PLANE AND BOUNDRAY EXTRACTION FROM LIDAR DATA USING CLUSTERING AND CONVEX HULL PROJECTION

Augustine Tsai [a], Chun.F Hsu [a], I-Chou Hong [a], Wen-Kai Liu [a]

[a] Institute for Information Industry - (atsai, chunhsu, issac, wkliu)@iii.org.tw

Commission WG III/2

KEY WORDS: Laser Scanner, LiDAR, Geometry Extraction, Planar segmentation, Convex hull

ABSTRACT:

In this paper, we propose a new approach to extract planar patches and boundary from a set of LiDAR point cloud. In the beginning, the 3D point cloud set is partitioned and assigned to fixed-size cubes. Secondly, local planar patches are generated by extracting surface normal vectors within each cube. Finally, the global planes are formed by grouping the planar patches. The boundary of global plans is retrieved by projecting point cloud onto 2D convex hulls.

1. INTRODUCTION

Since the availability of Google Earth to the general public, the overlay of 3D buildings on top of the aerial images draws great interests. Microsoft also offers its own Virtual Earth. 3D urban scenes offer a variety of applications, such as navigation, location-based services, augmented reality, real estate, GIS and risk management, etc.

Most of the large scale commercial 3D urban scenes are captured from aerial imagery or airborne LiDAR data. In recent years, efforts and systems for street level 3D scanning begin to flourish (Fruh and Zakhor, 2003; Goulette et al, 2007; Haala et al, 2008; Pfaff el al, 2007, Zhao and Shibasaki, 2001)

In 3D reconstruction, the street level building facades are assumed to consist of mainly planar surfaces. Instead of building triangular mesh directly from the raw point cloud, the geometry features, such as points, lines and surfaces are extracted first from the point cloud. Then, their correspondences on the two dimensional image planes are located for texture mapping (Stamos2000).

In this paper, we emphasize in geometry processing from terrestrial laser scan (TLS) data. The geometry of point cloud data, such as lines, planar surfaces can be extracted by two major methods: region-based and boundary-based segmentation.

With reference to region-based approaches, Weingarten et al (2003) proposed a cube sweeping algorithm to extract local planar patches, and then merge these patches into global planes. Tseng and Wang (2005) used octree splitting and merging to extract planar patches. Their approach is referred to the top-down segmentation.

In boundary based approaches, Jiang and Bunk (1999) used scan-line approximation, where each 3D point is assigned to an edge strength, and then a threshold is used to determine the type of edge. The prerequisite of this method is that the data points have to be in order spatially. In following researches, the point projection method for boundary extraction has been proposed (Zhou and Newmann, 2006).

Our main contributions are in two folds: The first is a novel bottom-up 3D space partition for local planar patch extraction. The second is the convex hull projection approach to extract both straight and curved boundaries.

The rest of paper is organized as follows. In section 2, we describe the 3D cube partition, point assignments, and surface normal extraction. Then the merging process is applied to create a minimum number of planar surfaces with maximum size. Finally, a method to extract the merged boundaries using convex hull projection is proposed. In section 3, the algorithm is applied to a 14-story urban building and a synthetic 3D data.

2. PLANE AND BOUNDARY EXTRACTION

2.1 Cube Assignment for Point Clustering

Our objective here is to extract planar patches from a set of spatially unstructured point cloud data. Each point of the data is expressed in a 3-tuple Cartesian coordinate, $\mathbf{P} = (p_x, p_y, p_z)$. For a typical outdoor building scan project, it can produce over five million 3D points. It is obvious that data reduction and feature extraction are necessary.

First, the data points have to be partitioned and assigned to individual 3D cube space. A plane patch surface normal is then estimated by fitting the points inside the cube.

In the beginning, a cube C with length l is selected (l is depend on the sparseness of the data). The following algorithm is the pseudo-code of the cube creation and assignment process.

