A Viewpoint Planning and Navigation Algorithm for Mobile Robots using Depth Images

Davide Valeriani, Dario Lodi Rizzini, Fabio Oleari, Stefano Caselli
Dipartimento di Ingegneria dell'Informazione, University of Parma, Italy
{davide.v,dlr,oleari,caselli}@ce.unipr.it

Abstract

In this paper, we illustrate a viewpoint planning and local navigation algorithm for mobile robot exploration using 3D perception. Laser scans and stereo camera data are read and composed into a single point cloud. The latest acquired point clouds are registered and used to detect relevant objects and obstacles in space. To demonstrate the capability of the proposed perception system, a viewpoint planning application has been developed. The algorithm selects a goal object, computes a viewpoint to observe the region occluded by such object, and plans a path to reach the desired configuration. The navigation method executes the computed path and reaches the goal. Experiments have assessed the capability of the system to plan using three-dimensional occupancy information and to find new viewpoints in cluttered environments.

1 Introduction

The capability of navigating in unstructured and cluttered environments is a crucial task in service robotics. A typical human-populated environment is full of objects, possibly with protuberant parts, and may not be faithfully represented by a map built by a planar sensor. Furthermore, it would be convenient to qualify semantically interesting obstacles as objects and to obtain a complete view of the interesting parts of the environment. Thus, the perception of three-dimensional shapes significantly allows a mobile robot to perform high-level tasks and to achieve context-aware interactions with the environment.

The diffusion of cheap 3D sensors like range cameras and tilted laser scanners has popularised navigation in three-dimensional space and scene interpretation. The *point cloud* data structure provides a common representation for measurements acquired from multiple sensors. Since the point clouds acquired by range sensors are often rather accurate, navigation algorithms may exploit both occupancy and semantic information. Standard mobile robots move on a plane and their configuration can be often represented by two position and one orientation variables. Such convenient assumption, that avoids complex motion planning, holds even for mobile robots equipped with manipulators, since the robotic arm is usually idle during navigation. A worst-case estimation of free space can be achieved by projecting the points of the cloud on the motion plane. Several navigation algorithms operate under such assumption [1; 2]. However, a planning technique exploiting such approximation may not detect narrow passages among obstacles, since the free space is usually underestimated. More sophisticated methods discriminate between obstacles at different heights [3; 4].

Furthermore, 3D sensors allow a detailed perception of shapes which can improve performance in object detection, robot motion and manipulation in cluttered enviroments [5]. Most research on 3D perception has focused on scene segmentation and object recognition, and several effective algorithms have been proposed. However, detection is often addressed as a separate sensor-guided task that weakly interacts with mobile robot navigation. As a matter of fact, exploration of the environment can benefit from a raw semantic assessment of the 3D obstacle shapes and their relative placement [6; 7].

In this paper, we present an integrated system comprising viewpoint planning and a local navigation algorithm for mobile robot exploration using 3D perception. The proposed system acquires and stores sensor measurements from both a range finder and a stereo camera pair into a point cloud. The latest measurements belonging to a sliding time-window are aligned and accumulated in a single point cloud. The point cloud is used to build a planar local occupancy grid map which is used for planning robot path. The planar occupancy map is built using only points which may collide with the robot according to their height from the ground. Although the construction of the map is performed using a standard local grid map component that projects the 3D points on the ground plane, the points are selected so as to avoid unnecessary restriction of the free space. Moreover, the point cloud is divided into clusters of strongly connected points.

To demonstrate the perception capability of the proposed system, a viewpoint planning application has been implemented. In particular, a cluster of interest is selected, according to conditions on size and relative position, and the application determines a new viewpoint to achieve the observation of the region occluded by the object. Then, the path for reaching the viewpoint is planned using the occupancy grid map built projecting the point cloud on a plane. The local map keeps a short-term memory of the obstacles since most of the considered objects are ephemeral elements of the environment and their location can change rather frequently. The planned trajectory is executed using a variant of the dynamic window method, while the local map is updated during the robot motion. Finally, the system checks whether the goal is reachable and a new goal is planned using latest observations.

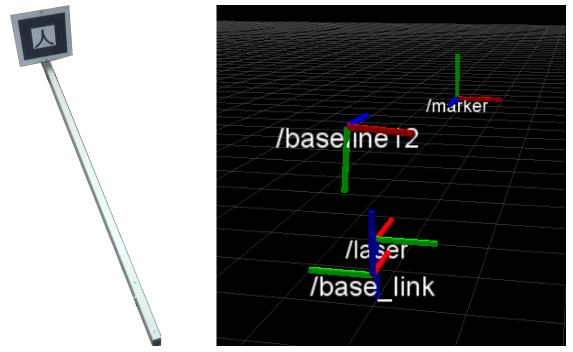
The main contribution of this work is the implementation of a complete robotic system that exploits 3D sensor sources for both occupancy and semantic segmentation using standard components. Moreover, the proposed approach avoids unnecessary restrictions of the free space available in the local map and achieves a safe navigation behaviour in cluttered environments.

