3D Perception for Autonomous Robot Exploration

Jiejun Xu(✉), Kyungnam Kim, Lei Zhang, and Deepak Khosla

HRL Laboratories, LLC, Malibu, USA
{jxu,kkim,lzhang,dkhosla}@hrl.com

Abstract. We propose an online 3D sensor-based algorithm for autonomous robot exploration in an indoor setting. Our algorithm consists of two modules, a proactive open space detection module, and a reactive obstacle avoidance module. The former, which is the primary contribution of the paper, is responsible for guiding the robot towards meaningful open spaces based on high level navigation goals. This generally translates to identifying open doors or corridor vanishing points in a typical indoor setting. The latter is a necessary component that enables safe autonomous exploration by preventing the robot from colliding with objects along the moving path. Assuming a 3D range sensor is mounted on the robot, it continues to scan and acquire signal from its surroundings as it explores in an unknown environment. From each 3D scan, the two modules function cooperatively to identify any open spaces and obstacles within the generated point cloud using robust geometric estimation methods. Combination of the two modules provides the basic capability of a autonomous robot to explore an unknown environment freely. Experimental results with the proposed algorithm on both real world and simulated data are promising.

1 Introduction

The ability to explore unknown and dynamic environments is an essential component of an autonomous mobile robot system. The basic problem is the following: after being deployed into an unknown environment, a robot must continuously perform sensing operations to decide the next exploration location, and possibly maintain an internal representation of the environment. A well-studied and related problem is the SLAM (Simultaneous Localization and Mapping) problem, whose goal is to integrate the information collected during navigation into an accurate map [1]. However, SLAM does not address the question of where the robot should go next. Another related and well-known problem is path planning [2]. Unlike the original problem, where complete or partial knowledge of the environment is given, exploring an unknown environment is usually performed in a step-by-step greedy fashion. Instead of planning the entire trajectory of the mobile robot in advance, we emphasize an exploration strategy which plans one step (or a few steps) ahead based on the information about the environment acquired by the onboard sensors. In other words, the main problem we address

G. Bebis et al. (Eds.): ISVC 2015, Part I, LNCS 9474, pp. 888–900, 2015.
DOI: 10.1007/978-3-319-27857-5_79

in this work is determining where the robot will move next. Specifically we focus our attention on an indoor environment, such as corridors or warehouses. In such a setting, the next "good" candidate positions to explore are typically meaningful open spaces based on high-level navigation goals (e.g., open doors and vanishing point direction).

Another necessary component for autonomous robot exploration is the "sense and avoid" capability [3,4]. To ensure safe navigation, a robot must be able to achieve self-separation and collision avoidance with other objects (e.g., pedestrians and other robots) on its moving path. To this end, we propose a robust online algorithm consisting of two modules: a proactive open space detection module and a reactive obstacle avoidance module. We believe an effective combination of the two modules provides the basic capability to an autonomous robot to freely explore an unknown environment.

Many 2D-based vision algorithms have been proposed for robot exploration in both indoor [5,6] and outdoor spaces [7,8]. However, 2D visual information could be difficult to use in a 3D world, especially when the 3D structure needs to be inferred for the specific application [9]. The advent of inexpensive 3D (or RGB-D) sensing hardware such as the Microsoft Kinect, Asus Xtion and PrimeSense Capri sensors brings new opportunities to capture 3D environment in a more straightforward manner [10–12]. In this paper, we focus on 3D perception for the indoor exploration task. Our proposed algorithm functions by continuously acquiring and processing 3D data as it moves in an unknown environment. From each 3D scan, the two modules in the proposed algorithm work cooperatively to identify any open spaces and obstacles within the 3D point cloud using robust geometric estimation methods. The next moving direction can be determined by using results from both of these modules. A key benefit of our algorithm is that it is platform-agnostic and can be applied to both ground-based and aerial (e.g., Micro Air Vehicles) robots. We believe our work is applicable to other related areas, such as search-and-rescue, mapping, and 3D modeling.

