A Fast and Accurate Plane Detection Algorithm for Large Noisy Point Clouds Using Filtered Normals and Voxel Growing

Jean-Emmanuel Deschaud, François Goulette

A Fast and Accurate Plane Detection Algorithm for Large Noisy Point Clouds Using Filtered Normals and Voxel Growing

Jean-Emmanuel Deschaud
François Goulette
Mines ParisTech, CAOR-Centre de Robotique, Mathématiques et Systèmes
60 Boulevard Saint-Michel 75272 Paris Cedex 06
jean-emmanuel.deschaud@mines-paristech.fr francois.goulette@mines-paristech.fr

Abstract

With the improvement of 3D scanners, we produce point clouds with more and more points often exceeding millions of points. Then we need a fast and accurate plane detection algorithm to reduce data size. In this article, we present a fast and accurate algorithm to detect planes in unorganized point clouds using filtered normals and voxel growing. Our work is based on a first step in estimating better normals at the data points, even in the presence of noise. In a second step, we compute a score of local plane in each point. Then, we select the best local seed plane and in a third step start a fast and robust region growing by voxels we call voxel growing. We have evaluated and tested our algorithm on different kinds of point cloud and compared its performance to other algorithms.

1. Introduction

With the growing availability of 3D scanners, we are now able to produce large datasets with millions of points. It is necessary to reduce data size, to decrease the noise and at same time to increase the quality of the model. It is interesting to model planar regions of these point clouds by planes. In fact, plane detection is generally a first step of segmentation but it can be used for many applications. It is useful in computer graphics to model the environnement with basic geometry. It is used for example in modeling to detect building facades before classification. Robots do Simultaneous Localization and Mapping (SLAM) by detecting planes of the environment. In our laboratory, we wanted to detect small and large building planes in point clouds of urban environments with millions of points for modeling. As mentioned in [6], the accuracy of the plane detection is important for after-steps of the modeling pipeline. We also want to be fast to be able to process point clouds with millions of points. We present a novel algorithm based on region growing with improvements in normal estimation and growing process. For our method, we are generic to work on different kinds of data like point clouds from fixed scanner or from Mobile Mapping Systems (MMS). We also aim at detecting building facades in urban point clouds or little planes like doors, even in very large data sets. Our input is an unorganized noisy point cloud and with only three "intuitive" parameters, we generate a set of connected components of planar regions. We evaluate our method as well as explain and analyse the significance of each parameter.

2. Previous Works

Although there are many methods of segmentation in range images like in [10] or in [3], three have been thoroughly studied for 3D point clouds : region-growing, hough-transform from [14] and Random Sample Consensus (RANSAC) from [9].

The application of recognising structures in urban laser point clouds is frequent in literature. Bauer in [4] and Boulaassal in [5] detect facades in dense 3D point cloud by a RANSAC algorithm. Vosselman in [23] reviews surface growing and 3D hough transform techniques to detect geometric shapes. Tarsh-Kurdi in [22] detect roof planes in 3D building point cloud by comparing results on hough-transform and RANSAC algorithm. They found that RANSAC is more efficient than the first one. Chao Chen in [6] and Yu in [25] present algorithms of segmentation in range images for the same application of detecting planar regions in an urban scene. The method in [6] is based on a region growing algorithm in range images and merges results in one labelled 3D point cloud. [25] uses a method different from the three we have cited : they extract a hierarchical subdivision of the input image built like a graph where leaf nodes represent planar regions.

There are also other methods like bayesian techniques. In [16] and [8], they obtain smoothed surface from noisy point clouds with objects modeled by probability distribu-

tions and it seems possible to extend this idea to point cloud segmentation. But techniques based on bayesian statistics need to optimize global statistical model and then it is difficult to process points cloud larger than one million points.

We present below an analysis of the two main methods used in literature : RANSAC and region-growing. Hough-transform algorithm is too time consuming for our application. To compare the complexity of the algorithm, we take a point cloud of size N with only one plane P of size n. We suppose that we want to detect this plane P and we define n_{min} the minimum size of the plane we want to detect. The size of a plane is the area of the plane. If the data density is uniform in the point cloud then the size of a plane can be specified by its number of points.

2.1. RANSAC

RANSAC is an algorithm initially developped by Fischler and Bolles in [9] that allows the fitting of models without trying all possibilities. RANSAC is based on the probability to detect a model using the minimal set required to estimate the model.

