A Framework for Automatic Modeling from Point Cloud Data

Charalambos Poullis, *Member*, *IEEE*

Abstract—We propose a complete framework for the automatic modeling from point cloud data. Initially, the point cloud data are preprocessed into manageable datasets, which are then separated into clusters using a novel two-step, unsupervised clustering algorithm. The boundaries extracted for each cluster are then simplified and refined using a fast energy minimization process. Finally, three-dimensional models are generated based on the roof outlines. The proposed framework has been extensively tested, and the results are reported.

Index Terms—Three-dimensional reconstruction, 3D modeling, point cloud, clustering, segmentation, shape refinement

1 Introduction

The automatic generation of three-dimensional (3D) models and, in particular, the rapid and automatic modeling from point cloud data has been, and still is, of significant interest to the computer graphics and computer vision communities. An increasing number of applications require the use of models ranging from computer games and visual effects to flight simulators for training personnel. Somewhere in this wide range, there exists a set of specialized applications such as geographical information systems, which require high-fidelity 3D replicas of the real world, for example, large-scale urban areas. Other examples of popular specialized applications include urban planning, emergency management training, military training, and so on.

In this work, we present a complete framework for automatic modeling from point cloud data. First, the unstructured, noisy point cloud data are preprocessed and split into memory manageable datasets. Second, a novel unsupervised clustering algorithm separates each dataset into clusters based on a hierarchical statistical analysis of the points' geometric properties. Third, the boundaries extracted for each cluster are refined using a fast energy minimization with graph cuts. An important advantage of this technique is the fact that it reduces the number of boundary orientations because it penalizes the different orientations by taking into account the newly proposed label costs [1]. Finally, 3D models are generated by extruding the roof outlines. The result is a set of non-overlapping, vastly simplified, watertight, polygonal 3D models. The proposed framework has been extensively tested with several large point cloud datasets, and the results and performance are reported.

• *The author is with the Immersive and Creative Technologies Lab, Cyprus University of Technology, PO Box 50329, Archbishop Kyprianos 31, Limassol 3036, Cyprus, Greece. E-mail: charalambos@poullis.org.*

Manuscript received 25 Aug. 2012; revised 6 Jan. 2013; accepted 19 Mar. 2013; published online 22 Mar. 2013.
Recommended for acceptance by R. Yang.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2012-08-0663.
Digital Object Identifier no. 10.1109/TPAMI.2013.64.

Our technical contributions are as follows:

- A complete framework for the automatic modeling from point cloud data.
- A robust unsupervised clustering algorithm P2C, based on a hierarchical statistical analysis of the geometric properties of the data.
- A fast boundary refinement process based on energy minimization with graph cuts.

The paper is organized as follows: In Section 2, we provide a brief overview of the state of the art in the area. Section 3 presents an overview of the proposed framework, and Section 4 describes the preprocessing process. In Section 5, we present the unsupervised clustering algorithm P2C, and in Section 6, we present the boundary extraction and refinement process. Finally, Section 7 presents the results produced by the proposed framework and Section 8 their evaluation.

2 Related Work

Reconstruction from point cloud data has been a very active area in computer vision. Many methods have been proposed for automatic modeling from point cloud data. Below, we provide a brief overview of the state of the art.

Lafarge et al. [2] present a structural approach for building reconstruction from single digital surface models using a library of 3D parametric blocks. The building boundaries are extracted interactively (or automatically in simple cases), and primitives are fitted to the extracted data. A Bayesian decision then finds the optimal configuration between the primitives using a Markov Chain Monte Carlo sampler associated with original proposition kernels. Although the results appear impressive, the process, however, is computationally expensive and very hard to scale up to large-scale datasets because it requires (as reported in their paper) about 3 hours of human labor and 35 minutes of computing to reconstruct about 800 objects.

The same authors [3] proposed a hybrid approach that uses geometric 3D primitives such as planes, cylinders, and so on, and combines them with patches from the mesh

corresponding to irregular roof components. The geometric primitives and the other urban components are then arranged in a common planimetric map through a multi-label energy minimization formulation. Finally, the objects are represented in 3D using various template fitting and meshing procedures.

Symmetries occurring in the geometry of the data have also been exploited in recent work [4] to refine or extract geometric primitives. Of particular interest is the mesh-simplification approach proposed by Zhou and Neumann [5] in which a data-driven approach is applied where global regularities appearing in singled-out buildings are identified and enforced through an iterative process of alignment steps. At each step, planes are fitted and aligned on the point cloud data, the height and position of the segments are aligned, and then the final model is generated.

Verma et al. [6] propose a technique for the automatic recognition and estimation of simple parametric shapes. First, they begin by segmenting the roof and terrain points to infer the roof topology using subgraph matching. Their technique scales well with large-scale datasets; however, the subgraph matching (an NP-complete problem) remains the Achilles' heel of their system.

A more recent approach proposed by Xiao and Furukawa [7] exploits the abundance of ground-level images in combination with 3D scans to automatically reconstruct high-fidelity 3D textured models of interior areas. Using their proposed algorithm, which is called inverse-CSG, they report very impressive results. Addressing the same problem of reconstructing interior areas, Huber and Adán [8] propose a method that relies only on 3D scanned data for the detailed modeling of wall surfaces in the presence of occlusions and clutter. Similarly, Turner and Zakhor [9] present a novel approach of generating floor plans that fits curves as well as straight segments with indoor walls and are guaranteed to be watertight. The results seem promising provided there is minimal registration error.

On a different note, Friedman and Stamos [10] propose an online algorithm for the detection and extraction of repeated structures in point cloud data and present promising results. The processing is done automatically without user interaction or training data, and the detection of the repeated structures allows for compressing large-scale data while maintaining their architectural details.

Although a plethora of techniques have been proposed for processing point cloud data, the gap between the state of the art and the desired goal of automatic modeling from point cloud data still remains wide. In this work, we introduce a framework that does not make any particular assumptions about the data, for example, spatial coherence, symmetry constraints, and so on, and its performance and the quality of the results are independent of the size of the data. Moreover, the proposed framework is fully automatic and does not require interaction with the user to mark roof boundaries nor to exactly separate the points corresponding to a building.

3 Technical Overview

The proposed framework is comprised of three main phases: the preprocessing of the data, the main parallel processing for the automatic modeling of 3D models, and the postprocessing of the results. Fig. 1 shows the processing pipeline.

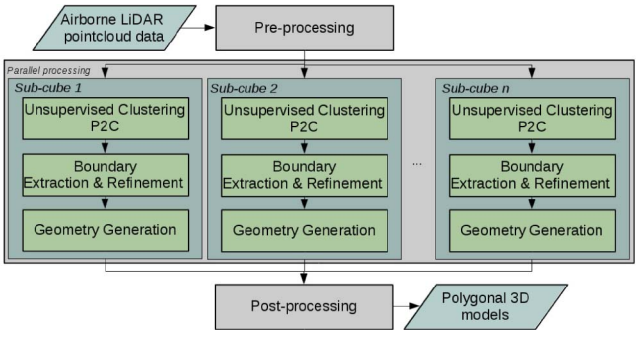


Fig. 1. Processing pipeline—three phases: the preprocessing of the data, the main parallel processing for 3D model generation, and the postprocessing of the results.

First, a point cloud captured by an airborne LiDAR scanner is structured and subdivided into subcubes. This results in a set of memory manageable components and ensures that all further processing is performed independently. Then, each subcube is processed in parallel in three steps:

1. An unsupervised clustering algorithm P2C is applied that results in a set of small area *patches*. Neighboring patches are then merged into higher level geometric elements, called *surfaces*, based on their similarity.
2. The boundaries for each of the resulting surfaces are extracted and refined using a novel refinement process based on energy minimization using graph cuts.
3. The boundaries corresponding to each surface are extruded to form polygonal 3D models.

Finally, the resulting 3D models corresponding to each of the subcubes are merged together.

4 Preprocessing

The first step in the processing pipeline is the preparation of the input data. Initially, the data have the form of a point cloud where each point in the cloud represents a Cartesian coordinate in 3D euclidean space. When dealing with such a type of data, one has to take into account the several issues that exist.

A typical problem that arises when dealing with the reconstruction of large-scale areas is the vast amount of data involved which quite often cannot be processed effectively as a single dataset. The size of a point cloud representing a large-scale area such as a city depends on 1) the size of the scanned area, and 2) the resolution of the scanning device and typically can range from a few million points to several hundred million points.

