

TOPGN: Real-time Transparent Obstacle Detection using Lidar Point Cloud Intensity for Autonomous Robot Navigation

Kasun Weerakoon, Adarsh Jagan Sathyamoorthy, Mohamed Elnoor, Anuj Zore, and Dinesh Manocha.
Supplemental version including Tech Report, and Video at <http://gamma.umd.edu/topgn/>

Abstract—We present TOPGN, a novel method for real-time transparent obstacle detection for robot navigation in unknown environments. We use a multi-layer 2D grid map representation obtained by summing the intensities of lidar point clouds that lie in multiple non-overlapping height intervals. We isolate a neighborhood of points reflected from transparent obstacles by comparing the intensities in the different 2D grid map layers. Using the neighborhood, we linearly extrapolate the transparent obstacle by computing a tangential line segment and use it to perform safe, real-time collision avoidance. Finally, we also demonstrate our transparent object isolation’s applicability to mapping an environment. We demonstrate that our approach detects transparent objects made of various materials (glass, acrylic, PVC), arbitrary shapes, colors, and textures in a variety of real-world indoor and outdoor scenarios with varying lighting conditions. We compare our method with other glass/transparent object detection methods that use RGB images, 2D laser scans, etc. in these benchmark scenarios. We demonstrate superior detection accuracy in terms of F-score improvement at least by 12.74% and 38.46% decrease in mean absolute error (MAE), improved navigation success rates (at least two times better than the second-best), and a real-time inference rate (~ 50 Hz on a mobile CPU). We will release our code and challenging benchmarks for future evaluations upon publication.

I. INTRODUCTION

Modern buildings are filled with glass walls, full-height windows, or other transparent obstacles that are challenging for a robot to detect and autonomously navigate around. This is primarily because a majority ($> 90\%$) of the light energy incident on transparent objects gets transmitted through them [5] and only a small amount gets reflected for robot-mounted sensors (e.g. RGB, depth cameras, 2D/3D lidars) to detect. The inability of a robot to accurately perceive and avoid transparent obstacles in real-time could lead to serious collisions, which can result in damage to the robot and its environment [5], [6], [2], [3].

To address this challenge, there have been several methods for glass/transparent object detection using various sensor modalities such as RGB [2], [3], depth [7], [8], and thermal images [9], ultrasound [10], 2D laser scans [4], 3D point clouds [11], etc. Methods utilizing RGB images typically semantically segment transparent objects [2]. Some works have combined RGB images with the depth of objects in the scene, or with the thermal signature [9] of transparent objects to improve detection accuracy. Since RGB cameras and the images they capture are impacted by the environmental lighting conditions, the detection accuracy of RGB-based methods can deteriorate significantly in low-light or extremely bright conditions (see Fig. 1). This is

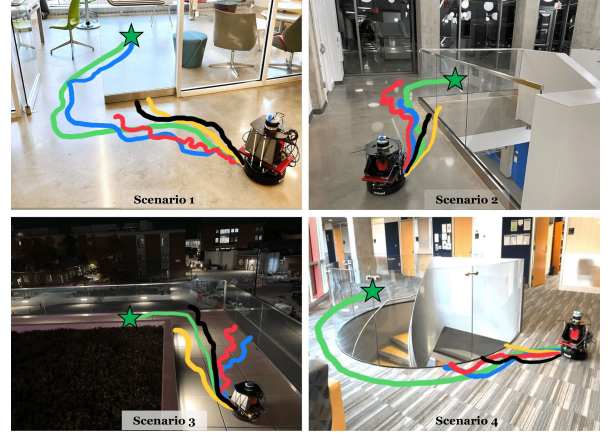


Fig. 1: Our method can robustly detect transparent obstacles in scenes with varying illumination in real-time (~ 50 Hz). The figure shows the robot’s trajectories in one trial when a planner [1] used our method (in green), GDNet [2] (in blue), Translab [3] (in red), Glass-SLAM [4] (in yellow), and 2D laser scans (in black) to detect transparent obstacles in unknown environments. RGB segmentation methods [2], [3] are affected by strong lighting changes, and motion blur in these environments, causing collisions with glass, or freezing during navigation. SLAM methods such as [4] require ~ 3 seconds to update the locally sensed obstacles on to a map, leading to collisions. Our method’s accurate detection of transparent and opaque obstacles facilitates safe, collision-free navigation in unmapped, unknown environments.

further exacerbated if the environment contains multiple light sources and highly reflective surfaces.

Conversely, methods that use 2D scans or 3D point clouds from lidars are resilient to external lighting changes, and excessive reflections from various surfaces since lidars have an in-built light source. However, most of the light from lidars passes through transparent obstacles and $< 10\%$ of the intensity is reflected to the sensor at certain angles [5], making detection of the shape and size of the obstacle challenging. In addition, lidar-based methods [5], [4], [6], [12] have been mainly limited to mapping tasks that typically require a robot to perform several loops in the environment to detect and map transparent obstacles. This makes them impractical for real-time navigation and collision avoidance.

Main Contributions: We present TOPGN (Transparent Obstacle Perception for Guidance and Navigation), a novel approach to accurately detect transparent obstacles in the environment, and extrapolate (predict regions where transparent obstacles could be present) them in 2D grid maps to avoid collisions preemptively. The novel components of our work include:

- A novel method to accurately isolate *transparent obstacle neighborhoods* (TONs) that arise in a multi-layer 2D grid map representation [13], [11], [14], [15], [16] that are computed from 3D lidar point clouds. Our approach is based on identifying a Gaussian pattern exhibited by point cloud intensities while they strike transparent obstacles. Our approach has a low computational overhead and can execute real-time on mobile CPUs at ~ 50 Hz, and can also be used to create a 2D map of an environment containing transparent obstacles. Further, we observe superior transparent obstacle detection accuracy compared to existing RGB-based and lidar-based methods (at least 12.74% improvement in F-score compared to the second-best performing method).
- A novel method to linearly extrapolate transparent obstacles from an instance of a TON such that the regions that could potentially contain transparent obstacles are encompassed. The robot trajectories that avoid extrapolation are guaranteed to avoid transparent obstacles in completely unknown environments. Our method can handle curved transparent obstacles, and we demonstrate that our method leads to superior navigation success rates (at least 2 times better than the second-best method) in reaching the robot's goal.
- We implement our method on a real Turtlebot equipped with a Velodyne VLP16 lidar. We perform extensive real-world evaluation in challenging indoor and outdoor scenarios with severe lighting changes, reflective surfaces, and transparent obstacles with curved shapes. We create and also release our test benchmarks with these scenarios that contain RGB images, and 2D grid maps with transparent obstacle annotations. We demonstrate that our approach is robust in all these scenarios while existing methods deteriorate in terms of detection accuracy and lead to collisions.

II. RELATED WORK

In this section, we provide an overview of methods that use point cloud intensities, or some kind of images (RGB, depth, thermal) to perceive transparent obstacles.

A. Transparent Obstacle Detection using Lidars and Point Clouds

There have been several approaches [4], [5], [17], [6] that use the intensity of lidar point clouds to detect glass and other transparent obstacles for SLAM (Simultaneous Localization And Mapping) applications. One of the earliest works in this domain is by Foster et al. [5], who proposed to track the angles of incidence on glass from which a lidar's laser rays are reflected with maximum intensity. They identified that $\sim 0^\circ$ incidence leads to high-intensity returns back to the lidar. In a recent extension, Foster et al. [12] proposed a more general approach that constructs a Reflectance Field Map (RFM) based on the reflectance of points from different perspectives. Next, they identify a distinct H-pattern that transparent obstacles cause in the RFM. [12] can robustly map glass in real-time even when the lidar is disturbed

by bumps and suspension loading, and in the presence of dynamic pedestrians.

Subsequent works such as [4] use this theory to recognize the reflected light intensity profile on the glass to detect it, and construct a map using the particle filter. However, it requires several walk-throughs in the environment to map glass, and the resulting map could still miss some portions of glass. Weerakoon et al. [6] improved this accuracy of map building using Graph SLAM. Tibebe et al. [18] identified the changes in the distance and intensity measurements between neighboring point clouds to estimate the glass profile. Other works such as Wei et al. [17] have augmented the distance (from the obstacle) information obtained from a lidar with the distance from ultrasound sensors to map environments with glass. However, such methods are limited by the short range of ultrasound sensors making them unsuitable for real-time collision avoidance.

Additionally, such SLAM methods cannot be directly used for real-time navigation in unknown environments since they require several seconds to construct the obstacles in the map. Our approach is based on the specular reflection at 0° incidence but extrapolates transparent obstacles in real-time for navigating unknown, unmapped environments with transparent obstacles. Further, we demonstrate our approach's generality by also demonstrating its applicability in real-time mapping.

B. Transparent Obstacle Detection using Images

There has been extensive work on glass and transparent object segmentation in RGB, depth, and thermal images. Huang et al. [10] developed a wearable setup with a depth camera and ultrasound sensors to improve the depth measurement accuracy for glass detection to guide the visually impaired in real time. However, due to the low range and fields-of-view of these sensors, they cannot be reliably used for robot navigation.

