Photorealistic Large-Scale Urban City Model Reconstruction

Charalambos Poullis, *Student Member*, *IEEE*, and Suya You, *Member*, *IEEE*

Abstract—The rapid and efficient creation of virtual environments has become a crucial part of virtual reality applications. In particular, civil and defense applications often require and employ detailed models of operations areas for training, simulations of different scenarios, planning for natural or man-made events, monitoring, surveillance, games, and films. A realistic representation of the large-scale environments is therefore imperative for the success of such applications since it increases the immersive experience of its users and helps reduce the difference between physical and virtual reality. However, the task of creating such large-scale virtual environments still remains a time-consuming and manual work. In this work, we propose a novel method for the rapid reconstruction of photorealistic large-scale virtual environments. First, a novel, extendible, parameterized geometric primitive is presented for the automatic building identification and reconstruction of building structures. In addition, buildings with complex roofs containing complex linear and nonlinear surfaces are reconstructed interactively using a linear polygonal and a nonlinear primitive, respectively. Second, we present a rendering pipeline for the composition of photorealistic textures, which unlike existing techniques, can recover missing or occluded texture information by integrating multiple information captured from different optical sensors (ground, aerial, and satellite).

Index Terms—Large-scale modeling, rapid reconstruction, photorealistic model.

1 Introduction

Virtual reality technologies are becoming increasingly popular and are being widely used in a range of different applications. In particular, civil and defense applications employ such technologies for the simulation of real-world operations areas. In such cases, the models of urban buildings are of significant value since they facilitate planning, response, and real-time situational awareness in highly occluded urban settings. The personnel that simulate, plan, monitor, and execute responses to natural or man-made events can gain insight and make better decisions if they have a comprehensive view of the structures and activity occurring at an operational scene. The models are essential components of such a view, helping people comprehend spatial and temporal relationships. In addition, a photorealistic appearance is essential for the enhancement of the visual richness of the models and the immersive experience of the users.

While models are important assets, the creation of photorealistic large-scale models remains at best a difficult, time-consuming manual task. The science for rapidly sensing and modeling wide-area urban sites and events, in particular, pose difficult or unsolved problems. Over the years, a wealth of research, employing a variety of sensing and modeling technologies, has been conducted to deal with the complex modeling problem. Different types of techniques, ranging from computer vision, computer graphics, photogrammetry, and remote sensing, have been proposed and developed so far; each has unique strengths and weaknesses, and each performs well for a particular data set but may fail under another.

Similarly, the generation of high-resolution photorealistic textures has been extensively investigated and many algorithms have been already proposed. However, the complexity of these algorithms increases considerably when dealing with large-scale virtual environments. In addition, these texturing techniques for large-scale environments are limited to using a single image per building which makes it impossible to recover textures for missing or occluded areas and requires even more time-consuming manual work. These problems impose a serious limitation in achieving a realistic appearance for the models, and in extent, the immersive experience of the users is diminished.

In this work, we address the problem of rapid creation of photorealistic large-scale virtual environments. First, we address the 3D model reconstruction problem and propose a novel, extendible parameterized geometric primitive for the automatic identification and reconstruction of building models. We leverage the symmetry constraints found in man-made structures to reduce the number of unknown parameters needed during the model reconstruction, thus considerably reducing the computational time required. In addition, buildings containing complex linear and nonlinear surfaces such as churches, domes, stadiums, etc., are interactively reconstructed using a linear polygonal and a nonlinear primitive, respectively.

Second, we address the problem of texturing, and propose a rendering pipeline for the composition of photorealistic textures. A significant advantage of this pipeline is that textures for missing or occluded areas in one image can be recovered from another image, thus eliminating the need for any manual editing work. Images captured from multiple sensors (ground, aerial, and satellite) are integrated together to produce a set of view-independent, seamless textures.

- *C. Poullis is with the Computer Graphics and Immersive Technologies Laboratory, University of Southern California, 3737 Watt Way, PHE 108, Los Angeles, CA 90089. E-mail: charalambos@poullis.org.*
- *S. You is with the Computer Graphics and Immersive Technologies Laboratory, University of Southern California, 3737 Watt Way, PHE 432, Los Angeles, CA 90089. E-mail: suyay@graphics.usc.edu.*

Manuscript received 29 Feb. 2008; revised 1 Aug. 2008; accepted 6 Oct. 2008; published online 10 Oct. 2008.
Recommended for acceptance by B. Guo.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2008-02-0029.
Digital Object Identifier no. 10.1109/TVCG.2008.189.

We have extensively tested the proposed approach with a wide range of sensing data including satellite, aerial, ground photographs, and Light Detection and Ranging (LiDAR), and present our results.

2 Related Work

2.1 Modeling

A good survey on large-scale modeling techniques can be found in [8]. In [7], the authors present a method for reconstructing large-scale 3D city models by merging ground-based and airborne-based LiDAR data. The elevation measurements are used to recover the geometry of the roofs. Facade details are then incorporated by the high-resolution capture of a ground-based system which has the advantage of also capturing texture information. The textures aid in the creation of a realistic appearance of the model. However, at the cost of having detailed facades, they neglect to deal with the complexities and wide variations of the buildings' roof-types. The same authors later extended their method to incorporate texture information from oblique aerial images. Although they combine multiple aerial images to determine the model's textures, their method is restricted to traditional texture mapping rather than combining all available texture information to generate a composite texture, i.e., blending. Therefore, a significant color difference between images will cause visible and nonsmooth transitions between neighboring polygons of different texture images.

You et al. [20] present an interactive primitive-based modeling system for the reconstruction of building models from LiDAR data. Using the user input, their system automatically segments the building boundary, performs model refinement, and assembles the complete building model. However, user input is required for the detection as well as the identification of buildings and their roof-types.

In [17], the authors developed an interactive system for reconstructing geometry using nonsequential views from uncalibrated cameras. The calculation of all 3D points and camera positions is performed simultaneously as a solution of a set of linear equations by exploiting the strong constraints obtained by modeling a map as a single affine view. However, due to the considerable user interaction required by the system, its application to large-scale areas is very limited.

The proposed system can deal with uncalibrated image sequences acquired with a handheld camera in [14]. Based on tracked or matched features, the relations between multiple views are computed. From this, both the structure of the scene and the motion of the camera are retrieved. Although the reconstructed 3D models are visually impressive, they consist of complex geometry—as opposed to simple polygonal models—which requires further processing and limits their applications.

Nevatia et al. [13] propose a user-assisted system for the extraction of 3D polygonal models of buildings from aerial images. Low-level image features are initially used to build high-level descriptions of the objects. Using a hypothesize and verifying paradigm, they are able to extract impressive models from a small set of aerial images. The authors later extended their work in [9] to automatically estimate camera pose parameters from two or three vanishing points and three 3D to 2D correspondences.

A ground-based LiDAR scanner is used [4] to record a rather complex ancient structure of significant cultural heritage importance. Multiple scans were aligned and merged together using a semiautomatic process, and a complete 3D model was created from the outdoor structure. The reconstructed model is shown to contain high level of details; however, the complexity of the geometry (90 million polygons for one building) limits this approach to the reconstruction of single buildings rather than large scale. Another ground-based approach is presented in [12], where a two-stage process is employed in order to quickly fuse multiple stereo-depth maps. The results are impressive especially for a real-time system; however, the resulting geometry is too complex and requires further processing in order to make it usable in another application. A similar ground-based approach is presented in [21], where multiple-range images are integrated by minimizing an energy functional consisting of a total variation regularization force and an L^1 data fidelity term. Similarly, the resulting geometry, although impressive, is too "heavy" for most applications.

In a different approach, an interactive system is proposed in [5], which can reconstruct buildings using ground imagery and a minimal set of geometric primitives. More recently, this system was extended in [15] to incorporate point cloud support as part of the reconstruction; however, the required user interaction increases considerably for large-scale areas. Moreover, the user interaction depends on the desired level of detail of the reconstructed models, which may vary considerably according to the application.

In [16], the authors presented an interactive modeling system that can detect and model large-scale buildings having limited shape variations, such as single plane, cube, saltbox, cylinder, sphere, etc. The system first segmented interactively the building boundaries from LiDAR data, and then used geometric primitives for model fitting and refinement. In this work, we significantly extend the system to handle more complex building types by introducing several new techniques. A fully automatic approach is employed for the segmentation of buildings and vegetation. The parameterization technique is extended to handle more complicated buildings such as L-shaped buildings, and a new parameterized primitive is introduced for the automatic identification and reconstruction of the buildings. Any complex building resulting from the combination of single building structures can be modeled by using a new polygonal primitive. The new system has been extensively tested, and it demonstrated its flexibility and capability for rapidly modeling a wide range of complex buildings.

2.2 Texturing

One of the most popular and successful techniques in this area is the one introduced in [5] which uses a small set of images to reconstruct a 3D model of the scene. View-dependent texture mapping (VDTM) is then performed for the computation of the texture maps of the model. By interpolating the pixel color information from different images, new renderings of the scene can be produced. The contributions of each image to a pixel's color is weighted based on the angle difference between the camera's direction and the novel viewpoint's direction. The authors then extended their work and showed how VDTM can be efficiently implemented using projective texture mapping,

a feature available in most computer graphics hardware. Although this technique is sufficient to create realistic renderings of the scene from novel viewpoints, its computation is still too expensive for real-time applications, like games or virtual reality. In addition, view-dependent texture mapping works seemingly in cases where the images are taken at regularly sampled intervals, which is not generally true in the context of large-scale areas. Most images come from satellites and aerial images are taken from a wide baseline; thus, view-dependent texture mapping would most definitely fail in such cases.

In [11], Martinez and Dreiiakis order geometry into optimized visibility layers for each photograph. The layers are subsequently used to create standard 2D image-editing layers which become the input to a layered projective texture rendering algorithm. However, this approach chooses the best image for each surface rather than combining the contributions of all the images to minimize information loss.

Reflectance properties of the scene were measured in [4] and lighting conditions were recorded for each image taken. An inverse global illumination technique was then used to determine the true colors of the model's surfaces which could then be used to relight the scene. The generated textures are photorealistic; however, for large-scale areas, it requires considerable amount of manual work for the capturing and the processing of the data.

A method for creating renders from novel viewpoints without the use of geometric information is presented in [10], where densely regularly sampled images are blended together. The image capture for small objects is very easily achieved; however, this task is close to impossible to perform for large-scale areas.

A slightly different approach for texture generation and extraction is proposed in [19]. Given a texture sample in the form of an image, a similar texture is created over an irregular mesh hierarchy that has been placed on a given surface; however, this approach cannot capture the "actual" appearance of the models.

3 System Overview

The system's overview is summarized in Fig. 1. It consists of three components:

1. preprocessing discussed in Section 4,
2. modeling discussed in Section 5, and
3. texturing discussed in Section 6.

