Accurate and Fast Extraction of Planar Surface Patches from 3D Point Cloud

Huu Hung Nguyen, Jaewoong Kim
Yeonho Lee, Naguib Ahmed
Intelligent Systems Research Institute
Sungkyunkwan University
Suwon, Rep. of Korea
+8231-299-6471
hunghvktqs2003@gmail.com

Sukhan Lee[1, 2, *]
[1]Intelligent Systems Research Institute
[2]Department of Interaction Science
Sungkyunkwan University
Suwon, Rep. of Korea
+8231-299-6470
lsh@ece.skku.ac.kr

ABSTRACT

Planar surface patches, extracted for instance from a captured 3D point cloud, play an important role as a feature in modeling and/or recognizing objects and environments. In terms of extracting such planar surface patches, the main technical issue involved may be of overcoming the trade-off between the modeling accuracy and the computational efficiency, especially, when the 3D point cloud captured is noisy. Conventionally, they have taken approaches that seek either for high computational efficiency at the expense of modeling accuracy or vice versa. This paper contributes the advancement in the methodology of extracting planar surface patches from a noisy 3D point cloud by presenting a method that provides high modeling accuracy yet under low computational cost. The major contribution of the proposed method is summarized as follows: 1) A robust estimation of surface normal vectors in such a way as to minimize the effect of noise in the data. 2) An accurate localization of density peaks based on a spherical coordinate operator with a flexible size of sliding window used for estimating the density of surface normal vectors represented on a unit spherical surface as a means of accurately identifying those planar surface patches of the same orientation.3) A segmentation of individual planar surface patches of the same orientation by projecting those 3D points clustered as belonging to a same peak onto x, y and z axes of the Cartesian coordinate with the Y axis representing their orientation. The experimental results show the effectiveness of the proposed method in comparison with conventional ones for various indoor and outdoor images.

Categories and Subject Descriptors

1.5.4 **[Pattern Recognition]:** Applications - *Computer vision*

General Terms

Algorithms, Reliability, Theory.

Keywords

Plane Segmentation, Clustering, Peak Detection.

ICUIMC(IMCOM)'13, January 17–19, 2013, Kota Kinabalu, Malaysia.

1. INTRODUCTION

For object recognition, segmentation to extract geometric features, like planes, cylinders, and cones, from the 3D point cloud is one of the most important processes. Planar primitives presented within an object can provide useful information. Planes, and their spatial relationships, help the robot to determine the kind of object in its field of view.

A lot of works related to plane extraction exist in [1]–[3]. In [3], only three corresponding points from the stereo images are used to estimate the normal vector of a plane and the paper collects triangles with the same normal vector for plane detection. The algorithm complexity is $O(n^2)$ where n is the number of triangles. It will take a longer time if there are a large number of triangles. The approach presented in [4] detects multiple planar regions using a progressive voting procedure from the solution of a linear system exploiting the two-view geometry. They work directly with data from the camera. A method for detecting the three-dimensional planar surfaces using 3D Hough transformation to extract candidates to plane segment is presented in [2]. These solutions work well, but the level of computational complexity is high. In [5], a real-time solution for plane segmentation is proposed through classification in normal and distance space using a RGB-D camera with low accuracy and resolution. This method works well with a low resolution. A fast surface normal estimation method is accepted so that the average deviation is large, at around 10 degrees (QVGA). For that reason, if the resolution is high and the object is more complex, some close patches can be merged.

Recently, a structured light camera has been used to generate a 3D point cloud with a high resolution. RANSAC and 3D Hough Transform are applied to detect geometric features, like planars, cylinders, and cones. These methods usually have high computational complexity. This paper presents a new method for plane segmentation that can work in real time by combining clustering in Spherical space and distance space. In Spherical space, called IJK space, a solution is provided to detect peaks by locating the high density areas that represent plane candidates with a similar normal direction. In distance space, candidate points are projected to the plane, that has average orientation, and these projections are used to calculate density and build a distribution graph. Peak detection based on Hill Climbing and Simulated Annealing is applied to extract single planes separately. Figure 1 is a schematic diagram of the proposed plane segmentation algorithm.

The remainder of this paper is organized as follows: the next section shows the robust estimation of surface normal vectors

based on largest triangles and clustering process in normal space based on peak detection in Sphere Coordinate. Planes with same normal vector are separated in Distance Space in Section 3. Section 4 shows the methodology to separate patches on same plane. The flexible size of sliding window for the building density graph is proposed in Section 5. Section 6 summarizes the experimental result.