Another common problem is the random error introduced to the captured data due to the imperfect calibrations of the measurement instruments and unknown/unpredictable changes in the environment. Moreover, the data are unstructured and contain holes because they result from the merging of several scans (sweeps) of the same area. This, in

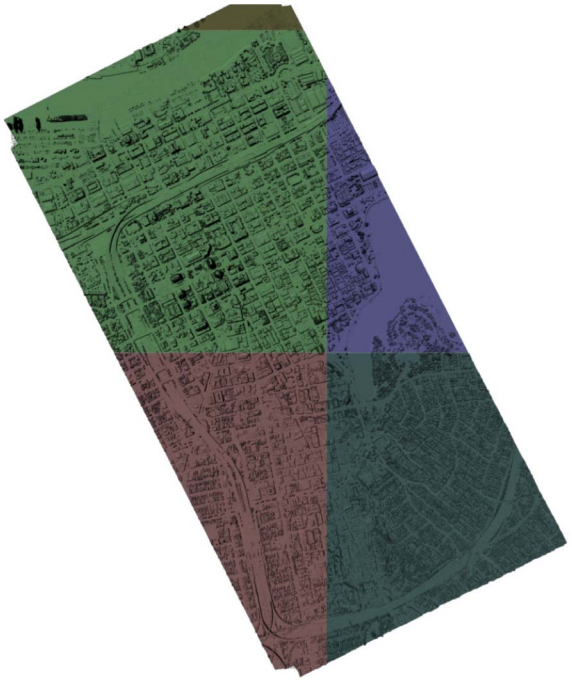


Fig. 2. Dataset A consisting of about 120 million points is structured and subdivided into five subcubes, each with an initial resolution of $2\mathbf{K} \times 2\mathbf{K} \times 2\mathbf{K}$.

effect, significantly reduces and in some cases eliminates the spatial coherence in the data.

To overcome these problems and reduce the computational complexity, Isenburg et al. [11] proposed a streaming approach that exploits the natural spatial coherence in a stream of points. An advantage of this approach is that it considerably accelerates the triangulation in cases where spatial coherence exist in the data. Further processing [12] can then be performed efficiently on large point clouds.

An alternative methodology for dealing with the aforementioned problems was proposed by Poullis and You [13], where the point cloud is divided into memory manageable components that are processed independently. This approach has the advantage of allowing parallel processing of each component since there is no information sharing between them. This, however, introduces "seams" between the final models corresponding to each component, which requires a postprocessing merging step to remove.

Following the latter approach, we employ an improved out-of-core variant. The first step is to structure the data into a more efficient and usable form such as octrees. This is achieved by first computing the bounding cube enclosing the total point cloud data, then subdividing the bounding cube into memory manageable subcubes, and finally, assigning the points into appropriate subcube. The initial resolution of each subcube is defined by the user ($r_x = r_y = r_z = 2K$ for all datasets shown unless stated otherwise). The internal representation for each subcube is an XYZ map stored as a high-dynamic range image able to retain the entire range of values of each point without loss of information due to quantization. A subcube can be adaptively subdivided (into eight subcubes, each having half the resolution of the original) whenever the maximum permissible number of points assigned to the subcube is exceeded (i.e., $2{,}048 \times 2{,}048$ points). To enable the application of this technique to datasets of arbitrary sizes, the assignment of the data to the subcubes is performed out of core: Each time a point is assigned to a subcube, it is saved to disk. Once all the points have been assigned to the subcubes, we apply an edge preserving smoothing (bilateral filter) and hole filling (based on local neighborhood information) to remove the noise and the holes, respectively. All subsequent processing is performed on each subcube separately.

The result is a set of subcubes saved on disk, each containing zero or more points. The complexity of the data preparation step is $O(2N)$, where N is the total number of points in the dataset and 2 represents the number of traversals needed: one for computing the bounding cube and one for assigning each point into the appropriate subcube.

Dataset A representing a large US city is shown in Fig. 2. The dataset consists of about 120 million points and covers an actual area of about $17\ \mathrm{km}^2$. During data preparation, the dataset is structured and subdivided into five subcubes, each with an initial resolution of $2\mathbf{K} \times 2\mathbf{K} \times 2\mathbf{K}$. The points are color coded according to the subcube to which they belong.

Fig. 3 shows a closeup of the result of the data preparation step for dataset B having lower resolution than dataset A and containing about 30 million points. The actual size of the area covered is about $16\ \mathrm{km}^2$. The dataset is structured and subdivided into 10 memory-manageable subcubes, each with an initial resolution of $2\mathbf{K} \times 2\mathbf{K} \times 2\mathbf{K}$. The size of each subcube is automatically determined based on the user-specified maximum memory available. In this example, each subcube contains on average about three million points.

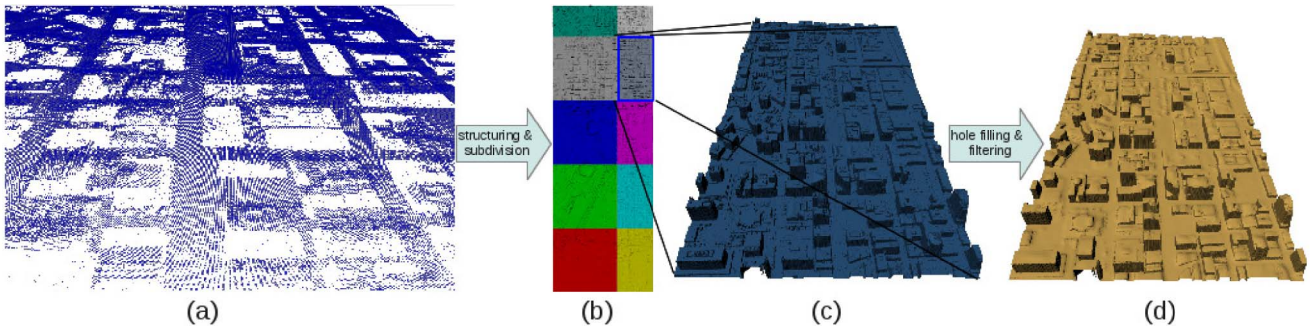


Fig. 3. (a) Dataset B. Structured and subdivided into memory-manageable color-coded subcubes shown in (b). The actual size of the area is about $16\ \mathrm{km}^2$. The dataset consists of about 30 million points.

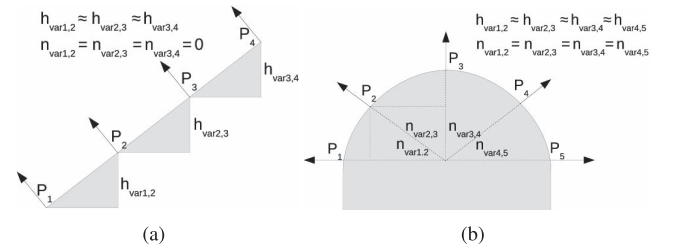


Fig. 4. (a) A slanted, linear surface. (b) A uniformly varying nonlinear surface. Incorporating the height and normal variances instead of only the height and normal measurements reduces problems during clustering, i.e., "staircase" effects in (a), and "co-centric" effects in (b).

5 P2C—An Unsupervised Clustering Algorithm

The points contained in the subcubes comprise a finite set of geometric elements corresponding to various structures in the world such as buildings, trees, and so on. In this section, we present the robust unsupervised clustering algorithm P2C. P2C is based on a hierarchical statistical analysis of the geometric properties of the data and consists of two steps: the clustering of points into patches described in Section 5.1 and the clustering of patches into surfaces explained in Section 5.2.

5.1 Point Clustering

We leverage the knowledge that points corresponding to the same structure exhibit similar geometric properties and cluster the data into multiple separate geometric elements called *patches*. To achieve this, we build upon the algorithm presented by Poullis and You [14] and propose a second step of hierarchical processing on the local geometry information. Instead of separating the information into depth and normal distributions and processing each one independently, we propose a more efficient, integrated method.

Each point P is represented using a six-dimensional feature vector f_P describing the local geometry and is given by

$$f_P = \langle N_x, N_y, N_z, P_z, h_{var}, n_{var} \rangle, \quad (1)$$

where N_x, N_y, and N_z are the components of the normal at each point computed by averaging the eight-neighborhood normal vectors, P_z is the height of the point, h_{var} is the local height variance around the eight neighbors of the point, and similarly, n_{var} is the local normal variance around the eight neighbors of the point.

The local variances (h_{var} and n_{var}) capture the *relative change* of the height and normal at each point with respect to the neighborhood. This is a very significant factor in the feature vector f_P that resolves commonly occurring problems with slanted linear and uniformly varying nonlinear surfaces. For example, consider the simple 1D case of points lying on a slanted linear surface in Fig. 4a. If only height measurements are considered when comparing the points, the result will be separate clusters for each point, i.e., the well-known "staircase" effect. However, if considering the relative change between the points, then this problem is overcome (up to a factor controlled by parameter κ). Similarly, in the case of points lying on a uniformly varying nonlinear surface in Fig. 4b, although the height measurements of the points are again different, the height and normal variance at each point is approximately the same; hence, it will result in clustering the points together (up to a factor controlled by parameter κ). This reduces cocentric clusters commonly occurring when clustering sampled points from nonlinear surfaces.

