Integrating LiDAR, Aerial Image and Ground Images for Complete Urban Building Modeling

Jinhui Hu, Suya You, Ulrich Neumann
University of Southern California
{jinhuihu,suyay, uneumann}@graphics.usc.edu

Abstract

This paper presents a hybrid modeling system that fuses LiDAR data, an aerial image and ground view images for rapid creation of accurate building models. Outlines for complex building shapes are interactively extracted from a high-resolution aerial image, surface information is automatically fit with a primitive based method from LiDAR data, and high-resolution ground view images are integrated into the model to generate fully textured CAD models. Our method benefits from the merit of each dataset, and evaluation results are presented on a university campus-size model.

1. Introduction

Urban models are used widely for development planning as well as climate, air quality, fire propagation, and public safety studies. In most of these cases the models of buildings are the primary features of interest. Although urban models are useful in many fields, the creation of detailed large-scale models remains at best a difficult and time-consuming task [2,14].

A variety of sensing and modeling technologies have been used to create detailed building models from imagery or from laser sensing data. Photogrammetry offers a cost-effective means to obtain large-scale urban models [7,9]. Recently, active sensors, such as LiDAR (Light Detection and Ranging)[15], have become an important information source for generating high quality 3D digital surface models.

Different sensors provide various data for building modeling, however, each of these data sources has its own advantages and drawbacks, so it is unlikely to create accurate and detailed models with a single sensor. Aerial images provide detailed texture and color information in high-resolution, making them necessary for texture data and appealing for extracting detailed model features. However, reconstruction from stereo aerial images only lead to sparse points, which makes them unsuitable for reconstruction of complex surfaces, such as curved surfaces and roofs with slopes. On the other hand, LiDAR data samples are dense in surface points and they directly measure model heights with accuracy up to centimeters. However, edges from LiDAR data are jaggy due to the relatively low sample rate (usually one meter) of the sensor. Ground images offer high-resolution texture information and details of model façades, but lack roof and global information. A natural conclusion is to combine the geometry, photometry, and other sensing sources to compensate for the shortcomings of each sensing technology, and fuse these data sources to obtain more detailed, accurate and fully textured models for complex urban buildings.

In this paper, we present a hybrid modeling system based on the LiDAR modeling system [15]. The hybrid system combines LiDAR and a single aerial imagery for rapid creation of accurate building models,

Figure 1.From left to right. (a) an input aerial image; (b) input LiDAR data; (c) the hybrid modeling result.

and enhances these models with ground view images to create fully textured models. We first interactively extract outlines (including curves) for complex building shapes from a high-resolution aerial image (Figure 1. (a)); then automatically fit parameterized surface information with a primitive based method from LiDAR data (Figure 1. (b)); finally we integrate ground view images with automated pose recovery to generate fully textured CAD models (Figure 1. (c)).

Our method benefits from the merit of each dataset, and creates accurate complex photorealistic models with minor user interactions, thus suitable for large-scale urban modeling. The hybrid modeling method is superior to techniques based on a single dataset. Compared with results from LiDAR [15] only, the hybrid model has accurate edge information and more building details. Compared with results from stereo aerial images, the hybrid model has detailed surface information (such as slope and curved surface), and the height is more accurate thanks to the accuracy of LiDAR data. Our system creates the complex building compound (Figure 1. (c)) in only a few minutes.

1.1. Previous Work

A wealth of research has been conducted to create urban building models. According to the data acquisition method, we classify the different modeling methods into those based on photogrammetry, active sensors, and hybrid sensor systems.

Pthotogrammetry methods include those using ground imagery [9] and aerial images. A successful system employing ground view images is the facade system [5]. This interactive system allows users to recover a basic geometric model of the photographed scenes. Users select edge features to help the system build a model from 3D primitives; the model is then projected back into the original image to verify accuracy, and view dependent texture mapping is employed to render photorealistic novel views. Marc Pollefeys et al. [13] use a hand-held camera to capture videos of natural scenes, and automatically reconstruct detailed realistic models. Both systems present very impressive results, however, the modeling results are local due to the physical limitation of the input data source.

