Proc. of the Third Intl. Conf. on Advances in Computing, Electronics and Electrical Technology - CEET 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-056-9 doi: 10.15224/ 978-1-63248-056-9-85

Combined Plane and Point Registration of Sequential Depth Images for Indoor Localization

Nguyen Vu Duy Hau[1], Nguyen Duc Thang[1*], Trinh Thi Lan Anh[2] and Tran Cong Hung[2]
[1] Department of Biomedical Engineering, International University - VNU-HCMC, Vietnam
[2]Posts and Telecommunications Institute of Technology-HCM, Vietnam

Abstract
Indoor environment usually has complicated structures and contains planes which can be used as extra features for registration. This paper proposes a method to use planar features extracted from depth camera for indoor localization. Two consecutive depth images are converted to point clouds and segmented to planes. Two sets of planes are then matched together to estimate the rotation matrix which is used as initial guess of iterative closest point (ICP) of point-to-plane registration. Our experimental result shows that ICP of point-to-plane algorithm with the extra step of estimating the rotation guess matrix performs faster than conventional ICP and overcomes the drawback of ICP point-to-point algorithm in indoor environment.

Keywords—Iterative closest point (ICP), Plane segmentation, Registration.

I. Introduction

Over decades, localization always plays important roles in autonomous mobile robotic [1] especially for indoor localization in which accurate and safe navigation is strictly required due to space limitation and complicated indoor environment. However, an indoor environment in general contains a floor and the localization is narrowed down to the problem of estimating the poses, i.e., the position and orientation, of a mobile robot on the floor plane. The most conventional method for localization applied wheel odometer. The rotational encoder was mounted on the wheels of a mobile robot to provide distance and direction of each wheel which was later used to calculate the pose of the robot. However, this method results to unbounded accumulated errors caused by multiple factors such as different wheel diameter, wrong estimated wheelbase, wheel slip and limited resolution of wheel encoder. Another localization method namely visual odometry has been developed to track the pose of a robot by analyzing just images captured by a camera attached to it [2]. An obstacle preventing practical implementation of this approach involves the lacks of 3D information when a normal RGB camera is used. Recently, it is possible to obtain useful information including depths via a single camera, making mobile robot navigation and indoor localization feasible. So far, many types of depth cameras have been developed. A stereo camera achieves depth perception in a manner similar to human eyes by finding the correspondences between two images from the left and right cameras to estimate the disparity images. The disparity let us know how far from the interested points to the camera through perspective projection. However, finding the correspondences of pixels from the two images is not an easy task. If the global view of image is concerned, it consumes a lot of time to process the whole image. Meanwhile, if just local areas of an image are taken into account, the correspondence from one pixel to others cannot be exactly evaluated. A time of arrival (TOA) camera emits a beam of laser and receives the light reflection from the object surfaces to reveal depths. With complicated implementation, TOA camera is expensive. The structured light-based camera uses a structured-dot pattern to light an object. The deformation of dot-light let us know the distance from a point in 3-D to the camera. With current implementation of Kinect camera released by Microsoft in 2011, the structured light-based camera achieves better quality of depths with reasonable price so that it has been accepted widely for indoor applications.

To deal with the localization problem for a mobile system using depth images captured by a Kinect camera, a frame to frame registration is critical to track the position of a moving wheel chair. So far, the iterative closest point utilizing point-to-point approach (ICP) [3] has been the most conventional technique to align the two 3D point clouds of two consecutive depth images. However, ICP usually leads to local optimization because of the bad initialization of transformation parameters. Besides, the appearance of big planes such as floors or walls in a depth images mainly determines the transformation to fit the two 3D points. If a robot is going straight away, somehow the main planes between two frames are similar. Consequently, the moving information cannot be captured correctly. Although there has been a variant of ICP utilizing point-to-plane (ICP-P2P) approach [4], the above problem was not completely solved. In this paper, we address the registration problem of 3D point clouds captured by a depth camera. To overcome the disadvantages of ICP and its variants to handle the appearances of big planes in a depth image, we propose a two-stage registration ICP. In the first stage, the planes are detected from two 3D point clouds and an estimated rotation matrix is found by matching different planes of two point clouds. The fine registration is later performed with ICP-P2P and use estimated rotation matrix as guess matrix when the effects of the big planes are mitigated than ICP. Actually, by using estimated rotation matrix as initial guess can help ICP-P2P to converge faster and more robust.

