and has yet to go.

In order to perform such exploration operations, the robot is equipped with range and vision sensors like lidar, IR, sonar, and RGBd cameras that create point cloud data to help build maps of the environment and navigate around obstacles. These sensors' fields of view, however, are often occluded by objects in the environment, reducing the robot's visibility and thus reducing the speed at which exploration occurs, in many cases also limiting the complete coverage in complex environments. For example, consider a vehicle deployed in a dense, heavy forested area tasked to map the environment for search and rescue purposes or to find an item of interest, or deployed in a warehouse tasked to clean, or inspect and survey the area. Occlusions created by different types of obstacles surrounding the robot can create several unknowns restricting the possible reachable regions. If the robot could infer and extract information about the expected environment around such occlusions, it could increase its performance, in particular its ability to cover the environment. Current state-of-the-art approaches rely on using known free space within the map built at every step to generate waypoints, however, and occlusions in cluttered environments greatly reduce the number of candidate waypoints that can be generated. We note here that such occlusions often hide accessible regions that, if leveraged, can increase the throughput of the robot during exploration tasks. This work builds around this idea, that is, *to design a method to reason about occlusions and infer and extract useful information about expected hidden traversable regions to generate waypoints that can lead to faster map coverage updates*. As an intuition, for example, large jumps in data between any consecutive point cloud data in a lidar measurement typically indicate the presence of a traversable area like a corridor, as shown in Fig. 1. Similar considerations can be made for shadowed regions behind obstacles which can be assumed to be reachable by the vehicle.

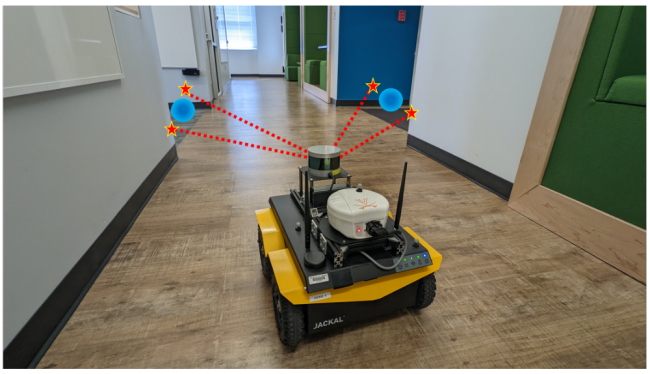


Fig. 1. Pictorial representation of the problem covered in this work in which traversable regions (blue spheres in the figure) are inferred from range sensor readings (dashed red lines) around occlusions to create frontier waypoints to be used by the robot during exploration of unknown environments

Finally we note that exploration can be leveraged beyond just the purpose of building a map. As a robot explores an unknown environment, it may discover and save states of interest, for example regions of higher visibility, regions that were difficult to navigate and thus should be avoided in the future, or regions that contain some optimal properties. These states of interest can be thought of as virtual breadcrumbs that a robot drops as they get discovered, which robots in future missions may leverage to help complete tasks like search and rescue or inspections faster than they would be able to without them.

With these motivations in mind, in this paper we propose a novel exploration method for autonomous robots in which frontier points are inferred and selected based on information extracted from sensor data patterns around occlusions. The three main cases considered at runtime are *gaps* in range sensor data, *shadows* in the sensor field of view, and *open space* frontier waypoints. By considering these cases we demonstrate that a robot can cover a completely cluttered

Nicholas Mohammad and Nicola Bezzo are with the Departments of Computer Science, Engineering Systems and Environment, and Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904, USA {nm9ur, nbezzo}@virginia.edu

environment faster than state of the art frontier-based exploration methods. We also propose a breadcrumbing method to facilitate exploitation of these explored environments. Specifically in this work we focus on breadcrumbing to increase visibility and propose a solution to solve the so called watchman tour problem, where a shortest route is found such that every point in the environment is visible from at least one point along the route [4]. Our approach solves this problem *online* by utilizing a greedy maximum coverage algorithm based on the geometry of saved sensor data at strategic positions during exploration.

The contribution of this work is two-fold: 1) **Exploration:** a robust occlusion-based exploration path planner that enables a robot to quickly map an unknown, cluttered environment by leveraging occluded regions of space, and 2) **Exploitation:** a maximization algorithm for the online generation of approximate watchman tours of an environment.

The remainder of this paper is organized as follows: in Section II, we provide an overview of related work in both exploration path planning and watchman tour generation. In III we outline the mathematical notation used in this work and formulate the problem of map coverage and watchman tour generation. The proposed path planner and tour generation frameworks are presented in Section IV and V respectively and are tested with extensive simulations and experiments in Sections VI and VII. Lastly, we draw conclusions and discuss future work in Section VIII.

II. Related Work

A large body of work is available on autonomous exploration of unknown environments. Over the years, two popular strategies have been developed: *frontier-based* and *sampling-based* navigation.

The first approach utilizes the notion of frontiers, which are intermediate regions between known and unknown spaces in a map. At each map update, a detection algorithm is run to detect new frontiers for use in the next planning step. In the formative work of [5], frontiers are detected within an occupancy grid map, and the closest is targeted as the next waypoint for the vehicle. In contrast, [6] aims to maximize velocity by selecting frontiers closest to the camera frustum.

In sampling based exploration frameworks, the goal is to sample poses which could grow the known regions of a map. A major benefit in using such sampling based approaches is that they remove the need to perform expensive global frontier detection algorithms at every map update and allow any desired utility definition to be used for pose selection [7]. [8] leverages the Next-Best-Views (NBV) model [9] to sample views which aim to maximize a utility function based on volumetric gain and time-of-flight for the vehicle. The NBV waypoints are generated by growing a Rapidly-exploring Random Tree (RRT) to sample positions and yaws from the configuration space of the vehicle [10]. [11] and [12] combine both sampling and frontier based approaches to rapidly map an environment. Despite the success of sampling-based techniques, they can still get stuck in local minima (e.g., dead-ends). [13] provides a history-based technique to mitigate these limitations.

