Large-Scale Urban Reconstruction with Tensor Clustering and Global Boundary Refinement

Charalambos Poullis, *Member, IEEE*

Abstract—Accurate and efficient methods for large-scale urban reconstruction are of significant importance to the computer vision and computer graphics communities. Although rapid acquisition techniques such as airborne LiDAR have been around for many years, creating a useful and functional virtual environment from such data remains difficult and labor intensive. This is due largely to the necessity in present solutions for data dependent user defined parameters. In this paper we present a new solution for automatically converting large LiDAR data pointcloud into simplified polygonal 3D models. The data is first divided into smaller components which are processed independently and concurrently to extract various metrics about the points. Next, the extracted information is converted into tensors. A robust agglomerate clustering algorithm is proposed to segment the tensors into clusters representing geospatial objects e.g., roads, buildings, etc. Unlike previous methods, the proposed tensor clustering process has no data dependencies and does not require any user-defined parameter. The required parameters are adaptively computed assuming a Weibull distribution for similarity distances. Lastly, to extract boundaries from the clusters a new multi-stage boundary refinement process is developed by reformulating this extraction as a global optimization problem. We have extensively tested our methods on several pointcloud datasets of different resolutions which exhibit significant variability in geospatial characteristics e.g., ground surface inclination, building density, etc and the results are reported. The source code for both tensor clustering and global boundary refinement will be made publicly available with the publication on the author's website.

Index Terms—Tensor clustering, pointcloud segmentation, pointcloud tensor field, parameter-free clustering, LiDAR reconstruction, boundary refinement

1 Introduction

Large-scale urban reconstruction has long been of great interest and significant importance to the computer vision and computer graphics communities. Specifically, following the recent advances in virtual and augmented reality technologies there has been an increasing demand for robust and efficient methods for generating these virtual environments. Rapid acquisition techniques such as airborne LiDAR have been around for many years and are indeed capable of capturing very large areas in a single deployment however the difficulties arising with processing the captured data considerably limit their uses. For one, as with every scanning, noise is introduced in the measurements due to possible system error or sensor miscalibration. In addition, the zig-zag nature of the scanning almost always produces spurious measurements at object boundaries which manifest themselves as jagged edges in the data. Another significant limiting factor is the resolution or sampling density of the captured data which depends on the sampling rate of the LiDAR sensor as well as the flying altitude of the aircraft as explained in [1].

Even with the tremendous advances in remote sensing technologies given the above mentioned capture characteristics, processes in current practice still require trained personnel with extensive experience in order to produce models useful in the end application, i.e., lightweight, polygonal 3D models. The process is expensive since it requires manual or at best semi-automatic work, and is human effort intensive and slow. A primary cause for this is the fact that existing state-of-the-art systems for large-scale urban reconstruction require a plethora of parameters [e.g., number of user-defined non-adaptive parameters in ([2] $\simeq 15$, [3] $\simeq 12$) which have to be carefully tweaked by the user since these often depend on the input data characteristics. This sensitivity to the input data and the large choice of multiple parameters are always major concerns in the application of these methods in practice. Their optimal values are not known and cannot be easily computed either. This makes it necessary for the user/operator to experimentally search in this large parameter space for values that yield the desired quality of 3D models, resulting in making this process so difficult and time consuming. Another serious limitation with existing solutions is scalability; primarily in terms of the size of area which can be successfully processed and second, in terms of performance i.e., how long it takes to generate the results.

To summarize, there still exists a wide gap between the current state-of-the-art and the desired goal of automated large-scale urban reconstruction of real-world areas, for applications requiring digital 3D environments.

In this paper we address the difficult problem of large-scale urban reconstruction and propose a novel and automatic

• *The author is with the Immersive and Creative Technologies Lab, Department of Computer Science and Software Engineering, Concordia University, Montreal, Quebec H3G 1M8, Canada. E-mail: charalambos@poullis.org.*

Manuscript received 27 Jan. 2018; revised 23 Aug. 2018; accepted 12 Jan. 2019. Date of publication 18 Jan. 2019; date of current version 1 Apr. 2020.
(Corresponding author: Charalambos Poullis.)
Recommended for acceptance by S. Savarese.
Digital Object Identifier no. 10.1109/TPAMI.2019.2893671

solution for generating simplified, polygonal 3D models. The proposed technique takes as input the raw, unstructured, incomplete, and noisy pointcloud captured by an airborne LiDAR scanner during multiple sweeps, and first separates it into a set of smaller components. Each component is a resampled XYZ map containing a different part of the data as in [1] which is then processed independently and concurrently to extract various metrics about the points in the maps. This technique adaptively calculates the number of XYZ maps according to the user-suggested map resolution. Next, information extracted directly from this data is encoded using a tensorial representation. This representation allows multiple types of information about each point to be encoded at the same time. A robust tensor clustering algorithm is proposed for segmenting the tensors into clusters representing geospatial objects such as buildings, cars, trees, etc. Most significantly, our clustering method has no input data dependencies and does not require any user defined parameter making it distinct from earlier solutions for this problem. It is based on adaptive computation of the statistical parameters of the underlying distribution for points belonging to a cluster. The result of the clustering is a set of contiguous clusters of points which are further processed in order to extract the boundaries. This is achieved with a new multi-stage boundary extraction and refinement process which reformulates the boundary extraction as a global optimization problem. Unlike existing work, our proposed technique requires no user interaction, makes no assumptions on the input data, such as requiring Manhattan-style building orientations, and is able to process the entire data without any user inputs, i.e., does not require like in earlier work that the user accurately marks the boundaries of every single building before reconstructing it. Our technique has been extensively tested on five different datasets of different resolutions/densities which have significant variability in the ground surface elevation/inclination, building density, type, etc.

The rest of the paper is organized as follows: Section 2 provides a brief overview of the state-of-the-art in the area of large-scale reconstruction. In Section 3 we provide a technical overview of the proposed solution. The extraction of the information from the pointcloud data and its encoding to tensors is presented in Section 4. In Section 5 we present Tensor Clustering and in Section 6 the global boundary refinement. Section 7 reports on experimental results and evaluation, and lastly, Section 8 has the conclusion and future work.

2 RELATED WORK

Many algorithms and systems have already been proposed for the problem of urban reconstruction by researchers in computer vision and graphics communities. A comprehnsive summary can be found in [4]. The most relevant to the proposed work are categorized according to the line of approach they followed and summarized below.

First, there are techniques which use geometric primitives. In [5] the authors proposed a system which included a minimal set of three parameterized-primitives which the operator could use to model any type of linear and non-linear surfaces. Fast forward to the current state-of-the-art, there are techniques [6] and [7] which given a set of points automatically produce a set of primitives describing the site. However, these approaches require that the input points correspond only to a single building. In fact, one of the most difficult tasks of reconstructing large urban areas is the automatic detection of buildings and other components. Which is why, these approaches combine manual detection with automatic extraction.

A different line of approach uses symmetries and regularities. Extensive work using this approach has already been done and has shown to yield impressive results, but mostly for small scale objects [8]. In [9] this same approach is used to propose a system for urban reconstruction. However, again, solutions based on this approach require that the detection is performed manually.

Finally, a rather different approach by [10] used inverse constructive solid geometry techniques. Rather than using boolean operations on simple primitives to generate a complex structure, they start off with a point cloud representing the indoor area of a structure and decompose that into layers which are then grouped into higher-order elements.

Perhaps the closest work related to the proposed tensor clustering is the work of [11] on multi-type feature extractor and classifier. Information extracted from color satellite images is represented using tensors. Next, the tensors are decomposed and pixels are classified as junctions, curves or surfaces via graph-cut optimization. In our work, we employ 3D data and propose a different way to encode extracted information into tensors. Furthermore, there is no classification but instead all comparisons are performed in terms of the tensorial representation of each point.

In summary, previous solutions require extensive user input, in the form of manual identification of components in the urban area and/or in the form many threshold values to be tweaked on a case by case basis. As a result, these solutions do not scale well to large urban areas. In contrast to the above work, in this paper, we present an automatic urban reconstruction system which requires no user interaction, and yet efficiently generates accurate urban reconstructions. Our technical contributions are:

- An elegant representation of all the multiple information at each point in a LiDAR depth map encoded in the form of a single-second order symmetric tensor. The tensor encapsulates into it the property of a point belonging to the surface, curve and junction categories without the need for various user specified thresholds. This formulation which fuses per point information into a single entity leads to considerable simplification of the similarity comparison between points.
- A robust clustering technique for depth maps captured by LiDAR scanners which retains important details without the need for user defined thresholds and yields better results than earlier solutions.
- An innovative method of computing the parameters needed for region growing to group points into clusters based on adaptive computation of per-point and per-cluster statistical parameters. The Weibull probability distribution function (pdf) is assumed for points within a cluster. The Weibull pdf parameters are dynamically updated as new points get added to the cluster. The clustering results achieved are superior to previous results.

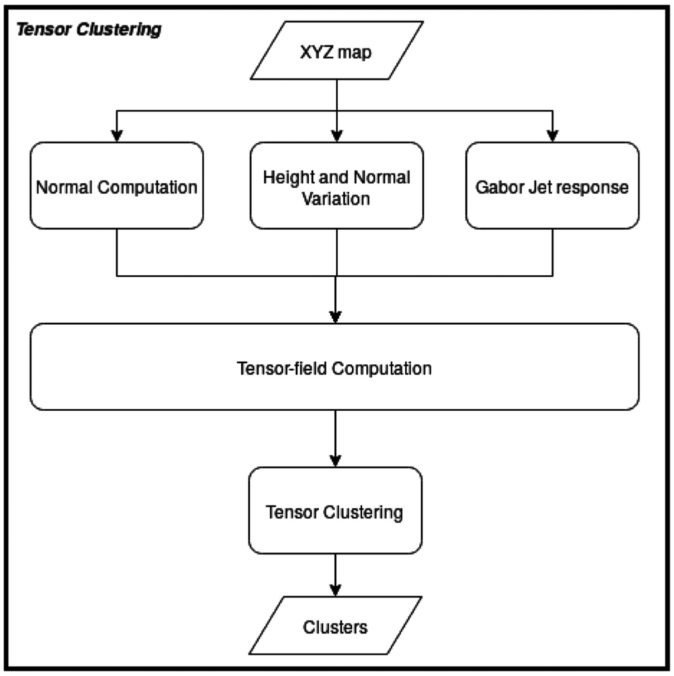


Fig. 1. An overview of the proposed clustering technique.