More recently, GDNet [2], [19] released a large-scale glass detection dataset and proposed a semantic segmentation method for detecting large-sized glass from RGB images using the contextual features from a large receptive field. Similarly, TransLab [3] proposed using boundary cues as a means to improve large and small transparent objects. Lin et al. [20] overcame the inaccuracies in GDNet and TransLab models (e.g. confusing open spaces as glass) by adding a module to refine glass detection by identifying reflections. This was later extended by integrating the missing depth data from glass in a depth image to detect the presence of glass surfaces [7]. Other methods [9] have fused RGB with thermal images by using the fact that thermal energy is blocked by transparent objects while visible light passes through.

The challenge posed by transparent objects has also been studied in regards to stereo matching [21], object reconstruction and manipulator grasping [22], [23], [24], [25], [8], [26], [27]. All these methods use depth from a time-of-flight depth camera [22], a stereo camera [23], or fuse the slight discoloration observed in RGB images with depth information [25] to detect graspable small 3D objects, assess

their position and orientation, and plan a way to grasp them. Using neural radiance fields [26] and additional lights to obtain more reflections to improve detections have also been proposed for this task.

However, methods that use RGB, RGB-D, and other types of cameras suffer from several key downsides that make them inappropriate for real-time navigation. RGB/RGB-D cameras suffer from low range and fields-of-view compared to lidars. Further, the quality of images deteriorates as the environmental illumination sharply increases or decreases. Most of the segmentation methods for glass detection get confused by reflective surfaces and tend to classify them as glass. This could severely restrict a robot's notion of navigable free space causing undesirable behaviors such as halting/freezing [28].

C. Multi-Layer Representations

For robot navigation, 2D grid/cost maps [29], [30], [31] have been used as a standard data structure to represent the distribution of obstacles, and navigation costs in an environment. The robot's planner uses the costs in these maps to compute a least-cost, collision-free path or velocity to navigate to its goal. Multi-layer Image Representations have been widely used for image processing tasks such as instance retrieval [14], image compression [15], and interpretation [32]. Other Multi-layer, hierarchical representations such as MIP-maps [13], hierarchical occlusion maps [16], quad trees [33], [34], multi-layer intensity maps [11] have existed that use several layers of grid arrays to represent various applications in graphics rendering, and obstacle detection. Our approach reduces the dimensions of point cloud intensities to 2D using [11], and uses it for transparent obstacle detection and navigation.

III. BACKGROUND

In this section, we first define the symbols and notations used in our work, then explain the underlying intensity maps and the preliminary concepts used to detect transparent obstacles.

A. Definitions and Assumptions

We make the following assumptions in our formulation for transparent obstacle detection. We assume that a robot modeled as a cylinder of radius r_{rob} , and height h_{rob} is equipped with a lidar mounted at height h_{lid} (and $h_{lid} \leq h_{rob}$) that shoots out light rays and generates 3D point clouds (PC) with their associated intensities. For simplicity, we assume that the robot's and the lidar's centers coincide. Each point in the point cloud is represented as $\mathbf{p} = \{x, y, z, int\}$, where x, y, z denote the point's location relative to the lidar, and $int \in [0, i_{max}]$ denotes its intensity, and i_{max} denotes the maximum possible intensity. Our coordinate frame convention is defined with the positive x, y, and z axes pointed forward, leftward, and upward respectively attached to the ground ($z = 0$) beneath the robot's center of mass. Throughout the text, symbols j, k are used to denote indices, and t denotes a time instant.

B. Light Intensity and Transparent Obstacles

The intensity measures the amount of light energy that is reflected back to the lidar from any object in the environment. Typically, points reflected from diffuse surfaces (opaque, smooth, and planar/uncurved) satisfy $int \approx i_{max}$. This is because the opacity ensures that most of the light rays do not get transmitted through the object, and the smoothness and planarity ensure that they do not scatter away from the lidar. Further, int depends on an object's proximity to the lidar, and the angle of incidence of the light. For transparent obstacles especially, the highest intensity is observed at 0° incidence [5].

To detect transparent obstacles from point clouds and navigate, our approach uses multiple layers of 2D grid maps obtained by projecting the intensity of point clouds belonging to certain height intervals H_j along the z-axis. The multiple layers together form a Multi-layer Intensity Map [11]. In which, each 2D layer has dimensions $n \times n$, and $(n/2, n/2)$ denotes the robot/lidar's position w.r.t the map. Each grid is represented by its row and column coordinate (r, c) . Formally, a single layer at time t is constructed from points belonging to an interval H_j , denoted as $I_{z \in H_j}^t$ is defined as,

$$I_{z \in H_j}^t(r, c) = \frac{\sum_x \sum_y int}{s^2} \\ \forall x \in [x_{low}, x_{low} + s], y \in [y_{low}, y_{low} + s], \\ x_{low} = \left\lfloor \left(r - \frac{n}{2}\right) \cdot s \right\rfloor \text{ and } y_{low} = \left\lfloor \left(c - \frac{n}{2}\right) \cdot s \right\rfloor \\ H_j = [low_j, high_j], low_j \leq high_j. \quad (1)$$

Here, s denotes the side length of the real-world square that each grid (r, c) represents in $I_{z \in H_j}^t$ that is computed continuously for every time instant.

IV. TOPGN: TRANSPARENT OBSTACLE PERCEPTION

In this section, we explain how transparent obstacles are detected using the layers of the intensity maps. Next, we explain how we isolate a Transparent Obstacle Neighborhood (TON) in real-world scenarios, and linearly extrapolate the transparent obstacle shape for autonomously navigating in their presence.

A. Transparent Obstacle Intensity Distribution

The intensities of point clouds incident on transparent obstacles along a horizontal plane as a function of the incident angle can be approximated as a Gaussian curve [6], [17]. Due to symmetry, this also holds true along the vertical/longitudinal plane as depicted in Fig. 2. The points with the peak intensity occur near the location where the angle of incidence is $\sim 0^\circ$ [5] at height h_{lid} , and the intensity dissipates for points farther from this center. This Gaussian pattern of point cloud intensities is observed for all transparent obstacles taller than h_{lid} . To detect this pattern efficiently and use it for navigation, we use a three-layered intensity map $I_{3L}^t = [I_{low}^t | I_{mid}^t | I_{high}^t]$ that stacks three 2D grid maps $(I_{low}^t, I_{mid}^t, I_{high}^t)$. Each of these layers is defined

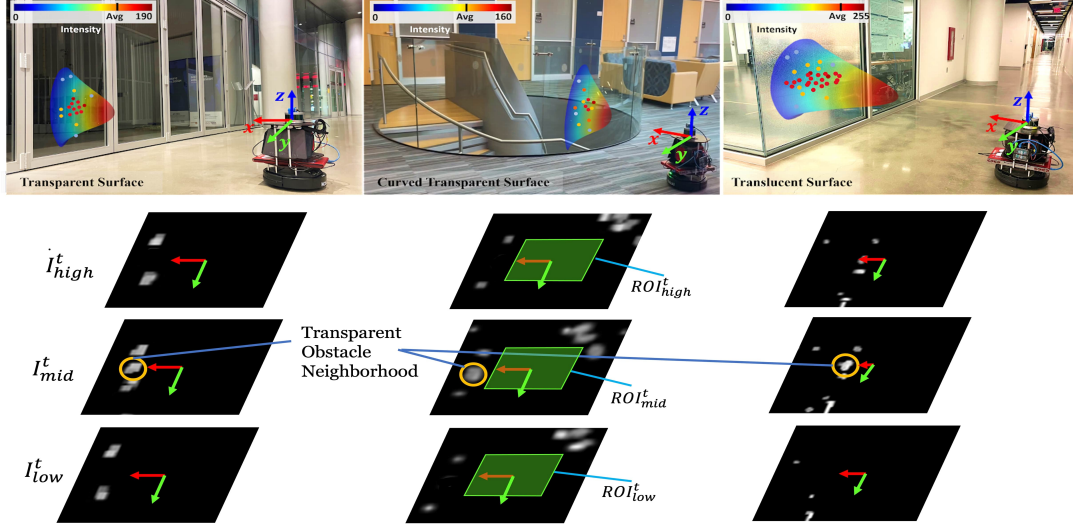


Fig. 2: The Gaussian distribution of point cloud intensities observed when the robot faces obstacles with different levels of transparency and shapes. The average intensities along with the distribution depend on the transparency (high transparency leads to lower average intensity), and shape (high curvature leads to lower average reflected intensity). The corresponding intensity maps at three different height ranges defined by equation 2 are shown, and the peak intensity neighborhood reflected from the glass appears in I_{mid}^t due to our definitions of its range, and is highlighted by the yellow circle. Our formulation detects this pattern and linearly extrapolates the transparent obstacle's shape from it to safely navigate unknown environments. The green parallelograms show regions of interest which are defined by equation 4.

according to equation 1 by the limits in the z axis specified as,

$$\begin{aligned} I_{low}^t &: z \in (0, h_{lid} - \Delta) \\ I_{mid}^t &: z \in (h_{lid} - \Delta, h_{lid} + \Delta) \\ I_{high}^t &: z \in (h_{lid} + \Delta, h_{lid} + 2\Delta). \end{aligned} \quad (2)$$

Here, Δ is a height parameter that controls the number of points that are projected on to I_{mid}^t . It is chosen empirically such that all the points with the highest intensity lie within $h_{lid} \pm \Delta$ when the lidar is d_{thresh} meters (threshold distance to maintain with obstacles) away from a completely transparent obstacle. Our definition of these layers leads to I_{mid}^t registering a prominent region with high intensities while I_{low}^t , and I_{high}^t register lower intensity values ≈ 0 in the same grid position (r, c) as shown in Fig. 2. We refer to any region that satisfies this condition as a *Transparent Obstacle Region* (TON).