To detect a plane with RANSAC, we choose 3 random points (enough to estimate a plane). We compute the plane parameters with these 3 points. Then a score function is used to determine how the model is good for the remaining points. Usually, the score is the number of points belonging to the plane. With noise, a point belongs to a plane if the distance from the point to the plane is less than a parameter γ. In the end, we keep the plane with the best score. The probability of getting the plane in the first trial is $p = (\frac{n}{N})^3$. Therefore the probability to get it in T trials is $p = 1 - (1 - (\frac{n}{N})^3)^T$. Using equation 1 and supposing $\frac{n_{min}}{N} \ll 1$, we know the number T_{min} of minimal trials to have a probability p_t to get planes of size at least n_{min} :

$$T_{min} = \frac{log(1-p_t)}{log(1-(\frac{n_{min}}{N})^3)} \approx log(\frac{1}{1-p_t})(\frac{N}{n_{min}})^3. \quad (1)$$

For each trial, we test all data points to compute the score of a plane. The RANSAC algorithm complexity lies in $O(N(\frac{N}{n_{min}})^3)$ when $\frac{n_{min}}{N} \ll 1$ and $T_{min} \to 0$ when $n_{min} \to N$. Then RANSAC is very efficient in detecting large planes in noisy point clouds i.e. when the ratio $\frac{n_{min}}{N}$ is $\simeq 1$ but very slow to detect small planes in large point clouds i.e. when $\frac{n_{min}}{N} \ll 1$. After selecting the best model, another step is to extract the largest connected component of each plane. Connnected components mean that the minimum distance between each point of the plane and others points is smaller (for distance) than a fixed parameter.

Schnabel *et al.* [20] bring two optimizations to RANSAC : the points selection is done locally and the score function has been improved. An octree is first created from point cloud. Points used to estimate plane parameters are chosen locally at a random depth of the octree. The score function is also different from RANSAC : instead of testing all points for one model, they test only a random subset and find the score by interpolation. The algorithm complexity lies in $O(\frac{N}{r}\frac{4Nd}{n_{min}})$ where r is the number of random subsets for the score function and d is the maximum octree depth. Their algorithm improves the planes detection speed but its complexity lies in $O(N^2)$ and it becomes slow on large data sets. And again we have to extract the largest connected component of each plane.

2.2. Region Growing

Region Growing algorithms work well in range images like in [18]. The principle of region growing is to start with a seed region and to grow it by neighborhood when the neighbors satisfy some conditions. In range images, we have the neighbors of each point with pixel coordinates. In case of unorganized 3D data, there is no information about the neighborhood in the data structure. The most common method to compute neighbors in 3D is to compute a Kd-tree to search k nearest neighbors. The creation of a Kd-tree lies in $O(NlogN)$ and the search of k nearest neighbors of one point lies in $O(logN)$. The advantage of these region growing methods is that they are fast when there are many planes to extract, robust to noise and extract the largest connected component immediately. But they only use the distance from point to plane to extract planes and like we will see later, it is not accurate enough to detect correct planar regions .

Rabbani *et al.* [19] developped a method of smooth area detection that can be used for plane detection. They first estimate the normal of each point like in [13]. The point with the minimum residual starts the region growing. They test k nearest neighbors of the last point added : if the angle between the normal of the point and the current normal of the plane is smaller than a parameter α then they add this point to the smooth region. With Kd-tree for k nearest neighbors, the algorithm complexity is in $O(N + nlogN)$. The complexity seems to be low but in worst case, when $\frac{n}{N} \simeq 1$, example for facade detection in point clouds, the complexity becomes $O(NlogN)$.

3. Voxel Growing

3.1. Overview

In this article, we present a new algorithm adapted to large data sets of unorganized 3D points and optimized to be accurate and fast. Our plane detection method works in three steps. In the first part, we compute a better estimation of the normal in each point by a filtered weighted plane fitting. In a second step, we compute the score of local planarity in each point. We select the best seed point that represents a good seed plane and in the third part, we grow this seed plane by adding all points close to the plane. The

growing step is based on a voxel growing algorithm. The filtered normals, the score function and the voxel growing are innovative contributions of our method.

As an input, we need dense point clouds related to the level of detail we want to detect. As an output, we produce connected components of planes in the point cloud. This notion of connected components is linked to the data density. With our method, the connected components of planes detected are linked to the parameter d of the voxel grid.

Our method has 3 "intuitive" parameters : d, $area_{min}$ and γ. "intuitive" because there are linked to physical measurements. d is the voxel size used in voxel growing and also represents the connectivity of points in detected planes. γ is the maximum distance between the point of a plane and the plane model, represents the plane thickness and is linked to the point cloud noise. $area_{min}$ represents the minimum area of planes we want to keep.

3.2. Details

3.2.1 Local Density of Point Clouds

In a first step, we compute the local density of point clouds like in [17]. For that, we find the radius r_i of the sphere containing the k nearest neighbors of point i. Then we calculate $\rho_i = \frac{k}{\pi r_i^2}$. In our experiments, we find that $k = 50$ is a good number of neighbors. It is important to know the local density because many laser point clouds are made with a fixed resolution angle scanner and are therefore not evenly distributed. We use the local density in section 3.2.3 for the score calculation.

