Toward Rich Geometric Map for SLAM: Online Detection of Planes in 2D LIDAR

Received 10th October 2012; accepted 22nd November 2012.

Cyrille Berger

Abstract:
Rich geometric models of the environment are needed for robots to carry out their missions. However a robot operating in a large environment would require a compact representation. In this article, we present a method that relies on the idea that a plane appears as a line segment in a 2D scan, and that by tracking those lines frame after frame, it is possible to estimate the parameters of that plane. The method is divided in three steps: fitting line segments on the points of the 2D scan, tracking those line segments in consecutive scan and estimating the parameters with a graph based SLAM (Simultaneous Localisation And Mapping) algorithm.

Keywords: *laser scanner, SLAM, LIDAR*

1. Introduction

1.1. Environment Modelling

Accurate and rich environment models are essential for a robot to carry out its mission. However those models are not always available to the robot, and if they are available (through geographic information database) their information might be inaccurate and outdated, for instance, due to the consequences of a catastrophe, where part of the environment has been destroyed, but it also happen continuously, for instance, when new building are constructed or demolished... This raise the need for a robot to be able to construct environment model using its own sensor, and at the same time, this model can be used to improve the knowledge of the robot position. This is the problem known as Simultaneous Localisation and Mapping (SLAM) [1, 5].

However most research around SLAM have focused around providing improvement to the robot localisation, by tracking features in the environment. For that purpose, using a sparse model of the environment (such as interest points) is generally sufficient. On the other hand, many techniques used for 3D reconstruction assumes that a very accurate localisation is available (generally through GPS like systems), which is not necessary practical for a real-time robotic system and also they work offline. This is why, in our research, we are interested in using SLAM techniques to construct rich and dense model of the environment, this have the advantage of improving the localisation of the robot and the quality of the model, as well as creating the model in real-time.

To create a model of the environment, a robot can use different sensors: the most popular ones are camera [12], LIDAR [18], RADAR [8] or multiple sensors [7] are combined. Cameras are very cost effective, they are also cheap and provide the most information, since it is possible to recover structure and appearance, however they are less accurate than LIDAR and RADAR, and they are not very effective at recovering the structure from a distant view point, which is especially important when used on board of an unmanned aircraft.

1.2. Using a 2D LIDAR for 3D modelling

Most approaches that use LIDAR start by generating a cloud of points, combined with a shape registration technique [19], to correct uncertainty on the localisation of the robot, and this is usually result in a very accurate model.

While a cloud of points is a dense model of the environment, the amount of information it contains make it difficult to use in practice. It needs to be transformed into a different representation, that will abstract the model, for instance, points cloud are commonly used to create grid occupancy [17], used for collision detection.

Also a cloud of points, in itself, contains very little information on the actual structure of the environment. One of the solution used to get the structure has been to combine the information coming from the cloud of points with images [13]. An other solution is to transform the cloud of points into higher level of geometric information. Classification process like the hough transform [9, 16] can be used to achieve that goal, making it possible to process the LIDAR data in high level geometric features such as cube, cylinder or sphere [18].

Not only the use of higher level of geometric information allow to create more compact model, which reduce the amount of memory and computing power necessary to use them, but they allow other application such as object recognition, matching information coming from different sensors and with existing schematics [2].

However the use of a cloud of points to generate a high level geometric representation requires to wait until the robot has finished to acquire the data, before the robot can construct the model. But we are interested in an online and incremental approach.

As shown in [6] and figure 1, in a 2D scan of a LIDAR sensor, a plane appears as a line segment, which makes it possible to extract planes in the environment directly from those scans, by simply tracking the line segment, scan after scan, the aggregation of those scans allow to recover the parameters of the plane. For such a system to work, it is essential that the plane of the 2D scans are not parallel to the displacement of the robot, otherwise, the rays of the LIDAR will always hit the plane at the same place, which would not provide any information on the 3D structure.

In this article, we present a method for extracting line segment from the 2D scan, tracking them between two

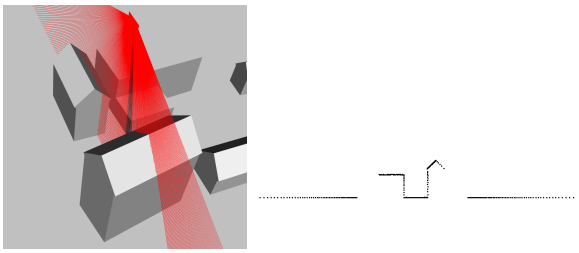


Fig. 1. The left picture show the 3D view of a scan of a two buildings, while the right figure shows how it display in 2D.