In the Preprocessing component, the airborne LiDAR data are resampled into a 2D regular grid structure, refined (hole filling, smoothing) using graph-cut optimization and gradient-descent, and segmented into two clusters, vegetation or ground. The Preprocessing component plays an essential role in the accuracy and success of the Modeling component, since it reduces the noise and inconsistencies from the data while ensuring that important features such as discontinuities are preserved. In addition, the use of a regular grid structure greatly reduces the processing time since all subsequent optimizations are performed in 2D rather than in 3D.

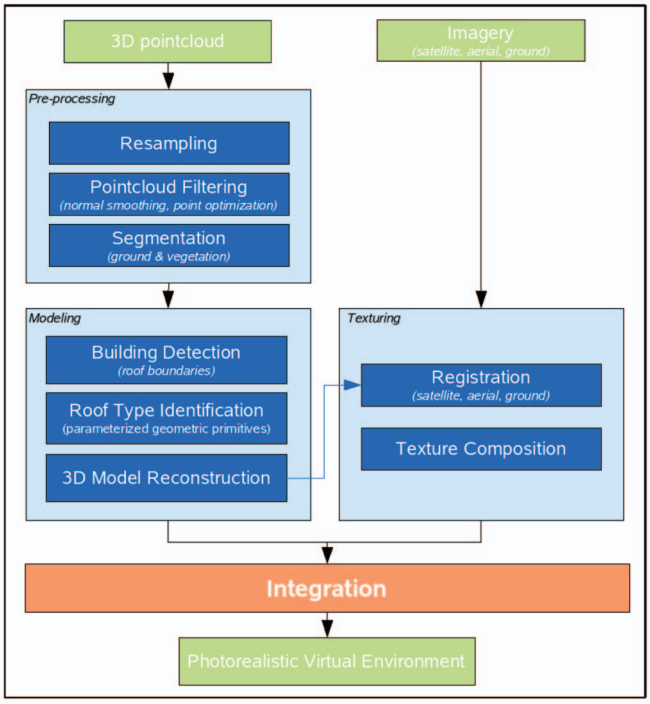


Fig. 1. System overview. Three main components: preprocessing, modeling, and texturing.

In the Modeling component, initial 2D roof boundaries are extracted from the previously generated regular grid, either automatically for simple roof-types, or interactively for complex and nonlinear roof-types. Three novel geometric primitives are developed to refine the initial roof boundaries in a coordinate-wise optimization:

1. an extendible parameterized primitive which can automatically identify the most commonly occurring roof-types;
2. a polygonal primitive for complex linear roof-types;
3. a nonlinear primitive for nonlinear roof-types. Once the refined boundaries are determined by the optimization, the 3D models are generated by fitting the appropriate surface on the interior roof points and extruding the roof boundaries to 3D. A significant advantage of this approach is that a single parameterized primitive can handle multiple roof-types and therefore reduce the total number of primitives required to reconstruct a large-scale urban area.

Finally, in the Texturing component, the camera poses for the ground, aerial, and satellite images are calculated using interactively registered correspondences to the reconstructed 3D models, and are used by the rendering pipeline to generate the composite photorealistic textures. A key aspect of this pipeline is the integration of multiple information for the efficient and effective handling of textures for missing or occluded areas.

4 Point Cloud Data Preprocessing

The preprocessing of the raw 3D point cloud data is performed in three steps: resampling, filtering, and

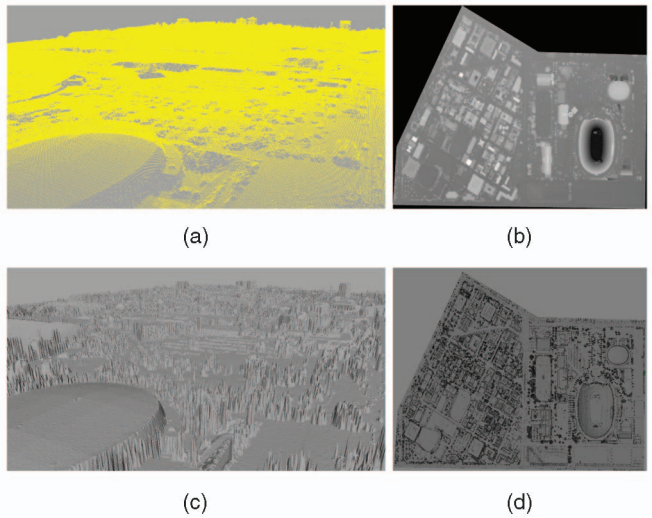


Fig. 2. Resampling of the 3D point cloud data into a regular grid. (a) 3D Point cloud, (b) 2D Regular grid, (c) Triangulated mesh, and (d) Triangulated mesh (top view).

segmentation. The result is a classification of the refined points into ground and vegetation.

4.1 Resampling

The unstructured 3D point cloud (Fig. 2a) is initially resampled into a 2D regular grid (map shown in Fig. 2b). To overcome the problem of information loss of the samples stored in the grid, the dimensions of the grid are determined on the basis of the sampling rate which is automatically calculated either by specifying the desired resolution or by specifying the maximum allowed error tolerance between the samples. Figs. 2c and 2d show the triangulated mesh generated from the resampled points.

4.2 Filtering

Noise in the measurements of the original data and inconsistencies introduced by the resampling process are removed while ensuring that important features such as discontinuities are preserved. To achieve this, we exploit the fact that normals provide directional information which define the surface geometry of an object and perform a normal optimization based on graph cuts to smooth the normals, followed by a point optimization using gradient descent to smooth the points.

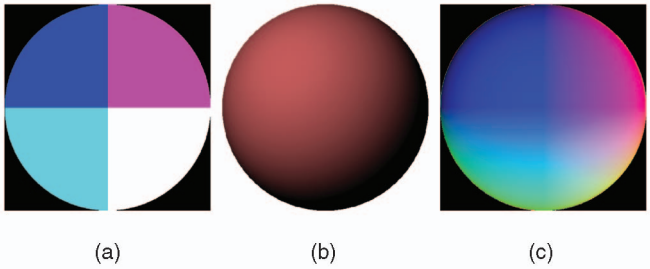


Fig. 4. Computation of smooth normal labels from a half-sphere. (a) Half-sphere point map, (b) Half-sphere in 3D (top-view), and (c) Smooth normal labels computed from a.

The process is summarized in Fig. 3. A set of uniformly distributed normals (Fig. 4c) computed from a half-sphere (Figs. 4a and 4b) are used to relabel the noisy normals computed from the original points [3]. Local neighborhood information is used to compute the normal at each point in the resampled map. For every point P_i, we define the normal N_{P_i} of that point as

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

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

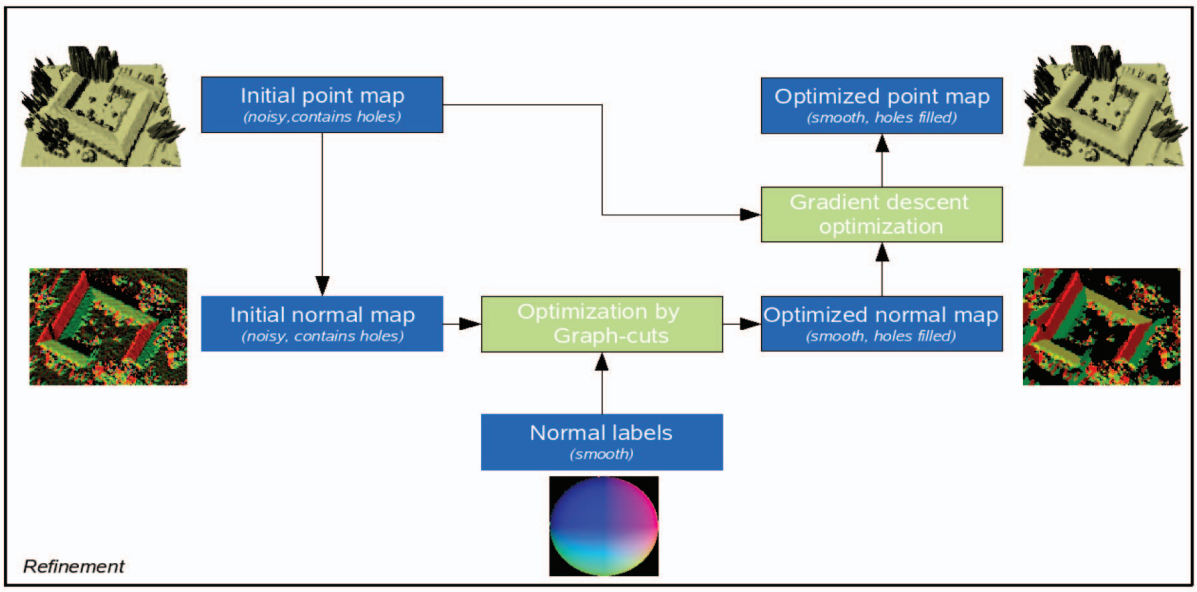


Fig. 3. Refinement process overview.

This results in smoothing the normals while preserving important features such as discontinuities [7]. A gradient-descent optimization then refines the original points such that their computed normals are identical to the smooth relabeled normals.

We restate the problem of smoothing the noisy normals as a problem of finding an optimal labeling $f : N_p \longrightarrow L$ which assigns a normal label $l \in L$ to each point $p \in N$, where f is piecewise smooth and consistent with the original data, and N is the initial normal map. This labeling problem can be efficiently solved using graph cuts to minimize an energy function of the form

$$E(f) = E_{data}(f) + \lambda * E_{smoothness}(f), \tag{2}$$

where λ is a smoothness weighting factor. Refer to the Appendix for further details about graph cuts.

The *energy data term* in (2) provides a per-pixel measure of how appropriate a label $l \in L$ is for a pixel $p \in N$ in the *observed data* and is given by

$$E_{data}(f) = \sum_{p \in N} D_p(f(p)), \tag{3}$$

where $D_p(f_p)$ is the data function for a point $p \in N$ and is defined as

$$D_p(f_p) = |N_p - N_{f(p)}|, \tag{4}$$

where N_p is the original normal of the point p and $N_{f(p)} \in L$ is the assigned normal label under the labeling $f(p) : N_p \longrightarrow N_{f(p)}$. Thus, the energy data term in (3) becomes

$$E_{data}(f) = \sum_{p \in N} |N_p - N_{f(p)}|. \tag{5}$$

The *energy smoothness term* in (2) provides a measure of the difference between two neighboring pixels $p, q \in N$ with labels $N_{f(p)}, N_{f(q)} \in L$, respectively, and is given by

$$E_{smoothness}(f) = \sum_{\{p,q\} \in N} V_{\{p,q\}}(N_{f(p)}, N_{f(q)}), \tag{6}$$

where $N_{f(p)}, N_{f(q)} \in L$ are the assigned normal labels under the labeling f and $V_{\{p,q\}}$ is the interaction potential between the neighboring pixels $p, q \in N$. Let N_p and N_q be the original normals in the *observed data* of the pixels $p, q \in N$, respectively, then we define a measure of the *observed smoothness* between pixels $p, q \in N$ as

$$\Delta_{p,q} = |N_p - N_q|. \tag{7}$$

Similarly, we define a measure of smoothness for the global minimization. Let $N_{f(p)}$ and $N_{f(q)}$ be the assigned normal labels under a labeling f, then we define a measure of the *smoothness between the labels* of neighboring pixels $p, q \in N$ as

$$\tilde{\Delta}_{p,q} = |N_{f(p)} - N_{f(q)}|. \tag{8}$$

Using the smoothness measure defined for the observed data and the smoothness measure defined for any given labeling, the energy smoothness term in (6) becomes

$$E_{smoothness}(f) = \sum_{\{p,q\} \in N} (K_{p,q} * \tilde{\Delta}_{p,q}), \tag{9}$$

where $K_{p,q} = 1 + \epsilon - e^{-\frac{2-\Delta_{p,q}|}{\sigma^2}}$ is the Boltzmann distribution of the energy function $\Delta_{p,q}$ which gives the probability of the initial smoothness, σ controls the smoothness uncertainty, and ϵ is a small constant. This ensures that two neighboring pixels $p, q \in N$ with similar normal orientations in the *observed data* will have small $\Delta p, q$ and, thus, a high probability, given by $K_{p,q}$, which *the assigned labels under the labeling* f will also have similar orientations and therefore $\tilde{\Delta}_{p,q}$ will be small.