Despite the accomplished work in exploration path planning, less progress has been made on occlusion-based navigation. [14] discusses optimal path planning when obstacles in the environment are occluded by other closer or larger obstacles. Similarly, [15] and [16] utilize occluded lidar regions to generate gaps to assist in navigation through unknown, cluttered environments. Given that these approach only consider static obstacles, [17] leverages an MPC-based approach for safe navigation in cluttered environments in which dynamic obstacles may be occluded by other obstacles. While these works consider the occlusions within unknown environments, they only focus on goal-to-goal settings and don't attempt to solve the problem of map coverage.

In this paper we also propose a breadcrumbing technique for path planning as an environment is explored. Related to this topic, we find a few works mostly centered on increasing wireless connectivity. [18] uses wireless beacons as breadcrumbs, allowing a UAV to localize itself during navigation to goal locations within an environment. However, the breadcrumbs must be placed manually by a human prior to navigation. [19] deploys wireless communication nodes autonomously in GPS denied underground environments by leveraging convolutional neural networks to assist in optimal placement for maximum coverage, allowing nearly full network coverage with a minimal number of nodes. While both of these works leverage breadcrumbs to assist in navigation, they don't use them for the purpose of speeding up map coverage. Lastly, [2] uses their *City-CNN* approach to find vantage points for area coverage, but their approach is not easily transferable to vehicles with limited FOV sensors and they don't generate shortest paths to navigate between vantage points.

III. preliminaries

A. Notation

In this paper we denote vectors with bold, italic letters (e.g., $\boldsymbol{x}$) and sets with upper case greek and calligraphic letters (e.g, Ω and $\mathcal{S}$). We use the indicator $\mathbb{I}_{\mathcal{S}}(b(s))$ to denote if there exists $s \in \mathcal{S}$ which satisfies the boolean condition b. Given a set $\mathcal{S}$ denoting a region of space, we use $|\mathcal{S}|$ to denote the area of $\mathcal{S}$. A $\wedge$ symbol on top of a variable represents its estimated value (e.g. $\hat{\boldsymbol{x}}$). $||\cdot||$ represents the Euclidean norm and the robot state is denoted by $\boldsymbol{x} = (\boldsymbol{p}_u, \theta)$ where $\boldsymbol{p}_u = [x, y]' \in \mathbb{R}^2$ refers to the position. $\boldsymbol{z}$ denotes the sampled point cloud distances and is indexed as $\boldsymbol{z}_i$. The corresponding point to $\boldsymbol{z}_i$ is referred to as $\boldsymbol{p}_i \in \mathbb{R}^2$. poly($\boldsymbol{z}$) takes in point cloud data and returns the polygon defined by those points. Lastly, abs$(\cdot)$ denotes the absolute value.

B. Problem Formulation

The research discussed in this work can be split into two main problems:

Problem 1: ***Fast Map Coverage:*** Given an a priori unknown cluttered environment $\mathcal{W}$ with N obstacles in which a region $\mathcal{M} \subseteq \mathcal{W}$ is traversable by a robot, the problem of map coverage is a multi objective optimization problem to find a policy which completely covers $\mathcal{M}$ while also trying to minimize the time needed to cover and map $\mathcal{M}$. Formally, let $\boldsymbol{x}$ denote the pose state of the robot (e.g., $\boldsymbol{x} = (x, y, \theta)$, the positions and angles for a ground vehicle configuration) and $M(t)$ be the map covered by the robot at time t. The fast map coverage problem is then defined as finding the control

policy $\mathcal{U}(t)$ to minimize the total time T to cover the entire space such that the following constraint is satisfied:

$$\left| \mathcal{W} \setminus \bigcup_{t=t_0}^{T} M(t) \right| < \epsilon$$

where ϵ is an arbitrarily small threshold and $\setminus$ is the set-minus operation. The total covered map after this operation is $M_T = \bigcup_{t=t_0}^{T} M(t)$.

This problem is solved by considering a modified frontier-based exploration method that reasons about consecutive data in point cloud measurements, inferring environmental properties behind occlusions.

Problem 2: ***Watchman Tours Generation:*** Finding a shortest distance trajectory $x^*_{0:T}$ such that every point in $\mathcal{M}$ is in line of sight is known as the watchman tour problem. Let M_T be the total map generated from time 0 to T after solving Problem 1. The objective of the watchman tour problem is to minimize the total travelled trajectory $x^*_{0:T}$ to cover M_T.

We solve this problem by monitoring and recording the robot's state along with its corresponding sensor readings during the exploration step in the previous problem.

IV. Occlusion-Based Exploration Path Planner

In this section, we describe our proposed path planner for fast exploration of unknown, cluttered environments. The diagram in Fig. 2 summarizes the architecture of our framework. We first take as input 2D point cloud sensor readings $\boldsymbol{z}$ and generate waypoints which allow the robot to cover the entire environment (i.e., there will be no more reachable frontiers).

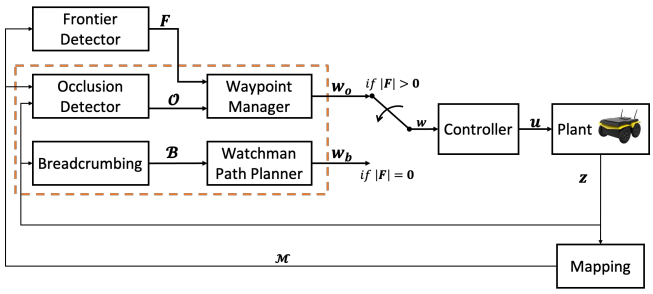


Fig. 2. Block diagram of proposed approach. The contributions of this paper are within the orange box.

The waypoints can be broken down into three types based on the three main situations a robot can encounter while exploring an unknown environment: *gap occlusions*, *shadow occlusions*, and *frontiers*. Gap occlusions are measured by discontinuities in contiguous point cloud samples as depicted in Fig. 3(a) while shadow occlusions are measured by the occluded region behind an obstacle, shown in Fig. 3(b). Lastly, frontiers are the points in open space which lie on the border of explored space by the robot [5].

As these waypoints are constructed, they are saved and evaluated based on criteria like the robot's state and waypoint position. In the following sections, we describe in detail each component of our framework depicted in Fig. 2, starting with occlusion detection.