Algorithm *cubeAssign(p, l)*
Function arguments: N_{pn}: number of input points, l: length of cube, C_s: cube sets, C_{msi} : merged cube sets, T_s: threshold of the minimum points in a cube

1. C_s=Null, C_{msi}=Null

2. **While** N_{pn}>0
3. point ← *readPoint*(p)
4. boundp ← *getBoundary*(point,*l*)
5. **if** (*sizeof*(C_s) = 0)
6. C_s ← *assign*(point, boundp)
7. **end if**
8. C_s ← *fitPointToCube*(point, boundp)
9. **End while**
10. *removeCube*(C_s , T_s)

Line 1: points are read sequentially;
Line 2: compute the cube boundary of the point :
x_min = floor(p_x/*l*), *x_max*= *ceiling*(p_x/*l*),
y_min = floor(p_y/*l*), *y_max*= *ceiling*(p_y/*l*),
z_min = floor(p_z/*l*), *z_max*= *ceiling*(p_z/*l*),
Line 5-7: If there is no existed cube enclose the current point, a new cube is created and the point is assigned to it
Line 8: If there is an existed enclosed cube, then the current point is assigned to it.
Line 9: Iterate through every point.
Line 10: Remove cubes which enclosed points are less than a threshold.

The following figure shows an example of cube assignment process base on function "*cubeAssign*".

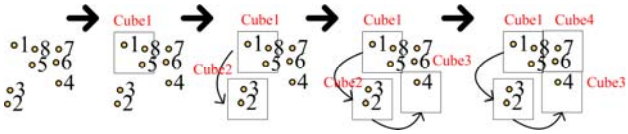

Figure 1. Cube assignment

Suppose there are 8 points to begin with. The numerals indicate the order of data appeared in the file. The first two cubes "*Cube 1*" and "*Cube 2*" are created to enclose point 1 and point 2, respectively. Since point 3 is within the range of "*Cube 2*", a new "*Cube 3*" has to be created to enclose point 4. The same process is repeated until every point is assigned to an enclosed cube.

At the end of cube assignment, a point which assigned to cube α can be denoted as

$$\mathbf{P}_{\beta}^{\alpha} = (p_{x\beta}^{\alpha}, p_{y\beta}^{\alpha}, p_{z\beta}^{\alpha}), \tag{1}$$

where β is the point index within the cube. The following figure shows all the points are partitioned by stack-up cubes.

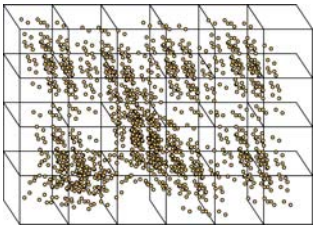

Figure 2. Segmented points data with cube.

Each of the cube and the surrounded points are well clustered. It is efficient for further analysis such as plane or boundary extraction and polygon reconstruction. A planar patch (a surface supported by local points) is a basic component to formal a globalized planar surface.

2.2 Planar Patches Extraction and Visualization

In this section, the local planar patch will be extracted from the points cloud and available for visualization in corresponding RGB color space. The so-called planar patch can be explained as the best-fit plane of the 3D points within a cube. Suppose there are *M* points ($\mathbf{P}_i^{\alpha}$=[x_i^{α}, y_i^{α}, z_i^{α}], i=1...*M*) distributed in the cube α (α = 1 ~ *K*). The vector $\mathbf{N}^{\alpha}$ = [n_x^{α}, n_y^{α}, n_z^{α}] with unit length, which is the normal vector, **N**, of the local planar patch, Θ^{α}, can be found by minimizing the objective function *S*

$$\begin{aligned} S &= \sum_{i=1}^{M} (\mathbf{N}^{\alpha} \cdot \mathbf{P}_i^{\alpha} + 1)^2 \\ &= \sum_{i=1}^{M} (n_x^{\alpha} x_i^{\alpha} + n_y^{\alpha} y_i^{\alpha} + n_z^{\alpha} z_i^{\alpha} + 1)^2 \end{aligned} . \tag{2}$$