We define the set of smooth normal labels L to be a set of uniformly distributed normals lying on the surface of a half-sphere. Fig. 4a shows the point map corresponding to the 3D half-sphere in Fig. 4b. Fig. 4c shows the set of smooth normal labels computed from the half-sphere's point map in Fig. 4a. The radius of the half-sphere can be altered accordingly to control the smoothness and the number of normal labels in the set L.

The result of the normal-based graph cut optimization is a smooth normal map. Using the initial noisy points as initial estimates, a point-based gradient-descent optimization is employed, in which the spatial position of the initial points is adjusted in every iteration, such that the computed normal N_p at a point $p \in N$ matches the smooth label normal $N_{f(p)}$ returned by the graph cut optimization.

Our experiments have shown that the point-based gradient-descent optimization is extremely fast since the initial points are used as estimates and the optimization converges to a solution within 10 iterations. Another significant advantage of the refinement process is that the hole filling is performed as a by-product of the normal smoothing since the labeling f returned by the graph cut optimization will include a label for all points in the map even if the point initially has no label due to missing information, i.e., a hole. In addition, the labeling is performed on the premises of the observed data and labels of the local neighborhood of the missing points, which is controlled by the smoothness term, thus ensuring that the new labels will not affect the consistency of the geometry.

For example, the initial normals in Fig. 5b are computed from the original points in Fig. 5a. A graph cut optimization performed using these initial normals and the smooth normal labels from Fig. 5c results in the relabeled normals shown in Fig. 5c. Finally, a gradient-descent optimization iteratively refines the original points such that the computed normal at each point matches the relabeled normal (Fig. 5d). Fig. 5e shows the difference measured as the dot product between the initial and smooth normal map.

As previously explained, there are two parameters which control the refinement process:

1. the smoothness term weight λ in (2), and
2. the size of the set of smooth labels L, which was defined in terms of the radius of the half-sphere.

A closeup of a mesh produced by triangulating the refined points is shown in Fig. 6. Different number of smooth normal labels $\|L\|$ (computed by varying the radius of the half-sphere) and different values for the smoothness weight λ are shown. Our experiments have shown that $\lambda = 0.25$ and $L = [100, 500]$ generate satisfactorily smooth results.

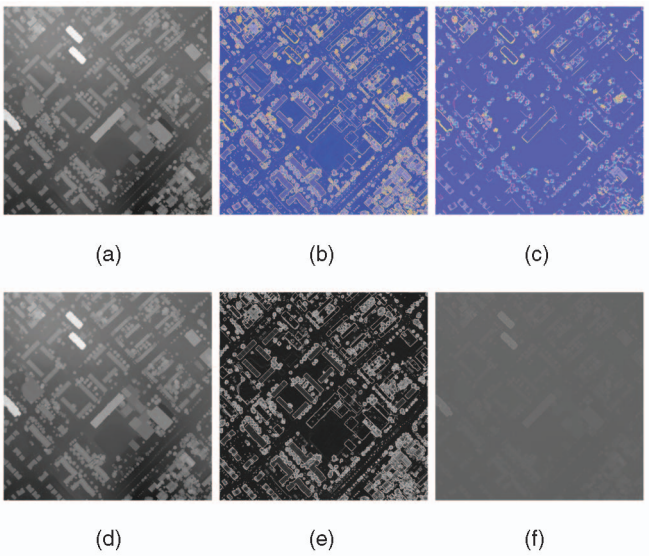


Fig. 5. The filtering is performed in two steps: normal-based optimization using graph cuts for smoothing the normals and point-based gradient-descent optimization for smoothing the points using the smooth normals. (a) Initial point map, (b) Initial normal map, (c) Smooth normal map, (d) Smooth point map, (e) Dot-product between initial and smooth normal maps, and (f) Euclidian distance between initial and smooth point maps (normalized).

4.3 Segmentation

Buildings are man-made structures with the characteristic of having roofs with uniform elevation and uniform orientation. On the other hand, the elevation and orientation of trees and other vegetation rapidly changes between neighboring points. The goal of the segmentation is to determine points with similar (or linearly varying) elevation and similar orientation, and to group them into regions representing building candidates. In addition, regions consisting of a number of points less than a minimum are discarded.

4.3.1 Automatic Segmentation

We employ an automatic segmentation technique called skewness-balancing which was introduced by [1] based on the central limit theorem [6], which states that naturally measured samples will lead to a normal distribution. The assumption is that the terrain can be modeled by a normal distribution and any man-made structures will disturb that distribution. Thus, by removing the (nonground) points which disturb the distribution from the data, the ground points can be obtained. To achieve this, the third moment about the mean—also known as skewness—is used to measure the assymetry of the distribution and the fourth moment—also known as kurtosis—is used to measure the size of the distribution's tail. Experiments have shown that depending on the characteristics of the distribution, i.e., of the terrain points, the skewness and kurtosis factors vary, as summarized by Table 1 [1].

An iterative algorithm repeatedly removes the point with the highest elevation until the skewness and kurtosis factors are equal to the desired values. Figs. 7a and 7b show the ground and nonground points, respectively, resulting from the interactive segmentation.

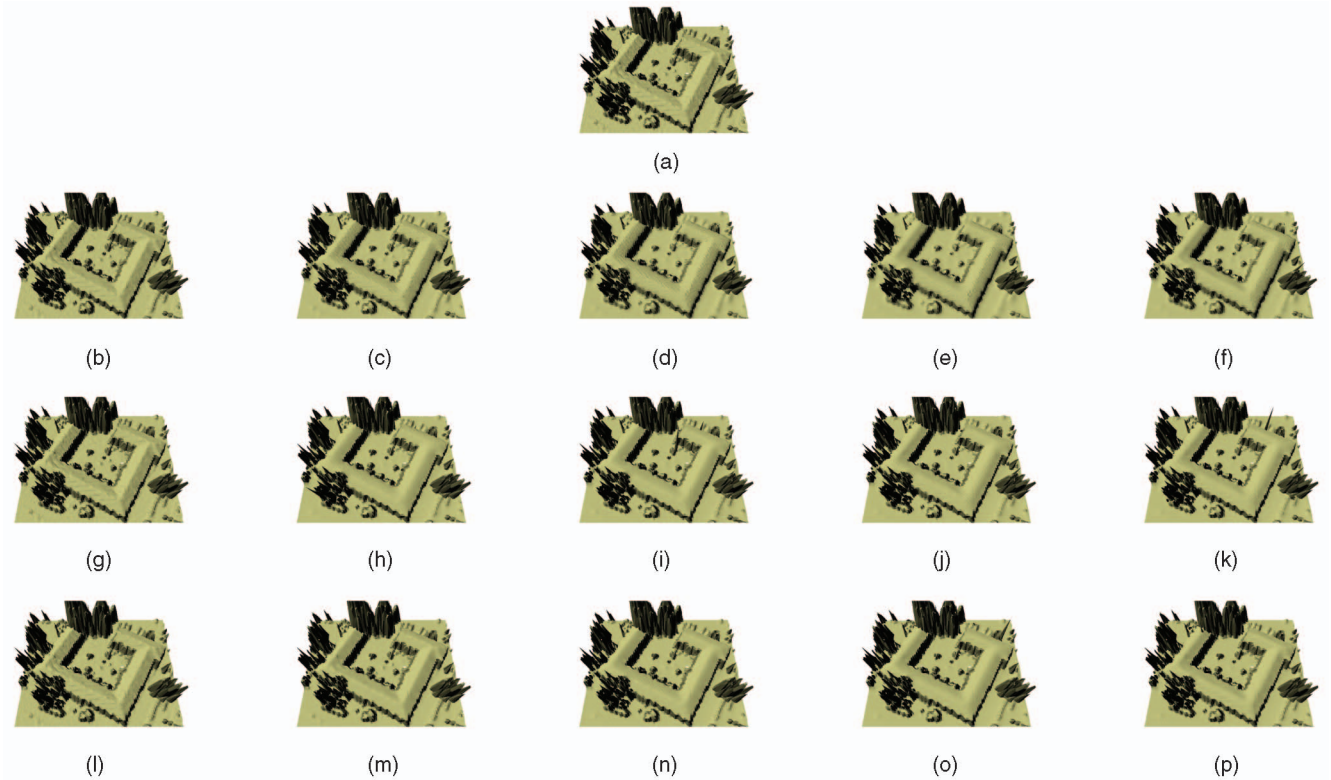


Fig. 6. Results for different smoothness weights λ and different sizes $\|L\|$ of the normal label set L. (a) Original data, (b) $\|L\| = 17$, $\lambda = 0.1$, (c) $\|L\| = 17$, $\lambda = 0.25$, (d) $\|L\| = 17$, $\lambda = 0.5$, (e) $\|L\| = 17$, $\lambda = 0.75$, (f) $\|L\| = 17$, $\lambda = 1.0$, (g) $\|L\| = 104$, $\lambda = 0.1$, (h) $\|L\| = 104$, $\lambda = 0.25$, (i) $\|L\| = 104$, $\lambda = 0.5$, (j) $\|L\| = 104$, $\lambda = 0.75$, (k) $\|L\| = 104$, $\lambda = 1.0$, (l) $\|L\| = 492$, $\lambda = 0.1$, (m) $\|L\| = 492$, $\lambda = 0.25$, (n) $\|L\| = 492$, $\lambda = 0.5$, (o) $\|L\| = 492$, $\lambda = 0.75$, and (p) $\|L\| = 492$, $\lambda = 1.0$.

TABLE 1
Skewness and Kurtosis Values in Relation to the Distribution's Characteristics

Distribution's Characteristic	Dominance of peaks	Dominance of valleys	Normal distribution
Skewness	> 0	< 0	0
Kurtosis	> 3	< 3	3

4.3.2 Interactive Segmentation

The automatic segmentation performs very well when dealing with relatively large buildings of high elevations. However, problems arise when small buildings of low elevation are present, which is clearly demonstrated in the result shown in Fig. 7a. Although the majority of the buildings (about 70 percent) has been successfully classified as being nonground elements, there are still several buildings which were misclassified as ground elements due to their low elevation.