3.2.2 Filtered Normal Estimation

Normal estimation is an important part of our algorithm. The paper [7] presents and compares three normal estimation methods. They conclude that the weighted plane fitting or WPF is the fastest and the most accurate for large point clouds. WPF is an idea of Pauly and al. in [17] that the fitting plane of a point p must take into consideration the nearby points more than other distant ones. The normal least square is explained in [21] and is the minimum of $\sum_{i=1}^{k}(\boldsymbol{n_p} \cdot p_i + d)^2$. The WPF is the minimum of $\sum_{i=1}^{k} \omega_i \ (\boldsymbol{n_p} \cdot p_i + d)^2$ where $\omega_i = \theta(\|p_i - p\|)$ and $\theta(r) = e^{-2\frac{r^2}{r_i^2}}$. For solving $\boldsymbol{n_p}$, we compute the eigenvector corresponding to the smallest eigenvalue of the weighted covariance matrix $C_w = \sum_{i=1}^{k} \omega_i \, {}^t(p_i - b_w)(p_i - b_w)$ where b_w is the weighted barycenter. For the three methods explained in [7], we get a good approximation of normals in smooth area but we have errors in sharp corners. In figure 1, we have tested the weighted normal estimation on two planes with uniform noise and forming an angle of 90 °. We can see that the normal is not correct on the corners of the planes and in the red circle.

To improve the normal calculation, that improves the plane detection especially on borders of planes, we propose a filtering process in two phases. In a first step, we compute the weighted normals (WPF) of each point like we described it above by minimizing $\sum_{i=1}^{k} \omega_i \ (\boldsymbol{n_p} \cdot p_i + d)^2$. In a second step, we compute the filtered normal by using an adaptive local neighborhood. We compute the new weighted normal with the same sum minimization but keeping only points of the neighborhood whose normals from the first step satisfy $| \boldsymbol{n_p} \cdot \boldsymbol{n_i} | > cos(\alpha)$. With this filtering step, we have the same results in smooth areas and better results in sharp corners. We called our normal estimation filtered weighted plane fitting (FWPF).

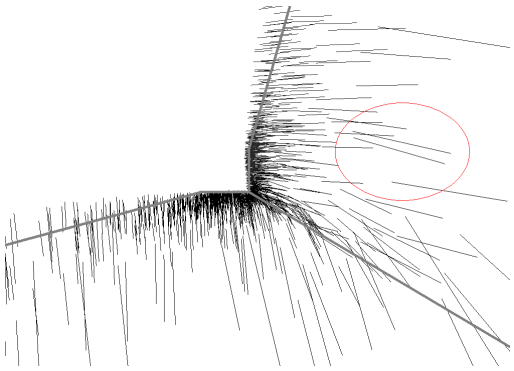

Figure 1. Weighted normal estimation of two planes with uniform noise and with 90 ° angle between them.

We have tested our normal estimation by computing normals on synthetic data with two planes and different angles between them and with different values of the parameter α. We can see in figure 2 the mean error on normal estimation for WPF and FWPF with $\alpha = 20\,^\circ, 30\,^\circ, 40\,^\circ$ and $90\,^\circ$. Using $\alpha = 90\,^\circ$ is the same as not doing the filtering step. We see on Figure 2 that $\alpha = 20\,^\circ$ gives smaller error in normal estimation when angles between planes is smaller than $60\,^\circ$ and $\alpha = 30\,^\circ$ gives best results when angle between planes is greater than $60\,^\circ$. We have considered the value $\alpha = 30\,^\circ$ as the best results because it gives the smaller mean error in normal estimation when angle between planes vary from $20\,^\circ$ to $90\,^\circ$. Figure 3 shows the normals of the planes with $90\,^\circ$ angle and better results in the red circle (normals are $90\,^\circ$ with the plane).

3.2.3 The score of local planarity

In many region growing algorithms, the criteria used for the score of the local fitting plane is the residual, like in [18] or [19], i.e. the sum of the square of distance from points to the plane. We have a different score function to estimate local planarity. For that, we first compute the neighbors N_i of a point p with points i whose normals $\boldsymbol{n_i}$ are close to

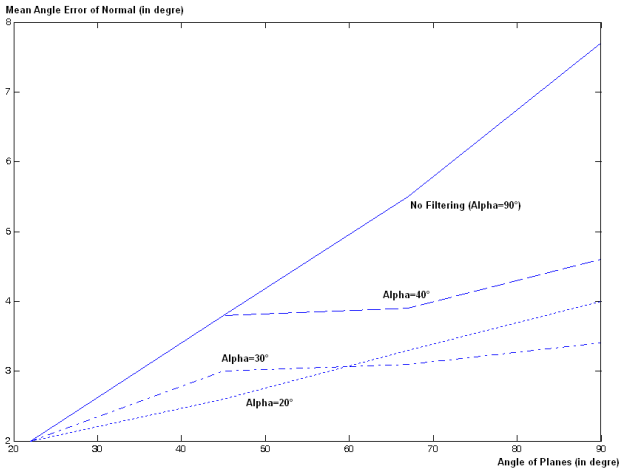

Figure 2. Comparison of mean error in normal estimation of two planes with $\alpha = 20^\circ, 30^\circ, 40^\circ$ and 90° (=No filtering).