B. Transparent Obstacle Isolation

To isolate a TON from I_{3L}^t , we formulate the following condition,

$$\mathcal{G}^t(r, c) = \begin{cases} 1 & \forall \{(r, c) \text{ s.t. } ROI_{mid}^t(r, c) \in \mathbf{R}, \text{ and} \\ & ROI_{low}^t(r, c) < \max(\mathbf{R})/3, \text{ and} \\ & ROI_{high}^t(r, c) < \max(\mathbf{R})/3\} \\ 0 & \text{Otherwise.} \end{cases} \quad (3)$$

Here, \mathbf{R} denotes a range of intensities, and $ROI_{mid}^t, ROI_{low}^t, ROI_{high}^t$ represent an $m \times m$ ($m < n$) regions of interest defined in the corresponding intensity

maps as,

$$ROI_{low/mid/high}^t = \{I_{low/mid/high}^t(r, c) | r, c \in [\frac{n}{2} - \frac{m}{2}, \frac{n}{2} + \frac{m}{2}]\}. \quad (4)$$

The isolated ROIs contain minor artifacts due to noise which are filtered out. Our filtering approach is based on identifying the contours of all the regions with 1's, and removing the ones with low areas. We use ROIs to further reduce computation costs. They are visually represented in green in Fig. 2. \mathcal{G}^t is an $m \times m$ grid map that contains only the grids belonging to various TONs (that contain value 1) at any time instant t .

C. Transparent Obstacle Extrapolation

\mathcal{G}^t only indicates the presence of a transparent obstacle (see Fig. 5) at a time instant and does not represent its true shape, which is required to avoid collisions during navigation. Therefore, we propose a method to linearly extrapolate the transparent object based on the j^{th} transparent obstacle neighborhood TON_j in \mathcal{G}^t . To this end, we first compute the centroid grid for TON_j as $(r_{cen}^j, c_{cen}^j) = (\sum r / \text{size}(TON_j), \sum c / \text{size}(TON_j)) \forall r, c \text{ s.t. } \mathcal{G}^t(r, c) = 1$, where $\text{size}(TON_j)$ returns the number of grids in TON_j . Next, bounding circles C_{bound}^j centered at r_{cen}^j, c_{cen}^j , and radius equal to the distance from the centroid to the farthest point in TON_j as shown in Fig. 3 are computed.

Now, let us consider a light ray in 3D that is incident at $\sim 0^\circ$ on a transparent surface. It is by definition perpendicular to the tangent to the surface at the point of incidence (see Fig. 3a). The line equation of such a light ray in 2D, relative

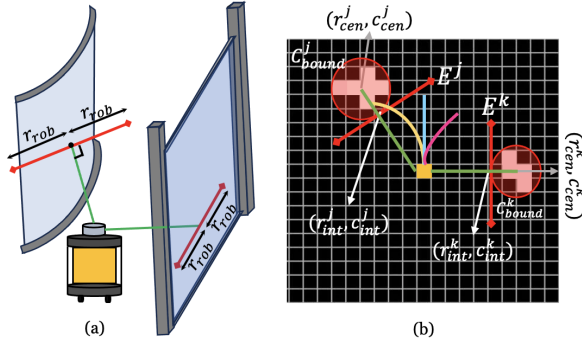


Fig. 3: (a) The green lines depict the light ray that is incident at 0° on a transparent obstacle (in blue). The line segment perpendicular to this light ray is represented in red and is extended from the point of incidence on either side by the radius of the robot r_{rob} . (b) The same scenario on \mathcal{G}^t where the light ray connects the robot (in yellow) with (r_{cen}^j, c_{cen}^j) . The red lines are extended on either side by r_{rob}/s grids in red. All grids on the other side of the red lines are considered as obstacles for time instant t . A few of the robot's instantaneous candidate trajectories when integrated with a planner [1] are shown in blue, yellow, and pink. An optimal trajectory is chosen based on its distance away from obstacles, and the progress/heading towards the goal. In this scenario, the pink trajectory is preferred over the others as it is away from obstacles.

to \mathcal{G}^t can be obtained by connecting the position of the lidar, and the centroid of a TON (see Fig. 3). Then, the vectors of the incident light ray, and the tangent line perpendicular to it can be represented as,

$$\begin{aligned} \mathbf{light}^j &= [r_{cen}^j - m/2, c_{cen}^j - m/2]^\top, \\ \mathbf{tangent}^j &= [c_{cen}^j - m/2, -(r_{cen}^j - m/2)]^\top. \end{aligned} \quad (5)$$

Let the point of intersection of the **light** ray with the corresponding C_{bound}^j be (r_{int}^j, c_{int}^j) . We extrapolate the tangent line segment from (r_{int}^j, c_{int}^j) on either direction by the robot's radius r_{rob}/s grids as,

$$E^j = \{[r_{int}^j, c_{int}^j] \pm \frac{r_{rob}}{s} (\frac{\mathbf{tangent}^j}{\|\mathbf{tangent}^j\|})\}, \quad (6)$$

We extrapolate by the robot's radius on either side to minimize the amount of free space that is considered an obstacle by the robot's planner. Line segment E^j is considered as a half-plane beyond which the robot should consider an obstacle region and avoid. This is depicted in Fig. 3. All grids corresponding to vectors and line equations are integerized. We omit this in the equations for readability. Finally, we obtain a grid map containing all the extrapolated TONs, and refer to it as \mathcal{G}_{extrap}^t . It is defined as $\mathcal{G}_{extrap}^t(r, c) = 1 \forall (r, c) \in E_j \forall j$.

D. Collision Avoidance

Our transparent obstacle extrapolation can be integrated with any velocity/trajectory planning method [1], [35] that evaluates navigation *costs* for the robot's candidate trajectories based on their proximity to obstacles, and the robot's goal. We use the work by Fox et al. [1] along with our map layers, and \mathcal{G}_{extrap}^t to navigate unknown environments with transparent obstacles. To first obtain a complete representation of all the obstacles (opaque and extrapolated transparent

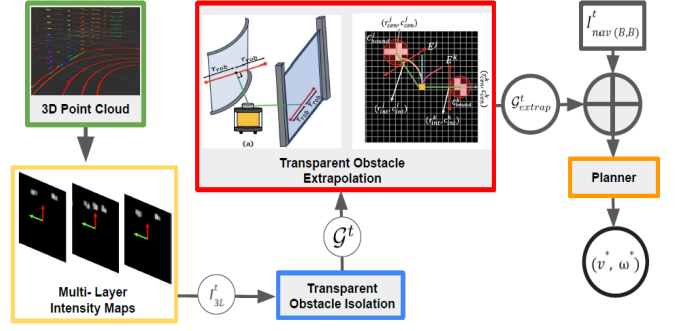


Fig. 4: TOPGN's overall system architecture. Three-layered intensity map I_{3L}^t is extracted from the 3D lidar point cloud to isolate the transparent obstacles. The linear extrapolation is performed on isolated obstacles to estimate the true shape of the obstacles for collision avoidance. The extrapolated transparent obstacles are combined with the navigation cost map I_{nav}^t to generate collision-free and goal-reaching actions from the planner. This overall framework demonstrates superior real-time transparent obstacle detection capabilities compared to state-of-the-art vision-based and lidar-based approaches.

obstacles) in the environment, we obtain a grid map that can be used for navigation as I_{nav}^t ,

$$\begin{aligned} I_{nav}^t &= I_{low}^t + I_{mid}^t + I_{high}^t \\ I_{nav}^t(B, B) &= I_{nav}^t(B, B) + \mathcal{G}_{extrap}^t \\ B &= (\frac{n}{2} - \frac{m}{2} : \frac{n}{2} + \frac{m}{2}). \end{aligned} \quad (7)$$

In equation 7, $I_{nav}^t(B, B)$ represents an $m \times m$ subset similar to the green ROI in Fig. 2 where the extrapolated transparent obstacles are added. I_{nav}^t represents *all* environmental obstacles and can be used by [1] to evaluate the costs for candidate trajectories and compute the least cost trajectory for the robot to follow. Please refer to the supplementary document for additional details on the planner.