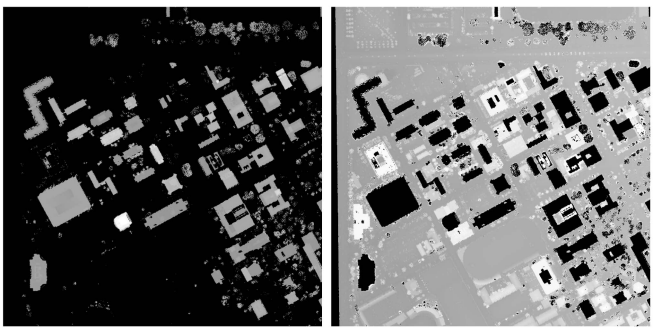
In order to overcome this problem, we also employ an interactive segmentation based on region growing which ensures that all the buildings have been correctly classified as nonground elements, and perform a region growing on a few user-defined points to segment groups of similar elevation and orientation ($\Delta\chi = 0.01$ (normalized values) and an orientation threshold of $\Delta\theta = 30^{o}$ for the results shown). Fig. 8 shows the resulting ground (Fig. 8b) and nonground (Fig. 8a) maps after the interactive segmentation.

5 Modeling

The modeling of the buildings from the segmented candidate points is performed in three steps: building detection, roof-type identification, and model reconstruction.

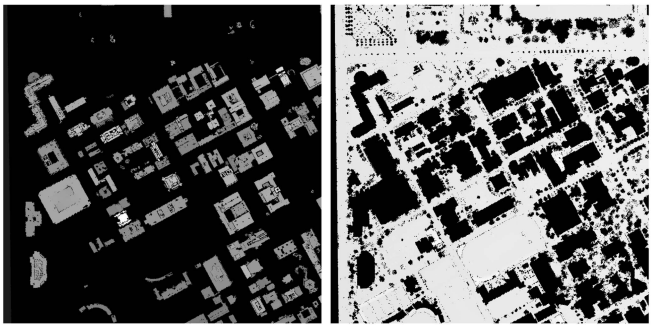
5.1 Building Detection

One of the main limitations of airborne LiDAR scanners is the accurate measurements near discontinuities due to the rapid change in elevation which often leads to boundaries with a zig-zag shape. To overcome this problem, the contours of the detected regions are linearized using Douglas-Peucker polygonal approximation. In addition, a series of polygonal Boolean operations are applied to the boundaries to resolve any inconsistencies produced by overlaps due to structures on the roofs such as chimneys, elevator engines, air conditioners, etc.



(a) (b)

Fig. 7. Automatic segmentation into (a) nonground and (b) ground maps.

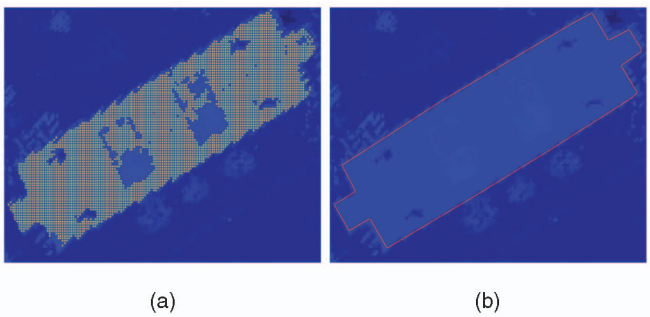


(a) (b)

Fig. 8. Supervised segmentation into (a) nonground and (b) ground maps.

Automatic building detection. Commonly found buildings have simple roofs and consist of planar surfaces. We exploit this characteristic and employ an automatic method for the detection of buildings with simple, convex, or concave roofs. Fig. 9 shows an example of a building with a concave roof shape. The point with the highest elevation is automatically extracted from the segmented data and is used as a starting point for region-growing. As mentioned previously, a set of polygonal Boolean operations is applied to the building's interior and exterior convex hulls in order to extract the actual boundaries of the building.

Interactive building detection. Although the majority of the buildings contain linear surfaces which can be automatically detected, in some cases where the buildings contain complex and nonlinear surfaces, an interactive approach is employed for the detection. Currently, interactive methods rely on user input for the detection and require the precise marking of the buildings' boundaries. We relax the requirement for precision marking and allow the user to roughly specify a set of control points which define the shape of the buildings. The spatial position of the user-marked control points is then refined through a coordinate-wise optimization, as explained in Sections 5.2 and 5.3.



(a) (b)

Fig. 9. Automatic building detection. (a) Building detection. Initial seed point is displayed in red. (b) Recovered roof boundaries.

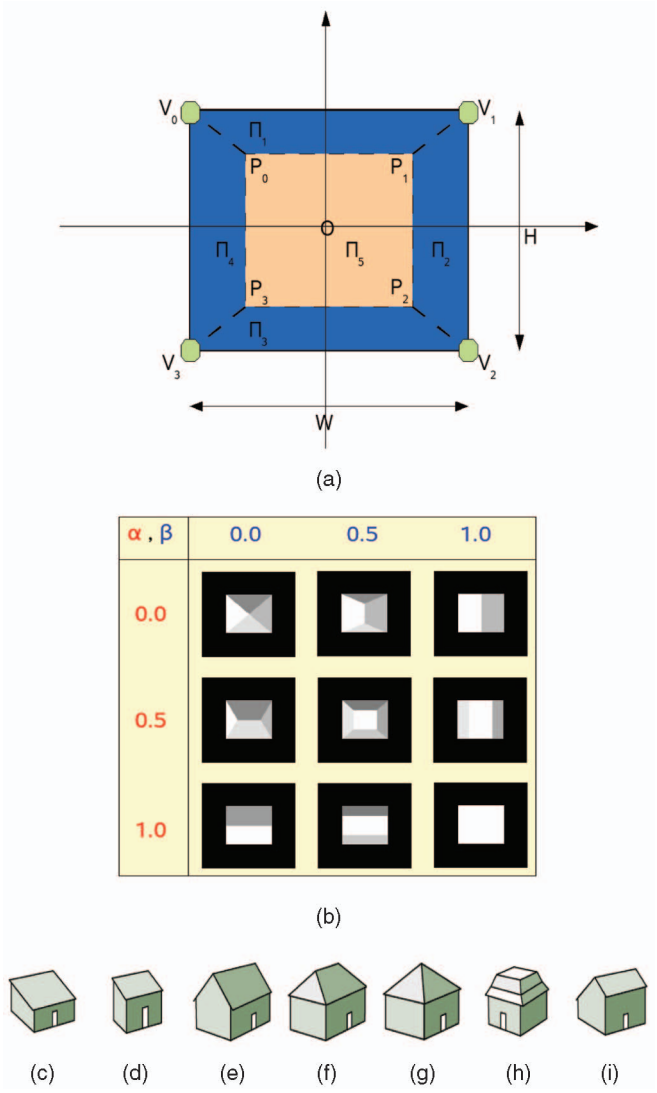


Fig. 10. The novel parameterized geometric primitive and the types of roofs automatically identified. (a) Parameterized primitive. (b) The different roof-types generated by varying the parameters $\alpha\ \beta$ (top view). (c) Flat. (d) Shed. (e) Glabe. (f) Hip. (g) Pyramidal. (h) Mansard. (i) Saltbox.

5.2 Automatic Identification and Reconstruction of Linear Roof Types

The identification of building roof-types is an essential part of the modeling process. In many cases, especially for complex buildings and low-resolution data, this is still a very difficult task even for a human operator. We address this problem and present a novel, extendible parameterized geometric primitive for the automatic identification of the most commonly occurring roof-types, as defined in Fig. 10. In addition, we show how such a primitive parameterization can be extended in order to similarly handle other groups of complex roof-types.

We leverage the symmetry constraints found in man-made structures and parameterize a geometric primitive using only two variables. Fig. 10a shows the geometric structure of the parameterized primitive. By specifying a local coordinate system, the control points (external points $V_0, \ldots, V_3$ and internal points $P_0, \ldots, P_3$) are expressed as functions of two variables α, β such that

$$V_0 = \left[-\frac{W}{2}, -\frac{H}{2}\right], \qquad V_1 = \left[\frac{W}{2}, -\frac{H}{2}\right], \tag{10}$$

$$V_2 = \left[-\frac{W}{2}, \frac{H}{2}\right], \qquad V_3 = \left[\frac{W}{2}, \frac{H}{2}\right], \tag{11}$$

$$P_0 = \left[-\alpha\frac{W}{2}, -\beta\frac{H}{2}\right], \qquad P_1 = \left[\alpha\frac{W}{2}, -\beta\frac{H}{2}\right], \tag{12}$$

$$P_2 = \left[\alpha\frac{W}{2}, \beta\frac{H}{2}\right], \qquad P_3 = \left[-\alpha\frac{W}{2}, \beta\frac{H}{2}\right], \tag{13}$$

where W is the width and H is the height of the primitive. This parameterization allows for the symmetry constraints to be enforced since a change of a single point will affect all other points. In order to ensure that all the internal points $P_0, \ldots, P_3$ are always within the area defined by the exterior points $V_0, \ldots, V_3$, the parameters are bound in the range $0 \leq \alpha, \beta \leq 1$. By varying the values of the two parameters α, β, all commonly occurring roof-types can be produced by a *single* primitive, as demonstrated in Fig. 10b. This is a major advantage since it significantly reduces the number of primitives required to model a scene and allows a *single* primitive to be used for the automatic identification of *multiple* roof-types.

A nonlinear, bound-constraint minimization is then performed to find the optimal values of the two variables α, β, such that E_{error} in (14), i.e., the sum of the squared fitting error of the five planar surfaces, is minimum:

$$E_{error} = \sum_{n=0}^{N} \sum_{\forall p \in P} (\chi_p - f(\chi_p, \alpha, \beta))^2, \tag{14}$$

where N is the number of planes (5) and $f(.,.,.)$ is a least-squares fitting function.

Another significant advantage is that the optimization involves only two unknowns which are bound constraint due to the condition that $0 \leq \alpha, \beta \leq 1$, thus making the convergence to a solution extremely fast.

