Automatic Reconstruction of Cities from Remote Sensor Data

Charalambos Poullis*
CGIT/IMSC
University of Southern California

Suya You†
CGIT/IMSC
University of Southern California

Abstract

In this paper, we address the complex problem of rapid modeling of large-scale areas and present a novel approach for the automatic reconstruction of cities from remote sensor data. The goal in this work is to automatically create lightweight, watertight polygonal 3D models from LiDAR data(Light Detection and Ranging) captured by an airborne scanner. This is achieved in three steps: preprocessing, segmentation and modeling, as shown in Figure 1.

Our main technical contributions in this paper are: (i) a novel, robust, automatic segmentation technique based on the statistical analysis of the geometric properties of the data, which makes no particular assumptions about the input data, thus having no data dependencies, and (ii) an efficient and automatic modeling pipeline for the reconstruction of large-scale areas containing several thousands of buildings.

We have extensively tested the proposed approach with several city-size datasets including downtown Baltimore, downtown Denver, the city of Atlanta, downtown Oakland, and we present and evaluate the experimental results.

1. Introduction

In recent years, there has been an increasing demand for applications which employ 3D models of real world areas in order to recreate a realistic and immersive virtual environment. The use of such models has already been successfully employed in applications for urban planning, development and architectural design [16, 1], training of emergency response personnel [9] and military personnel [4], simulation of man-made and natural events [13], autonomous robot navigation [18], etc.

Currently, the creation of such realistic 3D models remains a very difficult and complicated problem and particularly the modeling of large-scale areas is still a time-consuming, expensive and labor intensive task. In fact, the creation of models is still widely viewed as a specialized art, requiring personnel with extensive training and experience to produce useful models. Recent advances in the areas of remote sensing have allowed for an efficient and effective way of directly capturing 3D structural information of large-scale areas, however the size of the captured point cloud data and the complexity of the geometry generated from that data limits its uses.

In this paper, we address the complex problem of rapid modeling of large-scale areas. In particular, we focus on reconstructing realistic 3D models for cities entirely from remote sensor data captured by an airborne LiDAR scanner, and present a fast and automatic method for the reconstruction of the 3D models. The LiDAR data is processed in three steps: preprocessing, segmentation and modeling to produce a simplified, polygonal 3D model representation of the scene as shown in Figure 1.

The main contributions of this work are: (i) a novel, robust, automatic segmentation technique based on the statistical analysis of the geometric properties of the data, which makes no particular assumptions about the input and therefore has no data dependencies, and (ii) an efficient and automatic modeling pipeline for the reconstruction of large-scale areas. Our approach is fully automatic, having only 3 tunable parameters. Our extensive tests have shown that those parameters are very stable for urban areas and were fixed for all results presented. Using the proposed approach we were able to process large-scale cities with urban and suburban areas of size more than $20km^2$, containing several thousands of buildings, and produce an accurate 3D reconstruction in about 11 hours on a single computer.

We have extensively tested the proposed approach with several city-size datasets including downtown Baltimore, downtown Denver, the city of Atlanta, downtown Oakland, and we present and evaluate the experimental results.

The following sections will detail the proposed approach. Section 2 summarizes the state-of-knowledge in the areas of geometry modeling. Section 3 presents the overview structure of the system. Section 4, Section 5 and Section 6 describe the three main components of our approach: Preprocessing, Segmentation and Modeling, respectively. Section 8 presents and evaluates the experimental results, and finally the Section 9 concludes the presented work.

2. Related Work

A plethora of work has been proposed for solving the complex problem of modeling cities from airborne LiDAR data. Almost all of the existing work shares the same processing pipeline, however several distinctions exist between the different processing components. Below is an overview of the state-of-the-art in this area.

In [7] Zakhor et al. present an automatic approach for the generation of textured 3D models. By combining ground-based and airborne-based LiDAR scans they create buildings with roof and facade details. Despite the fact that the generated models are visually pleasing their approach relies on thresholds which need to be tuned depending on the data.

In [19] the authors employ an automatic approach for the creation of 3D geometric models of the buildings and terrain. A set

*e-mail: charalambos@poullis.org
†e-mail:suyay@graphics.usc.edu

978-1-4244-3991-1/09/$25.00 ©2009 IEEE

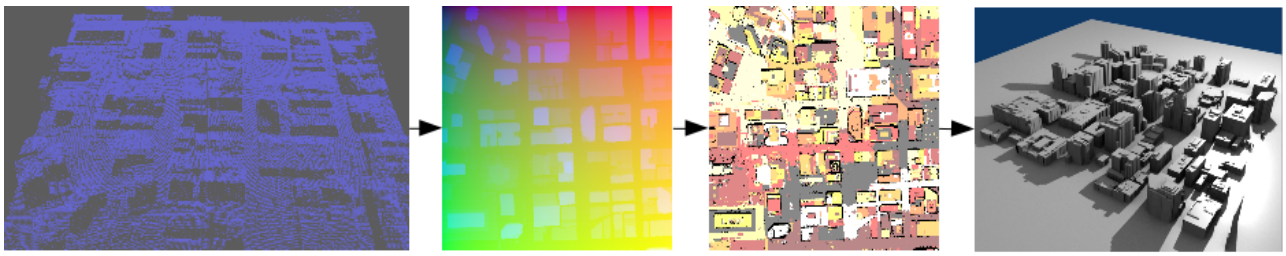


Figure 1. *System overview: Preprocessing converts the input unstructured 3D point cloud in (a) to the 2D XYZ maps in (b), Segmentation performs clustering on the 2D XYZ map in (b) and outputs regions as shown color-coded in (c), and Modeling extracts the boundaries from the segmented regions in (c) and creates the simplified 3D polygonal models in (d).*

of simple prismatic models are combined together to model different classes of buildings. Complex roof shapes are handled by automatically decomposing them into multiple simpler geometric primitives. A roof topology graph is used to identify the different parametric shapes that can be combined to construct complex roof shapes. However, the assumption that all constructed graphs will be small is not always true and this can complicate matters since graph/sub-graph matching is a NP-complete problem.

