

**Plane Extraction and Map Building Using a Kinect Equipped Mobile Robot**

Ahmad Kamal Nasir, Christof Hille, Hubert Roth  
Lehrstuhl für Regelungs- und Steuerungstechnik (RST), Department Elektrotechnik und Informatik, Fakultät IV Universität Siegen, 57068, Siegen Germany, (email: \{ahmad.nasir, christof.hille, hubert.roth\}@uni-siegen.de)

Abstract—This research work describes a method to create a geometric feature (3D plane) based map using a differential drive mobile robot in an indoor environment. Two algorithms namely Hough Transformation and Random Sample Consensus are used separately to extract multiple 3D planes from the point cloud data and the results are compared. An Octree based data structure is used to create and store the generated planner map of the environment. Furthermore a simple error analysis on the estimated parameters of the plane from both RANSAC and Hough Transformation algorithm in a test environment is presented.

I. INTRODUCTION

3D Map building is fundamental to the autonomous navigation of the mobile robots in real world environment. Furthermore it could help mobile robots to reason about environment. State of the art mobile robots use 3D range scanning devices such as laser scanner, time of flight cameras, stereo cameras and RGB-D [21] cameras to sense the spatial environment and construct the map from acquired point clouds. Traditional computer vision solutions to construct 3D maps from multi-view videos or related images are computational resource demanding and time consuming. Geometric features such as lines and planes are prevalent into the manmade environments such as offices and factory floors. Mobile robots can use such geometric features to construct a map for collision free autonomous navigation and localization in such environments.

In this research work we demonstrate building a 3D geometric map of an office environment by using a ground mobile robot equipped with a Microsoft Kinect camera. It is an inexpensive camera which provides a color image stream and a depth image stream in an indoor environment in real time which can be very useful for dense 3D color mapping in cluttered indoor environments. Despite of the impressive acquisition rate the raw data is unsuitable for navigation and real-time 3D mapping because of the enormous amount of the data to be processed. Therefore, geometric features such as planes are extracted from the raw 3D point clouds.

To create the model of the environment several scans have to be fused. The fusing process is easy if the position of the scanner is known otherwise scan registrations have to be performed to estimate the pose of the scanner. In this research work we have not concentrated on the scan registration and a basic analysis of Kinect range measurement error is performed, but have not used to correct the measurement because of small error. We have also assumed that the mobile robot has been already localized thus an accurate mobile robot pose is available for mapping. For detailed Kinect sensor range measurement error model one could refer [22], [8].

A. Related Work

Various research works [1], [2], [3], [4] have been done until now to extract the 3D planes from the point cloud data acquired from different range sensor devices and build the 3D map of the environment. Asad [4] has proposed a mapping system for mobile robots which used height maps created from range images for path planning. Pathak [1] proposed a method for 3D mapping by a mobile robot, furthermore, his proposed method utilizes the uncertainty of the plane parameters to compute the uncertainty in the pose computed by scan registration. Weingarten et al. [3] proposed a method for plane fitting for laser range scanner data and fuses matching planes together to find a compact 3D model. Andreasson et al. [5] uses an approach which fuses both color and range information to detect 3D planes.

Apart from various mapping algorithms for mobile robots different sensors have also been used in combination with mapping algorithms to map 3D environments. Such sensors include laser scanners [19], stereo vision and monocular cameras [11] and time of flight camera [20]. 2D laser scanners are limited in use for mapping environments which contains simple geometric shapes; furthermore the obstacles which are above or below the scanned planes cannot be detected e.g. downward stairs. Where the stereo systems are dependent on lighting conditions and cannot detect planes in homogenous regions. Kinect sensor has brought acquiring colored 3D point clouds cheaper and quicker which in the past require expensive time of flight cameras. Furthermore, to acquire colored point clouds the system consisting of time of flight camera and image camera must be setup and calibrated. But Kinect combines the 3D range finding capability and the color information.