The minimum solution to *S* can be solved directly by least square method. As presented in (Weingarten, 2004), the function *S* has to be partial differentiated with respect to n_x^{α}, n_y^{α}, n_z^{α} and set to 0. The optimal **N** can be evaluated as follow:

$$\mathbf{N}^{\alpha} = \mathbf{A}^{\alpha^{-1}} \cdot \mathbf{b}^{\alpha}, \tag{3}$$

where

$$\mathbf{A}^{\alpha} = \begin{bmatrix} \sum x_i^{\alpha 2} & \sum x_i^{\alpha} y_i^{\alpha} & \sum x_i^{\alpha} z_i^{\alpha} \\ \sum x_i^{\alpha} y_i^{\alpha} & \sum y_i^{\alpha 2} & \sum y_i^{\alpha} z_i^{\alpha} \\ \sum x_i^{\alpha} z_i^{\alpha} & \sum y_i^{\alpha} z_i^{\alpha} & \sum z_i^{\alpha 2} \end{bmatrix}, \tag{4}$$

$$\mathbf{b}^{\alpha} = \begin{bmatrix} -\sum x_i^{\alpha} \\ -\sum y_i^{\alpha} \\ -\sum z_i^{\alpha} \end{bmatrix}. \tag{5}$$

From now on, the planar patch of each cube is found. The maximum and the minimum value n_{max} , n_{min} can be selected as

$$\begin{cases} n_{\max} \geq n_i^j, i \in \{x, y, z\}, j \in [1, K] \\ n_{\min} \leq n_i^j, i \in \{x, y, z\}, j \in [1, K] \end{cases}. \tag{8}$$

These two extreme values can be applied to bound the range of visualization color in RGB format. In a specified cube α, all the points can be colorized by

$$\begin{cases} R^{\alpha} = (n_{x\alpha} - n_{\min})/(n_{\max} - n_{\min}) \\ G^{\alpha} = (n_{y\alpha} - n_{\min})/(n_{\max} - n_{\min}) \\ B^{\alpha} = (n_{z\alpha} - n_{\min})/(n_{\max} - n_{\min}) \end{cases}. \tag{9}$$

Based on the framework described above, the distributed plane is found, and the points can be visualized by corresponding color from well segmented 3D point data.

2.3 Merging of Planar Patches

The geometry features and the proposed visualization features are extracted through the above methods. Furthermore, in order to reduce the computation cost in the following boundary extraction and to describe the geometry features in a more semantic and realistic fashion, it is essential to merge the planar patches

Merging is similar to region growing in computer vision problems, in which Weingarten(2004), proposed a dynamic list structure of cubes and the computation complexity is ***O***(n) after sorting. In our design, we adopted the dynamic linked list tree structure to present the data set plane merging in ***O***(n) without sorting. By run-time encoding of planes, we only pass the normal vectors and the point of geometric center according to the index to the merging decision maker. The merging of planar patches is based on two criterions: proximity and the surface normal disparity (Stamos and Allen 2000). The proximity matrix is created by combining one dimension distance vectors of individual point to planes. D_{ij} denotes the distance from the geometric center of *i-th* patch to *j-th* patch.

$$\mathbf{P} = \begin{bmatrix} D_{11} & D_{12} & \cdots \\ D_{21} & \ddots & \vdots \\ \vdots & \cdots & \ddots \end{bmatrix} \tag{10}$$

Taking advantage of the symmetric property, we can reduce the matrix memory access by creating the mutual distance matrix **M** such as

$$\mathbf{M} = \mathbf{P} + \mathbf{P}^{\mathrm{T}}, \tag{11}$$

where the superscript T denotes the matrix transpose. For each element m_{ij} in **M**, a threshold value t is chosen to merge the similar patches. Thus, a set **m** is then obtained according to

$$\mathbf{m} = \{m_{ij}\} \ \forall m_{ij} \le t \tag{12}$$

The final step is to merge the planar patches with similar surface normal vectors.

