3D PLANE REGISTRATION USING UNCERTAINTIES

eingereichte
MASTERARBEIT
von

cand. ing. Christos Verginis

geb. am 12.04.1989
wohnhaft in:
Landsbergerstrasse 289
80678 München
Tel.: 0160 5400126

Lehrstuhl für
STEUERUNGS- UND REGELUNGSTECHNIK
Technische Universität München
Univ.-Prof. Dr.-Ing../Univ. Tokio Martin Buss
Univ.-Prof. Dr.-Ing. Sandra Hirche

Betreuer:	M.Sc. Sheraz Khan
Beginn:	15.10.2012
Zwischenbericht:	06.02.2013
Abgabe:	15.04.2013

Abstract

Acquiring models of the environment belongs to the fundamental tasks of mobile robots. In the last few years several researchers have focused on the problem of 3D simultaneous localization and mapping (SLAM). The most important SLAM subtask is the scan registration procedure, which deals with the deduction of the movement of the robot between consecutive scans, based on the shape of overlapping portions of the scans. An accurate pose estimate also enhances the autonomy of the robot by allowing it to navigate to the desired goal position in the map. Different approaches for SLAM of unknown environments have been proposed. However, most of them utilize point-based scan registration using also odometry information as initial guess, which can be inefficient in terms of time and memory and inaccurate, since the odometry measurements deviate extensively, even over short distances, especially in rough environments. In this thesis, we will focus on the accurate and efficient extraction of planar segments from 3D point clouds and on pose estimation based on plane registration techniques, since the smaller number of planes leads to greater efficiency. In order to avoid the error of odometry information, none is used. Instead, we establish correspondences of planes between consecutive scans. Certain uncertainty assumptions about the pose of the robot are made, which produce an uncertainty in the planar attributes that is modeled as a Gaussian distribution. The registration problem is then posed as an optimization, which iteratively refines the planar correspondences at each optimization step.

Abstract

Μία από τις βασικότερες εργασίες των κινουμένων ρομπότ είναι η απόκτηση μοντέλων του περιβάλλοντος. Τα τελευταία χρόνια η έρευνα έχει επικεντρωθεί στο πρόβλημα του 3D SLAM. Η σημαντικότερη υποεργασία του SLAM είναι η διαδικασία του ταυτοποίησης σαρώσεων του χώρου που λαμβάνει το ρομπότ (scans – scan registration), που ασχολείται με τον προσδιορισμό της κίνησης του ρομπότ μεταξύ διαδοχικών σαρώσεων, βασιζόμενη στο σχήμα επικαλυπτόμενων τμημάτων των σαρώσεων. Μία ακριβής εκτίμηση της στάσης (θέσης και προσανατολισμού) του ρομπότ ενισχύει επίσης την αυτονομία του επιτρέποντάς του να πλοηγηθεί στην επιθυμητή θέση-στόχο στο χάρτη. Έχουν προταθεί διάφορες προσεγγίσεις για το SLAM άγνωστων περιβάλλοντων. Ωστόσο, οι περισσότερες από αυτές χρησιμοποιούν ταυτοποίηση σαρώσεων που βασίζεται σε 3D σημεία και οδομετρικές μετρήσεις ως αρχική υπόθεση καθιστώντας τες μη αποδοτικές σε θέματα υπολογιστικού χρόνου και κατανάλωσης μνήμης και μη ακριβείς, αφού οι οδομετρικές μετρήσεις αποκλίνουν εκτενώς, ακόμη και σε μικρές αποστάσεις, ειδικά σε ένα τραχύ περιβάλλον. Σε αυτή την εργασία, θα επικεντρωθούμε στην ακριβή και αποδοτική εξαγωγή επιπέδων τμημάτων από 3D pointclouds και στην εκτίμηση της στάσης του ρομπότ με βάση τεχνικές ταυτοποίησης επιπέδων, αφού ο μικρότερος αριθμός επιπέδων οδηγεί σε μεγαλύτερη αποδοτικότητα. Προς αποφυγή του οδομετρικού σφάλματος, δεν θα χρησιμοποιηθούν πληροφορίες οδομετρικών

μετρήσεων. Αντί αυτού, δημιουργούμε αντιστοιχήσεις μεταξύ επιπέδων διαδοχικών σαρώσεων. Γίνονται συγκεκριμένες υποθέσεις αβεβαιότητας για τη στάση του ρομπότ, που παράγουν μία αβεβαιότητα στα χαρακτηριστικά των επιπέδων, η οποία μοντελοποιείται ως μία κατανομή Gauss. Στη συνέχεια, το πρόβλημα της ταυτοποίησης διαμορφώνεται ως μία επαναληπτική βελτιστοποίηση στις αντιστοιχήσεις των επιπέδων σε κάθε βήμα.

Contents

1 Introduction

The advance of technology the recent years, especially in computer science and in the field of robotics, has increased the capabilities of robot navigation and mapping and the motivation for it. Several approaches to the 2D SLAM problem have been proposed the last decade and even more researchers extend their work to 3D SLAM. Navigation in unknown 3D environments demands accurate estimation of the robot’s pose and mapping of the surrounding area, things that cannot be achieved by the sole use of odometry. Most algorithms that have been suggested so far consider the use of 3D points. The most common among them are the Iterative closest point (ICP) and the 3D Normal distribution transform (NDT) algorithms, introduced in [1] and [2], respectively. However, due to the fact that these algorithms adopt pointwise techniques, the computation time and memory complexity can be very high, as the number of points received from the recently developed sensors is usually significantly large. For this reason, an extension to 3D Plane SLAM is necessary. The number of planar segments that are extracted from point clouds is comparatively smaller than the amount of points, since they are formulated by groups of the latter, and that makes the use of plane-based algorithms much more efficient in terms of time and memory. In addition, planar segments provide a more intuitive representation of the environment. Moreover, the exploitation of odometry measurements for the 3D SLAM problem is presented in several algorithms, in order to register pairwise scans; that is, align consecutive scans over time, either of points or of planar segments. In [3] and [4] a lightweight orthogonal 3D SLAM algorithm is presented. However, their work does not apply for many environments, focusing mostly on indoor ones. In [5] a plane-based solution is suggested, using also unknown correspondences, i.e. similar attributes between planes of consecutive scans. In it, the plane parameters uncertainty is also taken into account. A very comprehensive discussion on finding correspondences between two sets of planar or quadratic patches using attribute-graphs is found in [6]. In [7] an approach is considered utilizing correspondences without the use of odometry as an initial guess.

The accurate plane extraction can be considered as the core of the 3D Plane SLAM idea, since it has a great impact on the result. This procedure has an increased level of difficulty both in indoor and outdoor environments. Especially in the latter case, the plane parameters estimation, such as the normal vector and the offset, is very challenging, as the point clouds received by several types of sensors are noisy and the structure of the environment can be of high complexity (e.g. complex objects such as trees are contained). Several approaches for plane fitting algorithms have been proposed in [8], [9] and [10].

The main goal of this topic is the extension of the 3D NDT algorithm utilizing correspondences of planar segments without the use of odometry as an initial guess to perform scan registration. The pipeline of Fig. 1.1 summarizes the above procedure.

Although the plane extraction step is not the main goal of the thesis, it is crucial for the accuracy of the scan registration procedure. For that reason, a new algorithm is introduced. The rest of the topic is organized as follows:

Chapter 2 mainly focuses on a brief presentation of several plane extraction algorithms. Furthermore, the basic idea of a new algorithm is featured and experimentally evaluated in real world data sets.

Chapter 3 presents the most common point-based 3D registration algorithms and attempts a comparison of them. In addition, the concept of 3D plane registration is analysed and the extension of the 3D NDT algorithm considering the use of planes and correspondences without any odometry information is presented.

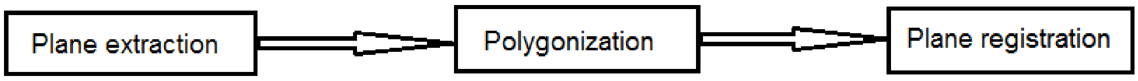


Figure 1.1: Registration Pipeline

2 Segmentation

Many 3D robotic applications commonly utilize the basic primitive of 3D points mostly for the representation of the surrounding environment but for other tasks as well. Until now points have been used in most 3D scan registration algorithms and autonomous exploration tasks of unknown environments. However, one could say that this kind of environmental representation is far from human intuition. Furthermore, due to the fact that the amount of points received is usually significantly large, the computational cost in several procedures becomes prohibitive for on-line applications. Therefore, the use of plane polygons and rectangles is considered as a better alternative. The transition from points to plane polygons is of high significance for all the robotic applications, especially for 3D plane SLAM, which is the main goal of this topic. Planes offer a more intuitive representation of the environment, clearly contain much more information than points do, such as the normal vector of the plane, through which the orientation knowledge of it is obtained, contributing to the localization problem. Moreover, planes require much less memory space than points do, since they are formed by groups of points, a fact that makes their use more efficient.

A variety of algorithms have been proposed the recent years for efficient extraction of planes from 3D point cloud received from various types of sensors, such as laser range finders. Due to the fact that planes play a very important role in 3D scan registration, it is critical that the results must strictly correspond and fit to the planes of the environment in the real world. The noise caused by the sensors makes the plane extraction procedure significantly difficult, since the estimation of the plane parameters such as the normal vector and the offset is not accurate. Moreover, an outdoor environment has more complicated structure and the plane fitting process is more challenging than it would be in an indoor environment. For these reasons, several types of errors must be considered, such as the mean square error of the points that form each plane. The prior knowledge of the sensor employed allows the modelling of the noise in the plane extraction procedure using calibration techniques. This Chapter is organized as follows. Section 2.1 provides some mathematical background and overview of related work. In Section 2.2 we propose a segmentation algorithm whose main idea is based on the octree data structure.

2.1 Mathematical background and related work

2.1.1 Mathematical background

The basic context behind the plane fitting procedure is the estimation of the plane parameters; that is, the normal vector and the offset. The normal vector is represented by the eigenvector that corresponds to the minimum eigenvalue of the covariance matrix C of the *n* points that form the plane. The covariance matrix C is computed using the equation:

$$C = \begin{bmatrix} C_{xx} & C_{xy} & C_{xz} \\ C_{yx} & C_{yy} & C_{yz} \\ C_{zx} & C_{zy} & C_{zz} \end{bmatrix} \qquad (2.1)$$

where $C_{ab} = \sum_{i=1}^{n}(a_i - m_a)(b_i - m_b)$, *a*, *b* is the notation for {*x*, *y*, *z*}and $\vec{m}$ is the 3D centroid of *n* points:

$$\vec{m} = \frac{1}{n}\sum_{i=1}^{n}\vec{pi},\ n \geq 3,\ \vec{pi} = (x_i, y_i, z_i),\ i = 1,\ldots,n \qquad (2.2)$$

The minimum number of points must be 3 so that a plane can be formed uniquely in the 3D space. The offset of the plane is computed according to the formula:

$$d = -(n_x m_x + n_y m_y + n_z m_z) \qquad (2.3)$$

where n_x, n_y, n_z are the coordinates of the normal vector assuming that the centroid computed satisfies the plane equation:

$$\vec{n}^T \vec{p} + d = 0 \qquad (2.4)$$

More detailed information about planes can be found in [8]. [11] gives information about eigenvalues and eigenvectors and [12] about the covariance matrix computation.

2.1.2 Related work

1. **PCL Region growing segmentation**

This algorithm is described in [8] and this is how it works. First of all it sorts the points by their curvature value. This needs to be done because the region begins its growth from the point that has the minimum curvature value. The reason for this is that the point

with the minimum curvature is located in the flat area (growth from the flattest area allows to reduce the total number of segments).
So we have the sorted cloud. Until there are unlabeled points in the cloud, the algorithm picks up the point with minimum curvature value and starts the growth of the region. This process occurs as follows:

- The selected point is added to the set called seeds
- For every seed point the algorithm finds neighbor points
    - Every neighbor is tested for the angle between its normal and the normal of the current seed point. If the angle is less than a threshold value then the current point is added to the current region.
    - After that every neighbor is tested for the curvature value. If the curvature is less than a threshold value then this point is added to the seeds.
    - Current seed is removed from the seeds

If the seed set becomes empty this means that the algorithm has grown the region and the process is repeated from the beginning. .

2. Fast plane detection in noisy 3D range images

The second plane extraction algorithm is described in [9] and its implementation is slightly different than the one presented above. Firstly the input pointcloud is transformed to a range image to achieve the notion of vicinity. A random point p_1 and its nearest neighbor p_2 from point cloud data *PC* are taken through the range image. This is the initial set of points – region Π. Then an extension of this region by considering points in increasing distance from set Π is done. Now suppose point *p’* is such that the distance between it and the region is less than the distance δ. Then if the mean square error (MSE) to the optimal plane Ω of the region $\Pi \cup p'$ is less than ϵ and if the distance between the new point and the optimal plane Ω is less than γ, then *p’* is added to the current region Π. This region is expanded until no points can be added. Afterwards if the region size is more than θ it is added to the set of regions *R*, else these points are treated as unidentified and are added to the set *R’*. This is repeated until each point from *PC* is either in *R* or in *R’*.
The key part of this region growing algorithm is that the computation of the mean square error is being done in an incremental way. In particular, if *C* is the covariance matrix described above and C_{ij} are its elements, then each time a new point is added to the region, the update formula for the new C_{ij} $(n + 1)$ is:

$$C_{ij}^{+} = C_{ij} + i_{n+1} j_{n+1} - m_{n+1}^{i} S_{n+1}^{j} + m_{n}^{j} S_{n}^{j} \quad (2.5)$$

Where S_n^i is the sum of *n* points in the *i*-th coordinate and m_n^i the *i*-th coordinate of the centroid of *n* points.
The general formula for the mean square error computation is:

$$MSE(k) = \frac{1}{k}\sum_{i=1}^{k}(\vec{n}\,\vec{p}_i + d)^2 \qquad (2.6)$$

where *k* is the number of points, p_i is the 3D point, *n* is the plane normal vector and *d* is the plane offset. Expanding this equation gives a form which is suitable for incremental calculation:

$$MSE(k) = \frac{1}{k}\sum_{a,b}\left(n_a\, n_b \sum_{i=1}^{k} p_a^{(i)} p_b^{(i)}\right) + \frac{2d}{k}\sum_{i=1}^{k} n_a S_a(k) + d^2 \qquad (2.7)$$

where *a*, *b* is the notation for {*x*, *y*, *z*}.
The mean square error and the distance between the point and the optimal plane are two factors that enhance the performance of the algorithm, making it suitable even for noisy pointclouds. Another advantage one could notice from the above description is that the incremental way of calculating essential attributes makes the algorithm efficient in terms of computational time, despite the point-based region growing procedure.