Recently most of the research work which uses Kinect camera [6], [7], [8] has focused on extracted plane segmentation because of the sparsity, measurement range limitation and occlusion of the measurements. These research works have used the color image to complement the range limitation and sparsity of the depth measurement. The intensity information can help in segmentation of the 3D point cloud data by detecting edges in the intensity images corresponding to the area of interest in the 3D point cloud.
A common approach for mapping is to align point clouds by finding rotation and translation between consecutive 3D scans [13]. Henry [12] maps the environment using ICP and SIFT features. There exist numbers of other methods which extract the 3D planes from the raw point clouds. Bommann [15] uses the Hough transform to extract the 3D plane from the raw point clouds. Triebel [16] uses expectation maximization, Gallo [17] used RANSAC to extract the planes and Pathak [18] used the split and merge techniques to detect the planes.

Our focus in this research work is to build a framework for 3D mobile robot mapping which can be used in real time for SLAM, obstacle avoidance and path planning. Semantic mapping [14] can be applied as a post processing step to group the related geometric features in the map. The paper is organized as follows; section II describes the methodology of our research work in details. Section III describes the implementation of plane extraction algorithms and mapping. Section IV discusses the result obtained from the experiment. Section V concludes our work.

II. METHODOLOGY

This research work uses the plane detection algorithms to detect the planes from the raw Kinect data and registers them using octree data structure. Since Kinect sensor acquires enormous amounts of data, 9.2 million 3D points in one sec, it is challenging to process the data in real time because of the limited amount of computation resources available on mobile robots, furthermore, raw 3D point clouds from Kinect sensor are not directly useable. Some processing is required to reduce this amount of data to extract features information present in the raw 3D point cloud. The features could be point features, line features, color segmentations and shape detections. Extracting multiple geometric features from the range data is computationally demanding and directly related to the number of parameters required to represent the geometric model to be found in the raw point clouds. In our geometric mapping approach we have used 3D planes as geometric features because a plethora of 3D planes are available in structured environments. We have tested two algorithms namely RANSAC and Hough transformation to extract the 3D planes from the raw point cloud so that we can compare the performance of real-time geometric map building from the Kinect equipped ground mobile robot

A. Data Association

Octrees are data structures which can be used to partition the three dimensional space in an efficient way. The base octree is represented by a cube or a cuboid and is called root node. Its volume can be further discretized into eight new octants, which partition the space of their parent node. Depending on the needs of the application the depth of nodes can vary. Fig. 1 depicts an octree with a root node, a node and one leaf node, where data can be stored.

![Octree with three nodes.](image)

The main advantage of this approach is that there is no need to create child nodes, where there is no data for them and it’s always possible to create a root node, if one exceeds the space given by the actual octree. Octrees offer a great flexibility to store and access 3D data. In most approaches each node represents a voxel in octree. Octrees offer a great flexibility to store and access 3D data. In most approaches each node represents a voxel in octree. In their experiments they successfully show that their geometric mapping approach we have used 3D planes as geometric model to be found in the raw point clouds. In our experiments they successfully show that their geometric model to be found in the raw point clouds. Since in this contribution the position of the plane greatly depends on the odometry of the mobile robot, which is not very accurate after a few meters of movement, it is almost impossible to decide which plane is true. Therefore each plane which is registered in a leaf node overwrites the existing one, in case there is one.

Uniquely registered planes in the octree can be easily rendered into a 3D map. This is a simple but efficient way to reduce the amount of data in the map. Another advantage of the octree structure is the possibility to implement efficient path planning and collision avoidance algorithms. Jung [32] presents one of the first approaches which use a three dimensional octree map of voxel information to plan a path for a manipulator robot. Kazakov et al. [28] propose an octree based map for large scale environments to simplify the problems of path planning and collision detection for mobile robots. In their experiments they successfully show that their suggested path planning algorithms are able to suggest a path and to correct it, if necessary.