5.1.1 Initialization

Next, a new patch π_i is initialized, and its associated probability density function (pdf) ϕ_{π_i} is computed for an initial starting point P_j in the dataset. The pdf is modeled by a six-dimensional Gaussian function $\phi_{\pi_i} = \mathcal{N}_{\pi_i}(\vec{\mu_{\pi_i}}, \Sigma_{\pi_i})$ which describes the relative likelihood of a tuple of random variables, i.e., the feature vector of each point $\langle f_{P^j} \rangle$ (1), to occur at a given point in the observation space. Initially, the pdf ϕ_{π_i} corresponding to patch π_i contains only the feature vector f_{P^j} associated with the initial point P^j.

5.1.2 Iterative Processing

The clustering continues in an iterative fashion, each time considering all n neighboring points of P^j, P^{1-n} and determining the likelihood of their associated feature vectors $f_{P^{1-n}}$ as being (or not) part of ϕ_{π_i}. A likelihood test $\lambda(f_{P^k})$ determines if the feature vector f_{P^k} of point P^k is added to ϕ_{π_i} and, if successful, the mean and covariance matrix are recalculated. After the recalculation, care must be taken so that the first three components representing the normal vector at each point remain normalized, i.e., $\|N_x, N_y, N_z\|^2 = 1$. This process is repeated until all candidate neighboring points fail the likelihood test, in which case the next patch π_{i+1} with pdf $\phi_{\pi_{i+1}}$ is created and a point from the rejected candidate points is chosen as the new starting point. The process is repeated until all points belong to a patch.

5.1.3 Likelihood Test $\lambda(f_{P^k})$

Considering that the likelihood of a point P^k with feature vector f_{P^k} being part of a patch π_i with pdf ϕ_{π_i} is given by $\mathcal{N}_{\pi_i}(f_{P^k})$, the likelihood test is defined as follows:

$$\lambda(f_{P^k}) = \begin{cases} 1, & \text{if } \mathcal{N}_{\pi_i}(f_{P^k}) \geq \mathcal{N}_{\pi_i}(\vec{\mu_{\pi_i}} \pm \kappa * diag(\vec{\Sigma}_{\pi_i}) \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

where $\vec{\mu_{\pi_i}}$ and Σ_{π_i} are the mean and covariance matrix of $\mathcal{N}_{\pi_i}$, respectively, $diag(\vec{\Sigma}_{\pi_i})$ is the diagonal vector of the covariance matrix, and κ is empirically assigned the value of 1 as follows: Considering that a Gaussian distribution requires $6\sigma - 1$ values (3σ about the mean) and that we are interested in small perturbations around the mean, therefore the value of κ must lie within the range of [0-1.5]. After several experiments, we reached the conclusion that the value of 1 is the best tradeoff between the number of rejections/acceptances (by definition, 68 percent of the values lie within 1 standard deviation of the mean). It should also be noted that decreasing the value of κ will result in more clusters. However, during the second step, the patch clustering will again merge them into a minimal number of patches. In other words, changing the value of the parameter will affect the performance and not the accuracy.

5.1.4 Implementation Issues

At the initial stages of the iterative processing, each pdf contains only a few feature vectors. These are degenerate

cases that not only cause difficulties with the subsequent calculations of the covariance matrix (zero matrix inversion), but also result in continuous rejections from the likelihood test in (2). Consider, for example, the case where only one feature vector is contained in the pdf. In this case, all values of the diagonal vector of the covariance matrix $diag(\vec{\Sigma}_{\pi_i})$ will equal zero and as a result the likelihood test $\lambda(f_{P^k})$ reduces to

$$\lambda(f_{P^k}) = \begin{cases} 1, & \text{if } \mathcal{N}_{\pi_i}(f_{P^k}) \geq \mathcal{N}_{\pi_i}(\vec{\mu_{\pi_i}}) \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

which is true if f_{P^k} equals with μ_{π_i}. To resolve this problem, we introduce a control function $\delta(n)$ that ensures that the initial values of the diagonal vector of the covariance matrix are nonzero and is given by

$$\delta(x) = e^{\frac{1}{max(\epsilon,x)}}, \quad (4)$$

where x is the number of tuples contained in a pdf so far and $\epsilon = 10^{-6}$. Intuitively, during the early stages where only a few tuples are contained in the pdf, the control function $\delta(x)$ increases the likelihood of two tuples belonging to the same pdf. As the number of tuples contained in the pdf increases, the computation of the mean and covariance matrix become more stable and the effect of the control function $\delta(x)$ is diminished.

The result of the point clustering is a set of disjoint patches, each containing points with small spatial disparity and of similar feature descriptions. The computational complexity of this step is a constant $O(4NM - 3(N + M) + 2)$, where N and M are the width and height of each XYZ map representing each cube. Fig. 5a shows the clustered points into patches for a sample area.

5.2 Patch Clustering

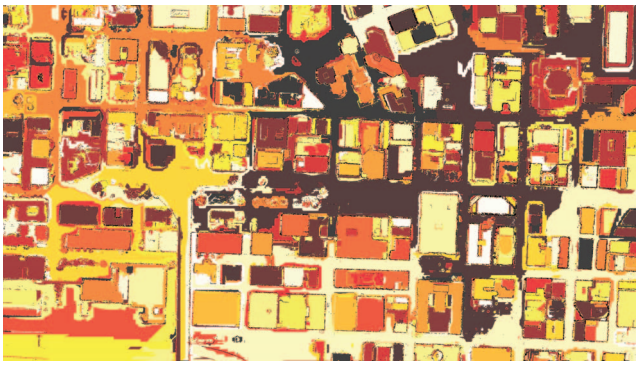
The patches are merged to form higher level geometric elements, called surfaces, as explained below.

Neighboring patches corresponding to the same structure are likely to have similar geometric properties. Hence, their pdfs modeling the distribution of these properties are expected to be similar. We leverage this characteristic and iteratively compare the pdfs of neighboring patches using as a metric the Bhattacharya distance.

The Bhattacharyya distance d_{bhat} is a computationally very simple quantity that measures the separability between two normal distributions $\mathcal{N}_1 = \langle \vec{\mu}_1, \Sigma_1 \rangle$ and $\mathcal{N}_2 = \langle \vec{\mu}_2, \Sigma_2 \rangle$ and is given by

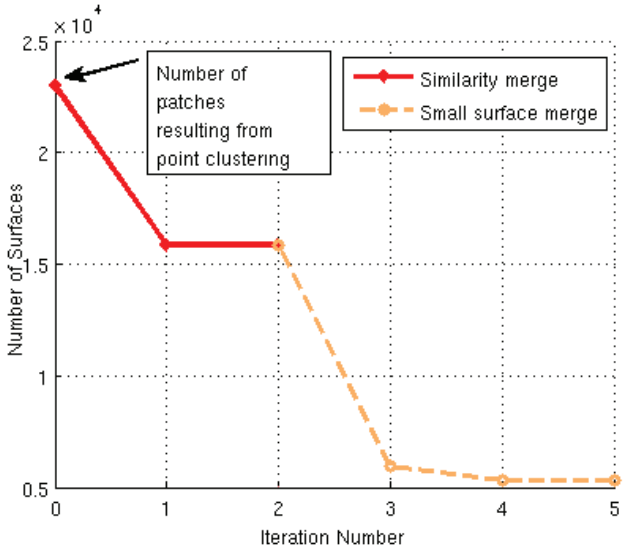
$$d_{bhat} = \frac{1}{8}(\vec{\mu}_2 - \vec{\mu}_1)^T \left[\frac{\Sigma_1 + \Sigma_2}{2}\right]^{-1} (\vec{\mu}_2 - \vec{\mu}_1) + \frac{1}{2}\ln \frac{|\frac{\Sigma_1+\Sigma_2}{2}|}{\sqrt{|\Sigma_1||\Sigma_2|}}. \quad (5)$$

Equation (5) gives the separability between two normal distributions; the first term measures the separability due to the difference of the means $\vec{\mu}_1$ and $\vec{\mu}_2$, whereas the second term measures the separability due to the difference between the covariance matrices Σ_1 and Σ_2. The range of values for the Bhattacharyya distance $|d_{bhat}|$ is $[0, \infty)$, starting with $d_{bhat} = 0$ in the case where the two normal distributions $\mathcal{N}_1$ and $\mathcal{N}_2$ are identical.