3. Fast plane detection for SLAM from noisy range images in both structured and unstructured environments

Two segmentation algorithms are suggested in the particular paper, which are described thoroughly in [10]. The first one is a slight variant of the algorithm introduced by [9] and briefly described above. The basic differences lie in the initialization of each region and in the computation of the mean square error each time a new point is added to a region. Regarding the initialization part, a characterization is assigned to each point, depending on the eigenvalue of the covariance matrix that is calculated by it and its neighbors. As initial points to each region, those characterized as planar are chosen and expanded using the same procedure that is described in the previous algorithm. The mean value computation is done in a much faster way resulting in high efficiency in terms of time and memory. In particular, a plane can be described by the equation:

$$\vec{n}^T\,\vec{p} = d \qquad (2.8)$$

where *n* is the normal vector of the plane, *p* is an arbitrary point on the plane and *d* is the plane offset. Assuming that the centroid *m* of the points that form the plane is part of the plane, it derives that:

$$\vec{n}^T\,\vec{m} = d \qquad (2.9)$$

Hence, using (2.9), (2.6) can be rewritten as:

$$MSE(k) = \frac{1}{k}\sum_{i=1}^{k}(\vec{n}\,p_i + \vec{n}\,m)^2 \qquad (2.10)$$

which is

$$MSE = \frac{1}{k}(\vec{n}^T C \vec{n}) \qquad (2.11)$$

The normal *n* is the eigenvector of *C* that corresponds to the minimal eigenvalue, as mentioned above. Therefore, the *MSE* can be derived:

$$MSE = \frac{1}{k}\lambda_{min}(C) \qquad (2.12)$$

where λ stands for the minimum eigenvalue of *C*.

The second algorithm considers a grid based growing segmentation. Initially, the same procedure as in the above algorithm is followed. The growing procedure now is done using planar grids instead of points. The normal vectors of these small grids that resulted from the initialization are compared in order to check the difference in their orientation. Moreover, the perpendicular distance of the centroid of each grid to the current plane is checked in order to decide whether this grid should be added to the plane or not. Finally, if two planar grids are merged, a further check on the mean square error of the resulted plane is conducted.

2.2 Plane extraction using the octree data structure

This algorithm was implemented for the purposes of this topic. Its main idea consists of two parts, an initialization part and a merging one. The initialization part exploits the advantages of the octree data structure, for which information can be gained in [13] and [14]. This is done by fitting initial planes in small neighborhoods of points already stored in the octree. In the octree data structure the sense of space is conceived therefore the notion of vicinity has direct correspondence to the physical environment. Each leaf node of the octree corresponds to a certain volume defined by the resolution of the tree. For example, the leaves of an octree with resolution 0.05 correspond to a cube with edge of 0.05m. Every node other than the leaves divides the space into eight octants and its volume equals to the sum of the volumes of its children. Utilizing these two properties, it is possible to extract planes locally from points that are contained in small neighborhoods. These neighborhoods are described by a certain volume which is defined by the initialization level chosen in the tree (certain height of the tree). Moreover, the octree data structure can perform a down sampling of points due to its resolution (e.g., two different points whose distance is less than the resolution will both be assigned to the same node), resulting in higher efficiency. In addition, the noise that the sensors cause can be reduced by checking the density of each volume that is used for the plane fitting technique. Areas with small point density (determined by a threshold value) and with large mean square error are ignored.

The second part of the algorithm consists of merging the initial planes into bigger ones in order for their number to be decreased and to gain a clearer and a more compact representation of the environment. The initial planes are inserted in a 3D array structure using their centroids to determine the position they will be assigned to. This provides an efficient neighbor search, as nearby planes will be assigned to nearby cells of the array.
For the neighbor planes, a test is conducted between the angle of their normal vectors, and their perpendicular distance. If two or more planes have similar orientation and small perpendicular distance, they are candidates for merging. After that, three more checks are conducted. Firstly, the mean square error of the merged plane must be under a certain threshold value. Secondly, the total number of points that the merged plane contains must be over a threshold value so that small planes or planes with small point density are rejected. Finally, the maximum eigenvalue of the plane covariance matrix is checked. This value describes the radius which forms a hypothetic circle around the centroid of the planar segment that contains all of its points. Hence, it gives an estimation of its size. This is described analytically in [15]. So in order to reject small planes, this value must overcome a certain threshold. In Fig. 2.1 a pseudocode of the above algorithm is presented.

```
PLANE EXTRACTION USING OCTREE:
Input:
        Pointcloud PCL
        Octree O
        Octree resolution r
        Initialization height h
        Thresholds T{LocalMseThres, LocalPointThres, MseThres, PointThres,
AngleThres. PerpendicularDisThres, MaxEigenvalueThres }
Output:
        Planar Segments {PS}
Procedure:
                -   Local Planes {LP} ← Ø
                -   Insert points from PCL into octree
                -   Points {P} ← Ø
                For all $N_h$ do   // Nodes of the tree at height h
                        o   P = $N_{h \to p}$   //points in all the descendant leaf nodes from inner
                            //node N
                        If ((size of P) > T(LocalPointThres) )
                                ▪   IPS = Fit_Plane {P}      //initial Plane Segment
                                ▪   Mse = compute_mse{IPS}
                                If (Mse < T(LocalMseThres) )
                                        •   {LP} ← {LP} ∪ IPS
                                end if
                        end if
                end for
                -   Grid = Construct_3D_array(LP)
                -   {PS} = Merging_function{LP, T, Grid}
```
Figure 2.1(a): Pseudocode of the plane extraction algorithm using octree

```
MERGING_FUNCTION
Input:
        Initial Plane Segmets {LP}
        Thresholds T
        3D array of initial planes Grid
Output:
        Planar Segments {PS}
Procedure:
                for all initial planes LP(i) in Grid do
                        {N} = compute_neighbors(LP(i), Grid)
                        For all neighbors j do
                                ▪ P_d = ComputePerpendicularDistance(LP(i), N(j) )
                                ▪ theta = $\cos^{-1}$( dot(LP(i).normal, N(j).normal) )
                                if (P_d < T(PerpendicularDisThres)) && (theta < T(AngleThres)
                                        • LP(i) = merge(LP(i), N(j) )
                                end if
                                if (LP(i).mse < T(MseThres) ) && ( size of(LP(i)) > T(PointThres) ) && (radius > T(MaxEigenvalueThres) )
                                        • {PS} ← {PS}∪{ LP(i)}
                                end if
                        end for
                end for

//radius is the maximum eigenvalue of the covariance matrix of the plane
```

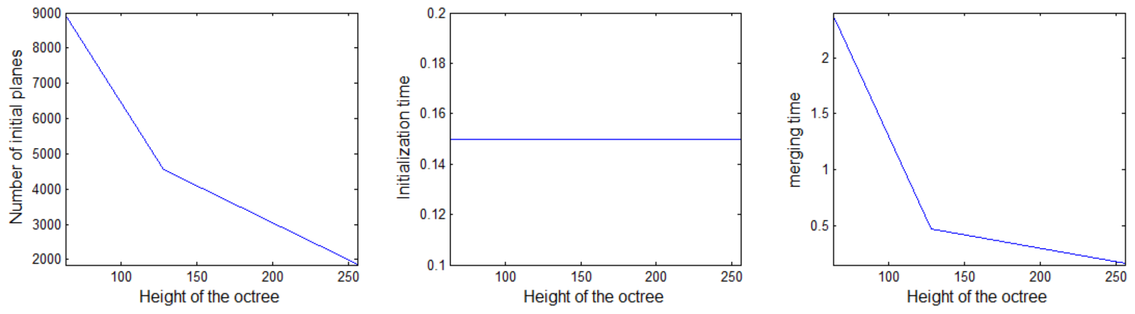
Figure 2.1(b): Pseudocode of the merging function

2.2.1 Experimental Evaluation

This Section focuses on the evaluation of the suggested algorithm in real world data sets. Specifically, three data set were tested, the Bremen city center data set[1], the Freiburg campus data set[2] and an indoor environment data set[3]. For the first data set a noise model was also employed to estimate the uncertainty of the points imposed by the sensor. The octree implementation from [34] was employed. The experiments mentioned in this section were carried out on an Intel® Core i5-2500K, 3.30 GHz processor with 16GB memory.
Concerning the Bremen city center data set, it consists of 14 files containing pointcloud data, with an average number of 295000 points per file. The Freiburg Campus data set consists of 78 files containing pointcloud data, with an average number of 159000 points per file. Finally, the indoor environment data set consists of 60 files, with an average number of 112498 points.
The algorithm was tested in the first file of the Bremen city center data set, which contains 233399 points. Fig. 2.2 shows the number of initial planes, the initialization time and the merging time for different values of the height *h* of the tree at which the

[1] Courtesy of Dorit Borrmann and Jan Elseberg available at the Osnabrueck robotic 3D scan repository, http://kos.informatik.uniosnabrueck. de/3Dscans/
[2] Courtesy of B. Steder and R Kuemmerle, available at http://ais.informatik.uni-freiburg.de/projects/datasets/octomap/
[3] Courtesy of Martin Magnusson, available at the Osnabrueck robotic 3D scan repository, http://kos.informatik.uni-osnabrueck.de/3Dscans/

initial planes are constructed. It must be pointed out that *h* must be a number that is power of two. The values used for the extraction of these results are *h* = 64, 128 and 256 with octree resolution *r* = 2cm. That is, for *h* = 64, we extract initial planes in cubic cells of size 2*64 = 128x128x128 cm^3. For the other cases, this size can be calculated respectively. The other parameters of the algorithm (e.g., mean square error threshold value) are chosen using the sensor features and calibration techniques.
In Fig. 2.2 (a) and (c) we see that the number of planes and the merging time increase as *h* decreases. That is an expected outcome, because the lower the value of *h*, the smaller the cells in which initial planes are constructed. Hence we expect the initial planes to be more as the same space is described by a larger number of them when the value of *h* is smaller. Also, the larger the number of the initial planes, the higher the time taken to merge them. Fig. 2.2 (b) shows that the time needed to construct the initial planes does not change with respect to *h*. This can be explained by the fact that the same octree resolution was utilized for all three values of *h* so the total number of the octree nodes that has to be traversed is almost the same.



(a) Number of Planes (b) Initialization time (c) Merging time

Figure 2.2: (a) Number of initial planes, (b) Time to construct initial planes and (c) Time to merge the initial planes, with respect to the height of the octree that the initialization is conducted

Fig. 2.3 shows the segmentation procedure for the 1st scan of the Bremen city center data set. In Fig. 2.3 (a) we see two views of the input pointcloud. Fig. 2.3 (b) and (c) show the same views using the initial and final planes, respectively. The algorithm parameters and runtimes are depicted in Table 2.1(a) and (b) respectively.

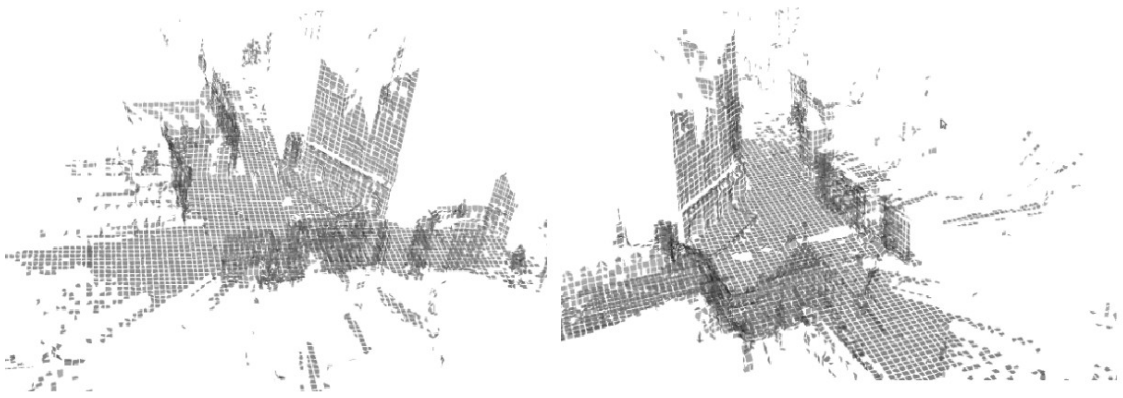
Octree Resolution (m)	0.05
Initialization height *h*	32
Local MSE threshold(m)	$8x10^5$
Local points threshold	10
MSE threshold(m)	$6x10^6$
Angle threshold(º)	15
Perpendicular distance threshold(m)	0.1
Points threshold	80
Max eigenvalue threshold	0.25

Table 2.1 (a): Parameters of the segmentation algorithm for the 1st file of the Bremen city center data set

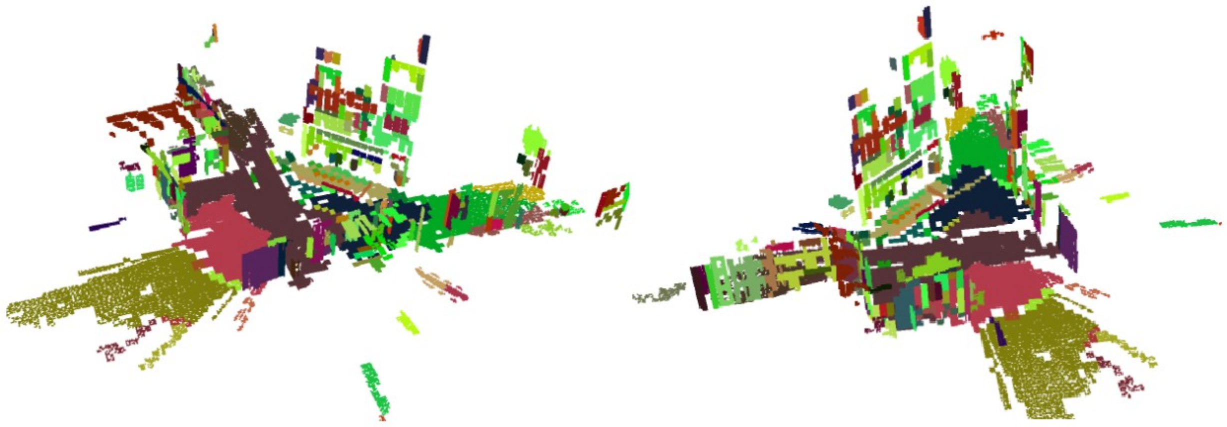
Input Pointcloud points	233399
Time to insert in octree (s)	0.08
Initialization time (s)	0.11
Number of initial planes	5807
Merging time (s)	1.53
Number of resulted planes	216
Total time (s)	1.72

Table 2.1 (b): Times and number of planes for the 1st file of the Bremen city center data set

(a): Initial pointcloud of the 1st scan of the Bremen city center data set



(b): Planes after initialization



(c): Final planes

Figure 2.3: Segmentation procedure for the 1st scan of the Bremen city center data set. (a) initial pointcloud, (b) planes after initialization, (c) final planes

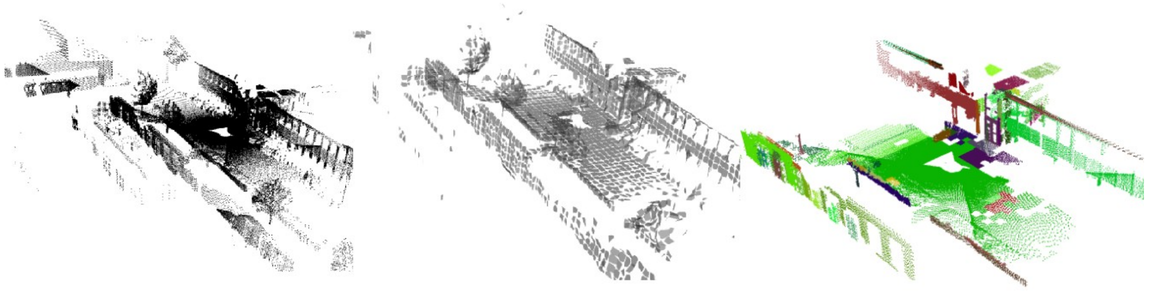
Fig. 2.4 shows the segmentation procedure for the first file of the Freiburg campus data set, which consists of approximately 176251 points. The algorithm parameters utilized are depicted in Table 2.2(a). The runtimes and number of planes are depicted in Table 2.2(b).