Aerial images can provide accurate building footprints and global information. Extensive research has been done on aerial images for urban modeling. Methods based on aerial images vary from a single image using shape from shadow technique [10] to multiple images using stereo techniques [12]. Automatic methods often generate simple shapes. On the other hand, commercial software, such as Cybercity [18], reconstructs complex surfaces from stereo aerial images. However, the user interactions are intensive. The results from aerial images are often sparse 3D points, making them inefficient for slope and curved surface modeling.

Active sensors directly measure the surface points, which makes them ideal for surface reconstruction. Ahmed [1] uses a 2D parameter space for plane detection to extract building roofs. Morgan [11] re-samples the irregular-spaced LIDAR data into a regular grid and then uses least square fitting to locate roof planes. However, the results generally lack building edge details and façade textures.

Recently, hybrid methods have become a new trend for urban modeling. Ribarsky et al. [14] combine aerial images and ground images to generate complete models. Although their model is fully textured, the geometry is relatively simple. Fruh etc. [6] present a system using a truck equipped with one camera and two 2D laser scanners to capture the façade texture and geometry data, and an aerial image is used to globally register the laser scanner data. They present detailed modeling results of the whole Berkeley town, however, the models have millions of triangle meshes, which are hard to render. Furthermore, curved roofs cannot be modeled since only ground laser scanners are used. A number of other hybrid methods employing aerial images, 2D GIS data and LiDAR are surveyed in [2]. However, these methods often generate simple shapes.

2. System Overview

Our modeling system (Figure 2) is a hybrid system that employs data from three resources: an aerial image, LiDAR point cloud, and ground images (shown in yellow rectangles). There are three main steps in the modeling system. In the first step, we interactively extract outlines from a high-resolution aerial image, then map it to the LiDAR data, and automatically extract surface information with primitive based fitting techniques. We then combine the two data sets for full

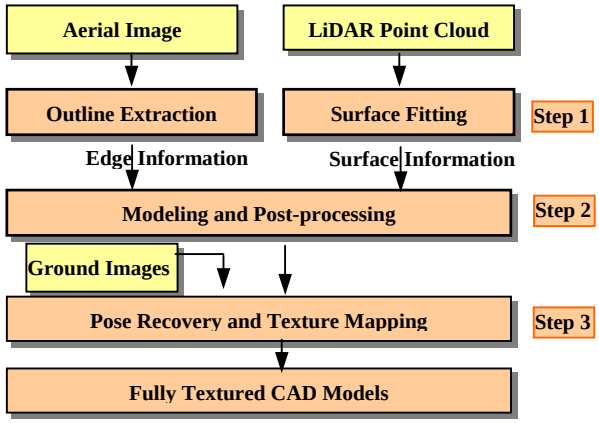


Figure 2. The hybrid modeling system.

model reconstruction in the second step, and the models are post-processed. In the third step, poses are automatically recovered for ground view images, and textures are generated and mapped to the refined models. Details of step 1 and 2 are described in Section 3, and Section 4 discusses step 3.

One key feature of our modeling system is that it is a hierarchical primitive based system. Based on the observation that man-made buildings are generally of regular shapes, we classify a building section into one of several groups. For each group we define a set of appropriate geometry primitives, including standard CG primitives and high-order surface primitives. The primitives are used both in the outline extraction and surface fitting process, which allows users to create a complex building model composed of different geometric primitives. Another key feature is that it is a hybrid modeling system, hence it benefits from the advantage of each dataset. Our approach has demonstrated its flexibility and capability for a wide range of complex buildings with irregular shapes.

3.Hybrid Modeling with LiDAR and Aerial Image

3.1 Registration

We need to register the three datasets to fuse different information into an integrated framework. The registration of the aerial image to LiDAR is presented in this subsection, while the registration of ground images to the model is discussed in Section 4.

To register the aerial image to the LiDAR data, we opt to use a rectification step followed by an affine registration procedure. Since our goal of using the aerial image is to extract footprints for buildings, it is necessary to estimate the pose of the aerial image and rectify it to create an orthographic image. The rectified image can then be registered to LiDAR using affine transformation.