In a similar approach the authors in [10] present a building segmentation system for densely built areas. They are able to segment and delineate buildings as well as structures on top of buildings. The results look impressive however, the use of thresholds and data dependent parameters is abundant throughout their system.

Pollofeys et al. [8], propose a multi-view plane-sweep-based stereo algorithm which runs in real-time on the GPU for the reconstruction of buildings from video captures. Although, their results seem very promising the geometry of the reconstructed 3D models is relatively complex and becomes a bottleneck when dealing with large-scale areas. In addition, their approach uses video captured from ground, thus providing information only for the building facades.

On a different approach You et al. [20] developed a modeling system that extracts complex building structures with irregular shapes and surfaces. The system uses a set of geometric primitives and fitting strategies to model a range of complex buildings with irregular shapes. A more recent primitive-based system by Poullis et al. [12] presented a method for the rapid reconstruction of photorealistic large-scale virtual environments using a minimal set of three primitives. In this case the authors sacrifice full automation to achieve better model accuracy and quality.

In [14] they develop an interactive system for reconstructing geometry using non sequential views from uncalibrated cameras. The calculation of all 3D points and camera positions is performed simultaneously as a solution of a set of linear equations by exploiting the strong constraints obtained by modeling a map as a single affine view. However, due to the considerable user interaction required by the system, its application to large-scale areas is very limited. In [3] they exploit the strong rigidity constraints of parallelism and orthogonality present in indoor and outdoor architectural scenes. Using these constraints they calibrate the cameras and recover the projection matrices for each viewpoint.

In a similar context Nevatia et al. [11] , propose a user-assisted system for the extraction of 3D polygonal models of buildings from aerial images. Low level image features are initially used to build high level descriptions of the objects. Using a hypothesize and verify paradigm they are able to extract impressive models from a small set of aerial images.

Rottensteiner et al. [15] describe a method for roof plane delineation that tries to eliminate user-defined thresholds and relies on the framework of uncertain projective geometry and on robust estimation. In addition, the authors describe a new algorithm for the detection of step edges for delineating roof polygons, taking into account domain specific information in order to eliminate disturbances caused by trees adjacent to buildings.

3. Overview

The processing pipeline consists of three main modules as shown in Figure 1:

1. Preprocessing. In the preprocessing the raw 3D point cloud data is resampled into a 2D XYZ map. Local neighbourhood information is used to recover missing data and fill in any existing holes.
2. Segmentation. A novel automatic segmentation is used to segment the resampled data into disjoint, contiguous regions. The boundaries of the segmented regions are then extracted.
3. Modeling. Surface fitting is performed on the segmented regions. A simplification and refinement process smooths the extracted region boundaries and 3D models are extruded using the refined boundaries and fitted surfaces.

4. Preprocessing

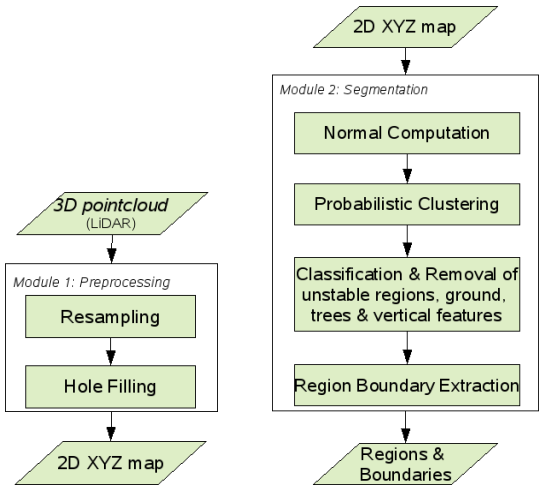
The role of the Preprocessing module is to primarily divide the data into memory manageable parts and to convert those parts into the internal data representation used by our system. This module takes as input the raw 3D point cloud data and converts it into a set of 2D XYZ maps, in two steps: Resampling and Hole filling, as shown in Figure 2(a).

Typically, the raw 3D point cloud data captured by the airborne LiDAR scanner cannot be processed as a whole. For this reason, we subdivide the raw data into smaller parts and resample each part into a 2D XYZ map. A 2D XYZ map is an efficient data representation which considerably reduces the computational complexity and processing time, since all subsequent operations are

performed in 2D, rather than in 3D. In addition, fast image processing techniques such as hole filling and normal computation can be easily performed as explained below.

Ideally, we would like to minimize the information loss of the input data by using all points, however this is not practically feasible considering the space/memory limitations and the size of the data. We overcome this trade-off between information loss and space/memory limitations by having the user specify the size of the maps in terms of an error tolerance τ between neighbouring samples in the map. Any points in the map which may contain more than one 3D data point are replaced by their average.