(a)

(b)



(c)

Fig. 5. (a) Points are color coded according to the patch they belong to. In this case, the total number of patches is 22,972. Processing time: 413 seconds. (b) Patches are color coded according to the surface they belong to. The number of surfaces after two iterations of similarity merging is 15,815 and after three iterations of the small surface merging is 5,300. Processing time: 182 (first step) + 156 (second step) seconds. ($\kappa = 1.0$, $\tau = 0.75$). (c) The patch clustering graph showing the surface reduction during the merging process.

5.2.1 *Initialization*

The patch clustering algorithm begins by instantiating a surface s_i for every patch π_i containing only the patch π_i. A surface is defined as a higher level geometric element that can contain multiple patches. Similarly to the pdfs described previously for the patches ϕ_{π_i}, a surface has a pdf ϕ_{s_i} modeled

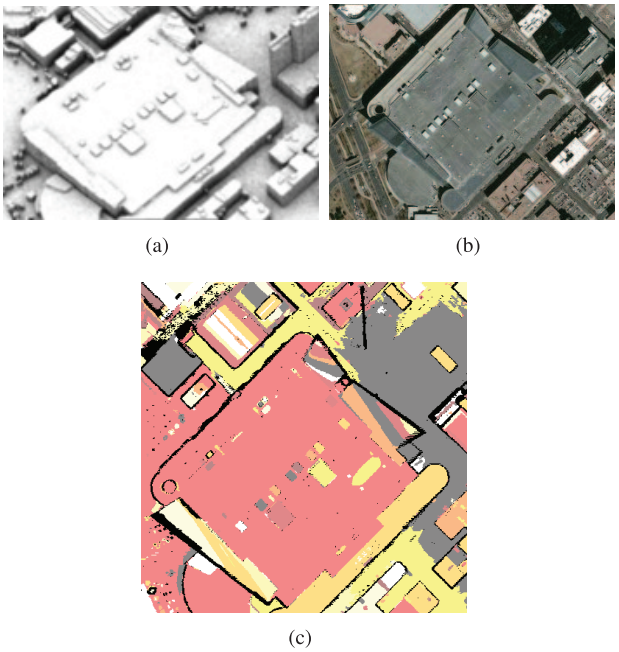


Fig. 6. (a) An automatically reconstructed building using the proposed algorithm. (b) A closeup from a satellite image available from Bing maps [15]. (c) The color-coded result of the P2C algorithm corresponding to the reconstructed building in (a). Note: The color codes are repeated, i.e., the building and ground are two different clusters that were labeled with the same color-code. Similarly, one of the air conditioners on the roof of the building can be seen in the reconstructed building.

by a six-dimensional Gaussian function $\phi_{s_i} = \mathcal{N}_{s_i}(\vec{\mu_{\pi_i}}, \Sigma_{\pi_i})$. At the time of the instantiation, the normal distribution $\mathcal{N}_{\pi_i}$ of the contained patch is used to initialize the normal distribution $\mathcal{N}_{s_i}$ of the newly created surface.

5.2.2 Iterative Processing

The algorithm then proceeds by comparing neighboring surfaces. Two neighboring surfaces s_1 and s_2 are merged if the separability measure $d_{bhat}(\mathcal{N}_{s1}, \mathcal{N}_{s2})$ between their two normal distributions is below an empirically defined value τ, in which case:

1. the patches contained in s_2 are added to the patches contained in s_1,
2. the surface s_2 is removed from further processing, and
3. the pdf ϕ_{s_1} is updated to reflect the combination of $\mathcal{N}_{s_1}$ and $\mathcal{N}_{s_2}$ by refitting to the union of the tuples (1) contained in the two $\mathcal{N}_{s1}, \mathcal{N}_{s2}$.

Alternatively, in the case where two neighboring surfaces are not similar (i.e., have a separability measure greater than τ), the algorithm continues to the next neighboring surfaces. This process is iteratively repeated until all neighboring surfaces are checked and no further merging occurs.

Finally, a second application of the same process is performed for surfaces containing a small number of points (i.e., less than 10). These small surfaces are merged to the neighboring surface with the highest matching score determined by the Bhattacharyya distance d_{bhat}.

The result is a set of surfaces, each containing one or more patches exhibiting similar geometric properties. The computational complexity of this step is $O(N)$ in the best-case scenario and $O(N(M-1)$ in the worst-case scenario, where N is the number of surfaces and M the number of neighbors of a surface. The results of the patch clustering applied to the patches of Fig. 5a are shown in Fig. 5b. Fig. 5c shows the reduction in the number of surfaces at each iteration of the algorithm. As explained earlier, the initial number of surfaces equals the number of patches resulting from the point clustering.

The proposed clustering algorithm results in high-accuracy results and does not remove details by clustering them together. This is demonstrated by the example in Fig. 6a. The reconstructed building is automatically reconstructed based on the clusters extracted shown in Fig. 6c. A closeup from a satellite image available from Bing Maps is shown in Fig. 6b [15].

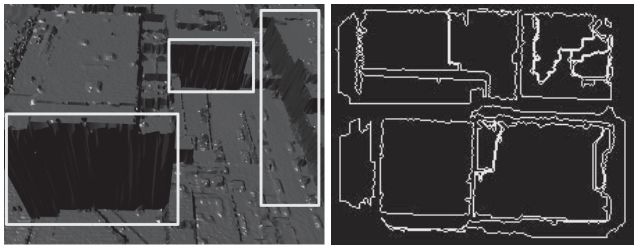


Fig. 7. (a) The "zig-zag" effects caused by the scanning process marked with white rectangles. (b) Boundaries extracted for a small area using Suzuki's algorithm [16].

6 Boundary Extraction and Refinement

Next, for all the resulting surfaces, their boundaries are extracted using the Suzuki and Abe [16] contour finding algorithm. The result is a set of exterior boundaries corresponding to each surface.

A common problem that arises when dealing with point cloud data is its representation of linear elements in the scene. In particular, when dealing with airborne LiDAR sensors, linear elements are captured as stepwise linear, i.e., "zig-zag," as shown in Fig. 7a. This is primarily due to the light beams emitted by the sensor in a zig-zag fashion as well as the sweeping motion of the airborne sensor. Fig. 7b shows the result of boundary extraction for a surface. To overcome this problem, we propose a novel boundary refinement process. Uniquely, the proposed boundary refinement process leverages the strengths of Gaussian mixture models (GMMs) for the boundary orientation extraction and classification and the global energy optimization using graph cuts for the boundary classification refinement as explained in the following Section 6.1.

6.1 Boundary Refinement

The boundary refinement process consists of three steps:

- orientation extraction and classification, described in Section 6.1.1,
- orientation classification refinement, described in Section 6.1.2,
- adjustment of the boundary points, described in Section 6.1.3.

6.1.1 Orientation Extraction and Classification

Buildings vary widely in terms of the orientations they may contain. For example, simple buildings can have two orientations, i.e., box shaped, whereas complex buildings can have several more orientations. Thus, to determine the shape of a building, we first need to extract and classify the orientations it contains.

First, for each surface, the local tangent at each boundary point P_i^B is calculated as $\vec{t_{P_i^B}} = \langle P_{i+1}^B - P_{i-1}^B \rangle$. The local tangents are then modeled by a GMM, a superposition of several two-dimensional N Gaussian densities of the form

$$p(x) = \sum_{i=1}^{N} \beta_i \mathcal{N}(x \mid \mu_i, \Sigma_i), \tag{6}$$

where each component of the mixture $\mathcal{N}(x \mid \mu_i, \Sigma_i)$ has mean μ_i and covariance matrix Σ_i. The parameters β_i are the mixing coefficients for which the following conditions hold:

$$\beta_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{N} \beta_i = 1. \tag{7}$$

The calculation of the parameters $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_N\}$, $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_N\}$, and $\boldsymbol{\Sigma} = \{\Sigma_1, \ldots, \Sigma_N\}$ is performed using an expectation maximization (EM) algorithm that maximizes the log of the likelihood function given by

$$ln\ p(\mathbf{X} \mid \boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{j=1}^{M} ln \left\{ \sum_{i=1}^{N} \beta_i \mathcal{N}(\mathbf{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i \right\}, \tag{8}$$

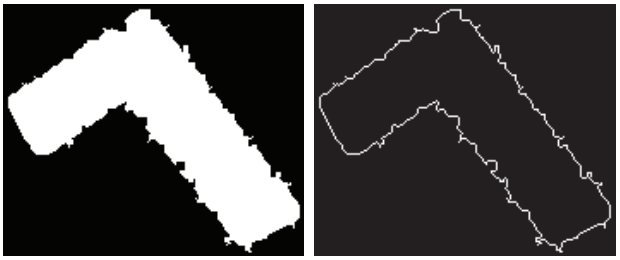
where $\mathbf{X} = \{x_1, \ldots, x_M\}$ are the data samples, i.e., the local tangents.