Octree Resolution (m)	0.02
Initialization height *h*	32
Local MSE threshold(m)	10^{-2}
Local points threshold	5
MSE threshold(m)	$7x10^{-2}$
Angle threshold(º)	15
Perpendicular distance threshold(m)	0.1
Points threshold	100
Max eigenvalue threshold	0.35

Table 2.2 (a): Parameters of the segmentation algorithm for the 1st file of the Freiburg campus data set

Input pointcloud points	176251
Time to insert in octree (s)	0.06
Initialization time (s)	0.09
Number of initial planes	2417
Merging time (s)	0.3
Number of resulted planes	66
Total time (s)	0.45

Table 2.2 (b): Runtimes and number of planes for the 1st file of the Freiburg campus data set



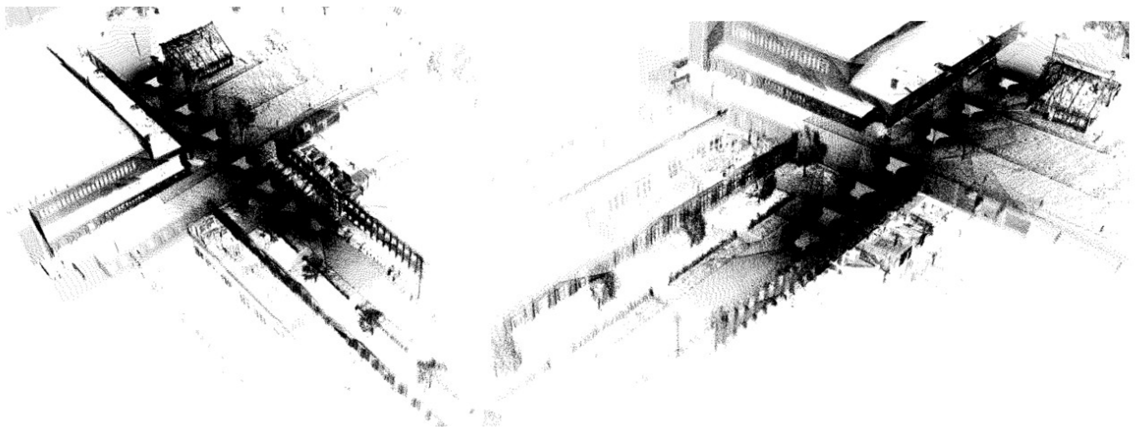
(a) Initial pointcloud (b) Initial planes (c) Merged planes

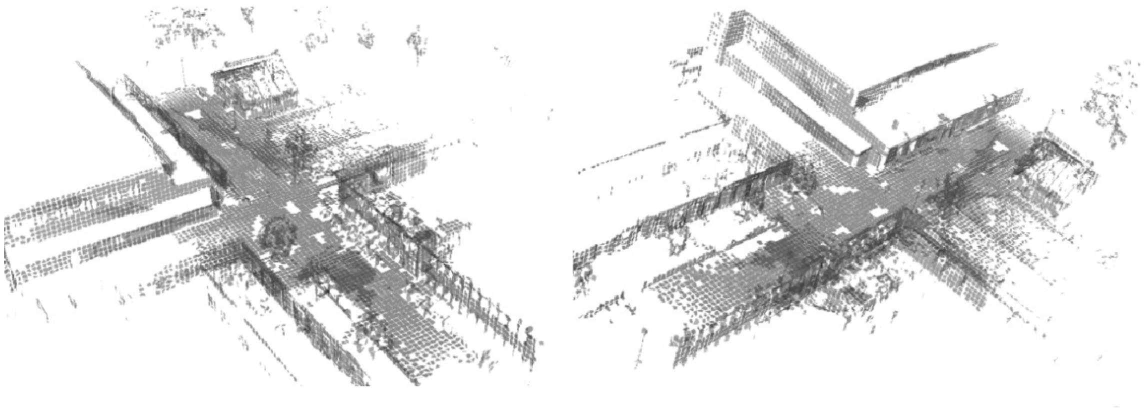
Figure 2.4: Segmentation procedure for the 1st scan of the Freiburg campus data set. (a) initial pointcloud, (b) planes after initialization, (c) final planes

In order to get a more compact view of the Freiburg campus, the algorithm was also tested in the first 7 files of the specific data set. Fig. 2.5 shows the results. The runtimes and number of planes are depicted in Table 2.3. The algorithm parameters are the same as in Table 2.2(a).

Input pointcloud points	1141086
Time to insert in octree (s)	0.84
Initialization time (s)	0.82
Number of initial planes	9701
Merging time (s)	4.95
Number of resulted planes	319
Total time (s)	6.61

Table 2.3: Runtimes and number of planes for the first 7 files of the Freiburg campus data set


(a): Initial pointcloud of the first 7 scans of the Freiburg campus data set


(b): Planes after initialization

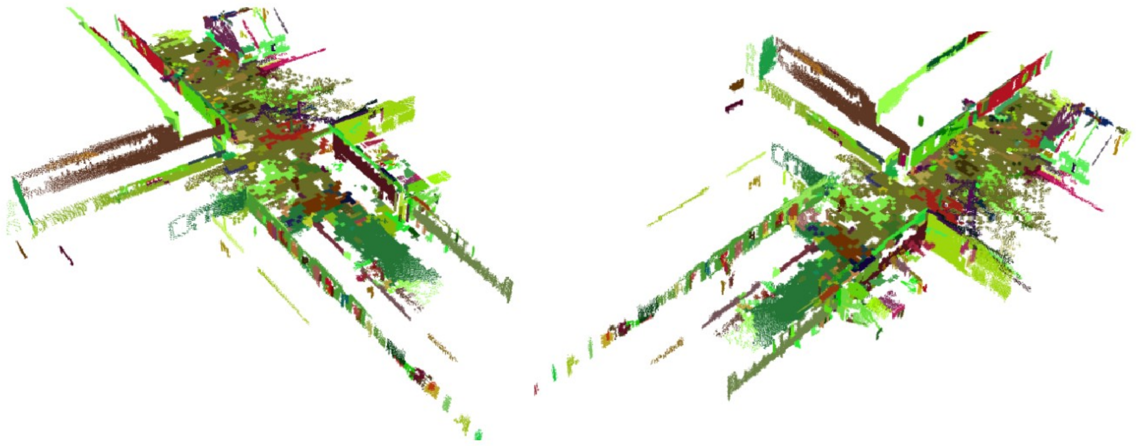

(c): Final planes

Figure 2.5: Segmentation procedure for the first 7 scans of the Freiburg campus data set. (a) initial pointcloud, (b) planes after initialization, (c) final planes

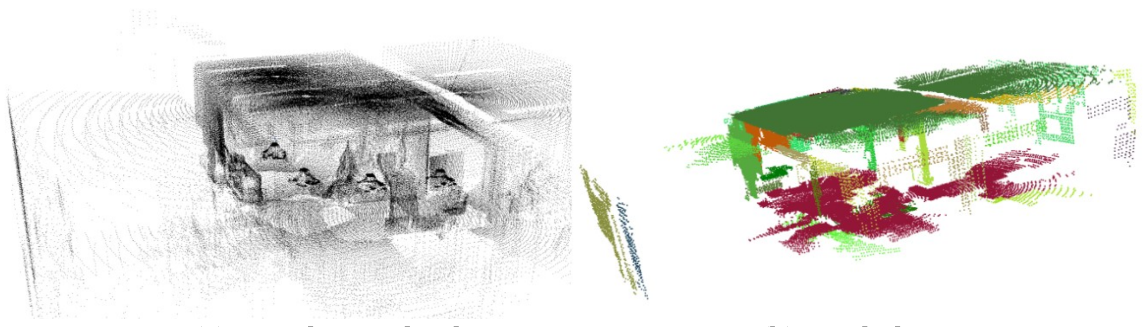
Finally, the algorithm was also evaluated for the first 4 scans of the indoor dataset. Table 2.4(a) and (b) show the algorithm parameters and results respectively. Fig. 2.6 depicts the initial pointcloud (a) and the final planar segments (b). We can see from the visualization that the results are clearer and more compact. This can be explained by the fact that indoor environments are more structured and consist mainly of big planar segments.

Octree Resolution (m)	0.02
Initialization height *h*	16
Local MSE threshold(m)	8x10^{-3}
Local points threshold	5
MSE threshold(m)	10^{-2}
Angle threshold(º)	10
Perpendicular distance threshold(m)	0.05
Points threshold	100
Max eigenvalue threshold	0.35

Table 2.4 (a): Parameters of the segmentation algorithm for the first 4 files of the indoor data set

Input pointcloud points	449992
Time to insert in octree (s)	0.05
Initialization time (s)	0.08
Number of initial planes	3908
Merging time (s)	0.6
Number of resulted planes	47
Total time (s)	0.73

Table 2.4: Runtimes and number of planes for the first 4 files of the indoor data set



(a) Initial pointcloud (b) Final planes

Figure 2.6: Segmentation results for the first 4 scans of the indoor data set. (a) Initial pointcoud and (b) Final planes

One could notice from the above results that the algorithm is suitable for online applications, such as 3D SLAM and navigation procedures, as the runtimes are fairly low. This can be attributed mainly to the octree structure, which allows for fast plane initialization.

3 3D Scan registration

The 3D simultaneous localization and mapping, widely known as SLAM problem, has been one of the most popular research issues in the field of robotics and particularly of mobile robots the last few years. The main part of the SLAM procedure is the scan registration. That is, the procedure of aligning two consecutive scans received from a sensor in order to achieve the proper calculation of the transformation of the robot between the two scans. Many approaches to the problem have been proposed so far not only utilizing a 3D point-based representation of the environment but also a plane-based representation. In [16] and [17] ICP based algorithms are presented and the 3D NDT approach is featured in [2]. Plane-based techniques are suggested in [5] and in [18].

In the following subsections, the two approaches to the scan registration problem using points and planar segments are considered. In 3.1 the most commonly used point-based algorithms are presented and compared, the ICP and 3D NDT. Subsequently, 3.2 deals with the advantages of 3D scan registration using planes and the concept of applying the use of planes in the 3D NDT algorithm. Moreover, a way of establishing correspondences is suggested, as no odometry information is used. Finally, in Section 3.3 the algorithm is evaluated for the indoor and the dwelling scenario[4] data sets.

3.1 3D scan registration using points

Most 3D SLAM algorithms that have been implemented so far are based on the ability of registering two range scans or a range to a map, using 3D points. The goal of two range scans registration is to find the relative pose between the two positions, at which the scans were taken. The basis of most successful algorithms is the establishment of correspondences between the primitives of the two scans (e.g. points). Out of this, an error can be derived and minimized. The most general approach, using points, is the ICP algorithm introduced in [1] and a variant of it introduced in [19]. These approaches require an establishment of explicit correspondences between points (points that correspond to the same physical point in the real world). Another approach is the Normal Distribution Transform (NDT) algorithm, which was introduced for 2D SLAM in [20] and an extension of it for 3D SLAM that can be found in [2] and in [21]. In the following subsection, the ICP and 3D-NDT are briefly presented, as they are considered to be two of the most significant scan registration algorithms using the primitive of points in the 3D space and a comparison is made.

[4] Courtesy of Jacobs University Robotics department, available at: http://robotics.jacobs-university.de/node/293

3.1.1 ICP algorithm

The Iterative Closest Point (ICP) algorithm was developed by P.Besl and N.McKay [1] and is usually used to register two consecutive clouds of points in a common coordinate system. The ICP algorithm has commonly been used for many robotic applications including SLAM, as described in [22]. The basic idea here is to minimize the difference between the points of the two sets. The procedure is done iteratively. That is, in each step, the algorithm selects the correspondence points according to the minimum distance and calculates the transformation (R,t) using an initial guess (odometry estimation) for minimizing a certain heuristic mean square error function, usually

$$E(R,t) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} W_{i,j} \left| m_i - \left(R d_j + t \right) \right|^2 \quad (3.1)$$

where N_m and N_d represent the number of points in the two sets and $w_{i,j}$ are the weights for a point match, which are either equal to 1 when the points *i* and *j* are the closest ones between the two scans and 0 otherwise. According to the above equation, the transformation can be calculated using a variety of algorithms, as suggested in [23], [24] and [25].

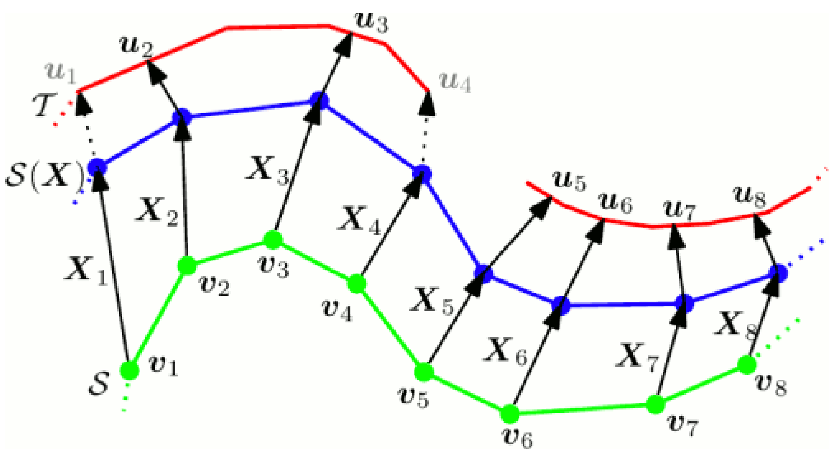


Figure 3.1: Example of the ICP algorithm

3.1.2 3D-NDT algorithm

The Normal Distributions Transform can be described as a method for compactly representing a surface. As mentioned above, its 2D variant was introduced by Biber and Strasser in [20] and an extension for 3D applications can be found in [21]. The transform maps a point cloud to a smooth surface representation, described as a set of local probability density functions (PDFs), each of which describes the shape of a section of the surface. The algorithm firstly divides the occupied 3D space into a grid of cells (i.e. cubes) and a PDF is assigned to each cell, based on the point distribution within it. An appropriate PDF could be a normal distribution such as the following

$$p(\vec{x}) = \frac{1}{(2\pi)^{D/2}\sqrt{|\Sigma|}} \exp\left(-\frac{(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})}{2}\right) \qquad (3.2)$$

where D is the dimension notation and $\vec{\mu}$ and Σ denote the mean vector and covariance matrix of the reference scan surface points within the cell where $\vec{x}$ lies, or a mixture of a normal and a uniform distribution.