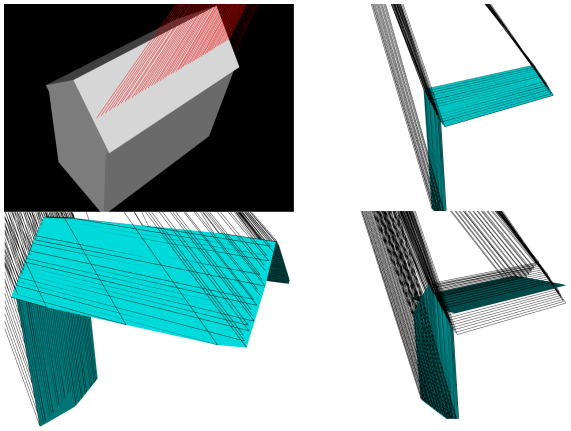


Fig. 2. This figure shows an unlikely example where the assumption that the tracked segment belongs to the same plane is wrong. The robot is flying exactly perpendicularly to the house, meaning that the observation on the two side of the roof appears parallel. The figure on the top left shows the model of the house (the red lines are scan rays), on the top right the robot has scanned only half of the house, and the resulting plane is still correct, on the bottom right, the robot has finished flying over the house (the black line illustrates the constraints). The bottom left image shows that if the house has a 2° angle with the sensor then the roof is correctly detected as two planes.

frames, and then the line segment is treated as an observation of the plane within the graphical SLAM framework, using a weak constraint network optimiser (WCNO) [3]. Our main contributions are the error and observation model used for WCNO as well as the overall combination of algorithms to extract planes from 2D scan.

The first section describes the line extraction which is a classical split-and-merge [10] and the line tracking. The second section explained the error and observation models used in WCNO. The last section provides simulation and real data results.

2. Detecting planes in the LIDAR data

As mentioned in the introduction, when a 2D LIDAR scan hit a plane, the points from the plane will appear as a line segment in the 2D scan. Our assumption is that whenever there is a line segment in the 2D scan, it does belong to a plane, especially, if it is tracked between two

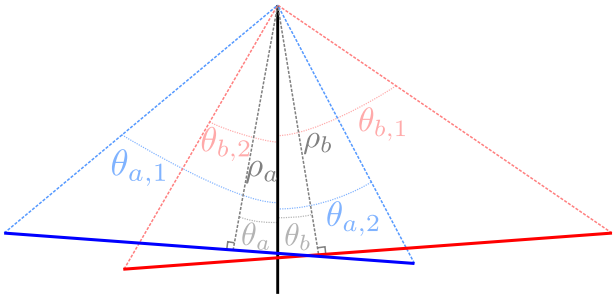


Fig. 3. This figure shows the angle used for comparing two segments during tracking. ρ_x and θ_x are the parameters of a line while $\theta_{x,1}$ and $\theta_{x,2}$ are the angles of the extremities of the segment.

consecutive frames. However, as shown in Figure 2, there are corner cases where this assumption that two consecutive segments belong to the same plane does not hold. This case happens if the robot is precisely on a trajectory that is perpendicular to the intersection line between two planes or perpendicular to a cylinder, however, we argue that those cases are unlikely to happen in real life, since it would require that the robot is precisely on that trajectory, as a slight movement off that trajectory and the algorithm will detect a break in the line segment, as shown on the bottom left image of Figure 2. Furthermore, using the graph optimisation technique it is possible to detect those corner cases, since they will fail to optimise.

The algorithm to detect planes works in two steps, first the scans of the LIDAR data are converted into line segments and then those line segments are tracked with the previous plane. Those line segment are then used as observation of a plane.

2.1. Extracting line segments

The best method to transform a set of points from a LIDAR scan is the split-and-merge algorithm [10]. Then the parameters of each line segment are optimised with a weighted line fitting algorithms [11] to provide a better fit on the points.

2.2. Tracking line segments

Segments are tracked between two consecutive frames. By comparing the distance to the origin (ρ), and the angle of the line (θ) and checking whether the segments are overlapping, see Figure 3.

Given $\mathcal{S}_t = s_0^t ... s_m^t$ the set of segments at time t and $\mathcal{S}_{t-1} = s_0^{t-1} ... s_n^{t-1}$ at the previous time $t-1$. The goal of the tracking algorithm is to find (k, l) such as $s_k^t \in \mathcal{S}_t$ is collinear and overlapping with $s_l^{t-1} \in \mathcal{S}_{t-1}$.

