2.5D Dual Contouring: A Robust Approach to Creating Building Models from Aerial LiDAR Point Clouds

Qian-Yi Zhou and Ulrich Neumann*

University of Southern California

Abstract. We present a robust approach to creating 2.5D building models from aerial LiDAR point clouds. The method is guaranteed to produce crack-free models composed of complex roofs and vertical walls connecting them. By extending classic dual contouring into a 2.5D method, we achieve a simultaneous optimization over the three dimensional surfaces and the two dimensional boundaries of roof layers. Thus, our method can generate building models with arbitrarily shaped roofs while keeping the verticality of connecting walls. An adaptive grid is introduced to simplify model geometry in an accurate manner. Sharp features are detected and preserved by a novel and efficient algorithm.

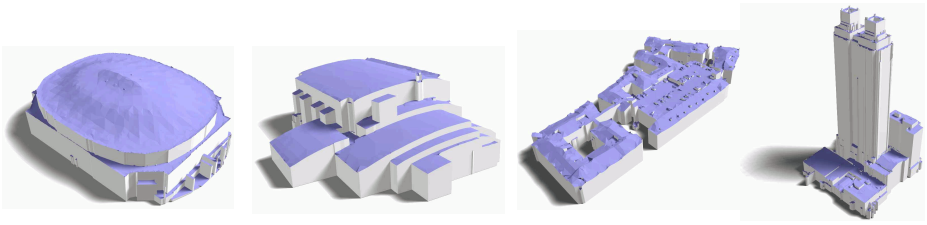

Fig. 1. Various kinds of building models are created using 2.5D dual contouring.

1 Introduction

Three dimensional building models are very useful in various applications such as urban planning, virtual city tourism, surveillance, and computer games. The advance of acquisition techniques has made aerial LiDAR (light detection and ranging) data a powerful 3D representation of urban areas, while recent research work (*e.g.*,[10, 15]) has introduced a successful pipeline to extract individual building point clouds from city-scale LiDAR data.

* The authors would like to thank Airborne 1 Corp. for providing data sets. The authors acknowledge Mark Pritt of Lockheed Martin for his support. The authors thank Tao Ju, Suya You, and anonymous reviewers for their valuable comments.

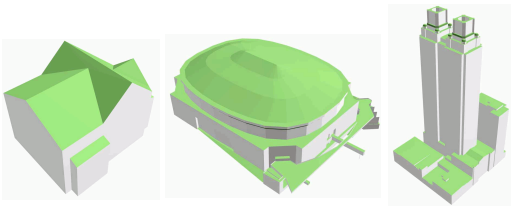

Fig. 2. Manually created models [3] show the 2.5D nature of building structures.

The aerial LiDAR point clouds are 2.5D data, *i.e.*, the LiDAR sensor captures the details of roof surfaces, but collects few points on building walls connecting roof boundaries. In addition, manually created building models (Figure 2) also show a 2.5D characteristic. Nearly all of them consist of complex roofs (green faces) connected by vertical walls (white faces). Thus, we desire a 2.5D modeling method with the following properties:

- **Accuracy:** The method should produce simple polygonal models fitting the input point clouds in a precise manner.
- **Robustness:** Regardless of the diversity and complexity of building roof shapes, the method should always generate crack-free models, even with the existence of undesired elements such as residual sensor noise and small roof features.
- **2.5D characteristic:** The method should create 2.5D polygonal models composed of detailed roofs and vertical walls connecting roof layers.

Most of the previous research work is based on the detection of some pre-defined roof patterns, such as planar shapes [8, 10, 11, 15] or a small set of user-given primitives [5, 12–14]. These methods work well for buildings composed of pre-defined shapes, but lose accuracy and robustness when dealing with arbitrary roof shapes such as those shown in Figure 1. Another way to attack this problem is with traditional data-driven approaches. Polygonal models are first generated directly from input data using rasterization or delaunay triangulation, then simplified with general mesh simplification algorithms. The latter step significantly reduces triangle number while preserving a low fitting error. However, since the general simplification algorithms are usually 'blind' to the 2.5D nature of the problem, they can hardly produce models satisfying our 2.5D requirement.

We propose a novel, data-driven approach to solve this problem, named *2.5D dual contouring*. Like the classic dual contouring [4], we use an adaptive grid as the supporting data structure, and reconstruct geometry in each grid node by minimizing the quadratic error functions known as QEFs. Model simplification is easily achieved by merging grid nodes and combining QEFs.

In order to represent the detailed roof surfaces, our approach works in a 3D space. However, unlike the classic 3D dual contouring, we use a 2D grid as our supporting data structure. We generate a *hyper-point* in each grid cell, which contains a set of 3D points having the same x-y coordinates, but different z

values. They can be regarded as a set of points intersected by a vertical line and multiple roof layers. Hence, the consistency between boundary footprints of different roof layers is guaranteed, and vertical walls are produced by connecting neighboring hyper-points together.