2 Related Work

In this section, we provide a brief survey of existing literature related to our work on indoor exploration, specifically focusing on door detection and obstacle avoidance. Several visual door detection algorithms have been proposed since it is a topic of much relevance in both robot navigation and manipulation tasks. Most of the earlier works focused on extracting 2D appearance and shape features to recognize doors. For example, Murillo et al. [13] proposed a probabilistic approach by defining the likelihood of generating the door configurations with various features. Tian et al. [14] further proposed a more generic geometric-based approach to detect doors based on stable features, such as edges and corners. In [15], a dedicated real-time edge-based tracker is used to locate and track both open and closed doors within a corridor. A similar approach is also seen in [16]. However, as in most 2D vision algorithms, these approaches are sensitive to common factors such as illumination changes, occlusion, clutter, and perspective distortion. On the other hand, recent advances in sensor technology have enabled capture of 3D data with inexpensive range sensors (e.g., Kinect, Xtion). The additional depth information is especially helpful in detecting open spaces in indoor environments. In addition, the quality of 3D point cloud data is less sensitive to the aforementioned environmental factors. Recently, promising results have been shown by Rusu et al. [17] in detecting doors with only 3D point clouds. In their work, the main focus was to open and close doors through precise arm manipulation based on a pre-learned door template. Similarly Derry et al. proposed a 3D based algorithm to detect doors for assistive wheelchairs control [18]. Their algorithm imposes a prior assumption on the geometric orientation of the point cloud. For instance, the corners and boundaries of the doors are defined based on the assumption that the robot and doors reside on the same surface; thus it is not applicable to aerial robots. Furthermore, their algorithm only detects doors from a wall that is mostly front-facing. In contrast, our proposed algorithm explicitly computes the geometry of the 3D point cloud to detect open doors independent of robot position and is also capable of detecting doors from multiple walls at different angles with respect to the robot.

Another research area relevant to this paper is obstacle detection and avoidance. Due to the growing interest in small robots navigating in indoor and other GPS denied environments, quite a few research works have been carried out to utilize visual sensors to ensure safe maneuver in these conditions. Techniques focusing on monocular (2D) sensors have long been the main studied area. This includes an optical flow-based approach by Zingg et al. [5] and an imitation learning-based approach by Ross et al. [19]. In the recent years, there has been much interest in developing similar types of methods for 3D range sensor-based obstacle avoidance. However, regardless of 2D or 3D vision, a common theme across most works on obstacle avoidance is the reactive action control for robots. This means that the control system will react and prevent the robot from colliding with an impending obstacle. However, it does not plan ahead for exploring far away spaces. Our proposed algorithm overcomes this weakness by coupling obstacle avoidance with active open space detection in autonomous exploration.

3 Methodology

This section presents the details of our algorithm. We assume a 3D range sensor is mounted on the robot. In our work, we utilize the Xtion sensor. The device consists of two cameras and an Infrared (IR) laser light source. The IR source projects a pattern of dark and bright spots onto the environment. This pattern is then received by one of the two cameras which are designed to be sensitive to IR light. Depth measurements can be obtained from the IR pattern by triangulation. Subsequently, a point cloud is constructed internally in the sensor and is then requested by our algorithm directly [10]. Note that the returned point cloud is in real-world scale, and is described in a common metric system. Given the 3D point cloud, the outputs of the algorithm are the direction of the open space (e.g., open doors or the vanishing point) and the direction to avoid the obstacle (if applicable) with respect to the sensor coordinates. Both of these directions

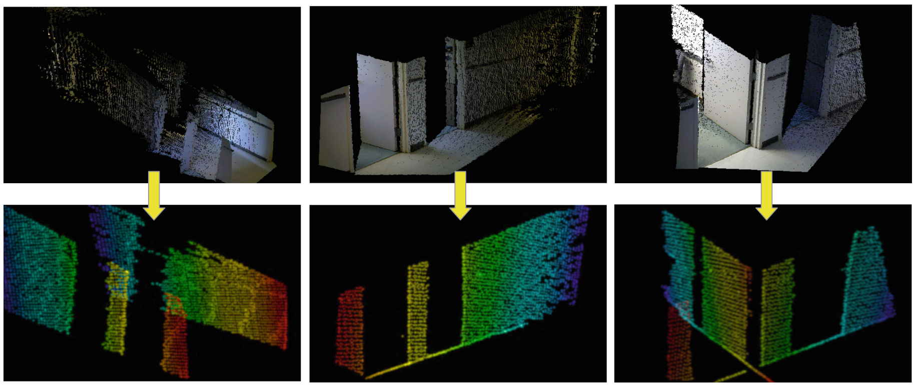


Fig. 1. Examples of vertical plane extraction. Top: input point clouds from an Xtion sensor. Note that viewing angle w.r.t sensor decreases from top to bottom. Bottom: extracted vertical planes after the filtering and downsamping step (color indicates distance from the sensor) (Color figure online).

can be combined to guide the robot towards the next exploration position. We now describe the two modules of the proposed algorithm.

3.1 Proactive Open Space Detection

Filtering and Downsampling: 3D range sensors typically generate voluminous data that cannot be processed in its entirety for many real-time applications. Thus the first step of our algorithm is to reduce the number of points in the point cloud by filtering and downsampling for efficient subsequent computations. Filtering consists of removing points which lie outside of a specific range. Typically an Xtion sensor has an effective range of 5 meters. Thus in our work, we remove all points beyond 5 meters.