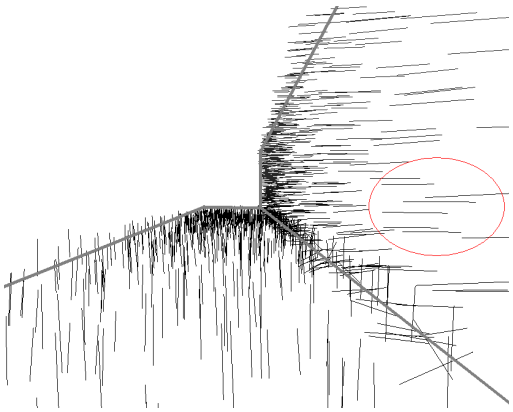

Figure 3. Filtered Weighted normal estimation of two planes with uniform noise and with 90° angle between them ($\alpha = 30^\circ$).

the normal $\boldsymbol{n_p}$. More precisely, we compute $N_i = \{p$ in k neighbors of $i / \mid \boldsymbol{n_i} \cdot \boldsymbol{n_p} \mid > cos(\alpha)\}$. It is a way to keep only the points which are probably on the local plane before the least square fitting. Then, we compute the local plane fitting of point p with N_i neighbors by least squares like in [21]. The set N_i' is a subset of N_i of points belonging to the plane, i.e. the points for which the distance to the local plane is smaller than the parameter γ (to consider the noise). The score s of the local plane is the area of the local plane, i.e. the number of points "in" the plane divided by the local density ρ_i (seen in section 3.2.1) : the score $s = \frac{card(N_i')}{\rho_i}$. We take into consideration the area of the local plane as the score function and not the number of points or the residual in order to be more robust to the sampling distribution.

3.2.4 Voxel decomposition

We use a data structure that is the core of our region growing method. It is a voxel grid that speeds up the plane detection process. Voxels are small cubes of length d that partition the point cloud space. Every point of data belongs to a voxel and a voxel contains a list of points. We use the Octree Class Template in [2] to compute an Octree of the point cloud. The leaf nodes of the graph built are voxels of size d. Once the voxel grid has been computed, we start the plane detection algorithm.

3.2.5 Voxel Growing

With the estimator of local planarity, we take the point p with the best score, i.e. the point with the maximum area of local plane. We have the model parameters of this best seed plane and we start with an empty set E of points belonging to the plane. The initial point p is in a voxel v_0. All the points in the initial voxel v_0 for which the distance from the seed plane is less than γ are added to the set E. Then, we compute new plane parameters by least square refitting with set E.

Instead of growing with k nearest neighbors, we grow with voxels. Hence we test points in 26 voxel neighbors. This is a way to search the neighborhood in constant time instead of $O(logN)$ for each neighbor like with Kd-tree. In a neighbor voxel, we add to E the points for which the distance to the current plane is smaller than γ and the angle between the normal computed in each point and the normal of the plane is smaller than a parameter α: $\mid cos(\boldsymbol{n_p}, \boldsymbol{n_P}) \mid > cos(\alpha)$ where $\boldsymbol{n_p}$ is the normal of the point p and $\boldsymbol{n_P}$ is the normal of the plane P. We have tested different values of α and we empirically found that 30° is a good value for all point clouds. If we added at least one point in E for this voxel, we compute new plane parameters from E by least square fitting and we test its 26 voxel neighbors. It is important to perform plane least square fitting in each voxel adding because the seed plane model is not good enough with noise to be used in all voxel growing, but only in surrounding voxels. This growing process is faster than classical region growing because we do not compute least square for each point added but only for each voxel added.

The least square fitting step must be computed very fast. We use the same method as explained in [18] with incremental update of the barycenter b and covariance matrix C like equation 2. We know with [21] that the barycenter b belongs to the least square plane and that the normal of the least square plane $\boldsymbol{n_P}$ is the eigenvector of the smallest eigenvalue of C.

$$
\begin{aligned}
&b_0 = 0_{3x1} \quad C_0 = 0_{3x3}. \\
&b_{n+1} = \frac{1}{n+1}(nb_n + p_{n+1}). \\
&C_{n+1} = C_n + \frac{n}{n+1}{}^t(p_{n+1} - b_n)(p_{n+1} - b_n).
\end{aligned} \tag{2}
$$

where C_n is the covariance matrix of a set of n points, b_n is the barycenter vector of a set of n points and p_{n+1} is the $(n+1)$ point vector added to the set.

This voxel growing method leads to a connected component set E because the points have been added by connected voxels. In our case, the minimum distance between one point and E is less than parameter d of our voxel grid. That is why the parameter d also represents the connectivity of points in detected planes.

