Hole Filled 3D Map Using Mobile Robots in the Urban Environment

Sang Un Park¹, Inwook Shim² and Myung Jin Chung³
¹ Department of Electrical Engineering, KAIST, Daejeon, 350-701, Korea
(Tel: +82-42-350-8029; E-mail: psu@cheonji.kaist.ac.kr)
² Robotics Program, KAIST, Daejeon, 350-701, Korea
(Tel: +82-42-350-5429; E-mail: inugi00@kaist.ac.kr)
³ Department of Electrical Engineering, KAIST, Daejeon, 350-701, Korea
(Tel: +82-42-350-3429; E-mail: mjchung@ee.kaist.ac.kr)

Abstract - 3D mapping and expression of the environment are very important with various applications in many different fields. Among the various sensors (vision sensor, infrared sensor, and LiDAR (Light Detection and Ranging) sensor), we used the LiDAR sensor in this paper because its accuracy is good and easy to handle. There exist holes in the result of 3D reconstruction. When these holes are filled, improved results of localization, and path planning are expected. We propose a contour based sampling algorithm to fill holes. It uses the information from regions near the holes for the prediction, and the results confirm that the filled regions look similar to the real environment.

Keywords – 3D Map, Hole Filling, Contour Based Sampling Algorithm

1. Introduction

Three-dimensional models of urban environments are useful in a variety of applications such as urban planning, object recognition, localization, path planning, and virtual reality. It has been ongoing research to map three-dimensional environments [1-3]. If we create a map with this LiDAR sensor, there exist holes like Fig. 1.

Fig. 1 Result of 3D mapping

If we filled these holes, improved results are expected. For the case of localization, a filled point cloud has more information than a non-filled point cloud so it can match more, so its accuracy will be good [4]. For the case of path planning, if holes are filled, the robot knows whether path is traversable or not [5] well. Therefore, it is necessary to fill holes in 3D map.

Most researcher use vision sensor to fill holes instead of laser sensors, but when there exist few feature points, its performance goes to bad [6]. Frueh, C. et al [7] use laser sensor and vision sensor but they propose just linear interpolation method and holes occluded by objects such as trees or cars cannot be filled well. We propose contour based sampling algorithm to overcome above problems and consider its computation time to apply robot field.

2. 3D Mapping

2.1 The Pose Estimation

To estimate robot’s position first, we should know the position from a GPS (Global Positioning System) sensor and angles from an IMU (Inertial Measurement Unit) sensor. Those sensors’ update rate is different, so we should synchronize update rate by using Kalman Filter [8].

2.2 The Generation of a Point Cloud

To generate a point cloud, we used only LiDAR sensor because its accuracy is good and easy to handle. As shown in Section 2.1, the LiDAR’s update rate also should be synchronized. From pose data and range data, we can make 3D map. Let assume that a robot (vehicle) fixed like Fig. 2, we set coordinates of a point cloud and make 3D map using below Eq. (1).

\[
\begin{bmatrix}
{x_i} \\
{y_i} \\
{z_i}
\end{bmatrix} = 
\begin{bmatrix}
\sin \theta_i & 0 & -h \\
0 & 1 & 0 \\
\cos \theta_i & 0 & \bar{h}
\end{bmatrix}
\begin{bmatrix}
P_{ix} \\
P_{iy} \\
P_{iz}
\end{bmatrix}
\]

\(x, y, \) and \(z\) are position of a point cloud, \(i\) is index of laser, \(\theta_i\) is \(i\)-th scanning angle, \(h\) is height of a LiDAR, \(P_i, P\) and \(P\) are position of a vehicle, and \(t\) is synchronized time. (-) sign means left LiDAR and (+) sign means right LiDAR.

Now, a vehicle moves like Fig. 3, we consider its direction using heading angle (\(\alpha\)).

(-) sign means left LiDAR and (+) sign also means right LiDAR.
Fig. 3 Top view of a moving vehicle

\[
\begin{bmatrix}
x_i \\
y_i \\
z_i \\
\end{bmatrix} = \begin{bmatrix}
r_j \sin \theta_j \cos \alpha_j \\
r_j \sin \theta_j \sin \alpha_j \\
0 \\
\end{bmatrix} \begin{bmatrix} 
0 \\
1 \\
h_j \\
\end{bmatrix} \begin{bmatrix} 
P_{x,i,j} \\
P_{y,i,j} \\
P_{z,i,j} \\
\end{bmatrix}
\]

(2)

3. The Filling Holes

3.1 The Depth Image

As the vertical order is inherent to the scan itself, all scan points of a segment form a 3D scan grid with regular, quadrilateral topology. This 3D point cloud allows us to transform the scan points into a depth image, i.e. a 2.5D representation where each pixel represents a scan point, and the color for each pixel is proportional to the depth of the scan point. The advantage of a depth image is its intuitively easy interpretation, and the increased processing speed the 2D domain provides [7].

Fig. 4 (a) 3D mapping result (b) Depth image

To make depth image, first we transform coordinate of a point cloud from Euclidean to cylindrical projection, and we can compute the height of depth image easily using Eqn. (3). D is a constant to make depth image.

\[
r_j \sin \theta_j \cos \alpha_j : r_j \sin \theta_j \sin \alpha_j : 0 = D \cdot (H_{i,j} - h_j)
\]

\[
H_{i,j} = h_i + \frac{D}{\tan \theta_{i,j}}
\]

(3)

Next, we should compute the distance of depth image like Fig. 4 (b) with color. In other words, 3D map changes to 2.5D depth image and its x direction means path from a vehicle, y direction means the height of objects, and color means its depth distance.