Given two line segments a and b of parameters (ρ_a, θ_a) and (ρ_b, θ_b), they are considered collinear if and only if:

$$\cos(\theta_a - \theta_b) > \cos(\theta_{\max}) \quad (1)$$

$$|\rho_a - \rho_b| < \rho_{\max} \quad (2)$$

Furthermore, if two collinear line segments a and b have the extremities $(\theta_{a,1}, \theta_{a,2})$ and $(\theta_{b,1}, \theta_{b,2})$, then the following conditions guarantee that they are overlapping:

$$\theta_{a,1} \leq \theta_{b,1} \leq \theta_{a,2} \text{ or } \theta_{a,1} \leq \theta_{b,2} \leq \theta_{a,2}$$
$$\text{or } \theta_{b,1} \leq \theta_{a,1} \leq \theta_{b,2} \text{ or } \theta_{b,1} \leq \theta_{a,2} \leq \theta_{b,2} \quad (3)$$

Given a segment $s_k^t \in \mathcal{S}_t$, $\mathcal{S}_{t-1}(s_k^t)$ is the subset of segments of $\mathcal{S}_{t-1}$ that satisfy the collinearity conditions (equations 1 and 2) and the overlapping conditions (equation 3). If $\mathcal{S}_{t-1}(s_k^t) = \emptyset$, then s_k^t correspond to a new plane, otherwise, it is necessary to select a segment $s_l^{t-1} \in \mathcal{S}_{t-1}$ that will be the tracked segment for s_k^t, and that will be used as the new observation of a plane. s_l^{t-1} is defined such as:

$$\forall s_i^{t-1} \in \mathcal{S}_{t-1}(s_k^t) \; |\rho_k^t - \rho_l^{t-1}| < |\rho_k^t - \rho_i^{t-1}| < \rho_{\max} \quad (4)$$

Using a prediction of the parameters of the segment, based on the motion of the robot (using the SLAM algorithm) or using the evolution of the parameters of the tracked segment, improve the results of the tracker, it also allow to limit the problem where the assumption that a line is a plane is wrong (see Figure 2).

2.3. Detecting and tracking planes with a 2D LIDAR sensor

The algorithm for detecting and tracking planes in consecutive scan from a 2D LIDAR sensor follow the following steps:

1) At time t, extract the segments $\mathcal{S}_t = s_0^t ... s_m^t$ in the scan,
2) for each segment $s_k^t \in \mathcal{S}_t$, find the segment $s_l^{t-1} \in \mathcal{S}_{t-1}(s_k^t)$ that fulfil the equation 4
   - if there is no such segment s_l^{t-1}, then s_k^t is the beginning of a new plane,
   - if there is a segment s_l^{t-1}, then s_l^{t-1} is used as a new observation of the plane associated with s_k^t,
3) for each segment $s_l^{t-1} \in \mathcal{S}_{t-1}$, which has not been selected as a tracked segment for any $s_k^t \in \mathcal{S}_t$, check if the associated plane has enough observations, otherwise, remove it from the map.

The next section is going to explain how the parameters of a plane are estimated.

3. Optimisation model

The estimation of the parameters of the plane and the position of the robot is done using a weak constraints network optimisation (WCNO) as described in [3]. This algorithm works by optimising a graph of constraints, where the nodes are objects (robot poses and landmarks) and the edges are the observations (coming from various sensors: odometry, LIDAR, …). The key features of WCNO is that it allows to use partial observations, which is interesting for estimating the plane parameters as observed with a 2D LIDAR, since in such a case, only a few parameters of the plane are observed at each time. WCNO relies on the steepest gradient descent and it requires the definition of an object model and of a constraint model.

Three functions are needed for the object model:

- *initialisation* this function uses one or several constraints to compute a first estimation of the parameters of an object, as well as the associated uncertainty, in case of a plane, a plane perpendicular to the first observation is used,
- *rotation update* this function rotates the object on itself,

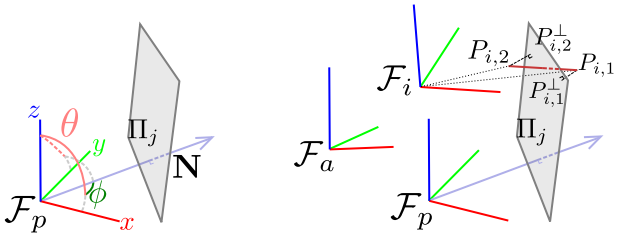


Fig. 4. The left figure show the feature model for a plane, represented using spherical coordinates (ρ, θ, ϕ) expressed in the frame $\mathcal{F}_p$. The right figure represent the observation and feature model for a plane, the observation in frame $\mathcal{F}_j$ is the dark red segment $P_{i,1}P_{i,2}$, while $P_{i,1}^{\perp}$ and $P_{i,2}^{\perp}$ are respectively the projection of $P_{i,1}$ and $P_{i,2}$ on the plane Π_j. $\mathcal{F}_a$ is the common ancestor in the tree of $\mathcal{F}_j$ and $\mathcal{F}_p$.