3.2.6 Plane Detection

To get all planes with an area of at least $area_{min}$ in the point cloud, we repeat these steps (best local seed plane choice and voxel growing) with all points by descending order of their score. Once we have a set E, whose area is bigger than $area_{min}$, we keep it and classify all points in E.

4. Results and Discussion

4.1. Benchmark analysis

To test the improvements of our method, we have employed the comparative framework of [12] based on range images. For that, we have converted all images into 3D point clouds. All Point Clouds created have 260k points. After our segmentation, we project labelled points on a segmented image and compare with the ground truth image. We have chosen our three parameters d, $area_{min}$ and γ by optimizing the result of the 10 perceptron training image segmentation (the perceptron is portable scanner that produces a range image of its environment). Bests results have been obtained with $area_{min} = 200$, $\gamma = 5$ and $d = 8$ (units are not provided in the benchmark). We show the results of the 30 perceptron images segmentation in table 1. GT Regions are the mean number of ground truth planes over the 30 ground truth range images. Correct detection, over-segmentation, under-segmentation, missed and noise are the mean number of correct, over, under, missed and noised planes detected by methods. The tolerance 80% is the minimum percentage of points we must have detected comparing to the ground truth to have a correct detection. More details are in [12].

UE is a method from [12], UFPR is a method from [10]. It is important to notice that UE and UFPR are range image methods and our method is not well suited for range images but 3D Point Cloud. Nevertheless, it is a good benchmark for comparison and we see in table 1 that the accuracy of our method is very close to the state of the art in range image segmentation.

To evaluate the different improvements of our algorithm, we have tested different variants of our method. We have tested our method without normals (only with distance from points to plane), without voxel growing (with a classical region growing by k neighbors), without our FWPF normal estimation (with WPF normal estimation), without our score function (with residual score function). The comparison is visible on table 2. We can see the difference of time computing between region growing and voxel growing. We have tested our algorithm with and without normals and we found that the accuracy cannot be achieved whithout normal computation. There is also a big difference in the correct detection between WPF and our FWPF normal estimation as we can see in the figure 4. Our FWPF normal brings a real improvement in border estimation of planes. Black points in the figure are non classified points.

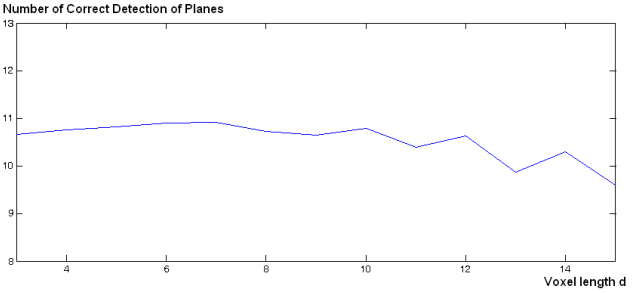

Figure 5. Correct Detection of our segmentation algorithm when the voxel size d changes.

We would like to discuss the influence of parameters on our algorithm. We have three parameters : $area_{min}$, which represents the minimum area of the plane we want to keep, γ, which represents the thickness of the plane (it is generaly closely tied to the noise in the point cloud and especially the standard deviation σ of the noise) and d, which is the minimum distance from a point to the rest of the plane. These three parameters depend on the point cloud features and the desired segmentation. For example, if we have a lot of noise, we must choose a high γ value. If we want to detect only large planes, we set a large $area_{min}$ value. We also focus our analysis on the robustess of the voxel size d in our algorithm, i.e. the ratio of points vs voxels. We can see in figure 5 the variation of the correct detection when we change the value of d. The method seems to be robust when d is between 4 and 10 but the quality decreases when d is over 10. It is due to the fact that for a large voxel size d, some planes from different objects are merged into one plane.

	GT Regions	Correct detection	Over-segmentation	Under-segmentation	Missed	Noise	Duration (in s)
UE	14.6	10.0	0.2	0.3	3.8	2.1	-
UFPR	14.6	11.0	0.3	0.1	3.0	2.5	-
Our method	14.6	10.9	0.2	0.1	3.3	0.7	308

Table 1. Average results of different segmenters at 80% compare tolerance.

Our method	GT Regions	Correct detection	Over-segmentation	Under-segmentation	Missed	Noise	Duration (in s)
without normals	14.6	5.67	0.1	0.1	9.4	6.5	70
without voxel growing	14.6	10.7	0.2	0.1	3.4	0.8	605
without FWPF	14.6	9.3	0.2	0.1	5.0	1.9	195
without our score function	14.6	10.3	0.2	0.1	3.9	1.2	308
with all improvements	14.6	10.9	0.2	0.1	3.3	0.7	308

Table 2. Average results of variants of our segmenter at 80% compare tolerance.