A. Gap Occlusion Detection

The primary waypoint we consider in our approach is the gap occlusion, which encourages the robot to consider regions of space between two obstacles, where a distant obstacle is being partially occluded by a closer obstacle (see Fig. 3(a)). These gap occlusions can be defined as a region of space located between a large gap in two contiguous point cloud readings, $\boldsymbol{z}_i$ and $\boldsymbol{z}_{i+1}$.

Each gap occlusion can be represented as a ball centered at $\boldsymbol{p}_o = \frac{\boldsymbol{p}_i + \boldsymbol{p}_{i+1}}{2}$ with radius $r_o = \tau_o ||\boldsymbol{p}_i - \boldsymbol{p}_{i+1}||$, where τ_o is a tunable parameter used to account for noise in sensor data and prior knowledge about the size of obstacles.

Given sensor readings $\boldsymbol{z}$ with cardinality $\#\boldsymbol{z}$, the set of gap occlusions $\mathcal{G}$ is defined as:

$$\mathcal{G} = \{(\boldsymbol{p}_o, r_o) \mid \text{abs}(\boldsymbol{z}_i - \boldsymbol{z}_{i+1}) > \delta_g \; \forall i \in [1, \#\boldsymbol{z}]\} \quad (1)$$

Since occluded regions are considered traversable depending on the size of the robot, the tuning parameter, δ_g, is introduced which controls the minimum distance to be considered a gap occlusion. Given that (1) should only detect gaps that the vehicle could navigate through, δ_g should be set at least to the width of the robot. Fig. 4 shows the effect of δ_g on gap detection. Smaller gaps that the vehicle can't navigate through don't generate an occlusion, while the larger gaps do. Even though the point cloud readings all belong to the same obstacle, the occlusions are still published since the robot has no way to know if the perceived gaps are valid or not until it navigates closer.

Fig. 5(a) shows a case where the vehicle is located near a long, narrow corridor that it cannot fit through, but a gap occlusion is still published. In Fig. 5(a), A and B represent $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ respectively, where $\boldsymbol{z}_i < \boldsymbol{z}_{i+1}$ and the gap between their measurement readings is $\delta_{AB} > \delta_g$. To solve this problem, a window-based approach is used to check if the gap is traversable. More formally, we define a window of points $\mathcal{L}^+$ with size κ as

$$\mathcal{L}^+ = \{||\boldsymbol{p}_{i+2} - \boldsymbol{p}_{i+1}||, \ldots, ||\boldsymbol{p}_{i+1+\kappa} - \boldsymbol{p}_{i+1}||\} \quad (2)$$

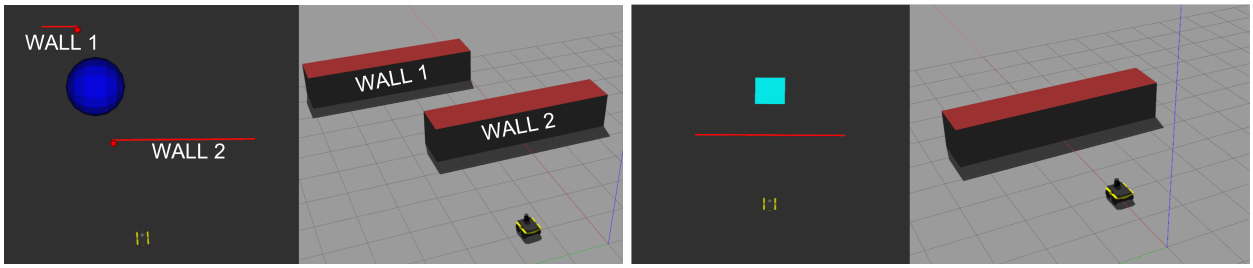
$\mathcal{L}^+$ is highlighted blue from point B to C in Fig. 5(b). Given $\mathcal{L}^+$, the boolean function Λ which filters gap occlusions as either valid or invalid can be defined as follows:

$$\Lambda(\mathcal{L}^+, \xi) = \mathbb{I}_{\mathcal{L}^+}(d < \xi) \quad (3)$$

If $\boldsymbol{z}_i < \boldsymbol{z}_{i+1}$ and there exists distance $d < \xi \in \mathcal{L}^+$, then the occlusion is not published. Fig. 5(b) shows that some point D in $\mathcal{L}^+$ is within distance ξ to $\boldsymbol{p}_i$ and thus the gap occlusion is not published. This process is mirrored in the case where $\boldsymbol{z}_i > \boldsymbol{z}_{i+1}$ by using $\mathcal{L}^-$ and $\Lambda(\mathcal{L}^-, \xi)$.

Continuing with validation of waypoint placement, it is desirable to prevent the planner from creating waypoints in already explored regions. To accomplish this, the occupancy grid map $\mathcal{M}$ is consulted to determine where the robot has yet to explore. For a cell $m_i \in \mathcal{M}$, it can fall into one of three classes depending on its value: 1) **Open:** $m_i < .5$; 2) **Unknown:** $m_i = .5$, and 3) **Occupied:** $m_i > .5$.

To prevent a gap occlusion g from being generated in explored space, the occupancy grid is consulted to ensure that the ratio of known to unknown cells around g is below some tunable threshold ψ_g. Let $A_g \subset \mathcal{M}$ denote the bounding box



(a) Gap occlusion (b) Shadow centroid

Fig. 3. Demonstrating the environment conditions responsible for the creation of specific waypoints in the proposed approach. Fig. 3(a) demonstrates a large discontinuity in point cloud data responsible for creating a gap occlusion. Fig. 3(b) shows an obstacle casting a shadow from the range sensor, which generates a shadow occlusion behind the obstacle.

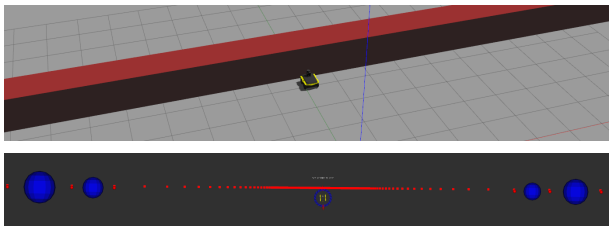
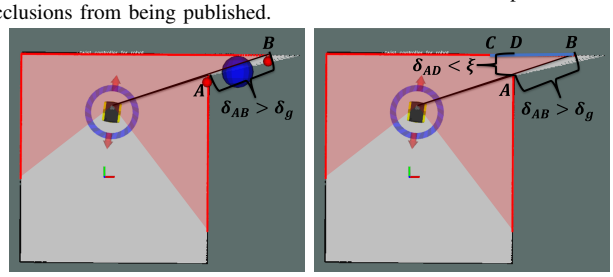


Fig. 4. Example of point cloud distance samples becoming sparse as the distance from the vehicle increases. A distance threshold prevents these occlusions from being published.