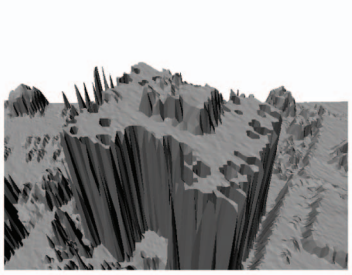
During this optimization, a Gaussian mixture model (GMM) is used to model the distribution of the elevation of all points $p \in P$ inside the planar surface boundaries. A GMM is a superposition of K Gaussian densities of the form

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k), \tag{15}$$

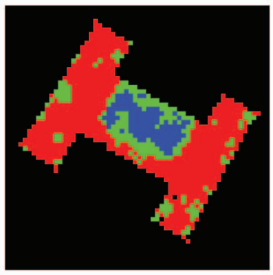
where each Gaussian density $N(x|\mu_k, \Sigma_k)$ is called a component of the mixture and has its own mean μ_k and covariance Σ_k. The parameters π_k are called mixing coefficients for which $\pi_k \geq 0$, and if the individual Gaussian components are normalized, then

$$\sum_{k=1}^{K} \pi_k = 1. \tag{16}$$

The calculation of the parameters $\pi = \{\pi_1, \ldots, \pi_k\}$, $\mu = \{\mu_1, \ldots, \mu_k\}$, and $\mathbf{\Sigma} = \{\Sigma_1, \ldots, \Sigma_k\}$ is performed using



(a)



(b)

GMM comp.	Mixing coeff.(κ)	Mean(μ)	Variance(σ^2)
G_0	0.684208	0.526408	0.0000000479
G_1	0.225278	0.538507	0.0000405911
G_2	0.0905144	0.493301	0.000859696

(c)

Fig. 11. Example classification of points using a Gaussian mixture model consisting of three components. (a) Roof cluttered with objects. (b) Point classification (G0-red, G1-green, G2-blue). (c) The values of the GMM for (a).

an expectation-maximization (EM) algorithm which minimizes the log of the likelihood function given by

$$ln \quad p(\mathbf{X}|\pi,\mu,\mathbf{\Sigma}) = \sum_{n=1}^{N} ln\left\{\sum_{k=1}^{K}\pi_k N(\mathbf{x}_n|\mu_k,\mathbf{\Sigma}_k\right\}, \qquad (17)$$

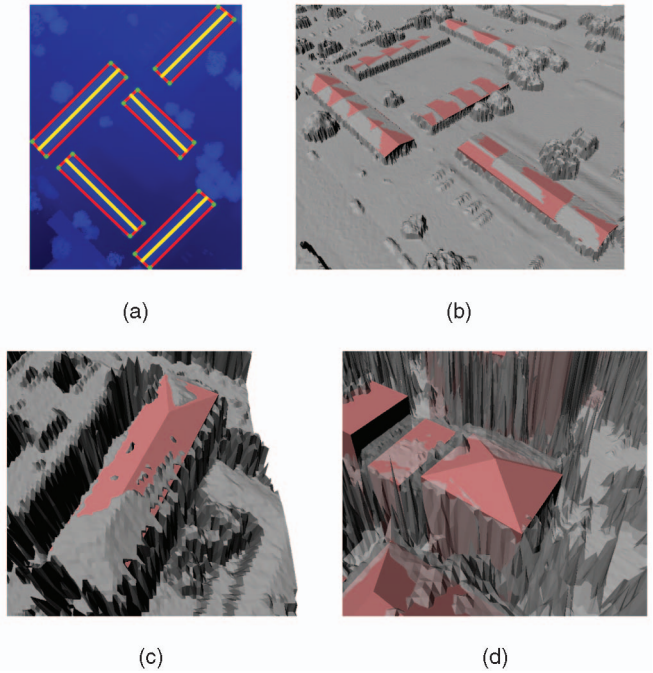
where $\mathbf{X} = \{x_1,\ldots,x_N\}$ are our data samples, i.e., elevation.

The advantage of using a GMM for the classification of the points is that it can better separate the outlier points produced by objects such as elevator engines, air conditioners, and chimneys (which are commonly found on the roofs), thus removing the otherwise significant bias of those outliers from the distribution of the true surface points lying on the roof.

Fig. 11a shows an example of a building with several objects of different elevation located on the roof. The dominant component (the component with the highest mixture coefficient of the GMM—G_0 in Fig. 11c) is used to classify all the points whose probability is maximized in this distribution as the true surface points $P_{inliers} \in P$ and everything else as outliers $P_{outliers} \in P$. Fig. 11b shows all the points inside the detected boundaries being color-coded based on the distribution that maximizes their probability. Red points have the highest probability in G_0—which is the most dominant component, green points in G_1, and blue points in G_2. Using only the inlier points, (14) now becomes

$$E_{error} = \sum_{n=0}^{5}\sum_{\forall p\in P_{inliers}} (\chi_p - f(\chi_p,\alpha,\beta))^2. \qquad (18)$$

This classification using the GMMs has the effect of making the plane fitting more robust and less susceptive in the presence of outliers. This greatly improves the performance of the optimization as well as the accuracy of the reconstructed surfaces.



(a) (b) (c) (d)

Fig. 12. Automatic roof-type identification using the parameterized geometric primitive. (a) Detected boundaries (red lines) and recovered interior boundaries (yellow lines). (b) Reconstructed building from a overlaid on original data (Gable roof-type). (c) Hip roof-type. (d) Pyramidal roof-type.

An example is shown in Fig. 12a where the detected boundaries are shown in red and the automatically recovered interior boundaries produced by the optimization are shown in yellow. Note that even for a human operator, it is hard to identify the roof-type of these buildings from Fig. 12a. The reconstructed buildings are shown in Fig. 12b, overlaid to the original data. Figs. 12c and 12d show different roof-types successfully identified by the primitive.

A saltbox roof-type is shown in Fig. 13a and the automatic identification result using the parameterized primitive is shown in Fig. 13b. Due to the symmetry constraints, the roof is divided into three planes instead of the actual two planes. However, this does not affect the visual appearance of the reconstructed model since the two lower planes shown in Fig. 13b with white arrows are coplanar. An example of a shed roof is shown in Fig. 13c and the automatically reconstructed model in Fig. 13e. Similarly, the two planes in this case are coplanar due to the symmetry constraints. In addition, in this example, any value for α, β is a global minimum since the roof surface is flat and all planes will be coplanar.

Fig. 13f shows an example of a complex twin gable roof-type. Although the primitive was not designed to handle such roof-types, the minimization successfully converges to the single gable shape. This demonstrates the robustness of the parameterized primitive and effectiveness in handling even complex cases where the roof shape significantly deviates from the shapes in Fig. 10.

In this work, we focus on identifying the most commonly occurring roof-types which can be handled entirely by the previously described primitive. However, such a parameterization can be extended to handle other

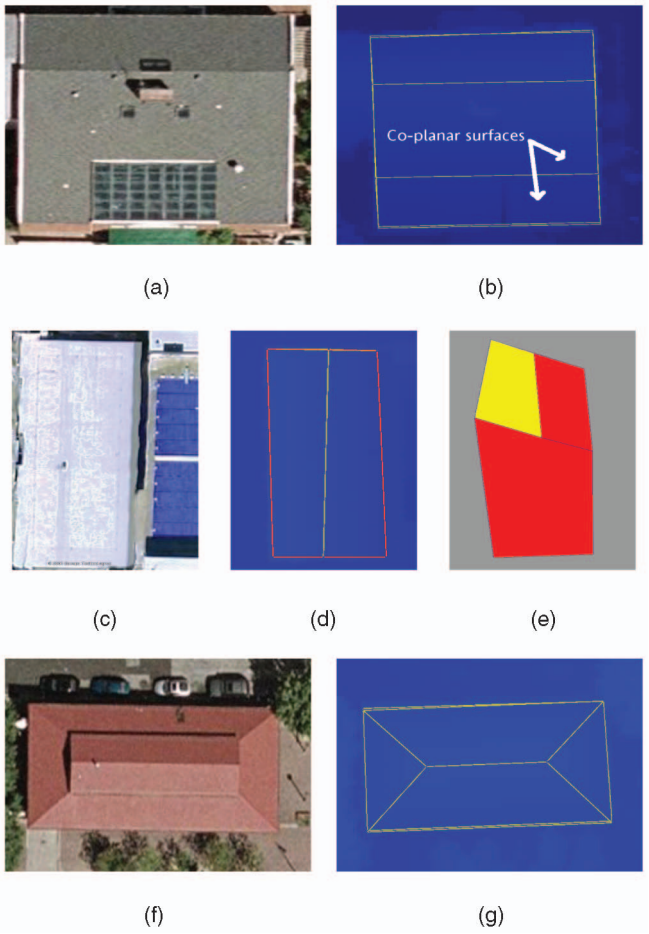


Fig. 13. Automatic roof-type identification using the parameterized geometric primitive. (a) Saltbox roof-type. (b) Automatic identification result. The lower two planes are coplanar. (c) Shed roof-type. (d) Any solution for α, β is a global minimum. The two planes are coplanar. (e) Reconstruction in 3D. The yellow plane indicates a selection. (f) Complex twin gable roof. (g) Best identification found.

groups of complex linear roof-types and basic building blocks L, H, U, and T, as described by Schmitt [18]. Fig. 14 shows a parameterized primitive which can identify all variants of L-shaped roof-types. This parameterization includes only four parameters $\alpha, \beta, \gamma, \delta$ subject to $0 \leq \alpha, \beta, \gamma, \delta \leq 1$, and it subdivides the space into six planes $\Pi_{1\ldots7}$. The external and internal control points are again expressed as a function of the four parameters $\alpha, \beta, \gamma, \delta$ and the structural parameters width w and height h as shown below. Similarly, more parameterized primitives can be created or combined in order to handle more complex groups of roof-types:

$$V_1 = \left[-\frac{w_1}{2}, -\frac{h}{2}\right], \qquad V_2 = \left[\frac{w_1}{2}, -\frac{h}{2}\right], \tag{19}$$

$$V_3 = \left[\frac{w_1}{2}, \frac{h}{2} - h_2\right], \qquad V_4 = \left[\frac{w}{2}, \frac{h}{2} - h_2\right], \tag{20}$$

$$V_5 = \left[\frac{w}{2}, \frac{h_2}{2}\right], \qquad V_6 = \left[-\frac{w_1}{2}, \frac{h}{2}\right], \tag{21}$$

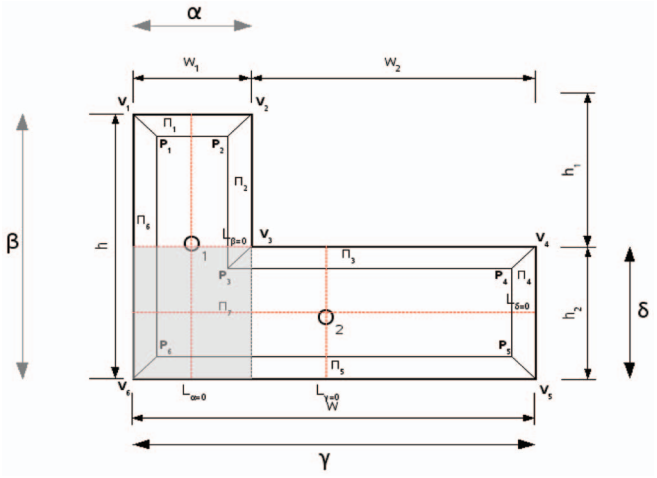


Fig. 14. A parameterized primitive for the identification of all L-shaped roof-type variants. O_1, O_2 are the local origins for the two subdivided rectangles. The red lines indicate where the parameters $\alpha, \beta, \gamma, \delta$ are equal to zero, respectively. The shaded area shows overlap between the two parameterized rectangles. The points P_3, P_6 are parameterized with α, β as well as γ, δ.

$$P_1 = \left[-\alpha\frac{w_1}{2}, -\beta\frac{h}{2}\right], \qquad P_2 = \left[\alpha\frac{w_1}{2}, -\beta\frac{h}{2}\right], \tag{22}$$

$$P_3 = \left[\alpha\frac{w_1}{2}, \beta\left(\frac{h}{2} - h_2\right)\right], \quad \text{and} \quad P_3' = \left[-\gamma\left(\frac{w}{2} - w_1\right), -\delta\frac{h_2}{2}\right], \tag{23}$$

$$P_4 = \left[\gamma\frac{w}{2}, -\delta\frac{h_2}{2}\right], \qquad P_5 = \left[\gamma\frac{w}{2}, \delta\frac{h_2}{2}\right], \tag{24}$$

$$P_6 = \left[-\alpha\frac{w_1}{2}, \beta\frac{h}{2}\right], \quad \text{and} \quad P_6' = \left[-\gamma\frac{w}{2}, \delta\frac{h_2}{2}\right]. \tag{25}$$

5.3 Polygonal Primitive

In addition to the automatic parameterized primitive, we introduce a flexible polygonal primitive for the reconstruction of complex linear surfaces and buildings. An important advantage of the polygonal primitive is that it generalizes the reconstruction such that single surfaces, i.e., parts of a roof (singular case) as well as entire buildings can be reconstructed using the same primitive.