2.4 Boundary Extraction Based on Planar Convex-hull

In definition, the convex hull for a points set X in a real vector space V is the minimal convex set containing X. For a planar convex-hull, the boundary can be used to present an edge of some plane. In a 2D points set, there exists a convex for all the point in $\mathbf{R}^2$. After a plane in $\mathbf{R}^3$ is rotated to x-y plane, the 3D planar edge detection problem is reduced to a 2D convex hull problem. By experiment, the computation cost of 3D convex hull is two times larger than in 2D space. The following procedure shows the detail to extract the edges of each merged planes.

Step 1. For each points $\mathbf{P}_i^{\alpha}$ (i = 1...M), we can evaluate the orthographic projected points $\mathbf{P}'^{\alpha}_i \in \mathbf{R}^3$ on the distributed plane Θ^{α} alone the direction of $\mathbf{N}^{\alpha}$. In the 3D-Euclid space, $\mathbf{P}'^{\alpha}_i$ must be constrained by the following condition:

$$\mathbf{P}'^{\alpha}_i \cdot \mathbf{N}^{\alpha} = C^{\alpha}, \forall i \in [1, M], \tag{12}$$

where C^{α} is a constant. Figure 3 shows the projection of $\mathbf{P}_i^{\alpha}$ to $\mathbf{P}'^{\alpha}_i$.

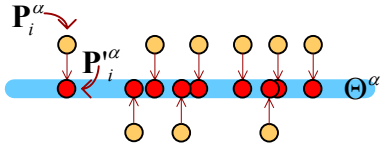

Figure 3. The projection of $\mathbf{P}_i^{\alpha}$ to $\mathbf{P}'^{\alpha}_i$.

Step 2. For constructing the planar convex-hull in $\mathbf{R}^2$, it is desired to find a rigid transformation (a rotation matrix $\mathbf{\Phi}^{\alpha}(\theta,\phi,\varphi) \in$ SO3 and a translation vector $\mathbf{t}^{\alpha}(t_x^{\alpha}, t_y^{\alpha}, t_z^{\alpha})$) which is able to transform $\mathbf{P}'^{\alpha}_i$ from Θ^{α} to $\mathbf{P}_i^{\alpha T}$ on x-y plane, such as

$$\mathbf{P}_i^{\alpha T} \cdot \hat{\mathbf{z}} = 0 \rightarrow (\mathbf{\Phi}^{\alpha} \cdot \mathbf{P}'^{\alpha}_i + \mathbf{t}^{\alpha}) \cdot \hat{\mathbf{z}} = 0, \tag{13}$$

where $\hat{\mathbf{z}} = [0,0,1]$ is an unit vector along **Z** axis. The corresponding $\mathbf{\Phi}^{\alpha}$ and $\mathbf{t}^{\alpha}$ can be determined by solving the above equation through "Levenberg Marquardt" algorithm and "Rodrigues" algebra. Then, the points $\mathbf{P}_i^{\alpha T}$ have the form [$p_x^{\alpha T}$, $p_y^{\alpha T}$, $p_y^{\alpha T}$=*0*] since $\mathbf{P}_i^{\alpha T} \in$ [x-y plane]. We can choose the first two elements to create new points $\underline{\mathbf{P}}_i^{\alpha} = [\, p_{xi}^{\alpha T}, p_{yi}^{\alpha T}] \in \mathbf{R}^2$. A new set $\underline{\mathbf{P}}^{\alpha}$ which contains $\underline{\mathbf{P}}_i^{\alpha}$ (i = 1...M) is established for developing the planar convex-hull.

Step 3. There are various convex-hull algorithms considering many aspects of constraints, including floating number precision, computational cost, and the implementation complexity. The one we adopt is the "2D qhull" (Barber 1996), the complexity of qhull is ***O***($n$log(n)), it works with double precision numbers, and it is fairly robust. The basic idea is applying "Divide-and-Conquer" method after sorting the points set in the fashion:

$$(x_0, y_0) < (x_1, y_1) \ \text{ or } (x_0 = x_1, y_0 < y_1). \tag{14}$$

The merge component takes linear time ***O***(n), therefore, the overall complexity is ***O***($n$log(n)). The procedure below shows the detail of merge algorithm:

Algorithm *2DConvexHullMerge*(H_L,H_R)
Function arguments: H_L: left convex polygon, H_R: right convex polygon