Lemma IV.1. *A candidate robot trajectory that does not intersect with any line segment E^j during navigation at time instant t guarantees collision avoidance with every transparent obstacle in the robot's vicinity at that instant.*

Proof. Let us consider an environment with transparent obstacles of which K unoccluded 3-dimensional, smooth (i.e., their shape is a differentiable curve) transparent obstacles lie within the lidar's sensing range. There exist K closest points on these obstacles from the lidar. The lines connecting the lidar and these closest points are normal to the transparent obstacle and will lead to corresponding transparent obstacle neighborhoods (TONs) in \mathcal{G}^t . Therefore, all the transparent obstacles satisfying smoothness, and 3D assumptions can be detected in \mathcal{G}^t at time instant t . This ensures that there exists a corresponding tangent line segment E^j that can be extrapolated for the j^{th} transparent obstacle neighborhood in \mathcal{G}^t as shown in Fig. 3b. We refer to the map with the extrapolated line segments as \mathcal{G}_{extrap}^t .

By construction, the closest point(s) in the j^{th} transparent obstacle are contained beyond E^j when viewed from the lidar. Let $traj_k^I$ denote the k^{th} candidate trajectory represented as a set of row and column coordinates relative to the

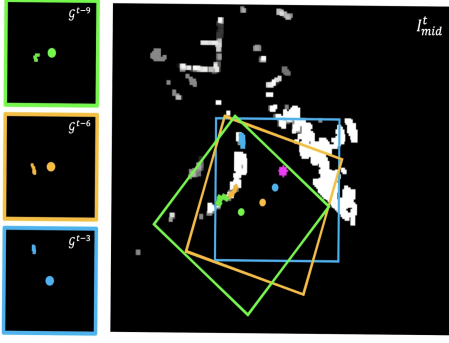


Fig. 5: I_{mid}^t for the curved glass scenario shown in Fig. 2 [center], and the transformed \mathcal{G}^{t-3} (blue square), \mathcal{G}^{t-6} (yellow square), and \mathcal{G}^{t-9} (green square) added to it. The centers of these squares depict the robot's movement as time progresses. This addition reconstructs the true shape of the transparent obstacle for mapping.

intensity map. If $\text{traj}_k^I \cap E^j = \emptyset \implies \text{traj}_k^I \cap \text{TON}_j = \emptyset$, guaranteeing collision avoidance with transparent obstacles at instant t . ■

Our approach of isolating (section IV-B) and extrapolating (section IV-C) transparent obstacles allows a robot to avoid collisions with them in completely unknown environments without any prior mapping. Next, we highlight the generality of our TON isolation by demonstrating how it can also be used for mapping transparent obstacles in real-time.

E. Application to Mapping

In this section, we explain how our transparent object isolation in various instances of \mathcal{G}^t can be used for mapping the environment in 2D in a single walk-through. To this end, the previous instances of the neighborhood ($\mathcal{G}^{t-1}, \mathcal{G}^{t-2}, \dots, \mathcal{G}^{t-t_{past}}$) transformed relative to the present time instant t are added to the current I_{mid}^t to construct the true shape of a transparent obstacle. Here, t_{past} is the number of past instances considered for mapping the transparent obstacle. To transform the position of obstacles in \mathcal{G}^{t-k} relative to I_{mid}^t , we use transformation matrices T_t^{t-k} computed based on the robot's motion between instances $t-k$ to t . That is,

$$\begin{aligned} \mathcal{G}_t^{t-k} &= T_t^{t-k} \cdot \mathcal{G}^{t-k}, \\ I_{mid}^t(B, B) &= I_{mid}^t(B, B) + \mathcal{G}_t^{t-k} \quad \forall k \in [1, t_{past}] \end{aligned} \quad (8)$$

This addition is depicted in Fig. 5. Finally, to map all obstacles, we perform $I_{mapping}^t = I_{low}^t + I_{mid}^t + I_{high}^t$.

V. RESULTS AND EXPERIMENTS

In this section, we explain TOPGN's implementation, and real-world experiments/evaluation, and demonstrate its advantages for autonomous navigation.

A. Hardware and Software Implementation

We implement our proposed approach on a Turtlebot 2 robot equipped with a Velodyne VLP16 lidar, and a laptop with an Intel i7 CPU and NVIDIA RTX 3060 GPU. The robot is also equipped with an Intel Realsense d435 camera to collect images to compare with RGB segmentation methods. We use the following parameters in the implementation:

$$n = 200, m = 80, \mathbf{R} = [100, 130], h_{rob} = h_{lid} = 0.5m, r_{rob} = 0.3m, \Delta = 0.2m.$$

B. Evaluations

Comparison Methods: We compare TOPGN with two types of methods for glass, and transparent obstacle detection: 1. Semantic segmentation methods that use RGB images, and 2. SLAM methods that use lidar 2D scans or 3D point clouds. We use the following semantic segmentation methods: GDNet [2], TransLab [3], Mirror-Net [36], RGB-T segmentation [9], and the following mapping methods: Gmapping [37], Glass-SLAM [4], and Glass-Cartographer [6]. Gmapping [37] is used as a baseline to indicate the challenges in detecting glass.

Test Dataset: To evaluate both types of methods in a uniform manner, we treat transparent obstacle detection as a segmentation problem and create a test dataset with three sets of inputs and masks. The first set of masks outlines transparent obstacles on RGB images. The second set outlines them on local square segments cropped from the global 2D grid maps computed by Gmapping [37], Glass-SLAM [4], and Glass-Cartographer [6]. The third set marks transparent obstacles on local 2D grid maps to evaluate TOPGN. We use TOPGN to create a local map of obstacles (similar to Fig. 5), and evaluate its detection capabilities based on that map. The test set is collected in various environments with sharp lighting changes, different levels of transparency, color, textures, and shapes in transparent obstacles (see Figs. 1, 6, supplementary material).

Metrics: To evaluate and compare various methods in terms of transparent obstacle detection, we use four widely adopted metrics:

- **Mean IoU (mIoU):** It is the area of overlap between the segmentation output and the ground truth divided by the area of union between the predicted segmentation and the ground truth.
- **Pixel Accuracy (PA):** It denotes the percent of pixels that are accurately classified in the image. It is calculated as,

$$PA = \frac{\sum_{\forall j} P_{jj}}{\sum_{\forall j} T_j}. \quad (9)$$

Here, P_{jj} is the number of pixels predicted to be in class j , and belonging to class j , and T_j denotes the total number of pixels labeled as class j .

- **F_1 score:** It is the harmonic mean of the average precision and average recall calculated as,

$$F_1 = \frac{2 \cdot \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (10)$$

- **Mean Absolute Error (MAE):** It is a measure of errors between paired observations (predictions and ground truth). It is calculated as,