In terms of downsampling the point cloud, a typical voxelized grid approach is taken. A 3D voxel grid is essentially a set of fixed width 3D boxes in space over the input point cloud data. In each voxel, all the points will be approximated by their centroid. A 3D voxel grid can be created efficiently with a hierarchical Octree [20] data structure. Each Octree node has either eight children or no children. The root node describes a cubic bounding box which contains all points. At every tree level, this space is further subdivided by a fixed factor, which results in an increased voxel resolution. In this work, we utilize the VoxelGrid functionality implemented in the Point Cloud Library (PCL). The size of each voxel is fixed at 0.05 meter. A significant portion of the points are removed by the end of this step.

Plane Extraction: In this step, we extract ground surface and walls (e.g., vertical planes which reside on the ground surface) from the previously processed point cloud. The ground surface is important as it serves as the key reference for geometrical estimates. The walls are the candidate search space for open doors in the context of our application. This step is essentially done by fitting a planar model to the point cloud and finding the ones with sufficient number of points. To speed up the search process, Random Sample Consensus (RANSAC) [21] algorithms is used to generate plane model hypotheses. The Point Cloud Library provides a convenient implementation to extract the planes in their parameter form $ax + by + cz + d = 0$. Planes are extracted according to their size in a sequential order. At each iteration, the set of points (inliers) aligned with the model hypotheses is selected as the support for the planar model, and they are archived and removed from the original point cloud. The remaining points will be used to identify the next best plane. This process continues to detect planes and remove points until the original point cloud is exhausted or its size reaches a certain threshold. Note that, for each detected plane, an additional step is taken to project all inlier points to the plane such that they lie in a perfect plane model. This makes subsequent computation more efficient and less error prone.

With the extracted planes, the next step is to determine the one that corresponds to the ground surface and the ones that correspond to the walls. Recall that the normal of a plane can be computed using the planar model coefficient directly. Given the plane ($y = 0$) with normal $\mathbf{n_0} =< 0, 1, 0 >$, the angle θ between an arbitrary plane $\mathbf{n_1} =< a_1, b_1, c_1 >$ and $\mathbf{n_0}$ is computed as $\theta = \arccos\left(\frac{b_1}{\sqrt{a_1^2+b_1^2+c_1^2}}\right) 180/\pi$. Thus, the ground plane can be identified by computing the angles between all planes with respect to $\mathbf{n_0}$ and keeping the one pair which is in parallel. If more than one pair is found, the lower plane will be kept. In our implementation, we allow a $\pm 15°$ to compensate for possible sensor movements. Similarly we can identify walls by keeping planes which are vertical to the identified ground surface. A similar strategy has been applied in [22] to identify walls, ceiling and floor to help maintain a Micro Air Vehicle (MAV) in the center of a corridor. Naturally this is also applicable to ground-based robots. Examples of extracted vertical planes from various input point clouds are shown in Fig. 1. Subsequent steps will operate on these planes for open door extraction.

Open Door Detection: At a high level, our algorithm scans horizontally at a certain height of the vertical plane (wall) for gaps which are within the width of a typical door. When a gap of the appropriate width is identified ($th_g^{min} \leq g_s \leq th_g^{max}$), a potential open door is detected. In order to detect open gaps, we first define a specific height h_s for scanning. Note that instead of defining the height with respect to sensor as in [18], we define the height with respect to the ground surface in the point cloud. After that, we collect a set of points which are nearby with respect to the reference height. A small distance threshold th_s (0.05 meter in our work) is used in our implementation to define the nearby radius. In actual implementation, our algorithm first goes through each point in the point cloud and computes its distance to the ground surface. The computation of the distance between a point and a plane in 3D is as follow:

Let $a_0x + b_0y + c_0z + d_0 = 0$ be the parameter form of the ground plane, and $\mathbf{x_q} = (x_q, y_q, z_q)$ be a point in 3D space. A vector from the plane to the point $\mathbf{x_q}$ is given by

$$\mathbf{w} = -\begin{bmatrix} x - x_q \\ y - y_q \\ z - z_q \end{bmatrix}. \tag{1}$$

Projecting $\mathbf{w}$ onto the normal vector of the plane $\mathbf{n_0} =< a_0, b_0, c_0 >$ gives the distance D from the point to the plane as

$$\begin{aligned} D &= \frac{|\mathbf{n_0} \cdot \mathbf{w}|}{|\mathbf{n_0}|} \\ &= \frac{a_0 x_q + b_0 y_q + c_0 z_q + d_0}{\sqrt{a_0^2 + b_0^2 + c_0^2}}. \end{aligned} \tag{2}$$

Since the point cloud has been previously filtered and downsampled, conducting the above operation on the point cloud is very efficient. A point whose distance to the ground surface is approximately equal to the defined height (i.e., $h_s \pm th_s$) will be included in the point set. Once the points are collected, the algorithm computes the dominant direction of the point set (either in X-axis or Z-axis), and then sorts all the points in the set according to the computed direction. After that, the algorithm goes through each point sequentially in the sorted order. All the gaps can be efficiently identified in just one pass by simply checking the distances between each consecutive pair of points. Whenever the distance between two consecutive points (p_s^1, p_s^2) falls within the pre-defined width range (0.8 to 1.6 meters in this work), a candidate open space is reported to be detected. Although it is usually enough to determine open doors in the indoor setting with one scan, our algorithm scans a wall at two different heights. An open door is only reported if the two detected gaps are sufficiently close. In the case when precise open space is required, we can then connect the boundary points based on the open gaps. In the rare event of a false positive, the obstacle avoidance module (see Sect. 3.2) will be activated to prevent collisions.