1. $\boldsymbol{P_i} = findMaxX(H_L)$;
2. $\boldsymbol{Q_i} = findMinX(H_R)$;
3. **while** (***not*** *tangent*($\boldsymbol{P_i}$, $\boldsymbol{Q_i}$))
4. $\quad \boldsymbol{P_{i-1}} = findMaxX(H_L - \{\, \boldsymbol{P_i} \,\})$;
5. $\quad \boldsymbol{Q_{i+1}} = findMinX(H_R - \{\, \boldsymbol{Q_i} \,\})$;
6. **end while**
7. Hull = *mergeTwoHull*(H_L,H_R);
8. **return** *createCounterClockHull*(Hull);

In order to draw the planar polygon more easily, it is important to produce a counter-clockwise permuted hull. From above, the geometric boundary of a plane can be extracted by applying 2D qhull algorithm. In most cases, the boundary can be considered as a good approximation to the real geometry shape; it can be a good starting point when there is an occlusion happened in the LiDAR data which hull helps to complement some point loss.

The overall algorithm proposed in this paper can be summarized in the following pipeline.

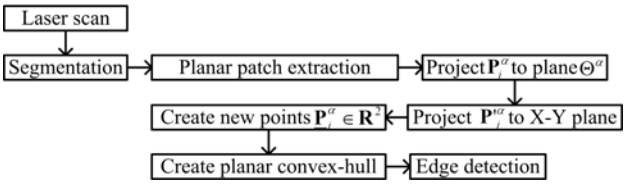

Figure 4. Processing pipeline

3. EXPERIMANTAL RESULTS

The presented approach is applied to the test data collected by laser scanner "RIEGL Z-420i". The point density is about 120 pts/m^2. Figure 5 shows the distribution of the raw data, in which the color is true color based on the external laser-camera registration. In segmentation process, the length of the cube is set to be 0.5cm. After distributed plane extraction and visualization, the results are shown in Figure 6. The points in blue indicate that the planes is parallel to the ground, and red and green colours represent those planes with normal vectors pointing to the east-west and the north-south directions, respectively.

The boundary of the extracted planes set is produced using less than five seconds to compute 500 thousand points by our proposed algorithm in Intel Duo2 QuadCore 2.83GHz. However, the overall extracted boundary is hard for eyes to recognize due to the large scale of data set. Therefore, we show another result by applying the same procedures as to the original data to a new data set as illustrated in figure 6. Some of the modern buildings contain non-straight lines or curved boundary. In order to illustrate the capability of convex-hull projection to deal these issues, we created synthetic 3D laser scan points. It is shown that by adopting our methods, the boundaries are extracted correctly.

