J Intell Robot Syst (2015) 80:525–536
DOI 10.1007/s10846-015-0188-0

Real-Time Ground-Plane Based Mobile Localization Using Depth Camera in Real Scenarios

Miguel Vaz · Rodrigo Ventura

Received: 7 July 2014 / Accepted: 9 January 2015 / Published online: 25 January 2015

Abstract Existing robot localization methods often rely on particular characteristics of the environment, such as vertical walls. However, these approaches loose generality once the environment does not show that structure, e.g., in domestic environments, making the deployment of autonomous self-locating robots difficult. This paper addresses the problem of absolute online self-localization in a known map, where the only required structure in the environment is a planar ground. In particular, we rely on the transitions between the ground and any other non-planar structure. The approach is based on the ground point-cloud and plane model perceived by a depth-camera. The ground detection algorithm is robust to small shifts on camera orientation during the robot motion, by determining the calibration parameters on-the-fly. Then the edges of the ground point-cloud are estimated, which can be originated by obstacles in the environment. The localization is obtained using a particle filter fusing the odometry with a novel observation model reflecting the quality of the match between the ground edges and the nearest obstacles. For this purpose, a cost function was implemented based on a distance-to-obstacles grid map. Experimental results using the ISR-CoBot robot are presented, run in different scenarios, including a bookshop during working hours.

M. Vaz · R. Ventura (✉)
Institute for Systems and Robotics, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
e-mail: rodrigo.ventura@isr.ist.utl.pt

M. Vaz
e-mail: miguelvaz@ist.utl.pt

Keywords Kinect · Ground detection · Particle filter · Point-cloud · Indoor mobile robot · Map-based position estimation · RGB-depth sensor · Boundary edges estimation · Absolute self-localization

1 Introduction

The use of autonomous assistance robots in home environments is becoming increasingly necessary. Their use is also imperative in other places where they can help or provide general information related to their environment. As far as the home use is concerned, robots are able to help in daily tasks or assist elderly people. As far as non-home use, robots can give access to information, serve as an intermediary over long distance or even improve the quality of life of patients with motion restrictions. One of the fundamental capabilities from this kind of applications is the full autonomous self-localization. This paper addresses this problem for general indoor environments.

A number of publications have addressed the localization problem in a variety of different approaches. Nowadays there are two major approaches: the former is based on Kalman filter solutions [9, 21] and its more recent developments [5] resorting to local sub-maps to solve the unbounded growth of the filter state overtime or [16] tackling its difficulty to recognize already seen features resourcing to a joint compatibility data association. The latter is based on the Particle Filter solutions [1, 11] and its more recent developments [12] resorting to a statistical approach to adapt the size of the particle set on-the-fly or [7]. A comparison of the two approaches can be found in [13]. The appearance of cheap depth-cameras, despite often computationally complex to process, provides a good resolution and perception of the 3D environment at a low price.

Recent work has been performed on the localization problem using 3D data sensors. Some methods based their observation model on wall-planes features, like in [6] and in [2], which use the depth information to detect walls or other vertical planes features and project them on a 2D map. However, besides their remarkable performance, these algorithms show some limitations, particularly in environments where walls are difficult to detect - hidden by furniture or not present (open-spaces). Furthermore, since they consider the pose of the camera relatively to the ground plane to remain constant, they are not robust to oscillations that naturally occur during robot motion. Other methods approach the problem from a different perspective, by using the 3D data for building 3D maps of indoor environments and consequently estimating the pose by data matching [3, 15] ,but the computation effort of these algorithms is very high, unpractical to be used in real-time.

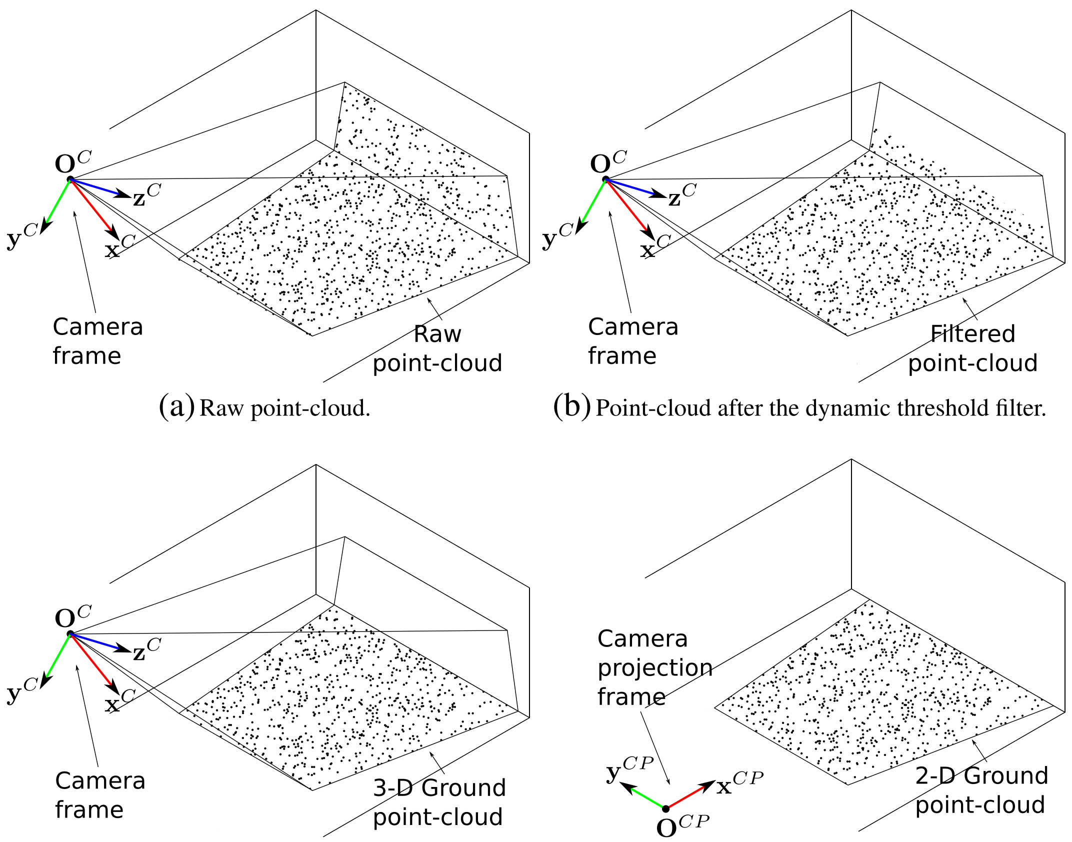
We tackled the localization problem using the RGB-Depth camera and the particle filter method to address some challenges found in the literature: generality and robustness. The developed system assumes a semi-structured environment based on a flat floor. An RGB-Depth camera is mounted on a differential wheel robot, pointing forward slightly down. The camera outputs a point-cloud, which is composed of a list of points, defined by their 3D position in space and their colour. Each 3D point of the set is associated to one pixel on the image plane. The robot used, the ISR-CoBot robot [20], is equipped with the aforementioned RGB-Depth sensor, a joystick for motion control and a differential drive kinematics for locomotion.