- *translation update* this function translate the object.

Also for the constraint between the frame and the plane, three other functions are required:

- *rotation error* $\hat{r}_{i,j}^c$ – this function computes the rotation error between the frame and the plane,
- *translation error* $\hat{t}_{i,j}^c$ – this function computes the translation error between the frame and the plane,
- *constraint observation model* $h(\mathcal{O}_i, \mathcal{O}_j)$ – it is the function that given the state of a frame and a plane, computes the segment that should be observed in the frame.

The work [3] includes more details on how to define the functions for the node and constraint models, it also includes the node use for robot frames and the constraint model used for odometry. The rest of this section will detail the object model for planes as well as the constraint model used with the plane detection algorithm provided in the section 2.

Figure 4 show the convention used for the spherical coordinates (ρ, θ, ϕ), where ρ is the distance of the plane to the origin, θ is the rotation of the normal around the axis Oy and ϕ around Oz.

We assume the scanner is in the plane Oxy, if that is not the case, one can simply add a frame for the sensor expressed in the robot frame with the transformation between the robot and the frame in the graph.

3.1. Feature model

The plane is represented using spherical coordinates, with the parameters ρ, θ and ϕ (Figure 4). Where ρ, θ and ϕ are the coordinates of a point of the plane and θ and ϕ are the angles of the normal $\mathbf{N}$ to the plane. A polygon representing the boundary of the plane is associated to the model, this boundary has little use for the purpose of estimating the parameter of the plane, but can be used for display purposes or in the matching process [2].

Rotation Update The rotation update is simply given by applying the rotation on the normal vector:

$$\mathbf{N_n} = R_{up} \cdot \mathbf{N_{n-1}} \quad (5)$$

ρ remains constant. N_n is the normal of the plane and R_{up} is the rotation update.

Translation Update The translation of an infinite plane only applies a change in the direction parallel to the normal, with a spherical representation, only the distance ρ_n to the origin is affected:

$$\rho_n = \rho_{n-1} + \langle \mathbf{N_n}, \mathbf{t_{n-1}} \rangle \tag{6}$$

3.2. Constraint model

Given a frame $\mathcal{F}_i$ and a plane Π_j, they are connected in the graph by a constraint which is the line segment that was detected and tracked in section 2.

The constraint between $\mathcal{F}_i$ and Π_j is represented by the extremities of the observed segment: $P_{i,1}$ and $P_{i,2}$, $P^{\perp}_{i,1}$ and $P^{\perp}_{i,2}$ are respectively the projection of $P_{i,1}$ and $P_{i,2}$ on the plane Π_j (see Figure 4).

Rotation error The rotation error is the minimal rotation that ensure the observation becomes parallel to the plane. This rotation is the rotation between the vector $\mathbf{P_{i,1}P_{i,2}}$ and $\mathbf{P^{\perp}_{i,1}P^{\perp}_{i,2}}$:

$$\mathbf{u_1} = \frac{\mathbf{P_{i,1}P_{i,2}}}{\|\mathbf{P_{i,1}P_{i,2}}\|} \tag{7}$$

$$\mathbf{u_2} = \frac{\mathbf{P^{\perp}_{i,1}P^{\perp}_{i,2}}}{\|\mathbf{P^{\perp}_{i,1}P^{\perp}_{i,2}}\|} \tag{8}$$

$$\hat{r}^c_{i,j} = R_i \cdot \left(cos^{-1}\left(\langle \mathbf{u_2}, \mathbf{u_1} \rangle \right), \mathbf{u_2} \wedge \mathbf{u_1} \right) \cdot R_i^{-1} \tag{9}$$

where R_i is the rotation of the frame $\mathcal{F}_i$.

Translation update The translation update minimise the distance between the points of the scan and the plane. It is therefore defined as:

$$\hat{t}^c_{i,j} = \frac{\mathbf{P_{i,1}P^{\perp}_{i,1}} + \mathbf{P_{i,2}P^{\perp}_{i,2}}}{2} \tag{10}$$

Recover plane parameters However the steepest gradient descent algorithm as described in [3] does not allow to recover the normal of the plane correctly, since it would require to define a rotation of the plane, but that would raise the question of the choice of an axis and of where to apply this rotation. Instead, the least square optimisation is applied, combined with the "rotation update" and the "the translation update" this should ensure that the observation points get coplanar.

