Orientation Correction of Kinect’s 3D Depth Data for Mapping

A A A Mosed¹, K Kamarudin¹ ² and S Puveneswari¹

¹School of Mechatronics Engineering, Universiti Malaysia Perlis (UniMAP), Arau, Malaysia.
²Centre of Excellence for Advanced Sensor Technology (CEASTech), Universiti Malaysia Perlis, Arau, Malaysia.

*Corresponding author: kamarulzaman@unimap.edu.my

Abstract. Simultaneous localization and mapping (SLAM) is considered to be one of the primary tasks for autonomous robot navigation process. The first element required by the robot to start the navigation process is to create and assemble an accurate map of the surroundings or ground plane to envision its location. Mobile robots that use 1D or 2D sensors such as laser range finder, ultrasonic or sonar for the mapping process cannot provide enough information on the obstacle locations while 3D sensors like Kinect can. However, if the 3D sensor is not mounted correctly, the orientation of the 3D image obtained will also be affected and thus producing inaccurate maps. In this project, Microsoft Kinect was used to scan the environment. RANSAC algorithm was implemented to detect the ground plane. The orientation of the ground plane was corrected, and the depth data acquired by Kinect was then converted into 2D map. It was found that the methods applied have successfully mapped the obstacles detected.

1. Introduction

Simultaneous localization and mapping (SLAM) is considered to be one of the primary tasks to allow autonomous robot navigation. This mobile robotics field was extensively studied in the last twenty years. Even though a number of researches have been done on this particular application, various problems and challenges still exist. Researchers have been doing sustained efforts to make sure that the navigation system more reliable as any failure can cause harm to human beings [1], [2], [3].

Navigation is a technique applied on a robotic or a gadget for the purpose of allowing it to navigate and move from an area to any other targeted place. Although this problem looks to be straightforward, it is a bit difficult in reality due to the sensors and actuators are behaving non-ideally [1].

To start navigation, the first element a robot usually needs is to create and assemble a map of the surroundings or ground plane. To envision its location, the robot requires an accurate map meaning that both approaches need to be held at the same time. The map is vital to a successful navigation procedure, in which the generated map ought to seize enough obstacles to save the robot from collision. The most common sensors that have been used for obstacle detection and avoidance were single range measurement sensors like laser range finders or sonar, or 3D ones like Xbox Kinect.

The possibility of missing an obstacle or misinterpret its location tends to be high with the usage of one dimensional or two-dimensional sensors for the mapping process. It has been proven that the generated maps using those sensors were not efficient as obstacles vary in shape and size. To elaborate more, Figure 1 shows a case where object size needs to be considered during mapping process. For...
this case, the 1D or 2D sensors will not be able to detect the front obstacle, which will lead to a collision during the robot navigation. To overcome this limitation, usage of 3D sensor is believed to be ideal because both shape and size of the object can be obtained. Microsoft Kinect is one of the available 3D sensors that can be used for this purpose [1].

![Figure 1](image1.jpg)

**Figure 1.** Effect of using a robot equipped with 1D or 2D sensors [1].

Kinect has been used instead of the laser range finder to extract object data in indoor environments. Having a three-dimensional view of the surrounding can still be done by rotating laser range finder; however, Kinect sensor is preferable due to its simple usage and advantages. Besides, RGB image and skeleton data, Kinect sensor can also provide the depth image which then can be manipulated to extract helpful information like the point cloud, object shape and laser scan data [4].

2. Previous Work

The conversion of 3D depth data to 2D obstacle location has been done by developing and implementing several algorithms and methods, however problems like detecting the ground plane and correcting the 3D depth data orientation is still a problem that is faced during the conversion process. Building maps during the navigation process was done by other researchers using several types of sensors. Those sensors are 2D ones like Ultrasonic, sonars and laser scan finder or 3D ones like Kinect. Some studies have utilized ultrasonic and sonars to perform 2D mapping. The results were not efficient because both systems that used ultrasonic and sonars can detect the objects that are only on the same level as the sensor. This limitation is undesired since any collision can happen at any instance due to the existence of the obstacles that are either above or below the sensor level [5],[6],[7]. Another study utilized Kinect and Laser scanner to perform 2D SLAM [8]. Although the combination of Kinect and laser has improved the accuracy of detecting obstacles, there are still some deficiencies and cost issues. Even though a 3D sensor was used, the system still cannot detect negative obstacles like holes or stairs. In addition, things become somehow difficult when the mobile robot is installed with Kinect sensor. Sometimes, the mobile robot has to encounter environments with a bumpy road or slope surface. This will cause the depth image captured not align with the floor which will lead to obtain an inaccurate reading and maps of the environment. Economically, the use of both laser scan and Kinect tends to be a bit expensive [3],[8]. 2D SLAM was also performed using Kinect only in several studies. Despite the efficient results obtained, negative obstacles as well as bumpy roads or slope surfaces that a robot may encounter in the environment were not considered [2],[4],[9],[10].
3. Methods