4.1.1 Large scale data

We have tested our method on different kinds of data. We have segmented urban data in figure 6 from our Mobile Mapping System (MMS) described in [11]. The mobile system generates 10k pts/s with a density of 50 pts/m^2 and very noisy data ($\sigma = 0.3m$). For this point cloud, we want to detect building facades. We have chosen $area_{min} = 10m^2$, $d = 1m$ to have large connected components and $\gamma = 0.3m$ to cope with the noise.

We have tested our method on point cloud from the Trimble VX scanner in figure 7. It is a point cloud of size 40k points with only 20 pts/m^2 with less noise because it is a fixed scanner ($\sigma = 0.2m$). In that case, we also wanted to detect building facades and keep the same parameters except $\gamma = 0.2m$ because we had less noise. We see in figure 7 that we have detected two facades. By setting a larger voxel size d value like $d = 10m$, we detect only one plane. We choose d like $area_{min}$ and γ according to the desired segmentation and to the level of detail we want to extract from the point cloud.

We also tested our algorithm on the point cloud from the LEICA Cyrax scanner in figure 8. This point cloud has been taken from AIM@SHAPE repository [1]. It is a very dense point cloud from multiple fixed position of scanner with about 400 pts/m^2 and very little noise ($\sigma = 0.02m$). In this case, we wanted to detect all the little planes to model the church in planar regions. That is why we have chosen $d = 0.2m$, $area_{min} = 1m^2$ and $\gamma = 0.02m$.

In figures 6, 7 and 8, we have, on the left, input point cloud and on the right, we only keep points detected in a plane (planes are in random colors). The red points in these figures are seed plane points. We can see in these figures that planes are very well detected even with high noise. Table 3 show the information on point clouds, results with number of planes detected and duration of the algorithm.

The time includes the computation of the FWPF normals of the point cloud. We can see in table 3 that our algorithm performs linearly in time with respect to the number of points. The choice of parameters will have little influence on time computing. The computation time is about one millisecond per point whatever the size of the point cloud (we used a PC with QuadCore Q9300 and 2Go of RAM). The algorithm has been implented using only one thread and in-core processing. Our goal is to compare the improvement of plane detection between classical region growing and our region growing with better normals for more accurate planes and voxel growing for faster detection. Our method seems to be compatible with out-of-core implementation like described in [24] or in [15].

	MMS Street	VX Street	Church
Size (points)	398k	42k	7.6M
Mean Density	50 pts/m^2	20 pts/m^2	400 pts/m^2
Number of Planes	20	2	142
Total Duration	452s	33s	6900s
Time/point	$\simeq$1ms	$\simeq$1ms	$\simeq$1ms

Table 3. Results on different data.

5. Conclusion

In this article, we have proposed a new method of plane detection that is fast and accurate even in presence of noise. We demonstrate its efficiency with different kinds of data and its speed in large data sets with millions of points. Our voxel growing method has a complexity of $O(N)$ and it is able to detect large and small planes in very large data sets and can extract them directly in connected components.

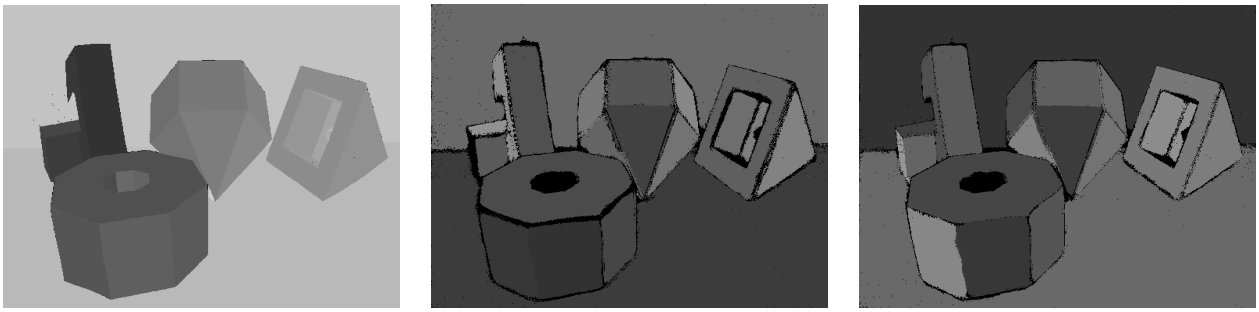

Figure 4. Ground truth, Our Segmentation without and with filtered normals.