Based on connectivity information automatically derived from the interactively defined control points, a singular or complex primitive is initialized on the fly. Linear surfaces with similar control points or sharing edges are integrated together in a single primitive. Thus, entire buildings are optimized as *single* objects with *shared* geometry (points and edges, thus reducing the number of unknowns) rather than optimizing the different roof surfaces separately, which has the significant consequence that the reconstructed polygonal models are watertight and do not contain any misalignments between neighboring surfaces.

The process is repeated as part of a coordinate-wise optimization which refines the spatial position of the control points.

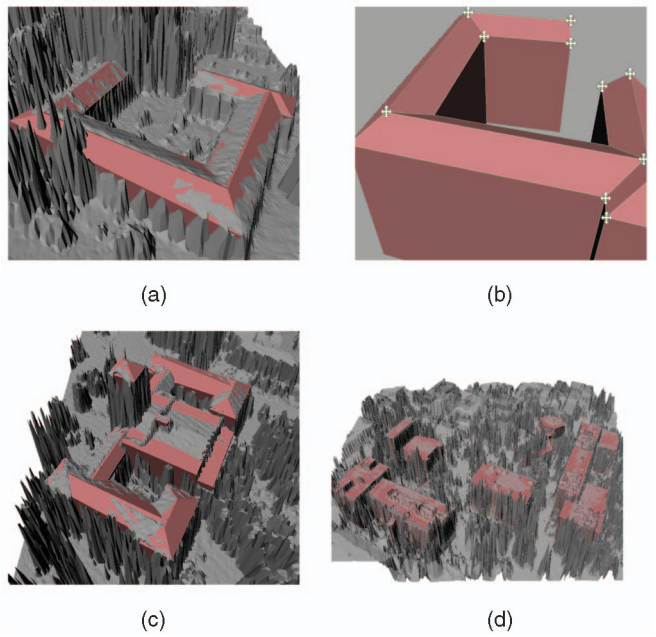


Fig. 15. Different types of buildings extracted using the polygonal primitive. (a) Complex building with sloped surfaces extracted using one complex polygonal primitive consisting of nine planar surfaces. (b) Building from (a). The optimization included all surfaces (nine) and shared the same control points shown in green. (c) Complex building extracted using four complex polygonal primitives. (d) An area containing complex buildings reconstructed using the polygonal primitive.

Fig. 15 shows examples of a variety of buildings extracted using the polygonal primitive. Fig. 15a shows the reconstructed model returned by a *single* primitive of a building containing *nine* surfaces. As explained before, the optimization minimizes the error function given by

$$E_{error} = \sum_{n=0}^{9} \sum_{\forall p \in P_{inliers}} (\chi_p - f(\chi_p, \alpha, \beta))^2 \qquad (26)$$

for all nine surfaces together. This reduces the number of parameters in the optimization which subsequently improves the computational time of the reconstruction.

The example of the complex building shown in Fig. 15c was reconstructed with four complex polygonal primitives. The tower and flat buildings were reconstructed using the parameterized primitive. Similarly, Fig. 15d shows an area reconstructed using singular and complex polygonal primitives.

5.4 Semiautomatic Nonlinear Roof-Type Reconstruction

The reconstruction of 3D models is performed using the two linear primitives (parameterized primitive and the polygonal primitive), as previously explained, to produce a watertight, lightweight polygonal geometric model of the scene. Despite the fact that the two linear primitives can model most of the buildings found in an urban area, they are limited to handling linear surfaces and cannot be used for the identification of complex buildings containing nonlinear surfaces such as domes, stadiums, etc. Therefore, an interactive semiautomatic approach is employed for the identification and reconstruction of such complex buildings.

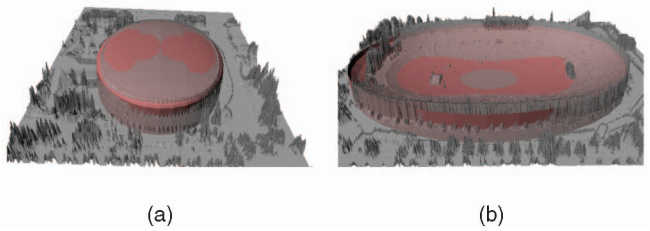


Fig. 16. Reconstruction of nonlinear structures. (a) Dome-like structure. (b) Stadium-like structure.

An ellipsoidal primitive is designed to handle all types of nonlinear surfaces either dome-like or stadium-like (hollow) by fitting an ellipsoid to the data. The primitive is initialized with the centroid and two perpendicular points on the surface's perimeter used to determine α and β in (27). A nonlinear optimization is then performed to recover the optimal value for the single unknown variable γ such that (27) is minimized as

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\beta^2} + \frac{z^2}{\gamma^2} = 0. \qquad (27)$$

Fig. 16a shows an arena with a dome-like nonlinear roof reconstructed using this primitive. Similarly, Fig. 16b shows a stadium-like structure being overlaid on the original data. The system automatically determines what type of surface is being reconstructed and automatically chooses the orientation of the surface type, i.e., dome or stadium-like.

6 Photorealistic Texture Generation

The generation of photorealistic textures is performed in two steps: registration of imagery to the reconstructed model and texture composition.

6.1 Registration

The goal of the registration is to recover the camera projection matrix which is expressed in terms of the intrinsic and extrinsic parameters of the camera

$$C = \underbrace{\begin{bmatrix} \alpha & -\alpha cot(\theta) & u_0 \\ 0 & \frac{\beta}{sin(\theta)} & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{intrinsic} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{extrinsic}, \qquad (28)$$

where $\alpha = kf_x$ and $\beta = kf_y$, (f_x, f_y) are the focal length on the x- and y-axis, respectively, θ is the skew angle, u_0, v_0 is the principal point on the x and y-axis, respectively, and r_{1-3}, t_{1-3} determine the camera's rotation and translation relative to the world.

Various methods have already been proposed for the estimation of these parameters. However, when dealing with images from multiple sensors, it is often required to use various methods for the recovery of the matrix C depending on the type and characteristics of the images.

6.1.1 Registration of Ground Images

The camera model of (28) is used to recover the camera pose for ground images. A set of user-specified correspondences

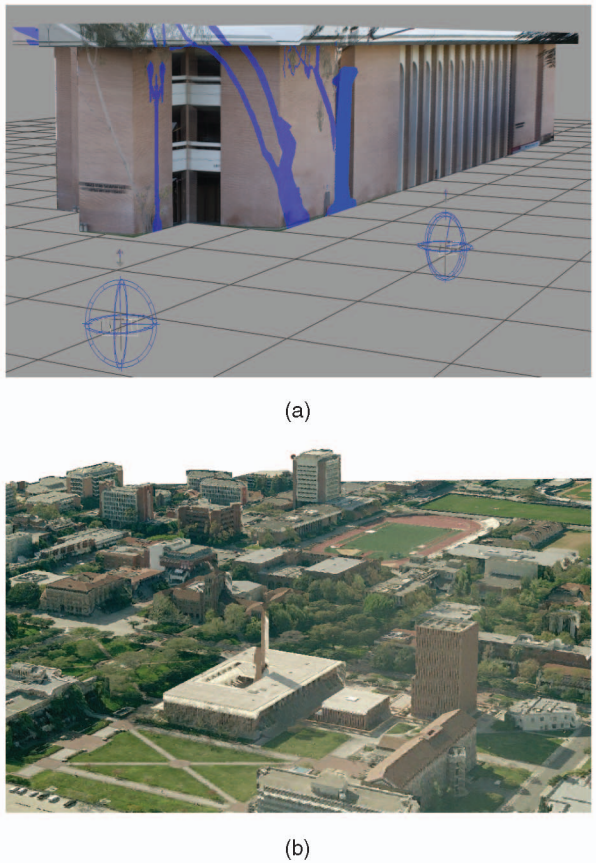


Fig. 17. Registration of ground, aerial, and satellite. (a) Registration of two ground images. The blue areas indicate occluded areas which will be recovered by the texture composition. (b) Camera pose recovery of aerial image. Projective texture maping is used to project the image on the reconstructed models for visual verification.

between the images and the reconstructed model is required, and the camera parameters are calculated using a linear least-squares method. In cases where there are not enough correspondences between the image and the model, vanishing points are used for estimating the parameters. A bundle adjustment optimization finally refines the camera parameters, thus improving the overall accuracy of the alignment. Fig. 17a shows ground images being aligned to a reconstructed model of a building. The blue marks are areas of unwanted textures (trees and light-post) which are discussed later.

6.1.2 Registration of Satellite and Aerial Images

Satellite and aerial images have weak perspective and are often produced by stitching together several perspective images. The camera model of (28) cannot be used to deal with multiperspective images and will most definitely fail. However, this type of images can be rectified by using vanishing points to recover the extrinsic parameters of the camera (rotation and translation). Additional user specified correspondences can then be used to compute a homography between the rectified image and the model. Fig. 17b shows the registration of an aerial image. The model is then back-projected (yellow lines) in the image for visual verification.

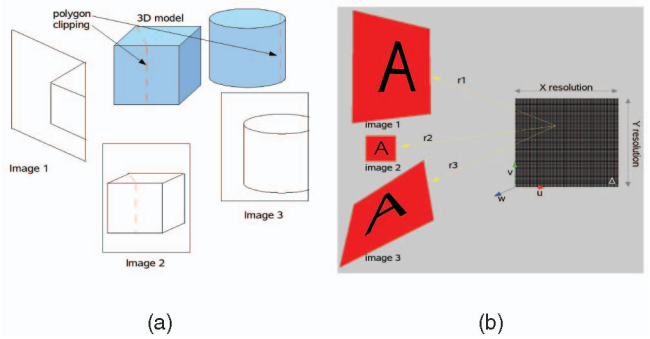


Fig. 18. Visibility clipping and texture map resolution computation. (a) Visibility-based clipping and (b) texture map resolution.