The rest of paper is organized as follows. Section 2 describes our proposed methodology. We present

Proc. of the Third Intl. Conf. on Advances in Computing, Electronics and Electrical Technology - CEET 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-056-9 doi: 10.15224/ 978-1-63248-056-9-85

experimental results with discussion in Section 3. Finally, we conclude this paper with Section 4.

II. Methodology

A. *Kinect camera and an electronic wheel chair*

The Kinect was introduced by Microsoft as a new controller for Microsoft Xbox 360 video game to capture gamers' motion without requiring a marker attached on their bodies. The two valuable sensors of Kinect are the basic RGB camera and the depth camera which contains an infrared (IR) emitter and an IR receiver as depicted in Fig. 1.

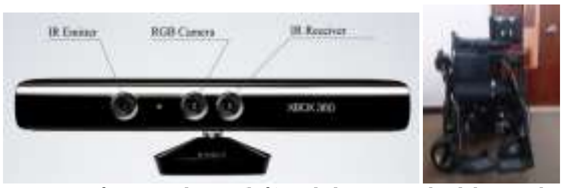

Figure 1. Microsoft Kinect device (left) and electronic wheelchair with Kinect camera (right).

In details, the Kinect camera has angular field of view of 57° horizontally and 43° vertically. The camera is well operated to capture depths from 0.8m to 6 m. Besides it can provide two images with full VGA resolution of 640×480 pixels at a recording rate of 30 frames per second, the former for 24 bit RGB images and the later for 11 bit depth image with 2048 levels of depths.

Because of its low cost and capability to produce depth image, it becomes the potential sensor for wide applications. Therefore beside the official SDK provided by Microsoft to support application development for Kinect, additional open-source driver [5] as well as processing tools of third party were created to make Kinect's application easy to develop and available to cross platforms (Mac, Linux, and Windows) [6]. After a depth image is captured by a depth camera, its pixels are transformed into 3D point clouds in a real world coordinate system as

$$X = \frac{(u-u_0)Z}{f} \qquad Y = \frac{(v-v_0)Z}{f} \tag{1}$$

where *u* and *v* are the row and column index of a pixel, u_0, v_0, and *f* are the parameters configured by a depth camera.

However, depth data provided by Kinect is huge and needs to be down-sampled and filtered before further process. In this work, a depth image with resolution 640×480 pixels is reduced to a resolution of 320×240 pixels. The spatial filter is applied to range depth data from 0.6 – 4*m* horizontally and below 1.6*m* from the floor along the vertical direction as illustrated in Fig. 2. The depth range is chosen between 0.6 – 4*m* to ensure valid data collection and minimize acquired errors of depth data to about 0.024*m* at maximum 4*m* of depth [7, 8]. The vertical range below 1.6*m* from the ground is due to the maximum height of the user sitting on a wheelchair. Lastly, the 3D point cloud is passed through a voxel grid filter with the leaf size of 0.05*m* provided by point cloud library (PCL) [9]. A voxel is a 3D box in space and contains several 3D points, so the point cloud was divided into many square voxel with the length of 0.05*m* and all points in each voxel are represented by its centroid.

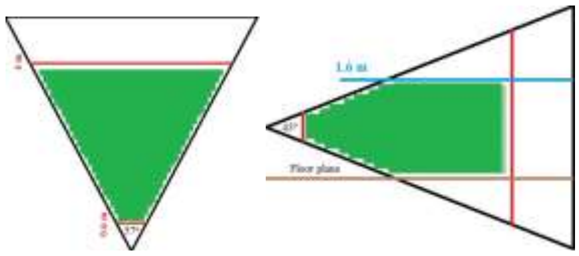

Figure 2. Spacial filter of the system, horizotal view (left) and vertical view (right). The valid areas are marked by green.

B. *Plane detection and segmentation*

After depth image is obtained and converted into 3D point clouds, plane segmentation is applied. Many algorithms for 3D data plane segmentation have been proposed in recent years. Some researches considered depth image as a 2D gray-scale image and applied segmentation methods such as mean-shift clustering or graph-base segmentation. Although those methods achieved good result in some cases, close or touching objects in a depth image usually make them to perform badly. Other algorithms used RANSAC to robustly estimate the parameters of all planes in a depth image. However, such algorithms are slow and unsuitable for real-time systems as well as large and complex environments such as office room with furniture. Dub et al [10] applied the randomized Hough transformation to extract planes from depth images. A noise model is constructed to solve the task of finding parameter metric for randomized Hough transformation. Although, this algorithm is capable for real-time running of mobile robot platform, plane segmentation cannot be accurately performed.