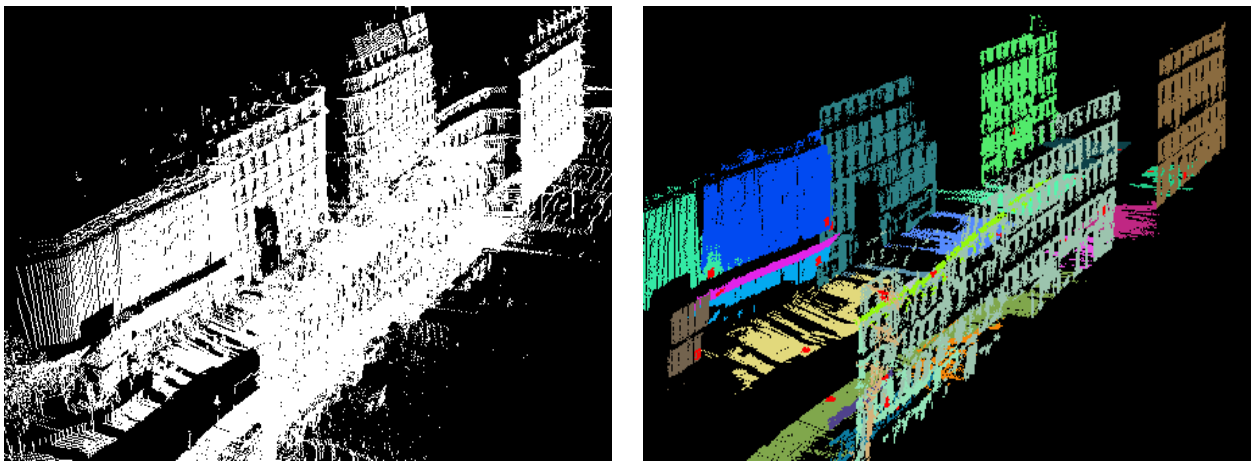

Figure 6. Planes detection in street point cloud generated by MMS ($d = 1m, area_{min} = 10m^2, \gamma = 0.3m$).

References

[1] Aim@shape repository http://shapes.aim-at-shape.net/. 6

[2] Octree class template http://nomis80.org/code/octree.html. 4

[3] A. Bab-Hadiashar and N. Gheissari. Range image segmentation using surface selection criterion. 2006. IEEE Transactions on Image Processing. 1

[4] J. Bauer, K. Karner, K. Schindler, A. Klaus, and C. Zach. Segmentation of building models from dense 3d point-clouds. 2003. Workshop of the Austrian Association for Pattern Recognition. 1

[5] H. Boulaassal, T. Landes, P. Grussenmeyer, and F. Tarsha-Kurdi. Automatic segmentation of building facades using terrestrial laser data. 2007. ISPRS Workshop on Laser Scanning. 1

[6] C. C. Chen and I. Stamos. Range image segmentation for modeling and object detection in urban scenes. 2007. 3DIM2007. 1

[7] T. K. Dey, G. Li, and J.Sun. Normal estimation for point clouds: A comparison study for a voronoi based method. 2005. Eurographics on Symposium on Point-Based Graphics. 3

[8] J. R. Diebel, S. Thrun, and M. Brunig. A bayesian method for probable surface reconstruction and decimation. 2006. ACM Transactions on Graphics (TOG). 1

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. 1981. Communications of the ACM. 1, 2

[10] P. F. U. Gotardo, O. R. P. Bellon, and L. Silva. Range image segmentation by surface extraction using an improved robust estimator. 2003. Proceedings of Computer Vision and Pattern Recognition. 1, 5

[11] F. Goulette, F. Nashashibi, I. Abuhadrous, S. Ammoun, and C. Laurgeau. An integrated on-board laser range sensing system for on-the-way city and road modelling. 2007. International Archives of the Photogrammetry, Remote Sensing and Spacial Information Sciences. 6

[12] A. Hoover, G. Jean-Baptiste, and al. An experimental comparison of range image segmentation algorithms. 1996. IEEE Transactions on Pattern Analysis and Machine Intelligence. 5

[13] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. 1992. International Conference on Computer Graphics and Interactive Techniques. 2

[14] P. Hough. Method and means for recognizing complex patterns. 1962. In US Patent. 1

[15] M. Isenburg, P. Lindstrom, S. Gumhold, and J. Snoeyink. Large mesh simplification using processing sequences. 2003.