Vanishing Point Estimation: In the case where doors are not available, the robot should continue to explore the indoor environment via wall following or moving to the end of the corridor. This can be usually achieved by computing vanishing points from the sensed data. Typically they are estimated by finding the intersecting point of major line segments in 2D images [6,15]. In the 3D case, we take a more direct approach to approximate vanishing points based on the wall-floor intersecting line. This line has been shown to provide important geometrical heuristics in indoor navigation [15,23].

To detect the wall-floor intersecting line, we utilize the identified ground plane and vertical walls. The computation is carried out with a standard Lagrange multiplier approach suggested in [24]. Basically the intersecting line from the two planes is represented by a point on the line $\mathbf{x} = (x, y, z)$ and a directional vector $\mathbf{n} =< a, b, c >$ originating from the point. Thus the goal is to identify the two. Assuming the two planes (the wall and the ground surface) are given by the normal vectors, $\mathbf{n_1} =< a_1, b_1, c_1 >$ and $\mathbf{n_2} =< a_2, b_2, c_2 >$, and arbitrary points on the two planes are $\mathbf{x_1} = (x_1, y_1, z_1)$ and $\mathbf{x_2} = (x_2, y_2, z_2)$. The directional vector of the intersecting line is easily computed as the cross product of the two normal vectors, i.e., $\mathbf{n} = \mathbf{n_1} \times \mathbf{n_2}$.

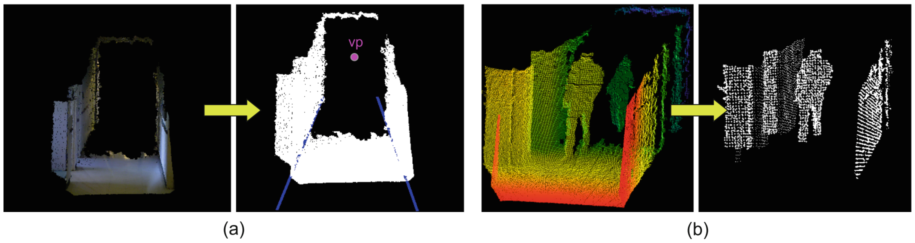


Fig. 2. (a) Left: a corridor snapshot. Right: detected wall-floor intersecting lines and the derived vanishing point. (b) Example of obstacle extraction. (Second pedestrian is not kept as he falls out of the predefined 5 m range).

To compute the point $\mathbf{x}$, we introduce another arbitrarily chosen point $\mathbf{x_0} = (x_0, y_0, z_0)$, which could simply be the origin or any other point. The point $\mathbf{x}$ on the intersecting line should be as close as possible to $\mathbf{x_0}$. This is a standard problem for Lagrange multipliers, with one objective function and two constraints: $(\mathbf{x} - \mathbf{x_1}) \cdot \mathbf{n_1} = 0$ and $(\mathbf{x} - \mathbf{x_2}) \cdot \mathbf{n_2} = 0$ (i.e., $\mathbf{x}$ must be on both planes). The cost function W containing both factors is

$$\begin{aligned} W &= \|\mathbf{x} - \mathbf{x_0}\|^2 + \lambda(\mathbf{x} - \mathbf{x_1}) \cdot \mathbf{n_1} + \mu(\mathbf{x} - \mathbf{x_2}) \cdot \mathbf{n_2} \\ &= (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 + \lambda x a_1 + \lambda y b_1 \\ &+ \lambda z c_1 - \lambda \mathbf{x_1} \cdot \mathbf{n_1} + \mu x a_2 + \mu y b_2 + \mu z c_2 - \mu \mathbf{x_2} \cdot \mathbf{n_2} \end{aligned} \tag{3}$$

where λ and μ are the two Lagrange multipliers. In order to solve for $\mathbf{x}$, we compute the partial derivatives $(\frac{\partial W}{\partial x}, \frac{\partial W}{\partial y}, \frac{\partial W}{\partial z}, \frac{\partial W}{\partial \lambda}, \frac{\partial W}{\partial \mu})$ and set them to zero. This can be rewritten as five linear equations stacking in matrix form

$$\begin{pmatrix} 2 & 0 & 0 & a_1 & a_2 \\ 0 & 2 & 0 & b_1 & b_2 \\ 0 & 0 & 2 & c_1 & c_2 \\ a_1 & b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} 2x_0 \\ 2y_0 \\ 2z_0 \\ \mathbf{x_1} \cdot \mathbf{n_1} \\ \mathbf{x_2} \cdot \mathbf{n_2} \end{pmatrix}. \tag{4}$$