By using NDT for scan registration, the goal is to find the pose of the current scan that maximizes the likelihood that the points of the current scan lie on the reference scan surface. This likelihood is expressed by the function

$$\prod_{k=1}^{n} p(T(\vec{p}, \vec{x}_k)) \qquad (3.3)$$

where $\vec{x}_k$ are the k points from the current scan, $\vec{p}$ is a pose and $T(\vec{p}, \vec{x}_k)$ is a spatial transformation function that moves the point $\vec{x}_k$by the pose $\vec{p}$. The best pose $\vec{p}$ should be the one that maximizes the above function. Given the above parameters, the NDT score function is

$$s(\vec{p}) = -\sum_{k=1}^{n} p(T(\vec{p}, \vec{x}_k)) \qquad (3.4)$$

which corresponds to the likelihood that the points $\vec{x}_k$ lie on the surface of the reference scan, when transformed by $\vec{p}$.

Newton's algorithm can be employed to find the parameters $\vec{p}$ that optimize $s(\vec{p})$. Newton's method iteratively solves the equation $H\Delta\vec{p} = -\vec{g}$ where H and $\vec{g}$ are the Hessian matrix and gradient vector of $s(\vec{p})$. The increment $\Delta\vec{p}$ is added to the current pose estimate in each iteration, so that $\vec{p} = \vec{p} + \Delta\vec{p}$. As initial transformation for the algorithm, the one estimated with the use of odometry is commonly used. More detailed information about Newton's algorithm can be found in [26] and about the maximization of the above likelihood function in [21].

3.1.3 Comparison

The basic difference between the NDT and ICP algorithm is that using the first, no explicit correspondences have to be found between points or features. Moreover, NDT is done in a probabilistic manner and that makes it more efficient in “difficult” scans; that is, scans with few prominent geometric features, little overlap, and high noise level. However, one could say that the complexity of the PDF and the Newton's algorithm computation may result in an increase in time and memory complexity. On the other hand, most ICP algorithms employ tree data structures (usually k-d trees as in [27]) for storing the points facilitating the establishment of the correspondences using a nearest neighbor search. An explicit comparison between the two algorithms was conducted in [28].

3.2 3D plane registration

Point-based algorithms used for 3D scan registration, such as the ICP algorithm and its variants, apart from being computationally expensive and slow for large point clouds, also suffer from premature convergence to local minima, especially when the overlap between scene-samples decreases. By using more abstract primitives, such as planes, instead of points, complexity problems can be overcome and more efficient solutions to the scan registration problem can be achieved. Planar segments are less in number than points, providing an advantage in terms of computational cost and memory consumption. Furthermore, the plane parameters, such as the normal vector and the offset provide useful information about the environment the robot moves.

The problem of estimating the robot's pose using planar segments can be formulated as follows:

If the robot moves from the frame F_1 to the frame F_2 and observes the coordinates of the same physical point as p_1 and p_2 respectively, these coordinates are related by the equation

$$p_1 = {}^{1}_{2}R\, p_2 + {}^{1}_{2}t \qquad (3.5)$$

Where ${}^{1}_{2}R$ and ${}^{1}_{2}t$ are the rotation matrix and translation vector from F_1 to F_2, respectively. More information can be found in [29].

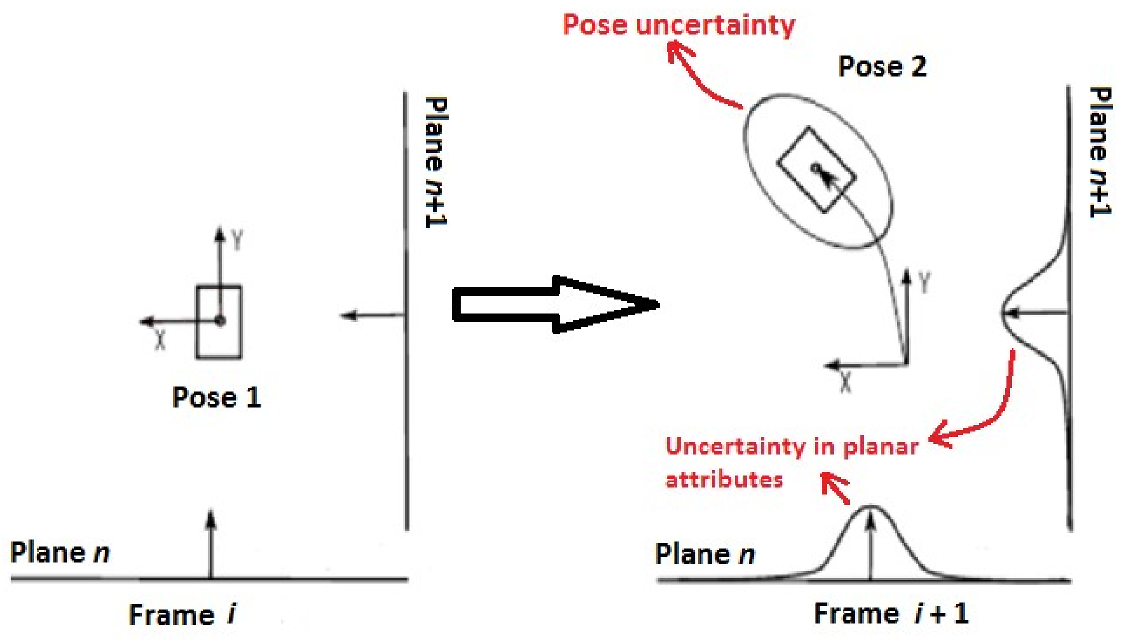
Now we wish to extend this equation for planes. Let's assume that sets of planar segments P_1 and P_2 are extracted from F_1 and F_2 and the planes $P_{1,i}$ and $P_{2,i}$ correspond to the same physical plane. Then using equation (2.8) that describes a plane, equation (3.5) becomes

$$\vec{n}_{1,i} = {}^{1}_{2}R\vec{n}_{2,i} \qquad (3.6)$$

$$\vec{n}_{1,i}\,{}^{1}_{2}t = d_{1,i} - d_{2,i} \qquad (3.7)$$

The problem consists of estimating ${}^{1}_{2}R$ and ${}^{1}_{2}t$.

The main goal of this topic is to modify the 3D NDT algorithm so that it can support the use of planar segments in order to estimate the above rotation and translation parameters. The uncertainty of the robot's pose estimation will produce an uncertainty in the attributes of the planes extracted from the reference scan. This error in the estimation of the robot translation and rotation is introduced by either the error of odometry measurements, if used, or by certain assumptions that can be made for the robot movement (e.g., distance between two consecutive scans should not be greater than 5 meters).



(a) Plane detection in scan *i* (b) Plane detection in scan *i* + 1

Fig. 3.2: Simplified 2D scan registration using plane uncertainty. (a) Initial pose of the robot in frame *i* with two planes detected and (b) Pose of the robot in the next frame. The ellipsoid around the robot models the uncertainty of the pose. This uncertainty implies the uncertainty of the plane features which is modeled as a normal distribution.

The main idea of the NDT algorithm is adopted here, by assigning normal distributions to approximate the uncertainties of the plane parameters. In particular, a 4-dimensional multivariate Gaussian can be employed to describe the uncertainty in the three coordinates of the normal vector and the offset of the plane, which is produced from the robot rotation and translation error, respectively, as depicted in Fig. 3.2. [30] gives detailed information about multivariate normal distributions. The procedure followed for the estimation of R and t is similar to the one followed in the point based 3D NDT. Initially the planar segments P_2 of the current scan are transformed back to the reference scan using odometry information (the notation here suggests that the current scan is the second one and the reference scan the first one). Then checks are held between the transformed segments P_2^* and the reference scan segments P_1. Particularly, the plane parameters of each transformed segment $P_{2,i}^*$ are compared to all the segments P_1 of the reference scan. The probability that two examined planes are similar (i.e. have similar normal vectors and offset values) can be calculated utilizing the gaussian distributions assigned to each plane P_1. More explicitly, in an 1-dimensional simplified case, the below calculation would be conducted:

$$p(a) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{a-\mu}{\sigma}\right)^2} \qquad (3.8)$$

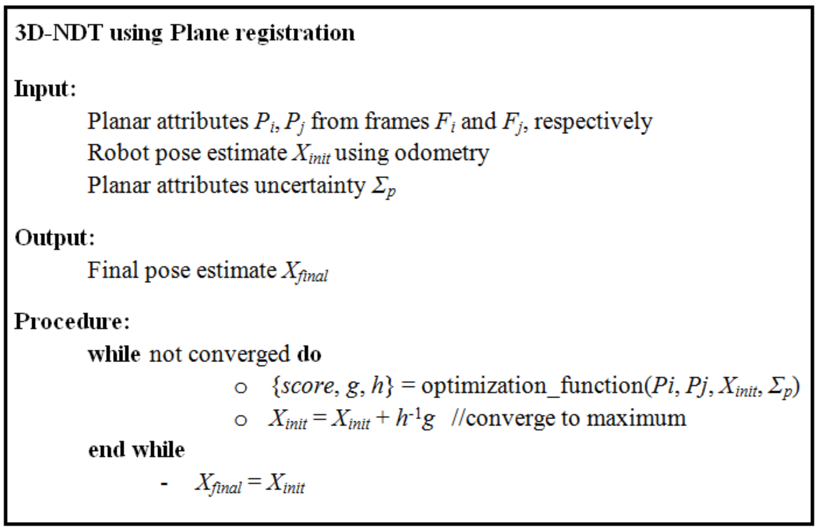
where *a* can be one of the three coordinates *x*, *y*, *z* of the normal vector or the offset value *d* of the examined plane $P_{2,i}^*$ of the current scan, μ is the mean value of the corresponding attribute in the reference scan and σ the deviation expressing the uncertainty of the specific plane attribute. The similarity between the two examined planes is more likely when the maximum of these probabilities is observed. The sum of all maximum value $\tilde{p}_{max}$ forms the score function

$$Score = \sum \tilde{p}_{max} \qquad (3.9)$$

which has to be maximized. This maximization can be achieved employing Newton's method as described for the point based 3D NDT in section 3.1, using the Hessian matrix and the gradient vector of the score function.

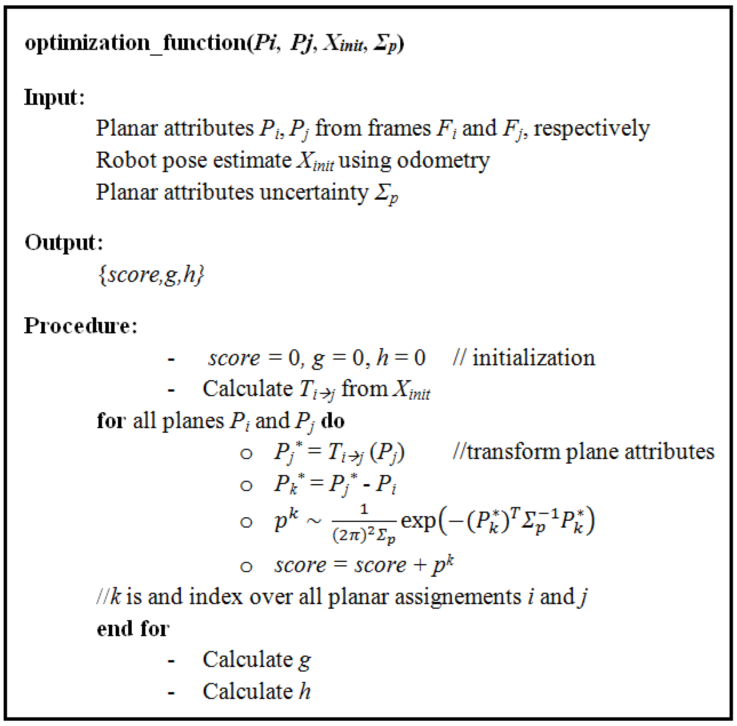
The pseudocode describing the above idea is presented in Fig. 3.3, presenting the notation first. Odometry measurements are used as an initial guess.

P_i and P_j, $j = i + 1$, denote the planes extracted from the consecutive frames F_i and F_j, respectively. With P_j^* we denote the planes extracted at F_j that are transformed with respect to F_i. Also, let Σ_p be the planar attributes uncertainty derived from the odometry uncertainty, *p* the maximum probability discussed above and $T_{i \rightarrow j}$ the transformation matrix from F_i to F_j (that is, rotation $R_{i \rightarrow j}$ and translation $t_{i \rightarrow j}$). Finally, *g* and *H* are the gradient and the Hessian required for Newton's algorithm.



```
3D-NDT using Plane registration

Input:
        Planar attributes $P_i$, $P_j$ from frames $F_i$ and $F_j$, respectively
        Robot pose estimate $X_{init}$ using odometry
        Planar attributes uncertainty $\Sigma_p$

Output:
        Final pose estimate $X_{final}$

Procedure:
        while not converged do
                o   $\{score, g, h\}$ = optimization_function($Pi, Pj, X_{init}, \Sigma_p$)
                o   $X_{init} = X_{init} + h^{-1}g$   //converge to maximum
        end while
            -   $X_{final} = X_{init}$
```

Figure 3.3(a): pseudocode of plane based 3D-NDT using odometry



```
optimization_function($P_i$, $P_j$, $X_{init}$, $\Sigma_p$)

Input:
        Planar attributes $P_i$, $P_j$ from frames $F_i$ and $F_j$, respectively
        Robot pose estimate $X_{init}$ using odometry
        Planar attributes uncertainty $\Sigma_p$

Output:
        $\{score, g, h\}$

Procedure:
                -   $score = 0, g = 0, h = 0$    // initialization
                -   Calculate $T_{i \to j}$ from $X_{init}$
        for all planes $P_i$ and $P_j$ do
                        o   $P_j^* = T_{i \to j}(P_j)$       //transform plane attributes
                        o   $P_k^* = P_j^* - P_i$
                        o   $p^k \sim \frac{1}{(2\pi)^2 \Sigma_p} \exp\left(-(P_k^*)^T \Sigma_p^{-1} P_k^*\right)$
                        o   $score = score + p^k$
        //k is and index over all planar assignements i and j
        end for
                -   Calculate $g$
                -   Calculate $h$
```

Figure 3.3(b): Optimization procedure of the 3D-NDT

The gradient and Hessian entries are derived as follows:

$$g_i = \frac{\delta_{score}}{\delta_{x_i}} = \sum_{k=1}^{n} -(P_k^*)^T \Sigma_p^{-1} J_i \exp\left(-(P_k^*)^T \Sigma_p^{-1} P_k^*\right) \quad (3.10)$$

$$h_{ij} = \sum_{k=1}^{n} \exp\left(-(P_k^*)^T \Sigma_p^{-1} P_k^*\right) \left[\left((P_k^*)^T \Sigma_p^{-1} J_i \right) \left((P_k^*)^T \Sigma_p^{-1} J_i \right) + (P_k^*)^T \Sigma_p^{-1} \frac{\delta^2 p_k^*}{\delta_{x_i} \delta_{x_j}} + (J_i)^T \Sigma_p J_i \right] \quad (3.11)$$

where J_i is the ith entry of the Jacobian matrix [34].