Observation model The observation model can be used for predicting the parameters of the line segment that needs to be tracked in section 2, and it is also used by WCNO to update the uncertainty of the plane Π_j and of the frame $\mathcal{F}_i$. It is defined as the intersection of the plane of the 2D LIDAR sensor expressed in the frame $\mathcal{F}_i$ with the plane Π_j. The following formula assumes that the 2D LIDAR sensor is in the plan Oxy of $\mathcal{F}_i$:

$$h_{i,j} = (\rho^{obs}_{i,j}, \phi^{obs}_{i,j}) = (\frac{\rho^i_j}{\sin(\theta^i_j)}, \phi^i_j) \tag{11}$$

where ρ^i_j, θ^i_j and ϕ^i_j are the coordinates of the plane Π_j expressed in the frame $\mathcal{F}_i$.

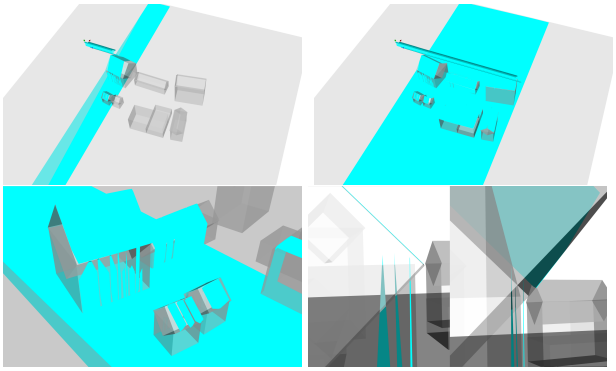


Fig. 5. This figure shows the algorithm at work on a simulated world. The transparent grey building are the ground truth, while the cyan represented the estimated planes. On the bottom left image, one can see that the detection of plane breaks on face with few hit points, and on the bottom right images, the roof of the building get a better estimation that the walls.

Boundaries of the plane The extremities of the observation are projected on the plane, and then it is possible to recover a boundary [4]. Since the boundary is only used for display, and for simplicity, in our experimentation we have been using the convex hull of the projection of the extremities.

4. Experiments

We have used the algorithm in both simulation and real data experiments. The simulation allow to check that the algorithms are able to make an accurate estimation of the parameters while the real data allow to validate the usefulness of the algorithm on a real system.

4.1. Simulation

Flying over a set of buildings In this experiment, an aircraft is flying over a set of buildings, while the LIDAR is pointing down. This is actually a very classical set-up for generating a model of the environment from an aircraft. The Figure 5 shows screenshots of the results.

For a total of 54 detected planes, the average error on ρ is 0.14 m, with a maximum of 0.77 m, while the average error on the angle of the normal is $3.88°$ with a maximum at $17.73°$.

Effect of correction from plane Since we are using a 2D LIDAR sensor, the observation of planes in the environment does not allow to recover the full 3D transformation between two frames. In fact, as shown in Figure 6, the observation of a plane provide a translation that is perpendicular to that plane, as well as a rotation whose axis is parallel to that plane. So for a plane that is perpendicular to the axis Ox, if the 2D scan is in the plane Oxy, then the observation of that plane provides a translation correction in the direction of Ox and a rotation correction around the axis Oz.

In order to illustrate the correcting effect brought by the plane, in simulation, we have made an experiment, where the robot is observing a horizontal plane and the scan laser has an angle of $45°$ with both the robot frame and the plane. For the first three positions, the odometry

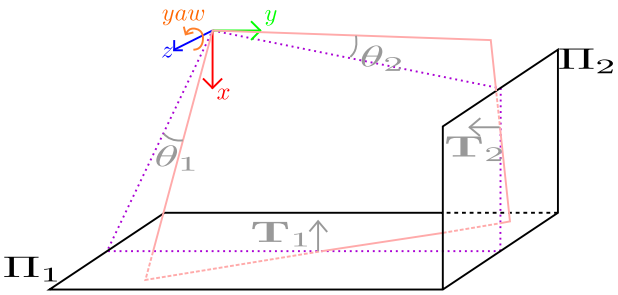


Fig. 6. This figure shows the correction effect from observing a plane with a 2D LIDAR sensor. The red lines shows the observed constraints, while the purple lines shows the ideal position of the constraint. The contribution of the planes Π_1 and Π_2 to the correction are the translations $\mathbf{T}_1$ and $\mathbf{T}_2$ and the rotations (Oz, θ_1) and (Oz, θ_2).

has a high accuracy, this allow to get a good estimate of the parameters of the plane, however, for the fourth step, one of the value coming from the odometry is defined with a high uncertainty, with either an error of 50 m in z or 22° in roll. The results are shows in Figures 7 and 8.

While the experiment in itself is unrealistic, it does show the benefit of using an iterative approach to estimate the parameters of the planes in the environment. And this is especially interesting for an aircraft, since GPS units have usually a poor estimate of the altitude, the detection of plane would allow to correct that information.