- A new multi-stage method of boundary extraction and refinement which reformulates this as a solution to a global optimization problem.

3 Technical Overview

Our solution involves the clustering first, followed by the boundary extraction. An overview of the proposed clustering technique is shown in Fig. 1. The LiDAR data is essentially an XYZ map consisting of 3D points. A crucial step in the 3D reconstruction using this data is to segment these points into surface patches and boundary curve segments. For this, first, a set of per-point metrics are computed from the input XYZ map and these are encoded into tensors. The per-point metrics include, the normal, height, local height variation and local normal variation. In addition, edge saliency is computed for multiple directions using multi-scale, multi-frequency filtering with Gabor jets. The encoding into the tensorial representation is based on the following reasoning. Surfaces should have low/constant height variation and low/constant normal variation and low curve response, while curves should have high height variation and high normal variation and high curve response. This formulation is presented in Section 4. The tensors are then clustered together based on their similarity. This clustering is presented in Section 5. The method for adaptive computation of per-point and per-cluster statistics used in region growing for cluster formation is discussed in Section 5.3 using a Weibull distribution of the points. Lastly, given a set of observations/samples, the estimation of the shape α and scale β parameters of the Weibull distribution is performed using Maximum-Likelihood Estimation and described in detail in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2019.2893671.

The clusters are then further processed to extract region boundaries. The pipeline for the boundary extraction and refinement process is shown in Fig. 2. First, the clusters' boundaries are extracted and grouped into neighbourhoods as described in Section 6.2. Next, the dominant orientations of each cluster are computed using Principal Component Analysis (PCA) and are used to compute the globally dominant orientations for the entire scene. To avoid assumptions on the number and type of orientations we employ Gaussian Mixture Models (GMMs) and use a Minimum Description Length (MDL) criterion to specify the number of Gaussian components. The boundaries are then refined based on the globally dominant orientations and a global optimization step which ensures that the boundary positions are optimal as described in Section 6.3.

Finally, simplified, polygonal 3D models are created from the boundaries.

4 Encoding Using the Tensorial Representation

The XYZ map contains points in 3D euclidean space. The objective of this phase is the clustering of similar neighbouring points. As has been current practice, this has involved computing of various metrics at each point in order to determine the similarity between them. Typically, these metrics can be derived directly from the XYZ map containing the points and many have already been reported: height h, normal $\vec{N}$, surface fitness error, height variation, normal variation, etc. Once the metrics have been computed the decision whether two points P_1 and P_2 are similar or not is expressed as either a combination $D_{component-wise}$ of the results of the per-metric comparisons:

$$D_{component-wise} = d_h(H_{P1}, H_{P2}) \; < \; \tau_H \bigwedge d_N(N_{P1}, N_{P2}) \; < \; \tau_N \bigwedge ..., \tag{1}$$

or as $D_{combined}$ the result of a single comparison between two N-dimensional feature descriptors $f_{P_1} = < H_{P1}, N_{P1}, ... >$ and $f_{P_2} = < H_{P2}, N_{P2}, ... >$ containing the N metrics at each point:

$$D_{combined} = d_f(f_{P_1}, f_{P_2}) \quad < \; (\tau_H, \tau_N, ...), \tag{2}$$

where d_h, d_N, d_f are distance functions for the height, normal, and feature descriptors respectively.

In either case, there is an explicit requirement that a set of thresholds $\tau_H, \tau_N, \ldots$ is specified which renders most

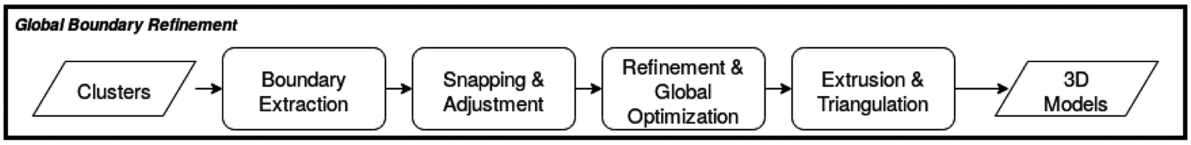

Global Boundary Refinement
Clusters
Boundary Extraction
Snapping & Adjustment
Refinement & Global Optimization
Extrusion & Triangulation
3D Models

Fig. 2. An overview of the proposed boundary refinement technique.

proposed techniques dependent both on the dataset and also on the user to provide the right values for each of the thresholds. Further, this requirement implies the inherent assumption that the linear hyper-plane defined by the specified thresholds divides the N-dimensional feature space into two parts where all points lying above the hyper-plane are similar and all below are not similar. Although for low dimensional feature spaces this may often be true, when dealing with higher dimensions this does not hold and can negatively impact the accuracy of the results.

In this work, we eliminate the above requirement and instead propose a third option: combine the metrics computed at each point into a single entity: a second-order symmetric tensor. This choice was based on the fact that a second-order symmetric tensor can encode information about multiple geometric types passing through each point and therefore can encode all the information extracted by the metrics. More importantly, it enables us to present a solution which does not require the user to guess even a single threshold value for it to function.

In the following section we describe the steps required to extract and encode the information into tensors. First, we begin by computing several per-point metrics from the input XYZ map required for subsequent processing. This is described in Section 4.1. Next, the metrics are encoded into tensors and a tensor field representing the 3D scene is computed. This is explained in Section 4.2. Finally, the information encoded in the tensors is used for clustering the points into surface patches and boundary lines.

4.1 Per-Point Metrics

In order to proceed with the classification, the following per-point metrics are first computed using the XYZ map containing the points in raster form:

- Normal computation. By taking into account the local 8-neighbourhood at each point P, a normal $\vec{N}_P$ is defined as the mean of the eight neighbouring normals. In our experiments the neighbourhood is defined as the 3×3 neighbouring points around point P since it provides the fastest and best results. The neighbouring normals are computed as the cross product of two vectors formed by connecting consecutive neighbouring points $P_i, P_{(i+1) \bmod 8}$, where $0 \leq i \leq 8$, to point P and is given by, $\vec{N}_{P_{i,(i+1) \bmod 8}} = (\vec{P}_i - P) \times (\vec{P}_{(i+1) \bmod 8} - P)$. Hence, the normal at point P is defined as $\vec{N}_P = \frac{1}{8}\sum_{i=0}^{8}(N_{P_{(i,(i+1) \bmod 8)}})$
- Height variation. The height variation is defined as the local neighbourhood height variation of point P and is given by,

$$H^P_{var} = \frac{||h_P - h_{min}||}{(h_{max} - h_{min})}, \tag{3}$$

  where h_P is the height at point P, and h_{min}, h_{max} are the minimum and maximum heights in the neighbourhood, respectively. In our experiments the neighbourhood is defined as the 7×7 neighbouring points around point P.
- Normal variation. Similar to the height variation, the normal variation is defined as the local neighbourhood normal variation of point P and is given by,

$$N^P_{var} = ||nd_{max} - nd_{min}||, \tag{4}$$

  where $nd_{max} = 1 - d_{max}$ and $nd_{min} = 1 - d_{min}$. d_{max} and d_{min} are the maximum and minimum dot products respectively, between vectors formed by connecting consecutive neighbouring points to point P. For example, the N^P_{var} at point P located in a neighbourhood where all points have similar(or equal) normal orientation will have a value closer (or equal) to zero.
- Gabor response. A bank of Gabor jets is applied on the input XYZ map at different frequencies (i.e., $\aleph_f = 5$) and orientations (i.e., $\aleph_\Theta = 16$). Since the XYZ maps are essentially depth maps (with X,Y coordinates) the Gabor jets respond to oriented depth discontinuities. The resulting response r^θ_P at each point P corresponding to the same orientation θ but different frequencies are added together to form a *per-orientation* response image. The combination of the multiple-scales per orientation accounts for features appearing at different scales.

The metrics H^P_{var}, N^P_{var} and r^θ_P are normalized and range between [0, 1].

4.2 Tensor Field Computation

The per-point metrics described in Section 4.1 are encoded into a second-order symmetric tensor T_P for each point P. A second-order symmetric tensor T is defined as $T = \lambda_1\vec{e}_1\vec{e}_1^T + \lambda_2\vec{e}_2\vec{e}_2^T + \lambda_3\vec{e}_3\vec{e}_3^T$ where $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ are eigenvalues, and $\vec{e}_1$, $\vec{e}_2$, $\vec{e}_3$ are the eigenvectors corresponding to $\lambda_1, \lambda_2, \lambda_3$ respectively. Using the Spectral theorem, the tensor T can be decomposed into a linear combination of three basis tensors(ball, plate and stick) as in

$$\begin{aligned} T = (\lambda_1 - \lambda_2)\vec{e}_1\vec{e}_1^T + (\lambda_2 - \lambda_3)(\vec{e}_1\vec{e}_1^T + \vec{e}_2\vec{e}_2^T) \\ + \lambda_3(\vec{e}_1\vec{e}_1^T + \vec{e}_2\vec{e}_2^T + \vec{e}_3\vec{e}_3^T). \end{aligned} \tag{5}$$

In Eq. (5), $(\vec{e}_1\vec{e}_1^T)$ describes a stick(surface) with associated saliency $(\lambda_1 - \lambda_2)$ and normal orientation $\vec{e}_1$, $(\vec{e}_1\vec{e}_1^T + \vec{e}_2\vec{e}_2^T)$ describes a plate(curve) with associated saliency $(\lambda_2 - \lambda_3)$ and tangent orientation $\vec{e}_3$, and $(\vec{e}_1\vec{e}_1^T + \vec{e}_2\vec{e}_2^T + \vec{e}_3\vec{e}_3^T)$ describes a ball(junction) with associated saliency λ_3 and no orientation preference.