The approach taken is thus performing the localization based on the ground point-cloud. We assume that the camera is always pointing both forward and to the floor, while fixed to the robot (see Fig. 2). The most important features of the floor are its boundaries, as these are the elements that perfectly define it. The localization system is then implemented by combining a dead reckoning based estimation with an absolute localization system based on the ground point-cloud boundaries. The absolute localization system is implemented with the particle filter algorithm using the ground floor boundary seen by the camera, extracted from the previously detected ground point-cloud. Therefore, in this work, a full-resolution ground point-cloud detection system was developed, which made the detection of the ground plane model and the estimation of a robust point-cloud boundary possible. These methods helped us create an innovative system, which merges these environment features on the particle filter, by resorting to a special cost function. Our approach uses a ground detection algorithm that is not sensitive to the absence of planar vertical features, usable for semi-structured environments and work in real-time.

This paper is structured as follows: Section 2 is devoted to the aspects of the ground point-cloud detection. In Section 3 the implementation of the ground-Plane Boundary Estimation, using the resulting ground point-cloud, is evaluated by proposing the general detection algorithm and an outliers filter. Section 4 presents the localization system and explains how we combined the ground boundary edges with it. Section 5 presents the tests and results of this work using ISR-CoBot in the *Barata*® bookshop, and compares it with two other localization algorithms. Finally, Section 6 presents our conclusions.

2 Ground Point-Cloud Detection

The ground point-cloud detection algorithm (see Fig. 1) has three main steps: 1) detect the ground point-cloud in the sensor space; 2) estimate the ground plane model based on the ground point-cloud; and 3) evaluate the transformation between the sensor frame and a newly defined frame coupled with the ground, based on the plane parameters. But before going into the ground point-cloud detection algorithm, an explanation of the framework used and the problem geometry in this algorithm will be provided in 2.1 which will be used ahead. In 2.2 we will explain how we detected the ground



(a) Raw point-cloud. (b) Point-cloud after the dynamic threshold filter.

(c) Point-cloud after the RANSAC. (d) Point-cloud after the rigid transformation.

Fig. 1 Illustration of the ground point-cloud detection algorithm steps

point-cloud from the raw point-cloud data and the plane model parameters used in this mathematical framework.

2.1 Ground Point-cloud Detection Framework

The ground is modeled as a plane and parametrized using the implicit normalized form of a plane equation defined below in the camera frame $\{C\}$:

$$a^C x_{\mathbf{G}^C} + b^C y_{\mathbf{G}^C} + c^C z_{\mathbf{G}^C} + d^C = 0\,, \tag{1}$$

where $\left[\, a^C \;\; b^C \;\; c^C \,\right]^T$ is the normal vector $\overrightarrow{\mathbf{n}}$, $x_{\mathbf{G}^C}$, $y_{\mathbf{G}^C}$ and $z_{\mathbf{G}^C}$ are the coordinates of a point $\mathbf{G}^C$ belonging to the ground-plane, and $d^C = -\overrightarrow{\mathbf{n}} \cdot \mathbf{G}^C$ is the distance of the origin $\mathbf{O}^C$ the plane along $\overrightarrow{\mathbf{n}}$. It has to be noted that the (1) is in fact a specification of the equation of the distance of an arbitrary point to the plane (points that have zero distance), and therefore we have:

$$\begin{aligned} height(\mathbf{P}^C) &= \overrightarrow{\mathbf{n}} \cdot \mathbf{P}^C + d^C \\ &= a^C x_{\mathbf{P}^C} + b^C y_{\mathbf{P}^C} + c^C z_{\mathbf{P}^C} + d^C\,, \end{aligned} \tag{2}$$

where $\mathbf{P}^C = \left[\, x_{\mathbf{P}^C} \;\; y_{\mathbf{P}^C} \;\; z_{\mathbf{P}^C} \,\right]^T$ is an arbitrary point. This equation will be particularly useful to estimate the height of a point in the point-cloud (the distance from the ground).

Figure 2 pictures the geometry of the problem, where one can see the robot base frame $\{R\}$ (represented by $\{\mathbf{x}^R, \mathbf{y}^R, \mathbf{z}^R\}$); the RGB-Depth camera frame $\{C\}$ (represented by the orthonormal base $\{\mathbf{x}^C, \mathbf{y}^C, \mathbf{z}^C\}$) observing the floor; and the new camera-projection frame $\{CP\}$ coupled to the ground (represented by the orthonormal base $\{\mathbf{x}^{CP}, \mathbf{y}^{CP}, \mathbf{z}^{CP}\}$). The $\{CP\}$ frame is defined with its origin equal to the projection of the $\{C\}$ origin in the

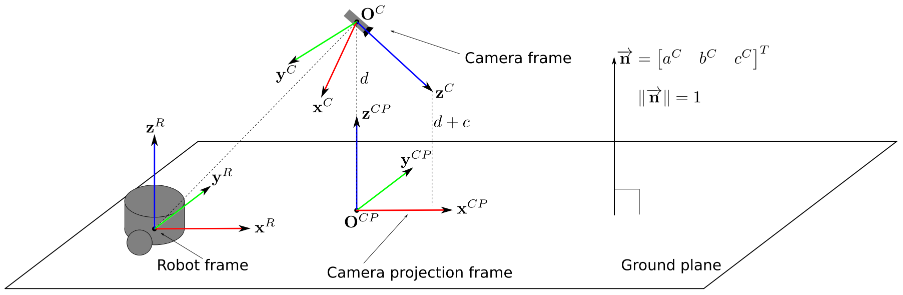


Fig. 2 Ground-plane estimation geometry

ground-plane, the $\mathbf{x}^{CP}$ axis with the same horizontal direction of $\mathbf{z}^C$, the $\mathbf{y}^{CP}$ left and the $\mathbf{z}^{CP}$ pointing up ($\vec{\mathbf{n}}$).