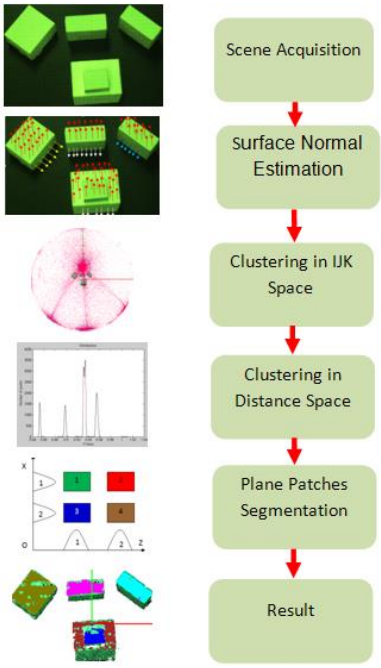


Figure 1.Plane segmentation algorithm overview

2. POINT CLUSTERING IN IJK SPACE

2.1 Surface normal estimation

Surface normal is the fundamental basis for extracting semantic information from 3D data. For a point ***P(x,y,z)*** in a 3D point cloud, fitting a plane to the point's local neighborhood is a conventional manner of determining the normal. The local neighborhood points can be defined by K, the nearest neighbors of ***P***, or points within a radius of ***P***. Both of these can work well for estimating the normal vector that represents the local surface at ***P***. There are different approaches for surface normal estimation, as shown in Figure 2.

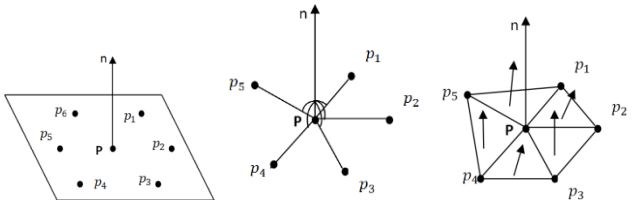


Figure 2. Different surface normal estimation approaches: (a) plane fitting; (b) Maximizing the angle between the normal vector and the tangential vectors; (c) Averaging the normal vectors of triangles.

Paper [6] provides a comparison between two types of surface normal estimations: optimization-based and averaging-based. The former works by solving the optimization problem:

$$\min_{n} J(P, NN, n)$$

Where ***NN*** is neighborhood, n is surface normal, and $J(P, NN, n)$ is a cost functional penalizing certain criteria.The result of a principal component analysis (PCA) or singular value decomposition (SVD) can be used to express the minimizer. The latter calculates the normal vector using a weighted average of the normal vectors of the triangles formed by **P** and the pairs of its neighbor. The basic averaging method is

$$n = \frac{1}{k}\sum_{i=1}^{k} w_i \frac{([p_i - P]\mathrm{X}\,[p_{i+1} - P]\,)}{|[p_i - P]\mathrm{X}\,[p_{i+1} - P]|}$$

where $p_i \in NN$ is the neighbor of P and $w_i = 1$.

They also indicate that, with the same initial condition, the optimization-based method can result in higher quality and require a higher computation time, but the averaging method can achieve an acceptable quality and should perform faster than the previous method.

We aspect to use the plane patches result to real application so we used the second method for our paper, but we make a difference approach for forming the tangential planes. Firstly, to reduce the intrinsic noise in the 3D point clouds, we performed Gaussian smoothing to remove the outlier. For each point **P,** we defined set of points collected by **N** by **N** mask on the image frame, as the neighbors of **P**. We undertook the following two-step process for surface normal estimation: 1) The four biggest triangles within the neighbors are formed around the set point, and the surface normal vector of each triangle is calculated; 2) By averaging the surface normal vectors of the triangles, the surface normal of the set point is determined. Figure 3 indicates that the conventional methods defined their N by N masks, either on the image frame of the 3D point cloud or through a nearest neighbor search. N by N mask can be 3x3, 5x5, 7x7 or 9x9 but the last one is the best for the trade-off between accuracy and computational efficiency. The two-step procedure for surface normal estimation turned out to be very robust against the noisy 3D point cloud. In terms of computational time and degree of accuracy, our proposed method performs faster and has a higher quality.

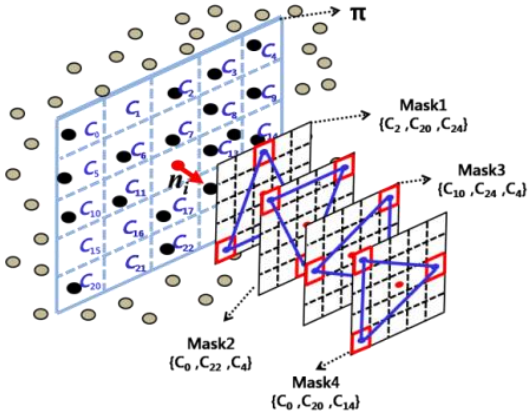


Figure 3. Four of the largest tangent plane masks

Surface normal vectors, of points on the 3D point cloud after estimation, are normalized and represented in a spherical coordinate (IJK). These unit vectors' lengths are one and so, they form points that lie on the unity sphere in IJK. Each point of the 3D point cloud corresponds to a point on the unity sphere in IJK space.

2.2 Clustering in the Sphere Coordinate

Points on a plane have similar surface normal vector so they form a high density area in the unit surface normal sphere. Figure 4 represents the distribution of the ideal plane in an IJK space. Multiple planes having similar orientation will reflect close distributions of points in IJK space. To extract the single planes from 3D point cloud, we will indicate location of high density areas in the IJK space (where each area represents multiple planes with similar orientation).