The paper is organized as follows. Section 2 reviews the state of the art in navigation and semantic segmentation using 3D perception. Section 3 illustrates the acquisition, registration and processing of the point clouds obtained from sensors. Section 4 discusses the construction of the local map and the mobile robot navigation. Section 5 presents the viewpoint planning application. Section 6 presents the experiments performed to test the implemented system and section 7 discusses the results.

2 Related Works

Three-dimensional perception has increasingly been used to perceive obstacles and perform collision-free navigation. For most purposes the point clouds representing samples of occupied space can be converted into a standard planar representation by projecting points on the ground plane. Since mobile robots, like humans, move on such surface, this approximation can be reasonably applied to address most navigation problems where the robot keeps far away from obstacles. In [8] the 3D data acquired by a *MS Kinect* are used only to represent on the plane obstacles that may collide with the robot. In modern automated warehouses, the movement of goods is increasingly managed with autonomous robots whose perception of the environment and safety usually rely on planar laser scanners. Despite reliability of these sensors, problems may arise in case of hanging obstacles. In [9] the use a time of flight camera for obstacle detection is investigated as a safety device for Automated Guided Vehicles, but no sensor fusion is considered. Mapping and navigation with a *MS Kinect* sensor are also investigated in [2]. In particular, this work presents a comparison between 2D mapping by projecting points on the ground and 3D feature-based SLAM. Madder *et al.* [1] illustrate a system that uses a voxel grid map of the environment, but still a planar cost map is built by projecting voxel map columns and by expanding the robot footprint around obstacles. In [10] the points of the acquired depth images are sampled and fit into planes. Planes are used to perform both localization and navigation. Hornung *et al.* [4] propose to use both voxel map and multi-layer occupancy grid map to check collisions. Such solution allows the robot to move closer to obstacles when the robot does not collide to the grid map at a given height. The challenge is due to the complexity of 3D collision checking. A fast method to incrementally build 3D occupancy maps is illustrated in [3]. This paper is one of the few works addressing collision check in space. However, the reported experiments are performed offline on a previously acquired dataset. 3D perception has also been applied to navigation, exploration and localization of humanoid robots. Maier *et al.* [11] propose to use an octree occupancy grid map for localization, but path planning and navigation are executed on a projected planar map. These works mainly focus on how to use 3D perception for localization, mapping and navigation and use point clouds to infer spatial occupancy information.

A limited number of works exploit object detection as a mean to semantically segment the environment and to plan the execution of meaningful navigation tasks in the environment. The detected objects are often used to enrich maps with higher-order semantic information [12]. The work in [7] presents a sensor model that takes into account object detection and robot viewpoint. A convenient path is then planned as a trade-off between gaining additional information about an object hypothesis and the cost for executing such trajectory. Aydemir *et al.* [13] propose an algorithm for discovering objects using relations between objects. In particular, the observation of a specific object is used by the view planner as a clue for the presence of other objects (e.g. if a table is partially observed, the robot looks for possible objects *on* the rest of the table). Since the state of a mobile robot is described by its position and orientation on the ground plane, the view planning task is usually formulated as a planar problem.



(a) (b)

Figure 1: The rigid bar with the ARToolkit marker used for calibration (a) and the estimation of the relative reference frames (b).

3 Perception System

This section illustrates the 3D perception system developed in this work and the segmentation of the acquired point cloud. This task is the core of the whole robotic system, since it allows the estimation of collision-free paths and the decision of navigation goals. The 3D perception system consists of a pair of Logitech C270 webcams and a planar laser scanner Sick LMS100. The two cameras are fixed to a bar with approximately vertical orientation and a baseline of 12 *cm* in order to compose a stereo camera. Both the laser scanner and the camera are placed on a Pioneer 3DX mobile robot and are oriented to the front of the robot. The stereo camera acquires colored point clouds that can be used to detect objects and protruding 3D shapes, but has a limited field of view (about 20°) and returns rather sparse and noisy range measurements [14]. On the other hand, planar range finders are part of standard sensor equipment since they allow an accurate representation of occupied regions on the scanning plane, but are scarcely useful for object recognition. To overcome these limitations, sensor fusion is performed by accumulation of the synchronized measurements into a single point cloud. Furthermore, the latest point clouds are aligned and registered in order to build a short-term more extensive representation of the environment.

In the following, we present the sensor calibration, the processing and registration of point clouds, and the object detection components.

3.1 Calibration

The aim of the calibration procedure is the estimation of the intrinsic and extrinsic parameters that make the stereo camera and range finder work correctly. The intrinsic parameters are related to camera optics and other device features and are rather stable during time. The extrinsic parameters encode the relative position among the devices. An accurate estimation of these parameters is required to compute the disparity image by matching homologous points between the two cameras and to achieve a consistent point cloud from the fusion between range finder and stereo camera measurements. In particular, sensor fusion is achieved by overlapping measurements expressed w.r.t. the same reference frame. Furthermore, the assessment of the sensor pose w.r.t. the robot reference frame influences the results of point cloud registration.