(a) (b)

Fig. 5. Leveraging a filtering window prevents the placement of erroneous gap occlusions. Fig. 5(a) shows a gap occlusion incorrectly being placed in a narrow corridor. Fig 5(b) shows that no gap occlusion is placed in the corridor when a distance-based filter (highlighted in blue) is applied.

of g, then the occlusion is valid only if the ratio of known cells in A_g to the area $|A_g|$ is below ψ_g. That is,

$$\frac{1}{|A_g|}\left[\sum_{m_i \in A_g, m_i < .5} (1 - m_i)\right] < \psi_g \tag{4}$$

B. Shadow Occlusion Detection

The second type of waypoint we consider in the path planner is the shadow occlusion, which is defined as the centroid of the region of space behind an obstacle in the environment (see Fig. 3(b)).

The shadow occlusion's formulation follows from the definition of an obstacle in the environment. Given $\boldsymbol{z}_i$, $\boldsymbol{z}_{i+1}$ and the tunable parameter α, the obstacle Ω is defined as

$$\Omega = \{\boldsymbol{p}_i \mid \text{abs}(\boldsymbol{z}_i - \boldsymbol{z}_{i+1}) < \alpha\} \tag{5}$$

In other words, an obstacle Ω is defined as a collection of points $\boldsymbol{p}_i$ that are no further than distance α from their adjacent points. If the points are too far apart, then they may belong to separate obstacles and thus a gap occlusion should be considered.

The planner keeps track of a set of these obstacles so long as they meet a size criterion. An obstacle Ω with two points does not produce an occluded region large enough to significantly obscure the point cloud sensor. Thus, a parameter β is introduced defining the minimum cardinality Ω. The obstacle set, then, is defined as

$$\mathcal{O} = \{\Omega \mid \#\Omega > \beta\} \tag{6}$$

From $\mathcal{O}$, the shadow occlusions are defined as the centroids of the occluded regions behind these obstacles. More formally, let ρ be a tunable parameter for how far behind an obstacle a centroid can be, then given the robot position $\boldsymbol{p}_u$, the coordinates for the shadow centroid $f(\Omega, \boldsymbol{p}_u)$ of obstacle Ω are defined as:

$$f(\Omega, \boldsymbol{p}_u) = \frac{1}{2(\#\Omega)}\left[\sum_{\boldsymbol{p} \in \Omega} \boldsymbol{p} + (\boldsymbol{p} - \boldsymbol{p}_u) * \rho\right] \tag{7}$$

Just as in the case with gap occlusions, it is possible for shadow occlusions to be detected in regions which have already been explored. To prevent such a centroid $\boldsymbol{c} = f(\Omega, \boldsymbol{p}_u)$ from being considered as a waypoint, we define $A_r \subset \mathcal{M}$ as the bounding box of the robot centered at $\boldsymbol{c}$ and ψ_c as a tunable threshold parameter. Equation (4) is used where A_r is used in place of A_g and ψ_c in place of ψ_c.

$$\mathcal{C} = \{f(\Omega) \mid \Omega \in \mathcal{O}\} \tag{8}$$

C. Frontier Detection

The last type of waypoint generated by the planner is the frontier, which is used to encourage the robot to explore open regions of the map.

In the occupancy grid $\mathcal{M}$, any open cell adjacent to an unknown cell is denoted as a frontier cell, and all adjacent frontier cells are then grouped together into frontier regions [5]. There are a number of approaches to determine these frontier regions, but the algorithm we use, wavefront detection, utilizes a double breadth first search on only newly mapped regions in $\mathcal{M}$ [20].

D. Occlusion Manager

At every time step the planner is generating new waypoints from the sensor readings $\boldsymbol{z}$ and the occupancy grid $\mathcal{M}$. These are sent to the occlusion manager, which decides when to add new waypoints and remove old ones.

Let a waypoint be defined as $\boldsymbol{w_i} = (x_i, y_i)$, $\mathcal{W}$ denote the set of waypoints already present in the waypoint manager, and $\mathcal{W}_t$ denote all waypoints detected at the current time step t. The first criterion the manager checks is to see if the vehicle has reached any waypoint in $\mathcal{W}$ and removes it if so.

The second criterion deals with safety and is considered for both old and new waypoints. If any waypoint $\boldsymbol{w} \in \mathcal{W} \cup \mathcal{W}_t$ is within a minimum distance threshold δ_s to an obstacle, it is discarded as a valid waypoint

The last criterion dictates that, if a new occlusion $\boldsymbol{w}_n \in \mathcal{W}_t$ is created within distance δ_d to any old occlusions $\boldsymbol{w}_o \in \mathcal{W}$, the old occlusions are removed. More formally, the update to $\mathcal{W}$ is as follows

$$\mathcal{W} \leftarrow \{\boldsymbol{w}_o \mid ||\boldsymbol{w}_n - \boldsymbol{w}_o|| > \delta_d \; \forall \boldsymbol{w}_o \in \mathcal{W}\} \tag{9}$$

Once all criteria have been checked for the current timestep, the final set of waypoints $\mathcal{W} \cup \mathcal{W}_t$ is generated.

E. Goal Selection

Provided with the waypoints in the occlusion manager, at each time step the planner generates a cost for each waypoint $\boldsymbol{w}$ based on the vehicle's position $\boldsymbol{p}_u$ and heading θ

$$\Gamma(\boldsymbol{w}) = \tau_D * ||\boldsymbol{p}_u - \boldsymbol{p}_w|| + \tau_H * \text{abs}(\theta - \phi(\boldsymbol{p}_u - \boldsymbol{p}_w)) \tag{10}$$

where $\phi(\cdot)$ is a function which gives the angle $\gamma \in [-\pi, \pi]$ from the origin of a passed in vector. Together with the tuning parameter τ_D, the distance term penalizes the waypoints which may be far away from the robot, therefore encouraging local exploration of a region before navigating elsewhere. Due to the distance between waypoints constantly changing as the robot navigates, it becomes possible for a different waypoint to be selected as the new goal at every time step, causing sporadic navigation. To mitigate this, the heading term, defined as the angle between the robot heading and a straight line connecting the robot and waypoint, was introduced along with its tuning parameter τ_H. This term provides incentive for the planner to select waypoints that minimize the need to turn.