$$MAE = \frac{\sum_{j=1}^{tot} |y_j - x_j|}{tot}. \quad (11)$$

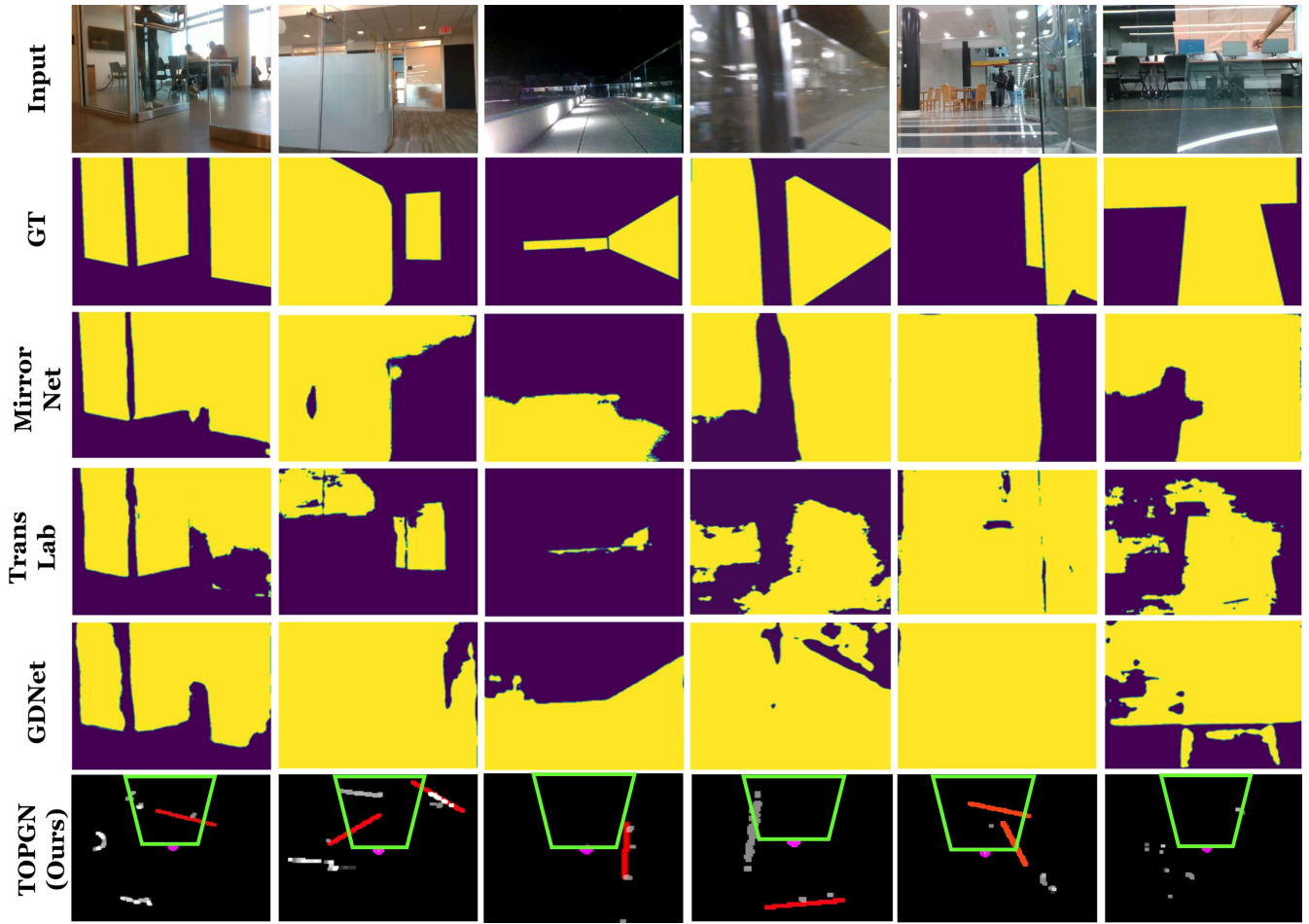


Fig. 6: The glass detection outputs (yellow indicates transparent obstacle) of MirrorNet [36], TransLab [3], and GDNet [2] in a few images from our test benchmarks, which contains scenarios with strong lighting changes (columns 1, 3), reflections (column 5), motion blur (column 4), and curved glass (columns 2, 5). In many instances, these segmentation methods wrongly classify free space as glass. Further, the robot’s motion could cause blurring in some sets of frames, which leads to inaccurate detections. During navigation, such errors cause the robot’s planner to freeze or collide. TOPGN accurately extrapolates (red lines) the transparent obstacle in most scenarios. The pink circle denotes the robot’s position and the green polygons represent the camera’s field of view (FOV). We also depict some failure cases (columns 5 and 6) with highly non-convex and tilted transparent obstacles.

Here, tot represents the total number of data points (e.g. pixels), and y_j, x_j represent the prediction and the ground truth of the data point respectively.

Additionally, to evaluate a method’s applicability for real-time robot navigation, we also measure its inference rate. We define inference rate as the inverse of the time taken by a method to compute a segmentation mask or update a map based on local observations.

Navigation Evaluations: To assess the benefits of TOPGN for robot navigation, we compare the navigation success rate when using: 1. vanilla planner using 2D lidar scans [1], 2. planner using the segmentation, 3. planner using SLAM methods, and 4. planner using our proposed extrapolation method (section IV-C). The success rate is defined as the number of times the robot reaches its goal without colliding with any obstacle and freezing or halting forever in 10 trials. For the evaluation, we utilize the outputs (which contain transparent obstacle locations w.r.t the robot) of all these methods to evaluate the robot’s trajectories using [1]. The planner [1] chooses trajectories that do not intersect with any

Methods	MIoU \uparrow	PA \uparrow	F_1 \uparrow	MAE \downarrow	Infer. Rate \uparrow
TransLab [3]	0.564	0.809	0.554	0.190	1.61
MirrorNet [36]	0.412	0.631	0.701	0.369	0.40
GDNet [2]	0.502	0.659	0.627	0.341	9.32
RGBT Segmnt. [38]	0.215	0.724	0.385	0.275	0.45
Gmapping [37]	0.497	0.623	0.484	0.368	0.23
Glass-SLAM [4]	0.841	0.976	0.796	0.013	0.36
Cartographer Glass [6]	0.813	0.965	0.824	0.025	7.34
TOPGN w VLP16 Lidar (ours)	0.872	0.992	0.929	0.008	45.67 (on CPU)
TOPGN w OS1-32 Lidar (ours)	0.881	0.928	0.893	0.011	38.24 (on CPU)

TABLE I: Transparent obstacle detection performance comparisons for semantic segmentation and lidar-based SLAM methods against TOPGN (Ours) using various metrics.

transparent obstacle, and also utilizes 2D laser scans to detect and avoid opaque obstacles. We evaluate it in five scenarios with transparent obstacles: 1. Indoor setting with a glass door (one open, and one closed) and strong backlighting, 2. Indoor setting with reflective surfaces and strong multiple lights, 3. Outdoor setting with low light, 4. Indoor setting with a curved acrylic transparent obstacle, 5. Indoor lab setting with mirrors, arbitrarily shaped transparent PVC, and acrylic sheets.

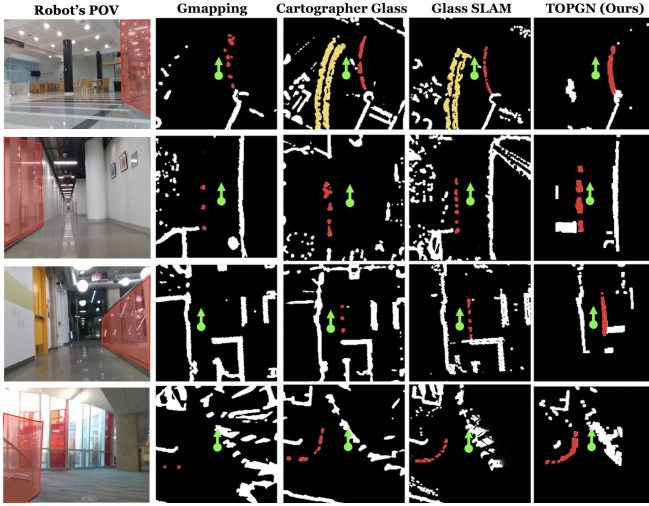


Fig. 7: Transparent obstacle mapping outputs from the lidar-based mapping methods: Gmapping [37]; Cartographer Glass [6] and Glass SLAM [4] compared to TOPGN (ours). Red regions indicate the transparent objects detected from each method on a local map where the robot is located at the center (green color arrow). Artifacts due to erroneous glass detection from SLAM methods are marked in yellow color. The perspective from the robot as it maps the environment is shown in the first column for context, and the transparent obstacles are marked in red.

Metrics	Method	Scn. 1	Scn. 2	Scn. 3	Scn. 4	Scn. 5
Success Rate (%) \uparrow	DWA Planner [1]	0	0	20	0	0
	TransLab [3]	20	50	0	30	0
	GDNet [2]	40	20	0	20	0
	Gmapping [37]	0	10	20	0	0
	Glass-SLAM [4]	10	30	40	30	0
	TOPGN (ours)	70	100	100	80	60

TABLE II: Navigation performance comparison for four scenarios that include transparent obstacles with different sizes, shapes, and under challenging lighting conditions.

C. Analysis

Table I shows the transparent obstacle detection performance of the segmentation-based, and SLAM-based methods. We choose to compare the performance of methods that use two different modalities to highlight the severe limitations of image-based segmentation. Of the segmentation methods, TransLab [3] performs the best, followed by GDNet [2], MirrorNet [36], and RGB-T segmentation’s [9] RGB-only model. All segmentation methods perform well in environments that are well-lit, provide ample context to indicate the presence of transparent obstacles, and contain planar/uncurved glass. When these conditions are not satisfied, their performance tends to deteriorate. Such scenarios are shown in Fig. 6 with the input RGB image, the corresponding ground truth (GT), and the outputs of three segmentation methods, and our method’s linear extrapolation results in these cases. We observe that in many instances these segmentation methods incorrectly mark free space as a transparent obstacle (in yellow). This predominantly occurs in the parts of the input image that have bright lights or reflections from shiny surfaces.

Such methods also struggle in low-light conditions (Fig.

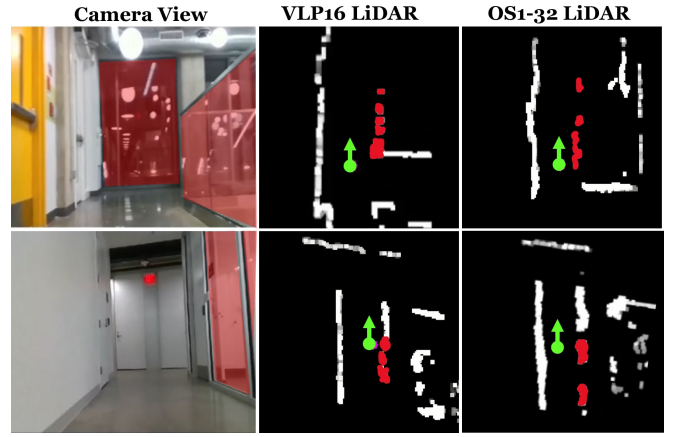


Fig. 8: TOPGN’s transparent obstacle detection performance comparison for two different 3D LiDARs with different channel resolution: 1. Velodyne VLP16 has 16 verticle channels; 2. OS1-32 Lidar has 32 verticle channels. We observe that our method demonstrates comparable detection performance for different 3D LiDAR sensors that have different channel resolutions. The robot’s camera views are presented on the left to help understand the transparent regions (marked in red) on the cost maps.