The estimation of the intrinsic and extrinsic parameters of stereo vision is a long standing problem and several tools are available to perform calibration. In particular, we used the ROS package `camera_calibration` that allows joint estimation of intrinsic and extrinsic parameters using a checkerboard of known size. The relative position and orientation of the laser scanner frame w.r.t. the robot can be measured manually with satisfactory accuracy. On the other hand, the relative pose between the stereo camera and the robot are difficult to assess and may easily change due to accidental bumps. To overcome this problem, a target marker has been fixed to a rigid bar so that its pose (`marker`) w.r.t. the robot frame (`base_link`) is stable. The transformation from `base_link` to `marker` can be manually measured with satisfactory accuracy. The pose of the marker w.r.t. the stereo camera frame (`baseline12`) is estimated with the *ARToolkit* library [15]. The rigid bar with the marker is shown in Figure 1(a). Thus, camera measurements can be straightforwardly transformed from `baseline12` to `base_link` reference frame. Figure 1(b) shows all the frames used in this work. Hence, all the sensor data can be given w.r.t. the robot and overlapped.

In future works, we plan to add a plate on the rigid bar under the marker that can be observed by the range finder. The laser scanner would observe a segment of the target plate, thereby improving the estimation of sensor orientation and position. Thus, the position and orientation of the laser w.r.t. the marker would be measured directly and more accurately by the range finder.

3.2 Point Cloud Processing

The sensor data acquired by laser scanner and stereo camera are combined into a single point cloud that is used to navigate and to detect objects in the scene. The depth measurements tend to be sparse, due to failures in homologous point matching, and to be limited by the narrow camera field of view. For these reasons, the latest k point clouds are registered and accumulated.

Algorithm 1 shows the main operations performed to obtain a consistent point cloud. First, sensor data must be expressed in a single frame, the *robot frame*, in order to enable sensor fusion operations. Frame conversion is performed by applying to the point cloud the transformation matrix obtained in the calibration step. Due to the different rates, a software synchronization between these measurements is performed to merge them into a single consistent point cloud.

Then, three filtering operations are performed to clean the point cloud and remove noise from camera data. The first one removes points with *NaN* value, corresponding to points without depth generated by the stereo algorithm. The second filtering operation aims at removing isolated points, which are likely due to noisy perception or algorithm failures, and is performed using statistical outlier removal (SOR). The last one removes points too close to the camera and, thus, more subject to acquisition noise.

The corresponding laser scan point coordinates are transformed to the robot reference frame and processed for sensor fusion. Due to the lower amount of data acquired and the simpler data structure, this operation is quite fast and simple. The stereo camera point cloud and the laser point cloud are merged together with an operation of sensor fusion, with priority to laser scan due to the higher accuracy of the sensor, to obtain a single point cloud.

Finally, the accumulation of the latest k point clouds is performed. The previously acquired point clouds are stored in a FIFO vector and aligned to the current point cloud at iteration t. It is sufficient to estimate the relative pose T_t^{t-1} of the robot at time $t-1$ w.r.t. the current robot pose t and to apply such transformation to all the clouds stored in a vector $\mathcal{V}$. The computational complexity of the alignment is $O(N)$, where N is the number of point clouds to be aligned. The estimation of T_t^{t-1} is achieved using a 3D variant of *Iterative Closest Point* (ICP) algorithm, which is implemented in the Point Cloud Library (PCL) [16]. Before the execution of ICP, a regular spatial discretization is achieved by keeping the mean point for each occupied voxel. Such operation reduces the computation time of ICP and reduces association errors due to an irregular distribution of points.

Next, point clouds in $\mathcal{V}$ are added together to obtain a denser point cloud. This point cloud may cover a wider field of view, in particular when the robot changes its orientation. However, accumulation of unaligned point clouds would be deleterious because it would overlap sensor data referred to different scenes. Therefore, before point cloud addition, point clouds P_t^{t-i} and P_t^t points are associated using the PCL implementation of *kd-tree* algorithm. If the number of points that have a corresponding element in the other point cloud exceeds a certain threshold then the two point clouds will be fused; otherwise, this step will be skipped. Eventually, the perception system has all sensor data stored in a single point cloud that will be used for different purposes.