3.1. Ground Plane Detection and Orientation Correction

Plane detection is crucial for numerous vision-based tasks. As objectives of the study focus on detecting positive obstacles as well as correcting the orientation of Kinect’s point cloud data, detecting the ground plane will always be the first step that should be taken. Once the ground plane is successfully detected, both positive obstacles detection and Kinect’s point of cloud data orientation correction can be achieved.

One of the well-used algorithms in the field of computer vision is Random Sample Consensus (RANSAC). Its principle is based on finding the best plane among a 3D point cloud. The algorithm randomly elects three points and calculates the parameters of the corresponding plane. Then it detects all points of the original cloud belonging to the calculated plane, according to a given threshold. Afterwards, it repeats these procedures N times; in each one, it compares the obtained result with the last saved one. If the new result is better, then it replaces the saved result by the new one [11].

Correcting the orientation of the depth image acquired depends on identifying two main planes. One is the ground plane of the image and another one is the real plane where the ground plane of the depth image must lay on. By the time those planes are identified, a rotation matrix and a rotation angle can be computed by the mean of using the normal vectors of the two planes,

\[ \mathbf{n}_1 = (A_1, B_1, C_1) \]
\[ \mathbf{n}_2 = (A_2, B_2, C_2) \]

where \( n_1 \) and \( n_2 \) represent the normal vector of the plane where the depth image must lay on the normal vector of the ground plane detected by RANSAC algorithm respectively.

The rotation angle is then found by the following equation

\[ \theta = \tan^{-1} \left( \frac{A_1 \cdot A_2 + B_1 \cdot B_2 + C_1 \cdot C_2}{\sqrt{A_1^2 + B_1^2 + C_1^2} \cdot \sqrt{A_2^2 + B_2^2 + C_2^2}} \right) \]  

The extracted plane is then rotated according to the following rotation matrix.

\[ \text{Rot}(z, \gamma) = \begin{pmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \]  

\[ T(p) = \begin{pmatrix} \text{Rot}(z, \gamma) & T_{3 \times 1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P_{3 \times 1} \\ 1 \end{pmatrix} \]  

3.2. 3D Depth data to 2D Obstacle Locations

The conversion of 3D depth data to a 2D obstacle location depends directly on the possibility of detecting the floor surface. Once the floor surface is detected, it can be removed to start the conversion process. For this study, the Kinect sensor will be placed at a height of 30 cm from the floor. After the orientation correction algorithm is being applied, the floor surface is filtered out by considering all point cloud between 3 cm to be removed. Moreover, the pixels above the height of the Kinect will be eliminated since they are not representing an obstacle that can be faced by the mobile robot.
3.2.1. Nearest Point Method
A method that can be used to detect the positive obstacles is by implementing the concept of getting
the nearest point in each column of the array that represents the depth image [1]. To do that, the 11-bit
raw depth image of the environment that is represented as an array of 640 × 480 has to be retrieved
from Kinect and to be converted into real depth. Then, X and Y coordinate that are corresponding to
each pixel in the real depth image are calculated using Equations 6 and 7 resulting in three 640 × 480
arrays that represent X, Y and Z. Once the three are obtained, the minimum element in each column in
Z array is extracted out to get 640 × 1 array as shown in the Equation 8,

\[ X_{i,j} = \left( j - \frac{w}{2} \right) \times \frac{320}{w} \times M \times Z_{i,j} \]  \hspace{1cm} (6)

\[ Y_{i,j} = \left( j - \frac{h}{2} \right) \times \frac{240}{h} \times M \times Z_{i,j} \]  \hspace{1cm} (7)

\[ Z_j = \min(Z_{0,j}, Z_{1,j}, ..., Z_{479,j}) \]  \hspace{1cm} (8)

where I is the pixel’s row number, J is pixel’s column number, h and w is the height and width
obtained from Z array respectively and M is the NUI camera depth image to skeleton multiplier
constant.