B. Plane Extraction

1) Hough Transform

Hough transform is a well known algorithm in computer vision society to detect multiple models in the data compared to RANSAC which in its basic form assumes there is a single model present in the data. It can detect lines, planes, spheres and other parameterizable geometric objects in the input data. In spite of the robustness of the method against noisy data one drawback of this algorithm is its high computational requirement therefore many variations of the Hough transform exists to detect the desired model parameters. Apart from standard Hough transform other variations which exists are, probabilistic Hough transform, random Hough transform, adaptive probabilistic Hough transform and progressive probabilistic Hough transform. The plane equation in Hesse normal form can be defined by a point $p$ on the plane with normal vector $n$ to the plane which is at a distance $\rho$ from the origin, which is collinear to normal vector as shown in fig. 4. The normal vector or $\rho$ makes an angle $\theta$ with the z-axis and its projection in the x-y plane makes an angle $\varphi$ with the x-axis. Therefore, the equation of the plane can be defined as
The dimension of the Hough space is basically equal to the number of parameters of our model i.e. $\theta, \varphi, p$. Each plane in $\mathbb{R}^3$ corresponds to a point in the Hough space and each point in $\mathbb{R}^3$ corresponds to a surface in Hough space. This surface represents all the possible planes where the point could belong to. Therefore, the transformation of the points $p_i \subset P$ from $\mathbb{R}^2$ to Hough space will generate surfaces in Hough space. The intersection of three surfaces in Hough space results in a point in Hough space which corresponds to a plane in $\mathbb{R}^3$ on which the three points which generates the surface lies on. All points whose surfaces in Hough space intersect at a point correspond to the same plane in $\mathbb{R}^3$. 

\[ p_x \cdot \cos(\theta) \cdot \sin(\varphi) + p_x \cdot \sin(\theta) \cdot \sin(\varphi) + p_x \cdot \cos(\varphi) = p. \quad (1) \]

We have used a random Hough transform to detect the 3D planes in the raw 3D point clouds. Instead of generating surfaces for each point $p_i$ in $\mathbb{R}^3$ into Hough space, which is very time consuming we used the fact that a plane corresponds to a single point in Hough space, therefore, it is very fast to compute a plane from three random points from a small circular region and transform the estimated plane to Hough space, this results into a significant faster algorithm for real time implementation. The pseudo code of the randomized Hough transform is as follows:

\begin{algorithm}
\caption{Randomized Hough Transform Algorithm}
\end{algorithm}

The Hough discretization size of the Hough transform depends on the accuracy required and the available memory. For our implementation we have discretized the Hough space into 1cm for $p$ from 1cm to 500cm, $1^\circ$ for $\varphi$ from $-180^\circ$ to $180^\circ$ and $1^\circ$ for $\theta$ from $0^\circ$ to $180^\circ$. Using the above discretization the memory requirement for Hough space is found to be 125MB. We have found out that the predominant part of the time required by the randomized Hough transform is required to reset the Hough space, therefore the choice of discretization for plane parameters has been chosen based on the possible orientation of the planes in the input raw 3D point clouds.

2) RANSAC

A common approach to identify geometric objects in a scene of 3D points is the use of the RANSAC algorithm, which was introduced by Fischler [24]. It overcomes the weakness of a normal least square approach which can't distinguish between inliers and outliers in the measurement data. Instead of using all the available data to calculate the model parameters it checks for each data point if it should be considered as an inliner or an outlier to the model. Therefore in the end only inliers are used to derive the object parameters. In this paper the RANSAC algorithm is used to identify planes in clouds of 3D points. A basic RANSAC algorithm is used here. It takes in a first step three non-collinear, random points $P_1(x_1,y_1,z_1)$, $P_2(x_2,y_2,z_2)$ and $P_3(x_3,y_3,z_3)$ out of the point cloud and sets up a plane equation using the equation

\[
\begin{bmatrix}
    x - x_1 & y - y_1 & z - z_1 \\
    x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\
    x_3 - x_1 & y_3 - y_1 & z_3 - z_1 
\end{bmatrix} = 0.
\]

In the second step it sorts the remaining points into inliers, which have a minimum distance to the plane, and outliers, which have a greater distance to the plane than a defined threshold. The point to plane distance threshold for RANSAC is chosen to be 1 cm because the surfaces in our test environment are very flat and also we have small errors in Kinect’s range measurements. A distance between a 3D point and a plane is calculated by

\[
d = x \cos \alpha + y \cos \beta + z \cos y + p,
\]

where $d$ is the distance from the point $P(x,y,z)$ to the plane and $\cos \alpha$, $\cos \beta$, $\cos y$ and $p$ are the parameters of the plane in the hessian form. Steps one and two are repeated until a satisfying number of inliers are found or a maximum number of iterations are executed. The number of satisfying inliers is 50,000 and the maximum number of iterations are 500. The identified inliers are used to calculate the plane parameters using the hessian description of a plane which is given as:

\[
x \cos \alpha + y \cos \beta + z \cos y + p = 0,
\]

which can be rewritten as

\[
\begin{bmatrix}
x & y & z & 1
\end{bmatrix} \begin{bmatrix}
\cos \alpha & \cos \beta & \cos y & p
\end{bmatrix} = 0,
\]

where $A$ represents a matrix containing the plane parameters in each row which solves the equation for the matrix $X$ which contains the homogeneous coordinates of each point $P$ in its columns. This homogeneous equation must be solved by using least square techniques. The solution of the above mentioned over-determined system is found by using the Singular Value Decomposition (SVD) implementation of the library Sho [23].