The execution of all the steps in Algorithm 1 and of the sensor data acquisition requires about 6.44 s on a notebook equipped with an Intel Core i7-2620M 2.7GHz, 8GB RAM. Although planning tasks do not require high update rate, such execution time does not meet typical time constraints. Thus, point cloud processing is computationally too expensive for the on-board laptops or embedded CPUs commonly mounted on mobile robots. In order to reduce the complexity, the most expensive operations in Algorithm 1 have been identified and a simplified pipeline has been implemented. In particular, the execution time almost halves by avoiding statistical outlier removal. Furthermore, the distributed execution of acquisition and point cloud processing on two separate hosts has allowed to reduce the computation time under 1 s without significantly reducing the quality of the output.

Algorithm 1: Point Cloud Processing

```
   Data: $\mathcal{L}$: laser scan data; $\mathcal{C}$: camera data with
         disparity;
   Result: $\mathcal{P}$: point cloud obtained by sensor fusion;
1  while ros::ok() do
2      store $\mathcal{L}$ and $\mathcal{C}$ in point cloud $\mathcal{P}_\mathcal{L}$ and $\mathcal{P}_\mathcal{C}$ respectively;
3      express $\mathcal{P}_\mathcal{L}$ and $\mathcal{P}_\mathcal{C}$ sensor data w.r.t. robot frame;
4      filter NaN values from $\mathcal{P}_\mathcal{C}$;
5      if outlier removal enabled then
6          filter outlier values from $\mathcal{P}_\mathcal{C}$ using SOR;
7      end
8      filter $\mathcal{P}_\mathcal{C}$ values too close to camera using
       PassThrough;
9      $\mathcal{P}_t = \mathcal{P}_\mathcal{L} \cup \mathcal{P}_\mathcal{C}$; /* spatial sensor fusion      */
10     if first point cloud processed then
11         insert $\mathcal{P}$ into point clouds vector $\mathcal{V}$
12     else
13         if $\mathcal{V}$ is full then
14             pop older point cloud;
15         end
16         compute transformation matrix to align $\mathcal{P}_t$ to
           $\mathcal{P}_{t-1}$;
17         $\mathcal{P}_{tot} \leftarrow \emptyset$; /* temporal sensor fusion      */
18         forall $\mathcal{P}_i$ in $\mathcal{V}$ do
19             apply transformation matrix to $\mathcal{P}_i$;
20             $\mathcal{P}_{tot} \leftarrow \mathcal{P}_{tot} \cup \mathcal{P}_i$;
21         end
22         $\mathcal{P} \leftarrow \mathcal{P}_{tot}$;
23     end
24 end
```

3.3 Object Detection

The point cloud obtained by registration and sensor fusion in the previous step can be used to detect objects in the scene. In this context, objects correspond to point clusters lying on the ground plane and matching given size and shape conditions. Cloud segmentation allows the identification of candidate goals for robotic tasks and of interesting regions that are occluded in the current view. Algorithm 2 shows in pseudocode the main operations needed to identify objects.

Algorithm 2: Object Detection

```
Data: $\mathcal{P}$: point cloud containing acquired sensor data;
Result: $\mathcal{CP}$: cluster point clouds; $\mathcal{OP}$: object point cloud;
while ros::ok() do
    if floor removal enabled then
        remove floor from $\mathcal{P}$;
    end
    extract clusters from $\mathcal{P}$ primitive and store in $\mathcal{C}$;
    forall $\mathcal{C}_i$ in $\mathcal{C}$ do
        $\mathcal{CP}_i \leftarrow \emptyset$;
        $\mathcal{OP}_i \leftarrow \emptyset$;
        $minDist \leftarrow \infty$;
        forall point $p_i$ in $\mathcal{C}_i$ do
            $dp_i \leftarrow p_i$ euclidean distance;
            if $minDist > dp_i$ then
                $minDist \leftarrow dp_i$;
            end
            push $p_i$ in $\mathcal{OP}_i$;
            color $p_i$ with cluster color;
            push $p_i$ in $\mathcal{CP}_i$;
        end
        push $\mathcal{CP}_i$ in $\mathcal{CP}$;
        compute dimensions of cluster bounding box;
        if cluster satisfies user settings then
            $\mathcal{OP} \leftarrow \mathcal{OP}_i$;
        end
    end
end
```

First of all, the dominant plane is removed using the *SAC Segmentation* method, included in PCL. Normally, the dominant plane is the ground plane on which the objects stand. The ground may not be detected when the point cloud is not dense enough or the camera orientation leaves out the floor. In order to reduce processing time, it is possible to disable this function and use a plane removal that relies on an a-priori knowledge of plane equation.

Next, clusters of unconnected points are extracted from the point cloud using the *euclidean cluster extraction* method. A cluster is a set of points similar according to feature and position in space. Only the relevant clusters remain after setting a proper minimum and maximum number of points.