Because the aerial image (Figure 1. (a)) is downloaded from the Internet [17], we do not have access to the camera's internal parameters, so vanishing points [4] are used to estimate the camera's internal parameter and orientation. The aerial image only offers the vanishing points of x-y directions, while the vanishing point of z direction is inferred by assuming that the principle point is at the camera's center [4]. We automatically detect parallel lines (Figure 3. (a) shows part of the lines), and estimate the camera's focal length and orientation using the three vanishing points. Since the aerial image is taken far from the ground, the difference of the depth of buildings can be ignored. We project it onto a plane with the recovered camera pose to generate the rectified image (Figure 3. (b)).

Figure 3. (a) Line detection, (b) image rectification.

An affine model (Equation 1) is used to register the rectified aerial image to the 2D range image generated from LiDAR data [15]. We manually selected 10 pair of point correspondences to find the registration transform between the range and aerial image. Although full perspective transformation is possible to register the original aerial image to the 3D LiDAR data, experiments show that it fails for aerial images close to orthographic, which causes singular cases while using SVD to compute the pose.

$$\begin{bmatrix} I_{x,range} \\ I_{y,range} \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} I_{x,aerial} \\ I_{y,aerial} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (1)$$

3.2 Outline Extraction from Aerial Image

Both automatic and interactive methods have been attempted to extract the outlines from the aerial image. First we try to automatically extract lines from the original aerial image to avoid noise introduced by the rectification process. Edges are detected using Canny edge detector and lines are extracted using Hough Transform as a preprocess step. These lines are then filtered with a user defined length threshold and a slope threshold of the global building orientation. We then cluster the lines into two groups, x and y directions, based on their slopes (Figure 3. (a)).

Automatic method works well for simple shapes, however, complex buildings are more difficult to analyze. As shown in Figure 3.(a), many building outlines are missing, long lines tend to break into short pieces, curves are very hard to extract, and shadows and textures introduce unnecessary lines. Although performance can be improved by using cues from range images, it often fails for complex shapes such as those addressed in this paper (Figure 1.(c)).

Due to the difficulty of automated method, we leverage the situation to use a primitive based method with user interaction to extract every outline for buildings with curved surfaces. A number of primitives are designed to represent the outlines of building shapes, which are classified into linear primitive and nonlinear primitives. Linear primitive models include:

a b c

Figure 4. Outline extraction. Yellow dots: user clicks.

plane, cube, wedge etc, while non-linear primitives include quadric and high order curves.

Two principles are enforced in designing our algorithms for outline extraction, thereby minimizing the amount of user interaction and improving accuracy. To minimize the user interaction, we allow the user to click only a few points for each primitive rather than every vertex. For a cube primitive (Figure 4. (a)), only two user clicks along the diagonal are necessary since the other two vertices can be inferred given the building's orientation. For a roof with four sloped surfaces (Figure 4.(b)), only four rather than six input points are necessary. To improve accuracy, we automatically find the building corner close to the user click. If there are no detected corners within a distance threshold, the user click is used instead. For curves (Figure 4. (c)), we allow the user to click a few control points (generally less than 10), then a quadric function is used to fit a curve to the input data points, and the curve is discretized with the user defined resolution.

Our primitive based outline extraction method has many advantages over the manual method. As shown in Figure 5 top row, the extracted outlines keep the regularity of buildings, such as parallelism and orthogonality, which is true for man-made objects. On the other hand, the manual method (Figure 5. bottom row) tends to extract slanted outlines, and the results depend entirely on the user click, which causes co-linear lines to be non-co-linear and orthogonal lines to be non-orthogonal. For the complex building with 20 primitives, only 52 user clicks are needed, while the manual method needs 134 clicks (a 61% reduction). Our method also extracts curves and discretizes them according to predefined resolution, while the manual method can only approximate them with polygons.