The ground-plane parameters are used to estimate the coordinates of the camera-projection $\{CP\}$ basis axes in the camera $\{C\}$ frame to then compute the rigid transformation between the two, completely describing the geometry of the problem as in Fig. 2. Therefore, considering the ground-plane model (a^C, b^C, c^C, d^C) as in (1), the estimation of the camera-projection $\{CP\}$ basis in the camera $\{C\}$ frame, denoted $\mathbf{x}^C_{CP}$, $\mathbf{y}^C_{CP}$ and $\mathbf{z}^C_{CP}$, is performed in two steps: first it is computed its origin $\mathbf{O}^C_{CP}$ in the $\{C\}$ frame. Since $\mathbf{O}^C_{CP}$ is the projection of $\mathbf{O}^C = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, it is obtained by translating the latter in the direction of $\vec{\mathbf{n}}$ and with the distance equal to the camera height, which equals d^C. Therefore $\mathbf{O}^C_{CP}$ is:

$$\mathbf{O}^C_{CP} = \mathbf{O}^C - (d^C)\vec{\mathbf{n}} \,. \tag{3}$$

The second step is estimating the $\mathbf{x}^C_{CP}$ unitary vector by projecting the $\mathbf{z}^C$ vector on the plane to obtain its horizontal orientation and then normalize the result. Following the same procedure as in (3), the projection of $\mathbf{z}^C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ is translating it in the direction of $\vec{\mathbf{n}}$ by the distance of its height. Using (2), we obtain:

$$\mathbf{x}^C_{CP} = \frac{\mathbf{z}^C - (c^C + d^C)\vec{\mathbf{n}} - \mathbf{O}^C_{CP}}{\|\mathbf{z}^C - (c^C + d^C)\vec{\mathbf{n}} - \mathbf{O}^C_{CP}\|} \,. \tag{4}$$

Since $\mathbf{z}^C_{CP}$ is defined as pointing up, it is equal to the normal plane vector, or in other words:

$$\mathbf{z}^C_{CP} = \begin{bmatrix} a^C & b^C & c^C \end{bmatrix}^T \,. \tag{5}$$

Finally, the $\mathbf{y}^C_{CP}$ is the result of the external product of $\mathbf{z}^C_{CP}$ and $\mathbf{x}^C_{CP}$, as defined in the Cartesian systems:

$$\mathbf{y}^C_{CP} = \mathbf{z}^C_{CP} \times \mathbf{x}^C_{CP} \,. \tag{6}$$

The rigid transformation between the two frames is then computed using the orthogonal Procrustes problem [14]. Considering that $\mathbf{A}$ is the orthogonal basis matrix of $\{CP\}$ and $\mathbf{B}$ is the orthogonal basis matrix of $\{C\}$, we have:

$$\mathbf{A} = \begin{bmatrix} \mathbf{x}^C_{CP} & \mathbf{y}^C_{CP} & \mathbf{z}^C_{CP} \end{bmatrix} \tag{7}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{x}^C & \mathbf{y}^C & \mathbf{z}^C \end{bmatrix} = \mathbf{I}_{3\times 3} \,, \tag{8}$$

where the orthogonal Procrustes problem states that finding the orthogonal matrix ${}^{CP}_{C}\mathbf{R}$ (rotation matrix), which most closely maps $\mathbf{A}$ to $\mathbf{B}$, is equal to finding the nearest orthogonal matrix of $\mathbf{M} = \mathbf{A}^T\mathbf{B}$. Therefore, using the singular value decomposition ($\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^*$), we have:

$${}^{CP}_{C}\mathbf{R} = \mathbf{U}\mathbf{V}^* \,. \tag{9}$$

The translation vector can be estimated directly from the geometry of the problem since is performed only along $\mathbf{z}^{CP}$ by the absolute value of the height of the camera. The translation ${}^{CP}_{C}\mathbf{t}$ is equal to:

$${}^{CP}_{C}\mathbf{t} = \begin{bmatrix} 0 & 0 & |d^C| \end{bmatrix}^T \,. \tag{10}$$

Using the transformation Eqs. (9) and (10), the ground point-cloud is transformed from the original frame (camera frame $\{C\}$) to the camera-projection frame. Since $\{CP\}$ is contained within the ground-plane, the resulting component of $\mathbf{z}^{CP}$ of the point-cloud is equal to zero. Therefore, discarding $\mathbf{z}^{CP}$, we obtain a 2D representation of the ground point-cloud that is more suitable given the aim of this work. Figure 1d illustrates the transformation.

2.2 Ground Point-cloud Detection Algorithm

An initial calibration of the floor to obtain pre-established values for the ground plane model is required. The calibration also serves for the estimation of the camera orientation and height on the robot. The calibration process is shown in Fig. 3, where a

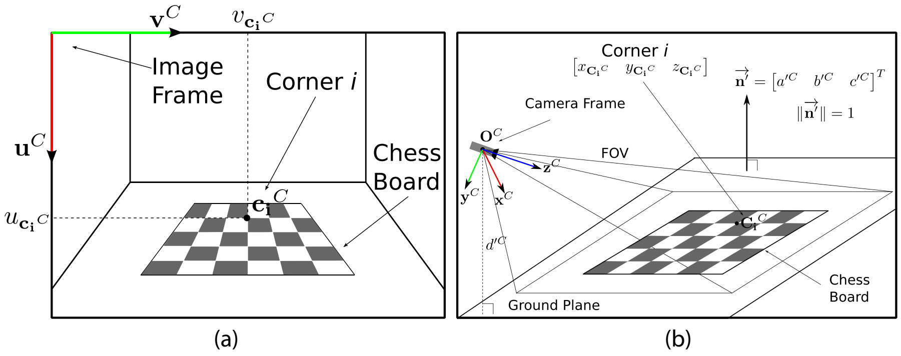


Fig. 3 Calibration setup. **a** RGB camera perspective, **b** 3D view

chess pattern is placed in the line of sight of the depth camera, maintaining the robot still. A chess pattern detection algorithm [22] is used to detect the corners pixels on the image. Then we obtain the corresponding 3D points from the point-cloud, that we know are part of the ground-plane.

Denoting $\mathscr{G}^C = \left[\, \mathbf{G}_1^C \cdots \mathbf{G}_i^C \cdots \mathbf{G}_n^C \,\right]$ as the list of n 3-D corner points in the camera frame found, we estimate the calibrated ground-plane parameters (a'^C, b'^C, c'^C, d'^C) using the linear least squares (LLS) estimator [17] to find the values which best fit the data. A rescale of the parameters was performed since the number of DOF of the plane model (1) used is bigger than the space it is defined in. This way, to solve the LLS we have:

$$\mathbf{y} = \mathbf{X} \cdot \boldsymbol{\beta} \tag{11}$$

$$\mathbf{X} = \left[\, \mathscr{G}^C \,\right]^T , \quad \boldsymbol{\beta} = \begin{bmatrix} a'^C \\ b'^C \\ c'^C \end{bmatrix} , \qquad \mathbf{y} = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix} \tag{12}$$

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}\,. \tag{13}$$

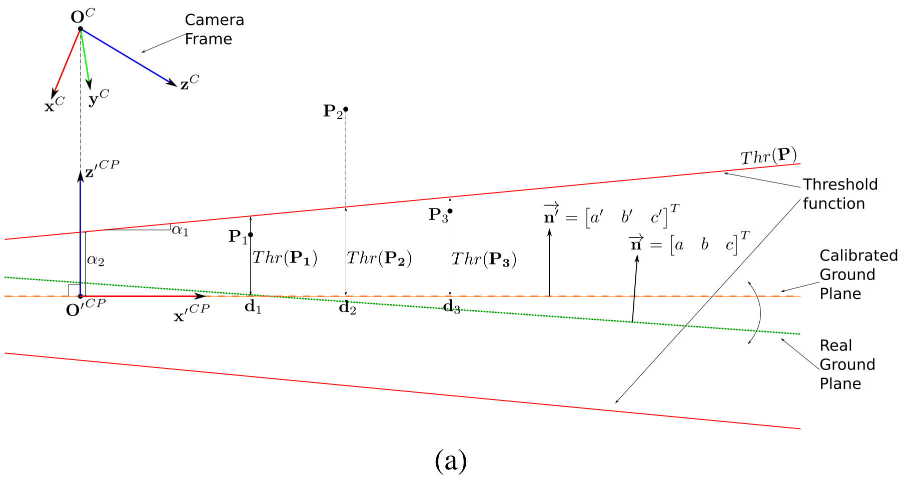
After obtaining the non-normalized solution $(\hat{a'}^C, \hat{b'}^C, \hat{c'}^C, 1)$, we multiply the result by its normalized factor, $1/\| \vec{\mathbf{n}'} \|$, obtaining the final normalized calibrated ground-plane model parameters.