Solving the linear system for (x, y, z, λ, μ) gives the point $\mathbf{x}$ on the intersecting line. Together with the directional vector $\mathbf{n}$, the wall-floor intersecting line is then determined. Finally, the vanishing point is obtained as the furthest point on the line parallel to the wall-floor intersecting line staring from the origin (i.e., sensor location). Figure 2(a) shows an example of detected wall-floor intersecting lines and the vanishing point for the input point cloud.

3.2 Reactive Obstacle Avoidance

In order for the robot to navigate autonomously, an important step is to enable it to avoid obstacles reliably in its environment. We have developed a variant of the navigation strategy proposed in [4] to determine an alternate moving direction for the robot when obstacles are present. The basic idea is to compute and compare open path lengths available to the robot for different angular directions. Prior to this step, the obstacles must be first identified. Recall that the ground plane has been identified and segmented out from the input point cloud for open space detection as described in Sect. 3.1. The rest of the point cloud contains points that may correspond to obstacles residing on the current moving path of the robot. Figure 2(b) shows the remaining points after the ground plane has been removed from the input point cloud. As can be seen, multiple clusters are clearly formed based on these points. Following this observation, we simply perform obstacle checks using these remaining 3D points. Given a point cloud O consisting of obstacles, a robot with radius r, and the current desired direction of travel θ_d, we first define the open path length $d(\theta)$ as a function of the direction of travel θ as: $d(\theta) = \min_{p \in \widehat{O}} \Big(\max(0, \Big\| p \cdot \widehat{\theta} \Big\| - r) \Big)$, assuming the origin of the coordinate system coincides with the sensor in the center of the robot. p is a 3D point in vector form, $\widehat{\theta}$ is a unit vector in the direction of the angle θ. Projecting p on $\widehat{\theta}$ through the dot product gives the open path length in the direction of θ. Subtracting r compensates for the size of the robot and excludes any open paths which are shorter than the robot radius. Note that we did not go through every point in the input point cloud as suggested in [4]. Instead, we modified the original strategy by only considering obstacle points $\widehat{O}$ within a certain range with respect to the desired direction of travel θ_d. In this work, only points within $\pm 45^\circ$ with respect to θ_d are considered. Subsequently the chosen obstacle avoidance direction θ^* is calculated as $\theta^* = \text{argmax}_\theta (d(\theta) cos(\theta - \theta_d))$. Essentially it trades off between: maximizing the open path length and minimizing the deviating angles.

Algorithm 1 summarizes the proposed autonomous robot exploration algorithm, which includes both proactive open space detection and reactive obstacle avoidance. Figure 3(a) shows a graphical example of the overall 3D point cloud processing by the proposed algorithm. Please refer to Figs. 4, 5 and 6 for more details.

4 Experiments and Results

We have fully implemented the proposed algorithm in C++ under the ROS (Robot Operating System) framework. Specifically, the proactive open space detection module and the reactive obstacle avoidance module are implemented in separate ROS packages complied with the general design principle. Experiments are carried out with a laptop computer with Intel Core i7 CPU and 8 GB of RAM. The computer is configured with ROS Hydro, OpenNi driver v1.5.4, and PCL (Point Cloud Library) 1.6.

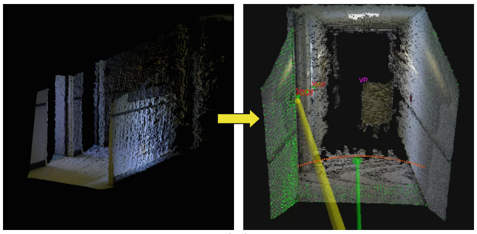
We primarily focus on two indoor scenarios, a corridor and a warehouse. In the first scenario, data is captured live by an Asus Xtion Pro RGB-D sensor mimicking the behavior of a ground-based robot. In the second scenario, data