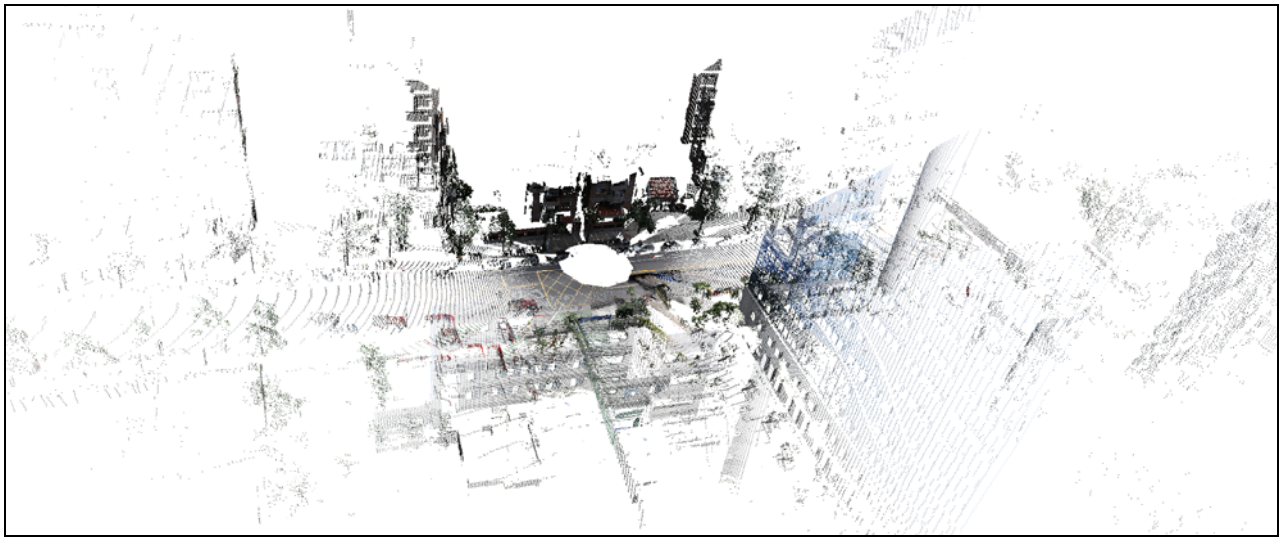

Figure 5. The distribution of the test data.

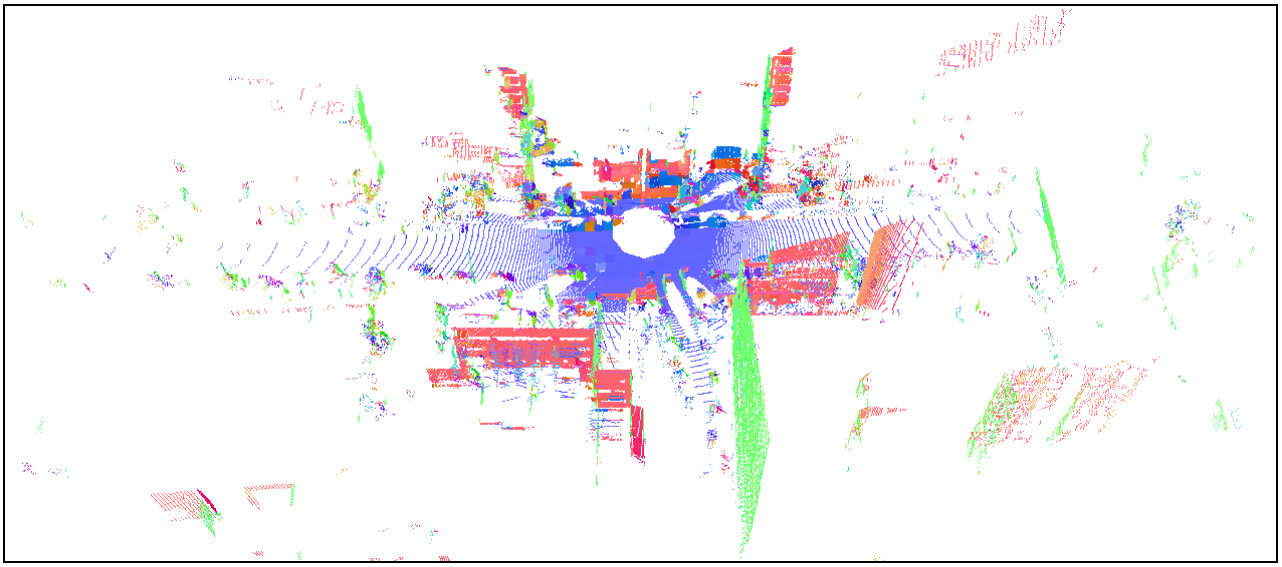

Figure 6. The visualized points.

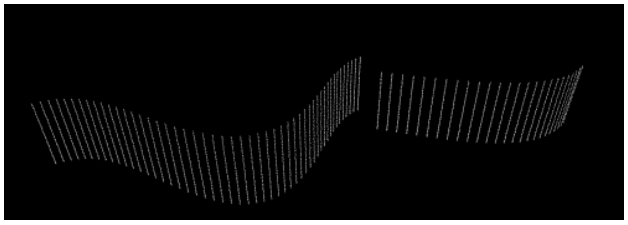

Figure 7. The synthetic 3D point clouds

Figure 7 is the synthetic point clouds; and figure 8 shows the extracted curved boundary by convex hull projection.