A tensor T_P is computed for each point P as the weighted sum of $\aleph_\Theta$ tensors corresponding to the orientations of the Gabor jets and is defined as,

$$T_P = \frac{1}{\aleph_\Theta}\sum_{\theta=1}^{\aleph_\Theta} T_\theta, \tag{6}$$

where T_θ is the tensor corresponding to the Gabor filter orientation θ and is calculated as described in the next section.

4.2.1 Eigenvectors

The eigenvectors $\vec{e}_1, \vec{e}_2, \vec{e}_3$ of the tensor T_θ form an orthonormal basis system in which the normal orientation $\vec{e}_1$ is aligned with the normal $\vec{N}_P$ at point P, and the tangent orientation $\vec{e}_3$ is aligned with the orientation $\frac{2\pi}{\theta}$ of the Gabor jet.

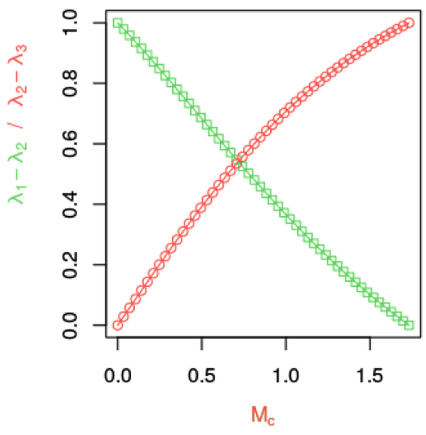


Fig. 3. Relation between the eigenvalue differences $\lambda_1 - \lambda_2$ and $\lambda_2 - \lambda_3$ with respect to the magnitude M_c which is used to calculate the eigenvalues.

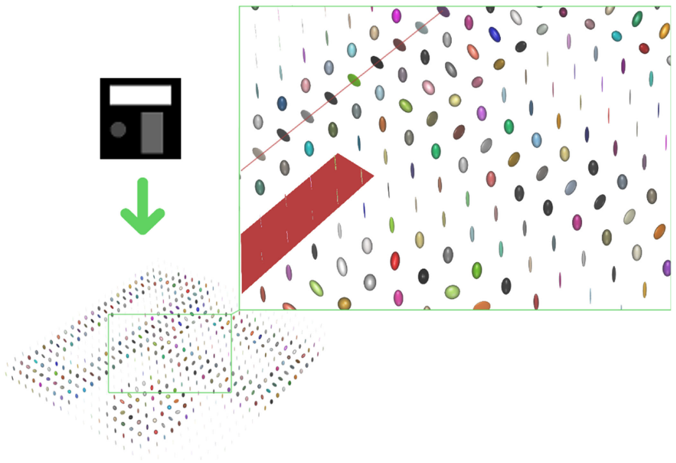


Fig. 4. Visual interpretation of the encoded tensors for the synthetic depth map shown. Tensors corresponding to points on a curve appear as ellipsoids with a plate-like shape where the normal to the plate is the tangent of the curve; one such case is shown with a straight red line. Tensors corresponding to points on a surface appear as ellipsoids having a stick-like shape where the direction of the stick is the normal to the surface; one such case is shown with a red plane.

$\vec{e}_2$ is computed as the cross product of $\vec{e}_1$ and $\vec{e}_3$ and finally, $\vec{e}_3$ is recalculated as the cross product of $\vec{e}_1$ and $\vec{e}_2$. The recalculation of $\vec{e}_3$ is essential to ensure a proper orthonormal basis system, since the initial orientation θ is possible to be a projection of the actual vector. Hence, the eigenvectors are given by,

$$\vec{e}_1 = \vec{N}_P \tag{7}$$

$$\vec{e}_3 = \langle cos(\theta), sin(\theta), 0\rangle \tag{8}$$

$$\vec{e}_2 = \vec{e}_1 \times \vec{e}_3 \tag{9}$$

$$\vec{e}_3 = \vec{e}_1 \times \vec{e}_2. \tag{10}$$

4.2.2 *Eigenvalues*

In what follows, let M_c be defined as the magnitude of the vector $\vec{c} = \langle r^i_P, H^P_{var}, N^P_{var}\rangle$. In order to determine the eigenvalues we define three equations as follows.

The tensor as defined in Eq. (6) is modelled after Gabor responses (the response is high only for a tensor belonging to an edge in a specific orientation), its "junction-ness" or junction-saliency should be 0 and hence we set $\lambda_3 = 0$. This provides us our first equation. The other two equations are derived based on the following observations:

- Points lying on a curve produced by a depth discontinuity—which is the always the case for XYZ maps—have high response to the Gabor filters and high normal and height variation. Thus, a point on a curve will have a maximal M_c. The vector $\vec{c}$ measures the "curve-ness" or curve-saliency of a point and the range of its magnitude is $0 \leq M_c \leq \sqrt{3}$].
- On the other hand, points lying on a surface have no (or low) response to the Gabor filters and low normal and height variation. Thus, a point on a surface will have a minimal M_c. Specifically, the "surface-ness" or surface-saliency of a point on a surface is defined as $M_s = \frac{\sqrt{3}-M_c}{\aleph_\Theta}$ where $\aleph_\Theta$ is the number of Gabor filter orientations. Note that the division by the number of Gabor filter orientations is imperative because points on curves have a high response to only one filter orientation; whichever is aligned to the curve. Although the resulting tensor at each point is the sum of $\aleph_\Theta$ tensors, $\aleph_\Theta - 1$ of those will not have high curve-saliency but will instead have high surface-saliency. Hence, the normalization ensures that there is no unjustified increase in surface-saliency.

From Eq. (5), we note that the surface-saliency of a point is given by $\lambda_1 - \lambda_2$ and the curve-saliency of a point is given by $\lambda_2 - \lambda_3$. Hence, following the above observations we get the other two equations with the unknown eigenvalues: $\lambda_1 - \lambda_2 = M_s$ and $\lambda_2 - \lambda_3 = M_c$.

Solving the three equations for the three unknown eigenvalues we get

$$\langle \lambda_1, \lambda_2, \lambda_3\rangle = \left\langle \frac{\sqrt{3} - M_c + M_c * \aleph_\Theta}{\aleph_\Theta}, M_c, 0 \right\rangle, \tag{11}$$

where $\aleph_\Theta$ is the number of Gabor filter orientations. Fig. 3 shows the relation between the eigenvalue differences $\lambda_1 - \lambda_2$ and $\lambda_2 - \lambda_3$ with respect to the magnitude M_c, used to compute the eigenvalues. The λ values have been scaled by a factor of $(1/\sqrt{3})$ to get them in the 0-1 range. As it can be seen, a point lying on a curve will have a high M_c. Therefore the eigenvalue corresponding to the curve-saliency will be higher i.e., $\lambda_2 - \lambda_3$.

Fig. 4 shows the results of the above tensor encoding for a synthetic image. The geometric interpretation of the tensor is an ellipsoid in 3D space. There are three basis cases which define the possible variations of the ellipsoid:

1) Tensors corresponding to points on a curve appear as ellipsoids with a plate-like shape where the normal to the plate is the tangent to the curve. This tangent corresponds to the eigenvector corresponding to the smallest eigenvalue i.e., $\vec{e}_3$ of the tensor.
2) Tensors corresponding to points lying on a surface appear as ellipsoids with a stick-like shape where the orientation of the stick is the normal to the surface. This normal corresponds to the eigenvector with the largest eigenvalue i.e., $\vec{e}_1$ of the tensor.

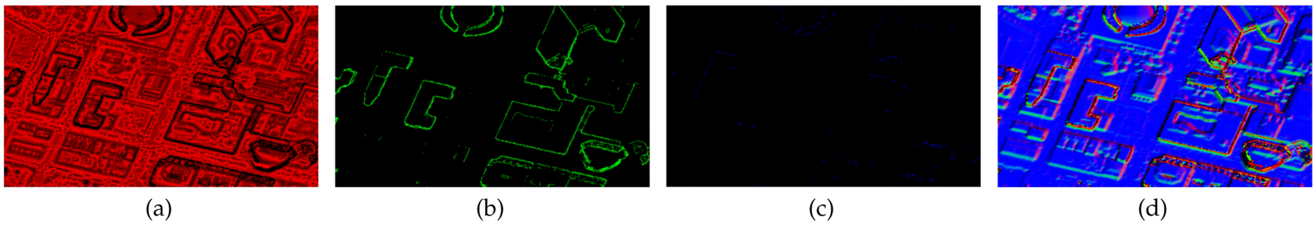


Fig. 5. Application of the Spectral theorem: each tensor is decomposed into three basis tensors. Figures (a), (b) and (c) depict the points classified as surfaces, curves or junctions respectively, according to their eigenvalue differences as explained in Eq. (5). Figure (d) shows the orientation corresponding to each type i.e., for surfaces it represents the normal to the surface, for curves it represents the tangent to the curve, and for junctions it appears black.

3) Tensors corresponding to points where there is no curve nor surface appear as perfect spheres/balls since there is no preference towards a particular orientation at those locations.

Examples are shown in Fig. 4 for the first two cases; the red straight line drawn shows a sequence of neighbouring tensors corresponding to points on the same straight line and the red planar surface drawn shows neighbouring tensors corresponding to points lying on the same surface.

An alternative validation procedure would be to apply the Spectral theorem to decompose the tensors into the three basis tensors i.e., stick - Fig. 5a, plate - Fig. 5b, and ball—Fig. 5c. The eigenvalue differences can then be used to classify each point into a surface, curve or junction as described in [12] and is shown in Fig. 5.

5 Clustering

Clustering is performed on the tensor field using a region growing approach. Tensors in the same region are grouped together based on their similarity. To achieve this, we first define a similarity measure between two tensors, then introduce a comparison condition for deciding when a new tensor should be added to an existing cluster, and finally use this condition to grow the cluster by applying it to all neighbour tensors of that cluster. A unique feature of this method is that even though we have all the different per point metrics as input, unlike all other methods, we do not need to ask the user to define a large number of threshold values. All the terms in the comparison are adaptively computed for each new tensor.