The problem of detecting the ground point-cloud and estimating the ground-plane parameters, during the robot movements, is solved by analyzing the point-cloud, filtered by a dynamic threshold function, using a random sample consensus algorithm (RANSAC) [10]. The dynamic threshold filter is defined as two planes symmetrical along the calibrated ground-plane model. The distance between the planes and the calibrated ground-plane model gradually increases as one moves further away from the robot, as shown in Fig. 4 where the orange/dashed line represents the calibrated ground-plane, the green/dotted line stands for the true position of the ground and the red line indicates the value of the threshold function. Since the position of the ground-plane changes significantly in relation to the camera during robot motion due to the vibrations, the calibration process by itself is not enough to detect the floor point-cloud on-the-fly. The design of this dynamic threshold filter ensures a very robust and trustworthy detection system during the operation of the robot.

The framework described in Section 2.1 is equally applicable to the calibrated plane model, and so we denote $\mathbf{O}'^{CP}$ as the camera origin projection on the calibrated model, the corresponding orthonormal basis as $\{\mathbf{x}'^{CP}, \mathbf{y}'^{CP}, \mathbf{z}'^{CP}\}$ and $\{\mathbf{x}'^C_{CP}, \mathbf{y}'^C_{CP}, \mathbf{z}'^C_{CP}\}$ as the same basis but in the camera frame $\{C\}$. The dynamic threshold function equation is the following:

$$Thr(\mathbf{P}^C) = \alpha_1 \mathbf{P}^C \cdot \mathbf{x}'^C_{CP} + \alpha_2\,, \tag{14}$$

where $\mathbf{P}^C$ is an arbitrary point in the $\{C\}$ frame, the α_1 is the slope of the dynamic threshold function and α_2 is the maximum height allowed at the origin projection. α_1 and α_2 are therefore the regulation parameters of the filter. $\mathbf{P}^C$ is then discarded by the filter if its height is higher then the value of its threshold (like $\mathbf{P}_2$ but not $\mathbf{P}_1$ and $\mathbf{P}_3$). After applying this filter, we obtain a point-cloud composed mostly by ground-plane points and some outliers.



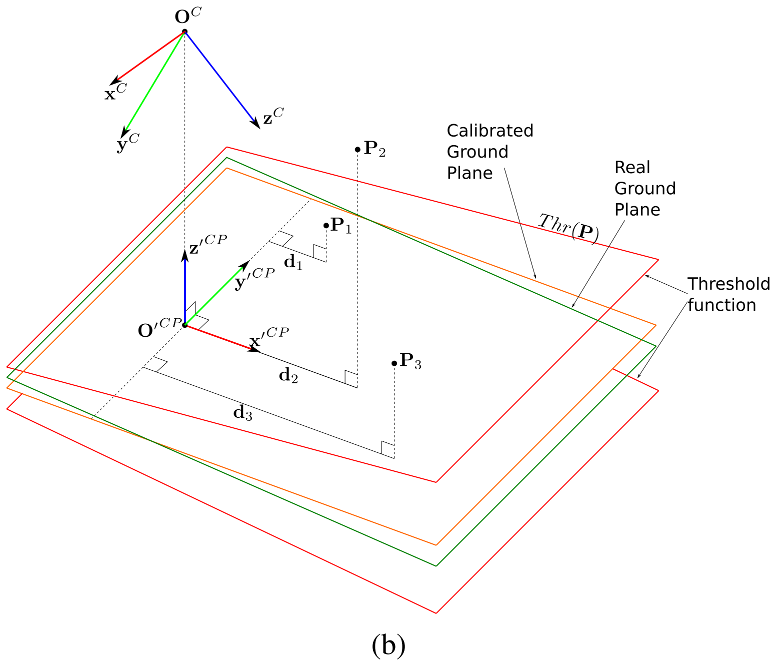


Fig. 4 Dynamic threshold filter definition. **a** Perspective view, **b** 3D view

To filter these outliers and to estimate the real ground model parameters, (a^C, b^C, c^C, d^C), a RANSAC algorithm is used with the model characterized in (1). The result incorporates the ground point-cloud (points considered as inliers). Figure 1d shows the resulting point-cloud after the dynamic threshold filter and the resulting point-cloud inliers after the RANSAC algorithm.

Using the methodology already explained in Section 2.1, we then estimate the true camera-projection $\{CP\}$ frame basis, the transformation between $\{C\}$ and $\{CP\}$, and more importantly, the 2D ground point-cloud.

3 Boundary Edges Estimation

3.1 Estimation Algorithm

After the ground detection, a boundary edges estimator was designed using the resulting 2D ground point-cloud obtained. This estimation is based on the concave hull algorithm, also known as α-shape [8]. Given a set of spacial points, it estimates the polygon with the minimum surface that best describes the enclosed shape of the points. Applying the concave hull algorithm to the 2D ground point-cloud, we obtain the list of edges forming the shape polygon.

Following the example of Fig. 1d, the output of the algorithm is shown in Fig. 5, where the black points are the entire point set, the red Xs represent the points belonging to the polygon shape, which are highlighted in green/grey.

However, the resulting edges of the ground point estimated with the use of the Concave Hull algorithm exhibits two challenges: firstly, part of the edges are caused by the intersection of the FOV limits with the ground plane and, in addition, some edges can be caused by shadows of objects in the line of sight of the camera. One can see these two challenges in the Figure 6 where in red we have the real edges, in blue/black the outliers caused by the FOV limits and in green/dashed the outliers caused by the shadows.

To solve these challenges, which is to say, to reject the edges that are not originated from the environment but from the sensor characteristics, we constructed an outliers filter algorithm composed by two steps. The first step of the filter algorithm copes with the outliers caused by the FOV limits. The second step deals with the outliers caused by the object shadows.