Main operations carried out by the cluster extraction method are:

- initialization of an empty list $\mathcal{C}$ of clusters and an empty queue $\mathcal{Q}$ of points to analyze ;
- for each point $p_i \in \mathcal{P}$, where $\mathcal{P}$ is the point cloud:
  - add p_i to $\mathcal{Q}$;
  - for each point $p_j \in \mathcal{Q}$:
    - search neighbour points set P_j in a sphere of radius $r < d_{th}$, where d_{th} is a tolerance value;
    - for each neighbour $p_k \in P_j$, check if it has already been processed; if not, add it to $\mathcal{Q}$;
  - add $\mathcal{Q}$ to clusters list $\mathcal{C}$ and reset $\mathcal{Q}$ to an empty queue;
- ends when all points $p_i \in \mathcal{P}$ have been processed and inserted in the clusters list $\mathcal{C}$.

For each cluster extracted, a particular point cloud containing only cluster points is created. Moreover, the euclidean distance between a cluster and the robot is calculated as $d = x_p^2 + y_p^2 + z_p^2$, where (x_p, y_p, z_p) are the coordinates of the cluster point closer to robot. The dimensions of the bounding box containing the cluster are also calculated to estimate the object size. If the object size is compatible with dimensions defined by the user, the cluster will be identified as a candidate object of interest. Otherwise, the next cluster will be processed. The output of the simple object detection component is a list of point clouds along with their centroid positions and bounding boxes, which can be used by the path planner to compute a goal configuration.

4 Navigation

Main purpose of 3D perception is to enable secure navigation for the mobile robot. Several works presented in Section 2 reduce the role of 3D perception in navigation to a projection on the ground plane. In this way, however, 3D obstacle detection incurs in a worst case scenario. In the approach described in this paper, the 3D objects are projected on the ground plane only if a part of the robot collides with the obstacle at least in a given orientation. Thus, the ability to navigate through enclosed passages (such as under a table) is retained, while ensuring safety and efficient planning using a local 2D map. For example, the robot used in our experiments, shown in Figure 4, is equipped with a fork lift to grab specific objects and must avoid collision with hanging parts. Navigation behaviour can be divided into two tasks: construction of a local map, to identify occupied areas, and path planning and navigation, to calculate and execute a safe path to the goal.

The task considered in this work is the detection and complete observation of objects in the scene rather than the construction of a complete representation of all the explored environment. Object categories of interest include chairs, tables, furniture and small human artifacts that can be moved. Hence, a global map is not required in this case due to the presence of ephemeral information.

The first step of the implemented navigation behaviour is the projection on a plane of the point cloud obtained in previous tasks. In fact, although 3D perception describes obstacles distribution in the space, the mobile robot moves on a plane (ground) and a 3D local map is not required. However, the free space should not be unnecessarily restricted by projecting all points. Points above the maximum height of the robot are not considered, while different rules may be defined by identifying different layers and footprints for the robot height. Cells around every obstacle are identified as *inflated obstacles*, since the mobile robot is approximated as a point. One contribution of this paper, thus, is to limit the restriction of free space, even though the construction of the local map is performed through a standard component like `costmap_2d` node.

For correct planning, it is necessary to construct a short-term memory local map allowing the integration of sensors data acquired in the last time interval. This map is obtained by *registration* of the point clouds projected on the motion plane. Point clouds are previously aligned as described in section 3.2. Odometry provides another information used in the creation of the local map.

The local map 2 is organized as an expanded occupancy map to avoid collisions. The map creation process is performed by `costmap_2d` node, included in `move_base` ROS package.

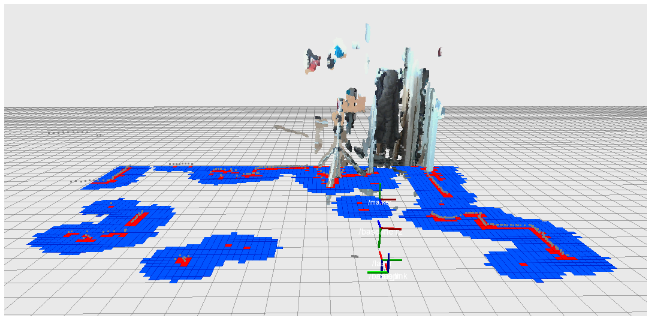


Figure 2: Example of short-term memory local map shown using Rviz.

5 Application: viewpoint planning

In this section, a viewpoint planning application of the system developed is presented. In particular, the 3D perception system has been used to compute a trajectory allowing the mobile robot to acquire different views of an object in the scene. In cluttered environments objects are often responsible for occlusions. Hence, the proposed application aims at detecting possible new viewpoints beyond the closest object. The system selects the target object using two criteria:

- *dimensions*: the object selected must have dimensions in ranges set by the user;
- *distance*: the object selected will be the one closer to the robot.

The object selected is approximated by its centroid. From this value, the goal configuration will be set at a certain distance, selected by the user. In order to see the object from the rear, the goal orientation is set as opposite w.r.t. the current robot orientation, as shown in fig. 3. In particular, if (x_c, y_c, z_c) are the coordinates of object centroid, d_c is the object depth and x_0 is the user defined distance, goal coordinates are set as $[x_c + d_c + x_0, y_c, z_c]^T$ and the orientation is toward the initial position of the robot.