In this work, we apply the method described in [11] which consists of voxel-wise initial segmentation and pixel-wise accurate segmentation. Firstly, the depth image is converted into 3D point cloud and divided into voxels. Then normal of each voxel is calculated and area-growing algorithm is used to extract raw planes. Later, unclassified points are examined whether they belong to a plane or not. Then fragments of the same plane are merged together.

In voxel-wise initial segmentation as illustrated in Algorithm 1, the point cloud is divided into voxels with a size of 0.2*m* in length. For each voxel if the number of point in that voxel is greater than a threshold, i.e., ten points in our experiments, it is considered as a valid voxel for estimating plane equation and points in a valid voxel is named as inlier-point. Least square equation is applied to estimate the plane

Proc. of the Third Intl. Conf. on Advances in Computing, Electronics and Electrical Technology - CEET 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-056-9 doi: 10.15224/ 978-1-63248-056-9-85

equation that fits to all point of the voxel. Let a set of points in a voxel be $X_i = \{x_i, y_i, z_i\}\ i = \{1, \dots, n\}$ that is used to formulate the plane equation $Y = aX + bZ + c$. The coefficients a, b, and c are found by the mean square estimation to minimize $\sum_i (ax_i + bz_i + c - y_i)^2$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sum x_i^2 & \sum x_i z_i & \sum x_i \\ \sum x_i z_i & \sum z_i^2 & \sum z_i \\ \sum x_i & \sum z_i & n \end{pmatrix}^{-1} \begin{pmatrix} \sum x_i y_i \\ \sum y_i z_i \\ \sum y_i \end{pmatrix}. \quad (2)$$

After estimating planes for valid voxel, we check for whether the estimated planes are good enough to be used. The sum of point to plane distance at each valid voxel are calculated. When this sum is close to zero, it means that all points belong to the estimated plane. However if this sum is greater than a threshold, the corresponding voxel has too many artifacts. Therefore the voxel becomes invalid and all points in that voxel become outlier-point.

After plane estimation, these voxel-planes are pieces of a large-real plane so normal and area-growing algorithms are used to extract initial plane. If adjacent voxels have the same normal direction or the angle of them smaller than a threshold, we cluster them together. Finally, if a voxel cluster is larger than a threshold, it is considered as a plane. In this process, voxels belong to a plane are called planar-voxels, and voxels that not belong to any plane are called non-planar voxels. Points in all the non-planar voxels should be accurately examined pixel by pixel in the refined segmentation later.

Algorithm 1: Initial plane segmentation

Step 1. Traverse the voxel grid and find a voxel V_o that has not been processed. Calculate average distance d of points of V_o to its own estimated plane. If d is smaller than a threshold then create a queue Q and add V_o to Q. Otherwise, find another V_o.

Step 2. Examine 26 neighbor voxels V_i (i=1,2….,25,26) of the voxel V_o. If the voxel V_i has not been processed then go to the calculation:
Let $n_i = (x_i, y_i, z_i)$ be the normal of V_i and $n_o = (x,y,z)$ be the normal of V_o. Calculate the angle of the two normal vectors as follows,

$$\cos\theta = \frac{xi.xo + yi.yo + zi.zo}{|ni|.|no|}$$

If $|\cos\theta|$ is larger than a threshold V_o and V_i are added to the same cluster. If the cluster is larger than a threshold we consider it as a plane. Then insert V_i to Q.

Step 3. Pick another element from Q and regard it as V_o and return to *step 2*.

Step 4. If Q is empty, return step 1.

Step 5. Repeat step 1 to step 4 until all voxels are processed.

In algorithm 2, pixel-wise accurate segmentation is the process to refine the roughly segmented plane that we get from initial segmentation. In this process, each point of non-planar voxel is checked to determine whether it belongs to a certain plane or not. It is obviously that a point is on a plane if the distance from the point to the plane is nearly zero. Hence for each point of non-planar voxel, the distance of it and neighbor voxel-plane is checked. If the distance is smaller than a threshold the point is considered to be of that plane. In details, the number of adjacent voxels is a parameter in this algorithm.

Algorithm 2: Refine segmentation

Step 1. Traverse voxel grid, find a voxel V_o that has not been clustered to a plane (non-planar voxel).

Step 2. Search in 26 neighbor voxels V_i (i=1,2….,25,26) of the voxel V_o, a plane P_i that has not been processed.