In the final step, an iterative hole filling algorithm is performed to fill in holes using local neighbourhood information. Figure 1(a) shows the input raw 3D point cloud data, and Figure 1(b) the resampled and hole-filled 2D XYZ map for an area of downtown Atlanta. Note that the color channels R, G, B of the map correspond to the X, Y, Z values of each point in the map respectively.



(a) Module 1: Preprocessing. (b) Module 2: Segmentation.

Figure 2. *The Preprocessing and Segmentation modules.*

5. Segmentation

The Segmentation module takes as input a 2D XYZ map and produces a set of disjoint, contiguous regions along with their associated 2D boundaries. This involves the four steps summarized in Figure 2(b).

Initially, a normal map is computed by considering local neighbourhood information to calculate a normal vector for each point in the XYZ map. The normal and XYZ maps are then used by the clustering which segments the map into disjoint, contiguous regions. Next, unstable regions and regions which ressemble ground, trees and vertical features are removed from further processing. Finally, the 2D exterior boundaries are extracted for all the remaining regions.

5.1. Normal Computation

Local neighbourhood information is used to compute the normal at each point in the map. For every point P_i of an XYZ map M we define the normal N_{P_i} of that point as,

$$N_{P_i} = \frac{1}{8} * \sum_{j=1}^{8} N_{P_j} \tag{1}$$

where N_{P_j} is the normal computed with the neighbouring point P_j within the 8-neighbourhood system. Each of the 8 normals N_{P_j} is computed as the cross product of the vectors connecting the point P_i and two consecutive(in clockwise order) neighbouring points P_j, P_{j+1}.

5.2. Clustering

A plethora of different approaches and techniques have been proposed throughout the recent years to resolve the complex problem of data segmentation and in particular of airborne LiDAR data. However, all existing approaches, such as [7, 10, 12], rely on data dependent parameters and thresholds in order to segment the data into different regions of similar height and/or orientations. Our proposed approach is an automatic, deterministic segmentation with an underlying region growing algorithm based on the statistical analysis of the geometrical properties of the data.

A region R_i is defined as a set of points $S_{R_i} \in M$, where M is the XYZ map. The clustering will result in T disjoint, contiguous regions such that,

$$M = \sum_{i=0}^{T} R_i \tag{2}$$

We define a region descriptor $D_{R_i} = (G_d, G_{\vec{n}})$ associated with each region R_i, consisting of two probability distributions:

1. A 1D probability distribution of the depths χ_d of all points $P \in S_{R_i}$ which is modeled by the 1D Gaussian function,

$$G_d = \frac{1}{\sqrt{2\pi\sigma_d^2}} e^{-\frac{1}{2*\sigma_d^2}*(\chi_d - \mu_d)^2} \tag{3}$$

   where σ_d^2 is the variance and μ_d is the mean depth.
2. A 3D probability distribution of the normals $\vec{N}_p$ of all points $P \in S_{R_i}$ (as computed earlier in Section 5.1), which is modeled by the 3D Gaussian function,

$$G_{\vec{n}} = \frac{1}{(2\pi)^{\frac{3}{2}}} \frac{1}{|\Sigma_{\vec{n}}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{N}_p - \vec{N}_\mu)\Sigma_{\vec{n}}^{-1}(\vec{N}_p - \vec{N}_\mu)} \tag{4}$$

   where $\Sigma_{\vec{n}}$ is the covariance matrix and $\vec{N}_\mu$ is the mean normal vector.

We initialize the first region R_0 with the left-most, top-most point in the map M, and progressively add to the region neighbouring points P_i which satisfy Equations (5) and (6) given by,

$$Pr_d(d_{P_i}) \geq Pr_d(\mu_d + \kappa\sigma_d) \tag{5}$$

$$Pr_{\vec{n}}(\vec{n}_{P_i}) \geq Pr_{\vec{n}}(\vec{N}_\mu + \kappa\vec{V}_{\Sigma_{\vec{n}}}) \tag{6}$$

where Pr_d and $Pr_{\vec{n}}$ are the probability functions G_d and $G_{\vec{n}}$, d_{P_i} and $\vec{n}_{P_i}$ is the depth and normal vector of the point P_i, μ_d and σ_d is the mean depth and variance of the distribution G_d of the region R_0 respectively, $\vec{N}_\mu$ is the mean normal vector of the distribution $G_{\vec{n}}$ of the region R_0, $\vec{V}_{\Sigma_{\vec{n}}}$ is the diagonal of the covariance matrix $\Sigma_{\vec{n}}$ of the distribution $G_{\vec{n}}$ from Equation (4), and κ which is set to 1, is the only free parameter in the proposed segmentation and

controls the maximum allowed deviation about the mean for G_d and $G_{\vec{n}}$.

Figure 3 shows the effect of the factor κ of Equations (5) and (6). Any point P_i with a higher probability than the probability of the distribution's mean perturbated by a factor κ of the variance, is accepted as belonging to the region R_0. The gray area shown in the graph of Figure 3 indicates all accepted probabilities of this region.

Once a point P_i is determined to belong to a region, the depth and normal distributions G_d and $G_{\vec{n}}$ are recomputed to reflect the depths and normals of all points in that region. This process is repeated until no more neighbouring points satisfy the conditions of Equations (5) and (6), in which case a new region R_1 is initialized with a point that has not yet been considered.