In many cases, assumptions are made for possible structural symmetries in the building [4]. In contrast to existing work, we do not assume that buildings always exhibit structural symmetries or have specific number of sides; therefore, we do not use a fixed N-order GMM, but instead N is computed using a minimum description length estimator criterion proposed by [17]. Hence, for each boundary of a surface, we first determine the best number of components the GMM should have and then perform the fitting using the EM algorithm to minimize (8).

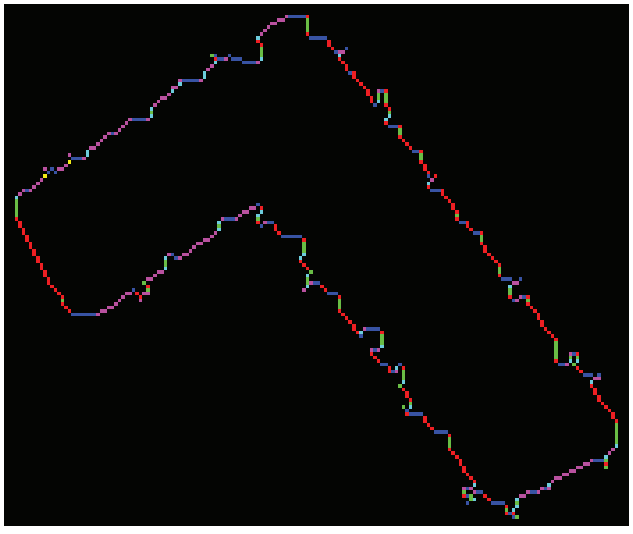
This process is applied on all surfaces and results in a set of orientations represented by the means μ_i of the Gaussian components in (8). An example is shown in Fig. 8a. The extracted boundaries are shown in Fig. 8b, and the orientations extracted for each boundary point are shown in Fig. 8c. In this example, the GMM consists of six Gaussian components whose information is shown in Fig. 8d. As is evident, the number of orientations extracted are far more than the actual number of two dominant orientations due to the noise in the data.

6.1.2 Orientation Classification Refinement

The extracted orientations contain a combination of dominant orientations, i.e., whose mixing coefficients β_i in (6) are large, and other not so dominant orientations. The next step is to refine the classification of boundary points such that they will be classified according to one of the extracted orientations while ensuring that only the dominant orientations



(a) (b)



(c)

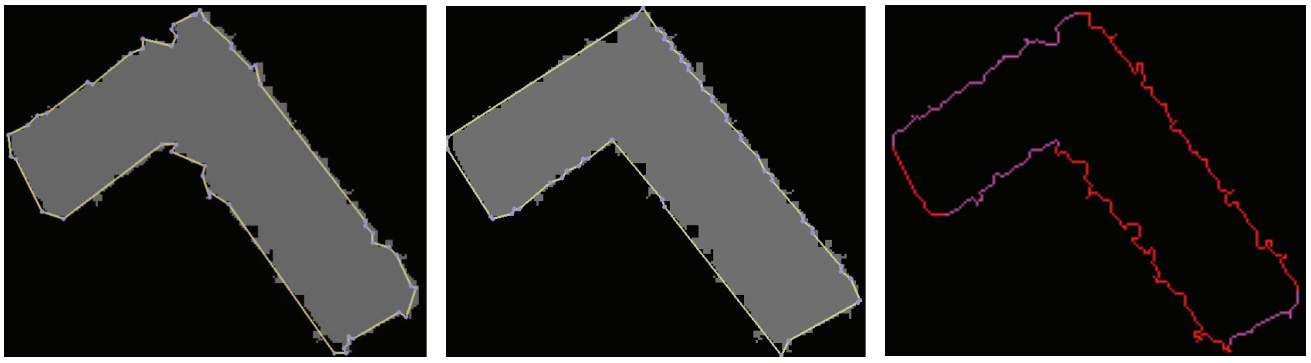
N_o	Mix Coeff. β_i	Means $\vec{\mu_i}$
1	0.224638	$(-0.832178 \quad 0.554509)$
2	0.240942	$(0.988452 \quad 0.151534)$
3	0.079710	$(-0.447214 \quad 0.894427)$
4	0.148551	$(0.000000 \quad 1.000000)$
5	0.302536	$(0.555184 \quad 0.831727)$
6	0.003623	$(-0.316228 \quad 0.948683)$

(d)

Fig. 8. (a) A surface representing a building. (b) The boundaries extracted using Suzuki's algorithm [16]. (c) The boundary point orientations extracted using GMM. Each boundary point is color coded according to the maximum probability of the Gaussian component in the GMM. In this example, the GMM consists of six Gaussian components shown in Fig. 8d. (d) The Gaussian components contained in the GMM for the example in (c). Each component is color coded to correspond to the colors in (c).

are considered. To achieve this, we perform an energy minimization using graph cuts [1].

Graph cuts: Given an input binary image I containing the boundary points of each surface, an undirected graph $G = <V, E>$ is created, where each vertex $v_i \in V$ corresponds to a boundary point $P_i^B \in I$ and each undirected edge $e_{i,j} \in E$ represents a link between neighboring boundary point $P_i^B, P_j^B \in I$. In addition, two distinguished vertices called *terminals* V_s and V_t are added to the graph G. An additional edge is also created connecting every boundary point $P_i^B \in I$ and the two *terminal* vertices, e_{i,V_s} and e_{i,V_t}. For weighted graphs, every edge $e \in E$ has an associated weight w_e. A *cut* $C \subset E$ is a partition of the vertices V of the graph G into two



(a) (b) (c)

Fig. 9. (a) Minimization without label costs. All six orientation labels are retained. (b) Minimization with label costs. Only two orientation labels are retained, corresponding to the two dominant orientations of the building. Boundary points are shown in purple and are connected with yellow lines. (c) The labeling of the boundary points is produced by graph cuts using label costs. The dominant orientations correspond to the Gaussian components 1 and 5 of Fig. 8d. The weight k_1 in (9) was the same for both cases.

disjoint sets S and T, where $V_s \in S$ and $V_t \in T$. The cost of each cut C is the sum of the weighted edges $e \in C$ and is given by

$$|C| = \sum_{\forall e \in C} w_e.$$

The minimum cut problem can then be defined as finding the cut with the minimum cost.

Labels: The binary case explained above can easily be extended to a case of multiple terminal vertices. We create two terminal vertices for each extracted orientation $\vec{t_i} = \mu_i$ corresponding to each Gaussian density in (6). Thus, the set of labels L has size $|L| = N$ and is defined to be $L = \{\vec{t_1}, \vec{t_2}, \ldots, \vec{t_N}\}$. Our experiments have shown that the number of labels is, on average, $|L| = [5 - 10]$.

Energy minimization function: Finding the minimum cut of a graph is equivalent to finding an optimal labeling $f : I_{P_i^B} \longrightarrow L$ which assigns a label $l \in L$ to each boundary point $P_i^B \in I$, where f is piecewise smooth and consistent with the original data. Thus, our energy function for the graph-cut minimization is given by

$$E(f) = E_{data}(f) + \kappa_1 * E_{smooth}(f) + \kappa_2 * E_{label}(f), \quad (9)$$

where κ_1 is the weight of the smoothness term and κ_2 the weight of the label term.

Data term: The data term provides a per-point measure of how appropriate a label $l \in L$ is for a boundary point $P_i^B \in I$ in the observed data and is given by

$$E_{data}(f) = \sum_{P_i^B \in I} D_{P_i^B}(f). \quad (10)$$

We define $D_{P_i^B}(f)$ for a boundary point P_i^B as the difference between the existing label $\vec{t_{P_i^B}}$, i.e., the local tangent at the boundary point P_i^B, and a new label $f(P_i^B) \in L$ and is given by

$$D_{P_i^B}\big(f(P_i^B)\big) = \big|\vec{t_{P_i^B}} - f\big(\vec{P_i^B}\big)\big|. \quad (11)$$

Therefore, the energy data term becomes

$$E_{data}(f) = \sum_{P_i^B \in I} \big(\big|\vec{t_{P_i^B}} - f\big(\vec{P_i^B}\big)\big|\big). \quad (12)$$

Smoothness term: The smoothness term provides a measure of the difference between two neighboring boundary points $P_i^B, P_j^B \in I$ with labels $\vec{l_i}, \vec{l_j} \in L$, respectively. Let $\vec{t_{P_i^B}}$ and $\vec{t_{P_j^B}}$ be the initial orientations of the neighboring boundary points in the observed data $P_i^B, P_j^B \in I$, respectively. We define a prior measure of the *observed* smoothness between boundary points P_i^B and P_j^B as

$$\Delta_{(P_i^B, P_j^B)} = \big|\vec{t_{P_i^B}} - \vec{t_{P_j^B}}\big|. \quad (13)$$

Equation (13) favors neighboring boundary points with similar orientations and penalizes otherwise.