Given that our method is built on some of previous work, we explicitly state our original contributions as follows:

1. We propose a new robust method to create 2.5D building models from aerial point clouds. We demonstrate how to simplify geometry in a topology-safe manner and construct polygons within a 2.5D framework. Our results are guaranteed to be accurate watertight models, even for buildings with arbitrarily shaped roofs.
2. We propose an algorithm to detect sharp roof features by analyzing the QEF matrices generated in 2.5D dual contouring. The analysis result is then used to preserve such features in polygon triangulation.
3. Benefiting from a post-refinement step, our algorithm has the ability to produce building models aligning with principal directions, as defined in [14].

2 Related Work

We review the related work on two aspects: building reconstruction methods and volumetric modeling approaches.

2.1 Building Reconstruction from Aerial LiDAR

Many research efforts have addressed the complex problem of modeling cities from aerial LiDAR data. Recent work (*e.g.*, [8, 10, 11, 14, 15]) introduced an automatic pipeline with the following characteristics: trees and noises are removed via a classification algorithm, and a segmentation module splits the remaining points into individual building patches and ground points. The building patches are then turned into mesh models by a modeling algorithm.

In the last step, these methods first apply a plane fitting algorithm to extract planar building roofs, then employ different heuristics to guide the modeling process. For example, Matei *et al.*[8] regularize roof outlines by estimating building orientations. Poullis and You [10] create simple 3D models by simplifying boundaries of fitted planes. Verma *et al.*[11] employ a graph-based method to explore the topology relationships between planar roof pieces. Zhou and Neumann [14] learn a set of principal directions to align roof boundaries and this principal direction learning procedure is further extended to city-scale data sets in [15].

To alleviate the problem that only planar shapes can be handled well, primitive-based methods are developed to reconstruct complex building roofs. Lafarge *et al.*[5] propose a two-stages method to find the optimal combination of parametric models based on a RJMCMC sampler. You *et al.*[12] and Zhou and Neumann [14] show the usage of primitives with the help of user-interaction. Zebedin *et al.*[13] detect planes and surfaces of revolution. However, as mentioned previously, all of these methods are limited by the user-defined primitive libraries, thus lose accuracy and robustness when dealing with arbitrary roof shapes.

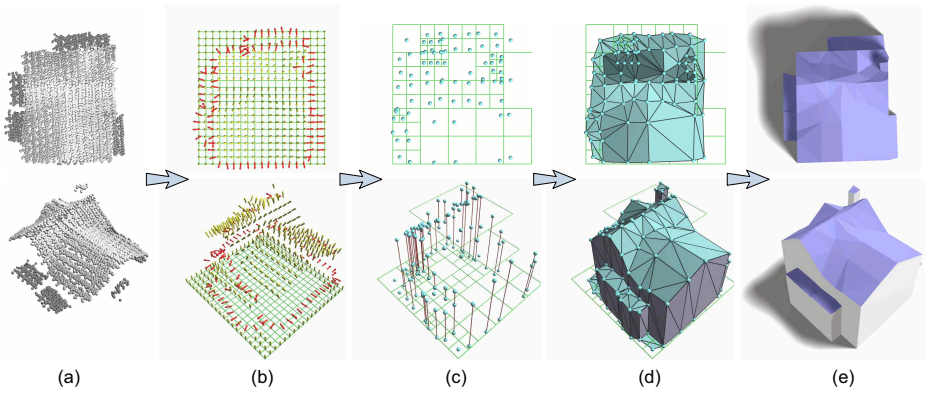

Fig. 3. Robust building modeling pipeline: (a) the input point cloud; (b) a 2D grid with surface Hermite data (gold arrows) and boundary Hermite data (red arrows) attached; (c) hyper-points (turquoise balls connected by red lines) generated by minimizing QEFs; (d) mesh model reconstructed via 2.5D dual contouring; and (e) final model with boundaries snapped to principal directions.

2.2 Volumetric Modeling Approaches

Volumetric methods [1, 4, 7] have proved to be a robust way of generating crack-free models: input points are first scan-converted into a regularized grid; then geometry and topology are created respectively. For example, the dual contouring method [4] creates one mesh vertex in each minimal grid node by optimizing a quadratic error function, and constructs polygons during a traversal over the adaptive grid. Based on this work, Fiocco *et al.*[2] develop a modeling method combining aerial and ground-based LiDAR.

Nevertheless, these volumetric approaches all work for regular 2D or 3D grids. None of them have the same 2.5D characteristic as our approach.