3.2 Outliers Filtering Algorithm

In order to obtain the first step of this filter, the FOV model of the camera is estimated and posteriorly its intersection with the ground-plane is computed, evaluating the location of its limits on the ground plane. For the estimation of the FOV model of the depth-camera, the pinhole model was used and consequently the FOV was modeled as a four planes system, all

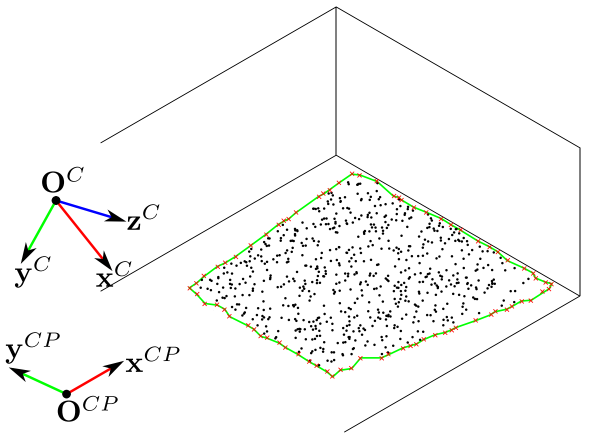


Fig. 5 Example the α-shape output result with red Xs being the edges, *black points* the point set and *green lines* the polygon

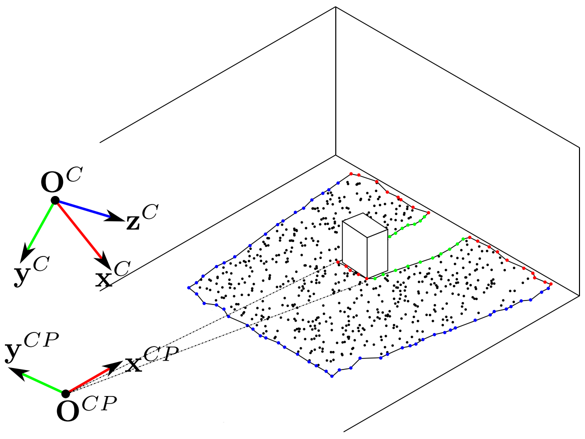


Fig. 6 Example of the edges resulted from the α-shape algorithm differentiating the type of edges: in *red/grey* the environment edges, in *blue/black* the FOV limits edges and in *green/dashed* the shadows edges

intersecting the focal point of the camera and forming angles equivalent to the camera viewing angles. These planes are estimated by the 3D positions of the left-hand and right-hand upper and lower corners of the camera image. Given these points and that all four planes intersect the origin of the camera frame, the model estimation comes through the definition of the plane with three points. The normal for i^{th} FOV plane, $\vec{\mathbf{n_i}} = \left[\, a_i{}^C \;\; b_i{}^C \;\; c_i{}^C \, \right]$ is then:

$$\vec{\mathbf{n_i}} = (\mathbf{C_i}{}^C - \mathbf{O}^C) \times (\mathbf{C_{i+1}}{}^C - \mathbf{O}^C) \,, \tag{15}$$

where $\mathbf{C_i}{}^C$ and $\mathbf{C_{i+1}}{}^C$ are the 3D points of the image corners and $\mathbf{O}^C$ the camera origin. Since $d_i{}^C$ represents the distance from the plane to the origin, it is equal to zero (all planes intersect the origin). The four FOV planes are therefore represented with the equations in the camera frame $\{C\}$ by:

$$a_i{}^C x_{\mathbf{F_i}^C} + b_i{}^C y_{\mathbf{F_i}^C} + c_i{}^C z_{\mathbf{F_i}^C} = 0 \qquad i = 1, \ldots, 4 \,, \tag{16}$$

where $\mathbf{F_i}{}^C$ is an arbitrary point belonging to the i^{th} FOV plane defined in the camera frame.

The lines of the intersection of the FOV planes with the ground-plane are easier to compute in 2D space, so a transformation of the FOV parameters space is first performed. By using the transformation equations from $\{C\}$ to $\{CP\}$, we transform the parameters in Eq. 16 into the $\{CP\}$ frame to subsequently estimate the parameters of the lines in $\{\mathbf{x}^{CP}, \mathbf{y}^{CP}\}$.

The new normal vectors of the FOV planes in $\{CP\}$, $\left[{a_i}^{CP} \; {b_i}^{CP} \; {c_i}^{CP} \right]^T$ are computed by the rotation ${}^{CP}_{C}\mathbf{R}$ estimated in (9), as following:

$$\begin{bmatrix} {a_i}^{CP} \\ {b_i}^{CP} \\ {c_i}^{CP} \end{bmatrix} = {}^{CP}_{C}\mathbf{R} \cdot \begin{bmatrix} {a_i}^{C} \\ {b_i}^{C} \\ {c_i}^{C} \end{bmatrix}, \qquad i = 1, \ldots, 4. \quad (17)$$

Next, the distance to the camera-projection origin, i.e. the parameters ${d_i}^{CP}$, is estimated by substituting the coordinates of a known point belonging to the plane in the plane model equation. A point that is shared by all planes is the camera origin, $\mathbf{O}^C$. The camera origin in $\{CP\}$, denoted as $\mathbf{O}_C^{CP}$, is equal, by the geometry of the problem, to the translation ${}^{CP}_{C}\mathbf{t}$ estimated in (10). And therefore, ${d_i}^{CP}$ is:

$$\begin{cases} \mathbf{O}_C^{CP} = \left[0 \; 0 \; |d^C| \right]^T \\ {a_i}^{CP} x_{\mathbf{O}_C^{CP}} + {b_i}^{CP} y_{\mathbf{O}_C^{CP}} + {c_i}^{CP} z_{\mathbf{O}_C^{CP}} + {d_i}^{CP} = 0 \end{cases} \quad (18)$$

$$\Rightarrow {d_i}^{CP} = -{c_i}^{CP} \cdot |d^C|, \quad (19)$$

resulting in the complete FOV planes equation defined in the camera-projection frame $\{CP\}$:

$${a_i}^{CP} x_{\mathbf{F_i}^{CP}} + {b_i}^{CP} y_{\mathbf{F_i}^{CP}} + {c_i}^{CP} z_{\mathbf{F_i}^{CP}} + {d_i}^{CP} = 0, \quad (20)$$

where $\mathbf{F_i}^{CP}$ is an arbitrary point belonging to the i^{th} FOV plane in the frame $\{CP\}$.

The intersection is mathematically performed by replacing in the FOV Eq. (20) the ground-plane equation ($z_{\mathbf{G}^{CP}} = 0$) and so we have:

$${a_i}^{CP} x_{\mathbf{I}^{CP}} + {b_i}^{CP} y_{\mathbf{I}^{CP}} + {d_i}^{CP} = 0, \quad (21)$$

where $\mathbf{I}^{CP}$ is an arbitrary point belonging to the i^{th} intersection line on the 2D frame and $({a_i}^{CP}, {b_i}^{CP}, {d_i}^{CP})$ are the respective i^{th} line parameters.