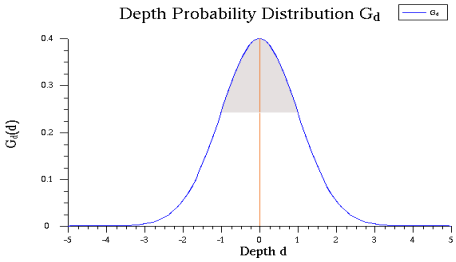


Figure 3. *An example of a depth distribution G_d. All points P_i whose depth probability $Pr_d(d_{P_i})$ falls inside the gray area defined by $Pr(\mu_d \pm \kappa\sigma_d)$ are accepted as belonging to the current region(iff the same is also true for the normal distribution $G_{\vec{n}}$). The orange line represents the degenerate case where the variance $\sigma_d = 0$.*

An obvious problem that arises, is the initial degenerate condition of the distributions of each region descriptor D_{R_i}. As previously mentioned, we begin the process by initializing a region R_i with a single point from the map M. Computing a Gaussian distribution using a single point is a degenerate case and causes the variance σ_d of the depth distribution and the covariance matrix $\Sigma_{\vec{n}}$ to be zero(orange line in Figure 3).

In order to overcome this problem during the initial stages, we express the variance and diagonal vector of the covariance matrix as a function of the current number of points in the region R_i given by,

$$\sigma^{'}(p) = max(\sigma_G, e^{\frac{1}{p}}) \tag{7}$$

where p is the current number of points in the region R_i, $max(\alpha, \beta)$ is a function that returns the maximum of the two input parameters α, β, and σ_G is the variance computed by fitting a Gaussian distribution to the data. We have empirically chosen the monotonically decreasing function e to control the initial values of the variance since it will exponentially descrease as the number of points increases as shown in Figure 4. This is desirable because the more points are contained in a region the more stable and robust the variance calculation will be when fitting the depth and normal distributions to the data.

Figure 5 shows results of the clustering applied on areas in Baltimore, Denver and Atlanta. Each disjoint region is represented by a different color. Note that the colors are reused and may appear for different regions.

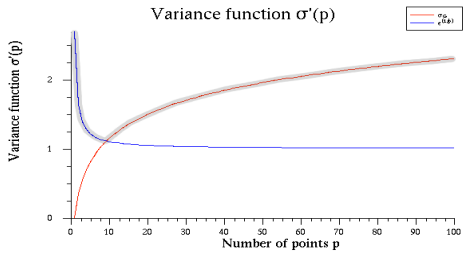


Figure 4. *The red line represents the variance σ_G computed from the data and the blue line represents the function $e^{\frac{1}{p}}$. The variance function $\sigma^{'}(p)$ ensures that the variance and covariance matrix of the depth and normal distributions respectively, will never equal zero, and is represented by the gray overlaid line.*

(a) Baltimore Area 1 (b) Atlanta Area 1

(c) Baltimore Area 2 (d) Denver Area 1

Figure 5. *Color-coded segmentation results. All points contained in the same disjoint, contiguous region are denoted by the same color. Each map size is 1Kx1K.*

5.3. Classification and Removal of Unstable Regions, Ground, Trees and Vertical Features

The segmentation returns a set of disjoint, contiguous regions which may include ground, trees, and other unwanted areas. We proceed by removing unstable regions, ground, trees and vertical features such as walls from further processing. Any region R_i which meets any of the following criteria is removed:

1. A region R_u is classified as unstable iff it contains less than three points, since a minimum of three points are required to define a surface.
2. A region R_{vf} is classified as a vertical feature iff the mean normal vector $\vec{N}_\mu$ of its normal distribution $G_{\vec{n}}$ is near per-

pendicular(i.e., $> 80^\circ$) to the XY plane.

3. The region R_g whose depth distribution G_d has the lowest mean depth μ_d and contains the largest number of points is classified as the ground.
4. A region R_t is classified as a tree iff every element of the diagonal vector $V_{\Sigma}\vec{n}$ of the covariance matrix $\Sigma_{\vec{n}}$ in Equation (4) is higher than a user-defined value τ_σ and, the mean depth μ_d in Equation (3) is less than a user-defined value τ_d. This criterion results from the observation that trees are non-linear surfaces which exhibit a large normal variation, as opposed to linear surfaces.

However, in special scenarios, such as simulations of man-made or natural events , a complete reconstruction of the entire city is required. Therefore it is imperative that we model all structures in the scene including buildings, trees, ground, bridges, raised highways, boats etc. In such cases, the criteria 3 and 4 described above are ignored during the processing. This results in a complete reconstruction of the entire area using simple polygonal 3D models as it will be shown in the following Section 6.

5.4. Region Boundary Extraction

The final step of this module is to extract exterior boundaries for the remaining regions. To achieve this, we employ Suzuki's algorithm for the extraction of the region boundaries as described in [17]. The result is a two-dimensional closed contour B_{R_i} for each remaining region R_i. The boundary B_{R_i} is a dense set of sequential neighbouring points. Figure 6(a) shows the extracted boundaries overlaid on the depth map for the complex building shown in Figure 6(d). This example demonstrates the high accuracy of the extracted boundaries resulting from the preservation of structural details by the segmentation such as elevator engines, air-conditioners, etc. The color-coded segmentation result computed as described in Section 5.2 is shown in Figure 6(b) for comparison.