Algorithm 1. Autonomous Robot Exploration

```
Input: A point could $P$ captured by a range sensor, and current desired direction $\theta_d$
Output: Direction $\theta_o$ for nearest open door, direction $\theta_v$ for vanishing point, direction
    $\theta_a$ to avoid obstacles if applicable.
 1: $P_f \leftarrow \{\}$, $O \leftarrow \{\}$, $D \leftarrow \{\}$, $V \leftarrow \{\}$, $A \leftarrow \{\}$
 2: $P_f \leftarrow DistanceFilter(P)$
 3: $P_f \leftarrow VoxelGridDownsample(P_f)$
 4: $N_p \leftarrow \alpha \cdot |P_f|$
 5: Find dominant ground plane within $P_f$; project nearby points onto the plane ($\rightarrow$
    $P_{ground}$)
 6: $O \leftarrow P_f \backslash P_{ground}$
 7: while $N_p < |P_f|$ do
 8:    $G \leftarrow \{\}$
 9:    Extract dominant plane, project nearby points onto plane ($\rightarrow P_{curr}$)
10:    if $P_{curr}$ is a vertical plane then
11:       Collect a set of points $L_s$ at predefined height $h_s \pm th_s$ w.r.t. ground plane
12:       Scan points in $L_s$ sequentially according dominant direction
13:       if $\|p_s^1 - p_s^2\| \in \left[th_g^{min}, th_g^{max}\right]$ for two consecutive points then
14:          $g_s \leftarrow (p_s^1, p_s^2)$
15:          $G \leftarrow \{G \cup g_s\}$
16:       end if
17:       (Optional) Repeat scanning at different height
18:       if $g_s \in G$ are consistent then
19:          $D \leftarrow \{D \cup mean(G)\}$
20:       end if
21:       Compute vanishing point $v_{curr}$ based on wall-floor intersecting line
22:       $V \leftarrow \{V \cup v_{curr}\}$
23:    end if
24:    $P_f \leftarrow P_f \backslash P_{curr}$
25: end while
26: Compute $\theta_o$ based on coordinates of nearest door in $D$
27: Compute $\theta_v$ based on coordinates of VPs in $V$
28: Compute $\theta_a$ based on $O$ and $\theta_d$
29: return $\theta_o$, $\theta_v$, and $\theta_a$
Note: $\alpha = 0.2$, $h_s = 0.8$, $th_s = 0.05$, $th_g^{min} = 0.8$, $th_g^{max} = 1.6$
```

is generated in simulation through Gazebo[1] mimicking the behavior of a aerial-based robot. In particular, we utilize the ROS Hector Quadrotor package[2] to model the dynamics and control of a MAV. Experimental parameters are consistent in both real world and simulated data with minor exceptions. Specifically, we increase the radius threshold th_s from 0.05 to 0.15 when collecting points for wall-scanning (see Sect. 3.1). This is because simulated point cloud appears to have different density (i.e.,sparser) compared to real world collected data. In

[1] http://wiki.ros.org/gazebo.
[2] http://wiki.ros.org/hector_quadrotor.



(a)

	Precision (Xtion)	Recall (Xtion)	Precision (Gazebo)	Recall (Gazebo)
Single wall single door	100%	100%	100%	100%
Single wall multiple doors	100%	100%	82.6%	100%
Multiple walls multiple doors	94.7%	100%	86.6%	93.3%

(b)

Fig. 3. (a) An integrated view showing various detections from 3D processing: open doors are marked with text "Door" (in red), yellow arrow indicates the nearest door, "VP" (in pink) indicates the detected vanishing point, green arrow indicates evading direction if obstacle is detected. (b) Quantitative results of open doorway detection on real world data captured with an Xtion sensor and simulated data generated in Gazebo (Color figure online).

addition, we adjust the distance filter to keep points within 8 meters from the sensor along the Z-axis, as simulated depth data has larger effective range.

In the first experiment, we focus on evaluating the effectiveness of the proactive open door detection module. Figure 3(b) summarizes the quantitative results of the proposed algorithm under different test configurations: single wall single door, single wall multiple doors, and multiple walls multiple doors. For each configuration, a total of fifteen test cases (at different angles and orientations) are randomly selected from experimental data. This is essentially done by taking point cloud snapshots from the 3D data. The first two configurations effectively cover open doors with angles ranging from 0° to 75° offset with respect to the sensor. The third configuration cover angle offset at about 90°. Overall, detection precision and recall are very high for real-world data (100 % in most cases). Detection results are slightly lower for simulation data, as the scenes are more diverse. We observe that most of false positives come from (non-door) open spaces that coincide with typical door width. Currently, the proposed algorithm does not provide a means to distinguish the two cases. However, this could be solved by introducing additional geometrical templates as done in [17]. Figures 4 and 5 show visual examples of detecting open doors in both real world and simulated data.

We also evaluate the obstacle avoidance module under different indoor settings. In total, we test the algorithm against six types of obstacles (e.g., pedestrian, box, and four types of chairs) in three different locations (e.g., corridor, office, and large conference room). In all cases, the reactive obstacle avoidance module is able to provide effective evading directions. Figure 6 presents a number of examples of the algorithm in action. The current implementation allows both modules to run simultaneously at about 10 Hz on the laptop computer, which makes it suitable for online applications.