6.2 Texture Composition

The texture composition is performed by integrating the appearance information of all registered images to create view-independent, seamless textures. A key aspect of this integration is the efficient and effective recovery of texture information for occluded and missing areas. The generation of the composite textures involves three steps: scene preprocessing, computation of the texture map resolution, and rendering and packing.

6.2.1 Scene Preprocessing

A view-dependent-based subdivision is first applied to the model to ensure that all surfaces are *entirely* visible in all images. For each image, a visibility check is performed and the following actions are taken:

1. A surface which is entirely visible in the image remains unchanged.
2. A partially visible surface is clipped at the boundary of the projection of the image plane.
3. A nonvisible (or backfacing) surface remains unchanged.

Fig. 18a shows an example of the visibility clipping. The cube is clipped at the projected boundary of the first image plane, and similarly, the cylinder is clipped at the projected boundary of the third image. This ensures that all surfaces are entirely visible in all images, which is imperative for the correct computation of the texture map resolutions.

6.2.2 Texture Map Resolution

A surface may appear in multiple images with different resolutions. To ensure minimal information loss and sharp textures, the resolution of a texture map is chosen to be the size of the projected polygon with the largest area in image space. Fig. 18b shows each surface (red polygons) projected into three image planes. The dimensions of the polygon with the largest area are chosen as the resolution of the texture map for the surface. Surfaces which are backfacing or are not visible in any of the images are automatically assigned a default color.

6.2.3 Rendering and Texture Packing

The composite textures are then rendered using ray tracing. For each point on the surface, a ray is cast to the image planes. A point on the surface is visible in an image if there is no other surface intersecting the ray casted from the point

Fig. 19. Recovery of occluded areas. Areas which are not visible by any camera and therefore no texture information is available appear as black.

to the image plane. If the point is visible in an image, then the corresponding pixel color is retrieved and weighted based on the following criteria:

1. The distance of the pixel to the image boundary: Pixels which are close to the edges receive a lower weight than pixels located in the middle. This is required in order to hide any stitching effects between images, and reduce the artifacts introduced when blending multiple images.
2. The angle between the camera's direction and the surface's normal: Images taken from oblique angles are down-weighted since they have a higher degree of perspective distortion.
3. The distance of the pixel from the principal point: Pixels which are further away from the principal point exhibit higher radial distortion. Although the radial distortion coefficients can be closely approximated and used to undistort the image, we have found that it is more likely to have misalignments between images at points further away from the principal point.
4. The resolution of the image: High-resolution images capture higher level of detail than low-resolution images; therefore, they are preferred.

The composite texture maps are finally sorted based on their resolution and are packed into a texture atlas. The process transforms the texture space of each map, thus requiring the recalculation of the texture coordinates for the models in the scene.

As previously mentioned, the rendering pipeline has the significant advantage that multiple information is integrated for the generation of the textures, thus allowing the efficient and correct handling of occluded and missing areas. An example was shown Fig. 17a where the occluded areas were indicated with a blue color. The composite textures produced for that building are shown in Fig. 19. The blue masked areas which indicated occluded areas due to the presence of a tree and a light-post are successfully recovered.

Another example of a textured building is shown in Fig. 20a. The composite textures were generated using a set of three ground images. In this case, building details such as the window intrusions are not modeled, thus causing perspective distortion effects, as shown in Fig. 20b. These effects become even more dramatic as the angle between the camera viewing direction and surface orientation increases. This is the main reason for the use of the second criterion during the weight computation.

(a) (b)

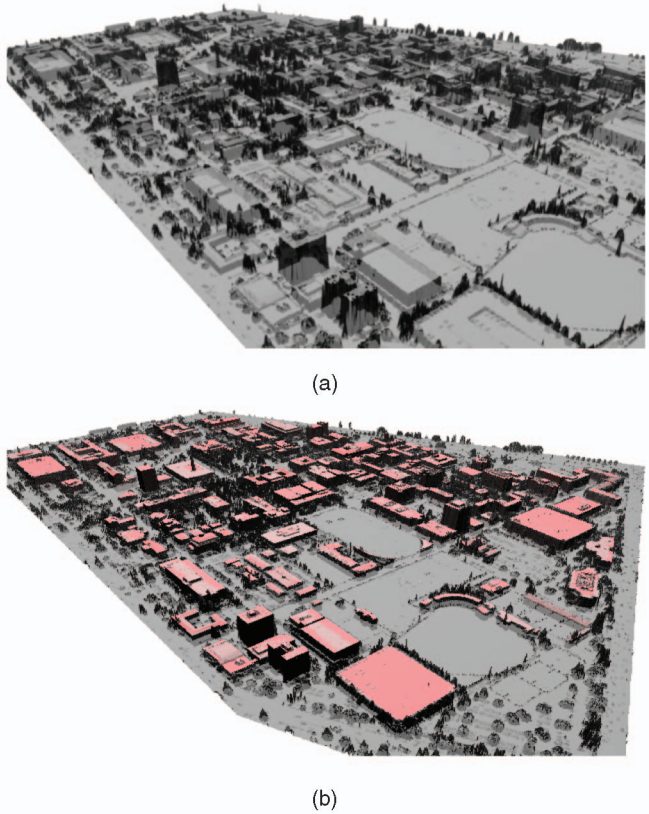
Fig. 20. The composite textures of this building model were generated using three high-resolution ground images. (a) Novel view point render. (b) Perspective distortion effects due to unmodeled geometry.

7 Experimental Results

Fig. 21 shows a large-scale virtual environment created with the proposed approach. The models in Fig. 21b were reconstructed from the original data in Fig. 21a. Fig. 22a shows the model textured with high-resolution (4 K) satellite imagery, and Fig. 22b shows an area in the scene with textures generated from satellite, aerial, and ground imagery. A novel viewpoint render of the complete environment using satellite imagery is shown in Fig. 23.

8 Evaluation

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



(a)

(b)

Fig. 21. Reconstructed virtual environment. (a) Original LiDAR data. (b) Reconstructed models of USC campus overlaid on original data.

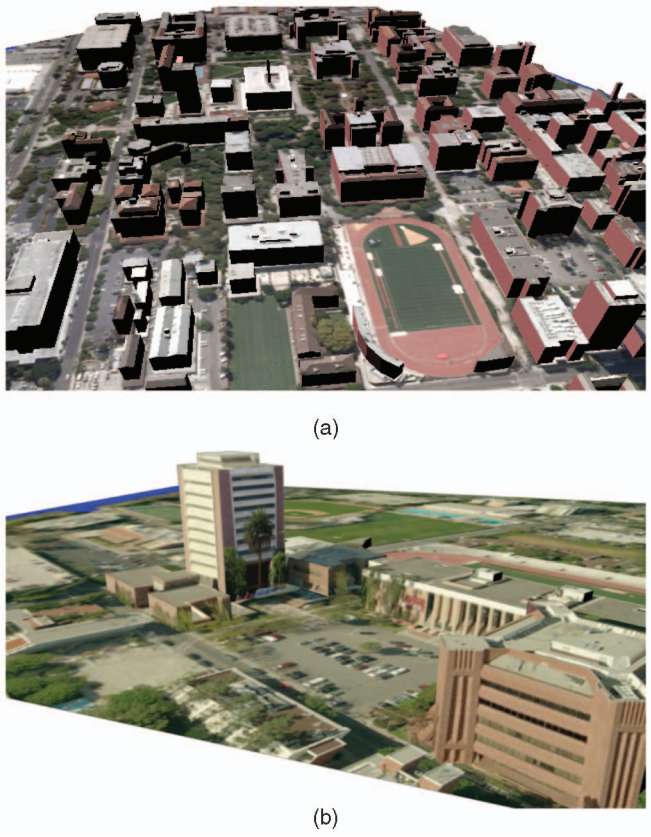


Fig. 22. Textured virtual environment. (a) Entire area textured with high-resolution (4,000) satellite imagery. (b) A reconstructed area with textures from satellite, aerial and ground imagery.

and quantitative evaluation of the reconstructed models in terms of the following parameters:

1. *Model* accuracy: The accuracy is measured by overlaying the reconstructed models on the original LiDAR data (Fig. 24a). We quantitatively evaluate the models by measuring the deviation of the fitted surfaces to the actual data. In addition, a qualitative evaluation is also performed by visually inspecting the model for any artifacts such as misalignments between neighboring surfaces and inconsistencies between the surfaces and the original data.
2. *Realistic representation and level of detail (Fig. 24b)*: We employ georeferencing for the evaluation of the realistic representation and the level of detail captured by our building models. Satellite, aerial, and ground imagery linked to geospatial locations are projected on the reconstructed model. The deviation of the back-projected 3D features from the 2D features in the images (measured as RMS) provides an estimate of the accuracy of the representation of the reconstructed models and the level of detail captured by the model. A noticeable shortcoming of our models is the lack of facade details and the failure to accurately represent the geometric structures such as window extrutions, indentations, etc. This is due to the fact that the models are derived from airborne LiDAR which cannot record any information about structures located perpendicular to the scanning direction such as buildings' facades.
3. *Scalability*: The scalability is measures in terms of the average time required for the reconstruction of the

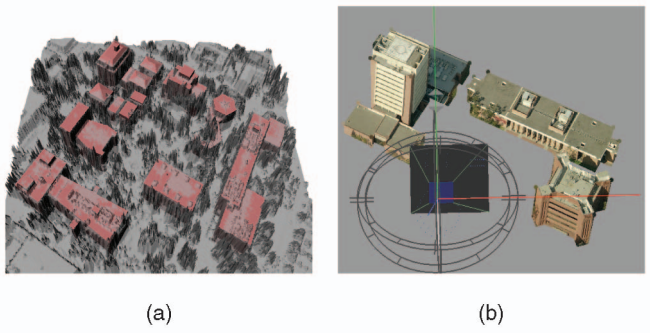


Fig. 24. (a) Evaluation of model accuracy. Model overlaid on original data. (b) Evaluation of realistic representation and level of detail. Aerial image projected on reconstructed building models.