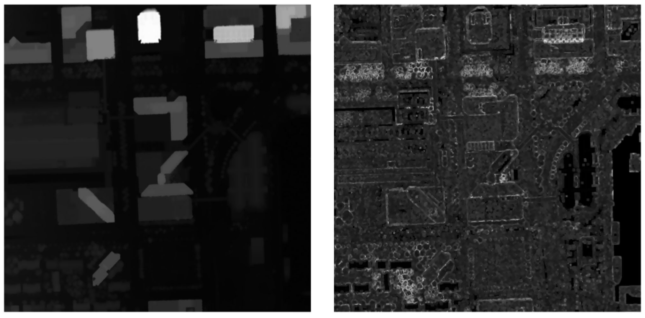


Fig. 6. (b) Similarity variation measured as the maximum minus the minimum similarity between the tensors in the 8-neighbourhood of each point in the depth map shown in (a). Note that the color curve for (b) has been adjusted for easier readability.

5.1 Similarity Measure

We define the similarity measure between two tensors T_i and T_j as,

$$d_{(T_i,T_j)} = 1 - \frac{trace(T_i.T_j)}{||T_i||.||T_j||}, \tag{12}$$

where $||.||$ is the Frobenius norm. The range of this similarity measure is $0 \leq d_{(T_i,T_j)} \leq 1$ and $d_{(T_i,T_j)} = 0$ iff the two tensors being compared are identical. Although several other similarity measures have been reported (e.g., [13], [14], [15]), this measure, which was first introduced by [16], is preferred because it does not require the eigen-value/-vector decomposition of the two tensors, and hence is faster to calculate. Fig. 6b shows the similarity variation measured as the maximum minus the minimum similarity between the tensors in the 8-neighbourhood of each point in the depth map shown in Fig. 6a.

5.2 Deciding Whether Two Tensors Are Similar

In the region growing method for clustering, typically at this point various thresholds need to be defined to determine whether two tensors are similar or not. As mentioned before, in contrast to existing techniques, we do not require user-defined thresholds at all. Instead we propose a new technique for adaptively computing a comparison condition using the *per-candidate* and *per-cluster* statistics as its basis. This is described next.

5.2.1 Per-Cluster Statistics

Assume that a cluster C contains N tensors T_i^C, $1 \leq i \leq N$ at iteration/time t. First, we compute the cluster's average tensor $\bar{T}^t = \frac{1}{N}\sum_{i=1}^{N} T_i^C$ at iteration/time t. Second, we calculate the probability distribution function (pdf) ϕ_W^C of the similarity distances between the tensors T_i^C contained in the cluster and the average tensor $\bar{T}^{t_i}$ at the time t_i that the tensor T_i^C was added to the cluster C. Thus the pdf gets continuously updated as new tensors get added to the cluster and in turn influences which new tensors can be added to the cluster in an adaptive manner.

The pdf ϕ_W^C for a cluster C is modeled as a Weibull distribution. It has already been shown [17] that Extreme Value distributions and in particular the Weibull distribution significantly outperforms other distributions such as Gaussian, Student-t, etc, in cases where the observations represent *similarities* and therefore are closer (or equal) to zero, leading to zero-mean and/or zero-variance. This is clearly brought out in Fig. 7, which shows three distributions being applied on the same set of tensors corresponding to the surface points of

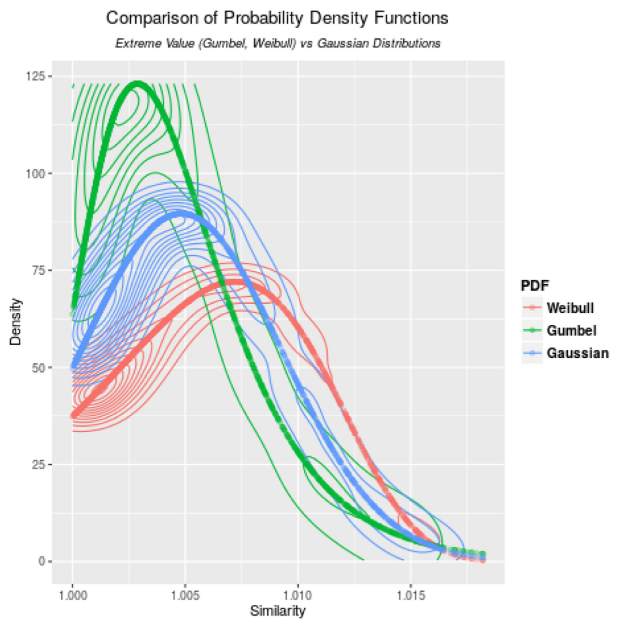


Fig. 7. Comparison between two extreme value distributions (Gumbel, Weibull) and the Gaussian distribution for the set of tensors corresponding to the surface points of the marked area shown in Fig. 8a. As it can also be visually confirmed, the Weibull distribution can provide a more accurate representation.

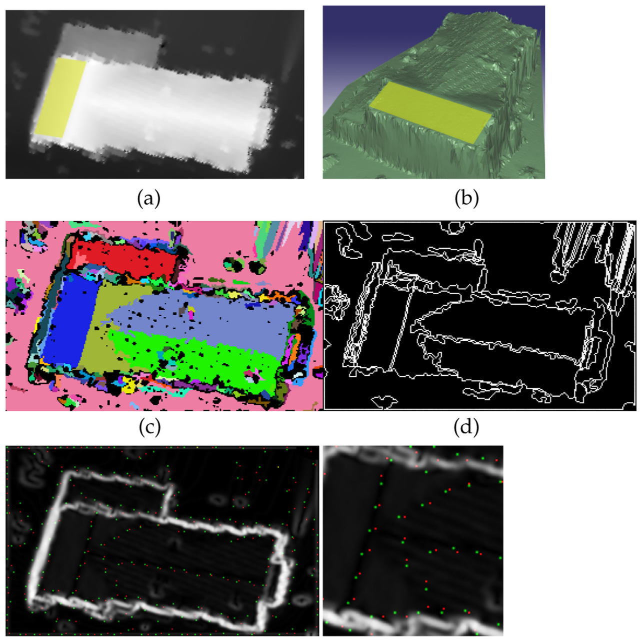


Fig. 8. (a) The normalized XYZ depth map of a building. The three distributions were tested on the tensors corresponding to the surface points in the marked area. (b) The mesh corresponding to the depth map in (a). (c) Color-coded clusters. (d) The complete sparse boundary map corresponding to the clustered regions in (c). (e) The boundary positions after snapping and adjustment are shown in red. The optimized boundary positions are shown in green. A closeup is shown in (f).

the area shown in Fig. 8a; two extreme value distributions Gumbel and Weibull, and the Gaussian distribution.

Thus, the pdf ϕ_W^C for a cluster C is given by,

$$\phi_W^C = \left(\frac{\alpha}{\beta}\right)\left(\frac{x}{\beta}\right)^{a-1} e^{-(\frac{x}{\beta})^{\alpha}}, \tag{13}$$

where $\alpha > 0$ is the shape parameter, $\beta > 0$ is the scale parameter, and x is the observation i.e., the similarity measure defined in Eq. (12). In practice, because of the many logarithm calculations involved when fitting the Weibull distribution, and the fact that some similarity measures are close to zero [and the logarithm of zero is undefined], x represents the shifted similarity measure given by $1 + d_{(T_i,T_j)}$ such that the minimum value of the range of the observations is 1, rather than 0.

The mean of ϕ_W^C is defined as $\mu_\phi = \beta\Gamma(1+\frac{1}{a})$ and the variance as $\sigma_\phi^2 = \beta^2[\Gamma(1+\frac{2}{\alpha}) - \Gamma^2(1+\frac{1}{\alpha})]$, where $\Gamma(.)$ is the gamma function given by $\Gamma(n) = \int_0^\infty e^{-x}x^{n-1}dx$.

Given a set of observations/samples, the estimation of the shape α and scale β parameters of the Weibull distribution is done using Maximum-Likelihood Estimation as described in detail in the Appendix, available in the online supplemental material.

5.3 Cluster Formation

Region growing proceeds as follows. A candidate tensor T_{new} considered for inclusion in cluster C is first compared with the average tensor T_C of the cluster and added to the cluster iff the probability of the similarity measure $\phi_C^W(d_{(T_{new},T_C)})$ is higher than the probability of the mean μ_C perturbed by the standard deviation σ. Thus if the following comparison condition is true,

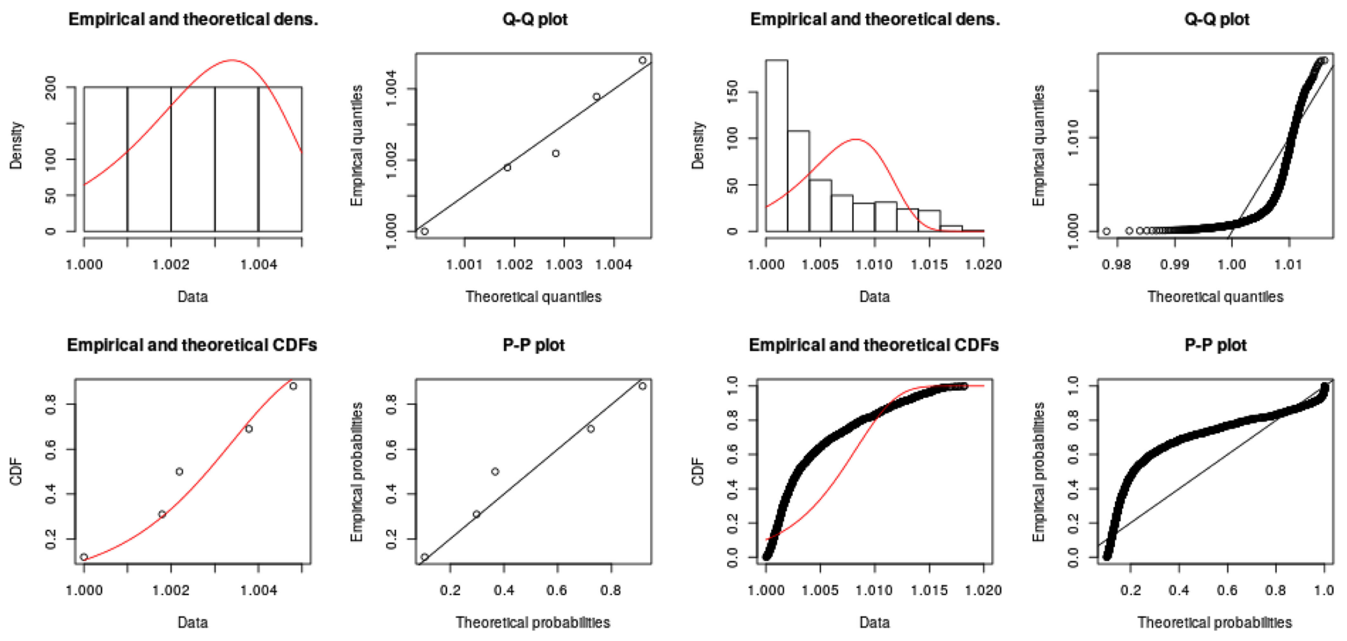
$$\phi_C^W(\mu_C + \sigma_C) \leq \phi_C^W(d_{(T_{new},T_C)}) \leq \phi_C^W(\mu_C), \tag{14}$$

the candidate tensor T_{new} is added to the cluster. The choice for this is based on the *empirical rule* for probability models which states that about 67 percent of the values are contained within one standard deviation of the mean. This has also been verified in practice and has proven to be stable over different datasets.