4.2. Ground robot

We have also run the algorithm using real data, like the New College data set [15], which was created using a robot, with a vertical laser on the side of the robot.

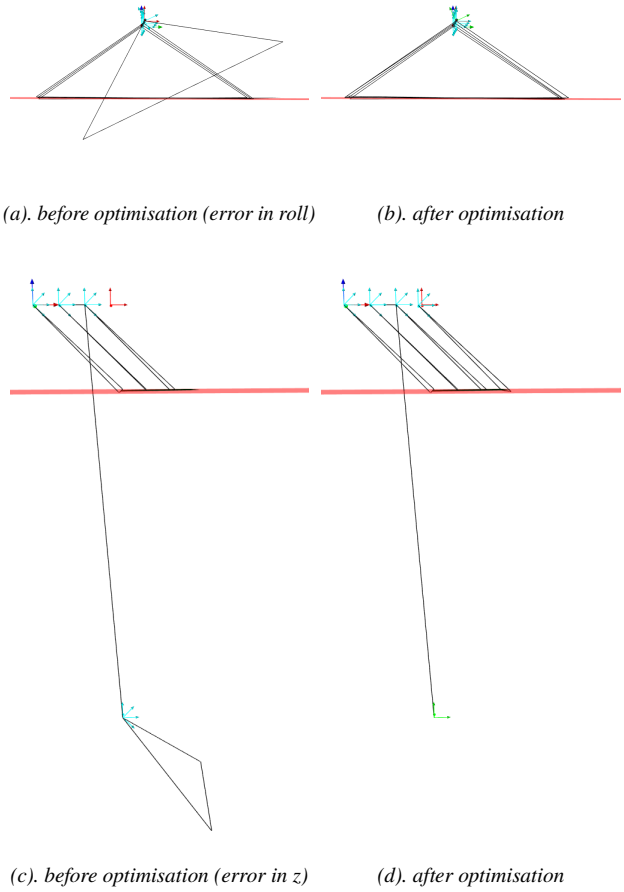
As can be seen in Figure 9, the algorithm has no problem with the detection of the ground. However, the detection of walls is often more split, this is due to the presence of vegetation in front of the building, and also because the building facade is not completely flat.

For comparison purposes, using the same sequence, a points cloud was generated, then a RANSAC technique was applied using the Points Cloud Library [14] to cluster the points in function of their coplanarity. The results are shown in Figure 10. The first point to note is that the result of the classification is unfiltered, meaning that points that do not really belong to an actual plane are still shown. Also, it is worth to note that the resulting segmentation is rather unstructured meaning that points that are classified as belonging to the same plane might actually be very far apart, this effect is particularly visible in the bottom Figure 10, where several plane are juxtaposed.

5. Conclusions and Future Works

We have presented an algorithm that extract observations of a plane using 2D scan from a LIDAR sensor, this algorithm works by fitting line segments on the points of the scan, and then tracking the segments frame after frame. Then those observations are used in a graph based framework to estimate the parameters of the plane and improve the localisation of the robot.

We have shown through simulation results, that the method is capable to accurately estimate the parameters of the planes, and we have shown that the method works with real data.



(a). before optimisation (error in roll) *(b). after optimisation*

(c). before optimisation (error in z) *(d). after optimisation*

Fig. 7. This figure show that the optimisation process is capable of recovering a large error on the estimation of the roll and z parameters. The black line represent the constraints between node. In particular, the black triangles are the observations coming from the lidar. The red polygon is the plane, the frame represents the robot and sensor position, while the cyan polygon is the estimated plane. On the first three robot pose, the estimation of the odometry is accurate, while on the fourth pose, the error has been exaggerated (in roll for the top row of images and in z for the bottom row.

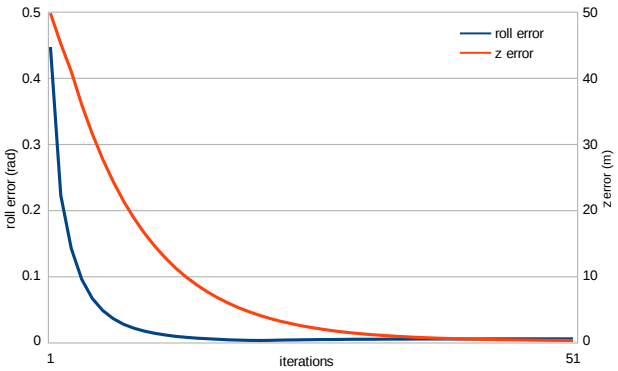


Fig. 8. This figure show the evolution of the correction from the first iteration to iteration 50. The number of iterations needed to correct the angle is much smaller than for correcting the error in translation.