The X locations for the corresponding Z-elements were also selected giving another 640x1 array.
These two arrays indicate the Z and X coordinates of the nearest pixels and can also be regarded as 2D
obstacle locations as obtained when using a laser scanner. The minimum-selection method in equation
8 was implemented to avoid false detection of the obstacle and minimize processing power and time.
Also note that the Y-Coordinate was ignored as the robot can only move in X and Z direction. The
arrays have already represented the nearest obstacle.

4. Results and Discussion
This Section presents and discusses the result after undergoing ground plane detection, depth image
orientation correction and 3D to 2D depth data conversion. The results are divided into several parts
for further discussion. The first part focuses on the results obtained with the implementation of 3D to
2D depth data conversion algorithm by showing several depth images. The second part focuses on the
results obtained with the implementation of the RANSAC algorithm to detect the ground plane of the
scanned environment and finding the rotation angle where the last part focuses on the results obtained
with implementation of orientation correction algorithm.

4.1. Results Obtained with The Implementation of 3D to 2D depth data Conversion algorithm

Figure 2 shows the RGB and depth Image of the surrounding environment that was obtained by
Kinect. The purpose of including RGB image is just to show view of the Kinect during the scanning
process. The depth image shows dark and bright areas. The bright areas represent the parts of the
environment that were accessible by the Kinect where the brighter the object is the closer it is to the
Kinect. On the other hand, the black areas represent an invalid data of the environment.
The invalid data comes from three main sources which are the sensor itself, the measurement setup and the properties of the object surface such as smooth and shiny surfaces. These surfaces that appear overexposed in the infrared impede the measurement of disparities and result in gaps which are represented in Fig 4 as black pixel with value of 0.

The algorithm starts by rotating the ground plane detected by RANSAC so that it is aligned with the horizontal plane. Then, the ground plane was removed by applying the vertical limit filter where all features that are higher than 27 cm and below than 8 cm were removed. In addition to the vertical limit filter, the horizontal limit filter was also applied to remove all pixels that are further than or equal to 2.5 m from Kinect resulting the image shown in Figure 3.

The conversion process was continued by extracting the smallest value stored in each column of the filtered image with its location along the X axis (see Section 3.2.1). The obstacles were then plotted based on the extracted data and the result of the conversion is shown in Figure 4.
4.2. Results Obtained with The Implementation of RANSAC Algorithm to Detect the Ground Plane

It is known that the results of RANSAC algorithm directly depend on two main variables. Those variables are the number of iterations which represent how many times RANSAC will be implemented on an image to extract the ground plane and the threshold distance that is set to test how far points are from the fitted plane. Results obtained with the implementation of RANSAC algorithm to detect the ground plane with different iterations and different threshold distance are represented in Table 1 and Table 2 respectively.

| Table 1. Ground plane detection with different iterations and threshold distance of 0.3cm. |
|---------------------------------------------|
| Number of iterations | 80 | 50 | 10 |
| Result | ![Ground Plane](image1) | ![Ground Plane](image2) | ![Ground Plane](image3) |

| Table 2. Ground plane detection with different threshold distance. |
|---------------------------------------------|
| Distance | 0.3cm | 0.4cm | 0.5cm |
| Result | ![Ground Plane](image4) | ![Ground Plane](image5) | ![Ground Plane](image6) |

The results obtained showed that a threshold distance of 0.3cm and 80 iterations are the best for RANSAC to detect the ground plane. It was also noticed that with 50 iterations, RANSAC successfully detected the ground plane; however, it is not efficient due to the detection of some pixels that represent the wall of the environment. This leads to the fact that the higher the number of iterations, the more accurate the RANSAC is. The reason behind this is when the number of iterations is high, it allows RANSAC to test more points and compare the results obtained to choose the best one. This does not mean that RANSAC will not test the point of the 3D point cloud set when the number of iterations is low. Testing the points of the 3D point cloud set is made in both cases but the higher the number of testing times the more accurate the results obtained are.

Trying different threshold values also influences RANSAC results. Table 2 shows the results obtained by implementing RANSAC with 0.3 cm, 0.4 cm and 0.5 cm as a threshold distance. It was found that 0.3 cm was the best threshold distance for this scenario while 0.5 cm was the worst as the output image was the same as the input image. This is because when the distance is increased, more points will be added to the set of points that represent inliers of the ground plane and the opposite happens when the distance is decreased.
4.3. Results Obtained with The Implementation of Orientation Correction Algorithm

Figure 5 shows a comparison between the point cloud image before and after applying the orientation correction algorithm. Image (a) is the same image shown in previous Figures but represented in terms of 3D point cloud. RANSAC result has clearly shown that there is an angle between the detected ground plane and the real plane where the ground plane must lay on. As a result, the orientation correction algorithm was implemented, and the result presented in image (b) shows that the orientation of the plane was successfully corrected.