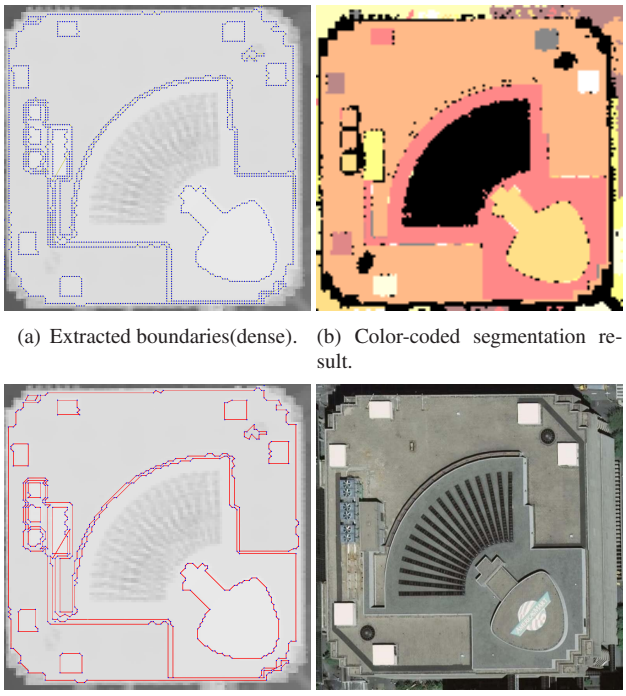
6. 3D Modeling

The 3D Modeling module takes the segmented regions and their boundaries and generates a set of lightweight, water-tight, polygonal 3D models. The process involves four steps shown in Figure 7.

Firstly, planes are fitted to each of the regions. Secondly, the dense boundaries extracted in Section 5.4 are simplified into boundaries consisting of a reduced set of points. A refinement process is then performed on the boundaries, in order to remove irregularities resulting from the noise of the data. Lastly, 3D models are created from the resulting boundaries and fitted surfaces.

6.1. Surface Fitting

A planar least-squares fitting is performed on every region R_i to determine the best plane passing through that region. We represent each fitted plane Π_i by a normal vector $\vec{N}_{\Pi_i}$ perpendicular to the surface and a scalar δ_{Π_i}, and we compute the new 3D positions for the points of each region, as the projection of each point to that plane. Due to the possibility of having a large number of points contained in each region, a RANSAC-guided sampling([2]) is used during the fitting to primarily reduce the computation time, increase stability and remove any small bias caused by outliers.



(a) Extracted boundaries(dense). (b) Color-coded segmentation result.

(c) The boundaries after the simplification and refinement. (d) Satellite image.

Figure 6. *Boundaries extracted from the segmented regions for a complex building. Boundary points are displayed in blue and the lines connecting two neighbouring boundary points are displayed in red.*

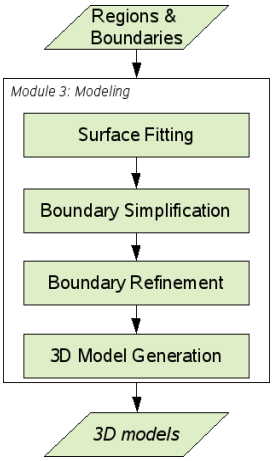


Figure 7. *Module 3: Modeling.*

6.2. Boundary Simplification & Refinement

The boundaries extracted, as described in Section 5.4, consist of a dense set of points. A Douglas-Peucker approximation([5]) followed by an Iterative End-Point Fit([6]) for refinement is applied on the dense boundaries. This results in a set of refined boundaries with a reduced number of points. Figure 6(a) and Figure 6(c) show the boundaries before and after the simplification and refinement step, respectively.

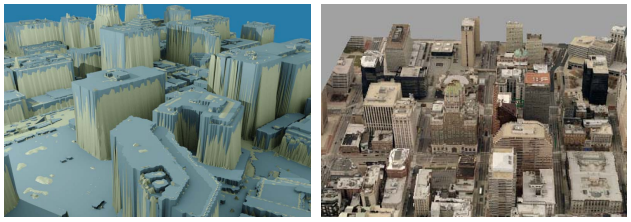
6.3. 3D Model Creation

In order to create a 3D model representing each region, the previously fitted planes are used to determine the elevation of the model. Next, the closed contour boundary B_{R_i} of each region R_i is used to generate facades for the model by extruding vertical surfaces to the ground. The result is a set of water-tight (i.e., no holes), light-weight (i.e., simple geometry), polygonal 3D models representing the scene.

7. Evaluation

The quantitative evaluation of the reconstructed 3D models is very difficult since there is no ground-truth for comparison. In our work, we use the following criteria for the qualitative and quantitative evaluation of the reconstructed models, in terms of the following parameters:

- Completeness. Completeness is defined as the ratio of the number of reconstructed buildings in the model and the actual number of buildings in the data.
- Quality. To determine the quality of the models, a qualitative evaluation is performed by visually inspecting the models for any artifacts such as misalignments between neighbouring surfaces and inconsistencies between the surfaces and the original data.
- Correctness. The correctness and accuracy of the reconstructed models is measured by overlaying the reconstructed models on the original LiDAR data(Figure 8(a)). We quantitatively evaluate the models by measuring the deviation of the fitted surfaces to the actual data. In our calculations we measure this deviation as the square root of the sum of the distance of each original point from the reconstructed surface it belongs to.



(a) Model overlaid on triangulated mesh of the original data. (b) Geo-referencing. Textured model with satellite imagery.