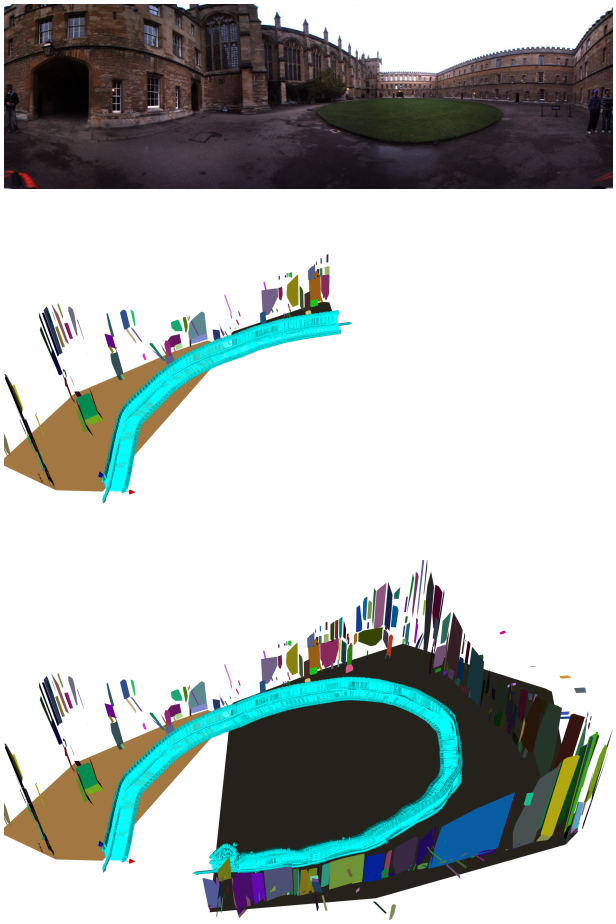


Fig. 9. Mapping using the New College dataset [15], the top image shows a view of the environment, the middle image shows half way through the loop, the bottom image show the resulting map.

A needed improvement to the algorithm itself, would be to get a better method to estimate the boundary of a plane, this would help to get a more representative limit of the plane, for visualisation, or for use by geometric-based loop closure algorithms [2].

While the method we have presented allow to detect planes, with a compact representation, which is useful in urban-like environment, it does not allow a full representation of the environment, especially, since not every objects in the environment is a plane. In future work, it would be interesting to investigate mixing other type of objects to get an even richer model environment, such as using cylinder and ellipse, or simply by adding local occupancy grid using the points that are not transformed into planes.

AUTHOR

Cyrille Berger – Department of Computer and Information Science,University of Linköping, SE-581 83 LINKÖPING, Sweden, e-mail: cyrille.berger@liu.se