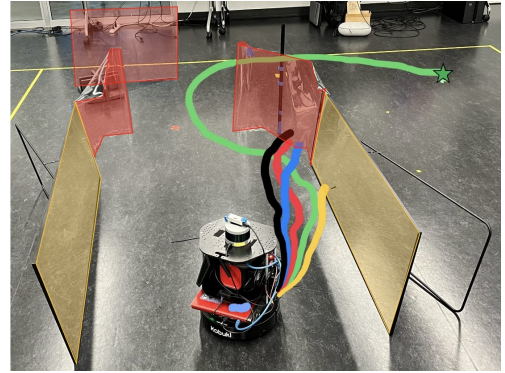


Fig. 9: Scenario 5 that requires the robot to navigate in the presence of mirrors (highlighted in yellow), arbitrarily-shaped transparent PVC and acrylic sheets (highlighted in red). We observe that TOPGN (green) navigates the robot by avoiding all transparent obstacles. Semantic segmentation methods (in blue [2] and red [3]), and using 2D lidar scans [1] cannot detect transparent PVC, leading to collisions. Glass-SLAM [4] (in yellow), due to slow map update rates, directly collides with the mirror.

6 columns 3 and 4), and frames with motion blur (Fig. 6 column 4), which could occur in images captured from a robot. In certain cases with glass doors (when one is open, and the other is closed), these methods predict the free space to also contain glass (see Fig. 6 column 1) similar to the observations in [20].

TOPGN, on the other hand, accurately extrapolates transparent obstacles linearly in the scenes in columns 1-3. For column 1, TOPGN extrapolates the closed door accurately. Although the line extended to cover the free space in the figure, as the robot moves and the line’s position and orientation change to open up the free space in most cases during navigation. For curved glass (columns 2 and 5), the line is extrapolated tangential to it. The scenario in column 5

is especially challenging as it contains two glass components: a curved glass wall, and an open door. TOPGN accurately extrapolates the curved wall. We observe that TOPGN does not extrapolate the glass in column 4. This is because it is already detected as an obstacle (in grey) due to the strong reflections from the dust settled on the glass. Lidar-based detection benefits from such real-world phenomena, and can detect obstacles regardless of the robot’s motion. We discuss the scenario in columns 5 and 6 further under failure cases.

SLAM-based methods [4], [6] are not affected by low-light, bright reflections, or motion blur, and can detect transparent obstacles accurately as presented in Fig. 7. Gmapping, which is not formulated to detect transparent obstacles, exhibits low detection accuracy (see Fig. 7 column 2). Its accuracy springs from detecting some portions of glass based on the opaque railings around it. It is used for comparison to highlight the difficulty in detecting transparent obstacles in general. Glass-SLAM [4], and Cartographer Glass [6] accurately detect glass of various shapes and conditions. However, the presence of dynamic obstacles could cause artifacts such as a trail of their positions to be recorded and marked as obstacles on the map as shown from yellow regions in row 1 of Fig. 7. This occurs because they are configured to record *all* reflected points (of various intensities) to detect glass. Configurations that prevent such artifacts lead to poor glass detection. TOPGN’s formulation, when applied for mapping robustly detects transparent obstacles in all these cases and is not affected by environmental conditions. Importantly, TOPGN’s TON isolation based on the condition in equation 3 ensures that other obstacles (e.g. dynamic pedestrians) are not detected as transparent obstacles. This is because the intensity condition in equation 3 cannot be satisfied by opaque obstacles such as humans. For collision avoidance in unknown/unmapped environments, equation 7 ensures that no other obstacle is missed as it combines the extrapolated glass with all the obstacles in I_{low}^t , I_{mid}^t , and I_{high}^t . For mapping, equation 8 ensures that all obstacles are detected at each instant, and artifacts from dynamic obstacles are not added to the map. Equation 8 combines the middle-intensity map (I_{mid}^t that contains all opaque obstacles) with a transformed \mathcal{G}^{t-k} (which contains an instance of a transparent obstacle).

Inference Rate: For real-world implementation, perception methods must possess a high inference rate. Comparing the inference rates of these methods, only GDNNet [2] executes ~ 9 Hz, making it suitable for real-time navigation. All other segmentation methods have a high computational overhead and do not execute in real-time on a mobile GPU. Glass-SLAM requires $\sim 3 - 4$ seconds to update its map, making it unsuitable for real-time navigation. Cartographer Glass [6] maps faster, but in some cases may not update the map in time to avoid obstacles. TOPGN has a superior inference rate for both extrapolation and mapping and executes at ~ 50 Hz on a mobile laptop CPU, enabling real-time navigation.

Navigation Success Rate: In terms of navigation success rate, we observe that using TOPGN’s extrapolation greatly

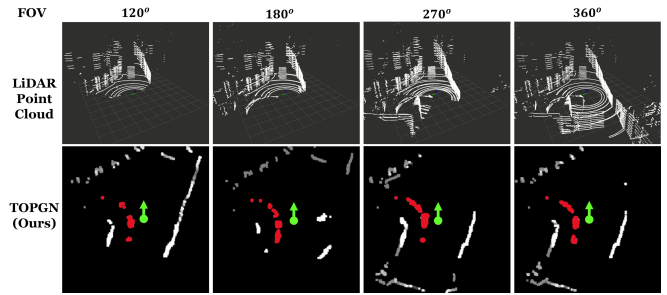


Fig. 10: TOPGN’s performance comparison for different 3D lidar horizontal field of views (FOV). We observe that TOPGN’s transparent obstacle detection performance is almost consistent until the lidar’s horizontal FOV reduces from 360° to 180° . However, TOPGN’s performance degrades when the FOV is less than 180° . Red regions indicate the transparent obstacle regions detected by TOPGN.

improves the robot’s rate of reaching the goal. The planner when only using 2D laser scans [1] to navigate and Gmapping, could only detect opaque obstacles, which led to collisions with transparent objects in all the trials in scenarios 1, 2, and 4. The planner when augmented with segmentation methods was able to successfully reach its goal in some trials. However, in most trials, incorrectly classifying free space as glass led to freezing issues, where the robot oscillates or halts indefinitely. We chose to evaluate TransLab [3], and Glass-SLAM [4] despite their low inference rates due to their reasonably high detection accuracy. However, their low inference rate led to severe oscillations when used with the planner. In scenario 1 in Fig. 1, Translab mistook the open door to be closed causing the robot to freeze before entering the room. GDNNet, on the other hand, was more accurate in this scenario (also depicted in Fig. 6 column 1, row 5). Both TransLab [3], and GDNNet [2] experienced performance degradation in scenarios 2 and 3 in Fig. 1 due to bright and low light conditions, respectively. In scenario 4, although the RGB-based methods managed to detect glass to a certain extent, the low field-of-view of the camera led to them not viewing the curved glass in many trials. Additionally, misclassifying free space as glass (see Fig. 6 column 2) also caused freezing.

TOPGN, in all these scenarios, was able to isolate the TONs in the vicinity, and linearly extrapolate the transparent obstacle. Since the grids beyond the extrapolated line are considered as obstacles, the planner chose trajectories avoiding this region and averted collisions. Due to the narrow passages, and high curvature of glass in scenarios 1 and 4 respectively, our method led to collisions in some cases. In scenario 1 especially (also depicted in Fig. 6 column 1), TOPGN’s linear extrapolation caused the robot to freeze in some cases due to the narrow passage. However, in most cases, the extrapolation aided in avoiding the closed door and reaching the goal.

The results of our experiments in scenario 5 are shown in Fig. 9. Our method (in green) is able to navigate around the mirrors, and the transparent PVC. Segmentation methods

such as GNet [2] (in blue), and TransLab [3] (in red) cannot accurately detect the transparent PVC and collide with it. The reflections from the floor also confuse the segmentation and navigation. In many trials, these methods lead to freezing as most of the image is classified as a transparent obstacle. Glass-SLAM [4] (in yellow) directly collides with the mirror due to its slow map update rate, and using 2D laser scans (in black) avoids the mirror but collides with the PVC similar to segmentation methods. This scenario highlights our method's linear extrapolation's capabilities in the presence of transparent obstacles with various shapes and materials.

Compatibility with Different Lidar Sensors: We evaluate our method's transparent obstacle detection capabilities for two different 3D lidar sensors in Fig. 8 and Table I. Even though our experiments are conducted using a VLP16 Velodyne lidar with 16-channel vertical resolution, our methods demonstrate similar detection performance with an Ouster OS1-32 lidar (32-channel resolution) without any changes to the algorithmic parameters or threshold values. However, the 32-channel Ouster lidar results in a relatively lower inference rate due to the processing of a significantly higher dimensional point cloud compared to the 16-channel Velodyne lidar.

Our method cannot be used with any 2D lidar sensors since it requires multi-level intensities to isolate the transparent object regions. Moreover, 3D lidars with significantly low vertical resolution (e.g., less than 6-8 channels) might not be suitable for our approach due to the requirement of reasonable point clouds in each layer of the multi-layer intensity maps. However, we noticed that commonly available 3D lidars have at least 16 vertical channels.