Figure 8. *(a) Evaluation of correctness and accuracy. (b) Geo-referencing. Satellite imagery is registered to the reconstructed polygonal models.*

- Geo-referencing. The reconstructed models represent areas from the real-world therefore, any imagery capturing the appearance of that same geometry should register precisely to the reconstructed model as well. The error of the registration is an indication of the deviation of the reconstructed geometry from the original(Figure 8(b)).This deviation is measured as the square root of the sum of the square of the differences - in pixels - between the projected 3D points of the model and the corresponding 2D image points. These correspondences are specified interactively by an operator.
- Scalability. The scalability is measured in terms of the time required for the generation of a 3D model. Our experiments have shown that the required time for the reconstruction is based primarily on the number and size of the segmented regions. Table 1 shows the times required for the reconstruction of U.S. cities, for each step of our pipeline:

Area → Module↓	Baltimore	Atlanta	Denver	Oakland
Pointsize	29292351	38797312	17677458	117415696
Size	$16km^2$	$28km^2$	$14km^2$	$17km^2$
Preprocessing	2hrs	3hrs	1.5hr	2hr
Segmentation	8hrs	9hrs	3hr	4hrs
Modeling	1hr	1.5hrs	1.2hr	1hr
Total	11hrs	13.5hrs	5.2hrs	7hrs

Table 1. *Scalability. Processing time required for the reconstruction of U.S. cities. Benchmark results were generated on an AMD Athlon 5200+ machine with 6Gb of RAM.*

Table 2 shows the quantitative and qualitative evaluation of our results. Densely built urban areas such as downtown districts generally resulted in very accurate and visually pleasing 3D models, which is mainly due to the fact that they are low-vegetation areas with buildings of high complexity and of relatively large areas. On the other hand, suburban and residential areas have performed the poorest primarily due to the high vegetation density, and in addition due to the minimal differences between the sizes and complexities of the buildings and the vegetation. For this reason, it is very hard even for a human operator to accurately evaluate the completeness and correctness of the suburban models strictly by looking at the input data.

Area → Criterion↓	Baltimore	Atlanta	Denver	Oakland
Completeness	100%	99%	94% – 96%	99%
Correctness & Accuracy	92% 0.972	87% 0.83	86% – 91% 0.4915	92% 0.417
Geo-referencing	0.556	0.672	0.4315	0.389

Table 2. *Qualitative and quantitative model evaluation.*

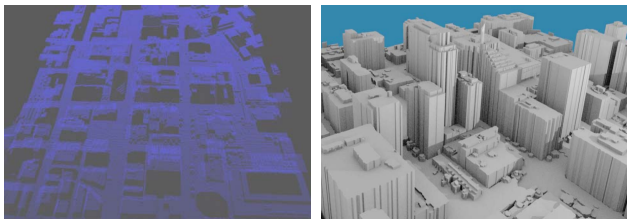
8. Experimental Results

Figure 9 shows a part of the downtown Baltimore consisting of a variety of different buildings of various shape complexities. Figure 9(a) shows a render of the original point cloud data for this area and Figure 9(b) shows a render of the automatically generated 3D model for the same area. The accuracy and high level of details can be visually verified. Note that for this case the comparison is between the reconstructed 3D model and the entire area, thus all regions including the ground, bridges, trees, etc are kept. A quantitave comparison between the geometric features of our generated 3D model and the triangulated mesh of the original data is shown in Table 3. As it is demonstrated, the reconstructed 3D models are visually pleasing and closely resemble the original mesh while at the same time, achieving a surface fitting error of $\epsilon = 0.972$ and a polygon reduction of 97.5% of the original.

Figure 10 shows the generated 3D models for a $16km^2$ area of the downtown Baltimore and surrounding areas. A close-up of the downtown area is shown in Figure 11. In this example, the input data was divided into 36 different maps of 1Kx1K each, during the preprocessing. The generated 3D models corresponding to

	Verts	Edges	Polys	Size
Mesh	1046529	3135496	2088968	151.4
Our Model	90656	135984	51442	9.3
Reduction	99.9%	95.6%	97.5%	93.8%

Table 3. *Mesh vs Model comparison. Size refers to the file size of an ASCII OBJ file format measured in Mb. The reduction % for each geometric feature is shown in red.*



(a) Original point cloud data. (b) The automatically generated 3D model for (a).

Figure 9. *Mesh vs Model comparison. Note that for this example we have kept all regions(ground, trees, etc) for comparison purposes.*

each of those maps were later combined together in a postprocessing step. As previously mentioned, the processing is performed on each individual map without any sharing of region information between neighbouring maps, hence some buildings which fall on the boundaries are subdivided. However, the segmentation using the proposed clustering returns equivalent regions for each neighbouring map and results in 3D models which perfectly fit the neighbouring 3D models on the boundaries. The Baltimore city model was generated in 11hrs as indicated in Table 1 using a single computer. Figure 12 shows a bird's eye view of the reconstructed 3D models for the downtown Denver and surrounding area.

9. Conclusion

We have presented a complete and novel approach for the automatic reconstruction of cities from remote sensor data. Unlike existing techniques, we proposed a fast and automatic approach for reconstructing simplified 3D polygonal models entirely from data captured by an airborne LiDAR scanner which does not rely on data dependencies. To achieve this, we have introduced a robust and effective segmentation technique which segments the data based on a statistical analysis of their geometric properties. Experimental results were demonstrated, which verify the validity of our approach and its successfull application on a variety of different data sets. Finally, we have presented the effective qualitative and quantitative measures we have employed for the evaluation of the generated 3D models.