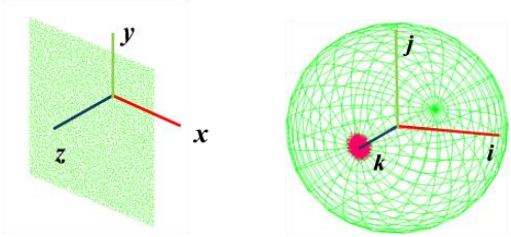


Figure 4. Distribution of a plane in the 3D (or IJK) space

In this subsection, we introduce a method to determine the high density areas using a peak detection algorithm. In the 2D image, LoG (Laplacian of Gaussian) is considered a good method for peak detection. In the 2D Cartesian coordinate system, Gaussian filter with standard deviation σ have the following form:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Thus, the LoG function takes the following form:

$$LoG(x,y) = \frac{1}{2\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

In our case, the data in the IJK Space is three-dimensional so applying LoG directly is difficult. Fortunately, the IJK Space consists of unit vectors. Therefore, by transferring the IJK Space into Spherical Coordinates $(r = 1, \theta, \varphi)$ where r = 1, we could obtain points distribution in a 2D Space.

For each point **P** on the spherical coordinates, there is only a pair of (θ, φ), where φ is the angle between the $\overrightarrow{OP}$&$\hat{K}$ directions, θ is the angle between $\hat{I}$&$\overrightarrow{OP'}$ and P' is the projection of point P on OIJ surface. If a point $P(x,y,z)$in the Cartesian coordinate system is defined, then the angles pair can also be defined as follows:

$$\varphi = \cos^{-1}\left(\frac{y}{r}\right)$$

$$\theta = \tan^{-1}\left(\frac{x}{z}\right)$$

$$\text{Where } r = \sqrt{x^2+y^2+z^2}$$

Each axis is divided into 180 cells so that each cell represents exactly 1 degree. By transforming from IJK space to spherical coordinates and counting the number of points in each cell, high density area in the IJK Space will form a high density in the Sphere Coordinate. A density 2D image was generated by normalizing using the function below:

$$I[x,y] = \frac{C[x,y] - min(C) + 1}{max(C) - min(C) + 1} * 255$$

where C[x, y] is the number of points at cell [x, y].

In the intensity image, the strong peaks with high density represent the large planes, and the weak peaks with lower density represent the small planes. Sometimes the intensity of the weak peaks is very small. Therefore, in order to detect all peaks—both strong and weak, we used a LoG to refine the image. First, a Gaussian filter is used to reduce the effect of noise. Thereafter, a Laplacian filter is used as a convolution filter to improve the peaks. When transferring to Spherical Coordinates, LoG can be defined

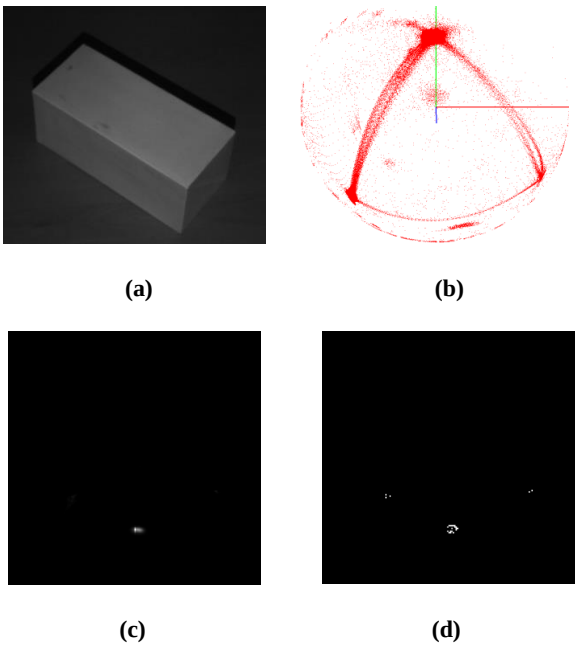
$$\nabla^2 f = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial f}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial f}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 f}{\partial\varphi^2}$$

where $f = G_g(\rho,\theta,\varphi)$.

Our input was represented as a set of discrete pixels, so we used some discrete convolution kernels that could approximate the second derivatives in the definition of the Laplacian. After many assessments, the following kernel was evaluated as an efficient mask:

0	−1	0
−1	4	−1
0	−1	0

Pixels with intensity greater than specified threshold are selected as peaks. Each high density pixel represents a set of 3D points with an almost similar surface normal. To find the 3D plane points among the peaks, we used a clustering method. Each chosen pixel was considered to be a cluster. The two peaks nearest in the mask KxK were merged. This was repeated until the number of clusters no longer changed. Aloof the clusters were then found. From the peaks in the clusters, close pixels belonging to a threshold were located and added to the cluster. Thereafter it was possible to search all the points in the 3D point cloud that had a similar orientation.