In addition, we define a measure of smoothness for the global minimization. Let $\vec{l_i} = f(\vec{P_i^B})$ and $\vec{l_j} = f(\vec{P_j^B})$ be the orientations under a labeling f. We define a measure of the smoothness between neighboring pixels P_i^B and P_j^B under a labeling f as

$$\tilde{\Delta}_{(P_i^B, P_j^B)} = |\vec{l_i} - \vec{l_j}|. \quad (14)$$

Using the smoothness prior defined for the observed data (13) and the smoothness measure defined for any labeling f (14), we can finally define the energy smoothness term as follows:

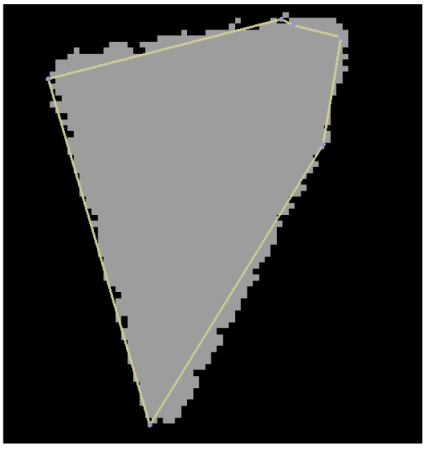


Fig. 10. Boundary refinement for surface with multiple dominant orientations from dataset A.

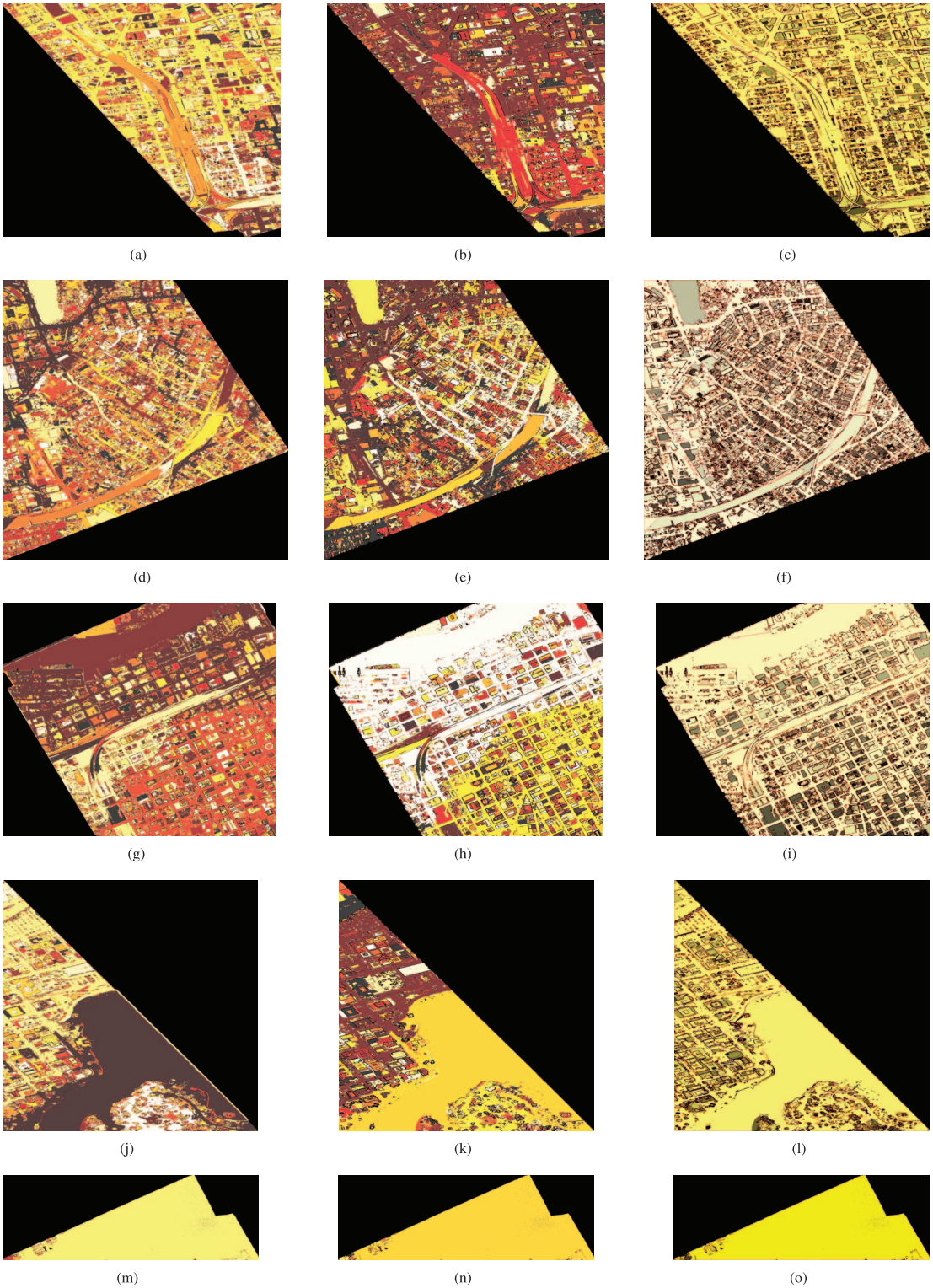


Fig. 11. (a), (d), (g), (j), (m) Point clustering. (b), (e), (h), (k), (n) Patch clustering. (c), (f), (i), (o) Boundary extraction and refinement. Boundaries are annotated in red color. Zoom in for detail.

$$E_{smooth}(f) = \sum_{(P_i^B, P_j^B) \in T} V_{P_i^B, P_j^B}\left(f\left(P_i^B\right), f\left(P_j^B\right)\right) \tag{15}$$

$$E_{smooth}(f) = \sum_{(P_i^B, P_j^B) \in T} \left[\sqrt{2} - e^{-\frac{\Delta_{(P_i^B, P_j^B)}^2}{2*\sigma^2}}\right] * \tilde{\Delta}_{(P_i^B, P_j^B)} \tag{16}$$

$$E_{smooth}(f) = \sum_{(P_i^B, P_j^B) \in T} K_{(P_i^B, P_j^B)} * \tilde{\Delta}_{(P_i^B, P_j^B)}, \tag{17}$$

where T is the set of neighboring boundary points,

$$K_{(P_i^B, P_j^B)} = \sqrt{2} - \left[e^{-\frac{\Delta_{(P_i^B, P_j^B)}^2}{2*\sigma^2}}\right],$$

TABLE 1
Processing Times for Dataset A

Subcube	Phase 2	Phase 3	Phase 4	Total
1	00:12:14	00:12:12	00:56:39	01:26:50
2	00:08:06	00:39:38	02:16:21	03:04:05
3	00:07:56	00:15:36	01:03:51	01:27:23
4	00:05:12	00:06:52	00:33:29	00:45:56
5	00:01:46	00:00:05	00:00:16	00:02:07

Time format used is hh:mm::ss. Phase 2 is point clustering, Phase 3 is patch clustering, and Phase 4 is boundary extraction and refinement.

and σ controls the smoothness uncertainty. Intuitively, if two neighboring boundary points P_i^B and P_j^B have similar orientation in the observed data, then $\Delta_{(P_i^B,P_j^B)}$ will be small, and thus there is a higher probability of $\tilde{\Delta}_{(P_i^B,P_j^B)}$ being small.

Label term: The label term penalizes each unique label that appears under a labeling f [1]. We define the label term as follows:

$$E_{label}(f) = \sum_{\vec{l} \in L} h_{\vec{l}} \cdot \zeta_{\vec{l}}(f), \tag{18}$$

where h_l is a nonnegative label cost of label $\vec{l}$ and is given by $h_{\vec{l}} = (1 - \beta_{\vec{l}})^2$ (where $\beta_{\vec{l}}$ is the mixing coefficient of the Gaussian density whose mean equals to $\vec{l}$ in (6)) and $\zeta_{\vec{l}}(f)$ is a function that indicates whether a label is unique under a labeling f and is given by

$$\zeta_{\vec{l}}(f) = \begin{cases} 1, & \exists P_i^B : f(P_i^B) = \vec{l} \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Intuitively, the label term penalizes heavily when there exist unique labels that correspond to a Gaussian density in (6) with a high mixing coefficient, i.e., a likely dominant orientation, and otherwise if the unique labels correspond to a nondominant orientation.

The energy function $E(f)$ heavily penalizes for severed edges between neighboring boundary point with similar orientation, and vice versa, which results in better defined boundaries. Moreover, it heavily penalizes any unique labels that have not been assigned to any boundary point, thus eliminating the nondominant orientations, as we demonstrate in Section 7.