10. Acknowledgements

The authors would like to thank the Airborn1, Inc., for providing the Oakland and Atlanta data, and Sanborn Corp. for providing the Denver data. They acknowledge the members and Prof. Ulrich Neumann in the Computer Graphics and Immersive and Technologies (CGIT) laboratory of USC. They also thank the reviewers for their valuable comments and suggestions.

References

[1] A real-time visualization system for large scale urban environments.

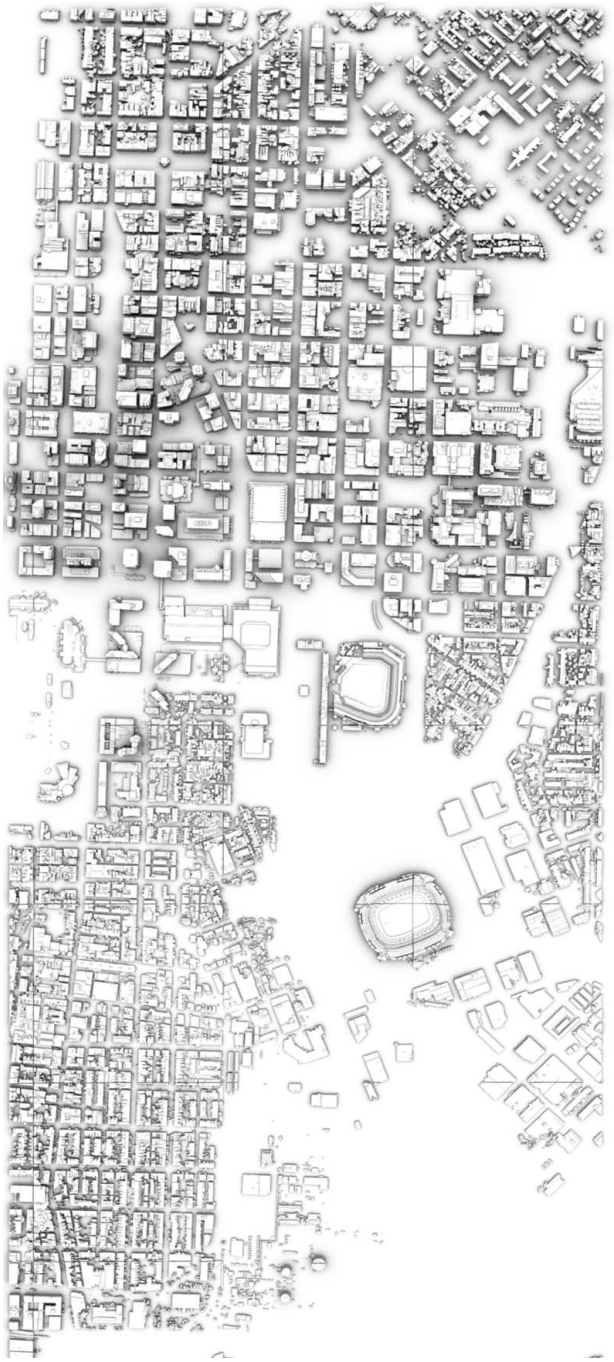


Figure 10. *Complete 3D reconstruction of downtown Baltimore and surrounding areas. (i.e., input data size 10Kx4K).*

[2] R. C. Bolles and M. A. Fischler. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Image Understanding Workshop*, pages 71–88, 1980.

[3] R. Cipolla and D. Robertson. 3D models of architectural scenes from uncalibrated images and vanishing points. In *ICIAP*, pages 824–829. IEEE Computer Society, 1999.

[4] J. Cohn, R. Schaffer, L. Milham, and K. Stanney. A virtual environment for urban combat training.