(a) **(b)** **(c)** **(d)**

Figure 5.(a)Original image; (b) Distribution in the IJK Space. (c) Intensity in Sphere Coordinate (d) Laplacian filter result

3. PLANE SEGMENTATION BY PEAK DETECTION

Until now, found plane candidates, from previous section, did not present a single plane but rather a set of planes with the same normal vector. For some applications, this information would be sufficient but, for others, they need to be spitted into a set of single planes. We are proposing a method for extracting planes based on the density in Distance Space. Figure 6 indicates an overview of the algorithm:

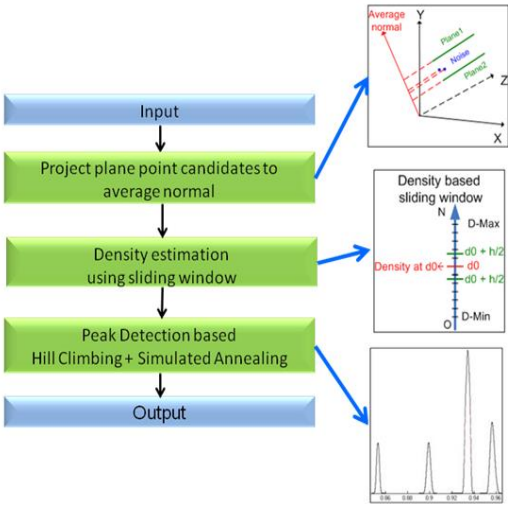


Figure 6. Plane segmentation Overview in Distance Space

We realized that the plane candidates were parallel to each other and that they have a similar normal vector. The average of the surface normal of the plane candidate could therefore be defined, and this direction is nearly perpendicular to the plane candidates, called average orientation. Assuming that the average orientation is $\vec{N}(a,b,c)$, a plane through point $P(x_0,y_0,z_0)$ has the equation:

$$ax + by + cz + d = 0\ (*)\ where\ d = -(ax_0 + by_0 + cz_0)$$

If $\vec{N}$ is the unit vector, it means that $a^2 + b^2 + c^2 = 1$, the distance from the Origin to this plane is *d*. Because the average orientation represented the surface normal of the plane point candidates, the plane equation of the points were different at ***d*** (the distance from the Origin). The two points P_1 and P_2 have the plane equation:

$$ax + by + cz + d_1 = 0$$
$$ax + by + cz + d_2 = 0$$

If P_1 and P_2 are on a plane, then d_1 and d_2 are similar. Alternatively, if P_1 and P_2 are on a different plane, then d_1 and d_2 are different. A 1-D space contains all ***d*** and is called Distance Space. A plane in 3D Space forms a high density area in Distance Space. Otherwise the noise points form a sparse density region. For that reason, based on point density, the planes could be extracted.

3.1 Density estimation

On the assumption that the ***N*** points $(P_1, P_2, \dots, P_N)$ in the 3D Space have the same orientation, we can define ***N*** equivalent points $(D_1, D_2, \dots, D_N)$ in Distance Space.To build the histogram graph of distribution, we used the sliding window method that is similar to the Parzen window method. Firstly, it is necessary to determine the D_{max} and D_{min} values. The **K–2** points are, then, put in the middle of the two extreme where they form **K-1** cells with the same width. Thereafter, a specified window with size **H** is formed, and moved from minimum to maximum to estimate the density of the **K-2** points and two extreme by counting the amount of points inside the window. A density graph or distribution graph is formed as a histogram graph. For the points inside the nearby projected areas of the 3D Space, the density was high, while in other areas it was low or almost equal to zero. There is a relationship between the window size and the fluctuation of the graph. A large window size generates a smooth distribution graph, but when it becomes too large, it provokes close planes to be merged. When a small window size is chosen, the high density area could be divided into multiple areas, which is not a good behavior as well. For this reason, in the next section we are proposing a method to obtain a good distribution graph.

Even when we provided an optimal window size, the density graph continued to oscillate at some intervals. We used Gaussian smoothing in one dimension to help the graph look more even. The one-dimensional Gaussian filter has a standard deviation as a parameter:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}$$

The standard deviation was chosen to be equal to **H/6**.

3.2 Peak Detection

By evaluating the graph, we realized that each plane formed a distribution like a Gaussian curve in the density graph, and this curve predominantly had only a local maximum, which was the global maximum of the curve. The global maximum of the curve represents that particular plane. However, the distribution of the projected points of the 3D plane on the Distance Space was not equal, so there was occasionally more than one peak per plane. Hill Climbing is known to be one of the most convenient methods for peak detection. All local optimal points were found, but the Hill Climbing or Down Hill process usually got stuck at these points.

A plane should be represented by a global peak instead of local peaks. To overcome this obstacle, we applied Simulated Annealing (SA) to jump over the local optimal point. Simulated Annealing is a generic probabilistic meta-heuristic for the global optimization problem, that locates a good approximation to the global optimum of a given function in a large search space. The original SA is modified as Table 1:

Table 1.SA algorithm

Input: ***K*** points represented by $(D_1, D_2, \dots, D_K)$ and density $(F_1, F_2, \dots, F_K)$correlatively. An initial D_0 point. **Output:** New optimal point
Initial step: Set 4 variables D← D_0; f← Density(D); dBest ← D; fBest ← f; k ← 0 **While** k < kmax **Step 1:** T ← temp (k); dNew← neighbor(D); fNew← Density(D); **Step 2**: ***If*** P(f,fNew,T) > random() ***Then***d← dNew; f← fNew **Step 3**: ***If*** f < fBest ***Then*** dBest ← dNew; fBest ← fNew k++; **Return** dBest.