A cluster C_0 is initialized with the first tensor T_0 in the tensor field. All tensors in the 8-neighbourhood of T_0 are considered for inclusion. A candidate tensor for which the comparison in Eq. (14) is true is added to the cluster C_0 and the statistics are updated. Moreover, the neighbours of the newly added tensor are also considered for inclusion in C_0. This process is repeated until all neighbouring tensors [neighbours of all tensors contained in the cluster] are considered for inclusion into the cluster C_0. A new cluster C_1 is then initialized with a neighbouring tensor for which the comparison was false, or had not been yet considered. This process is repeated until all tensors are considered and belong to some cluster.[1]

Typically region growing algorithms are very sensitive to the initialization however in this case changing the initial starting point will primarily affect the sequence in which the clusters are being formed and not so much the final outcome. It is exactly for this reason (i.e., to address the problem of sensitivity/robustness) that a distribution of the tensor *similarities* is calculated for each cluster rather than a distribution of the per-cluster metrics e.g., height, normal, etc. The samples used to calculate the distribution have very small values and are close to zero e.g., < 0.03, hence the decision for using an

1. An animation of the clustering algorithm is shown https://youtu.be/E5Xuj5k7vr4here



(a) 5^{th} iteration, 5 samples (b) 1925^{th} iteration, 1925 samples

Fig. 9. The shape and scale parameters of the distribution [and therefore the mean and variance] converge to the same values as the number of iterations [and therefore samples] increases. The results shown correspond to the surface shown in Figs. 8a and 8b.

Extreme Value Distribution to model these; and in particular the Weibull distribution which can be calculated with a much smaller sample size than other distributions. These (Weibull distribution and small sample size) are what make the region growing algorithm more robust to changes in the initialization, since a good fit for the distribution will be available even after the first few points.

Eq. (14) formulates the comparison in such a way that after each addition to a cluster the comparison also is updated according to the latest cluster's statistics. Thus every comparison test is potentially different from the previous. In practice, during the initial stages of the clustering there are moderate changes in the mean μ_C and variance σ_C which leads to moderately different comparison tests; once a cluster contains enough tensors these deviations are diminished since fitting the distribution converges to the same/similar set of parameters. Fig. 9 shows an example of a cluster's statistics during the initial iterations i.e., 5th containing 5 samples, and the final i.e., 1925th containing 1925 samples.[2]

As previously mentioned, fitting a Weibull distribution involves many logarithm calculations which as a result significantly increase the computation time. To address this, we re-fit the Weibull distribution at every new insertion until it reaches a stable level. Our experiments have shown that the changes occurring to the mean and variance of a cluster's distribution dramatically reduce after the first 50 samples. Thus, after the first 50 iterations, we opt out of recalculating the mean and variance unless (i) the ratio between the last two mean estimations and the ratios between the last two variance estimations is less than 95 percent or, (ii) an additional 50 samples have been added to the cluster *without* recalculating the mean and variance. This results in significant improvement in computation speed.

2. An animation of the Weibull distribution parameter estimation with Maximum Likelihood is shown https://youtu.be/Ray99d6H35where

5.4 Cluster Refinement

Clusters resulting from actual geospatial features with less or no significance such as bumps on the ground, shingles, grass, etc., or from noise during the acquisition process are removed by merging as described next.

First, an adjacency graph is built based on the result of the tensor clustering. In addition to keeping track of neighbouring clusters, the graph also stores point-level information i.e., how many and which points are neighbouring.

The cluster refinement is an iterative process which merges every cluster C^- containing a small number of points i.e., ≤ 5 to a cluster $C^{'}$ if and only if the following conditions hold true,

- the cluster $C^{'}$ is a neighbour of C^- in the adjacency graph
- the cluster $C^{'}$ has the highest number of neighbouring points from all neighbours of C^-
- the cluster $C^{'}$ contains at least twice the points of C^-

If the above conditions are true, then the cluster C^- is absorbed by $C^{'}$ and all the information in the clusters and the adjacency map are updated to reflect the change.

As mentioned above, this is an iterative process which is repeated until no merge has occurred. During our experiments the maximum number of iterations has been six with an average number of four iterations, and the maximum reduction in the number of clusters of around 90 percent with an average reduction in the number of clusters of around 75 percent. Further to this, an additional average reduction of 35 percent occurs after the removal of vertical surfaces i.e., surfaces for which the angle between their orientation and the nadir direction is greater than 45°.

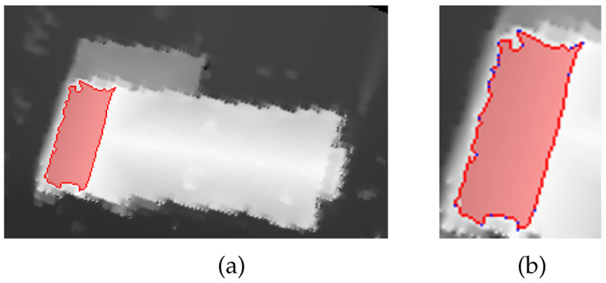


Fig. 10. An example of the boundary extraction process being applied to a cluster containing points representing part of a roof. The points are shown in red and the dense boundary points extracted are shown in bright red in (a). (b) shows a close up of the simplified boundary points in blue.

We tested our clustering method on several pointcloud datasets of different resolutions which exhibit significant variability in geospatial characteristics e.g., ground surface inclination, building density, etc.

6 Boundary Extraction and Refinement

The result of the clustering is the formation of a set of contiguous regions. Perhaps one of the most difficult tasks in urban reconstruction from LiDAR data is extracting boundaries corresponding to depth discontinuities. Due to the zig-zag scanning fashion of the LiDAR scanner, depth discontinuities appear jagged in the captured data. Several approaches have already been proposed [1], [3], [9] for refining the boundaries however they all treat each cluster of points individually. In this work we propose a different technique for refining boundaries which unlike the existing work reformulates this boundary refinement as a global optimization problem based on the following two observations; (i) all cluster boundaries [not on the captured image boundary] between adjacent clusters are complementary to each other and (ii) object boundaries must align to dominant directions. The boundary points of each cluster are extracted, adjusted, refined and finally extruded to produce 3D lightweight polygonal models. The following subsections further describe the boundary extraction process, the subsequent refinement and optimization steps, and the final extrusion to 3D models.

6.1 Boundary Extraction

The boundaries of each cluster $C_i, 0 \leq i \leq M$ where M is the number of clusters produced by the tensor clustering, are extracted as follows.

A 2D map is created for C_i marking all the points contained in the cluster. The cluster's boundary points B^{C_i} are then retrieved from the map using the algorithm of Suzuki et al. in [18]. The result is the dense set of the 2D image locations corresponding to the 3D exterior boundary points surrounding all points within the cluster. These are further reduced to a minimal set of the 2D image locations of the 3D boundary points after applying the iterative-end-point fit algorithm of Douglas-Peucker in [19] as shown in Fig. 8d. During the simplification only *colinear* points are removed from the set i.e., the threshold is very small and set to $\tau = 0.01$. This process is repeated until boundaries for all clusters have been extracted and simplified.

An example of the boundary extraction process is shown in Fig. 10a. The points contained in a cluster representing part of a roof are shown with red in the normalized depth map. The dense boundary points are shown with a bright red color, which upon further reduction result in the simplified boundary points shown as blue pixels in the close-up in Fig. 10b.

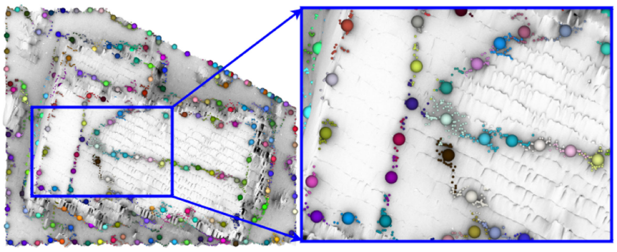


Fig. 11. Each large color-coded sphere represents the handle of a neighbourhood and the small spheres with the same color [visible in the closeup] represent the actual boundary points contained within it.

6.2 Snapping and Adjustment

Two adjacent clusters have complementary boundary components i.e., some of the boundary points in one cluster will correspond and complement some of the neighbouring cluster's boundary points. Refining the boundary points separately almost always leads to undesirable effects such as misalignment between the initially complementary boundaries; this problem often appears as holes in the resulting 3D models. In order to ensure that there is no misalignment between the final model's neighbouring boundaries, we group neighbouring boundary points together prior to the refinement. This helps in avoiding holes and making the generated geometry water-tight.