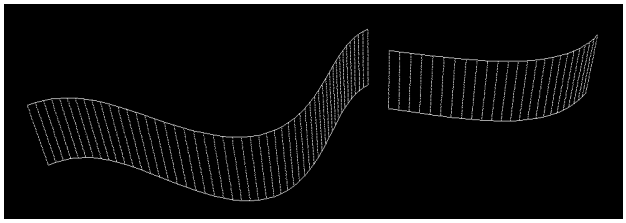

Figure 8. Extracted boundary by convex-hull projection

4. CONCLUSION

The proposed approach can be applied to the general large scale 3D point data in an efficient way without any pre-processing. All the necessary data structures are well described for engineering purpose. By the experiment results, the basic geometry features which include planes and edges are extracted. Base on these extracted features, the points are colorized according to the plane normal and connected by corresponding convex-hull. These characteristic of each points cluster are additive information and capable for further application.

5. ACKNOLEDEMENTS

The authors are grateful to the scanning assistance from the Strong Engineering Consulting Co, Ltd. This work is sponsored by the Ministry of Economics Affairs, Taiwan, with the project number: 98-EC-17-A-02-01-0809

6. REFERENCES

Barber, C.B., Dobkin, D.P., and Huhdanpaa, H.T., 1996, The Quickhull algorithm for convex hulls. In: *ACM Transaction on Mathematical Software, 22(4):469-483.*

Haala, N., Peter, M., Kremer, J. and Hunter, G., 2008. Mobile LiDAR Mapping for 3D Point Cloud Collection in Urban Areas a Performance Test. In: *Proceedings of the 21st International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* Beijing, China, Vol. XXXVII, Part B5, pp. 1119.

Goulette, F., Nashashibi, F., Abuhadrous, I., Ammoun, S. and Laurgeau. C., 2006. An Integrated On-board Laser Range Sensing System for On-the-way City and Road Modelling. In: *Proceedings of the ISPRS,* Paris, Part, III, pp. 43:1-43:6.

Jiang, X. and Bunke. H., 1999. Edge Detection in Range Images Based on Scan Line Approximation. *Computer Vision and Image Understanding*, 2(73), pp. 183-199.

Pfaff, P., Triebel, R., Stachniss, C., Lamon, P., Burgard, W. and Siegwart, R., 2007. Towards Mapping of Cities. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Rome, Italy.

Zhao H. and Shibasaki R., 2001. Reconstructing Urban 3D Model using Vehicle-borne Laser Range Scanners. In: *Proc. of the Third Int. Conf. on 3-D Digital Imaging and Modeling*, pp. 349-56.

Stamos, I. and Allen, P., 2000. 3-D Model Construction Using Range and Image Data. In: *IEEE International Conference, of Computer Vision and Pattern Recognition*, Hilton Head, SC Vol. I, pp. 531-536.

Talaya, J., Alamus, R., Bosch, E., Serra, A., Kornus, W. and Baron, A., 2004. Integration of a Terrestrial Laser Scanner with GPS/IMU Orientation Sensors. In: *International Archives of Photogr- ammetry and Remote Sensing,* Istanbul, Vol. V.

Tseng, Y-H and Wang. M., 2005. Automatic Plane Extraction from LiDAR Data Based on Octree Splitting and Merging Segmen-tation. In: *Geoscience and Remote Sensing Symposium,* Seoul, Korea,

Weingarten, J., Grüner, G. and Siegwart, R., 2003. A Fast and Robust 3D Feature Extraction Algorithm for Structured Environment Reconstruction. In: *International Conference on Advanced Robotics*, Coimbra, Portugal, Vol. III, pp. 390-397.

Q-Y Zhou and Neumann., 2008. Fast and Extensible Building Modeling from Airborne LiDAR Data**.** In: *SIGSPATIAL Conf. on Advances in Geographic Information Systems*, New York, USA, pp.1-8.