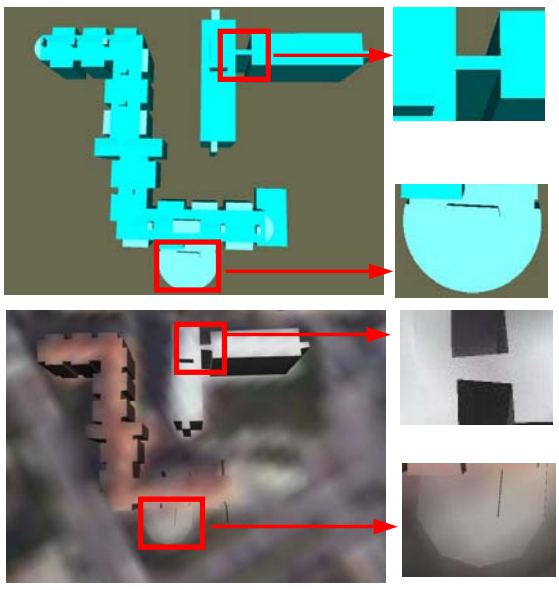


Figure 5.Outlines comparison.

3.3 Surface Extraction from LiDAR

LIDAR data samples are dense in surface points and directly measure building heights with high accuracy. We automatically reconstruct the 3D surface models using a hole-filling process followed by tessellation. The reconstructed models from LiDAR generally have jagged edges and bumpy surfaces (Figure 1.(b)), and the whole model for a University campus has more than 4 million triangle meshes. A model refining method is necessary to reduce the number of triangles and improve the visual effects.

We present a primitive based fitting technique to extract surfaces from the reconstructed LiDAR models. According to the shapes of building rooftops (flat-roof, slope-roof, dome-roof etc.), a set of geometric primitives are defined. They are linear fitting primitives: plane, cube, wedge, etc, and nonlinear fitting primitive: superquadrics. Each of the primitives is represented using a number of parameters. Equation 2 shows the function of superquadrics, where ε_1 and ε_2 defines the shape of superquadric, a_1, a_2 and a_3 defines the size of superquadric along *x*, *y* and *z* direction.

$$r(\eta,\omega)=\begin{bmatrix} a_1\cos^{\varepsilon_1}\eta\cos^{\varepsilon_2}\omega \\ a_2\cos^{\varepsilon_1}\eta\sin^{\varepsilon_2}\omega \\ a_3\sin^{\varepsilon_1}\eta \end{bmatrix} \quad \begin{matrix} -\pi\le\omega<\pi \\ -\pi/2\le\eta\le\pi/2 \end{matrix} \qquad (2)$$

For linear primitives, we automatically segment the surface points from the LiDAR model based on a height filter and the connectivity properties. Figure 6. (a) shows the segmented surfaces, where different

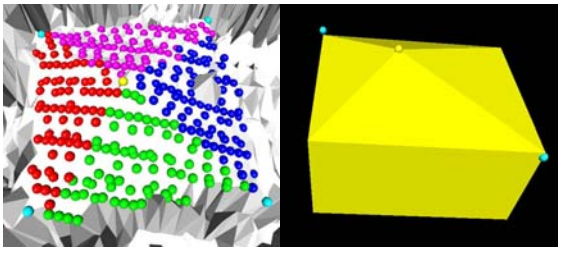


Figure 6. Linear primitive fitting.

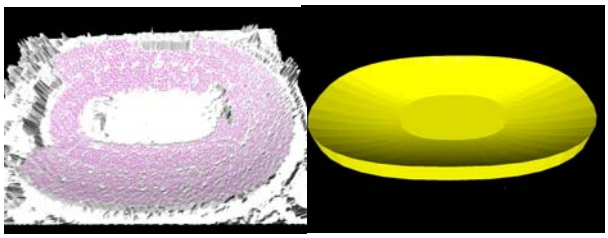


Figure 7. Non-linear primitive fitting.

color encodes points of different surfaces. The surfaces are fit using the points of each face with linear least square technique to obtain the optimal parameters (Figure 6. (b)). A region growing technique is used to segment surface points for curved rooftops (Figure 7. left, purple points), then the Levenberg-Marquardt (LM) algorithm is used to fit the non-linear parameters for superquadrics (Figure 7. right). Finally, the fitted local models are assembled to a complete building model (Figure 8). For further details, please refer to [15].