3.2.1 Correspondences

The key difference between the algorithm presented above and the algorithm that consists the main goal of this topic is the initial transformation of the planes from the current scan to the reference one. Unlike the approach presented, there will not be used any odometry information, since the deviation of its measurements from the real robot movement can be significantly large. Instead, an attempt to establish correspondences between planes of two consecutive scans will be made. The idea is to find characteristics of the planes of each scan that identify each one of them (or groups of them) uniquely. Using these features, we will try to determine approximately the plane (or the planes) of the reference scan that corresponds to each plane of the current scan; that is, refer to the same physical plane in the real world.

Given an average of N planes per view, there are (N + 1)! possible correspondences, if we include the case when a plane in one view is not present in the other. Non-parallel planes' correspondences have rotation information and parallel planes' correspondences have only translation information. Due to the high number of possible correspondences that can be determined, the need to reduce the search space arises. Several properties of the planar segments could be exploited for that reason, such as the number of points of each plane or the angle between the normal vectors of the planes (that can be examined using the dot product). Such attributes and more can be found in [5] where the authors use the MUMC (minimally uncertain maximal consensus) algorithm to extract planar segments and find correspondences that reduce the uncertainty volume of pose estimate.

In this Section we introduce an approach for the problem of establishing correspondences between planes of consecutive scans. More specifically, we create a fully connected graph ([31] and [32]) for each scan where each node corresponds to a single planar segment. Such a graph can be seen in Fig. 3.4. Each node contains information about certain attributes of the plane it represents, such as the number of points it consists of, its normal vector and its centroid, the eigenvalues of the covariance matrix and its mean square error. If the graph contains N nodes, then it will contain $N(N-1)/2$ edges connecting them. Each edge contains information about the relations between the planes of the nodes that it connects. These relations are chosen to be the perpendicular distance between the two planes, the angle between their normal vectors and the distance of their centroids. Fig. 3.5 shows the information contained in the graph. Using the above idea, the problem of establishing correspondences is diminished to the problem of matching similar subgraphs between the graphs of two consecutive scans. This procedure is carried out as follows. After the graphs for two consecutive scans have been built, they are cross checked to determine which plane of the first graph should correspond to which of the second graph. This is done in a probabilistic manner. That is, all possible pairs of the planes between the two graphs are formed and they are given the probability of corresponding to the same physical plane. A pair of two planes i and j that correspond to the same physical plane should have the highest probability. This probability is computed according to the similarity of the nodes of the two planes and of their edges.

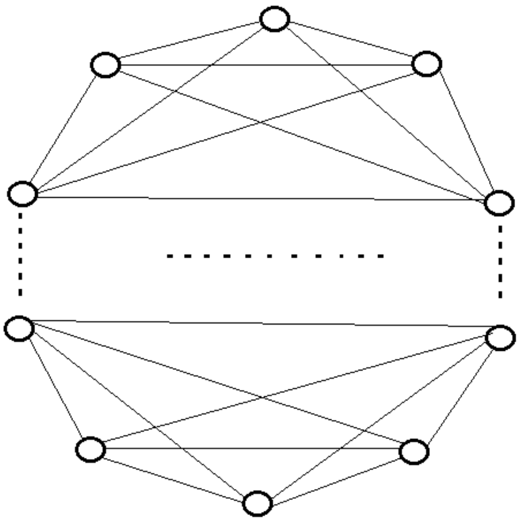


Figure 3.4: A fully connected graph of planes

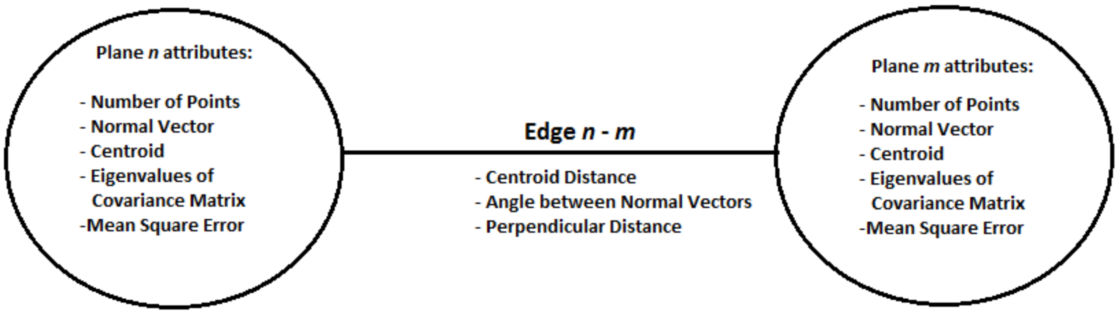
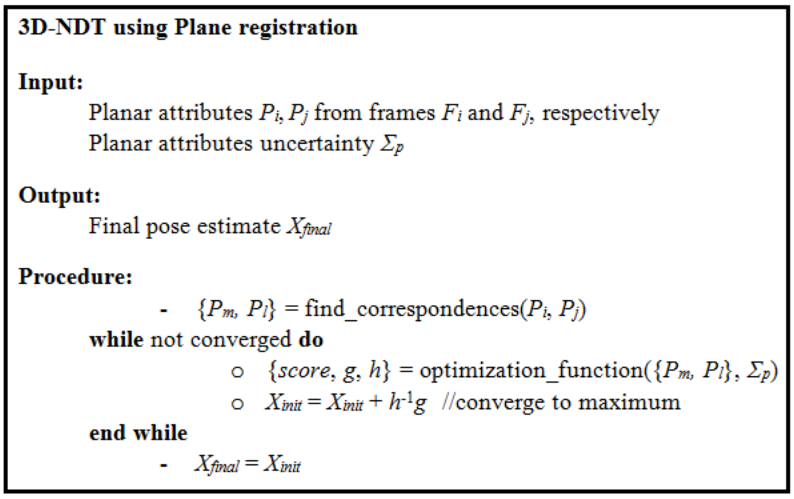


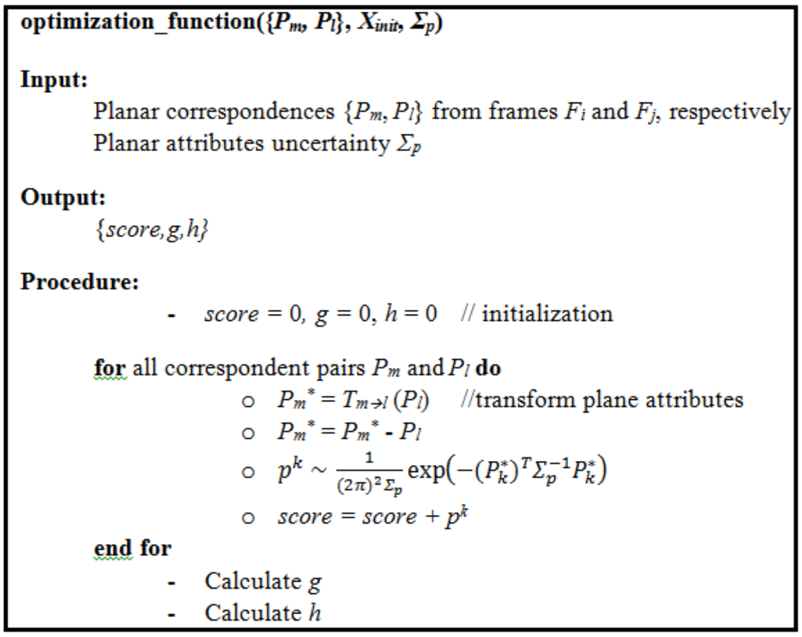
Figure 3.5: Information stored in each node and each edge of the graph

Since no odometry information is used, the initial guess X_{init} used in the pseudocode of Fig. 3.3 is set to zero and no initial transformation T is calculated. Furthermore, a procedure that establishes the necessary plane correspondences must be added, so that the optimization function uses only the correspondent planes (and not the whole amount) to calculate the optimal translation and rotation. The new pseudocode is shown in Fig. 3.6. More specifically, Fig. 3.6(a) shows the main body, Fig. 3.6(b) shows the new updated optimization function and Fig. 3.6(c) shows the pseudocode for the function that establishes plane correspondences. The optimization function that employs Newton's algorithm tries to align the correspondent planes through the score function. In that way, parallel correspondent planes between consecutive scans will determine the translation of the robot through their offset, whereas non-parallel ones will determine the rotation through their normal vector. P_i denotes the set of planes detected from the frame (scan) F_i. The notation $\{P_m, P_l\}$ denotes the set of pairs of the plane correspondences, which is the output of the corresponding procedure. In Fig. 3.6(c) G_i and G_j denote the graphs of the two frames F_i and F_j, respectively. Moreover, n_i, n_j denote the nodes of the graphs and $e_{n,i}$ is the set of edges of the n_i node. Finally, we use the notation $p_{i,j}$ for the probability of two nodes i and j to be correspondent.



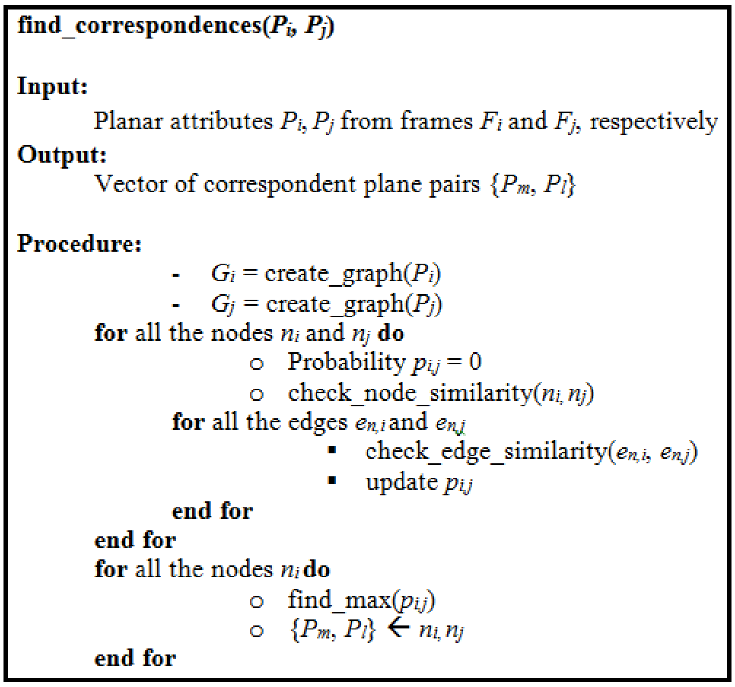
```
3D-NDT using Plane registration

Input:
        Planar attributes $P_i$, $P_j$ from frames $F_i$ and $F_j$, respectively
        Planar attributes uncertainty $\Sigma_p$

Output:
        Final pose estimate $X_{final}$

Procedure:
                - $\{P_m, P_l\}$ = find_correspondences($P_i$, $P_j$)
        while not converged do
                        o $\{score, g, h\}$ = optimization_function($\{P_m, P_l\}$, $\Sigma_p$)
                        o $X_{init} = X_{init} + h^{-1}g$   //converge to maximum
        end while
                - $X_{final} = X_{init}$
```

Figure 3.6 (a): pseudocode of plane based 3D-NDT without any odometry information



```
optimization_function($\{P_m, P_l\}$, $X_{init}$, $\Sigma_p$)

Input:
        Planar correspondences $\{P_m, P_l\}$ from frames $F_i$ and $F_j$, respectively
        Planar attributes uncertainty $\Sigma_p$

Output:
        $\{score, g, h\}$

Procedure:
                - $score = 0,\ g = 0,\ h = 0$   // initialization

        for all correspondent pairs $P_m$ and $P_l$ do
                        o $P_m^* = T_{m \to l}(P_l)$   //transform plane attributes
                        o $P_m^* = P_m^* - P_l$
                        o $p^k \sim \frac{1}{(2\pi)^2 \Sigma_p} \exp\left(-(P_k^*)^T \Sigma_p^{-1} P_k^*\right)$
                        o $score = score + p^k$
        end for
                - Calculate $g$
                - Calculate $h$
```

Figure 3.6 (b): Optimization procedure of the 3D-NDT without any odometry information



```
find_correspondences($P_i$, $P_j$)

Input:
        Planar attributes $P_i$, $P_j$ from frames $F_i$ and $F_j$, respectively
Output:
        Vector of correspondent plane pairs $\{P_m, P_l\}$

Procedure:
                -   $G_i$ = create_graph($P_i$)
                -   $G_j$ = create_graph($P_j$)
        for all the nodes $n_i$ and $n_j$ do
                        o   Probability $p_{i,j} = 0$
                        o   check_node_similarity($n_i$, $n_j$)
                for all the edges $e_{n,i}$ and $e_{n,j}$
                                ▪   check_edge_similarity($e_{n,i}$, $e_{n,j}$)
                                ▪   update $p_{i,j}$
                end for
        end for
        for all the nodes $n_i$ do
                        o   find_max($p_{i,j}$)
                        o   $\{P_m, P_l\} \leftarrow n_i, n_j$
        end for
```

Figure 3.6 (c): Pseudocode of function that finds correspondences between planes of consecutive scans

3.2.2 Experimental evaluation

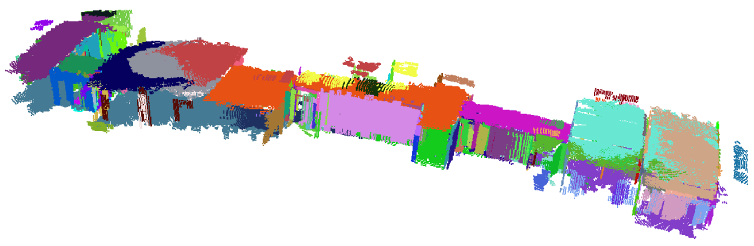
This section mainly focuses on testing the approach in real world data sets in order to evaluate its efficiency. More specifically, the algorithm will be applied in the indoor data set and the dwelling scenario. In each scan planar segments are extracted using the plane extraction algorithm presented in Chapter 2 and the approach tries to establish correspondences between consecutive scans and align them by calculating the optimal rotation and translation of the robot. Moreover, an attempt to build a 3D map of the environment will be made. In the first case, where the indoor data set is used, odometry information is available and is utilized as a correction in cases where the registration fails, in order to be able to build a consistent 3D map. A comparison will also be made with the algorithm that utilizes odometry as an initial guess in terms of time and accuracy. In the latter case, where the dwelling scenario is used, no odometry information is available. Hence, examples of successful alignments and small consistent parts of the whole map will be shown. Here it should be pointed out that the performance of the approach is highly dependent on the segmentation aglorithm that extracts planar segments. The experiments mentioned in this section were carried out on an Intel® Core i5-2500K, 3.30 GHz processor with 16GB memory.