Step 3. Calculate distance d from points of the non-planar voxel to plane P_i. Let p (x',y',z') denote coordinates of one point in center voxel V_o, V_i denote the i-th planar voxel in the neighborhood that belong to the plane P_i parameterized by Ax + By + Cz + 1 = 0 .The distance from p to P_i is:

$$d_{ij} = \frac{|Ax' + By' + Cz' + 1\,|}{\sqrt{A^2 + B^2 + C^2}}$$

If d_{ij} is smaller than a threshold p is considered belong to plane P_i, otherwise p is not a point in P_i.

Step 4. Return to step 2 until all surrounding planes are processed.

Step 5. Repeat step 1 to step 4 until all non-planar voxel have been processed.

After fine segmentation, all planes are segmented, but some plane may be divided into several parts. So in order to merge pieces of a real plane together, the angle of normal of adjacent planes is checked again. If it is smaller than a threshold, we merge them as a plane.

C. *Proposed plane registration*

Kinect captures depth image at the frame rates of 30Hz and a wheelchair speed is slow in indoor environment so that two continuous depth image will have their corresponding point clouds highly overlapped when presented on the same Kinect coordinate as illustrated by Fig. 3.

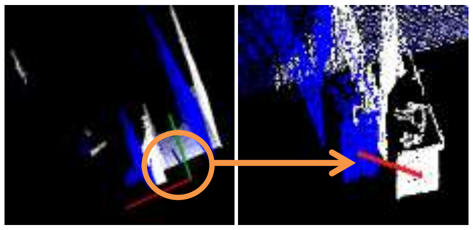

Figure 3. Two point clouds constructed from two consecutive depth images shown on the same coordinate system of Kinect camera (left). The circle area illustrates two distinguished table corners matched by a red line but they are actually the same corner table in the real world (right).

The blue point cloud corresponding to the first depth image captured by Kinect at the starting position and the white point cloud corresponding to the second depth image captured by Kinect when it was moved forward and turned left. Fig. 4 simply illustrates the point clouds of Fig. 3 in which it shows the 2D view of floor plane on the real world coordinate with the origin O and the green line is the table corner mentioned in Fig. 3. O_1 and O_2 are the position of Kinect on the 2D real world coordinate and the triangle is the horizontal field of view of the Kinect at each position. The blue point cloud is captured when Kinect at the position O_1 or the world origin O. The white cloud is captured at the position O_2 after the Kinect moves forward and turns left. The Fig. 4 shows that the distance from table corner – green line to position O_1 is larger than to position O_2. This means the distance from the table corner to O_1y vector is also larger than to O_2y vector. As a result, the white corner table point cloud is closer to the coordinate origin than the blue one. The corresponding table corner plane of the first point cloud (blue) was the result of the corner plane of the second point cloud (white) rotated to left

Proc. of the Third Intl. Conf. on Advances in Computing, Electronics and Electrical Technology - CEET 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-056-9 doi: 10.15224/ 978-1-63248-056-9-85

around the vertical axis which perpendicular to floor planes (the green axis in Fig. 3) and then translated forward. Consequently, a rotation matrix which transformed two actual corresponding planes to parallel was also the actual rotation matrix. So the task was try to find the potential corresponding plane pairs between two point clouds and calculate the best estimated rotation matrix which transformed all planes of second point cloud to parallel to corresponding planes of first point cloud.

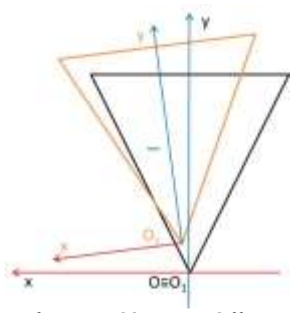

Figure 4. 2D view of corner table in two different Kinect position,

After segmenting planes of two point clouds, we have a list of planes in both point clouds. We aim to find the estimated rotation matrix by firstly finding the potential corresponding plane pairs between two point clouds. Two points cloud are highly overlapped because of the slow speed of indoor environment. Consequently the corresponding planes between two point clouds are close and the number of points in plane of both corresponding plane are similar. So two planes are potentially corresponding to each other when distance between two plane centroids are closest and the number of point in planes are not much different. Based on these properties for each plane of the first point cloud, we can find the potential corresponding plane of the second point cloud. The actual corresponding plane pairs provide the rotation matrix which transforms them and also other actual corresponding plane pairs to parallel. Each potential corresponding plane pairs lead an estimated rotation matrix and the best rotation matrix is the one that maximizes the parallel of all potential corresponding plane pairs. We use an estimated rotation matrix to transform all planes in the second point cloud and calculate cosine angle between each corresponding plane pairs (planes of first point cloud and transformed planes of second point cloud). For more robustness we multiply the number of point in plane of first point cloud to each cosine angle because the larger plane has more priority than the smaller plane and sum all of them. The best estimated rotation matrix is the one that has the largest sum. The algorithm to find estimated rotation matrix is described in Algorithm 3.