3.4 Modeling and Processing

We fuse the interactively extracted outlines from the aerial image and automatically extracted surfaces from the LiDAR model to reconstruct the complete model.

For each primitive, we map the outlines to the LiDAR model with the pre-computed affine transformation matrix to find the surface information. The surface for a flat cube is just one height, which is fitted from the LiDAR surface points. We assign the same height for each vertex on top of the cube, and use the predefined ground height for the bottom vertices, then triangulate the vertices to create the model. The same method is used for other flat roof primitives, such as polyhedrons and flat cylinders. For a roof with slopes, we find the corresponding height for each roof vertex using the parameterized slope and the mapping function, then triangulate them to generate the model.

The situation is more complicated for a curved roof surface with curved outlines. Since the surface is fitted from LiDAR data without considering the constraints of outlines from aerial images, the fitted surface will generally not pass through the mapped outlines. We use a non-linear optimization technique considering both surface and outline constraints to solve this problem. A weighted energy function is defined as in Equation 3, where f is the non-linear surface function, *Error* is the energy function defined as the distance of each point to the fitted surface, P_i is a surface point and $P_i^{'}$ is an outline point, and w is the weight for outlines. The z values of the outline points are found by mapping the outline from aerial image to LiDAR data. We set w to a high value to penalize surfaces that deviate from the extracted outlines. Again, the LM algorithm is used to find the best solution, and we tessellate the surface to create the model.

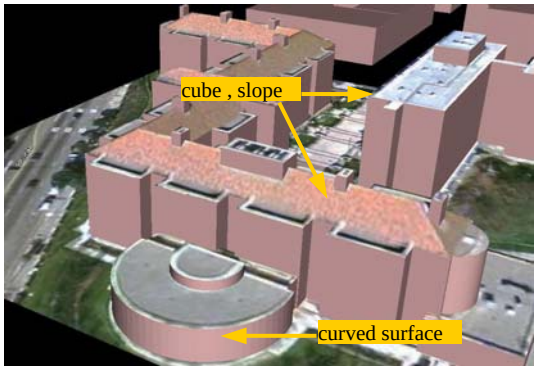


Figure 8. Hybrid modeling

$$E(f) = \underset{P_i \in Surface}{Error(f(P_i(x,y,z))} + w\,\underset{P_i^{'} \in Outline}{Error(f(P^{'}_i(x,y,z))} \qquad (3)$$

All the primitives are then assembled together to create the final model, and we map each vertex to the aerial image to find the texture coordinate for texture mapping. Figure 8 shows the result rendered using a VRML viewer with roof textures from the aerial image.

Model processing

To improve the quality, a series of processes are applied to the model, which includes normal correction, primitive intersection and redundant polygons removing.

Normals are important for correct lighting during real time rendering, and incorrect triangle vertex winding order will reverse the normal directions, hence cause wrong rendering results (Figure 10 (a)). Although models with reversed normals can be rendered correctly by turning on two-side-lighting in OPGL rendering, this method dramatically decreases the performance. In our system, we do not require the user to click the corners in a specific order while extracting outlines. This makes the system flexible and easy to use, however, the normals of the generated models may be reversed. We need an algorithm to generate correct normals for triangles with random ordered vertices in both clockwise and counter-clockwise orders.

Observing that a primitive is a closed object, we propose a region growing algorithm to compute the correct normals for each primitive. As illustrated in Figure 9, given a seed triangle ΔABC, and supposing its vertex winding order is correct as in counter-

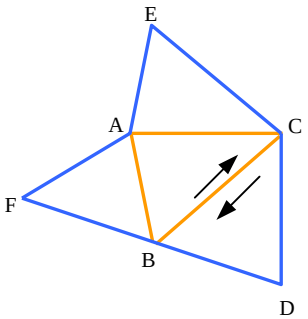


Figure 9. Compute normal.