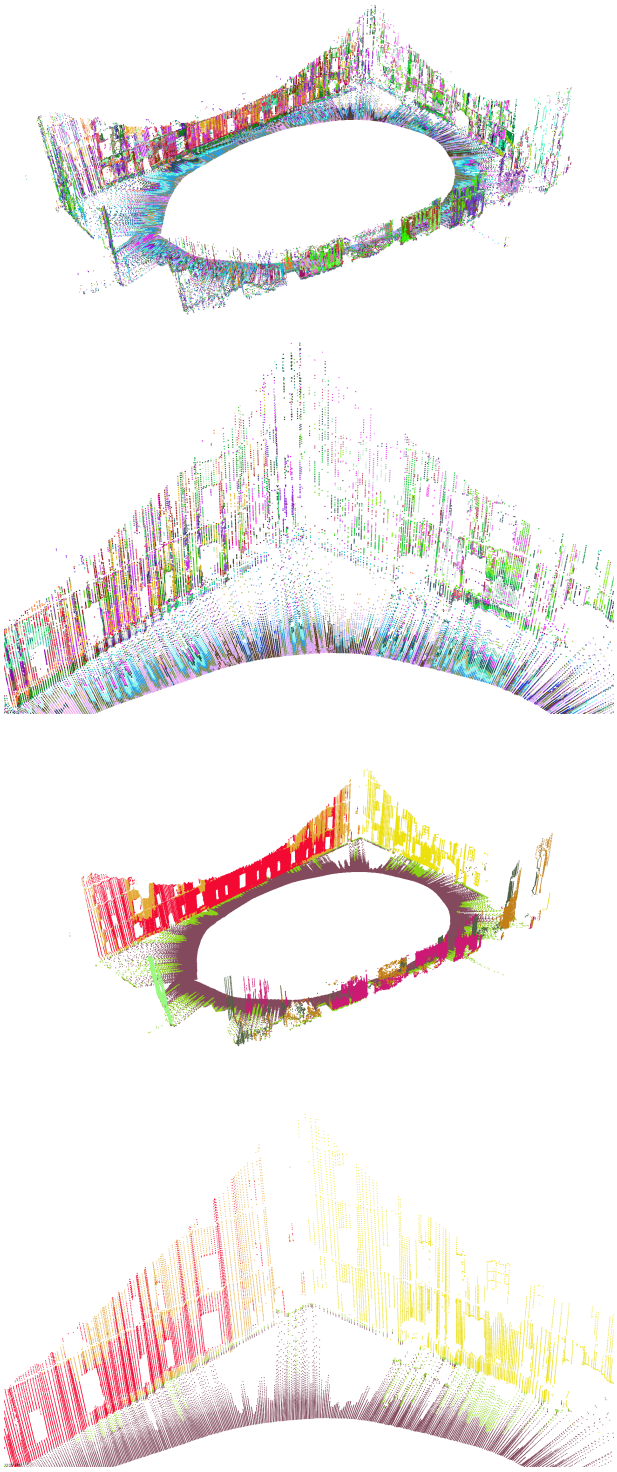


Fig. 10. This figure shows the segmentation of a points cloud accumulated on the New College data [15], using a RANSAC segmentation of the points cloud with the PCL library [14]. The top two figures use a precision of 1 cm while the bottom two use a precision of 10 cm. Points that belongs to the same plane are coloured using the same colours.

References

[1] T. Bailey, H. Durrant-Whyte, "Simultaneous Localisation and Mapping (SLAM): Part II – State of the Art", *Robotics and Automation Magazine*, September 2006.

[2] C. Berger, "Perception of the environment geometry for autonomous navigation", PhD thesis, University of Toulouse, 2009.

[3] C. Berger, "Weak constraints network optimiser". In: *IEEE International Conference on Robotics and Automation*, 2012.

[4] M. Duckham, L. Kulik, M. Worboys, A. Galton, "Efficient generation of simple polygons for characterizing the shape of a set of points in the plane", *Pattern Recognition*, vol. 41, no. 10, 2008, pp. 3224–3236.

[5] H. Durrant-Whyte, T. Bailey, "Simultaneous Localisation and Mapping (SLAM): Part I – The Essential Algorithms", *Robotics and Automation Magazine*, June 2006.

[6] M. Hebel, U. Stilla, "Pre-classification of points and segmentation of urban objects by scan line analysis of airborne lidar data". In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, 2008, pp. 105–110.

[7] J.H. Joung, K.H. An, J.W. Kang, M.J. Chung, W. Yu, "3d environment reconstruction using modified color icp algorithm by fusion of a camera and a 3d laser range finder". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3082–3088.

[8] M. Kirscht, C. Rinke, "3d reconstruction of buildings and vegetation from synthetic aperture radar (sar) images". In: *IAPR Workshop on Machine Vision Applications*, 1998, pp. 17–19.

[9] M. Morgan, A. Habib, "Interpolation of lidar data and automatic building extraction". In: *American Society of Photogrammetry and Remote Sensing Conference*, 2002.

[10] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, R. Siegwart, "A comparison of line extraction algorithms using 2d range data for indoor mobile robotics", *Autonomous Robots*, vol. 23, no. 2, 2007, pp. 97–111.

[11] S.T. Pfister, S.I. Roumeliotis, J.W. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation". In: *IEEE International Conference on Robotics and Automation*, 2003, pp. 14–19.

[12] T. Rodriguez, P. Sturm, P. Gargallo, N. Guilbert, A. Heyden, J.M. Menendez, and J.I. Ronda, "Photorealistic 3d reconstruction from handheld cameras", *Machine Vision and Applications*, vol. 16, no. 4, 2005, pp. 246–257.

[13] F. Rottensteiner, J. Jansa, "Automatic extraction of buildings from lidar data and aerial images". In: *International Society for Photogrammetry and Remote Sensing Symposium*, 2002.

[14] R.B. Rusu, S. Cousins, "3D is here: Point Cloud Library (PCL)". In: *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9–13, 2011.

[15] M. Smith, I. Baldwin, W. Churchill, R. Paul, P. Newman, "The new college vision and laser data set", *The International Journal of Robotics Research*, vol. 28, no. 5, May 2009, pp. 595–599.

[16] G. Vosselman, S. Dijkman, "3d building model reconstruction from point clouds and ground plans". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, pp. 37–44.

[17] K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems". In: *ICRA Workshop*, 2010.

[18] S. You, J. Hu, U. Neumann, P. Fox, "Urban site modeling from lidar". In: *International Conference on Computational Science and its Applications*, ICCSA'03, Springer-Verlag, Berlin, Heidelberg 2003, pp. 579–588.

[19] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces", *International Journal of Computer Vision*, vol. 13, 1994, pp. 119–152.