3 Pipeline Overview

Given a building point cloud as input, our modeling process executes four steps as illustrated in Figure 3:

1. **Scan conversion:** We embed the point cloud in a uniform 2D grid. *Surface Hermite data* samples (gold arrows) are generated at grid points and *boundary Hermite data* samples (red arrows) are estimated on grid edges connecting different roof layers (Figure 3(b)). This 2D grid is also regarded as the finest level of our supporting quadtree.
2. **Adaptive creation of geometry:** In each quadtree cell, we compute a *hyper-point* by minimizing a 2.5D QEF. Geometry simplification is achieved in an adaptive manner by collapsing subtrees and adding QEFs associated with leaf cells (Figure 3(c)).

3. **Polygon generation:** We reconstruct a watertight mesh model by connecting hyper-points with *surface polygons* (turquoise triangles) and *boundary polygons* (purple triangles), which form building roofs and vertical walls, respectively (Figure 3(d)).
4. **Principal direction snapping:** The roof boundaries are refined to follow the principal directions defined in [14] (Figure 3(e)).

4 Scan Conversion

The first step of our modeling algorithm converts the input point cloud into a volumetric form, by sampling Hermite data (in the form of point-normal pairs) over a 2D supporting grid. With elements being considered as their infinite extensions along the vertical direction, this 2D grid has a 3D volumetric connotation. *E.g.*, a grid cell represents an infinite three dimensional volume, while a grid point corresponds to a vertical line containing it.

4.1 Surface Hermite Data

Given a 2.5D point cloud as input, we first segment it into multiple roof layers using a local distance-based region growing algorithm[1], as shown in Figure 4(a). Ideally, each vertical line passing through a grid point intersects with one and only one roof layer. The intersection point is taken as a *surface Hermite data* sample, and estimated by averaging the heights and normals[2] of its k-nearest input points within the same roof layer, illustrated as points marked with blue or purple outlines (taking $k = 4$) in Figure 4(a).

The only difficulty in this process is to robustly detect the right roof layer crossing the vertical line. Intuitively, we say a roof layer L *covers* a grid point g *iff* each of g's four neighboring cells contains at least one input point p belonging to L or a higher cluster L'. *E.g.*, in Figure 4(a), point A is covered by no roof layers, and thus is assigned as ground; point B is only covered by and assigned to the dark-grey layer; covered by both the dark-grey layer and the light-grey layer, point C is assigned to the highest covering layer, *i.e.*, the light-grey layer.

4.2 Boundary Hermite Data

While surface Hermite data captures the surface geometry of building roofs, the shapes of roof boundaries are represented by the *boundary Hermite data*.

Considering a grid edge e connecting two grid points with surface Hermite data samples $\{s_0, s_1\}$ on different roof layers $s_0 \in L_0, s_1 \in L_1$,[3] the vertical

[1] The roof layers are always segmented in a local area, as global segmentation may erase local features such as those shown in Figure 8(c). Specifically, the segmentation for grid point g is applied to all the input points in g's four neighboring cells.

[2] Point normals are pre-computed using covariance analysis [14].

[3] To avoid ambiguity, roof layers are determined again by a local segmentation over $\{s_0, s_1\} \cup P$, where P is the input point set within e's two adjacent cells.

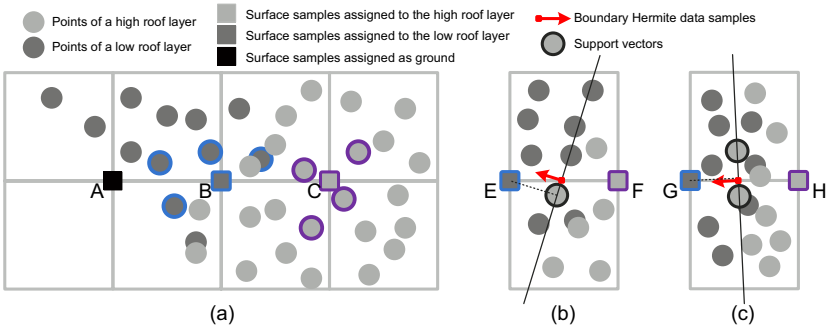

Fig. 4. Generating (a) surface Hermite data samples on grid points: the sample is assigned to the highest roof layer which *covers* the grid point; (b,c) boundary Hermite data samples on grid edges: we find the maximum margin line (thin black lines) to divide the lower surface Hermite data sample from the higher roof layer.

wall connecting L_0 and L_1 should split their projection images on the x-y plane. Inspired by the 2D support vector machine algorithm, we find the maximum-margin line l which separates L_0 and L_1 on the x-y plane, and estimate the boundary sample by intersecting line l and edge e.

In practice, with the existence of residual sensor noise, the projections of different roof layers may overlap on the x-y plane. Since our data is collected from a top view, we give more saliency to the higher roof layer L_1 (assuming $height(L_0) < height(L_1)$), and thus take the maximum-margin line l which separates $\{s_0\}$ and L_1 while maximizing $distance(s_0, l)$, shown as the thin black lines in Figure 4(b,c). Empirically, we find this method more robust than other methods including that using a maximum-soft-margin line dividing L_0 and L_1.