In the first step of the outliers filter, the previous estimated lines are used to remove the FOV limits outliers. So if the distance from the edge points detected to one of these four lines is lower than a fixed threshold, they are discarded. Following the examples before, a result of the filter is shown in Fig. 7 with the FOV intersection lines as indicated.

Since depth cameras compute the depth of the objects in the scene by projecting an IR light, in the estimated ground point-cloud it is possible to find some shadows caused by the obstruction of such light by random obstacles in the scene. These shadows will cause a number of detected edges in the ground point edges estimated algorithm that are not caused by environment features but by the edges of the shadows. One can see these outlier edges in green in the Fig. 6. However, because of the geometric characteristics of the problem, the outliers raised from shadows form a constant angle with the origin (polar angle), i.e they are aligned with each other and with the origin, as seen in Fig. 8a.

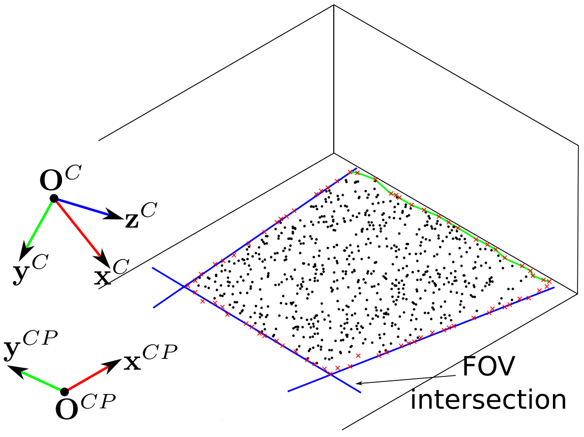


Fig. 7 Example of the output after the first step of the outlier filter execution

In the second step of the outliers filter, the polar angle of the remaining edges points is computed from their Euclidean coordinates $\left[x_{\mathbf{G}^{CP}} \; y_{\mathbf{G}^{CP}} \right]$, as follows:

$$\varphi_{\mathbf{G}^{CP}} = \operatorname{atan2}\left(y_{\mathbf{G}^{CP}}, x_{\mathbf{G}^{CP}} \right), \quad (22)$$

where $\varphi_{\mathbf{G}^{CP}}$ is the polar angle of the point $\mathbf{G}^{CP}$. After the edges are ordered according with their polar angle, this is, in the resulting list of edges after they have been sorted, the points that have smaller polar angle appear first and the points with bigger polar angular appear in the end. One can see in the Fig. 8b the resulting polar angle estimation, where the edge points are ordered by polar angle. After, to filter out the outlier edges, one simply discards the group of points that present a constant polar angle value from the list of ground edges detected. In Fig. 8b one can see the groups of points filtered out delimited by the blue circles.

4 Robot Localization System

The localization system problem for a vehicle operating in a known environment is addressed resorting to

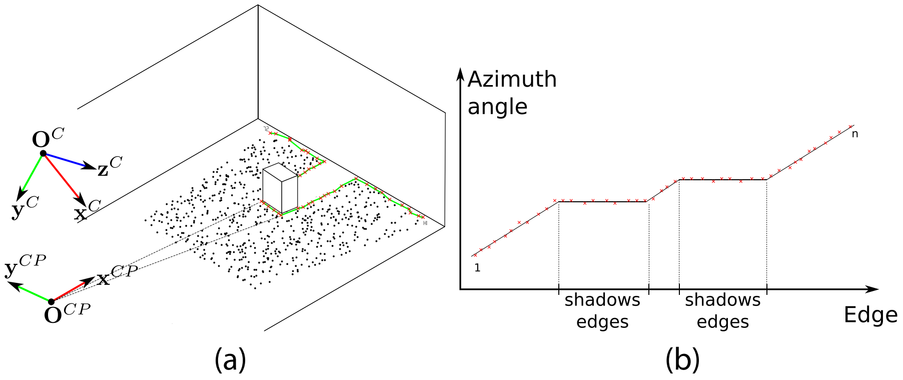


Fig. 8 Example of **a** the occlusion "shadow" edges and **b** its azimuth

a particle filter algorithm [11]. The available sensors are the differential drive encoders and the RGB-Depth camera, which provide, respectively, the odometry readings and the point-cloud, from which the ground boundary is extracted. This way, in the predict step, the particle set is altered according with the odometry readings and with the robot-specific noise model, as in standard particle filters. Next, the particle weight estimation is based on the particle cost defined as:

$$w_p = \exp(-k\frac{c_p}{\max(c_p)+\epsilon}), \tag{23}$$

where c_p is the cost of particle p, w_p is the weight of p and ϵ is a small number for numerical stability. The bigger the weight, the better the particle hypothesis is. Therefore, the probability of a particle being sampled is higher the higher its weight is, being the particle cost inversely proportional to it.

At the update step, the system obtains the point-cloud from Kinect, estimates the ground point-cloud, performs the edges detection and computes each particle's cost. For the estimation of the particle cost an occupancy grid map matrix [18] of the environment and a new, with similar size and resolution, matrix called distance-to-obstacles $\mathscr{D}$, where each cell contains the distance from that respective cell to the nearest occupied cell, are used. $\mathscr{D}$ is obtained by using the Euclidean Distance Transform algorithm [4] over the occupancy grid map matrix obtained by a mapping algorithm. In our case, the source points of the close curve are the obstacles, i.e. the closed curve propagates in the free areas. As a result, a matrix is obtained where each cell $\mathscr{D}_i$ indicates the distance between the i^{th} cell and the nearest occupied cell. A $\mathscr{D}$ close up is shown in (9) with a color range from red, for bigger distances, to blue, for smaller distances.

Consider a particle p at pose $\left[\, x_p^W \;\; y_p^W \;\; \theta_p^W \,\right]^T$ and the set of points of the ground edges polygon $\mathscr{L}^{CP}$, in the camera projection frame. The cost estimation of particle p starts by transforming the set $\mathscr{L}^{CP}$ into the world frame using the pose of the particle p,