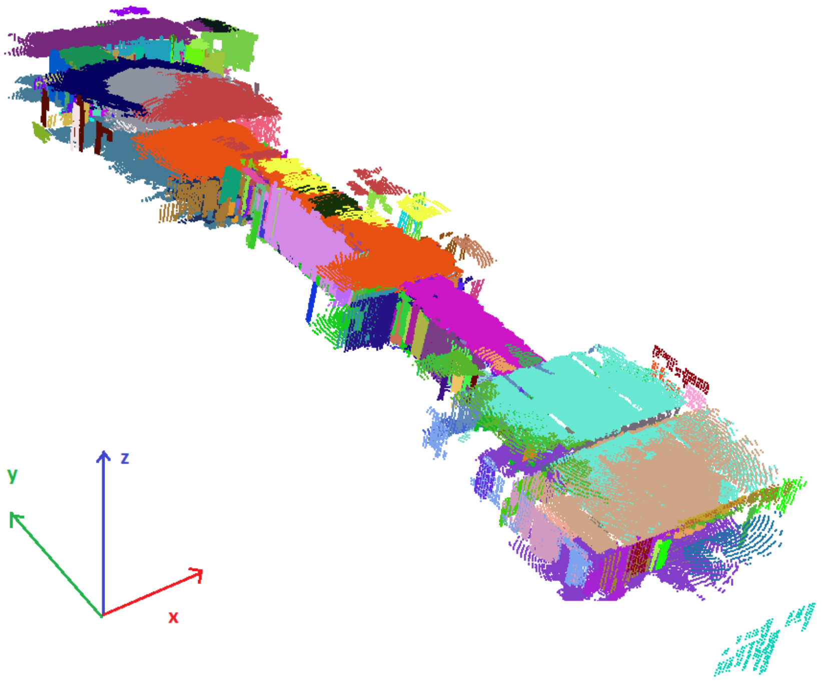
A) Indoor data set

For this evaluation, the indoor data set that was used in Section 2.2 is also used here. For this specific scenario, we imposed an assumption in translation and rotation such that the plane overlap between two consecutive scans is at least 50%. These values were taken into account in the 4-dimensional normal distribution modelling the plane uncertainty, as discussed in Section 3.2.1. The choise of these thresholds is highly dependent on the current data set used, the range of the sensor and the density of the received pointcloud. Moreover, this choise has to do with the overlap of planes that correspond to the same physical planar segment between consecutive scans, which is indispensable for the accuracy of the registration algorithm.

The algorithm was evaluated for the first 30scans of the data set, which are depicted in Fig. 3.7(a)-(c) using the points of the planes of each scan for more clarity. Planes with same colours were found as correspondent between consecutive scans. In cases that the algorithm failed to align two frames, the odometry information was used as a correction. As failures for the translation of the robot are considered values of x-, y- and z- that diverge more than 20cm from the actual values. For the rotation, we consider as a failure a deviation more than 5^o from the actual rotation values (roll, pitch, yaw).

(a) Top view


(b) Side view



(b) Side view

Figure 3.7: (a)-(c) Map of the first 30 scans of the indoor data set.

Table 3.1 depicts the percentage of successful scan registrations seperately for the translation and rotation of the robot both for the case that correspondences establishment is used without any odometry information and for the case that odometry measurements are used as an initial guess and no correspondences need to be found. Regarding the first case, we can see that for the rotation values (roll, pitch, yaw) the percentage of successful alignments is high. The failures can be attributed to the fact that the robot rotates more than the uncertainty value imposed in some cases, so the uncertainty assumptions are not satisfied. However, as one can notice in Fig. 3.7, the movement of the robot is mainly translational (it moves along a corridor), so these results for the rotational values are expected. In general, the failure percentages can be explained mainly by the symmetric space of the indoor environment. In such cases, the relations and the attributes of the planar segments are similar, so the algorithm fails to detect the correct correspondences. Fig. 3.8 depicts a simplified example of a symmetric space. For example, the plane extracted from the ceiling in frame F_i may have the same features as the plane extracted from the floor in frame F_{i+1}, infering misleading results. Moreover, to this outcome contribute the same relations between the planes in a symmetric space. In the previous example, the edges of the node that corresponds to the ceiling plane from the *i*-th scan are very similar to the edges of the node that corresponds to the floor plane from the *i*+1-th scan. The deviation in x- and z- coordinates can be attributed to the above reason.

Furthermore, the deviation of the y- value can be explained by the fact that in many cases no planes were detected perpendicular to the y-axis (it can be seen from Fig. 3.7(c) that we have a big corridor along the y-axis). For that reason, the translation in the y-axis could not be computed properly. In comparison to the case where odometry information is used as an initial guess, Table 3.1 shows that the algorithm with odometry has a better performance. This outcome is expected, since the odometry measurements used do not deviate much between two consecutive scans, leading to smaller uncertainty in the planar attributes.

	No odometry used (correspondences establishment)	**Odometry information as initial guess**
Success in x	90%	100%
Success in y	73.3%	96.7%
Success in z	73.3%	96.7%
Success in roll	96.7%	93.3%
Success in pitch	100%	100%
Success in yaw	86.7%	93.3%
Total success	63.3%	80%

Table 3.1: Percentage of successful alignments for the cases with and without odometry information

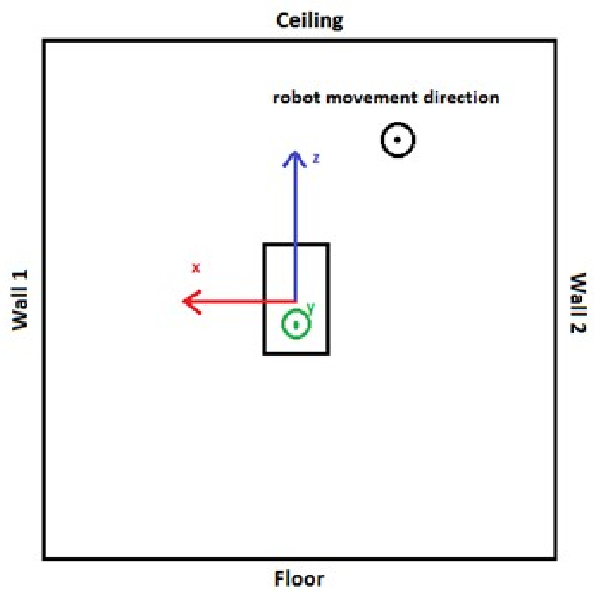


Figure 3.8: Example of symmetric space

Finally, Table 3.2 shows the pipeline times for the cases with and without odometry information. The segmentation time refers to the average time taken for extracting planar segments from each scan. The registration time refers to the average time taken for aligning two consecutive scans. The average number of planes per scan is 15. Although the registration times are similar for the two algorithms, we can state that the approach that utilizes correspondences is faster, since the optimization algorithm in that case applies only for the correspondent pairs which is a less complex procedure than doing it for all the possible pairs of planar segments.

	No odometry used (correspondences establishment)	**Odometry information as initial guess**
Segmentation Time	0.0436458	0.0436458
Registration Time	0.0170526	0.03
Total Time	0.061	0.074

Table 3.2: Runtimes for the whole registration pipeline both for the case where no odometry information is used and for the case where odometry is utilized as initial guess.

B) Dwelling scenario

For this case, the algorithm was evaluated for the dwelling scenario. In this scenario, a laser sensor produces pointcloud data for 96 scans. The uncertainty values here for the normal vector and the offset of the planar segments are derived from the assumptions we make that impose the minimum plane overlap to be at least 60% between two consecutive scans. The uncertainty value here is stricter than in the indoor data set, as the dwelling environment is more unstructured and the need of plane overlaps is bigger.

As no odometry information is available for this specific data set, a whole map of the environment could not be built, because the error from failed alignments would accumulate in the next scans. A successful alignment between two consecutive scans is determined from the consistency of the map constructed using the transformation that the algorithm calculated. Like in the indoor environment evaluation, as failures for the translation of the robot are considered values of x-, y- and z- that diverge more than

20cm from the actual values and for the rotation, we consider as a failure a deviation more than 5º from the actual rotation values (roll, pitch, yaw).

The whole amount of the 96 scans was tested and the percentage of successful alignments was approximately 60%. In the majority of the cases that the algorithm failed to perform the registration between two consecutive scans, the robot rotation is greater than uncertainty value imposed, so the uncertainty thresholds imposed are not satisfied. Unlike the evaluation for the indoor data set, a percentage for the translation and rotation values could not be given separately due to the lack of odometry information. Table 3.3 depicts the average segmentation time for each scan, the average registration time for each scan pair and the average number of planes per scan.

Segmentation time	0.126042
Registration time	0.0541053
Number of planes	21

Table 3.3: Average segmentation and registration times, average number of planes per scan

In the following, we present small consistent maps of successful alignments. The visualization is done both using the aligned pointclouds and the points of the aligned planar segments for clarity. Also, the images of the corresponding scans that are captured from the front camera of the robot are given. Fig. 3.9(a)-(g) shows the images for scans 60-66. One could notice that the overall movement in these scans consists of translations and rotations that satisfy the uncertainty restrictions. Figure 3.10(a) shows the aligned scans using the initial pointcloud and figure 3.10 (b) shows the aligned scans using the points of the aligned planar segments. Fig. 3.11(a)-(d) and 3.12(a)-(d) visualize the same results for scans 86-89. It can be concluded from the raw images that mainly small translations consist the movement of the robot. Two more examples for scans 70-72 and 20-23 are depicted in Fig. 13-14and 15-16 respectively.

(a) Scan 60 (b) Scan 61 (c) Scan 62

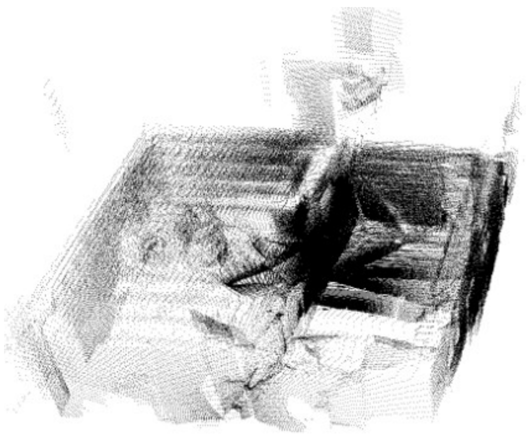
(d) Scan 63

(e) Scan 64

(f) Scan 65

(g) Scan 66

Figure 3.9 :(a)-(g) Scans 60-66 of the dwelling scenario environments as raw images


(a) Visualization of the aligned scans(60-66) using points

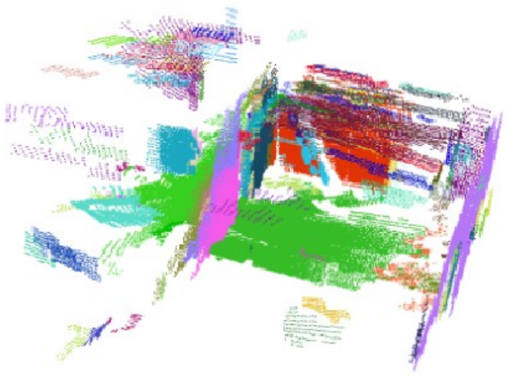
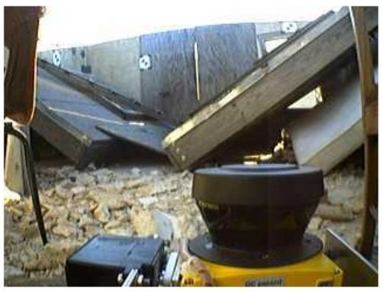
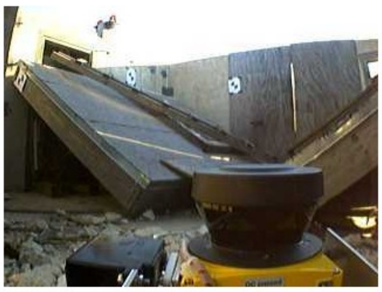

(b) Visualization of the aligned scans (60-66) using planar segments

Figure 3.10: Resulted alignment of scans 60-66. Visualization using points (a) and using points of the planar segments (b)

(a) Scan 86


(b) Scan 87


(c) Scan 88

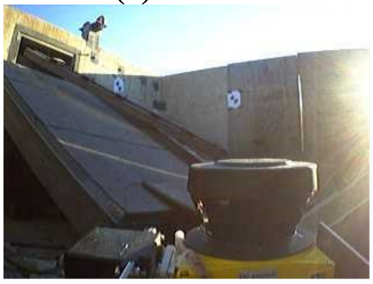
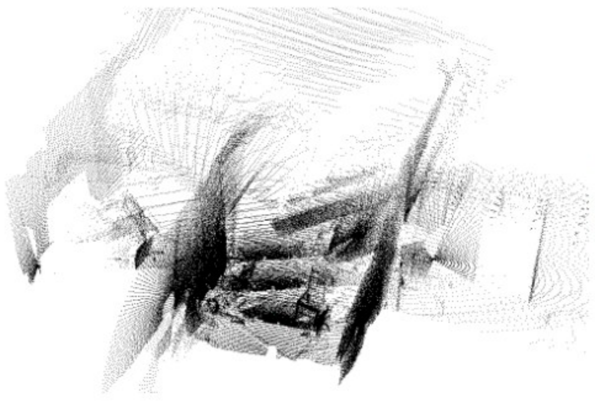
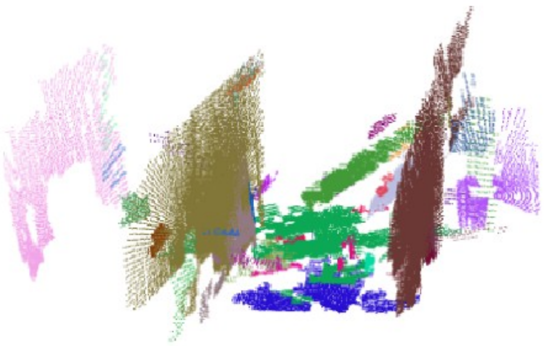
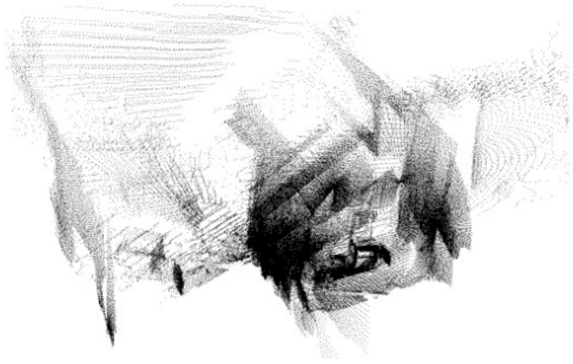