Energy minimization using graph cuts: The energy minimization is performed using the α-expansion algorithm. For each iteration, the algorithm selects an orientation label $\alpha \in L$, and then finds the best configuration within this α-expansion move. If the new configuration reduces the overall energy $E(f)$ in (9), the process is repeated. The algorithm stops if there is no α that decreases the energy. The α expansion algorithm requires that the user-defined energy be regular and thus graph representable [18]. Kolmogorov also proves that any class F^2 functions of one variable are regular; hence, the label term $E_{label}(f)$ in (18) is regular. Moreover, the data term $E_{data}(f)$ in (12) is regular since the following is true:

$$E^{i,j}(0,0) + E^{i,j}(1,1) \leq E^{i,j}(0,1) + E^{i,j}(1,0). \tag{20}$$

TABLE 2
Final Resolution, Number of Patches and Surfaces, Fitting Error for Each Subcube of Dataset A, and Average Error per Surface Measured in Meters

Subcube	Final Resolution	Patches	Surfaces	Total Fitting error $\Sigma(RMS)$	Average error per surface
1	1957x1640x2048	45831	7845	4172.35	0.53
2	2010x1964x2048	97107	14824	8583.06	0.58
3	1929x1639x2048	51187	8931	5721.65	0.64
4	1796x1764x2048	31918	4706	2259.47	0.48
5	1803x595x2048	785	102	32.63	0.32

To prove that the remaining smoothness term is regular, it suffices to show that the interaction potential V is a metric. V is a metric if for any pair of labels $\alpha, \beta, \gamma \in L$ satisfies the following three conditions:

1. $V(\alpha, \beta) = 0 \Leftrightarrow \alpha = \beta$.
2. $V(\alpha, \beta) = V(\beta, \alpha) \geq 0$.
3. $V(\alpha, \beta) \leq V(\alpha, \gamma) + V(\gamma, \beta)$.

The first two conditions are trivially true. The third condition can be proven as follows for any pair of neighbors p and q and labels $\vec{\alpha}, \vec{\beta}, \vec{\gamma} \in L$:

$$V_{p,q}(\vec{\alpha}, \vec{\gamma}) + V_{p,q}(\vec{\gamma}, \vec{\beta}) - V_{p,q}(\vec{\alpha}, \vec{\beta}) = \tag{22}$$

$$K_{p,q}\{|\vec{\alpha} - \vec{\gamma}| + |\vec{\gamma} - \vec{\beta}| - |\vec{\alpha} - \vec{\beta}|\}. \tag{22}$$

$\vec{\alpha} - \vec{\gamma}$, $\vec{\gamma} - \vec{\beta}$, and $\vec{\alpha} - \vec{\beta}$ are three vectors lying on the same plane, forming a triangle. By the triangle inequality theorem that states that for any triangle the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side, the third condition must hold.

Thus, the user-defined energy $E(f)$ in (9) is regular and can be minimized by the α-expansion algorithm in $\Theta(L)$ time.

Fig. 9a shows the result of energy minimization without the use of label costs. In this case, all six initial labels shown in Fig. 8d are retained. Fig. 9b shows the result of energy minimization using label costs. In this case, only the two labels were retained corresponding to the two dominant orientations of the building. In this example, the dense boundary points resulting from the minimization are simplified using Douglas-Peucker approximation with error tolerance equal to zero, i.e., *only* colinear points are removed. Fig. 9c shows the new labeling resulting from graph cuts using label costs.

A more complicated example is shown in Fig. 10. This corresponds to a surface in dataset A. In this case, the energy minimization resulted in five different dominant orientations. The final boundaries are annotated on the image; points are purple and the lines connecting them are yellow.

6.1.3 Boundary Adjustment

The result of the boundary refinement is a classification of the boundary points with the minimal set of dominant orientations corresponding to each surface. Next, the boundary points are adjusted by projecting the x and y components of each of the 3D points onto their assigned orientation.

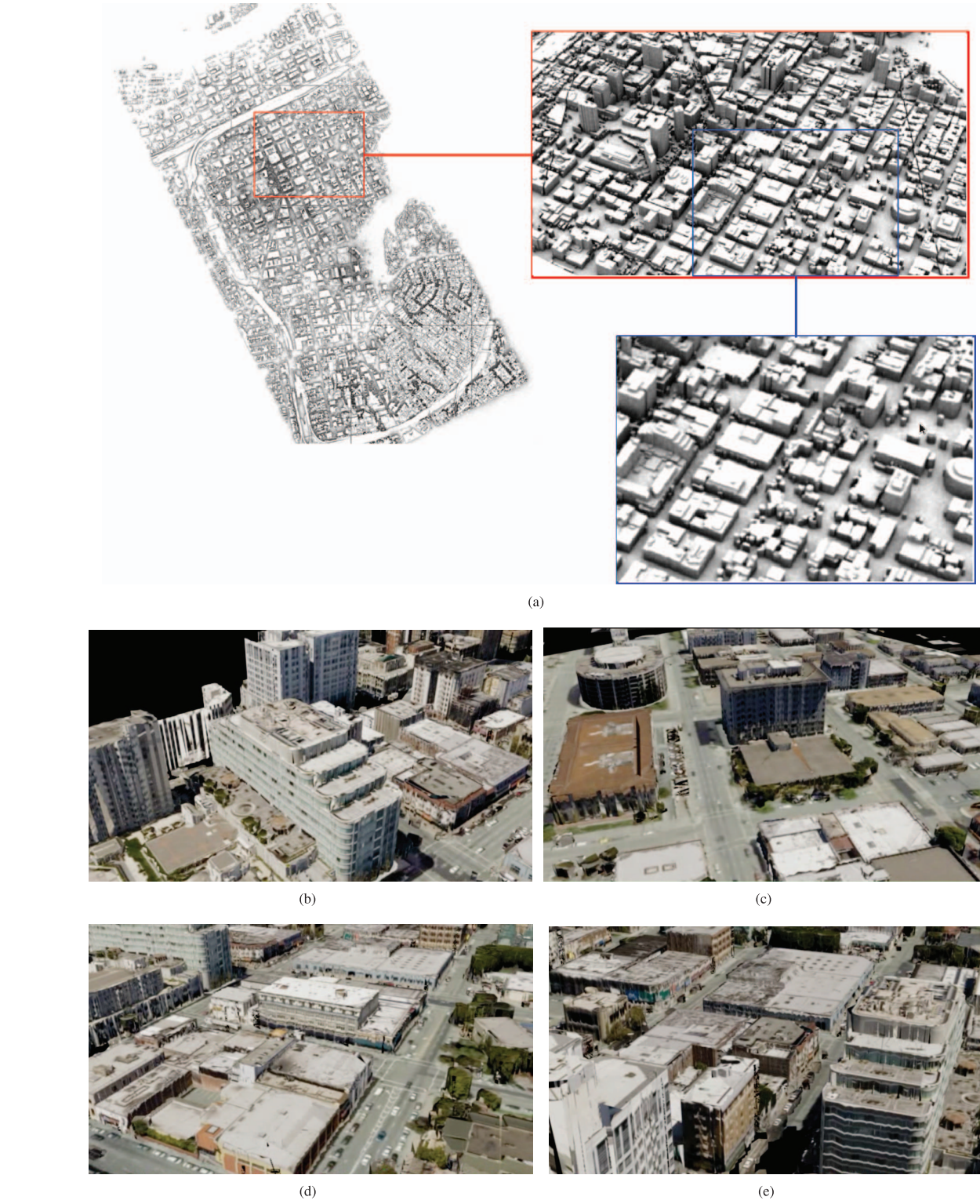


Fig. 12. (a) Renders of the 3D models corresponding to dataset A. (b)-(e) Textured 3D models of the area in (a).

7 Experimental Results

The proposed framework has been extensively tested on several datasets, and the results are reported. All results reported were conducted on an quad-core Athlon Phenom processor with 4-Gbytes RAM with the only two user-defined parameters set as $\kappa = 1.0$ and $\tau = 0.75$. The parameters remain unchanged for all datasets shown.

Fig. 2 shows Data set A for a US City. The dataset consists of about 120 million points and contains urban as well as some rural areas. During the data preparation phase of structuring and subdivision, five subcubes were generated. Each column in Fig. 11 shows the point clustering, patch clustering, and boundary extraction, and refinement results for each of the subcubes, respectively. The time performances for each of the subcubes are shown

Fig. 13. Models are created for all surfaces.

Fig. 14. Generated models for dataset C.

in Table 1. Similarly, the statistics for each of the subcubes are shown in Table 2. Phase 1 is the data preparation that completed in 00:05:45.

As can be seen from the reported times, the automatic modeling of the complete dataset A takes 06:46:21 to conclude if processed sequentially. However, as mentioned before, each subcube can be processed independently since no information sharing is needed between the subcubes. This allows for the parallel processing of the subcubes that can dramatically reduce the processing time down to 03:04:05, i.e., the processing time of the slowest subcube.