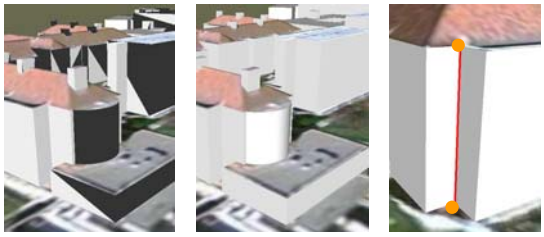
clockwise order, we can generate the correct normals for the whole primitive. We first find the three neighbor triangles of ΔABC, each of which has an incident edge with triangle ΔABC, and correct the vertex winding order for them based on ΔABC. E.g., the vertex winding order of ΔBCD should be CBD because ΔABC visits vertex B first, then C. We then push the three neighbor triangles into a stack, and find their neighbors and fix their vertex winding order. This growing process is continued until all triangles have been visited. Now the only problem is to find a seed triangle with a correct vertex winding order. This is accomplished by initializing the seed with a triangle at the bottom of the primitive since its normal direction always points to the negative Z direction. The algorithm is summarized as the following pseudo code:

1. *For each primitive, find a bottom triangle.*
2. *If the z component of its normal vector is positive, reverse the vertex winding order, and push it into the stack.*
3. *Pop a node from the stack, and find its three neighbors.*
4. *Correct the vertex winding order of the three neighbors based on the popped node, and push them into the stack.*
5. *Loop step 3 and 4, return when all triangles are visited.*

The algorithm is guaranteed to fix the vertex winding order for all triangles since each primitive is close and all triangles in a primitive are connected.

Another process is the primitive intersection. Our system is a primitive based modeling system, where complex buildings are represented by a number of different primitives. These primitives may intersect, however, there are no vertices or lines at the intersection part. We need to find the intersection, split the polygons, and insert some vertices. There are several reasons for this process. First, vertices and lines can be used for 3D to 2D correspondence to compute a camera's pose for façade texture mapping with ground view images. Another reason is that traditional texture mapping techniques also need vertices to assign texture coordinates. On the other hand, the intersection polygons of two primitives are completely invisible, which makes them redundant. It is necessary to find the redundant polygons and remove them to improve the rendering performance.

Since general 3D object intersection is complex, we simplify the problem to 2D based on the characteristic of our models. Building roofs can be texture mapped with a single aerial image, so we only focus on the intersection of facade walls, which are all vertical for our models. We project all the walls into x-y plane,



a b c

Figure 10. Normal correction and polygon intersection.

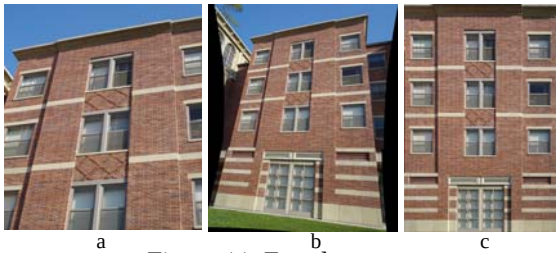
and find the intersections of lines in a 2D space. Figure 10. (c) shows that a intersection line is found and two vertices are inserted. This 2D method will not work for models hanging in the air, however, such cases are rare in our models.

After the primitive intersection process, we detect all the polygons completely inside a primitive, and remove these redundant polygons, which generally reduces by 20% the number of triangle meshes. The model is then exported as VRML format (Figure 8).

4.Enhance with Ground View Images

The modeling results from the hybrid method have textures on the building roofs, but lack façade textures. We enhance these models with ground view images for better visual effects. The key problem in generating facade textures is to automatically recover camera poses to compute the UV texture map. The pose recovery is often an under-constrained problem using a single image, so we mosaic images taken at almost the same viewpoint to form a large image that covers a whole face of a building primitive. We decompose a camera's pose into an orientation and a translation, and estimate them separately due to lack of features in rough building models and the narrow field of view of a camera. The orientation of the camera is estimated using vanishing points extracted by automatic line clustering, and the translation is computed using 3D to 2D corner correspondences [3]. Figure 11 (a) shows an input image that covers the upper part of a building face. The image is stitched together with other images taken at the same viewpoint to generate a mosaic image (b), and the mosaic is rectified with the automatically recovered pose to generate the texture (c).