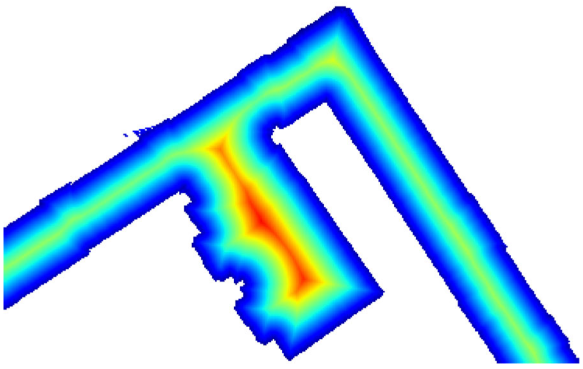


Fig. 9 Example of a distance-to-obstacles grid map

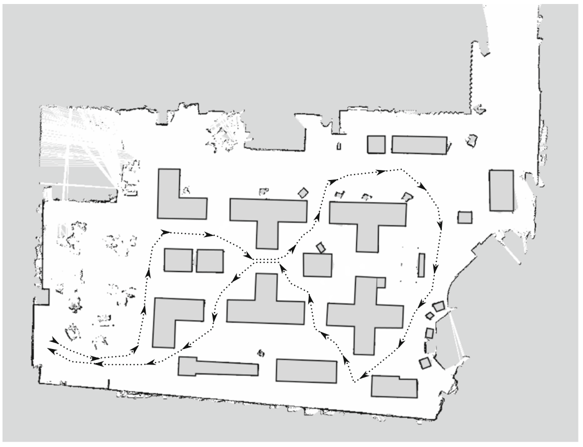


Fig. 10 Trajectory performed in the data-set

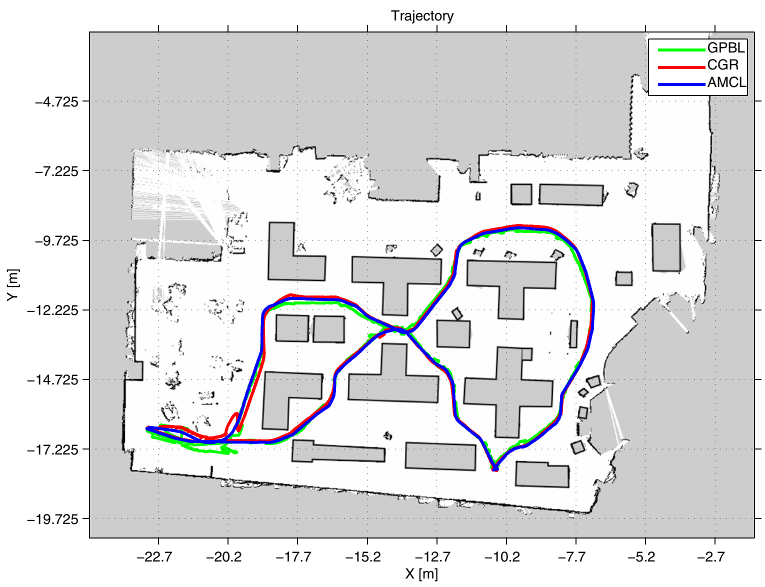


Fig. 11 Estimated path by the AMCL, CGR and GPBL algorithms

$\left[\, x_p^W \;\; y_p^W \;\; \theta_p^W \,\right]^T$, the pose of the camera in robot and the transformation from $\{C\}$ to $\{CP\}$, obtaining the list $\mathscr{L}_p^W$. $\mathscr{L}_p^W$ is therefore the observed list of edges in the world frame if the robot were in the particle p (hypothesis). The cost function is therefore meant to reflect the match between the robot observation and the predictable observation of p. The cost of p is defined as the L^1-Norm of all the distances between the ground edges $\mathscr{L}_p^W$ and the nearest obstacles, i.e. the L1-Norm of D cells that contain the point edeges

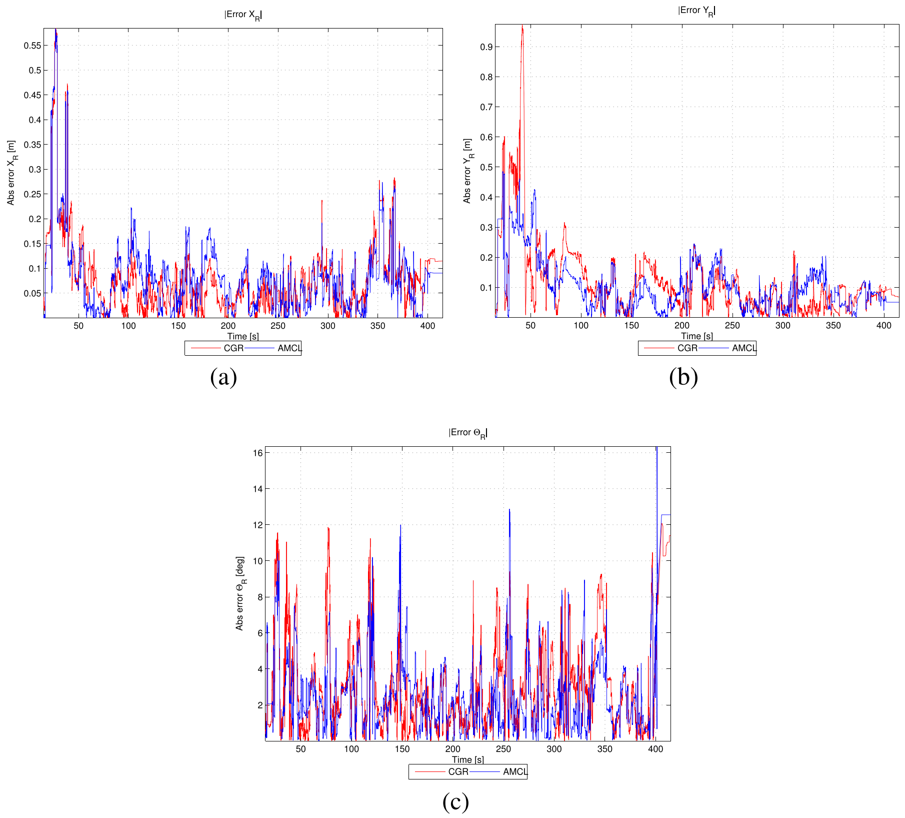


Fig. 12 Differences between the pose coordinates by the AMCL,CGR and GPBL algorithms

Table 1 Means and standard deviations of the localization differences between algorithms

	$Alg.$	
	CGR	$AMCL$
$\lvert X_R[GPBL] - X_R[Alg.]\rvert$ [m]	0.077	0.082
$\sigma(\lvert X_R[GPBL] - X_R[Alg.]\rvert)$ [m]	0.078	0.079
$\lvert Y_R[GPBL] - Y_R[Alg.]\rvert$ [m]	0.118	0.105
$\sigma(\lvert Y_R[GPBL] - Y_R[Alg.]\rvert)$ [m]	0.128	0.086
$\lvert \Theta_R[GPBL] - \Theta_R[Alg.]\rvert$ [deg]	3.04	2.69
$\sigma(\lvert \Theta_R[GPBL] - \Theta_R[Alg.]\rvert)$ [deg]	2.60	2.53

detected. We can describe mathematically the cost function with the following equation:

$$c_p = \sum_{i=1}^{n} \mathscr{D}(\mathscr{L}_p^W(i)) , \tag{24}$$

where n is the number of points in $\mathscr{L}_p^W$, $\mathscr{L}_p^W(i)$ the i^{th} point list and $\mathscr{D}(\mathscr{L}_p^W(i))$ the distance from the point $\mathscr{L}_p^W(i)$ to the nearest occupied pixel. The idea of this methodology is that the smaller the sum of the distances, the better the correspondence of the map with the edges detected at the respective hypothesis (particle) is. The cost of the particle is then converted to its weight. The conversion is done using the exponential function (23), i.e. the smaller the particle cost is, the better is the particle hypothesis. After, the resampling is performed and the particle set is updated accordingly as in standard particle filter.