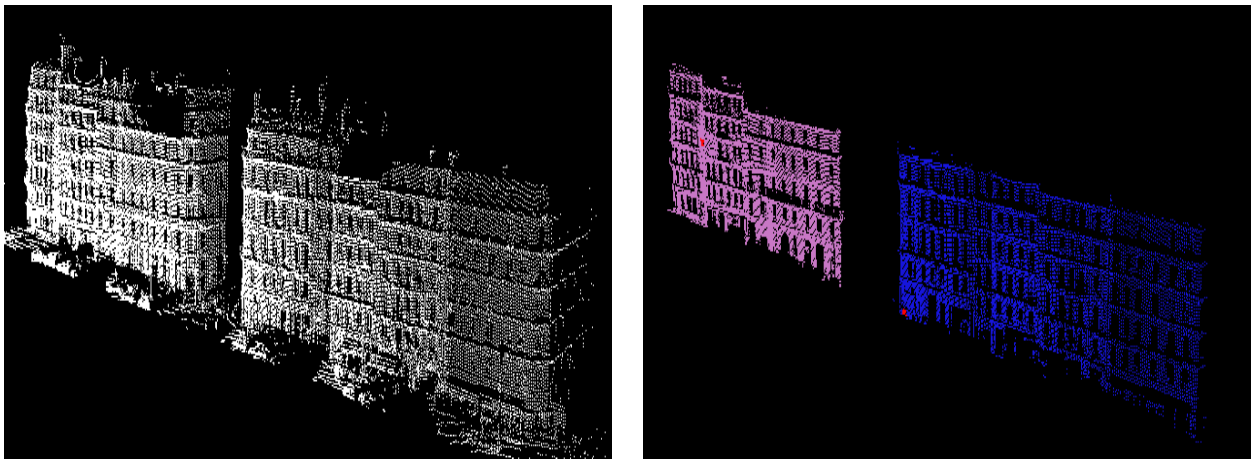

Figure 7. Planes detection in street point cloud generated by fixed Trimble VX Scanner ($d = 1m, area_{min} = 10m^2, \gamma = 0.2m$).

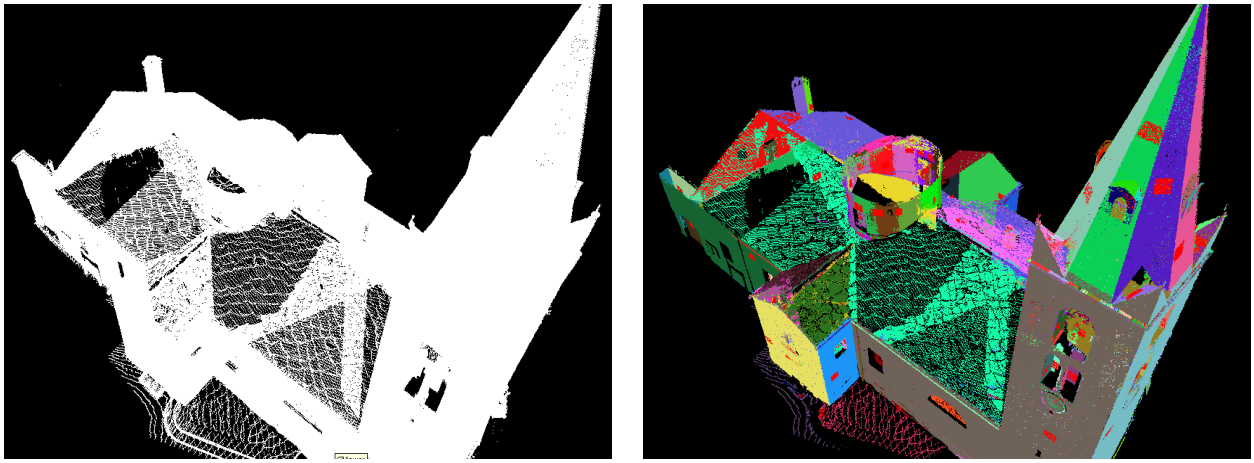

Figure 8. Planes detection in church point cloud generated by fixed LEICA Cyrax Scanner ($d = 0.2m, area_{min} = 1m^2, \gamma = 0.02m$).

Proceedings of the 14th IEEE Visualization. 6

[16] P. Jenke, M. Wand, M. Bokeloh, A. Schilling, and W. Strasser. Bayesian point cloud reconstruction. 2006. Computer Graphics Forum. 1

[17] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. 2003. Proceedings of ACM SIGGRAPH 2003. 3

[18] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak. Fast plane detection and polygonalization in noisy 3d range images. 2008. International Conference on Intelligent Robots and Systems. 2, 3, 4

[19] T. Rabbani, F. A. V. den Heuvel, and G. Vosselman. Segmentation of point clouds using smoothness constraint. 2006. ISPRS Commission V Symposium 'Image Engineering and Vision Metrology'. 2, 3

[20] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. 2007. Computer Graphics Forum. 2

[21] C. M. Sharkarji. Least-squares fitting algorithms of the nist algorithm testing system. 1998. Journal of Research of the National Institute of Standards and Technology. 3, 4

[22] F. Tarsha-Kurdi, T. Landes, and P. Grussenmeyer. Hough transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. 2007. Laser07. 1

[23] G. Vosselman, B. G. H. Gorte, G. Sithole, and T. Rabbani. Recognising structure in laser scanner point clouds. 2004. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. 1

[24] M. Wand. Rendering of very large models. 2007. In: M. Gross, H. Pfister (editors): Point-Based Graphics, Morgan Kaufmann/Elsevier. 6

[25] G. Yu, M. Grossberg, G. Wolberg, and I. Stamos. Think globally, cluster locally: A unified framework for range segementation. 2008. 4th Intl. Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT). 1