Finally, we run ICP-P2P algorithm to align the second point cloud to the first point cloud with the best estimated rotation matrix as guess matrix.

Algorithm 3: Estimate Rotation Matrix

Input: Plane coefficients of both point cloud

Output: The estimated rotation matrix

```
Step 1. Calculate centroids of each planes in both point clouds
Step 2. for each plane in the first cloud
        find corresponding plane in the second cloud which has minimum distance between centroids and ratio number of point between two planes in range 0.5 – 2m
        end for

Step 3. for each plane in the first cloud
        calculate cosine angle of two planes, get value of angle, rotate the corresponding plane in second cloud by calculated angle around the vertical axis, recalculate cosine angle of both rotated planes with plane in first cloud and finally get rotate angle corresponding to the larger cosine angle
        if larger cosine angle > 0.99 (corresponding planes were parallel)
            for each plane in the first cloud
            rotate all planes in second cloud  by rotate angle, calculate sum of number of point in first cloud multiplied with absolute cosine angle between plane in first cloud and corresponding rotated plane in second cloud, i.e., (sumOfCos = sumOfCos + numOfPointInPlaneOfFirstCloud * |cos(planeInFirstCloud & correspondingRotatedPlaneInSecondCloud)|)

            find the best rotate angle and construct the estimated rotation matrix with corresponding to the maximum of sumOfCos
            end for
        end if

end for
```

III. Experimental Results

Fig. 5 illustrates the results of plane segmentation. Obviously, it can be seen that the planes extracted were presented with different color and the points which did not belong to planes were removed.

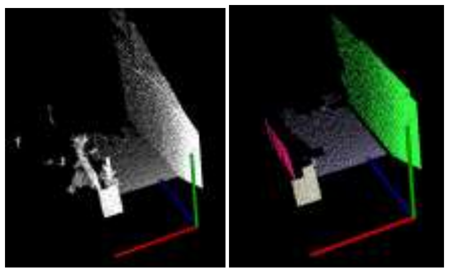

Figure 5. The point cloud (right) and the corresponding point cloud after plane segmentation (left).

As mention previously, the problem of ICP in indoor registration is illustrate in Fig. 6, the first cloud is the white cloud, the transformed second cloud is the green cloud, the second cloud is captured after wheelchair move forward about 0.2 m. Because of the overlap area of floor plane and the wall too large, ICP fails to move the second cloud forward to maintain the distance between point to point smallest. In the other hand, the ICP-P2P performs well in this case the second cloud is moved forward two corner table are in the same plane now [8].

Proc. of the Third Intl. Conf. on Advances in Computing, Electronics and Electrical Technology - CEET 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-056-9 doi: 10.15224/ 978-1-63248-056-9-85

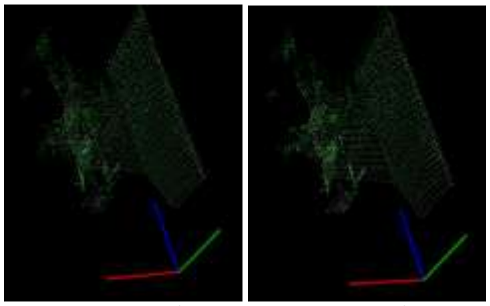
Figure 6. Two point cloud after registration by ICP (left) and ICP-P2PG (right).

In our experiments, the wheel chair with mounted Kinect was moved forward with small variations of moving left or right and the data was recorded with 200 frames. We divide the running steps of our algorithm into three data sets: 5 frames (SF-5), 10 frames (SF-10) and 20 frames (SF-20) and three algorithms: ICP, ICP-P2P and our proposed ICP-P2P with the estimated rotation matrix as initial guess (ICP-P2PG). We also test the effects of our spatial filter where the data is divided in two sets: using spatial filter and without using spatial filter. The results are showed in Table 1 (The spatial filter is used) and Table 2 (The spatial filter is not applied) respectively. We used the mean distance of transformed plane centroids of second point cloud to corresponding planes of the first point cloud to evaluate the accuracy. The smaller the mean distance was the more accuracy the registration performed.