The RANSAC algorithm is applied for a maximum of 8 iterations on the outlier point cloud from the previous iteration, but stops if the amount of remaining points drops below a threshold. This avoids that points could be part of different planes and helps the algorithm to find smaller planes in the point cloud. Much more complex RANSAC algorithms do exist like [27], [30] but since in this research work only planes are extracted and no spheres, cylinder or tori, this complexity is not needed.

Since the robot pose in the world coordinate system is known from the odometry, it is possible to register a detected plane in the world coordinate system and to construct a map.

III. EXPERIMENT

During all experiments a laptop, having an Intel® Core™2 Duo processor, running at 2.8 GHz, equipped with 8 GB RAM and running with a 64 Bit Linux, is used to acquire and to store all the sensor information needed for the different
experiments at a rate of 5 Hz. The amount of Kinect sensor data is very huge, which makes storing it to a problem where traditional HHDs are a bottle neck. Therefore the Laptop is equipped with an SSD which allows data storing at very high data rates, which helps to acquire the data at defined time stamps. Later the offline data sets are processed to extract the needed data for the different experiments.

Kinect sensor provides a horizontal and vertical field of view of 58° and 44° respectively and the angular resolution of 0.08°. Furthermore, the device can be tilted ±30°. The working depth measurement range is between 0.8m and 5m. Kinect consumes about 250mA at 12V DC. It can acquire both depth stream and color stream of full VGA resolution (640x480) at 30Hz. Park [9] has found out also that the Kinect variance of measurements error for dark objects is smaller compared to the Hokuyo UBG-04LX-F01 laser scanner. Since the color and depth cameras in Kinect are factory calibrated therefore it is now an easy task to correspond the pixel information in both cameras. One drawback of the Kinect is, it works only in indoor environments without sunlight.

To validate the approach and to compare the different plane extraction algorithms presented in chapter II, three experiments are conducted.

A. Experiment I

In the first experiment the Kinect sensor is mounted on a KUKA R16 manipulator robot and looks on a cuboid which is used as a reference object. All of its planes are orthogonal to each other and in this experiment two planes are in the field of view of the Kinect. The robot is used to change the view on the scene in defined angles. Under all changed view conditions the algorithms should be able to detect an angle around 90° between both detected planes of the reference object. The robot pose configuration is shown in fig. 2. Both Hough transform and RANSAC are supposed to identify the angle between the two planes of the reference object which correspond to the sides of the object. All together four different views of the cuboid are taken with the Kinect sensor. The robot angles of the four points clouds are listed in table I.

<table>
<thead>
<tr>
<th>Robot Axis</th>
<th>View 1</th>
<th>View 2</th>
<th>View 3</th>
<th>View 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 [°]</td>
<td>1.59</td>
<td>1.59</td>
<td>1.59</td>
<td>5.59</td>
</tr>
<tr>
<td>A2 [°]</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>A3 [°]</td>
<td>-90</td>
<td>-90</td>
<td>-90</td>
<td>-90</td>
</tr>
<tr>
<td>A4 [°]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A5 [°]</td>
<td>-50</td>
<td>-50</td>
<td>-50</td>
<td>-50</td>
</tr>
<tr>
<td>A6 [°]</td>
<td>0</td>
<td>20</td>
<td>-20</td>
<td>0</td>
</tr>
</tbody>
</table>

B. Experiment II

Here the Kinect sensor is placed on rotary indexed table which is placed on a table against a flat wall. The distance between wall and Kinect’s front wall facing side is measured to be 106 cm and the data is recorded. In a next step the Kinect is moved 50 cm further away from the wall and once more the data is recorded. From both positions we have calculated the standard deviation of the Kinect’s measurement range error, which will be discussed in the next section.