5 Adaptive Creation of Geometry

Given a quadtree cell c (not necessarily being a finest-level leaf cell), we denote the set of surface Hermite data samples on the grid points in c as S, and the set of boundary Hermite data samples on atomic grid edges in c as B. The roof layers in c are then determined by segmenting S into k clusters $S = S_1 \cup \cdots \cup S_k$. Intuitively, if an atomic grid edge in c has no boundary sample attached, it connects two surface samples of the same roof layer. Thus, we use an agglomerative clustering algorithm via repeatedly combining surface sample sets connected by edges without boundary samples.

Now our task is to generate k vertices for the k roof layers, denoted as a *hyper-point* $\chi = \{x_1, \ldots, x_k\}$. To maintain the consistency of roof layer boundaries, we require these k vertices to have the same projection onto the x-y plane, *i.e.*, they should have the same x-y coordinates, but different z values. Thus χ can be expressed as a $k+2$ dimensional vector $\chi = (x, y, z_1, \ldots, z_k)$. We let $x_0 = (x, y, 0)$ for convenience in following discussions.

5.1 2.5D QEF

The hyper-point χ is optimized by minimizing a 2.5D QEF defined as the linear combination of 2D boundary quadratic errors and 3D surface quadratic errors:

$$E(\chi) = \sum_{(p,n)\in B} (\omega n \cdot (x_0 - p))^2 + \sum_{i=1,\ldots,k} \sum_{(p,n)\in S_i} (n \cdot (x_i - p))^2 \tag{1}$$

where ω is a user-given weight balancing between boundary samples and surface samples. Empirically, a weight between $1 \sim 4$ satisfies most of our experiments.

Due to the horizontality of boundary sample normals, the third coordinates of p and x_0 do not affect the 2D error term. However, we choose to write all these variables uniformly in 3D, in order to express the energy function in a matrix product form:

$$E(\chi) = (A\chi - b)^T (A\chi - b) \tag{2}$$

where A is a matrix whose rows come from normals in $B, S_1, \ldots, S_k$, with those in B multiplied by ω. The x-y values of each normal are placed in the first two columns, while the z values of normals in S_i are placed in the $(i+2)$-th column. The remaining entries in A are padded with zeros. b is a vector composed of corresponding inner products $n \cdot p$ with the first $|B|$ entries multiplied by ω.

We employ the QR decomposition proposed in [4] to improve numerical stability during QEF optimization, *i.e.*,

$$(A \quad b) = Q \begin{pmatrix} \hat{A} & \hat{b} \\ 0 & r \\ 0 & 0 \\ \cdots & \cdots \end{pmatrix} \tag{3}$$

where Q is an orthogonal matrix and Equation 2 can be rewritten as:

$$E(\chi) = (A\chi - b)^T Q Q^T (A\chi - b) = (\hat{A}\chi - \hat{b})^T (\hat{A}\chi - \hat{b}) + r^2. \tag{4}$$

Thus, $E(\chi)$ is minimized by solving $\hat{A}\chi - \hat{b} = 0$. To handle the possible singularity of $\hat{A}$, we follow the solutions in previous methods [4, 6] by applying an SVD decomposition:

$$\hat{A} = U \Sigma V^T, \tag{5}$$

truncating small singular values in Σ with a magnitude of less than 0.1, and using the pseudo-inverse Σ^+ to compute the hyper-point χ as:

$$\chi = \bar{\chi} + V \Sigma^+ U^T (\hat{b} - \hat{A}\bar{\chi}) \tag{6}$$

where $\bar{\chi}$ is a guessed solution whose first two coordinates come from the centroid of B, and the $(i+2)$-th coordinate is the mean height of samples in S_i. If B is empty, the first two coordinates equal to those of the centroid of S.

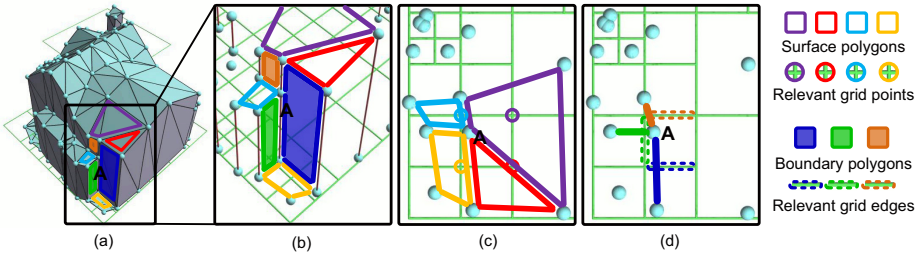

Fig. 5. (a,b) Creating surface polygons (colored hollow polygons) and boundary polygons (colored semitransparent polygons) around hyper-point A. Viewing from top, (c) surface polygons are generated at grid points, while (d) boundary polygons are produced for grid edges which exhibit a roof layer gap.