The comparisons were make after all fail registration were eliminated. Our results show that the mean distance of ICP-P2PG is smaller than ICP in all cases, revealing that ICP-P2PG performed better than ICP method. Meanwhile ICP-P2PG is faster than ICP-P2P as the averaging iteration needed for the convergence of ICP-P2PG is smaller. If we use spatial filters, more planes were eliminated. It likely leads to the plane matching more accuracy and successful registration.

TABLE I. COMPARISION BETWEEN ICP, P2P AND P2PG WITHOUT SPATIAL FILTER.

	SF-5			SF-10			SF-20		
	ICP	ICP-P2P	ICP-P2PG	ICP	ICP-P2P	ICP-P2PG	ICP	ICP-P2P	ICP-P2PG
Mean Distance	0.1113	0.1047	0.1048	0.1511	0.1654	0.1498	0.2072	0.1643	0.1642
Registration Fail	6|40	6|40	6|40	6|20	4|20	4|20	6|10	5|10	7|10
Mean Iteration	n/a	23.25	17.59	n/a	33.25	33.18	n/a	40.66	12.66

TABLE II. COMPARISION BETWEEN ICP, P2P AND P2PG WITH SPATIAL FILTER.

	SF-5			SF-10			SF-20		
	ICP	ICP-P2P	ICP-P2PG	ICP	ICP-P2P	ICP-P2PG	ICP	ICP-P2P	ICP-P2PG
Mean Distance	0.1193	0.1139	0.1122	0.2056	0.1226	0.1232	0.5701	0.3835	0.4815
Registration Fail	6|40	4|40	3|40	6|20	5|20	2|20	5|10	5|10	5|10
Mean Iteration	n/a	22.54	20.45	n/a	21.5	39.14	n/a	43.25	33.25

IV. Conclusion

In this paper, we presented a new approach of 3D point cloud registration for indoor environment based on plane segmentation and ICP-P2P. Because indoor environment is usually dominant by planes and especially flat floor plane, we can use plane feature to estimate the rotation matrix and down-sample data by removing points which are not in planes to make ICP converge faster and more robust. As the experimental results shows, ICP-P2PG outperforms ICP regarding accuracy and converges faster than ICP-P2P. However, Our proposed method performs ineffectively when the floor plane is not flat and when the number of planes detected is small. Our future works will focus on addressing these rising issues.

Acknowledgment

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2013.11.

References

[1] Cox, I.J. "Blanche-an experiment in guidance and navigation of an autonomous robot vehicle", IEEE Transactions on Robotics and Automation, 7(2), pp. 193-204, 1991.

[2] Nistér, D., Naroditsky, O., and Bergen, J., "Visual odometry', in Visual odometry (IEEE, 2004, edn.), pp. I-652-I-659 Vol. 651

[3] Besl, P.J., and McKay, N.D., "Method for registration of 3-D shapes', in Method for Registration of 3-D Shapes (International Society for Optics and Photonics, 1992, edn.), pp. 586-606.

[4] Chen, Y., and Medioni, G., "Object modeling by registration of multiple range images', in Object modeling by registration of multiple range images (IEEE, 1991, edn.), pp. 2724-2729.

[5] https://github.com/avin2/SensorKinect/, accessed 18th December, 2013

[6] https://github.com/OpenNI/OpenNI, accessed 18th December, 2013

[7] Peasley, B., and Birchfield, S., "Real-time obstacle detection and avoidance in the presence of specular surfaces using an active 3D sensor", (IEEE, 2013, edn.), pp. 197-202

[8] Khoshelham, K., and Elberink, S.O., "Accuracy and resolution of kinect depth data for indoor mapping applications", Sensors, 12 (2), pp. 1437-1454, 2012.

[9] http://pointclouds.org/, accessed 18th December,2013.

[10] Dub, D., Zell, A., "Real-time plane extraction from depth images with the randomized hough transform" IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 1084–1091, 2011.

[11] Fusiello, A., V. Murino, "Real-Time Plane Segmentation and Obstacle Detection of 3D Point Clouds for Indoor Scenes" ECCV. Workshops and Demonstrations, Springer Berlin Heidelberg. 7584: 22-31, 2012.