Boundary points B_i, B_j from any cluster which are within a user-defined radius i.e., $\tau_r = 2px$ from each other are grouped into a neighbourhood $\aleph$ such that $\forall B_i, B_j \in \aleph \Rightarrow ||B_i - B_j|| \leq \tau_r$. For all subsequent processing each boundary point is represented in terms of its *handle*:

- the neighbourhood's location in the image $(\bar{u_{\aleph_x}}, \bar{v_{\aleph_y}})$ computed as the average of the image locations [2D] of all points contained within the neighbourhood.
- the neighbourhood's X, Y components of its 3D position computed as the average X, Y components of all points within the neighbourhood.

The 3D position for each point B_i contained in a neighbourhood group is then snapped to the group's handle's X,Y components but retaining the Z component of the original 3D point $(\bar{X_{\aleph}}, \bar{Y_{\aleph_y}}, Z_{B_i})$. This allows the representation of boundary points with the same X,Y but different vertical values in the same neighbourhood, e.g., two points on a vertical wall may have the same X, Y components but different Z components between the ground and roof points.

After the snapping process, all boundary points are expressed in terms of a set of handles. Fig. 11 shows the neighbourhood handles resulting from the sparse boundary points in Fig. 8d. Each large color-coded sphere represents the handle of a neighbourhood and the small spheres with the same color [visible in the closeup] represent the actual boundary points contained within the neighbourhood.

6.3 Refinement and Global Optimization

The position of each neighbourhood's handle is further adjusted through a process which iterates between a refinement step and a global optimization step. The following sections describe these steps in detail.

6.3.1 Detection of Dominant Orientations and Refinement

Refinement involves (a) the detection of per-scene dominant orientations and (b) the handle position refinement for each neighborhood group based on the scene's detected dominant orientations

The dominant orientations present in the *entire scene* are extracted. First, for each group, a set of vectors representing boundary orientations is computed by subtracting each pair of consecutive boundary points B^{XY}_{prev} and B^{XY}_{next}. The vectors are kept unnormalized in order to account for different weights i.e., vectors resulting from the subtraction of pairs of boundary points whose distance is high will have a higher weight. The vectors are 2D and only contain the X,Y components of each 3D boundary point. Boundary points close to the image boundaries are excluded so as not to introduce spurious orientations corresponding to the image boundaries.

Next, we perform Principal Component Analysis (PCA) on the entire set of vectors derived from all the neighborhood boundary groups in the scene and reduce it into a smaller representative set which accounts for most of the variance in the original variables. In our experiments, the maximum number of principal components (PCs) is set to 5 although it has been observed that metropolitan areas following the Manhattan-style design contain primarily clusters corresponding to axes-aligned building structures and usually result in only 2-3 PCs.

Finally, a Gaussian Mixture Model $G^{2D}(\vec{v})$ is applied on the set of PCs. In order to avoid having to fix the number N_g of Gaussian distributions $g_i(\vec{v}), 1 \leq i \leq N_g$ contained in the $G^{2D}(\vec{v})$ we opt for using minimum-description length (MDL) [20] to adaptively determine this number. The result is the minimal set of the dominant orientations of all the neighbourhood boundary groups in the scene represented by the normalized means of each $g_i(\vec{v})$ contained in the $G^{2D}(\vec{v})$.

6.3.2 Refinement based on Dominant Orientations

The dominant orientations are used to refine the position of each neighbourhood's handle. First, for each vector $\vec{v}^B = B^{XY}_{next} - B^{XY}_{current}$ resulting from subtracting two consecutive boundary points $B^{XY}_{current}$ and B^{XY}_{next}, we determine the g_{max} contained in the GMM $G^{2D}(\vec{v})$ in which $\vec{v}^B$ is maximal,

$$g^{max}(\vec{v}^B) = argmax(g_i(\vec{v}^B))), \tag{15}$$

where $1 \leq i \leq N_g$ and $g_i \in G^{2D}$.

Once the g_{max} is determined, the vector $\vec{v}^B$ is projected onto the vector representing the means $\mu_{g_{max}}$. This results in a refined position for one of the boundary points, i.e., B^{XY}_{next}. This process is repeated until the X and Y components of all the boundary points have been refined.

6.4 Global Optimization Formulation

The refinement based on the dominant orientations considerably improves i.e., linearizes, the appearance of the boundaries. A final refinement using global optimization is performed to ensure that the displacement introduced by the previous steps also matches the observed data. For this global optimization we define an error function E_f which when maximized results in a mapping f for which the boundary points are in their optimal positions. A boundary location B is optimal when all the points along the boundary lines connecting the previous point B_{prev}, B, and the next point B_{next} have maximal Gabor response.

The error function E_f is defined as a Gibbs potential,

$$E(f) = D(\bar{B}^{XY}_{current}) \times e^{-\frac{d}{\rho}}, \tag{16}$$

where $d = ||B^{XY}_{current} - \bar{B}^{XY}_{current}||$ is the distance between the optimized position $\bar{B}^{XY}_{current}$ and the initial position $B^{XY}_{current}$, ρ is the optimization search radius, and $D(\bar{B}^{XY}_{current})$ is given by,

$$D(\bar{B}^{XY}_{current}) = \sum_{P=B^{XY}_{prev}}^{R(B^{XY}_{prev},\bar{B}^{XY}_{current})} r_P + \sum_{P=\bar{B}^{XY}_{current}}^{R(\bar{B}^{XY}_{current},B^{XY}_{next})} r_P, \tag{17}$$

where $R(.,.)$ is a function that rasterizes the line between the two input points using Bresenham's algorithm and, r_P is the Gabor response at location P. Intuitively, Eq. (17) gives a measure of how appropriate the optimized boundary position is by evaluating the Gabor responses along the lines beginning and ending at that position.

Fig. 8e shows the boundary points after snapping and adjustment as red. The optimized points are shown as green overlaid on the Gabor response image.

6.5 Reconstruction

After the adjustment, refinement, and optimization, the boundaries are extruded to the ground [ground elevation is set to the minimum height in the scene], to form 3D models. The refined boundaries of each cluster can have arbitrary [but not self-intersecting] shapes. In order to handle these complex shapes we employ a variant of the line-sweep triangulation algorithm which can handle complex and concave geometry robustly. Normal information about the boundary points is taken into account during the triangulation in order to ensure smooth transitions between the planar surfaces of the reconstructed model. Furthermore, texture coordinates are computed for easier assignment of texture maps.

7 Experimental Results & Evaluation

We have extensively tested our methods on real data of city-scale size. Several point clouds captured with airborne-LiDAR scanners have been used, those of which exhibiting different characteristics are shown in this paper. Namely, Baltimore, MD which covers an area of 16 km^2, Denver, CO which covers an area of 14 km^2 and Oakland, ON which covers an area of 17 km^2. Within each dataset there is significant variability in terms of the geospatial object density, sampling density, and area type i.e., rural, suburban, urban.

Figs. 12 and 13 clearly show that our techniques scale up to large datasets.[3] The city of Baltimore, MD consists of 36 components (each represented as an XYZ map of size $1024x1024$) generated using the structuring algorithm in [1].

3. An animation of the reconstructed models is shown https://youtu.be/puo5prNtbQUhere

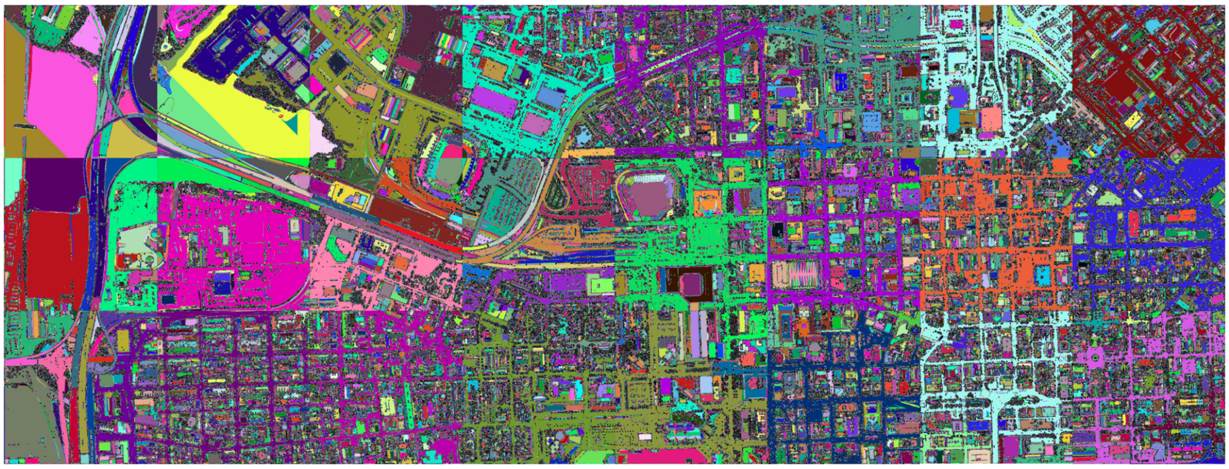


Fig. 12. The final clusters after tensor clustering. Baltimore, MD has a size of $16\ \mathrm{km}^2$, the most complex part of which is shown here.