5.2 Quadtree Simplification with QEFs

Taking a quadtree with QEF matrices pre-computed for all the finest-level cells, we simplify the geometry by collapsing leaf cells into parent cells and combining QEFs in a bottom-up manner. A user-given tolerance δ controls the simplification level by denying sub-tree collapse when the residual is greater than δ.

Combining four regular 3D QEFs can be simply achieved by merging the rows of their upper triangular matrices to form a 16×4 matrix [4]. We follow this method to combine our 2.5D QEF matrices, yet with the consideration of association between matrix columns and roof layers: as roof layers in leaf cells merge into one roof layer in the parent cell, corresponding matrix columns are placed in the same column of the combined matrix. Specifically, we redo the roof layer segmentation in the parent cell before merging matrices. Assuming the i-th roof layer in a leaf cell belongs to the j-th roof layer in the parent cell, we put the $(i+2)$-th column of the leaf cell matrix into the $(j+2)$-th column of the combined matrix. 0-columns are used to pad the leaf cell matrices where no roof layers belong to certain roof layers in the parent cell.

Once again, the merged matrix is brought to the upper triangular form via a QR decomposition. Due to the orthogonality of involved transformation matrices, it represents the 2.5D QEF in the parent cell.

6 Polygon Generation

Given the simplified quadtree with hyper-points estimated in each leaf cell, our next task is to create polygons connecting these hyper-points into a mesh. In particular, we generate two kinds of polygons to satisfy our 2.5D characteristic.

1. **Surface polygons:** At each grid point p, we generate a surface polygon by connecting vertices in the hyper-points on the same roof layer as p in its neighboring cells.
2. **Boundary polygons:** At each minimal quadtree edge e, we create a boundary polygon connecting two hyper-point segments in the adjacent cells.

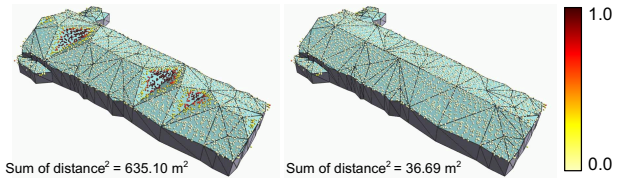

Fig. 6. Triangulation without (left) and with (right) our sharp feature preserving algorithm. The colors of input points represent the squared distances from the mesh.

Figure 5 shows an example of polygon generation around a hyper-point A. The surface polygons and boundary polygons are highlighted with colored outlines and colored semitransparent polygons respectively. To avoid cracks generated within a hyper-point, we make a boundary polygon sequentially pass through the vertices in hyper-point segment in height ascending or descending order. *E.g.*, the dark-blue boundary polygon in Figure 5 goes through all the three vertices in hyper-point A, from the top vertex to the bottom vertex.

Our method is guaranteed to produce crack-free models, which can be derived from the fact that except for the border edges created around the entire grid, the other mesh edges are contained by an even number of polygons. Proof is straightforward: a non-vertical mesh edge is either contained by two surface polygons, or by one surface polygon and one boundary polygon. As for the vertical mesh edges created within a hyper-point, we consider all the boundary polygons around this hyper-point (*e.g.*, the colored semitransparent polygons shown in Figure 5(a,b)). They go up and down though this hyper-point and finally return to the start vertex, forming up a closed edge loop. Thus, each vertical mesh edge in this hyper-point appears even times.

6.1 Sharp Feature Preserving Triangulation

By minimizing QEFs, 2.5D dual contouring has the ability to produce vertices lying on sharp features, which are a common pattern in building roofs. However, we find that a poor triangulation of surface polygons can spoil this advantage, as shown in Figure 6 left. To solve this problem, we propose an efficient sharp feature detection algorithm and preserve these features once detected.

In a grid cell c containing only one roof layer, we apply covariance analysis over the normals of all surface samples, *i.e.*, to get the eigenvalues of matrix:

$$C = \frac{1}{N} \sum_i n_i \cdot n_i^T, \tag{7}$$

and use Equation 3 and 5 to simplify it since c has no boundary samples:

$$C = \frac{1}{N} A^T A = \frac{1}{N} \hat{A}^T \hat{A} = \frac{1}{N} V \Sigma^T \Sigma V^T. \tag{8}$$

Thus, the diagonal of matrix $\frac{1}{N}\Sigma^T\Sigma$ gives the eigenvalues of C, while the columns of V are corresponding eigenvectors. As Pauly [9] suggests, the smallest