After setting the goal, a local trajectory planner computes the best path to reach it, considering the short-term memory local map constructed in the previous step. Motion planning is performed by the `nav_core` node, included in ROS `navigation` package [1]. This node provides both local and global planners that use odometry to estimate robot position. However, in this work only the local planner has been used. The local trajectory planner uses an implementation of the Dynamic Window algorithm [17], whose main execution steps are shown in algorithm 3.

Algorithm 3: Main execution steps of DWA algorithm.

```
Data: $\mathcal{M}$: short-term memory local map;
Result: $\mathcal{C}$: velocity commands to the robot motors;
1 while true do
2     discrete sampling of the robot control space
      $(dx, dy, d\theta)$;
3     forall sampled velocity do
4         perform forward simulation from the robot
          current state to predict what would happen if
          the sampled velocity were applied for some
          (short) period of time;
5         if current trajectory crosses an obstacle then
6             discard trajectory;
7         else
8             assign a score to the trajectory dependent
              from proximity to obstacles, goal and global
              path and from the speed.
9     pick the highest-scoring trajectory and send the
      associated velocity to the mobile base.
```

The planner produces as output linear and angular speed values to be sent to the mobile robot to perform movement actions. If the goal is unreachable, due to the absence of free path to reach it, the planner fails and the robot stops its motion. The robot will also stop when it reaches the goal position with correct orientation. In both cases, the robot can forward to the next task, such as lifting the object or searching for other interesting entities in the environment.

6 Experiments

Figure 4(a) shows the Pioneer 3DX wheeled robot used in our experiments. The robot is equipped with a Sick LMS100 laser scanner and a stereo camera consisting of

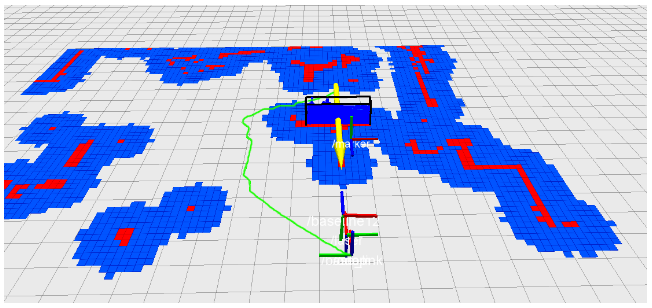
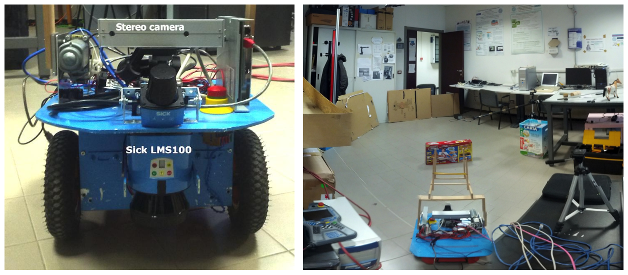


Figure 3: View of the complete navigation system: short-term memory local map, object detected enclosed in a black bounding box, goal configuration shown as a yellow arrow, planned trajectory shown in green.



(a) (b)

Figure 4: Robot platform used in experiments (a) and experimental environment (b).

two Logitech C270 cameras mounted in an aluminium case. The system uses ROS (Robot Operating System), an open source framework that includes several tools and libraries for robot programming. In this work, `costmap_2d` and `nav_core` standard nodes have been used respectively for short-term memory map computation and navigation behaviour. The values of main parameters used in our experiments are shown in Table 1 and Table 2.

Point cloud processing is subdivided between two nodes. The `cloud_processing` node is entrusted with sensor fusion and point cloud registration, while the `cloud_clustering` node is responsible of point cloud segmentation, object detection and viewpoint planning. The second node is based on the result of the first one; in fact, the same point cloud, processed by `cloud_processing` node, is used for both navigation and object detection. Before starting an experiment, the calibration bar described in section 3 is mounted on the robot. After few seconds, the calibration application is able to estimate position and orientation of the left camera. The rigid bar is then unmounted and calibration parameters are published to the other nodes.

Experiments were carried out in an indoor cluttered environment, comprising tables, packages, chairs, and other kinds of objects. The mobile robot was positioned in the center of the room (Figure 4(b)). The closest objects were at a distance of about 1.5 m, since the environment for experiments was cluttered. Initial tests showed that sensor data acquired from the stereo camera are quite noisy, especially the disparity image, so at least part of the filtering operations described in section 3.2 is required. Laser scanner data, instead, are very accurate, so no filtering was applied.

The robot tends to choose the objects in front of it due to the constrained field of view. In such cluttered environment it is likely to find a matching object in the front. If the robot does not find any object, the search continues in other regions and directions looking for farther objects. After having identified the object, the mobile robot reaches the region occluded by the object by planning and executing a trajectory path. The robot reaches the back of the target object avoiding obstacles, including those outside the scanning plane of the laser scanner, thereby proving the usefulness of 3D perception.