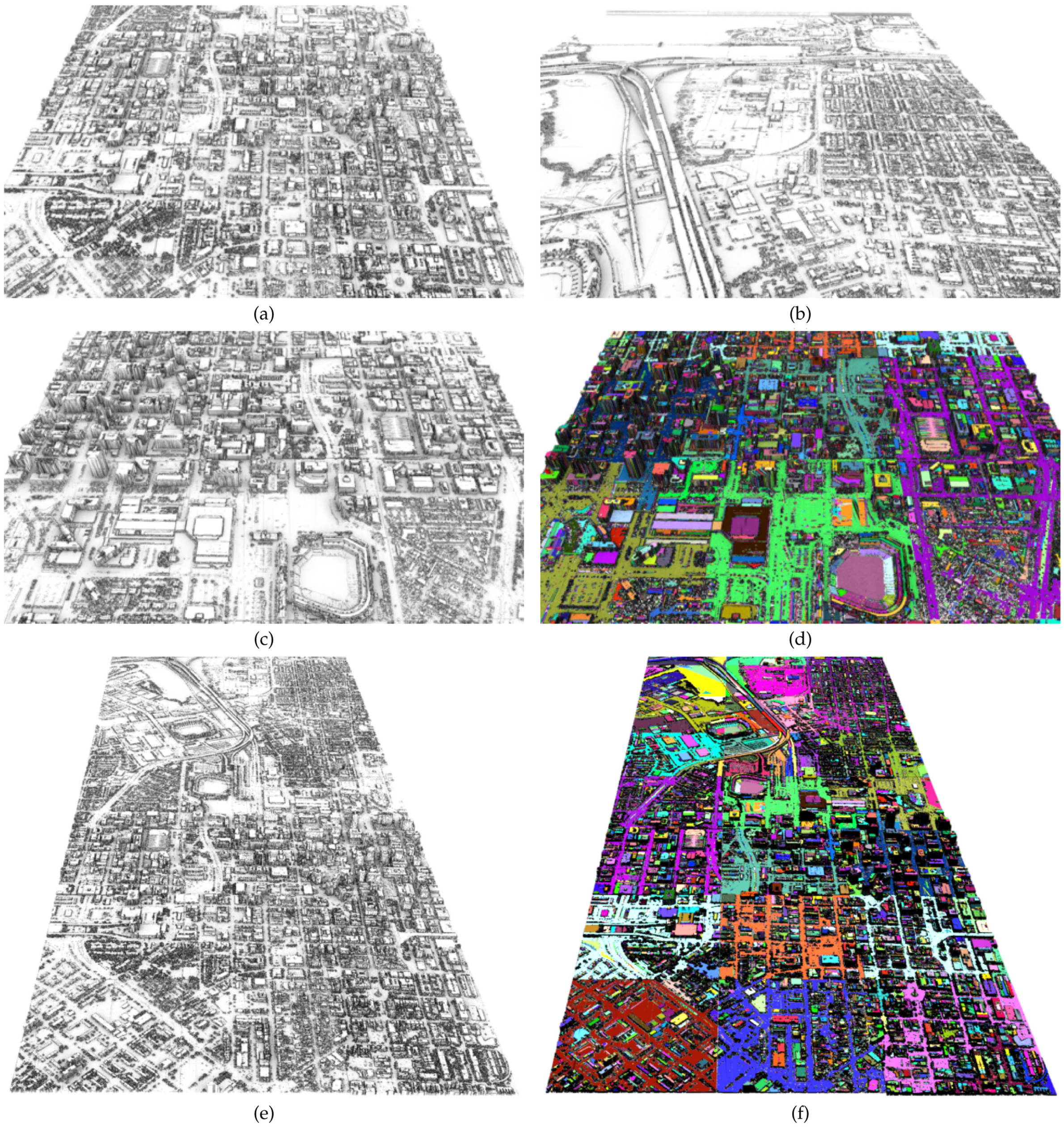


Fig. 13. (a), (b), (c), (e): Renderings of the reconstructed models for Baltimore, MD. (d), (f): Rendering of the reconstructed models in (c) and (e) respectively textured with the clustering results. Baltimore, MD has a size of $16\ \mathrm{km}^2$.

TABLE 1
Comparison Table between Our Approach and State-of-the-Art in [21] and [1]

Dataset	Area (km^2)	# Comp.	Comp. Resolution	Processing time (hr)			Geometric Accuracy m^2		
				Our approach	[21]	[1]	Our approach	[21]	[1]
Baltimore	16	36	1024×1024	22.75	9	1.3	6.96×10^{-03}	9.72×10^{-01}	0.48×10^{-01}
Denver	14	19	991×991	16.2	3.7	1	4.826×10^{-03}	4.915×10^{-01}	0.311×10^{-01}

Our approach requires more processing time for the same number of components primarily due to the recalculation of the Weibull distribution. However it is completely automatic, does not require any parameter value to be specified by the user and it produces superior results in terms of geometric accuracy.

Each component is processed independently and in-parallel using a Microsoft Azure Virtual Machine (Standard DS15 v2) with 20 cores and 140 GB memory. The processing time for the clustering is determined by the time required to process the slowest component, and for Baltimore that was 22.75 hours. The city of Denver, CO consists of 20 components (each represented as an XYZ map of size $991x991$) and the processing time was 16.2 hours. Table 1 summarizes these information and provides a comparison with the state-of-the-art large-scale modeling techniques in [21] and [1]. As can be seen, our method does require more processing time. This is primarily due to the computations involved in recalculating the Weibull distribution (as explained in detail in the Appendix, available in the online supplemental material). However, the clustering results obtained using our approach outperform the other techniques in terms of geometric accuracy, while not requirng any parameter tweaking whatsoever. The geometric accuracy is the equivalent to the surface fitting error and is defined as the RMS distance of the fitted surface points from their original position.

Fig. 15 shows all intermediate results of our technique for four test sites exhibiting variability in terms of building/trees/cars/roads size and density. The depth map is shown in the first column and the sum of the gabor responses for all ie. 16 orientations is shown in the second column. The third column shows the height and normal variation map. The last two columns show the resulting clusters after the application of tensor clustering (fifth column) and the refined clusters generated by the global boundary refinement (sixth column) after merging insignificant clusters i.e., <4 points and, removing clusters corresponding to vertical surfaces $<45^{\circ}$.

An additional comparison was performed with the approach presented in [9] and later extended by [3] which employs dual contouring on 2.5D pointcloud data. It should be noted that their approach requires a total of 12 user-defined thresholds. After extensive experimentation we found the optimal values for these thresholds and used them to generate the result in Fig. 14. The original captured data is assumed to be the ground truth. Each reconstruction is compared against the ground truth by computing the Hausdorff distance [22] for a fixed number of sample points for both (i.e., $300K$). A visualization is shown in Fig. 14a where the distances are shown with RGB colors. The RMS error, the Hausdorff distances' range, and the mean error for our reconstruction are 0.174456, [0,0.559475], 0.103138, respectively. For dual contouring these values are 0.212459, [0,0.562199], and 0.141765. As it is evident, our method results in improved reconstructions. Much of the errors occurring [note that the maximum error is 0.559475] are due to noise in original data which when compared with the planar surfaces may result in a high error.

Fig. 14 shows a visual comparison between our reconstruction (left), the original captured data (middle), and the dual contouring reconstruction (right). While, from such visual qualitative assessment, it may seem as though the results are quite similar, more careful visual inspection shows that for certain thin structures dual contouring seems to fail by creating vertical "ridges" in the reconstruction. Scaling to larger areas seems to be another problem with the dual contouring implementation. Despite the fact that we used the provided source code (without modifications other than experimenting to determine the optimal values of the 12 thresholds required), when running it on the same machine as the one we used for our method, the dual contouring program was unable to generate a 3D model for larger areas than this.

Our experiments have shown that the approach in [3] performs best for small building-scale reconstructions i.e., the input pointcloud represents a single building (or sequence of attached buildings) and is cropped at the buildings boundaries in a preprocessing step before being processed. In cases of multiple disjoint buildings (such as the own in Fig. 14) dual contouring produces building boundaries which may be visually pleasing but significantly deviate from the original data; hence the high error at the roofs/ver-

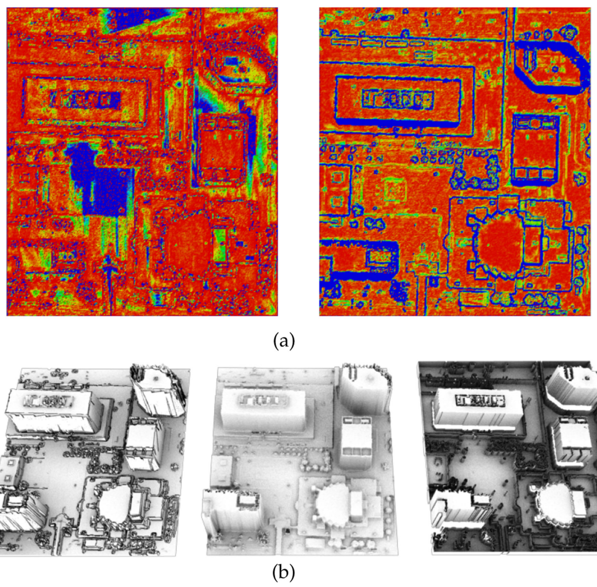


Fig. 14. (a) Visualization of the Hausdorff Distance between the original captured data and the result of our approach (left), and between the original captured data and the result of the approach in [3]. (b) 3D visualization of the reconstructions. For dual contouring, optimal values were determined via experimentation for the 12 required user-defined thresholds. (left) The result of our approach. (middle) Triangulation of the original captured data. (right) The result of dual contouring.