(d) Scan 89

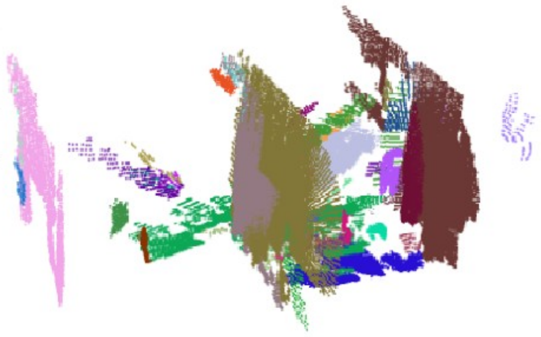
Figure 3.11 :(a)-(d) Scans 86-89 of the dwelling scenario environments as raw image


(a) Visualization of the aligned scans(86-89) using points


(b)Visualization of the aligned scans(86-89) using planar segments

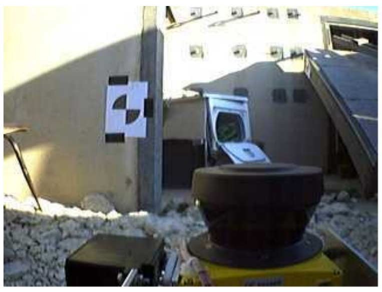


(c) Visualization of the aligned scans(86-89) using points

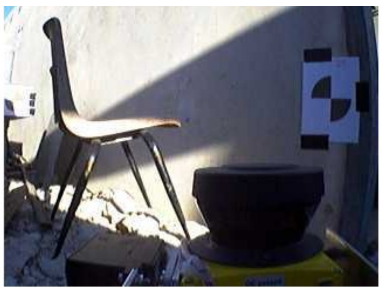


(d)Visualization of the aligned scans(86-89) using planar segments

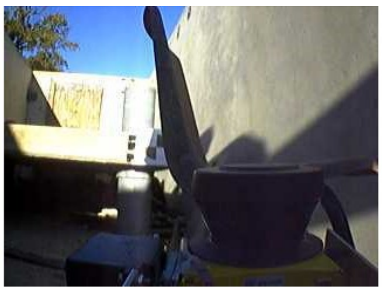
Figure 3.12: Resulted alignment of scans 86-89. Visualization using points (a) and (c) and using points of the planar segments (b) and (d)



(a) Scan 70

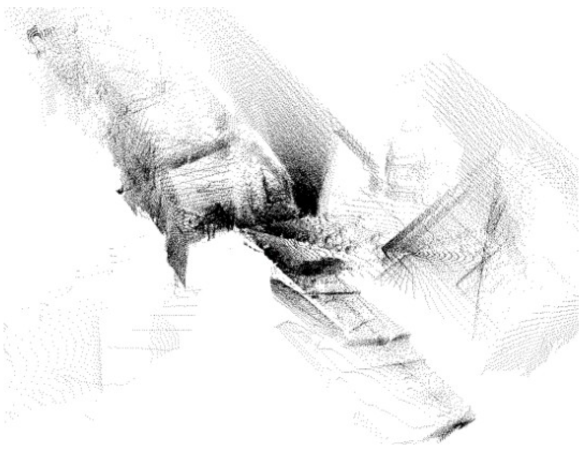


(b) Scan 71



(c) Scan 72

Figure 3.13 :(a)-(c) Scans 70-72 of the dwelling scenario environments as raw images


(a) Visualization of the aligned scans(70-72) using points

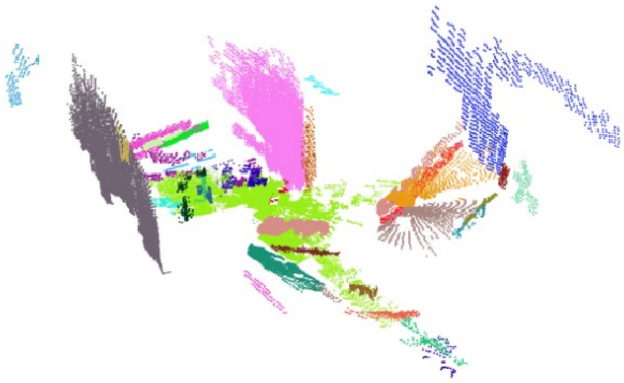
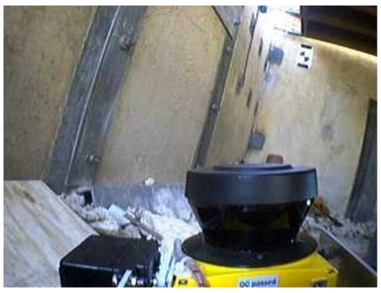
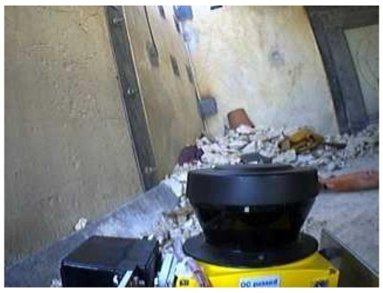
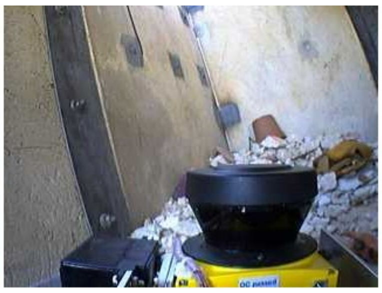

(b)Visualization of the aligned scans(70-72) using planar segments

Figure 3.14: Resulted alignment of scans 70-72. Visualization using points (a) and using points of the planar segments (b)


(a) Scan 20


(b) Scan 21


(c) Scan 22

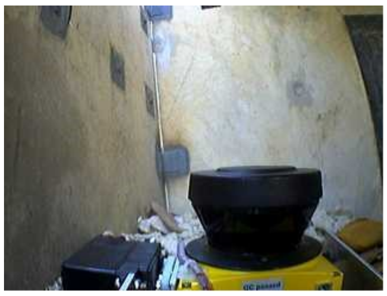
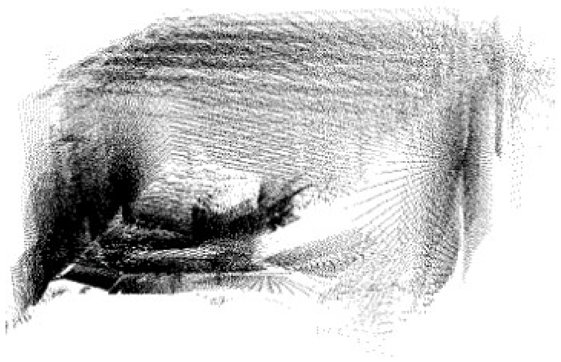

(d) Scan 23

Figure 3.15 :(a)-(d) Scans 20-23 of the dwelling scenario environments as raw images


(a) Visualization of the aligned scans(20-23) using points

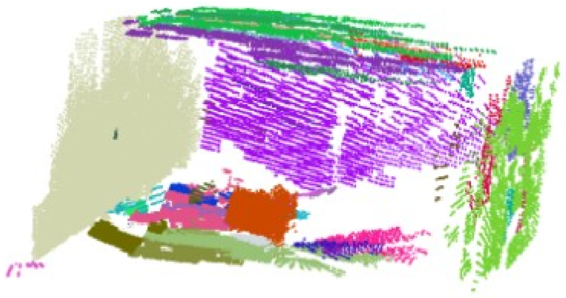

(b) Visualization of the aligned scans (20-23) using planar segments

Figure 3.16: Resulted alignment of scans 20-23. Visualization using points (a) and using points of the planar segments (b)

Fig. 3.17 and 3.18 below show the alignment of two consecutive scans for two cases. Fig. 3.17 refers to scans 4-5, where the transformation was calculated to be:

{x, y, z} = {0.287011, -0.500638, 0.0662119}m

{roll, pitch, yaw} = {-0.0243527, 0.019131, -0.663267}rad

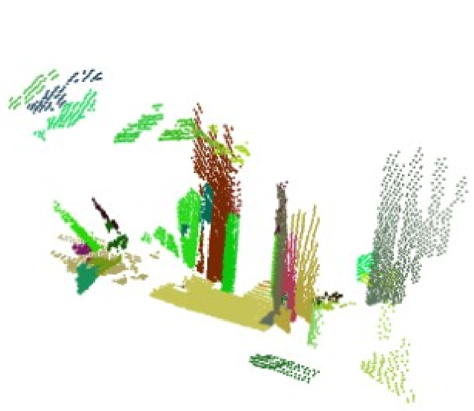
and Fig. 3.18 refers to scans 47-48, where the transformation was calculated to be:

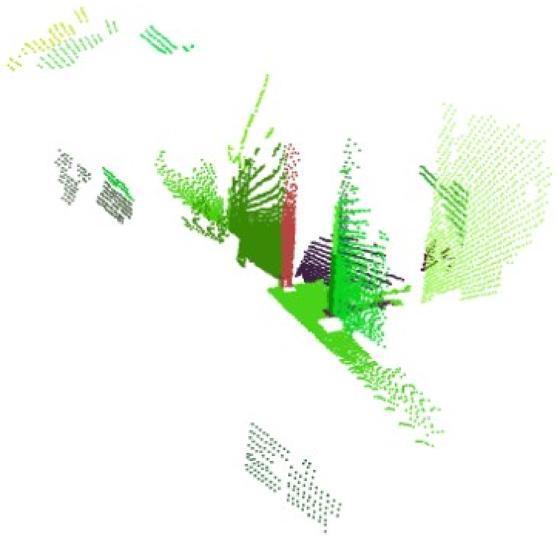
{x, y, z} = {0.887781, 0.0935841, -0.248495}m

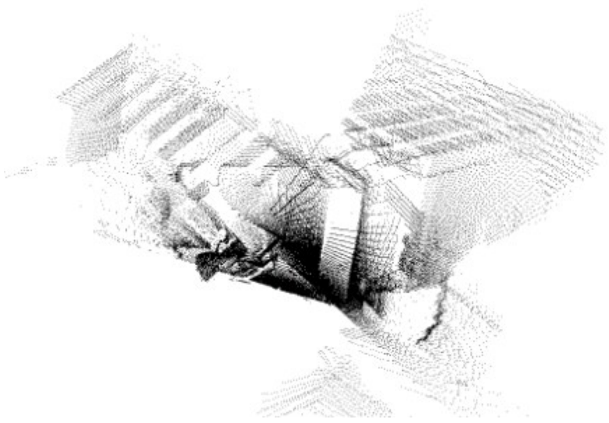
{roll, pitch, yaw} = {-0.00166798, 0.173416, 0.0452187}rad

(a) Scan 4 - image

(b) Scan 5 - image


(c) Scan 4 - planar segments


(d) Scan 5 - planar segments


(e) Aligned scans 4-5 - Visualization using points

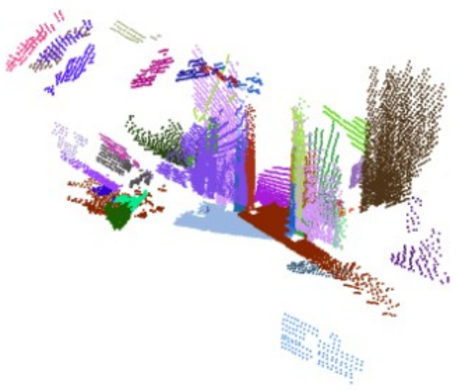
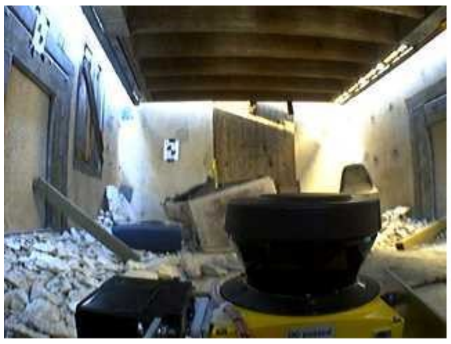
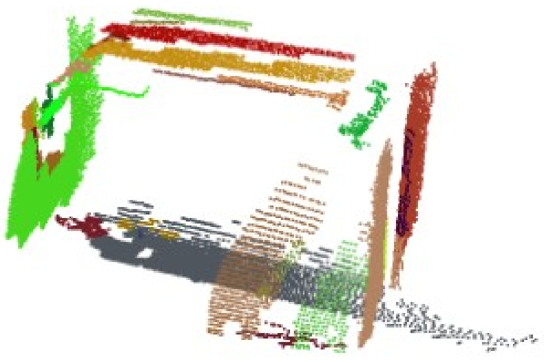
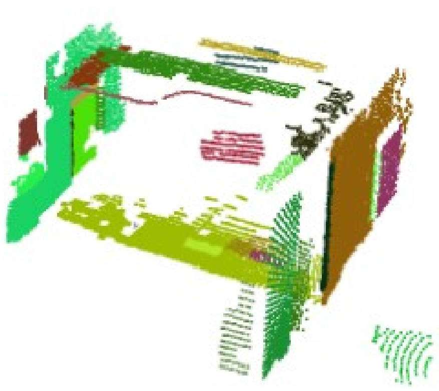

(f) Aligned scans 4-5-Visualization using planar segments

Figure 3.17: (a),(b) images of scans 4 and 5, (c),(d) planar segments of scans 4 and 5. Visualization of the alignment using points (e) and planar segments (f)


(a) Scan 47 - image

(b) Scan 48 - image


(c) Scan 47 - planar segments


(d) Scan 48 - planar segments

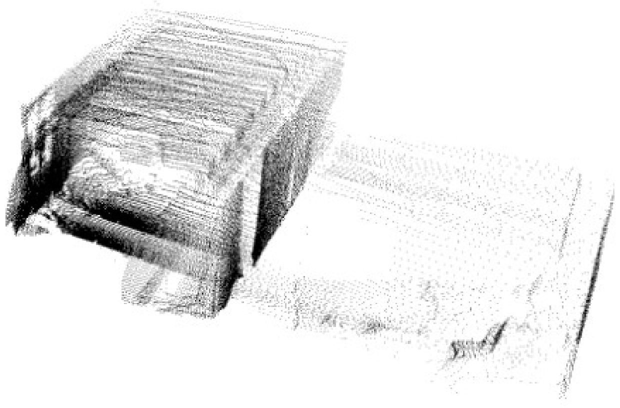
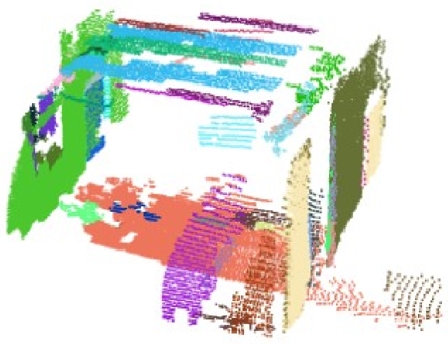


(e) Aligned scans 47-48 - Visualization using points (f) Aligned scans 47-48-Visualization using planar segments

Figure 3.18: (a),(b) images of scans 47 and 48, (c),(d) planar segments of scans 47 and 48. Visualization of the alignment using points (e) and planar segments (f)

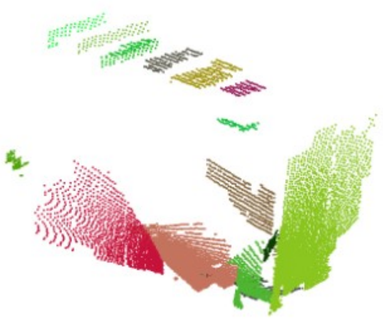
The failed alignments can be attributed mainly to the big rotation of the robot in some cases and does not satisfy the corrseponding uncertainty value we impose. However, in some cases even if the assumptions are satisfied, the algorithm fails to compute the correct transformation. In these cases the segmentation algorithm does not detect planes that can bound the movement along one axis, so the translation cannot be properly calculated. This can be mainly explained by the small pointcloud density in some regions or the complexity of the environment. Finally, there is also a small percentage of cases where the algorithm could not estabilsh the appropriate correspondences between planes of consecutive scans due to symmetric space (as discussed for the indoor data set) and due to the fact that no sufficient plane overlaps exist.