A sequence of images referred to a trial is shown in Figure 5. A black bar (shown on the left in the images) occludes one side to robot navigation. This bar cannot

Table 1: Main parameters used in `costmap_2d` node

Parameter	**Value**
obstacle_range	5.0 m
raytrace_range	5.0 m
inflation_radius	0.24 m
observation_sources	`point_cloud_sensor`
sensor_frame	`base_link`
data_type	`PointCloud2`
topic	`/cloud`
marking	*true*
clearing	*true*
observation_persistence	5.0 s
expected_update_rate	5.0 Hz
global_frame	`/odom`
robot_base_frame	`base_link`
update_frequency	5.0 Hz
publish_frequency	5.0 Hz
static_map	*false*
rolling_window	*true*
width	6.0 m
height	6.0 m
resolution	0.05 m

Table 2: Main parameters used in `nav_core` node

Parameter	**Value**
max_vel_x	0.2 m/s
min_vel_x	0.1 m/s
max_rotational_vel	1.0 rad/s
min_in_place_rotational_vel	0.4 rad/s
acc_lim_th	3.2 m/s^2
acc_lim_x	2.5 m/s^2
acc_lim_y	2.5 m/s^2
holonomic_robot	*true*

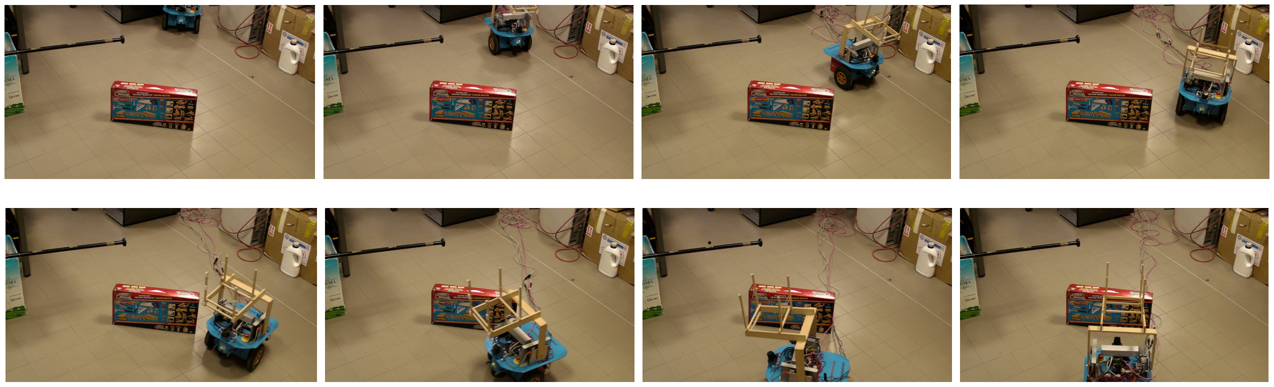


Figure 5: An experimental test (left to right, top to bottom). The robot keeps the left because on the right side of the obstacle a bar protrudes at a height that may lead to a collision.

be perceived by the laser scanner because it is above its scanning plane, whereas it is correctly perceived by the stereo camera. The robot uses this information to compute an occupancy grid suitable for safe navigation. Indeed, as shown in the figure, the robot reaches the object from the opposite free side.

Several experiments have been performed in order to test effectiveness and the robustness of the proposed system. For quantitative analysis, two controlled series of ten trials each have been carried out in an environment similar to the one in Figure 5. In the first set of trials the whole perception, planning and navigation application has been run on a single laptop on-board of the robot (Intel Core i7-2620M 2.7GHz, 8GB RAM). The results are shown in Table 3 (row (a)). The robot has successfully accomplished the exploration and planning task in 6 trials out of 10, with an average task execution time of about 42 *s* for successful trials. Failures are due to the computational complexity of 3D perception data processing. Indeed, the robot requires a new command before the time-out elapses, otherwise it stops and performs a shaky motion. When the new command is not received, the odometry error increases due to the irregular motion, thereby affecting alignment of sensor data. Moreover, the navigation algorithm relies on the occupancy grid map representing obstacles and unknown space: a temporal mismatch between the current map and the aligned point cloud may thus result into map inconsistencies.

A second set of trials has been performed by splitting the computational load between the on-board laptop and a more powerful workstation with an Intel Core7-3770 3.6GHz, 8GB RAM. In particular, the on-board laptop handles the acquisition of sensor data and the execution of robot motion, while the workstation performs the expensive point cloud processing task. Results for the second set of trials are shown in Table 3 (row (b)).

Table 3: Experimental results for reference task: success rate and task execution time. Processing: (a) fully on-board, (b) distributed.