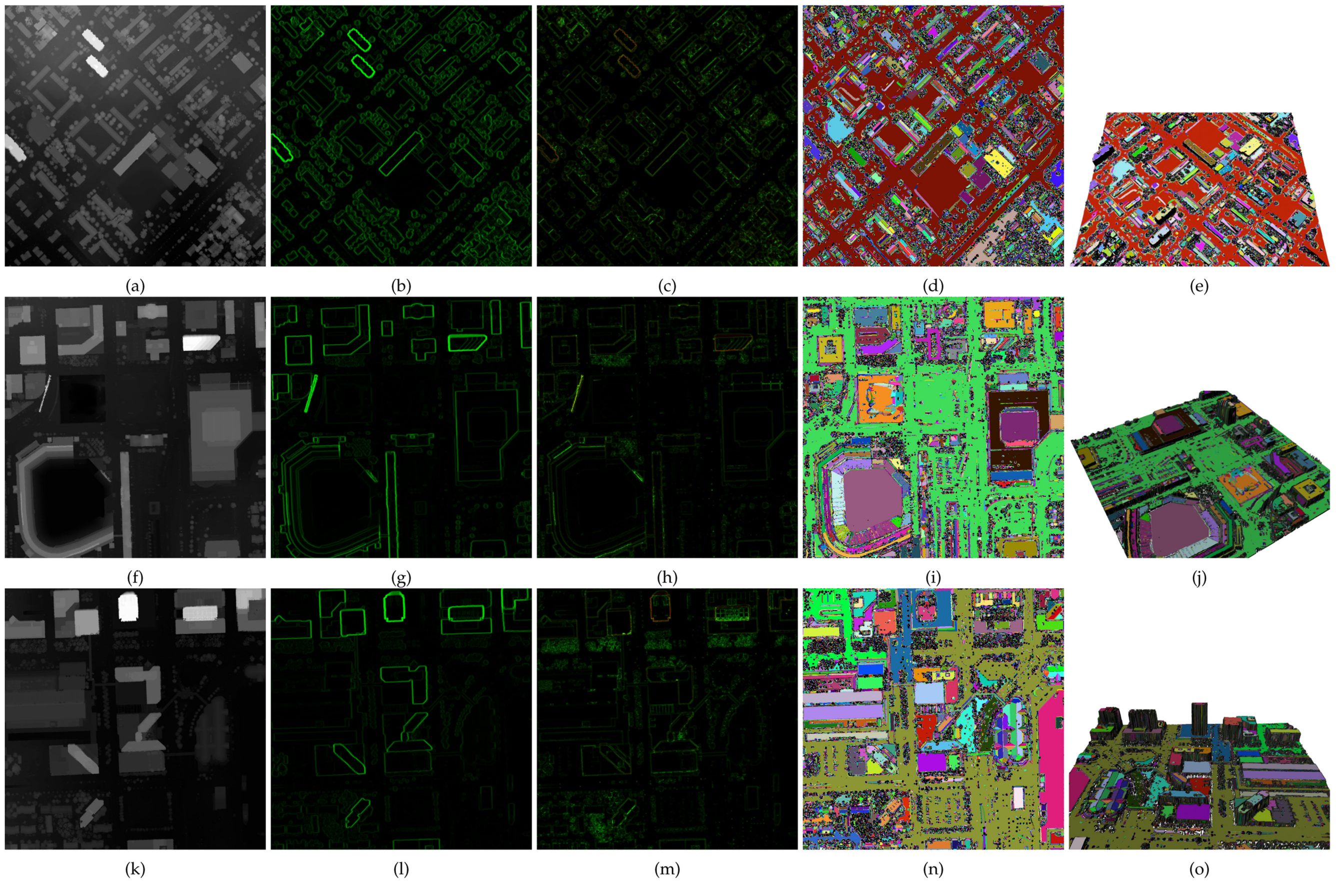


Fig. 15. Example results for different sites demonstrating tensor clustering and global boundary refinement. First column: Depth map. Second column: Sum of Gabor responses. Third column: Height and normal variation map. Fourth column: Color coded patches resulting from tensor clustering. Fifth column: The reconstructed models textured with the color coded patches. Tensor clustering has no user-defined thresholds.

tical walls. This can be attributed primarily to the quadtree simplification. The proposed approach makes no assumptions about the data and therefore all clusters are processed the same regardless of the type of object they represent e.g., building, tree, car, boats, etc. Furthermore, [3] use triangulation based on the curvature to model the clusters which, at the cost of increasing the geometry, produce models which are closer to the original data even in the presence of noise i.e., the noise is carried over the the reconstructed models. In the proposed approach, each cluster is modeled as a linear surface which drastically reduces the geometry but may lead to higher deviations from the original data; hence the higher error at the non-flat ground area in Fig. 14.

8 Conclusion

We have presented a new method for automatic 3D reconstruction of large scale urban areas from raw lidar (point cloud) data. Our most significant contribution, the elegant second-order symmetric tensor representation for encoding all metric information about points, does not require any user defined parameter for the entire region construction process, making our method automatic, effective and distinctly different from previous methods. All earlier methods are at best semi-automatic typically requiring the user to carefully specify a plethora of parameter values which are input data dependent and needed to produce usable reconstructions, making the process human effort intensive, difficult and inefficient. In contrast, we show that our solution works for any of the highly varied data sets we used to test and evaluate our reconstruction process. Our process includes a number of innovative techniques, a robust agglomerative tensor clustering technique for region finding, adaptive computation of per-point and per-cluster statistical parameters for the Weibull probability distribution function (pdf), whose parameters are dynamically updated as new points get added to the cluster, and a more accurate multi-stage region boundary extraction technique which reformulates it as a global optimization problem. We have tested and evaluated our method extensively on large scale urban areas from the United States with varying characteristics. To the best of our knowledge no existing method can generate this quality of reconstructions automatically for such large scale data. We plan to extend this work by using colour image data along with 3D lidar data, possibly using deep learning techniques to do the classification prior to 3D reconstruction. We will also investigate new methods for automating texture mapping to produce realistic 3D digital worlds.

Acknowledgments

This research is based upon work supported by the Natural Sciences and Engineering Research Council of Canada Grants DG-N01670 (Discovery Grant) and DND-N01885 (Collaborative Research and Development with the Department of National Defence Grant). The author would also like to thank Microsoft Research for making processing large data sets possible through its Azure for Research Award No. 0518306.

References

[1] C. Poullis, "A framework for automatic modeling from point cloud data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2563–2575, Nov. 2013.

[2] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.

[3] Q.-Y. Zhou and U. Neumann, "2.5 D building modeling by discovering global regularities," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 326–333.

[4] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. V. Gool, and W. Purgathofer, "A survey of urban reconstruction," *Comput. Graph. Forum*, vol. 32, no. 6, pp. 146–177, 2013.

[5] C. Poullis and S. You, "Photorealistic large-scale urban city model reconstruction," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 4, pp. 654–669, Jul./Aug. 2009.

[6] H. Lin, J. Gao, Y. Zhou, G. Lu, M. Ye, C. Zhang, L. Liu, and R. Yang, "Semantic decomposition and reconstruction of residential scenes from lidar data," *ACM Trans. Graph.*, vol. 32, no. 4, 2013, Art. no. 66.

[7] F. Lafarge and P. Alliez, "Surface reconstruction through point set structuring," *Comput. Graph. Forum*, vol. 32, pp. 225–234, 2013.

[8] N. J. Mitra, M. Pauly, M. Wand, and D. Ceylan, "Symmetry in 3D geometry: Extraction and applications," *Comput. Graph. Forum*, vol. 32, no. 6, pp. 1–23, 2013.

[9] Q.-Y. Zhou and U. Neumann, "2.5 D dual contouring: A robust approach to creating building models from aerial lidar point clouds," in *Proc. Int. Conf. Comput. Vis.*, 2010, pp. 115–128.

[10] J. Xiao and Y. Furukawa, "Reconstructing the worlds museums," *Int. J. Comput. Vis.*, vol. 110, no. 3, pp. 243–258, 2014.

[11] C. Poullis, "Tensor-cuts: A simultaneous multi-type feature extractor and classifier and its application to road extraction from satellite images," *ISPRS J. Photogrammetry Remote Sens.*, vol. 95, pp. 93–108, 2014.

[12] G. Medioni, M.-S. Lee, and C.-K. Tang, *A Computational Framework for Segmentation and Grouping*. Amsterdam, The Netherlands: Elsevier, 2000.

[13] W. Förstner and B. Moonen, "A metric for covariance matrices," in *Geodesy-The Challenge of the 3rd Millennium*. Berlin, Germany: Springer, 2003, pp. 299–309.

[14] A. Cherian, S. Sra, A. Banerjee, and N. Papanikolopoulos, "Efficient similarity search for covariance matrices via the jensen-bregman logdet divergence," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2399–2406.

[15] J. Malcolm, Y. Rathi, and A. Tannenbaum, "A graph cut approach to image segmentation in tensor space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.

[16] M. Herdin, N. Czink, H. Özcelik, and E. Bonek, "Correlation matrix distance, a meaningful measure for evaluation of non-stationary mimo channels," in *Proc IEEE 61st Veh. Technol. Conf.*, 2005, vol. 1, pp. 136–140.

[17] G. Burghouts, A. Smeulders, and J.-M. Geusebroek, "The distribution family of similarity distances," in *Proc. Advances Neural Inf. Process. Syst.*, 2008, pp. 201–208.

[18] S. Suzuki, et al., "Topological structural analysis of digitized binary images by border following," *Comput. Vis. Graph. Image Process.*, vol. 30, no. 1, pp. 32–46, 1985.

[19] 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," *Cartographica: The Int. J. Geographic Inf. Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[20] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[21] C. Poullis and S. You, "Automatic reconstruction of cities from remote sensor data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 2775–2782.

[22] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," *Comput. Graph. Forum*, vol. 17, no. 2, pp. 167–174, 1998.

Charalambos Poullis received the BSc degree in computing and information systems with first Class Honors from the University of Manchester, United Kingdom, in 2001, and the MSc degree in computing science with specialisation in multimedia and creative technologies, and the PhD degree in computer science from the University of Southern California (USC), Los Angeles, in 2003 and 2008, respectively. In 2010, after spending a year at the Department of Computer Science, University of Cyprus as a Visiting Lecturer, he joined the Department of Multimedia and Graphic Arts, Cyprus University of Technology as a lecturer, and in 2014 became an assistant professor. Since August 2015, he has been an associate professor with the Department of Computer Science and Software Engineering at the Gina Cody School of Engineering and Computer Science, Concordia University where he also serves as the Director of the Immersive and Creative Technologies (ICT) lab. His current research interests include the intersection of computer vision and computer graphics. More specifically, he is involved in fundamental and applied research covering the following areas: acquisition technologies & 3D reconstruction, photo-realistic rendering, feature extraction & classification, virtual & augmented reality. More specifically, he is involved in fundamental and applied research covering the following areas: acquisition technologies & 3D reconstruction, photo-realistic rendering, feature extraction & classification, virtual & augmented reality. Charalambos is a member of the Association for Computing Machinery (ACM); Institute of Electrical and Electronics Engineers (IEEE) Computer Society; Marie Curie Alumni Association (MCAA); ACM Cyprus Chapter, where he also served in the management committee between 2010-2015; and British Machine Vision Association (BMVA). Charalambos has been serving as a regular reviewer in numerous premier conferences and journals since 2003. He is a member of the IEEE.