3.2 The Separating Foreground and Background

We define foreground region when objects exist and background region when structures exist. We first get a threshold to separate foreground and background. When we scan in urban environments, most structures are scanned but objects are not. With the one vertical scanning line, most structure regions’ depth is bigger than objects’ region so we can separate. For improved results, we iterate this procedure more than twice. The separating result between foreground and background are shown in Fig. 5.

Fig. 5 (a) Foreground (b) Background

3.3 The Filling Holes

We should fill the holes which caused by occlusion and glass.

A. The filling holes by occlusion

To fill holes, we propose a contour based sampling algorithm which samples neighbors’ depth. An example is as shown in Fig. 6.

Fig. 6 Example of filling holes

The black parts mean holes. To fill these holes, first we randomly make a seed like Fig. 7. (a), and move to the left boundary of holes like (b). Next, we make a 3x3 mask like (c), and fill an hole, and then move a seed to the next boundary like (d). In this way, we move a seed and fill it along the boundary like (e-f). But some cases, we cannot move a seed like (g), when there does not exist a hole around the seed anymore or the number of zeros is less than half of mask size like (h). In this time, we stop and do again above method until holes are filled.
We fill the holes like above method. Figure 7 (i) is result before filling holes and Fig. 7. (j) is result after filling holes.

B. The filling holes by glass

The holes by glass are parts except for holes by occlusion like Fig. 8 (a-b). To fill these holes, we use same method as shown part A. Figure 8 (c) is the result before filling holes and Fig. 8 (d) is result after filling holes.

4. The Experiments and Results

4.1 The System Configuration

We use GPS and IMU sensor to estimate pose of a vehicle, and use LiDAR sensor to make 3D map. All specifications are shown in the table 1.

<table>
<thead>
<tr>
<th>Sensors</th>
<th>Model</th>
<th>Company</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>B20</td>
<td>HUACE</td>
<td>Resolution: 0.5m Update Rate: 1Hz</td>
</tr>
<tr>
<td>IMU</td>
<td>MTi</td>
<td>XSens</td>
<td>Resolution: 0.5deg Update Rate: 50Hz</td>
</tr>
<tr>
<td>LiDAR</td>
<td>LMS200</td>
<td>Sick</td>
<td>Resolution: 0.5deg Update Rate: 31Hz</td>
</tr>
<tr>
<td>Vehicle</td>
<td>EV e-ZONE</td>
<td>CT&amp;T</td>
<td></td>
</tr>
</tbody>
</table>

4.2 The Experiment Results

We experiment in the KAIST campus and Fig. 9-10 is the result.

<table>
<thead>
<tr>
<th>Error Range</th>
<th># Estimated Point Cloud</th>
<th>Percent.</th>
<th>Error Range</th>
<th># Estimated Point Cloud</th>
<th>Percent.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Below 0.1m</td>
<td>6221</td>
<td>85.56</td>
<td>Below 0.3m</td>
<td>4801</td>
<td>99.03</td>
</tr>
<tr>
<td>0.1 - 0.2m</td>
<td>980</td>
<td>13.48</td>
<td>0.3 - 0.6m</td>
<td>46</td>
<td>0.97</td>
</tr>
<tr>
<td>0.2 - 0.3m</td>
<td>70</td>
<td>0.96</td>
<td>Above 0.6m</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Above 0.3m</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>4847</td>
<td>100</td>
<td>Sum</td>
<td>4847</td>
<td>100</td>
</tr>
</tbody>
</table>
Table 3. Result of Image Process. Build. in KAIST

<table>
<thead>
<tr>
<th>Error Range</th>
<th># Estimated Point Cloud</th>
<th>Percent.</th>
<th>Error Range</th>
<th># Estimated Point Cloud</th>
<th>Percent.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Below 0.1m</td>
<td>3129</td>
<td>96.07</td>
<td>Below 0.3m</td>
<td>467</td>
<td>96.09</td>
</tr>
<tr>
<td>0.1 ~ 0.2m</td>
<td>128</td>
<td>3.93</td>
<td>0.3 ~ 0.6m</td>
<td>19</td>
<td>3.91</td>
</tr>
<tr>
<td>0.2 ~ 0.3m</td>
<td>0</td>
<td>0</td>
<td>Above 0.6m</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Above 0.3m</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sum</td>
<td>3257</td>
<td>100</td>
<td>Sum</td>
<td>486</td>
<td>100</td>
</tr>
</tbody>
</table>

5. The Conclusion

In this paper, we experiment in outdoor environments with various sensors and fill the holes by using contour based sampling algorithm. To apply this algorithm, we make a depth map image, separate foreground and background and fill the holes in background region. If these holes are filled, result of localization and path planning will be improved.

Acknowledgement

This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the Human Resources Development Program for Convergence Robot Specialists support program supervised by the NIPA(National IT Industry Promotion Agency) (NIPA-2011-C7000-1001-0007).

References