Where ***dBest*** is the point with the best density value Density(***fBest)***.This algorithm applied for going up, but when going down, a small change in Step 3, from ***f*** < ***fBest*** to ***f*** > ***fBest***, was required. The function ***random()*** generated a random number between 0 and 1. There were some items that needed to be carefully considered: temperature function ***temp (k)***, neighbor function ***neighbor(y)***, and probability ***P(f,fNew,T)***. The general equation for the probability function was defined as***P = exp (- ΔE / k_B.T)*** where ***ΔE*** is the energy difference and k_B is the Boltzmann constant. But in this paper, ***P(f,fNew,T)= exp((-|fNew-f|)/T)*** where **T** is defined by ***temperature(k)***.

$T = temp(k) = T0 * \theta^k$ where $T0 = Density(D_0)$ $\theta \approx 1$. Ordinarily, we would choose θ to be equal to 0.9, and ***kmax*** to be 20. ***Neighbor(D)*** generated the value in $[D - \mathbf{H}/4, D + \mathbf{H}/4]$ randomly, where **H** is the window size.

Density is a one-dimensional discrete function. Therefore, by applying Hill Climbing, the local optimal point could be easily detected, but it usually gets stuck at that point. Consequently, the application of SA only occurred at the local maximum and local minimum where Hill Climbing stops. If SA overcomes the local optimal point, then Hill Climbing would continue its process. Otherwise, the local optimal point was considered to be a good peak that represents the plane. Table 2 shows our algorithm:

Table 2.Hill Climbing and Simulated Annealing Mixture

Input: K points represented by $(D_1, D_2, \dots, D_K)$and density $(F_1, F_2, \dots, F_K)$correlatively **Output:** A set of intervals, each interval contains three values: start point, end point, and global optimal point.
Step 1: Find separated intervals so that the density at the start point is equal to zero; the density will increase to the global peak and then decrease to zero again. **Step 2**: From the start point (Density = 0), using ***Hill Climbing***, go up and stop at the local maximum. **Step 3**: Apply ***Simulated Annealing***. If the maximum density does not change, that is the global minimum, go to **Step 4**, else go back to **Step 2**. **Step 4**: From the global optimal point, go down using ***Down Hill*** and stop at the local minimum. **Step 5**: Apply ***Simulated Annealing***. Stop if the minimum density does not change, else goes to **Step 4**.

Because of noise, there are many low density peaks on the density graph. We specified a threshold to remove these peaks: only peaks with a density greater than the threshold were chosen.

4. PLANE PATCHES SEGMENTATION

Until now, planes are separated, but in each single plane, there is more than one patch. Different patches belong to different object ordinarily. To support for modeling and recognition, these patches should be separated into single one. Some approaches were proposed that could provide exact results. DBSCAN [7] is efficient method for clustering which discovers arbitrary shape based on the local density in Eps-neighbor. However, it took high computation time for implement. In [8] a clustering method for efficient segmentation based on radically bounded nearest neighbor (RBNN) strategy was proposed that can detect flat boxes exactly. Kd-trees structure was used to reduce time for searching nearest neighbor. Therefore, its average computational complexity was still around $O(\frac{n}{k_{average}}\log(n))$ for looking where $k_{average}$ represented the average number of neighbors over all queries so when applying to data with number of points larger than 50,000 would take some seconds for perform. Subspace clustering is method working based on projecting data to axes. It is not optimal solution for getting precise clusters however for high dimensional data it could be used because it can reduce computational time and the segmentation result almost is exact when shapes of patches are simple.

For found planes from previous step, a transformation is able to make to reduce data from three dimensions to two dimensions by rotating planes perpendicular to XOZ, it means that y value of points on plane are the same. Subspace clustering would be applied for splitting patches on same planar surface. Plane points would be projected on OX and OZ. Consider points on each axis, they are one dimensional data and a histogram graph is built by counting number of points in equal bins. Consecutive bins having positive number of points should be merged to same cluster. Clusters with number of points bigger than a threshold are chosen otherwise they are considered as noise and should be removed. A clustering process based on clusters on two axes should be done to detect patches. If two points on a plane belong to the same cluster in OX and same clustering in OZ then they should belong to a same patch. Otherwise they should belong to different patches.

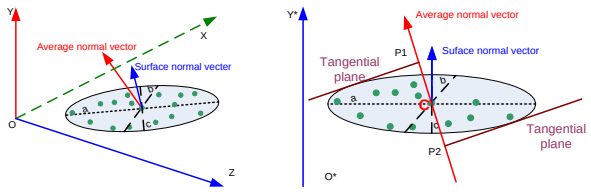
For this step, the computational complexity for each axis is ***O(n)*** and ***O(2n)*** for total, so it can apply in real time application. The experimental result will show result.