Effect of the Lidar's Horizontal FOV: We observe that TOPGN's transparent obstacle detection demonstrates consistent qualitative performance for lidar point clouds captures from 360° to 180° sensor Field-of-VIEWS (FOV) in Fig. 10. However, the performance degrades beyond 180° FOV. Hence, TOPGN can be used with relatively low FOV lidar sensors instead of the 360° lidars.

Failure Cases: We observe that in cases where a transparent obstacle has a sharp non-convex shape (Fig. 6 column 5 where the glass door is extrapolated incorrectly), or thin (almost 2-dimensional when viewed from the lidar) the lidar may not consistently be able to detect points that could avoid collisions at a future time instant. In such cases, the robot could get stuck or collide. However, we note that all existing methods also fail in such cases (e.g. see column 5 in Fig. 6). Additionally, when transparent objects are highly inclined, the reflected laser points may not lie in I_{mid}^t for it to be extrapolated. We depict this scenario in Fig. 6 column 6. We observe that the detection depends on the angle of inclination, material and thickness of the transparent object.

In our proof, the small TON is enclosed by a circle and the tangent line is constructed. This approximates the transparent object as convex at the TON. This holds true in real-world transparent obstacles and the way they are constructed. In extreme corner cases, our way of linear extrapolation may not be enough, and a higher-order curve could be more suitable.

In cases with curved glass, we observe some gaps between the TONs detected in subsequent time steps. However, these gaps can be closed up by inflating the blobs/neighborhoods based on the robot's radius.

VI. CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

We present TOPGN, a method to isolate instances of transparent obstacle neighborhoods in a multi-layer grid map representation containing point cloud intensities. Our method then extrapolates the regions that could potentially contain transparent obstacles, which when integrated with a navigation scheme, leads to superior rates of successfully reaching the robot's goal. We showed how TOPGN's transparent obstacle isolation can be useful in mapping applications. Our method is unaffected by adverse conditions such as harsh environmental lighting, reflections, motion blur from the robot, etc.

Our method has a few limitations. In general, lidars have a circular blind spot around them and could miss obstacles closer than 1.5m. Unlike RGB-based methods, lidar-based methods require several time instances to detect/map the true shape of transparent obstacles. We observe that in scenarios with sharply curved or tilted glass (e.g. curved glass with an open glass door), we may not be able to obtain sufficient TONs to detect and extrapolate the obstacle consistently. Our method is dependent on the specifications and quality of the lidar sensor. Therefore, using a lidar with a low vertical FOV would affect the glass detection accuracy. The transparent obstacles are implicitly assumed to be smooth or well approximated by the linear extrapolation. Similar to prior mapping methods that use lidar, our approach can be affected by odometry/localization errors when mapping transparent obstacles. This could lead to the robot freezing/halting indefinitely in narrow passages.

In the future, we would like to address these limitations and investigate methods to obtain the same capabilities using a low-FOV lidar or time-of-flight sensor.

REFERENCES

- [1] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.
- [2] H. Mei, X. Yang, Y. Wang, Y. Liu, S. He, Q. Zhang, X. Wei, and R. W. Lau, "Don't hit me! glass detection in real-world scenes," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [3] E. Xie, W. Wang, W. Wang, M. Ding, C. Shen, and P. Luo, "Segmenting transparent objects in the wild," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 696–711.
- [4] X. Wang and J. Wang, "Detecting glass in simultaneous localisation and mapping," *Robotics and Autonomous Systems*, vol. 88, pp. 97–103, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0921889015302670>
- [5] P. Foster, Z. Sun, J. J. Park, and B. Kuipers, "Visagge: Visible angle grid for glass environments," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 2213–2220.
- [6] L. Weerakoon, G. S. Herr, J. Blunt, M. Yu, and N. Chopra, "Cartographer-glass: 2d graph slam framework using lidar for glass environments," *arXiv preprint arXiv:2212.08633*, 2022.
- [7] J. Lin, Y. Hei Yeung, and R. W. H. Lau, "Depth-aware Glass Surface Detection with Cross-modal Context Mining," *arXiv e-prints*, p. arXiv:2206.11250, June 2022.

- [8] L. Zhu, A. Mousavian, Y. Xiang, H. Mazhar, J. Eenbergen, S. Debnath, and D. Fox, "Rgb-d local implicit function for depth completion of transparent objects," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2021, pp. 4647–4656. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00462>
- [9] D. Huo, J. Wang, Y. Qian, and Y.-H. Yang, "Glass segmentation with rgb-thermal image pairs," *IEEE Transactions on Image Processing*, vol. 32, pp. 1911–1926, 2023.
- [10] Z. Huang, K. Wang, K. Yang, R. Cheng, and J. Bai, "Glass detection and recognition based on the fusion of ultrasonic sensor and rgb-d sensor for the visually impaired," 10 2018, p. 14.
- [11] A. Jagan Sathyamoorthy, K. Weerakoon, M. Elnoor, and D. Manocha, "Using Lidar Intensity for Robot Navigation," *arXiv e-prints*, p. arXiv:2309.07014, Sept. 2023.
- [12] P. Foster, C. Johnson, and B. Kuipers, "The reflectance field map: Mapping glass and specular surfaces in dynamic environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8393–8399.
- [13] J. Ewins, M. Waller, M. White, and P. Lister, "Mip-map level selection for texture mapping," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 4, pp. 317–329, 1998.
- [14] Q. Deng, S. Wu, J. Wen, and Y. Xu, "Multi-level image representation for large-scale image-based instance retrieval," *CAAI Transactions on Intelligence Technology*, vol. 3, no. 1, pp. 33–39, 2018.
- [15] F. Meyer, A. Averbuch, J. Stromberg, and R. Coifman, "Multi-layered image representation: application to image compression," in *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, vol. 2, 1998, pp. 292–296 vol.2.
- [16] H. Zhang, D. Manocha, T. Hudson, and K. E. Hoff, "Visibility culling using hierarchical occlusion maps," *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997. [Online]. Available: <https://api.semanticscholar.org/CorpusID:496617>
- [17] H. Wei, X. Li, Y. Shi, B. You, and Y. Xu, "Fusing sonars and lrf data to glass detection for robotics navigation," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2018, pp. 826–831.
- [18] H. Tibebe, J. Roche, V. De Silva, and A. Kondo, "Lidar-based glass detection for improved occupancy grid mapping," *Sensors*, vol. 21, no. 7, p. 2263, 2021.
- [19] H. Mei, X. Yang, L. Yu, Q. Zhang, X. Wei, and R. W. H. Lau, "Large-field contextual feature learning for glass detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3329–3346, 2023.
- [20] J. Lin, Z. He, and R. W. Lau, "Rich context aggregation with reflection prior for glass surface detection," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 13 410–13 419.
- [21] Z. Wu, S. Su, Q. Chen, and R. Fan, "Transparent objects: A corner case in stereo matching," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 12 353–12 359.
- [22] C. Xu, J. Chen, M. Yao, J. Zhou, L. Zhang, and Y. Liu, "6dof pose estimation of transparent object from a single rgb-d image," *Sensors*, vol. 20, no. 23, 2020. [Online]. Available: <https://www.mdpi.com/1424-8220/20/23/6790>
- [23] X. Liu, R. Jonschkowski, A. Angelova, and K. Konolige, "Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, 2020.
- [24] S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song, "Clear grasp: 3d shape estimation of transparent objects for manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3634–3642.
- [25] T. Weng, A. Pallankize, Y. Tang, O. Kroemer, and D. Held, "Multi-modal transfer learning for grasping transparent and specular objects," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3791–3798, 2020.
- [26] J. Ichnowski*, Y. Avigal*, J. Kerr, and K. Goldberg, "Dex-NeRF: Using a neural radiance field to grasp transparent objects," in *Conference on Robot Learning (CoRL)*, 2020.
- [27] G. Zhai, D. Huang, S.-C. Wu, H. Jung, Y. Di, F. Manhardt, F. Tombari, N. Navab, and B. Busam, "Monograspnet: 6-dof grasping with a single rgb image," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1708–1714.
- [28] A. J. Sathyamoorthy, U. Patel, T. Guan, and D. Manocha, "Frozone: Freezing-free, pedestrian-friendly navigation in human crowds," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4352–4359, 2020.
- [29] X. Qi, W. Wang, M. Yuan, Y. Wang, M. Li, L. Xue, and Y. Sun, "Building semantic grid maps for domestic robot navigation," *International Journal of Advanced Robotic Systems*, vol. 17, no. 1, p. 1729881419900066, 2020.
- [30] H. K. Tripathy, S. Mishra, H. K. Thakkar, and D. Rai, "Care: A collision-aware mobile robot navigation in grid environment using improved breadth first search," *Computers & Electrical Engineering*, vol. 94, p. 107327, 2021.
- [31] D. De Gregorio and L. Di Stefano, "Skimap: An efficient mapping framework for robot navigation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2569–2576.
- [32] M. Ivasic-Kos, M. Pobar, and I. Ipsic, "Multi-layered image representation for image interpretation," in *Proceedings of the Third Workshop on Vision and Language*, 2014, pp. 115–117.
- [33] G. M. Hunter and K. Steiglitz, "Operations on images using quad trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 145–153, 1979.
- [34] H. Samet and M. Tamminen, "Computing geometric properties of images represented by linear quadtrees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, no. 2, pp. 229–240, 1985.
- [35] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," [1993] *Proceedings IEEE International Conference on Robotics and Automation*, pp. 802–807 vol.2, 1993. [Online]. Available: <https://api.semanticscholar.org/CorpusID:5641886>
- [36] X. Yang, H. Mei, K. Xu, X. Wei, B. Yin, and R. W. Lau, "Where is my mirror?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8809–8818.
- [37] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.
- [38] D. Huo, J. Wang, Y. Qian, and Y.-H. Yang, "Glass segmentation with rgb-thermal image pairs," *IEEE Transactions on Image Processing*, vol. 32, pp. 1911–1926, 2023.