	Trials	**Success Rate**	**Execution Time [s]**	
			Mean	St.Dev
(a)	10	60%	42.67	2.33
(b)	10	80%	30.69	2.35

In this case, the system succeeds in accomplishing the task in 8 out of 10 trials. Some failures still occur due to the computationally expensive 3D processing, thereby emphasizing the need of powerful computation for advanced sensor data processing. In the second set of trials, the time required to reach the target configuration is shorter, since the robot stops less frequently to wait for the incoming commands.

7 Conclusion

In this paper, we have presented an integrated system used for viewpoint planning and mobile robot exploration using 3D perception. The system acquires both laser scans and disparity images, and accumulates the sensor data into a single point cloud. The last k point clouds are registered to achieve a better representation of the scene. The 3D data are used both to build an extended occupancy grid map, that takes into accounts obstacles at different heights and not only on few planes, and to detect objects and interesting occluded regions, allowing the robot to perform safe goal-oriented navigation in the real scene.

Objects are detected by partitioning the point cloud into loosely connected components that meet requirements on size and shape. A specific object of interest is selected as a goal, and the algorithm computes a viewpoint for achieving its complete observation. A path is planned to reach the goal using the local map and the robot moves toward the goal configuration.

The developed 3D perception and navigation system has been tested with real-time sensor data acquisition on a physical robot. The experiments have proved the effectiveness in detecting relevant objects on the ground plane, in obtaining a short-term local map of the environment, in planning a collision free path and reaching the defined goal. Furthermore, they have shown the relevance of computation time constraints for the correct execution of a navigation. Future work will address applications exploiting 3D perception, such as global map computation or object classification according to multiple acquired viewpoints.

Acknowledgment

The authors wish to thank the reviewers for their constructive suggestions about how to improve the paper.

References

[1] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The Office Marathon: Robust navigation in an indoor office environment," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010, pp. 300–307.

[2] A. Oliver, S. Kang, B. Wünsche, and B. MacDonald, "Using the Kinect as a Navigation Sensor for Mobile Robotics," in *Proc. of Conf. on Image and Vision Computing New Zealand (IVCNZ)*, 2012, pp. 509–514.

[3] B. Lau, C. Sprunk, and W. Burgard, "Incremental Updates of Configuration Space Representations for Non-Circular Mobile Robots with 2D, 2.5D, or 3D Obstacle Models," in *Proc. of the European Conference on Mobile Robots (ECMR)*, 2011.

[4] A. Hornung, M. Phillips, E. Jones, M. Bennewitz, M. Likhachev, and S. Chitta, "Navigation in Three-dimensional Cluttered Environments for Mobile Manipulation," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012, pp. 423–429.

[5] M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. Le, A. Wellman, and A. Ng, "High-accuracy 3D sensing for mobile manipulation: Improving object detection and door opening," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009, pp. 2816–2822.

[6] K. Aydemir, A. Sjoo and P. P. Jensfelt, "Object search on a mobile robot using relational spatial information," in *Proc. of the Intl. Conf. on Intelligent Autonomous Systems (IAV)*, 2010.

[7] J. Velez, G. Hemann, A. Huang, I. Posner, and N. Roy, "Active exploration for robust object detection," in *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2011, pp. 2752–2757.

[8] J. Cunha, E. Pedrosa, C. Cruz, A. Neves, and N. Lau, "Using a Depth Camera for Indoor Robot Localization and Navigation," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2011.

[9] R. Bostelman, T. Hong, and R. Madhavan, "Towards AGV safety and navigation advancement obstacle detection using a TOF range camera," in *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, 2005, pp. 460–467.

[10] J. Biswas and M. Veloso, "Depth Camera Based Indoor Mobile Robot Localization and Navigation," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[11] D. Maier, A. Hornung, and M. Bennewitz, "Real-Time Navigation in 3D Environments Based on Depth Camera Data," in *Proc. of Int. Conf. on Humanoid Robotics (HUMANOID)*, 2012.

[12] B. Douillard, D. Fox, and F. Ramos, "Laser and vision based outdoor object mapping," in *Proc. of Robotics: Science and Systems (RSS)*, 2008.

[13] A. Aydemir, M. Godelbecker, A. Pronobis, K. Sjoo, and P. Jensfelt, "Plan-based Object Search and Exploration using Semantic Spatial Knowledge in the Real World," in *Proc. of the European Conference on Mobile Robots (ECMR)*, 2011.

[14] F. Oleari, D. Lodi Rizzini, and S. Caselli, "A Low-Cost Stereo System for 3D Object Recognition," in *Proc. of the Int. Conf. on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, Sept 2013.

[15] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *Proc. of the Int. Workshop on Augmented Reality*, 1999.

[16] R. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[17] D. Fox, W. Burgard and S. Thrun, "The Dynamic Window Approach to Collision Avoidance," *IEEE Robotics Automation Magazine*, March 1997.