![Figure 5](image)

Figure 5. Orientation correction result: (a) is the depth image represented in terms of 3D surface, (b) is the depth image.

5. Conclusion

The project was mainly focusing on correcting the orientation of Kinect so that 3D depth data acquired by the sensor can be properly converted to 2D obstacle location. To achieve that, ground plane had to be detected using RANSAC and checked either it lies on the horizontal plane of the Kinect. Orientation correction had to be applied to the detected ground plane in case it did not lie on the horizontal plane of the Kinect. The orientation correction method is inspired by the robotic theory and geometric coordinate system. Transformation matrix is used to solve the orientation problem. The 3D to 2D conversion method is then utilized to represent the depth data as 2D obstacle location.

As referring to the results obtained, ground plane detection, ground plane orientation correction and 3D depth data to 2D obstacle locations conversion were successfully achieved. The method used to detect the ground plane is affected by number of iterations and distance between points of plane while other methods have shown their reliability.

As future improvement for 3D depth data conversion to 2D obstacle location, RANSAC algorithm could be implemented and enhanced with estimated surface normal. The orientation correction is implemented on X-Z axis plane and for further improvement, the orientation correction can be implemented on X-Y plane.

The current work focussed on the detection of the positive obstacles. The detection of negative obstacles would make the system more reliable during the robot navigation. A suggested method to detect the negative obstacles is to utilize both farthest point method, instead of the nearest one, and virtual floor projection method.

6. Acknowledgement

The author would like to acknowledge the support from the Fundamental Research Grant Scheme (FRGS) under a grant number of FRGS/1/2018/TK04/UNIMAP/02/15 from the Ministry of Education Malaysia.
References

[1] Kamarudin K et. al. 2013 Method to convert Kinect’s 3D depth data to a 2D map for indoor SLAM Proc. - 2013 IEEE 9th Int. Colloq. Signal Process. its Appl. CSPA 2013 247–251
[2] Kamarudin K, Mamduh S M, Md Shakkaf A Y and Zakaria A 2014 Performance analysis of the microsoft kinect sensor for 2D simultaneous localization and mapping (SLAM) techniques Sensors (Switzerland) 14 23365–23387
[3] Graham B 2013 Obstacle Mapping with a Mobile Robot using the Microsoft Kinect Sensor The UNSW Canberra at ADFA journal of Undergraduate Engineering Researchers 11 1–13
[4] Ghani M F A, Sahari K S M and Kiong L C 2014 Improvement of the 2D SLAM system using Kinect sensor for indoor mapping Jt. 7th Int. Conf. Soft Comput. Intell. Syst. SCIS 2014 15th Int. Symp. Adv. Intell. Syst. ISIS 2014 776–781
[5] Barak N, Gaba N and Aggarwal S 2016 Two Dimensional Mapping by using Single Ultrasonic Sensor International Journal of Advanced Research in Computer Science Research 7 254–257
[6] Kalmegh S K 2010 Obstacle avoidance for a mobile exploration robot using a single ultrasonic range sensor International Conference on Emerging Trends in Robotics and Communication Technologies 8-11
[7] Avirup Basu S S, Ghosh S K 2016 Autonomous Navigation and 2d Mapping Using Sonar 5th International Conference on Wireless Networks and Embedded Systems (WECON) 1–5
[8] Md Shakkaf A Y, Kamarudin K, Visvanathan L M, Mamduh S M, Kamarudin A Y M S and Zakaria A 2015 Feasibility Analysis Of 2d-Slam Using Combination Of Kinect And Laser Scanner J. Teknol. 1–7
[9] Peasley B and Birchfield S 2013 Real-time obstacle detection and avoidance in the presence of specular surfaces using an active 3D sensor IEEE Work. Robot Vision, WORV 2013 197–202
[10] Aichele F, Schenke B, Eckstein B and Groz A 2016 A framework for robot control software development and debugging using a real-time capable physics simulation 47th Int. Symp. Robot. ISR 2016 173–180
[11] McDaniel M W, Nishihata T, Brooks C A and Iagnemma K 2010 Ground plane identification using LIDAR in forested environments Proc. - IEEE Int. Conf. Robot. Autom 3831–3836