VII. APPENDIX

In this document we provide additional definitions, and explanations that support and enhance the main manuscript.

A. Integration with Planning

We provide details on how a motion planner is integrated with our final navigation intensity map I_{nav}^t (equation 7). At any time instant t , the values in the grids of I_{nav}^t represent free space, opaque, or linearly extrapolated transparent obstacles. Therefore, I_{nav}^t can be regarded as a *cost map* containing the navigability costs (zero cost for free space, high positive costs for obstacles) of the robot's surroundings. Therefore, a motion planner could calculate costs for potential/candidate robot trajectories using I_{nav}^t , and select the trajectory with the lowest cost for execution by the robot.

1) *Background on Motion Planners*: We integrate TOPGN with the Dynamic Window Approach (DWA) [1] to perform real-time navigation. In DWA, the robot's actions are represented as linear and angular velocity pairs (v, ω) . $V_s = [[0, v_{max}], [-\omega_{max}, \omega_{max}]]$ is defined to be the space of all the possible robot velocities based on the maximum velocity limits v_{max} , and ω_{max} . DWA formulates the following constrained velocity sets to compute dynamically feasible (i.e. executable by the robot) and collision-free velocities: (1) V_d , the dynamic window set contains the reachable velocities during the next Δt time interval based on the robot's acceleration constraints; (2) V_a , the admissible velocity space includes the collision-free velocities. The optimal velocity pair (v^*, ω^*) is then selected from the resulting velocity space $V_r = V_s \cap V_d \cap V_a$ by minimizing the following objective function:

$$Q(v, \omega) = \sigma(\gamma_1 \cdot head(.) + \gamma_2 \cdot obs(.) + \gamma_3 \cdot vel(.)). \quad (12)$$

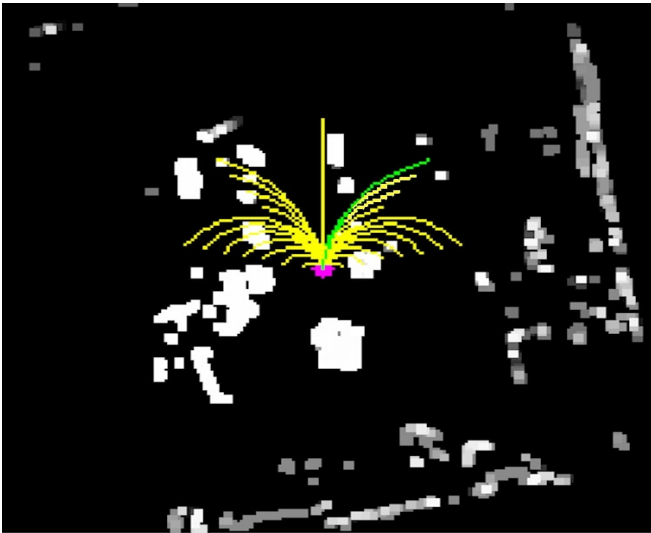


Fig. 11: Candidate trajectories (in yellow) shown relative to I_{nav}^t . The trajectory in green has the least-cost and is executed by the robot. The black regions represent free space, and grey/white regions represent opaque, and transparent obstacles.

where $head(.)$, $obs(.)$, and $vel(.)$ are the cost functions [1] to quantify a velocity pair's heading towards the goal, distance to the closest obstacle in the trajectory, and the forward velocity of the robot, respectively. σ is a smoothing function and γ_i , ($i = 1, 2, 3$) are adjustable weights.

2) *Trajectory Cost Calculation*: For our integration, the $obs(.)$ cost is calculated by extrapolating the trajectories that each (v, ω) pair in V_r would lead to within a time horizon t_{hor} . These trajectories are then transformed w.r.t I_{nav}^t using the transformation between real-world dimensions and grid locations as mentioned in equation 1. Then, the obstacle cost for the k^{th} trajectory can be calculated as,

$$traj_k^I = [(r_{1,k}, c_{1,k}), \dots, (r_{j,k}, c_{j,k}), \dots, (r_{lim,k}, c_{lim,k})] \\ obs(traj_k^I) = \frac{1}{\min(dist(\mathbf{O}^t, (r_{j,k}, c_{j,k})))}, \quad j = \{1, 2, \dots, lim\}. \quad (13)$$

Here, \mathbf{O}^t is the set of obstacles (all grids that are not black in Fig. 11) in I_{nav}^t . This cost calculation is repeated for every trajectory corresponding to a $(v, \omega) \in V_r$. The superimposed trajectories on I_{nav}^t is shown in Fig. 11.

For integrating the planner with SLAM [4], the trajectories are obtained w.r.t the map generated by the SLAM method. For integrating with segmentation methods [2], [3], the trajectories are transformed w.r.t the segmentation output image, and transparent obstacle costs are calculated. This is combined with the costs calculated by the planner [1].

B. Parameters and default values

We summarize the details of the important parameters used in our method. We demonstrate that **without changing any of the parameters** our method can be used with different 3D lidar sensors that have different vertical channel resolutions (See comparison between Velodyne VLP16 and Ouster OS1-32 in Fig. 8). However, we would like to highlight that the performance of our method for a given lidar sensor can be improved if certain parameters are tuned accordingly. Hence, we discuss the effect of the parameters and threshold values used in our approach in the table below.

Since our method's formulation depends on multi-layer intensity maps, it's mandatory to have enough vertical resolution in the point cloud to obtain at least 3 intensity layers. However, most commercially available 3D lidar sensors have at least 16 verticle channels or more which makes our algorithm compatible for general use. We further observed that even with enough vertical resolution, placing the lidar at a very low height can lead to difficulties in obtaining multi-layer intensities below the sensor's height level. Such issues can be mitigated simply by placing the sensor at a higher position. Hence, our overall approach can be easily deployed with any robot equipped with a 3D lidar.

Parameter	Description	Range	Default Value	Notes
n	2D Intensity map width/height	\mathbb{Z}^+	200	Size n of the intensity map indicates the sensing range of the lidar that we are interested in (~ 5 -meter radius around the robot in our case). For a fixed sensing range in the real world, larger intensity maps lead to better spatial resolution in the real world (denoted by s). Extremely low-resolution intensity maps can close the narrow passages between the objects which makes it difficult to navigate a robot.
s	Side length of the real-world square corresponding to a grid in a 2D map	\mathbb{Z}^+	0.05 meters	Smaller s values indicate that the intensity map has a higher spatial resolution and vice versa. The effect of this parameter is correlated with the size of the intensity map.
h_{lid}	Height of the lidar sensor location from the ground	\mathbb{Z}^+	0.48 meters	Height of the lidar sensor affects the TOPGN's performance only when the height is extremely low where even one additional intensity map layer cannot be defined below the sensor height level. Because our formulation requires at least one intensity map layer below the mid-intensity map (which is at the lidar's height level) to identify the TONs.
Δ	Height parameter that controls the height range considered for each intensity map layer	\mathbb{Z}^+	0.2 meters	Empirically chosen such that all the points with the highest intensity lie within $h_{lid} \pm \Delta$ when the lidar is d_{thresh} meters away from a completely transparent obstacle (threshold distance to maintain with obstacles). d_{thresh} is a robot-dependent parameter that can be obtained from the definition below. We believe that this is a simple calibration step one can perform if the hardware setup is significantly different from ours. Otherwise, our method will perform comparably with any similar robot setup without any changes to the parameters.
d_{thresh}	Threshold distance to maintain with a transparent obstacle	\mathbb{R}^+	1 meter	$d_{thresh} = 2 \cdot r_{rob} + 0.5$ meters. We observed that $d_{thresh} \sim 1$ meters for our Turtlebot robot. We defined this robot-dependant distance threshold after analyzing the point cloud intensity distribution for different transparent objects and by considering the safety clearance for a robot during navigation. Significantly smaller values can increase the risk of collisions with transparent obstacles.
m	Side length of an ROI	\mathbb{Z}^+	100	ROI is chosen such that it only covers the robot's nearby vicinity (~ 2.5 meters). Higher values closer to n could lead to artifacts (even though we are de-noising the ROI) in the final cost map (after the summation of multiple cost maps) since the ROI will include noisy objects further away from the robot.
r_{rob}	Radius of the robot	\mathbb{Z}^+	0.25	We use a Turtlebot 2 robot with a 0.25 meter radius for our experiments.

TABLE III: Details of the parameters used in our TOPGN approach.