Once navigation to all waypoints is complete, our approach will have left breadcrumbs which can be leveraged for future missions. The next section discusses the proposed breadcrumbing process in detail.

V. Breadcrumbing

In this section, we describe our framework for generating watchman tours at runtime during an exploration mission. Our proposed approach has three steps: 1) breadcrumbing, 2) max coverage optimization, and 3) tour generation. In the first step, the robot saves its position and sensor readings periodically during navigation as a breadcrumb. As these breadcrumbs are stored, a maximum coverage solver is run to find the approximately smallest set of breadcrumbs that cover the observed environment. Once the coverage set has been generated, an approximate traveling salesman algorithm is run to get a shortest-path order to visit the points.

A. Dropping Breadrumbs

A breadcrumb $\boldsymbol{b}_t$ is defined as a tuple of the robot's pose and the corresponding sensor readings at time t, as shown in Fig. 6. At each time step, the approach saves $\boldsymbol{b}_t$ into a breadcrumb set $\mathcal{B}$ if it satisfies two conditions. The first is a safety measure, where breadcrumbs are only recorded if they are farther than some minimum safe distance δ_o from an obstacle

$$\text{CA1: } \boldsymbol{z}_i > \delta_o \; \forall i \in [1, \#\boldsymbol{z}] \tag{11}$$

The second condition prevents $\mathcal{B}$ from being flooded while the robot is navigating in a small region of space. Formally, breadcrumb $\boldsymbol{b}_t$ with position $\boldsymbol{p}_t$ must be a minimum distance, δ_b, from all recorded breadcrumbs $\boldsymbol{b}_i \in \mathcal{B}$,

$$\text{CA2: } ||\boldsymbol{p}_i - \boldsymbol{p}_t|| < \delta_b \;\; \forall \boldsymbol{b}_i \in \mathcal{B} \tag{12}$$

If the new breadcrumb $\boldsymbol{b}_t$ is within δ_b of a crumb $\boldsymbol{b}_i$, then the one with highest sensor coverage is kept.

Given both conditions are satisfied, the breadcrumb $\boldsymbol{b}_t$ is ready to be added to $\mathcal{B}$ after a sensor reduction stage. Since the number of samples in $\boldsymbol{z}$ is large, recording it for every breadcrumb has high memory cost. Thus, the approach treats $\boldsymbol{z}$ as a visibility polygon $\mathcal{P} = \text{poly}(\boldsymbol{z})$. $\mathcal{P}$ is sent through a point reduction algorithm described in [21] to generate an estimate $\widehat{\mathcal{P}}$ with less samples, thus saving memory space at the cost of reducing measurement fidelity. The final breadcrumb $\boldsymbol{b}_t = (\boldsymbol{p}_u, \widehat{\mathcal{P}})$ is saved to $\mathcal{B}$. In order to keep

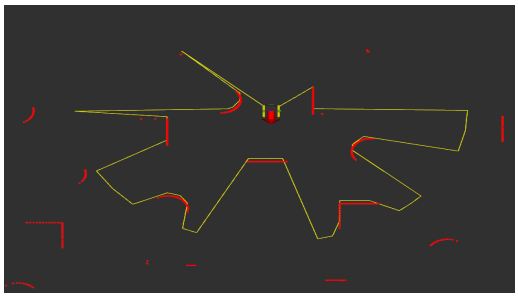


Fig. 6. Example of a breadcrumb. The pose is shown as a red arrow and the reduced sensor visibility is shown in yellow.

from running out of memory during large scale explorations, we limit the size of $\mathcal{B}$ to some maximum value N. To decide which breadcrumbs are removed when at capacity, we treat $\mathcal{B}$ as a cache with temporal locality. Deletion occurs at the end of $\mathcal{B}$, and any breadcrumb that was chosen for the optimal coverage set, $\mathcal{B}_t^*$, is moved to the front of $\mathcal{B}$. The assumption driving this decision is that a breadcrumb frequently used in $\mathcal{B}_t^*$ offers high area coverage or visibility into a secluded region and thus should be favored. In the following section, we discuss how $\mathcal{B}_t^*$ is found.

B. Finding Approximately Optimal Breadcrumbs

Since $\mathcal{B}$ contains all breadcrumbs recorded by the robot, the total explored region $\mathcal{M}_e$ can be defined as the union of polygons $\widehat{\mathcal{P}}$ of all breadcrumbs in $\mathcal{B}$

$$\mathcal{M}_e = \bigcup_{(\boldsymbol{p}_i, \widehat{\mathcal{P}_i}) \in \mathcal{B}} \widehat{\mathcal{P}_i} \tag{13}$$

Whenever a new breadcrumb is recorded, the approach recomputes the approximately minimal cardinality set of breadcrumbs $\mathcal{B}_t^* \subseteq \mathcal{B}$ such that

$$\left| \bigcup_{(\boldsymbol{p}_i, \widehat{\mathcal{P}_i}) \in \mathcal{B}_t^*} \widehat{\mathcal{P}_i} \right| = \zeta |\mathcal{M}_e| \tag{14}$$

That is, the area covered by $\mathcal{B}_t^*$ should be equal to a percentage ζ of the total explored area $|\mathcal{M}_e|$. We use ζ because, due to the complexity of environments, it may be that every breadcrumb in $\mathcal{B}$ is required to achieve full coverage, but

only a fraction are required to reach an acceptable coverage percentage.

In order to solve this problem, we leverage the fact that finding $\mathcal{B}_t^*$ is a maximum coverage problem where area coverage is to be maximized and the candidate sets are the visibility polygons $\widehat{\mathcal{P}}$ of each breadcrumb. Given that max coverage is an intractable problem, we use an approximate greedy algorithm which leverages the submodular, monotonic properties of area coverage. Doing so provides a solution where $|\mathcal{B}_t^*|$ is no larger than $\frac{e}{e-1}$ times the optimal, where e is the Euler's number [22].