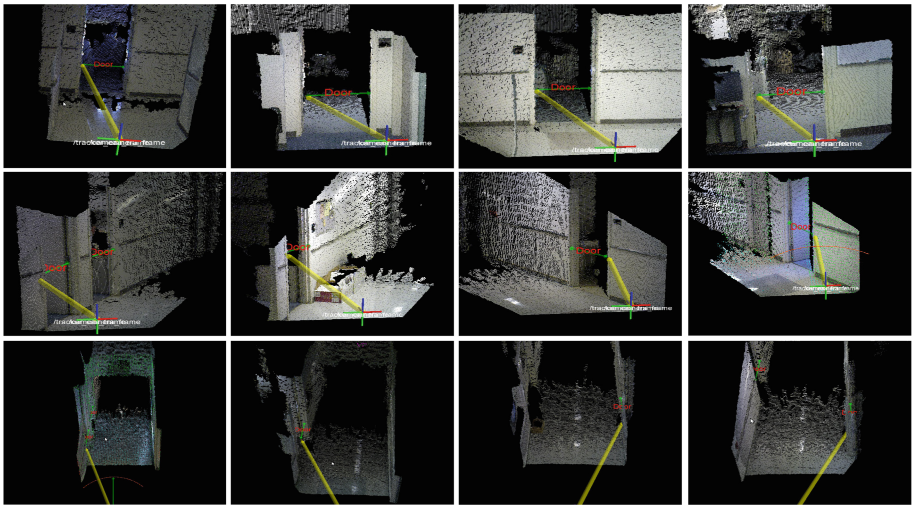


Fig. 4. Correctly detected doors from a ground-based robot platform using real world corridor data. Examples include doors detected at different angles (rows correspond to roughly 0°, 45°, 90°) and orientations with respect to the range sensor. Detected doors are indicated by green lines and marked with red text "Door". Yellow arrows indicate nearest doors to the robot based on the current scanned point cloud (Color figure online).

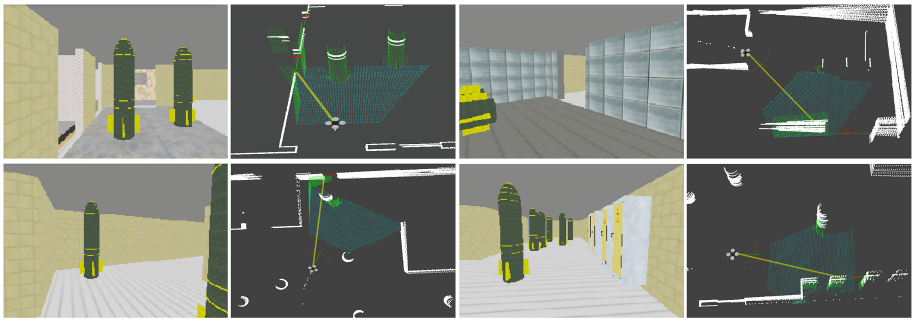


Fig. 5. Correctly detected doors from an aerial platform (i.e., MAV) using simulated warehouse data. Views from the MAV are shown as well.

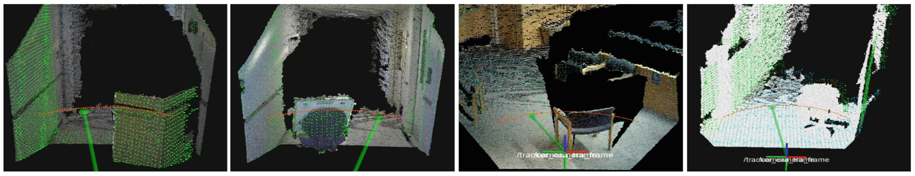


Fig. 6. Examples of obstacle avoidance. Green arrow indicates computed optimal moving direction to avoid collision. (Point cloud on the right is captured with a Capri sensor, thus lacks the color information) (Color figure online).

5 Discussion and Conclusions

This work presents an online algorithm for autonomous robot exploration in indoor settings using a 3D range sensor. Our system combines a proactive open space detection module with a reactive obstacle avoidance module to provide the basic autonomous functionality for a robot to navigate and explore an unknown environment. A key advantage of our proposed proactive algorithm's door detection capability is that it can detect doors accurately (1) irrespective of robot position and, (2) simultaneously from multiple walls at different angles with respect to the robot. By incorporating this accurate detection capability into a high level proactive controller and combining it with a reactive controller, we provide the robot with an autonomous exploration strategy. Promising results with this exploration strategy have been shown for a robot motion in both real and simulated worlds.

3D sensors are gradually becoming mainstream and available in small form factor at low cost. This makes them an attractive choice for mobile power-constrained robotic platforms such as MAV. This paper is a step towards enabling low-cost MAVs that can explore and map unknown indoor environments equipped with 3D range sensors. By combining the proposed algorithms with more traditional 2D sensors (e.g., camera) and algorithms, we can further improve the exploration and navigation capabilities and also extend navigation to outdoor settings. A future research direction is to integrate the proposed algorithms with long-term path planning and control schemes.