Buildings generally have repeated texture patterns,



a b c

Figure 11. Façade textures

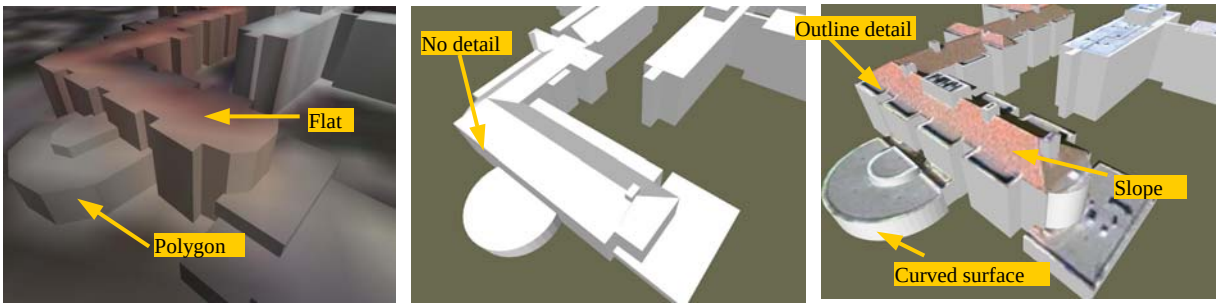


Figure 12. (a) Stereo aerial images result; (b) LiDAR data result (c) Hybrid modeling result.

so we use the same texture for different faces with similar texture patterns. This has several advantages. We only need to capture images of distinct patterns of building faces rather than capturing every face of the building, which is not possible under some circumstances, e.g., we cannot capture images at the southeast part of the building (Figure 1 (c)) because there is some field construction going on. Another advantage is that less texture memory is required to render the scene using repeated texture pattern technique, and the visual effects is almost the same as real photos.

We have developed an interactive tool to map textures to building facades. The system has four selecting modes: buildings, primitives, vertical walls and vertices. For texture mapping to vertical walls, the system needs only one mouse click, then it automatically finds the connected vertical wall in the same plane, and computes the texture UV for each vertex. The system also allows the user to use the shift key to select multiple walls and map it to the same texture. The other modes allow the user to map textures to non-rectangular shapes. We can map textures to a complex building with 160 faces within five minutes.

5.Results and Comparison

We have used our system to model a whole University campus and surrounding area with more than 100 buildings within two days. The original LiDAR has more than 4 million triangle meshes, and the refined model has only 20 thousand triangles with much better visual quality. Figure 13 (a) shows the whole rendered model with textures from an aerial image.

To demonstrate the effectiveness of the hybrid modeling technique, we have selected a complex building compound from LiDAR data (Figure 1 (b)), and downloaded a high-resolution aerial image from the Internet (Figure 1(a))[17]. We then modeled the building compound within five minutes using our system. More than 80 ground view images were taken to generate 24 façade textures with automatically recovered camera poses. These textures are mapped to the building with more than 160 faces using our tool within five minutes. Some of the textures are reused for building faces with similar texture patterns. Figure 13 (b) shows the overview of the building compound, two close up views are also shown in (c), (d). The texture for the curved surface is generated using multiple plane approximation and a stitching process.

Our hybrid modeling system is fast, accurate, and the rendering result is realistic. Since the surface information is fitted from LiDAR data points, it is loyal to the original data. Manual user verification shows that the error of building height is within several centimeters compared with the LiDAR ground truth. The precision of the outlines depends on the aerial image, which is within one foot for the image we used.

The hybrid modeling technique is superior compared with techniques based on aerial images or LiDAR only. As shown in Figure 12, the models from stereo aerial images [8] have only flat roofs (a), and the height precision is generally hard to achieve centimeter precision. The models from LiDAR data [15] have slope roof information and accurate height (b), but lack outline details. The hybrid modeling result (c) benefits from both datasets with accurate surface and detailed outlines. We also found that fusing information from the two datasets helps reduce errors in the modeling process of each dataset. Buildings with slightly different heights are hard to distinguish in aerial images, but such information is precisely maintained in LiDAR data. On the other hand, color information from the aerial image also helps the user to select the correct primitives in LiDAR data fitting.