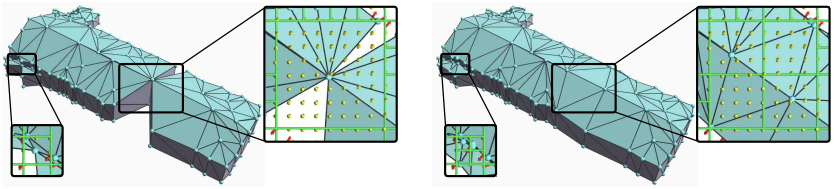

Fig. 7. Comparison between topology-unsafe simplification (left) and topology-safe simplification (right). Undesired features can be created by merging leaf cells in a topology-unsafe manner.

eigenvalue λ_0 and the middle eigenvalue λ_1 estimate the minimal and maximal curvatures, as the corresponding eigenvectors v_0, v_1 point to the curvature directions. Therefore, we find ridges and valleys by detecting vertices with small λ_0 and fairly large λ_1, and use v_0 as the feature direction. Since the involved matrices have all been computed in previous steps, the additional overhead of this algorithm is trivial.

Specifically, for each diagonal e of a surface quad, we calculate:

$$\sum_{p \in e \text{ and } \lambda_0(p) < \tau} \lambda_1(p) \cdot |v_0(p) \cdot e| \tag{9}$$

and choose the diagonal e^* which maximizes this value to split the quad into two triangles. Here τ is a user given threshold. Our experiments take $\tau = 0.01$.

7 Topology-Safe Simplification

So far the quadtree simplification is completely built on QEFs, and the topology of output models may change during this process. Undesired features can be generated as shown in Figure 7 left. To solve this problem, we insert an additional topology test right before sub-tree collapse happens; and reject collapse if the test reveals a danger of topology change. Regarding multiple roof layers as multiple materials, we use the topology test algorithm in [4], with an additional test (step 3) which prevents different roof layers in one leaf cell (top-left cell in Figure 8(a)) from merging into a same roof layer in the coarse cell (Figure 8(b)). This situation may cause removal of small vertical wall features (*e.g.*, Figure 8(c)).

1. Test whether each leaf cell creates a manifold; if not, stop.
2. Test whether the coarse cell creates a manifold; if not, stop.
3. Test whether any two roof layers in a same leaf cell belong to two different roof layers in the coarse cell; if not, stop.
4. Test whether the topology of the dual contour is preserved using following criteria; if not, stop; otherwise, collapse.
   (a) Test whether the roof layer on the middle point of each coarse edge agrees with the roof layer on at least one of the two edge endpoints.
   (b) Test whether the roof layer on the middle point of the coarse cell agrees with the roof layer on at least one of the four cell corners.

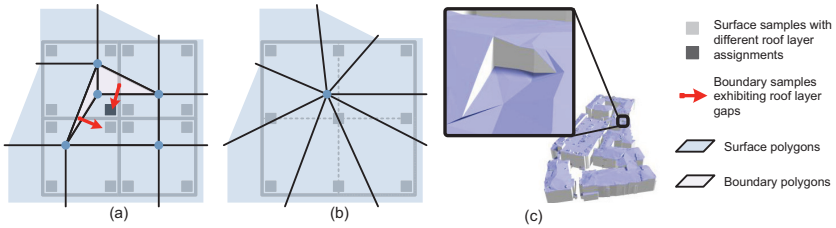

Fig. 8. An unsafe simplification case denied by the topology safety test step 3. Since the center grid point has different roof layer assignments in these leaf cells, two different layers in the top-left leaf cell (a) belong to the same roof layer in the coarse cell (b). Unsafe merging may erase wall features such as the one shown in (c).

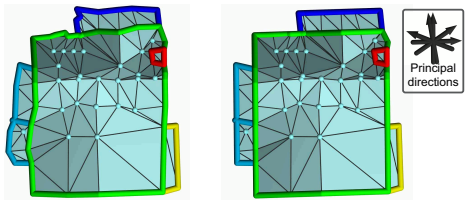

Fig. 9. Roof layer boundaries (thick colored lines) are snapped to principal directions.

8 Principal Direction Snapping

Our algorithm is completely data-driven, *i.e.*, no pre-assumptions about the roof shapes have been made. Thus our algorithm can handle complex roofs in a robust manner. On the other hand, in some cases, prior knowledge of the urban area is given and it is a desire to have building models concurring with such knowledge. In this section, we show a post-processing refinement to our results using the prior knowledge of principal directions, which are defined as the roof edge direction preference in a local urban area [14].

The idea is straightforward: once the boundaries of individual roof layers are extracted, we snap them to the principal directions as much as possible without exceeding a small error tolerance. In order to maintain the consistency between boundaries of different layers, the boundaries are handled one by one in height-descending order. *I.e.*, when a roof layer boundary has been processed, the x-y coordinates of the touched hyper-points are fixed, which are then considered as constraints during the subsequent processing of lower roof layers. Figure 9 shows clean and simple roof boundaries generated by the principal direction refinement.