Acknowledgements. This material is based upon work supported by Defense Advanced Research Projects Agency under contract numbers W31P4Q-08-C-0264 and HR0011-09-C-0001. Any opinions, findings and conclusion or recommendations expressed in this material are those of the author and do not necessarily reflect the view of the Defense Advanced Research Projects Agency. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

References

1. González-Baños, H.H., Latombe, J.C.: Navigation strategies for exploring indoor environments. I. J. Robotic Res. **21**, 829–848 (2002)
2. Dudek, G., Jenkin, M.R.M.: Computational principles of mobile robotics. Cambridge University Press, New York (2000)
3. Lai, J., Mejías, L., Ford, J.J.: Airborne vision-based collision-detection system. J. Field Robot. **28**, 137–157 (2011)
4. Biswas, J., Veloso, M.M.: Depth camera based localization and navigation for indoor mobile robots. In: RGB-D Workshop in Robotics: Science and Systems (RSS) (2011)
5. Zingg, S., Scaramuzza, D., Weiss, S., Siegwart, R.: MAV navigation through indoor corridors using optical flow. In: ICRA. IEEE (2010)

6. Bills, C., Chen, J., Saxena, A.: Autonomous MAV flight in indoor environments using single image perspective cues. In: ICRA, pp. 5776–5783. IEEE (2011)
7. Fraundorfer, F., Heng, L., Honegger, D., Lee, G.H., Meier, L., Tanskanen, P., Pollefeys, M.: Vision-based autonomous mapping and exploration using a quadrotor mav. In: IROS. (2012)
8. Meier, L., Tanskanen, P., Heng, L., Lee, G.H., Fraundorfer, F., Pollefeys, M.: Pixhawk: a micro aerial vehicle design for autonomous flight using onboard computer vision. Auton. Robots **33**, 21–39 (2012)
9. Celik, K., Chung, S.J., Clausman, M., Somani, A.K.: Monocular vision slam for indoor aerial vehicles. In: IROS. IEEE (2009)
10. Cruz, L., Lucio, D., Velho, L.: Kinect and RGBD images: challenges and applications. In: SIBGRAPI Tutorials (2012)
11. Herbst, E., Ren, X., Fox, D.: RGB-D flow: dense 3-D motion estimation using color and depth. In: ICRA, pp. 2276–2282 (2013)
12. Shen, S., Michael, N., Kumar, V.: Autonomous indoor 3D exploration with a micro-aerial vehicle. In: ICRA, pp. 9–15. IEEE (2012)
13. Murillo, A.C., Kosecká, J., Guerrero, J.J., Sagüés, C.: Visual door detection integrating appearance and shape cues. Robot. Auton. Syst. **56**, 512–521 (2008)
14. Tian, Y., Yang, X., Arditi, A.: Computer vision-based door detection for accessibility of unfamiliar environments to blind persons. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A. (eds.) ICCHP 2010, Part II. LNCS, vol. 6180, pp. 263–270. Springer, Heidelberg (2010)
15. Sekkal, R., Pasteau, F., Babel, M., Brun, B., Leplumey, I.: Simple monocular door detection and tracking. In: IEEE International Conference on Image Processing, ICIP 2013, Melbourne, Australie (2013)
16. Fernández-Caramés, C., Moreno, V., Curto, B., Rodríguez-Aragón, J., Serrano, F.: A real-time door detection system for domestic robotic navigation. J. Intell. Robot. Syst. **76**(1), 119–136 (2013)
17. Rusu, R.B., Meeussen, W., Chitta, S., Beetz, M.: Laser-based perception for door and handle identification. In: International Conference on Advanced Robotics (ICAR) (2009)
18. Derry, M., Argall, B.: Automated doorway detection for assistive shared-control wheelchairs. In: ICRA, pp. 1254–1259 (2013)
19. Ross, S., Melik-Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A., Hebert, M.: Learning monocular reactive uav control in cluttered natural environments. In: CoRR (2012)
20. Meagher, D.: Geometric modeling using octree encoding. Comput. Graph. Image Process. **19**, 129–147 (1982)
21. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**, 381–395 (1981)
22. Lange, S., Sünderhauf, N., Neubert, P., Drews, S., Protzel, P.: Autonomous corridor flight of a UAV using a low-cost and light-weight RGB-D camera. In: Rueckert, U., Joaquin, S., Felix, W. (eds.) Advances in Autonomous Mini Robots. Non-series, vol. 101, pp. 183–192. Springer, Heidelberg (2012)
23. Pasteau, F., Babel, M., Sekkal, R.: Corridor following wheelchair by visual servoing. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2013, Tokyo, Japon (2013)
24. Krumm, J.: Intersection of two planes (2000)