Fig. 12a shows the generated 3D models for dataset A and Figs. 10b, 10c, 10d, and 10e show the renders from novel viewpoints of the same 3D models being textured with satellite and aerial photographs. The proposed framework does not differentiate between different types of areas such as ground, trees, buildings, and so on. This, however, does not affect the accuracy of the results, as can be seen in Fig. 13, where 3D models can be seen corresponding to trees in the data in the form of cylinders.

The generated models for dataset C are shown in Fig. 14. Dataset C contains about 20 million points, and the actual size of the area it covers is about $15\ \mathrm{km}^2$. During the data preparation phase, 20 subcubes were produced with initial maximum resolution of $1\mathbf{K} \times 1\mathbf{K} \times 1\mathbf{K}$.

8 Evaluation

Finding ground truth (i.e., building blueprints) for such large-scale datasets is extremely difficult if not impossible. This makes the evaluation of the automatic modeling rather difficult. However, some qualitative and quantitative measures have been proposed for the evaluation of such results:

- *Quality*. A qualitative measure that involves the visual inspection of the resulting 3D models for imperfections such as misalignments between neighboring surfaces or any other erroneous artifacts.
- *Accuracy*. A quantitative measure that measures the deviation from the original data. We measure correctness as the root-mean-square (RMS) of the surface fitting error. The error for each subcube for dataset A is given in Table 2 and is defined as the sum of the RMS of each surface fitting. The fitting error for each surface is measured as the distance of the scanned surface points to the reconstructed surface.
- *Scalability*. A quantitative measure indicating how time varies depending on the number of points in the dataset. As mentioned in Section 5.2.2, the processing time depends on the number of points. Moreover, the experiments have shown that there is a slight delay in processing whenever there is a relatively large surface, i.e., more than 20K points, which is usually the case for the ground surface. This is due to the fact that whenever a merge of two surfaces occurs, a new normal distribution has to be computed containing all the points of the two surfaces. The processing times for dataset A are shown in Table 1.
- *Compression*. A quantitative measure indicating the compression achieved compared to the original size of the data. This measure is a very significant since if a considerable compression is achieved, it enables the use of the resulting models in a wide range of applications including computationally intensive applications such as dynamic simulations, and so on. The compression rates achieved for all datasets are well above 90 percent for all reported datasets in terms of resulting number of faces.

9 Conclusion

The proposed framework provides a complete alternative to existing interactive parameterized systems because it allows for rapid but, more importantly, automatic modeling from point cloud data. We have presented a novel unsupervised clustering algorithm P2C for the separation of the dataset into clusters of similar points. This is achieved by a hierarchical statistical analysis of the geometric properties of the points, and the results shown indicate its robustness and accuracy. Moreover, P2C has only two user-defined parameters (κ and τ) that are found to be very stable and in fact have remained unchanged for all reported results. In addition, we have presented a novel and fast boundary refinement process for the extraction, orientation classification, and orientation refinement of boundary points, where we leverage the strengths of variable-sized GMMs and the efficient energy minimization by graph cuts. The number of Gaussian components is adaptively computed based on Rissanen's criterion, and finally, only the labels corresponding to the dominant orientations are kept using energy minimization by graph cuts with incorporated label costs.

As shown, this results in smoother boundaries in the final 3D models.

In the future, we would like to focus on the improvement of the framework and, in particular, the generation of the 3D models. Currently, the 3D models are extruded based on the refined and smooth boundaries. We would like to explore the possibility of using shape matching to detect the shape of the complete building rather than just surfaces without loss of generality. Moreover, we would like to improve the boundary refinement process such that it handles nonlinear boundaries separately and not as piecewise linear.

REFERENCES

[1] A. Delong, A. Osokin, H.N. Isack, and Y. Boykov, "Fast Approximate Energy Minimization with Label Costs," *Int'l J. Computer Vision,* vol. 96, no. 1, pp. 1-27, http://dx.doi.org/10.1007/s11263-011-0437-z, Jan. 2012.

[2] F. Lafarge, X. Descombes, J. Zerubia, and M.P. Deseilligny, "Structural Approach for Building Reconstruction from a Single DSM," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 32, no. 1, pp. 135-147, http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.281, Jan. 2010.

[3] F. Lafarge and C. Mallet, "Building Large Urban Environments from Unstructured Point Data," *Proc. IEEE Int'l Conf. Computer Vision,* D.N. Metaxas, L. Quan, A. Sanfeliu, and L.J. van Gool, eds., pp. 1068-1075, http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6118259, 2011.

[4] N.J. Mitra, M. Pauly, M. Wand, and D. Ceylan, "Symmetry in 3D Geometry: Extraction and Applications," *Proc. EUROGRAPHICS Conf.,* 2012.

[5] Q.-Y. Zhou and U. Neumann, "2.5D Building Modeling by Discovering Global Regularities," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 326-333, http://dx.doi.org/10.1109/CVPR.2012.6247692, 2012.

[6] V. Verma, R. Kumar, and S. Hsu, "3D Building Detection and Modeling from Aerial LIDAR Data," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 2213-2220, http://dx.doi.org/10.1109/CVPR.2006.12, 2006.

[7] J. Xiao and Y. Furukawa, "Reconstructing the World's Museums," *Proc. European Conf. Computer Vision,* A.W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds., pp. 668-681, http://dx.doi.org/10.1007/978-3-642-33718-5, 2012.

[8] A. Adán and D. Huber, "3D Reconstruction of Interior Wall Surfaces under Occlusion and Clutter," *Proc. Int'l Conf. 3D Imaging, Modeling, Processing, Visualization and Transmission,* M. Goesele, Y. Matsushita, R. Sagawa, and R. Yang, eds., pp. 275-281, http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5954013, 2011.

[9] E. Turner and A. Zakhor, "Real Time Detection of Repeated Structures in Point Clouds of Urban Scenes," *Proc. Int'l Conf. 3D Imaging, Modeling, Processing, Visualization and Transmission,* 2012.

[10] S. Friedman and I. Stamos, "Real Time Detection of Repeated Structures in Point Clouds of Urban Scenes," *Proc. Int'l Conf. 3D Imaging, Modeling, Processing, Visualization and Transmission,* M. Goesele, Y. Matsushita, R. Sagawa, and R. Yang, eds., pp. 220-227, http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5954013, 2011.

[11] M. Isenburg, Y. Liu, J.R. Shewchuk, and J. Snoeyink, "Streaming Computation of Delaunay Triangulations," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 1049-1056, http://doi.acm.org/10.1145/1141911.1141992, 2006.

[12] Q.-Y. Zhou and U. Neumann, "A Streaming Framework for Seamless Building Reconstruction from Large-Scale Aerial LiDAR Data," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 2759-2766, http://doi.ieeecomputersociety.org/10.1109/CVPRW.2009.5206760, 2009.

[13] C. Poullis and S. You, "Automatic Reconstruction of Cities from Remote Sensor Data," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 2775-2782, http://doi.ieeecomputersociety.org/10.1109/CVPRW.2009.5206562, 2009.

[14] C. Poullis and S. You, "3D Reconstruction of Urban Areas," *Proc. Int'l Conf. 3D Imaging, Modeling, Processing, Visualization and Transmission,* pp. 33-40, http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5954013, 2011.

[15] Microsoft, "Microsoft Bing Maps," http://www.bing.com/maps/ Dec. 2012.

[16] S. Suzuki and K. Abe, "Topological Structural Analysis of Digitized Binary Images by Border Following," *Computer Vision, Graphics and Image Processing,* vol. 30, pp. 32-46, 1985.

[17] J. Rissanen, "A Universal Prior for Integers and Estimation by Minimum Description Length," *Annals of Statistics,* vol. 11, no. 2, pp. 416-431, 1983.

[18] V. Kolmogorov, and R. Zabih, "What Energy Functions Can Be Minimized via Graph Cuts," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 2, pp. 147-159, Feb. 2004.

Charalambos Poullis received the BSc degree in computing information systems from the University of Manchester, United Kingdom, in 2001, the MSc degree in computer science with specialization in multimedia and creative technologies and the PhD degree in computer science from the University of Southern California in 2003 and 2008, respectively. His area of expertise is in computer vision, computer graphics, virtual reality, and, in particular, large-scale modeling, photorealistic rendering, and 3D visualization methodologies. He is a member of the ACM and the IEEE and has served as a reviewer for several conferences and journals. He currently holds a lecturer (tenure-track) position in the Department of Multimedia and Graphic Arts, Cyprus University of Technology, where he also serves as the scientific coordinator of the Immersive and Creative Technologies lab (http://ict.cut.ac.cy).