The greedy algorithm works by starting with $\mathcal{B}_t^* = \emptyset$ and adding the breadcrumb $\boldsymbol{b}_i \in \mathcal{B}$ which generates the highest increase in coverage of $\mathcal{B}_t^*$ as computed in (14). Once $|\mathcal{B}_t^*| > \zeta * |\mathcal{M}_e|$, the algorithm terminates and a tour of the crumbs in $\mathcal{B}_t^*$ is ready to be generated.

C. Watchman Tour Generation

Provided with $\mathcal{B}_t^*$, the objective is to generate a tour through these breadcrumbs such that the robot achieves full coverage faster than during exploration. To accomplish this, 1) we cast the tour as a solution to a traveling salesman problem to determine the visiting order of breadcrumbs and 2) perform a route simplification to remove unnecessary points along the path

Since traveling salesman is an intractable problem, an approximation algorithm called two-opt [23] is used to compute the shortest route $\mathcal{R}$ which has path length no longer than $\sqrt{2}$ times the optimal. Once $\mathcal{R}$ is found, it is reduced by removing breadcrumbs deemed redundant. The reduction works by removing breadcrumbs that the robot will reach by navigating to other breadcrumbs. Let $\boldsymbol{b}_{i-1}$ and $\boldsymbol{b}_{i+1}$ denote the breadcrumbs before and after $\boldsymbol{b}_i$ in $\mathcal{R}$ respectively. There are three conditions that must be satisfied in order for $\boldsymbol{b}_i = (\boldsymbol{p}_i, \widehat{\mathcal{P}}_i)$ to be considered redundant. The first criterion checks if $\boldsymbol{b}_{i-1}$, $\boldsymbol{b}_i$, and $\boldsymbol{b}_{i+1}$ form roughly a straight line, meaning that the robot will pass close to $\boldsymbol{b_i}$ while navigating from $\boldsymbol{b}_{i-1}$ to $\boldsymbol{b}_{i+1}$. This is formulated as

$$\text{CC1: } \pi - \arccos\left(\frac{\boldsymbol{u}\cdot\boldsymbol{v}}{||\boldsymbol{u}||\,||\boldsymbol{v}||}\right) < \eta_1 \tag{15}$$

where $\boldsymbol{u} = \boldsymbol{p}_i - \boldsymbol{p}_{i-1}$, $\boldsymbol{v} = \boldsymbol{p}_i - \boldsymbol{p}_{i+1}$ and η_1 is a tunable threshold for the angle between $\boldsymbol{u}$ and $\boldsymbol{v}$.

Next, given the heading θ_i of $\boldsymbol{b}_i$, the second criterion checks if the robot will likely achieve that heading while navigating from $\boldsymbol{p}_{i-1}$ to $\boldsymbol{p}_{i+1}$. That is,

$$\text{CC2: } \theta_i - \arctan\left(\frac{\boldsymbol{w}_2}{\boldsymbol{w}_1}\right) < \eta_2 \tag{16}$$

where $\boldsymbol{w} = \boldsymbol{p}_{i+1} - \boldsymbol{p}_{i-1}$ and η_2 a threshold parameter. This constraint can be ignored if the vehicle has a 360° point cloud sensor, since the orientation of the breadcrumb no longer effects the sensor measurements.

The last condition is that the line l from $\boldsymbol{p}_{i-1}$ to $\boldsymbol{p}_{i+1}$ should not intersect any obstacles in $\mathcal{M}$. More formally, let $\mathcal{M} \cap l$ denote all occupancy map cells $m_i \in \mathcal{M}$ along line l, the criterion is

$$\text{CC3: } m_i < .5 \;\forall m_i \in \mathcal{M} \cap l \tag{17}$$

Given that all three conditions are true, the breadcrumb $\boldsymbol{b}_i$ is redundant and thus removed as a waypoint from the path. The process is repeated until no three consecutive breadcrumbs satisfy all three conditions, giving the final order to navigate through each breadcrumb.

The overall runtime for this breadcrumbing depends on both the number of breadcrumbs N_b in $\mathcal{B}$ and the number of samples N_s in the point cloud data $\boldsymbol{z}$. The bottleneck of this process is the unary union of visibility polygons. For worst case analysis, assume that each visibility polygon could not be reduced, meaning $\widehat{\mathcal{P}}$ has N_s vertices. Also assume that all N_b breadcrumbs were required to cover $\mathcal{M}_e$. The union is performed on the order of N_b^2 times and each union takes $O(N_s \log N_s)$ time, giving a final runtime of $O(N_b^2 N_s \log N_s)$. Since a theoretical upper bound is still unknown for two-opt, the tour generation is omitted from this runtime discussion.

VI. Simulations

Our approach was evaluated in simulation using three case studies: 1) a warehouse, 2) a large cluttered environment, and 3) a bookstore. All simulated experiments were performed in Gazebo using Ubuntu 16.04 and ROS Kinetic. The UGV testbed used in simulation is the Clearpath Jackal equipped with a 270° lidar with maximum range of $30m$. For path planning, the move_base[1] navigation stack was used along with GMapping[2] for SLAM. A maximum speed of 2 m/s and acceleration of 5 m/s^2 were set for the planner parameters. In all simulations, the watchman tour coverage was set to 99% and each breadcrumb could only use a truncated lidar scan with max range of $5m$. This was done because the map update was only performed in a $5m$ radius around the UGV. Finally, simulations were only terminated when no reachable waypoints are left. In the case of breadcrumb based navigation, the simulations were terminated once the UGV had navigated through all breadcrumbs in the tour.

A. Warehouse Case Study

Several simulations were performed in the Warehouse environment depicted in Fig. 7, where our approach was compared with a frontier-only exploration algorithm. From Table I, it can be observed that our occlusion-based framework can explore the environment significantly faster than the frontier-only approach over 3 runs. Furthermore, when navigating through the space utilizing breadcrumbs collected during initial exploration, we observe an additional reduction in distance traveled with no drop in exploration speed, giving large improvements to exploration time. In Fig. 8, it can be seen that our approach achieves a faster rate of exploration on average at all points of time when compared with frontier-only exploration.