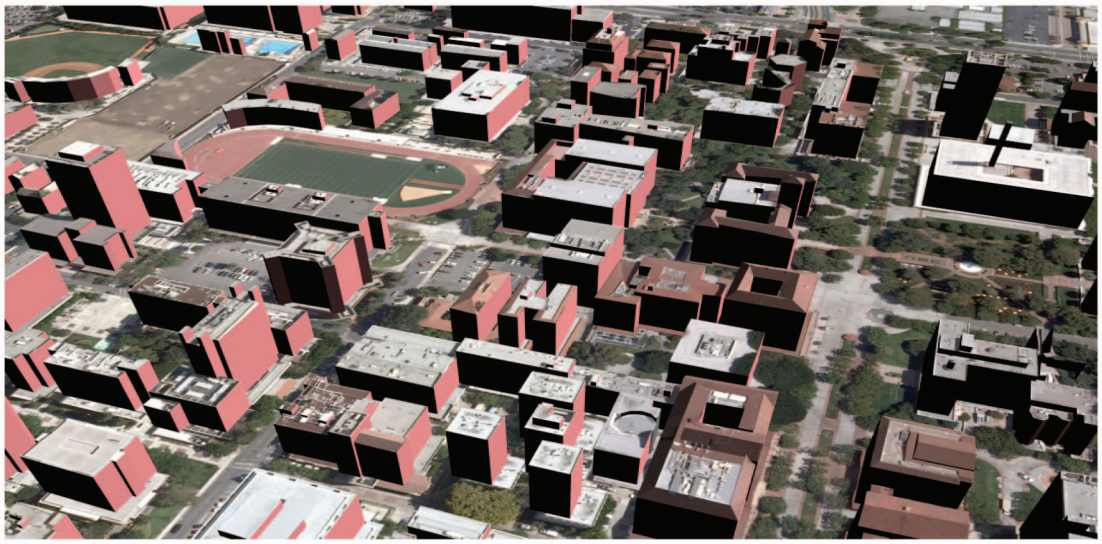


Fig. 23. A large-scale virtual environment rapidly created using the proposed approach.

TABLE 2
Scalability: Processing Time Required for the Reconstruction of Average-Sized Building Models

Task→ Primitive↓	Detection	Identification & Reconstruction	Total
Parameterized	1s	1s-5s	2s-6s
Polygonal	interactive(clicks)	1s-43s	$\simeq$ 45s
Ellipsoidal	interactive(clicks)	4s	$\simeq$ 6s

building models. Our experiments show that the required time is based on the complexity of the buildings and the primitives employed. Table 2 indicates the time required for the reconstruction of average-sized buildings, i.e., occupying an area of up to 1,000 pixels. The parameterized primitive refers to the flat, shed, gable, hip, pyramidal, mansard, and saltbox roof-types.

9 Conclusion

We have presented a novel and complete approach for the rapid creation of photorealistic large-scale virtual environments.

First, we resolved the 3D model reconstruction problem using a novel parameterized geometric primitive for the automatic identification and reconstruction of building models. This primitive significantly reduces the number of user interaction and increases the computational speed of the reconstruction process by exploiting common symmetry constraints found in man-made structures. In addition, buildings containing complex linear and nonlinear surfaces are interactively reconstructed using a linear polygonal and nonlinear primitive, respectively.

Second, we resolved the problem of texturing and presented a rendering pipeline for the composition of photorealistic textures which allows for the effective handling of missing or occluded areas. The result is a set of view-independent, seamless textures which are composited from images captured from multiple sensors (ground, aerial, and satellite).

Appendix: Graph Cuts

In [3], [2], the authors interpret image segmentation as a graph partition problem. Given an input image I, an undirected graph $G = <V, E>$ is created, where each vertex $v_i \in V$ corresponds to a pixel $p_i \in I$ and each undirected edge $e_{i,j} \in E$ represents a link between neighboring pixels $p_i, p_j \in I$. In addition, two distinguished vertices called *terminals* V_s, V_t, are added to the graph G. An additional edge is also created connecting every pixel $p_i \in I$ and the two *terminal* vertices e_{i,V_s} and e_{i,V_t}. For weighted graphs, every edge $e \in E$ has an associated weight w_e.

A *cut* $C \subset E$ is a partition of the vertices V of the graph G into two disjoint sets S, T, where $V_s \in S$ and $V_t \in T$. The cost of each cut C is the sum of the weighted edges $e \in C$ and is given by

$$|C| = \sum_{\forall e \in C} w_e. \tag{29}$$

The minimum cut problem can then be defined as finding the cut with the minimum cost. An algorithm for solving this problem has been proven to require polynomial time [3].

Energy minimization function. Finding the minimum cut of a graph is equivalent to finding an optimal labeling $f : I \longrightarrow L$ which assigns a label $l \in L$ to each pixel $p \in I$, and f is piecewise smooth and consistent with the original data. The energy function is then given by

$$E(f) = E_{data}(f) + \lambda * E_{smooth}(f), \tag{30}$$

where λ is the weight of the smoothness term.

Energy data term. The data term in (30) measures the cost of relabeling the original data with a new labeling f. It is defined as the sum of the per-pixel measure (D_p) of how appropriate each label $f_p \longrightarrow l \in L$ is for each pixel $p \in I$ in the original data and is given by

$$E_{data}(f) = \sum_{p \in I} D_p(f_p). \tag{31}$$

Energy smoothness term. The smoothness term in (30) measures the cost of relabeling neighboring pixels with a new labeling f. It is defined as the sum of the differences between two neighboring pixels $p, q \in I$ under a labeling $f_p \longrightarrow l_p \in L$ and $f_q \longrightarrow l_q \in L$, respectively, and is given by

$$E_{smooth}(f) = \sum_{\{p,q\} \in N} V_{\{p,q\}}(f_p, f_q), \tag{32}$$

where N is the set of neighboring pixels and $V_{\{p,q\}}$ measures the difference between the neighboring pixels, also known as the interaction potential function.

Acknowledgment

The authors would like to thank the Airborn1, Inc., for providing them with the University of Southern California (USC) campus LiDAR data. They acknowledge the members and Professor Ulrich Neumann in the Computer Graphics and Immersive and Technologies (CGIT) laboratory of USC. They also thank the reviewers for their valuable comments and suggestions.

References

[1] M. Bartels, H. Wei, and D.C. Mason, "DTM Generation from LIDAR Data Using Skewness Balancing," *Proc. Int'l Conf. Pattern Recognition (ICPR '06)*, pp. 566-569. 2006.

[2] Y. Boykov and M.-P. Jolly, "Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images," *Proc. Int'l Conf. Computer Vision (ICCV '01)*, pp. 105-112, 2001.

[3] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," *Proc. Int'l Conf. Computer Vision (ICCV '99)*, pp. 377-384, 1999.

[4] P. Debevec, C. Tchou, A. Gardner, T. Hawkins, C. Poullis, J. Stumpfel, A. Jones, N. Yun, P. Einarsson, T. Lundgren, M. Fajardo, and P. Martinez, "Estimating Surface Reflectance Properties of a Complex Scene under Captured Natural Illumination," Technical Report, Univ. of Southern California, ICT, 2004.

[5] P.E. Debevec, C.J. Taylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach," *Proc. ACM Conf. Computer Graphics*, pp. 11-20, Aug. 1996.

[6] R.O. Duda and P.E. Hart, *Pattern Classification*. John Wiley and Sons, 2000.

[7] C. Früh and A. Zakhor, "Constructing 3D City Models by Merging Ground-Based and Airborne Views," *Computer Vision and Pattern Recognition,* pp. 562-569, 2003.
[8] J. Hu, S. You, and U. Neumann, "Approaches to Large-Scale Urban Modeling," *IEEE Computer Graphics and Applications,* vol. 23, no. 6, pp. 62-69, 2003.
[9] S.C. Lee, S.K. Jung, and R. Nevatia, "Automatic Pose Estimation of Complex 3D Building Models," *Proc. IEEE Workshop Application of Computer Vision (WACV '02),* pp. 148-152, 2002.
[10] M. Levoy and P. Hanrahan, "Light Field Rendering," *Proc. ACM SIGGRAPH '96,* pp. 31-42, 1996.
[11] A.R. Martinez and G. Drettakis, "View-Dependent Layered Projective Texture Maps," *Proc. Pacific Conf. Computer Graphics and Applications,* pp. 492-496, 2003.
[12] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.M. Frahm, R.G. Yang, D. Nister, and M. Pollefeys, "Real-Time Visibility-Based Fusion of Depth Maps," *Proc. Int'l Conf. Computer Vision,* pp. 1-8, 2007.
[13] R. Nevatia and K.E. Price, "Automatic and Interactive Modeling of Buildings in Urban Environments from Aerial Images," *Proc. Int'l Conf. Image Processing (ICIP '02),* vol. 3, pp. 525-528, 2002.
[14] M. Pollefeys, L.J.V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual Modeling with a Hand-Held Camera," *Int'l J. Computer Vision,* vol. 59, no. 3, pp. 207-232, 2004.
[15] C. Poullis, A. Gardner, and P. Debevec, "Photogrammetric Modeling and Image-Based Rendering for Rapid Virtual Environment Creation," *Proc. Army Science Conf.,* pp. I: 1-7, 2004.
[16] C. Poullis, S. You, and U. Neumann, "Rapid Creation of Large-Scale Photorealistic Virtual Environments," *Proc. IEEE Virtual Reality Conf.,* pp. 153-160, 2008.
[17] D.P. Robertson and R. Cipolla, "Building Architectural Models from Many Views Using Map Constraints," *Lecture Notes in Computer Science,* vol. 2351, pp. 155-163, 2002.
[18] G. Schmitt, *Architectura et Machina.* Vieweg & Sohn, 1993.
[19] G. Turk, "Texture Synthesis on Surfaces," *Proc. ACM SIGGRAPH '01,* pp. 347-354, 2001.
[20] S. You, J. Hu, U. Neumann, and P. Fox, "Urban Site Modeling from liDAR," *Proc. Int'l Conf. Computational Science and Applications,* 2003.
[21] C. Zach, T. Pock, and H. Bischof, "A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration," *Proc. Int'l Conf. Computer Vision,* pp. 1-8, 2007.

Charalambos Poullis received the BSc degree in computing information systems from the University of Manchester, UK, in 2001, and the MSc degree in 2003 in computer science with specialization in multimedia and creative technologies from the University of Southern California (USC), where he is currently working toward the PhD degree in computer science. His area of expertise is computer vision, computer graphics, virtual reality, and in particular large-scale modeling, photorealistic rendering, and 3D visualization methodologies. He is currently working as a researcher at the Computer Graphics and Immersive Technologies Laboratory at USC and a teaching assistant at the Computer Science Department for graduate classes. He is a student member of the ACM and the IEEE, and has served as a reviewer in several conferences and journals.

Suya You received the PhD degree from Huazhong University of Science and Technology, China, in 1994. He is a research assistant professor in the Computer Science Department, University of Southern California (USC). His expertise is in the fundamental and applied aspects of digital media processing and applications. He also holds research positions at the Integrated Media Systems Center (IMSC), a US National Science Foundation (NSF) Engineering Research Center (ERC) at USC, and the Center for Interactive Smart Oilfield Technologies (CISOFT), a USC-Chevron Center of Excellence for Research and Academic Training on Interactive Smart Oilfield Technologies. His current research focuses on the mobile augmented reality, large-scale scene modeling and visualization, game technique for simulation and training, and multisensor data fusion for remote operations. He is the author or coauthor of more than 100 papers, and a coholder of several patents and technology disclosures in these areas. He is a member of the IEEE.