5. WINDOW SIZE ESTIMATION

From our experience, changing the window size affects the shape of the density graph directly. Choosing too small a value generates a fluctuant diagram, but choosing a value that is too large causes some close planes to merge. A fixed value can work well in some cases but is worse in others. An automatic search for an optimal window size in a practical case was required. Therefore, in this paper, we provide a solution for estimating window size based on the error distribution of the planes on distance space.

5.1 Error distribution in Distance Space

After acquiring the scene, a 3D point cloud was obtained. Even though the environment was ideal, there was an error within each point itself. The surface normal vector of a point in a plane was not the same as the average normal vector representing the perpendicular direction of the plane. Hence an error became apparent when we projected the points on this average orientation. It is known that an error distribution in a 3D point cloud is presented in the form of an ellipsoid. The equation of an error ellipsoid can be computed by the neighborhood around the center point. Figure 7 shows the error distribution of the 3D point and the method to calculate this error.



(a) (b)

Figure 7.(a)3D point cloud error distribution; (b) Error distribution estimation by finding the furthest tangential plane

To calculate the error, we used the neighborhood to calculate the ellipsoid parameters ***(a,b,c)***, and then projected the ellipsoid to obtain the average orientation. The two furthest points can be defined by the two farthest tangential planes that have a planar normal parallel with an average normal vector. It is convenient for error computation if the rotation is made where the surface normal vector of the center point is parallel with an axis, for example, OY. On the assumption that the new average normal vector is ***(nI,nJ,nK)***, the error can be defined as follows:

$$r = \sqrt{nI^2a^2 + nJ^2b^2 + nK^2c^2}$$

The error of all points for error distribution was averaged on the Distance Space. Following many tests, the error did not change greatly and remained around 2 mm. We could therefore predefine it as an error distribution in performance.

5.2 Error distribution of planes

For each 3D point, there exists a projected point on average orientation. In ideal case, all points on a plane projected on average orientation at only point. In reality, they form density area; and half of the width of this area is called the planar error distribution. To estimate this error, we proposed a clustering method to find the high density area. Figure 8 indicates the distribution of the plane points on the average orientation. The closest distance of a point in a cluster is small, whereas the closest distance of a noise point is large. The difference could be applied by separating the points into clusters.

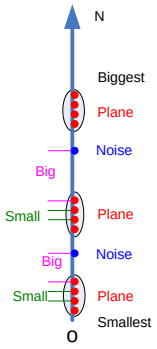


Figure 8. Plane point distribution in Distance Space

The projected points on average orientation are set in order of smallest to biggest using a quick sort algorithm and applied on an increasing scale. Clustering is based on a simple rule: if the distance between two consecutive points is smaller than the threshold, they should be placed in a cluster. In this way, clusters are formed, and clusters with a number of points greater than a threshold are chosen for the next step, while the remaining clusters are considered as noise. Next, the width is computed by subtracting the maximum value from the minimum value. These widths are considered to be the error distribution of the planes.

We used this cluster method because it is simple and not much time is required for a complex computation. Rather, ***O(nlog(n))*** is used for a quick sort and ***O(n)*** for grouping. The threshold used for the cluster mentioned previously is the error distribution computed and predefined in the above section.

5.3 Estimating window size

Until now, we were able to determine clusters, but they had different widths. These widths depended on the size of the plane and the angle between the average normal vector and the planar normal. The angle error was around five degrees. The width of the plane in the Distance Space was called the error distribution of the plane. Figure 9 shows the problem.

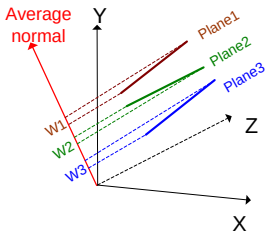


Figure 9. W_1,W_2,W_3 are the plane widths.

For a cluster with a width **W**, the window size chosen should be **W**. In the case of many clusters, the window size should be chosen by the average of the widths.

$$H = \frac{1}{N}\sum W_i \quad N\text{: amount of clusters}$$

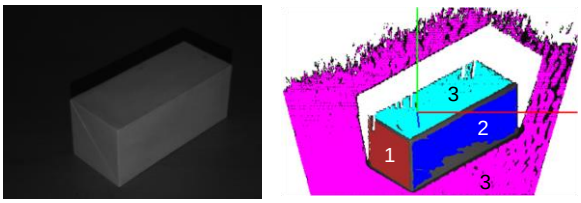
6. EXPERIMENTAL RESULTS

In this section, we evaluated our algorithm via several experiments. The algorithm was composed of a desktop computer that had the configuration: Intel Core 2 E6600 2.4GHz CPU and 2GB RAM. We evaluated the algorithm at each step of the process by measuring the computation time in the second unit. We showed the original image, segmentation result, and computation time for each test case. For each orientation, angle deviation also is shown.

The first case comprised a rectangular object with three planes, and each pair was perpendicular to each other. Figure 10 indicates the results. Each color represents a plane. Two planes were marked number 3, which means that they are parallel to each other. All planes were detected correctly, but there were some missing points. The missing points usually have bad surface normals, which meant that the angles between the surface normals and the average orientation were outside the specified limits. The result of the clustering in IJK is shown in Table 3 and Table 4 shown the computation time for each direction:

Table 3. Orientation information for Case 1.