TABLE I. Warehouse Exploration Results

Approach	avg. speed (m/s)	avg. distance (m)	avg. time (s)
frontiers	0.4	108.45	235.3
occlusions (ours)	**0.45**	83.37	171.3
breadcrumbs (ours)	**0.45**	**64.46**	**140.0**

[1]see http://wiki.ros.org/move_base
[2]see http://wiki.ros.org/gmapping

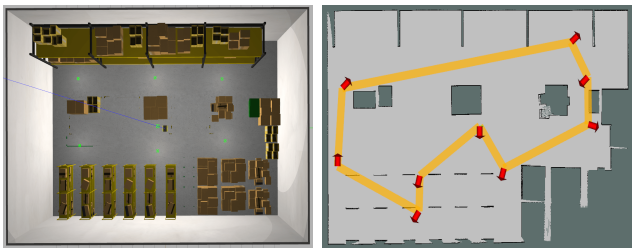


Fig. 7. Warehouse environment (left) and associated map with watchman tour (right).

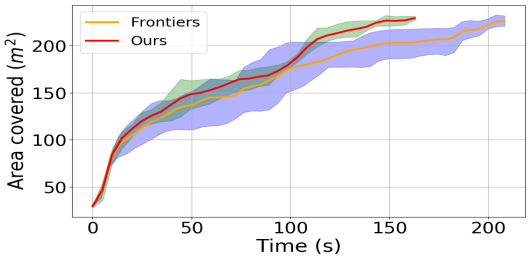


Fig. 8. Warehouse environment explored area over time.

B. Cluttered Environment Case Study

In order to evaluate our framework in a realistic outdoor environment, we created a $25m$ x $25m$ cluttered environment with cube and cylindrical obstacles (Fig. 9). The environment is meant to be a rough model of a dense forest, where each obstacle is akin to tree trunks and large rocks. As seen by Table II, our approach is further demonstrated to outperform the frontier-based approach, and performance is further improved when utilizing breadcrumbs in a second exploration mission to cover 99% of the environment. Fig. 10(a) shows the area covered over time when following the generated watchman tour. As can be seen, it only took 358 seconds to navigate through all breadcrumbs and achieve the desired coverage, which is much faster than both frameworks during initial exploration. Furthermore, Fig. 10(b) shows the area covered as a function of the number of breadcrumbs. As expected, coverage increases with the number of breadcrumbs. Additionally, there is a diminishing marginal increase in area as the number of breadcrumbs increases, which shows the submodularity of the coverage function as discussed in Section V-B.

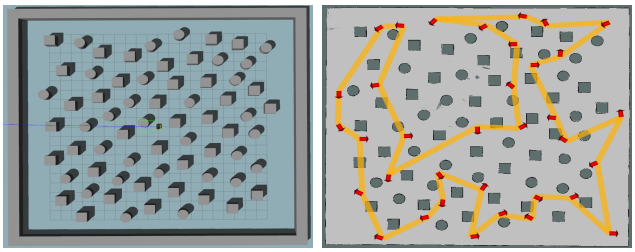


Fig. 9. Cluttered environment (left) and associated map with watchman tour (right).

C. Bookstore Case Study

Simulations were also performed in a bookstore environment (see Fig. 11) that is smaller than the warehouse, but more densely packed with obstacles, making it a good case study to test our approach. Table III shows the speed, distance, and time averaged over 3 runs for each navigation framework. In this environment, our approach achieved a slightly faster exploration speed while travelling less overall distance, thus completing exploration faster than the frontier framework. In the case of breadcrumb based navigation, we observe even faster exploration speeds while drastically reducing overall distance travelled when compared with both initial exploration missions.

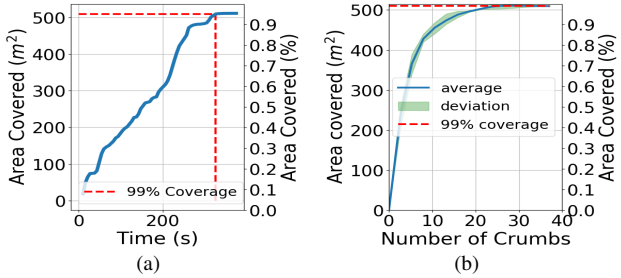


Fig. 10. Area covered over time for breadcrumb navigation (left) and area coverage vs number of breadcrumbs (right).

TABLE II. CLUTTERED ENVIRONMENT EXPLORATION RESULTS

Approach	avg. speed (m/s)	avg. distance (m)	avg. time (s)
frontiers	0.35	224.36	641.0
occlusions (ours)	**0.42**	211.79	504.3
breadcrumbs (ours)	0.35	**142.35**	**406.7**

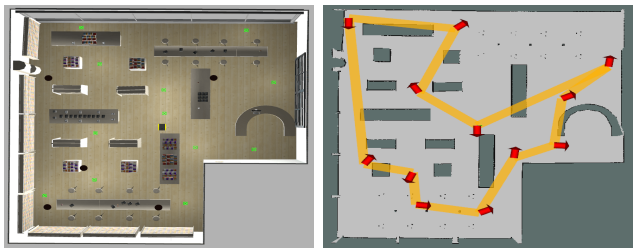


Fig. 11. Library environment (left) and associated map with watchman tour (right).

TABLE III. BOOKSTORE EXPLORATION RESULTS

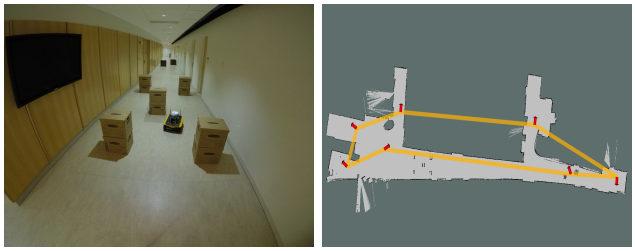
Approach	avg. speed (m/s)	avg. distance (m)	avg. time (s)
frontiers	0.29	99.11	277.0
occlusions (ours)	0.30	84.99	222.0
breadcrumbs (ours)	**0.32**	**62.83**	**152.0**

VII. EXPERIMENTS

Experimental validations were performed on a nearly identical test bed, with the exception that we use a Velodyne VLP16 lidar with 360° FOV and maximum range of $100m$ for sensing. We tested the approach inside the basement of a building and inside an office space. The resulting maps and corresponding watchman tours for both the basement and office experiments are shown in Fig. 12 and Fig. 13 respectively and the area covered over time is also shown for both in Fig. 14.