[5] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian*

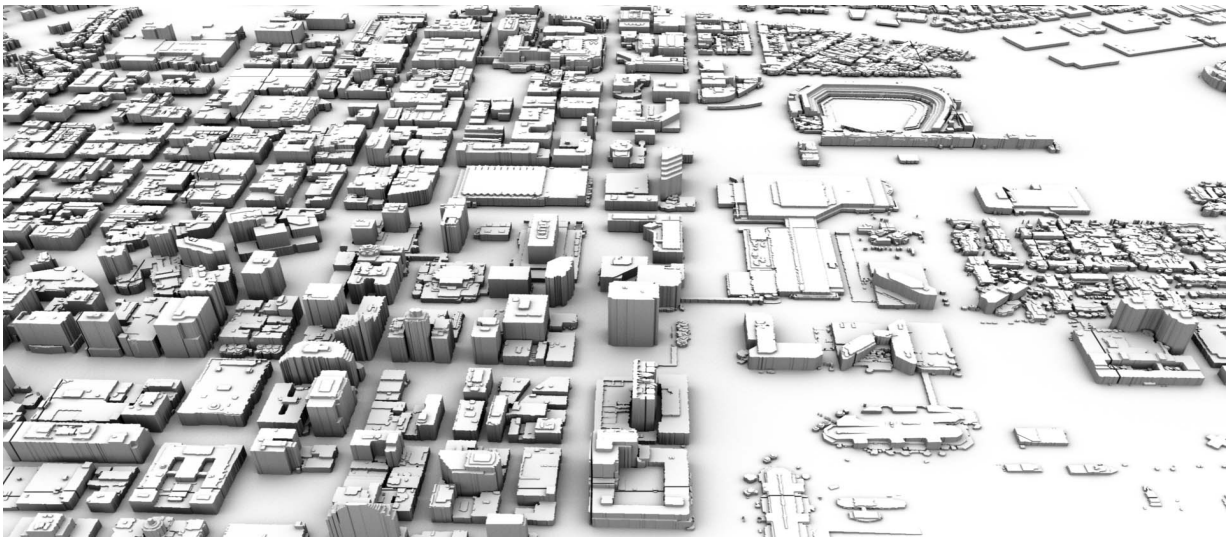


Figure 11. *Downtown Baltimore.*

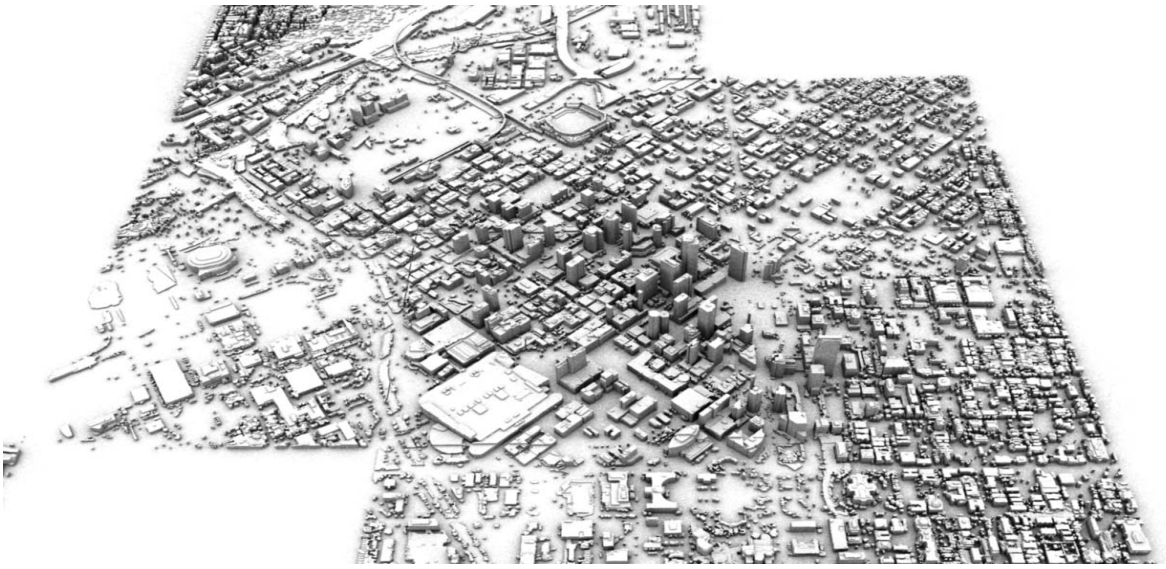


Figure 12. *Downtown Denver and surrounding areas.*

Cartographer, 10:112–122, 1973.

[6] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[7] C. Fruh and A. Zakhor. Constructing 3D city models by merging ground-based and airborne views. In *IEEE CVPR*, pages II: 562–569, 2003.

[8] D. Gallup, J. M. Frahm, P. Mordohai, Q. X. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *IEEE CVPR*, pages 1–8, 2007.

[9] L. Li, M. Zhang, F. Xu, and S. Liu. ERT-VR: an immersive virtual reality system for emergency rescue training. *Virtual Reality*, 8(3):194–197, 2005.

[10] B. C. Matei, H. S. Sawhney, S. Samarasekera, J. Kim, and R. Kumar. Building segmentation for densely built urban regions using aerial LIDAR data. In *IEEE CVPR*, pages 1–8, 2008.

[11] R. Nevatia and K. E. Price. Automatic and interactive modeling of buildings in urban environments from aerial images. In *ICIP (3)*, pages 525–528, 2002.

[12] C. Poullis, S. You, and U. Neumann. Rapid creation of large-scale photorealistic virtual environments. In *VR*, pages 153–160. IEEE, 2008.

[13] A. Ren, C. Chen, J. Shi, and L. Zou. Application of virtual reality technology to evacuation simulation in fire disaster. In H. R. Arabnia, editor, *CGVR*, pages 15–21. CSREA Press, 2006.

[14] D. P. Robertson and R. Cipolla. Building architectural models from many views using map constraints. *LNCS*, 2351:155–200, 2002.

[15] F. Rottensteiner, J. Trinder, S. Clode, and K. Kubik. Automated delineation of roof planes from liDAR data. In *Workshop on Laser Scanning*, pages xx–yy, 2005.

[16] D. M. Simpson. Virtual reality and urban simulation in planning: A literature review and topical bibliography. In *In Journal of Planning Literature*, volume 15, page 359376, 2001.

[17] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *CVGIP*, 30:32–46, 1985.

[18] N. Vandapel, R. Donamukkala, and M. Hebert. Experimental results in using aerial LADAR data for mobile robot navigation. In S. Yuta, H. Asama, S. Thrun, E. Prassler, and T. Tsubouchi, editors, *FSR*, volume 24 of *Springer Tracts in Advanced Robotics*, pages 103–112. Springer, 2003.

[19] V. Verma, R. Kumar, and S. Hsu. 3D building detection and modeling from aerial LIDAR data. In *IEEE CVPR*, pages II: 2213–2220, 2006.

[20] S. You, J. Hu, U. Neumann, and P. Fox. Urban site modeling from liDAR. In *ICCSA (3)*, volume 2669 of *LNCS*, pages 579–588. Springer, 2003.