9 Experiment Results

Figure 10 shows an urban area of Los Angeles reconstructed from 26M LiDAR points with 7 samples/sq.m. resolution. We employ the reconstruction pipeline

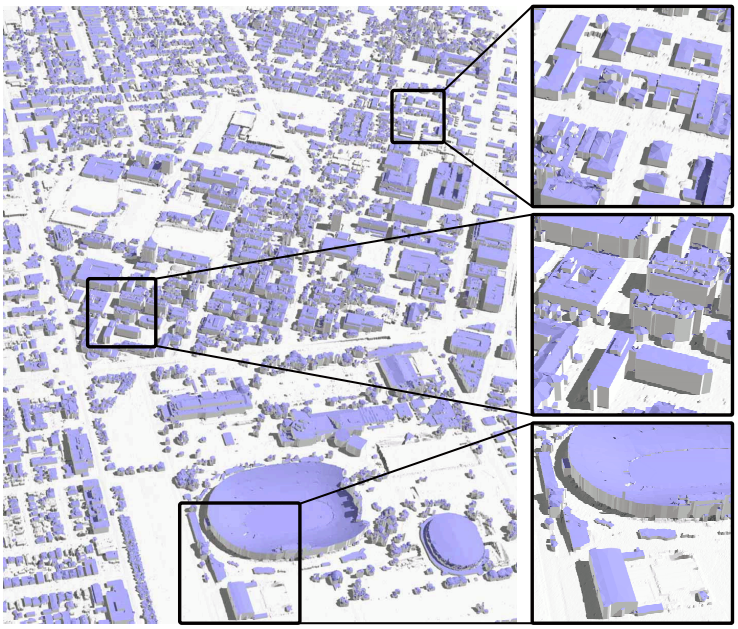

Fig. 10. Building reconstruction for a 2KM-by-2.5KM urban area of Los Angeles.

proposed in [15] to remove irrelevant parts such as noises, trees, vehicles and even ground. We then test our 2.5D dual contouring on point clouds of individual buildings to create 2.5D models with complex roofs. Our algorithm successfully creates 1879 building models consisting of 857K triangles within 6 minutes on a consumer-level laptop (Intel Core 2 1.8GHz CPU with 2GB memory).

To further demonstrate the ability of handling various kinds of building models, we test our method on a set of buildings from Atlanta, as illustrated in Figure 1. Figure 11 shows a comparison between our method and previous methods. In particular, we compare the average squared distance from input point sets to the generated models, and the ratio of points with squared distances greater than 1sq.m. In Figure 11, point colors denote the squared distances, and the colored bars show the percentage of points at different squared distance levels. As the quantitative results in Table 1 illustrate, our method (first column) is the most accurate algorithm to produce 2.5D models. Plane-based approaches such as [14] (second column) are unable to handle non-flat roofs (a,d) and small roof features (b,e). Cracks often exist when fitting is unsuccessful (c,d). A general mesh simplification over the DEM (third column) is competitive in the sense of fitting quality. However, it cannot produce 2.5D models composed of roofs and vertical walls. In addition, the fitting quality on roof boundaries is unsatisfactory (f,g,h). The last column demonstrates point clouds aligning with manually created models. Designed without knowledge from real-world data, they often lack of accuracy even after registration to the input points.

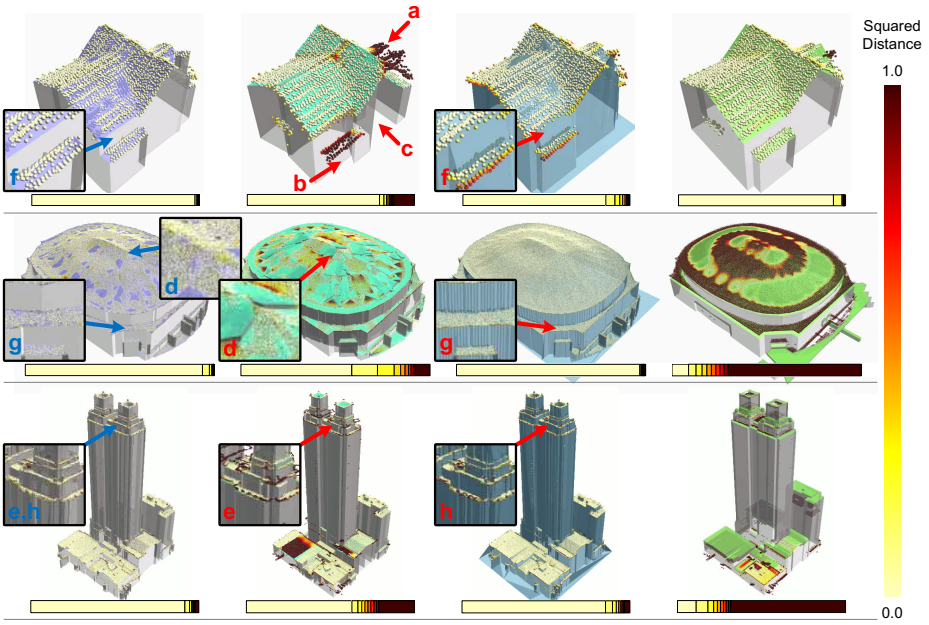

Fig. 11. Building models created using different approaches (from left to right): 2.5D dual contouring, plane-based method proposed in [14], general mesh simplification over a rasterized DEM, and manual creation. Color bars under the models show the ratio of points at different squared distance level.