5 Results

The real-time experimental results presented in this work where obtained in a semi-structured environment, in a bookshop during its normal opening hours.

The sensor data collected is composed by the depth image provided by the Kinect® camera, the odometry readings provided by the wheels encoders and a laser range finder (LRF) scan provided by an Hokuyo® UTM-30LX. The first localization algorithm used for comparison took as input odometry and laser data only, the other took, like GPBL, odometry and depth image data. The camera depth and the odometry encoders stream outputs at approximately 30Hz and the LRF streams outputs at approximately 40Hz. It is worth remembering that this frequencies are not the ones at which the algorithms run, but those at which they receive the data.

The experiment consists of a robot moving around the bookshop at an average speed of 0.4 m/s, no counting the moments the robot stayed still to let bookstore customers pass. The general manoeuvre is approximately a 8-shaped path, like one can see in Fig. 10. The total duration of the data-set is 7 min and 26 s (446 s). More experiments, data sets and their results can be found online.[1] Due to the impracticality of setting a reliable ground truth system such as Vicon® or similar, to analyze our algorithm performance, the poses assessed by our algorithm were compared directly with the ones assessed by the other two localization algorithms. The algorithms chosen were respectively the AMCL [12] since it is the standard localization system of the ROS platform and the CGR [2] since is the one used in the CMU-Cobot [19]. This makes it possible for us to validate the effectiveness of our method, since in the two methods used for the comparison, one employs a different algorithm but with the same type of sensor (Depth camera), while the other adopts both a different algorithm and a different type of sensor (LRF), thus bringing diversity to the comparison.

Figure 11 illustrates the robots estimated trajectory in a 2D map for the CGR, the AMCL and the GPBL. It is possible to visualize that during the experience, even in this crowded environment, the pose estimation gives a correct result, since the three algorithms present roughly the same outcome. When observing the 2D map it is important to consider the following details: (1) red/medium grey line - Trajectory of the robot by the CGR algorithm; (2) green/light grey line - Trajectory of the robot by our algorithm

[1] http://users.isr.ist.utl.pt/~mvaz/thesis

(GPBL); (3) blue/dark grey line - Trajectory of the robot by the AMCL algorithm. In Fig. 12 one can see the absolute value of the differences between our algorithm and other two used in the comparison along the three degrees-of-freedom.

The observed uncertainty (particles standard deviation) of our algorithm maintained a stable, low level, during the experience. The averages and standard deviations divergence with the other algorithms are presented in Table 1. The average processing time per iteration of the GPBL was around 0.17 s with a standard deviation of 0.13 s, using a Intel Core i5-2520M @ 2.5 GHz processor.

Note that while the other two algorithms used the books on the shelves for localization, ours on the other hand did not consider them. This is, the CGR algortihm uses the books in the shell as vertical features and the AMCL uses the books from the laser readings. We did not remove them for practical reasons.

6 Conclusions

In this paper we described a localization system based on a depth camera using a novel observation model. The proposed system was implemented based on the sensor perception of the ground, having shown that this method is robust and generally applicable in semi-structured environments. An evaluation in a real-life scenario with a real robot was performed, as well as a comparison with the state of the art. The algorithm estimated the position of the robot in a satisfactory way, corroborating our goal. In the future we intend to challenge our system with environments that other systems are unable to cope with, such as in the absence of vertical walls. This work was supported by the FCT project [PEst-OE/EEI/LA0009/2013].

References

1. Arulampalam et al.: A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Trans Signal Process **50**(2), 174–188 (2002)
2. Biswas, J., Veloso, M.: Depth camera based indoor mobile robot localization and navigation. In: Proc. IEEE Int. Conf. Robotics and Automation, pp. 1697–1702. IEEE Press, Saint Paul (2012)
3. Borrmann et al.: Globally consistent 3D mapping with scan matching. Robot. Auton. Syst. **56**(2), 130–142 (2008)
4. Breu, H., Gil, J., Kirkpatrick, D., Werman, M.: Linear time euclidean distance transform algorithms. IEEE Trans. Pattern. Anal. Mach. Intell. **17**(5), 529–533 (1995)
5. Castellanos et al.: Robocentric map joining: Improving the consistency of EKF-SLAM. Robot. Auton. Syst. **55**(1), 21–29 (2007)
6. Cunha et al.: Using a depth camera for indoor robot localization and navigation. In: Proc. Robotics Sci. and Syst. Conf., Los Angeles (2011)
7. Doucet, A., Johansen, A.M.: A tutorial on particle filtering and smoothing: Fifteen years later. In: The Oxford Handbook of Nonlinear Filtering, pp. 656–704. Oxford University Press (2011)
8. Edelsbrunner et al.: On the shape of a set of points in the plane. IEEE Trans. Inf. Theory **29**(4), 551–559 (2006)
9. Einicke, G., White, L.: Robust extended kalman filtering. IEEE Trans. Signal Process **47**(9), 2596–2599 (1999)
10. 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**(6), 381–395 (1981)
11. Fox et al.: Particle filters for mobile robot localization. In: Doucet, A., de Freitas, N., Gordon, N. (eds.) Sequential Monte Carlo Methods in Practice, pp. 499–516. Springer, New York (2001)
12. Fox, D.: Adapting the sample size in particle filters through KLD-sampling. Int J Robot. Res. **22**(12), 985–1003 (2003)
13. Fox, D., Gutmann, J.S.: An experimental comparison of localization methods continued. In: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst., Lausanne, pp. 454–459, Switzerland (2002)
14. Gower, J.C., Dijksterhuis, G.B.: Procrustes Problems, Oxford statistical science series, vol 30. Oxford University Press, Oxford (2004)
15. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments. Int. J. Robot. Res. **31**(5), 647–663 (2012)
16. Neira, J., Tardós, J.D.: Data association in stochastic mapping using the joint compatibility test. IEEE Trans. Robot. Autom. **17**(6), 890–897 (2001)
17. Rao, C.R., Toutenburg, H.: Linear Models: Least Squares and Alternatives. Springer series in statistics. Springer, New York (2008)
18. Thrun, S., Bücken, A.: Integrating grid-based and topological maps for mobile robot navigation. In: Proc. AAAI Nat. Conf. Artificial Intell., pp. 944–950. AAAI Press, Portland (1996)
19. Veloso et al. Deploying robots in an office environment: Web-based task requests by users and autonomous robot navigation. Submitted for publication (2011)
20. Ventura, R.: New Trends on Medical and Service Robots: Challenges and Solutions, Springer, chap Two Faces of Human-robot Interaction: Field and Service robots. MMS, (in press) (2014)
21. Welch, G., Bishop, G. University of North Carolina, Chapel Hill (2006)
22. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1330–1334 (2000)