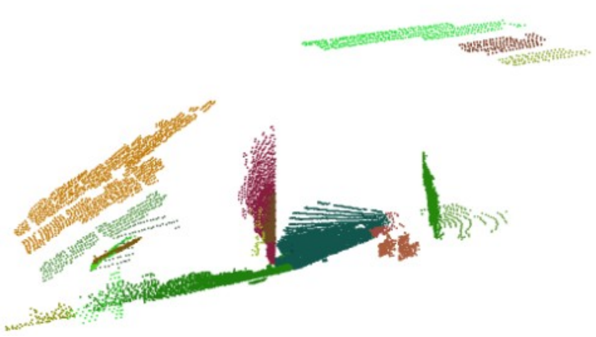
Fig. 3.19 shows an example of failed alignment due to big rotation of the robot (scans 6-7). The images in Fig. 3.19(a)-(b) show that the rotation is almost 90^o. Fig. 3.19(c)-(d) show the planar segments from these scans and Fig. 3.19(e)-(f) depicts the failed alignment.

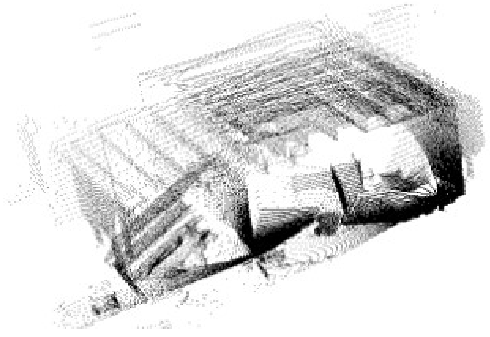
Fig. 3.20 shows an example of failed alignment due to the segmentation algorithm. Insufficient number of planes perpendicular to the axis along which the robots moves (x-axis) was detected, so the optimization algorithm could not calculate the optimal translation along this axis.

(a) Scan 6 - image

(b) Scan 7 - image


(c) Scan 6 - planar segments


(d) Scan 7 - planar segments


(e) Failed alignment of scans 6-7 – Visualization using points

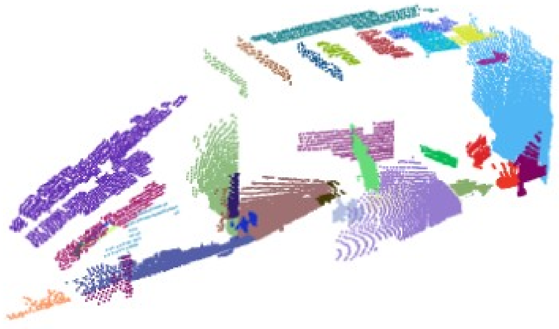
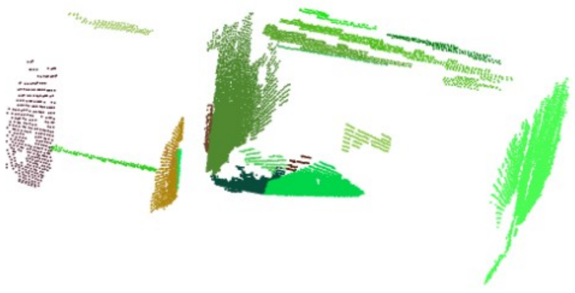
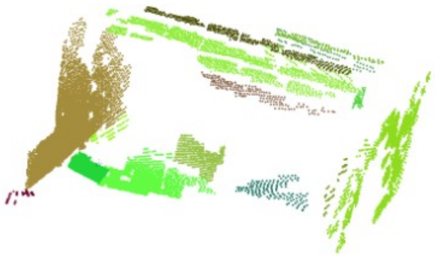

(f) Failed alignment of scans 6-7 – Visualization using planar segments

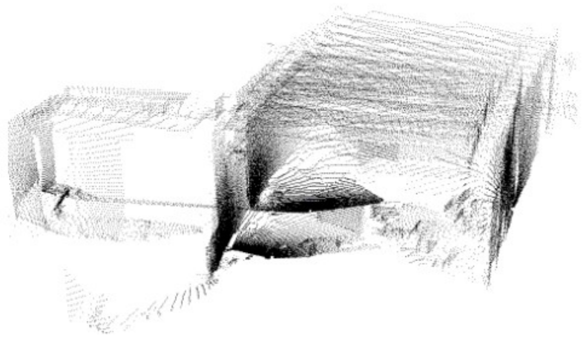
Figure 3.19: (a),(b) images of scans 6 and 7, (c),(d) planar segments of scans 6 and 7. Visualization of the failed alignment using points (e) and planar segments (f)

(a) Scan 19 - image

(b) Scan 20 - image


(c) Scan 19 - planar segments


(d) Scan 20 - planar segments


(e) Failed alignment of scans 19-20 - Visualization using points

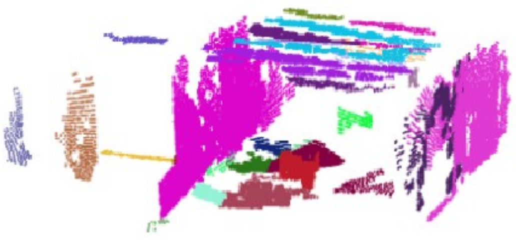

(f) Failed alignment of scans 19-20- Visualization using planar segments

Figure 3.20: (a),(b) images of scans 19 and 20, (c),(d) planar segments of scans 19 and 20. Visualization of the failed alignment using points (e) and planar segments (f)

Summarizing, in several cases of the above evaluation, the proposed algorithm did not perform successfully and failed to calculate the correct transformation of the robot. As already discussed, this can be attributed mainly to the following drawbacks of the overall approach:

- Segmentation algorithm accuracy
- Symmetry of the environment that causes misleading plane correspondences
- Assumptions about the movement of the robot are not satisfied
- Corridor effect: lack of planar segments to bound the movement of the robot

4 Conclusion and future work

In this thesis the aspect of 3D plane registration is examined. We present an approach that considers the use of planar segments instead of points for scan alignment without any use of odometry information in order to calculate the transformation of the robot between two consecutive scans and build a consistent 3D map. The uncertainty in the pose of the robot that is derived by certain assumptions is transformed to planar attributes uncertainty which is the main idea behind this framework. This uncertainty is modeled employing an extension to plane-based of 3D-NDT algorithm which utilizes Gaussian distribution functions. The approach was evaluated both in an indoor and an outdoor environment to evaluate its accuracy. Despite of the disadvantages of the algorithm, in both scenarios the success rate was over 50% and low computational times were achieved making it suitable for online applications. In addition, a plane extraction algorithm is introduced which exploits the advantages of the octree data structure (e.g. multi-resolutional representation of the environment) and is highly adaptive to different scenarios of 3D pointclouds. The experimental evaluation that was conducted showed that the use of this specific tree structure allows for low computational times, especially in the initialization procedure, making the algorithm computationally efficient.

As future work, the further development of the algorithm that establishes correspondences could be considered. More explicitly, the topological properties of a fully connected graph could be exploited at a higher level, so that the algorithm would be more accurate and robust to the noise imposed by laser sensors and the deficiencies of the segmentation algorithm. Moreover, the achievement of more robustness could result in greater values in the initial assumptions, both in translation and rotation, making the algorithm suitable for more demanding scenarios. Finally, instead of a fully connected graph, other less complicated types could be utilized that would conceive better the structure and the relations between the planar segments.

List of Figures

List of Tables

Bibliography

[1] P. Besl und N. McKay, „A method for registration of 3-D shapes,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Bd. 14, Nr. 2, pp. 239 - 256, 1992.

[2] M. Magnusson und A. Lilienthal, „Scan Registration for Autonomous Mining Vehicles Using 3D-NDT,“ *Journal of Field Robotics,* Bd. 24, Nr. 10, pp. 803 - 827, 2007.

[3] A. Harati und R. Siegwart, Orthogonal 3D-SLAM for indoor Environments Using Right Angle Corners, 2007.

[4] V. Nguyen, A. Harati und R. Siegwart, „A lightweight SLAM algorithm using orthogonal planes for indoor mobile robotics,“ *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. IROS 2007,* 2007.

[5] K. Pathak, A. Birk, N. Vaskevicius und J. Poppinga, „Fast registration based on noisy planes with unknown correspondences for 3-D mapping,“ *IEEE transactions on Robotic,* Bd. 26, Nr. 3, pp. 424 - 441, June 2010.

[6] D. Fischer und P. Kohlhepp, „3D geometry reconstruction from multiple segmented surface descriptions using neuro-fuzzy similarity measures,“ *Journal of Intelligent and Robotic Systems,* Bd. 29, Nr. 4, pp. 389 - 431, 2000.

[7] J. Xiao, B. Adler und H. Zhang, „3D Point Cloud Registration Based on Planar Surfaces,“ *2012 IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI),* Hamburg, 2012.

[8] D. Joyce, Euclid's Elements, Book I, Definition 7, Clark University, 1996.

[9] J. Poppinga, N. Vaskevicius, N. Birk und K. Pathak, „Fast plane detection and polygonalization in noisy 3D range images,“ *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008.* Bremen, 2008.

[10] J. Xiao, J. Zhang, J. Zhang, H. Zhang und H. Hildre, „Fast plane detection for slam from noisry range images in both structured and unstructured environments,“ *2011 International Conference on Mechatronics and Automation (ICMA),* Hamburg, 2011.

[11] G. Korn und T. M. Korn, Mathematical Handbook for Scientists and Engineers: Definitions, Theorems and Formulas for Reference and Review, 2nd revised Hrsg., Dover Publications, 2000.

[12] L. Wasserman, All of Statistics: A Concise Course in Statistical Inference, 2004.

[13] K. Wurm, A. Hornung, M. Bennewitz und C. B. W. Stachniss, „OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems,“ *Proc. of the ICRA 2010 workshop on best practise in 3D perception and modeling for mobile manipultation,* 2010.

[14] M. Donald, „High-Speed image generation of complex solid objects using octree encoding,“ in *U.S. Patent No. 4,694,404,* Washington, DC, 1987.

[15] F. Olver, D. Lozier, R. Boisvert und C. Clark, „Handbook of Mathematical Functions,“ 2010. [Online]. Available: http://dlmf.nist.gov/19.33.

[16] A. Nuechter und K. Lingermann, „6D SLAM-3D mapping outdoor environments,“ *Journal of Field Robotics* , Bd. 24, Nr. 8-9, pp. 699 - 722, 2007.

[17] D. Bormann, J. Elseberg, K. Lingermann, A. Nuechter und J. Hertzberg, „Globally consistent 3D mapping with scan matching,“ *Robotics and Autonomous Systems,* Bd. 56, Nr. 2, pp. 130 - 142, 2008.

[18] J. Weingarten und R. Siegwart, „3D SLAM using planar segments,“ *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems,* Lausanne, 2006.

[19] F. Lu und E. Milios, „Robot pose estimation in unknown environments by matching 2D range scans,“ *1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994. Proceedings CVRP '94,* Canada, 1994.

[20] P. Biber und W. Strasser, „The normal distributions transform: A new approach to laser scan matching,“ *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings.* 2003.

[21] M. Magnusson, „The Three-Dimensional Normal-Distributions Transform - an efficient representation for registration, surface analysis, and loop detection,“ Oerebro University, 2009.

[22] G. G., „SLAM scan-matching using ICP,“ 2007. [Online]. Available: http://www.informatik.uni-freiburg.de/~grisetti/teaching/ls-slam/lectures/pdf/ls-slam-07-icp.pdf.

[23] K. Arun, T. Huang und S. Blostein, „Least Square Fitting of Two 3-D Point Sets,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Nr. 5, pp. 698 - 700, 1987.

[24] B. Horn, „Closed-form solution of absolute orientation using unit quaternions,“ *Journal of the Optical Society of America A,* Bd. 4, Nr. 4, pp. 629 - 642, 1987.

[25] M. Walker, L. Shao und R. Volz, „Estimating 3-D location parameters using dual number quaternions,“ *CVGIP: Image Understanding* , Bd. 54, pp. 358 - 367, 1991.

[26] R. I. Jennrich und S. Robinson, „A Newton-Raphson algorithm for maximum likelihood factor analysis,“ *Psychometrica* , Bd. 34, Nr. 1, pp. 111 - 123, 1969.

[27] M. Greenspan und M. Yurick, „Approximate k-d tree search for efficient ICP, “*Proceedings of the fourth International Conference on3-D Digital Imaging and Modeling, 2003. 3DIM 2003.,* Canada, 2003.

[28] M. Magnusson, A. Nuechter, C. Lorken, A. Lilienthal und J. Hertzberg, „Evaluation of 3D registration reliability and speed - A comparison of ICP and NDT,“ *IEEE International Conference on Robotics and Automation, 2009, ICRA '09,* Orebro, Sweden, 2009.

[29] B. Siciliano und L. Sciavicco, Robotics: modelling, planning and control, Springer, 2011.

[30] G. Allan, in *An Intermediate Course in Probability*, Springer, 2009.

[31] T. Pirnot, Mathematics All Around, Addison Wesley, 2000, p. 154.

[32] D. Gries und F. Schneider, A Logical Approach to Discrete Math, Springer-Verlag, 1993, p. 436.

[33] M. Magnusson, L. Achim und T. Duckett, „Scan registration for autonomous mining vehicles using 3D-NDT,“ *Journal of Field Robotics*, pp. 803-827, 2007.

[34] S. Perreault, „Octree C++ Class Template,“ April 2007. [Online]. Available: http://nomis80.org/code/doc/classOctree.html.