Orientation	**1**	**2**	**3**
Number of points	25,604	12,378	137,132
Angle deviation	1.9±1.0	1.2±1.2	1.6±1.3



Original Image **Segmentation Result**

Figure 10. The origin and segmentation result of Case 1

Table 4. Computation time for Case 1

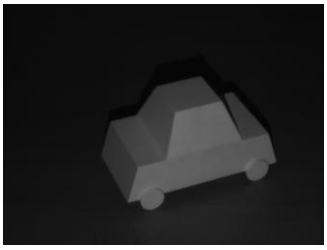
Process	**Computation time (sec)**		
Clustering in IJK Space	2.453		
Orientation	**1**	**2**	**3**
Clustering in Distance Space	0.140	0.094	0.578
Plane patches segmentation	0.281	0.156	1.500
Sum	**0.421**	**0.250**	**2.078**
Total	**5.202**		

The second case is a model car. There are four orientations in total. The clustering result in IJK is shown in Table 5. In Orientation 1, there are two planes, and two wheels are considered in each plane. In Orientation 4, there are three separate planes, and two red plane patches are in one plane.

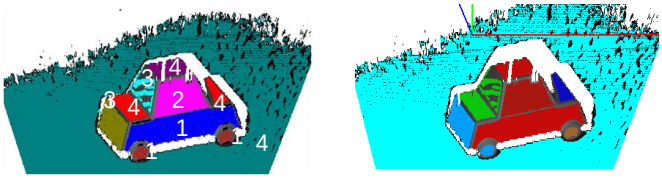
Table 5. Orientation information for Case 2

Orientation	1	2	3	4
Number of points	13,643	8,803	6,297	156,834
Angle deviation	1.2±1	1.4±1.4	3.4±4.3	1.6±1.2

Figure 11 shows the segmentation results. All of the planes on the car can be detected even though there are two wheel areas. In Orientation 4, two red patches are considered on same plane and shown in picture (b) after clustering in Distance Space process. However, these patches are detected as separated patches in next step. The same result also occurs in Orientation 1. Picture (c) shows this result.



(a)Original Image



b)Clustering in Distance **(c)Plane Patch Segmentation**

Figure 11. The origin and segmentation results of Case 2

Table 6. Computation time for Case 2

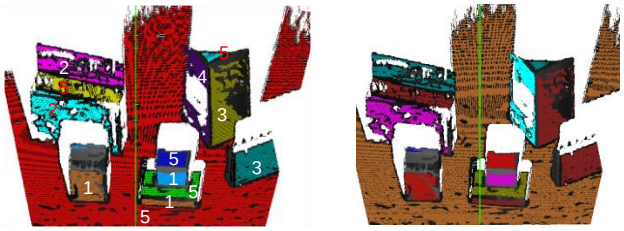
Process	Computation time (sec)			
Clustering in IJK Space	2422			
Orientation	**1**	**2**	**3**	**4**
Clustering in Distance Space	0.078	0.077	0.063	0.671
Plane patches segmentation	0.172	0.110	0.078	1.844
Sum	0.250	0.187	0.141	2.515
Total	**4.062**			

In the next test, Case 3 contained a set of objects. There were five orientations, and Table 6 indicates the information about these orientations after clustering in the IJK Space.

Table 7. Orientation information for Case 3

Orientation	1	2	3	4	5
Number of points	11,525	13,259	16,488	6,526	106,694
Angle deviation	3.5±1.8	3.9±1.9	2.8±1.6	3.6±3.5	2.1±1.9

(a)Original Image



(b)Clustering in Distance **(c) Plane Patch Segmentation**

Figure 12. The origin and segmentation results of Case 3

Figure 12 show the test result of Case 3. Each orientation has many planes. For example, Orientation 5 has five separate planes while Orientation 1 has three plane patches but on two planes. Two patches with the same color are on a single plane is shown is (b) but after plane patches process they are separated into single patches. The computation time for running the test for Case 3 is shown in Table 8.

Table 8. Computation time for Case 3

Process	Computation time (sec)				
Clustering in IJK Space	2.451				
Orientation	**1**	**2**	**3**	**4**	**5**
Clustering in Distance Space	0.109	0.094	0.092	0.063	0.469
Plane patches segmentation	0.125	0.141	0.187	0.078	1.266
Sum	**0.234**	**0.235**	**0.279**	**0.141**	**1.735**
Total	**5.075**				

After many experimental tests, it was realized that the time for clustering in the Distance Space was around one second, plane patches segmentation took two second and the total was around five seconds. The computation time for extracting the plane in Distance Space is directly proportional to the number of points. There are a huge number of points on back ground, removing back group helps reducing computation time much.