Compared with façade system [5], our system is global and generates visually similar results in a much shorter time. Compared with other commercial software, such as Maya, our system is much faster and more accurate. Although an artist can generate a visually similar model to our results, the manually generated model is often inaccurate. Furthermore, it is very time consuming to measure every aspect of buildings and adjust every vertex to the correct position, especially for irregular shapes with curved surfaces.

a b c d

Figure 13. Hybrid modeling results

6.Conclusion

This paper presents a hybrid modeling system using three datasets: LiDAR data, an aerial image and ground view images. By fusing information from all the datasets, we benefit from each dataset, and generates realistic complex models for large-scale with accurate surfaces and detailed outlines. Future work includes integrating façade models from stereo ground view images.

Acknowledgment

This work made use of Integrated Media Systems Center Shared Facilities supported by the National Science Foundation under Cooperative Agreement No. EEC-9529152.

References

[1] F.E., Ahmed, S.B. James, Building Extraction Using Lidar Data, *ASPRS-ACSM Annual Conference and FIG XXII Congress*, April 22-26, 2002.

[2] J. Hu, S. You, and U. Neumann. Approaches to Large-Scale Urban Modeling, *IEEE Computer Graphics and Applications*, Volume 23, Issue 6, pp. 62 - 69.

[3] J. Hu, S. You, and U. Neumann. Automatic Pose Recovery for High-Quality Textures Generation, to appear in *ICPR*, 2006.

[4] R. Cipolla, T. Drummond and D.P. Robertson. Camera calibration from vanishing points in images of architectural scenes. *In Proceedings of British Machine Vision Conference*, 1999, pp.382–391.

[5] P. Debevec, C. Taylor and J. Malik, Modeling and Rendering Architecture from Photographs: A Hybrid geometry and image based approach, *Proceedings of SIGGRAPH 96*, pp. 11-20.

[6] C. Früh and A. Zakhor, Constructing 3D City Models by Merging Ground-Based and Airborne Views**,** in *CVPR*, 2003.

[7] A. Gruen and R. Nevatia (editors), Special Issue on Automatic Building Extraction from Aerial Images, *Computer Vision and Image Understanding*, 1998.

[8] S.C. Lee, A. Huertas, and R. Nevatia, Modeling 3-D Complex Buildings With User Assistance, *IEEE Workshop on ACV*, 2000, pp.170-177.

[9] D. Liebowitz, A.Criminisi and A. Zisserman, Creating Architectural Modeling From Images, *EUROGRAPH* '99, pp. 39-50, 1999

[10] C. Lin, R. Nevatia, Building Detection and Description from a Single Intensity Image, *CVIU*(72), No. 2, November 1998, pp. 101-121.

[11] M. Morgan, K. Tempeli, Automatic Building Extraction from Airborne Laser Scanning Data. *Proceeding of the 19th ISPRS Congress*, Book 3B, pp. 616-623.

[12] S. Noronha, R. Nevatia, Detection and Modeling of Buildings from Multiple Aerial Images, *PAMI*(23), No. 5, May 2001, pp. 501-518.

[13] M. Pollefeys et al. Visual modeling with a hand-held camera, *IJCV* 2004, 59(3), pp. 207-232.

[14] W. Ribarsky, T. Wasilewski, and N. Faust, From Urban Terrain Models to Visible Cities, *IEEE Computer Graphics & Applications*, Vol. 22, No. 4, 2002.

[15] S. You, J. Hu, U. Neumann, and P. Fox. Urban Site Modeling From LiDAR, *CGGM'*2003

[16] L. C. Chen et al. Fusion of LIDAR Data and Optical Imagery for Building Modeling, *IAPRS*, Vol. 35, No. B4, pp. 732-737.

[17] *http://earth.google.com/*

[18] http://www.cybercity.tv