Models in Figure 11		2.5D dual contouring	Plane-based method [14]	DEM simplification	Manual creation [3]
First row (4679 points)	Triangle number	214	76	198	78
	Average distance^2	0.016	0.599	0.061	0.058
	Outlier ratio	0.06%	12.37%	0.53%	0.83%
Second row (684907 points)	Triangle number	8009	6262	8000	1227
	Average distance^2	0.037	0.465	0.035	7.780
	Outlier ratio	0.44%	7.93%	0.87%	70.38%
Third row (198551 points)	Triangle number	12688	1619	12999	1558
	Average distance^2	0.203	1.610	0.264	16.220
	Outlier ratio	2.03%	21.15%	3.08%	68.28%

Table 1. Quantitative evaluation of experiments shown in Figure 11.

We finally demonstrate the influence of grid configuration in Figure 12. As an adaptive approach, our method is insensitive to the grid size (top row). In addition, 2.5D dual contouring has the ability to place vertices at optimal positions, thus grid orientation affects the results insignificantly (bottom row).

10 Conclusion

We present a robust method to automatically creating building models from aerial LiDAR point clouds. Our results are 2.5D models composed of complex

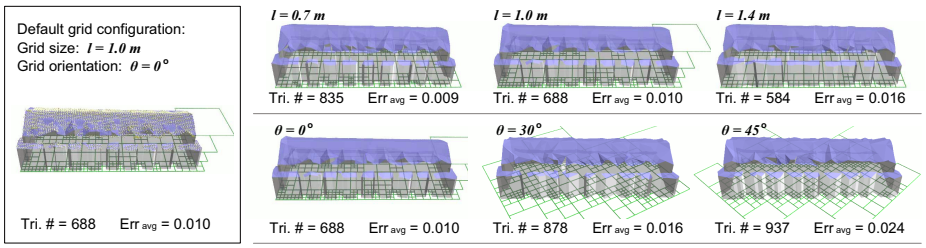

Fig. 12. Models of similar quality are generated with the same point cloud embedded into grids of different sizes or different orientations.

building roofs connected by vertical walls. By extending dual contouring into a 2.5D method, our algorithm optimizes the surface geometry and the boundaries of roof layers simultaneously. The output models are guaranteed to be crack-free meshes with small fitting error, faithfully preserving sharp features.

References

1. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: ACM SIGGRAPH (1996)
2. Fiocco, M., Boström, G., Gonçalves, J.G.M., Sequeira, V.: Multisensor fusion for volumetric reconstruction of large outdoor areas. 3DIM (2005)
3. Google: Google 3d warehouse. http://sketchup.google.com/3dwarehouse/
4. Ju, T., Losasso, F., Schaefer, S., Warren, J.: Dual contouring on hermite data. In: ACM SIGGRAPH (2002)
5. Lafarge, F., Descombes, X., Zerubia, J., Pierrot-Deseilligny, M.: Building reconstruction from a single dem. In: CVPR (2008)
6. Lindstrom, P.: Out-of-core simplification of large polygonal models. In: ACM SIGGRAPH (2000)
7. Lorensen, W., Cline, H.: Marching cubes: A high resolution 3d surface construction algorithm. In: ACM SIGGRAPH (1987)
8. Matei, B., Sawhney, H., Samarasekera, S., Kim, J., Kumar, R.: Building segmentation for densely built urban regions using aerial lidar data. In: CVPR (2008)
9. Pauly, M.: Point primitives for interactive modeling and processing of 3d geometry. PhD thesis, ETH Zurich (2003)
10. Poullis, C., You, S.: Automatic reconstruction of cities from remote sensor data. In: CVPR (2009)
11. Verma, V., Kumar, R., Hsu, S.: 3d building detection and modeling from aerial lidar data. In: CVPR (2006)
12. You, S., Hu, J., Neumann, U., Fox, P.: Urban site modeling from lidar. In: Proceedings, Part III. ICCSA (2003)
13. Zebedin, L., Bauer, J., Karner, K., Bischof, H.: Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In: ECCV (2008)
14. Zhou, Q.Y., Neumann, U.: Fast and extensible building modeling from airborne lidar data. In: ACM GIS (2008)
15. Zhou, Q.Y., Neumann, U.: A streaming framework for seamless building reconstruction from large-scale aerial lidar data. In: CVPR (2009)