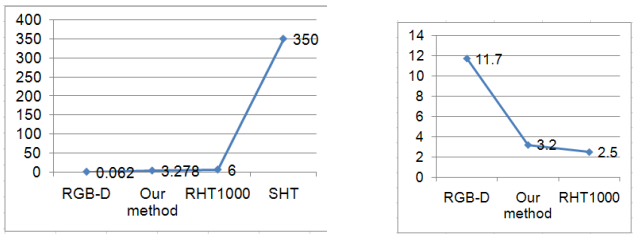
7. CONCLUSION&FUTURE WORK

In this paper, we proposed a new algorithm for plane segmentation in a 3D point cloud. We presented a method for estimating the error distribution of points in a direction as well as a method for window size estimation. A peak detection algorithm in Distance Space was also shown. The results demonstrate that both accuracy and runtime were within acceptable operational limits for our system and can be compared to recently proposed approaches as follows:

Table 9.Comparison with other methods.

Methods	Scene / Number of Points	Computation Time	Angle Deviation
[7]	Tabletop with an object	62 ms	11.7 ± 3deg
[3]	Cube 10000 points for each plane	SHT:350s RHT1000:6s	RHT6000 2.5±1 deg
Presented approach with plane patches segmentation	5 cluttered objects with 5 orientation of surfaces / 154,492 points	5,075 ms	3.2 ± 2 deg
Presented approach without plane patches segmentation		3.278ms	

Compare to [7], our proposed method could detection planar patches with high accuracy. However, because of high resolution and improved process for high quality so computation time takes longer. Compare to [3], Hough Transform usually take longer than our solution even though apply random Hough Transform (RHT) and tested on a cube. Increasing the number of cells in accumulator of RHT makes a increasing of the complexity and quality. For RHT 1000 it takes 6 second bigger two times (3.2) comparing with our method. So for RHT 6000, the computation time will be greater but it can make a litter bit higher accuracy. However, the angle deviation we shown here was estimated error for all direction not for a patch. The angle deviation for each patch can be higher accuracy than the RHT6000. For instance, Orientation 1 and 2 of Case 1, the angle deviation are 1.9 ± 1 and 1.2 ± 1.2 smaller than 2.5 ± 1. In [3] they also indicate the complexity is subsidiary to the number of plane in data. The following table shows the comparison.



(a)

(b)

Figure 13.(a) Computation Time(s). (b) Angle Deviation

This low angle deviation was achieved by combining 3 methodologies in peak detection and segmentation: - Laplacian of Gaussian (LoG) was applied for accurate peak detection of candidate planes in Sphere Coordinate.

- The generic probabilistic meta-heuristic, Simulated Annealing is also used for segmentation of planes with local peaks. -Dynamically-sized sliding window for smooth density supporting for estimation of segmented planes.

And the fast performance was achieved by reducing the rank of the search space from 3 (XYZ, IJK) into 2 (θ,φ) and applying Laplacian of Gaussian instead of traditional iterative peak detectors.

In future, we will endeavor to improve the method for searching for the best orientation instead for average normal to improve density graph in Distance Space and reduce the runtime for the application to a real-time system.

8. ACKNOWLEDGMENTS

This research was supported by WCU(World Class University) program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology(R31-2011-000-10062-0), by the KORUS-Tech program (kt-2010-SW-AP-FS0-0004). This paper was provided funding by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2012-008188). This paper also was funded by PRCP through NRF of Korea funded by MEST (2011-0018397), by MKE, Korea under ITRCNIPA-2010-(C1090-1021-0008)(NTIS-2010-(1415109527)) and by the Intelligent Robotics Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy of Korea(F0005000-2010-32). Finally, The Han Shin Power Tech Laboratory also provided funding and expertise for cooperation.

9. REFERENCES

[1] Okada, K.,Kagami, S.,Inaba, M., and Inoue, H. 2001 . Plane segment finder: Algorithm, implementation and applications. In IEEE International Conference on Robotics and Automation. IEEE Computer Society, 2120-2125.

[2] Borrmann, D. Elseberg, J. Lingemann, K. Nuchter,A. 2011. The 3D Hough Transform for Plane Detection in Point Clouds:A Review and a new Accumulator Design *3D* Res.2, 2 *(*Mar. 2011*)*, 32:1–32:13

[3] Piazzi, J. and Prattichizzo, D. 2006. Plane detection with stereoimages.In IEEE International Conference on Robotics andAutomation. IEEE Computer Society,922-927.

[4] Silveira, G., Malis, E., and Rives, P.2006. Real-Time Robust Detection of Planar Regions in a Pair of Images. In InternationalConference on Intelligent Robots and Systems. IEEE Computer Society, 49-54.

[5] Holz, D. Holzer, S.,Rusu, R.B., and Behnke, S. 2011. Real-Time Plane Segmentation using RGB-D Cameras. In Proceedings of 15thRoboCup International Symposium, 306-317.

[6] Klasing, K. Althoff, D. Wollherr, D. Buss, M. 2009. Comparison of surface normal estimation methods for range sensing applications. Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA) , Kobe, Japan 1977-1982

[7] Ester, M. Kriegel, M. P. Sander, J. Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96).226-231

[8] Klasing, K. Wollherr, D. Buss, M. 2008. A clustering method for efficient segmentation of 3D Laser Data.International Conference on Robotics and Automation