C. Experiment III

In the third experiment the mobile robot moves inside the corridor of a building within the campus. It starts in the lower right side of the corridor and drives through it counter clock wise as is depicted in fig. 3. TOM3D has a differential drive wheel base with two wheels and one castor wheel. Each wheel is equipped with a DC-motor integrated with gearbox and quadrature encoder. From those two encoders the x and y position of the robot and its orientation in the global coordinate system is calculated.

An electronic control board is designed for this robot based on a 16-bit Infineon MCU. The robot’s firmware is designed in a way that all sensor information is pre-processed at robot’s control unit and reports are sent to a PC via RS 232. On the top of the robot the Kinect sensor is mounted at an orientation of -90° around the robot's z-axis. Rotations around the x-axis and y-axis are set to zero, because the ground should not be in the field of view since the existence of the ground plane is preconditioned. TOM3D's and Kinect's coordinate systems are shown in Fig 4. It also shows that the normal vector of the plane is described by the spherical angles.
\( \theta \) and \( \phi \) where the shortest perpendicular distance from the origin to the plane is described by \( \rho \). If the Kinect would look straight forward into the moving direction of the robot, most information would be lost since the Kinect has maximum measurement range of 4 m and the parts of the corridor have a length of 10-16 m. Therefore the Kinect sensor looks to the right side of the robot, where most geometric information of the surrounding is located. The global coordinate system is placed at robot power-up position.

Figure 4. Coordinate system of Kinect and Robot; representing the plane's normal vector in spherical coordinates.

IV. RESULTS

In this section the results of the experiments described in chapter 3 are discussed. It is also divided into three subchapters where the results of the different experiments will be presented and discussed.

A. Experiment I

From the first experiment the reference is the angle between the two planes on the cuboid which is 90°. Tab II shows the results on different views. Taken into account that the sides of the reference cube are not truly 90° due to production tolerances, we think that this result is acceptable for our approach.

B. Experiment II

In the second experiment the distance between a wall and the Kinect sensor is changed. We have measured the standard deviation of measurement error in rho for RANSAC and Hough transform to estimate the accuracy of the distance measurement from Kinect sensor. Tab III lists the results. First of all we can see the standard deviation of the Kinect measurement error increases with the distance, furthermore since the standard deviation calculated from both algorithms are almost the same therefore this in fact is the standard deviation of the Kinect’s measurement irrespective of the algorithm used. Because of the small standard deviations of Kinect’s measurement error we think it is still useful to map environment with this sensor.

C. Experiment III

From the resulted 3D generated map by the RANSAC, fig. 6, and the Hough Transform, fig. 5, both produce a visually comparable result. The difference between the two resulted maps is in the top left corner, where the RANSAC fails to find the correct planes, because the corresponding point clouds contain a high number of invalid points. In term of the execution time RANSAC took on average 50 ms to extract the first plane, whereas the Hough Transform took an average of 170 ms to extract a plane. Since no loop closure was used the difference between start and end point in both maps was expected.

V. CONCLUSION

The approach described in this paper is a good base to expand the framework for data association and loop closure. We could extract the plane boundaries by clustering the 3D points based on their position and color information, therefore more accurate plane boundaries would be expected if the objects have uniform color and smaller data association uncertainty because the additional color knowledge is added to the system.

Figure 5. 3D Map generated using Hough Transform

| TABLE II. DETECTED ANGLES BETWEEN TWO ORTHOGONAL PLANES IN DIFFERENT VIEWS |
|-----------------|-----------------|-----------------|-----------------|
| Data Set        | View 1          | View 2          | View 3          | View 4          |
| Angle between planes extracted by RANSAC | 87.7°          | 88.3°          | 87.3°          | 87.1°          |
| Angle between planes extracted by Hough Transform | 87.1°          | 87.4°          | 87.2°          | 86.3°          |

| TABLE III. X-COORDINATES OF THE PLANE BASE POINTS AT DIFFERENT POSITIONS |
|-----------------|-----------------|-----------------|
| Standard deviation in \( \rho \) measurement error | Position 1 | Position 2 |
| RANSAC          | 3.9 mm         | 8.8 mm         |
| Hough Transform | 2.5 mm         | 7.4 mm         |
REFERENCES