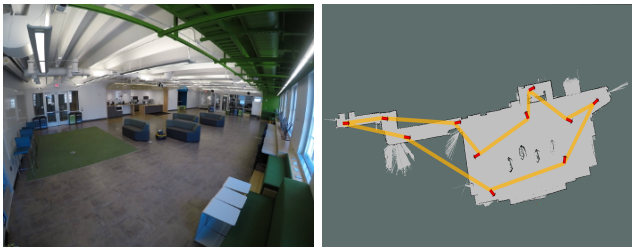
Note: Videos for these simulations and experiments are available in the provided supplemental material. More simulations and experiments are also available in the following

link: https://www.bezzorobotics.com/nm-iros22. The code for the proposed framework is available at https://github.com/UVA-BezzoRobotics-AMRLab/OcclusionBasedFrontierNavigation



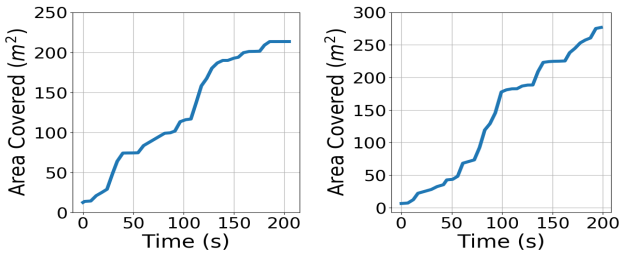
(a) (b)

Fig. 12. Basement setup (a) and associated map with watchman tour (b).



(a) (b)

Fig. 13. Office setup (a) and associated watchman tour (b).



(a) (b)

Fig. 14. Basement area vs time (a) and office area vs time (b)

VIII. Conclusions and Future Work

In this work, we have presented a novel exploration strategy that uses both frontier-based and occlusion-based frameworks that can also record important information in the form of breadcrumbs for use in future missions. Our approach demonstrates improvements upon frontier-only planners in terms of exploration speed and distance traveled as shown in the simulation studies. Furthermore, through experiments we show that our approach can also be used in the real world on-board a UGV.

From here, future work includes adding an entropy-based waypoint selection process to help improve exploration speeds. Furthermore, we would also like to utilize sparse raycasting at points sampled along the watchman tours to give better breadcrumb coverage approximations and further reduce the distance the robot must travel to reach desired coverage thresholds.

Acknowledgments

This work is based on research sponsored by DARPA under Contract No. FA8750-18-C-0090 and NSF under grant number #1816591

References

[1] P.-L. Richard, N. Pouliot, F. Morin, M. Lepage, P. Hamelin, M. Lagac©, A. Sartor, G. Lambert, and S. Montambault, "Lineranger: Analysis and field testing of an innovative robot for efficient assessment of bundled high-voltage powerlines," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9130–9136.

[2] L. Ly and Y.-H. R. Tsai, "Autonomous exploration, reconstruction, and surveillance of 3d environments aided by deep learning," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5467–5473.

[3] N. Mimmo, P. Bernard, and L. Marconi, "Avalanche victim search via robust observers," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4066–4072.

[4] W.-P. Chin and S. Ntafos, "Optimum watchman routes," in *Proceedings of the second annual symposium on Computational geometry*, 1986, pp. 24–33.

[5] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 1997, pp. 146–151.

[6] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2135–2142.

[7] T. Dang, C. Papachristos, and K. Alexis, "Autonomous exploration and simultaneous object search using aerial robots," in *2018 IEEE Aerospace Conference*, 2018, pp. 1–7.

[8] V. M. Respall, D. Devitt, R. Fedorenko, and A. Klimchik, "Fast sampling-based next-best-view exploration algorithm for a mav," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 89–95.

[9] C. Connolly, "The determination of next best views," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 432–435.

[10] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep., 1998.

[11] A. Brunel, A. Bourki, C. Demonceaux, and O. Strauss, "Splatplanner: Efficient autonomous exploration via permutohedral frontier filtering," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 608–615.

[12] A. Dai, S. Papatheodorou, N. Funk, D. Tzoumanikas, and S. Leutenegger, "Fast frontier-based information-driven autonomous exploration with an mav," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 9570–9576.

[13] C. Witting, M. Fehr, R. Bähnemann, H. Oleynikova, and R. Siegwart, "History-aware autonomous exploration in confined environments using mavs," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[14] L. Janson, T. Hu, and M. Pavone, "Safe motion planning in unknown environments: Optimality benchmarks and tractable policies," *CoRR*, vol. abs/1804.05804, 2018. [Online]. Available: http://arxiv.org/abs/1804.05804

[15] M. Mujahed and B. Mertsching, "The admissible gap (ag) method for reactive collision avoidance," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1916–1921.

[16] Z. Ullah, X. Chen, S. Gou, Y. Xu, and M. Salam, "Fnug: Imperfect mazes traversal based on detecting and following the nearest-to-final-goal and unvisited gaps," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5175–5182, 2022.

[17] J. Higgins and N. Bezzo, "Negotiating visibility for safe autonomous navigation in occluding and uncertain environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4409–4416, 2021.

[18] D. P. Grabowsky, J. M. Conrad, and A. F. Browne, "A breadcrumb system for assisting outdoor autonomous robots with path identification and localization," in *SoutheastCon 2021*, 2021, pp. 1–6.

[19] M. Tatum, "Communications coverage in unknown underground environments," *Masters's thesis, The Robotics Institute, Carnegie Mellon University, USA*, 2020.

[20] M. Keidar and G. A. Kaminka, "Efficient frontier detection for robot exploration," 2013.

[21] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, pp. 112–122, 1973.

[22] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions," in *Mathematical Programming*, 1978, pp. 1–30.

[23] G. A. Croes, "A method for solving traveling-salesman problems," *Operations research*, vol. 6, no. 6, pp. 791–812, 1958.