3D Reconstruction of Urban Areas

Charalambos Poullis
Cyprus University of Technology
Immersive and Creative Technologies Lab
Limassol, Cyprus
Email: charalambos@poullis.org

Suya You
University of Southern California
Computer Graphics and Immersive Technologies Lab
Los Angeles, USA
Email: suyay@graphics.usc.edu

***Abstract*—Virtual representations of real world areas are increasingly being employed in a variety of different applications such as urban planning, personnel training, simulations, etc. Despite the increasing demand for such realistic 3D representations, it still remains a very hard and often manual process. In this paper, we address the problem of creating photorealistic 3D scene models for large-scale areas and present a complete system.**

The proposed system comprises of two main components: (1) A reconstruction pipeline which employs a fully automatic technique for extracting and producing high-fidelity geometric models directly from Light Detection and Ranging (LiDAR) data and (2) A flexible texture blending technique for generating high-quality photorealistic textures by fusing information from multiple optical sensor resources. The result is a photorealistic 3D representation of large-scale areas(city-size) of the real-world.

We have tested the proposed system extensively with many city-size datasets which confirms the validity and robustness of the approach. The reported results verify that the system is a consistent work flow that allows non-expert and non-artists to rapidly fuse aerial LiDAR and imagery to construct photorealistic 3D scene models.

***Keywords*-reconstruction; large-scale; modeling; LiDAR; texturing**

I. Introduction

Virtual representations of real world areas are increasingly being employed in a variety of different applications ranging from computer graphics, virtual reality, games, feature films to Geographical Information Systems(GIS). Despite the increasing demand for such realistic 3D representations, it still remains a very hard and often manual process - current approaches and systems to produce the photorealistic 3D representations are still time-consuming, expensive and labor intensive. In fact, the creation of models is still widely viewed as a specialized art, requiring personnel with extensive training and experience to produce useful models.

In this paper, we address the problem of creating photorealistic 3D scene models for large-scale areas and present a complete modeling system for the rapid and realistic production of city models from multiple sensor data. The proposed system comprises of two main components: (1) A reconstruction pipeline which employs a fully automatic technique for extracting and producing high-fidelity geometric models directly from Light Detection and Ranging (LiDAR) data and (2) A flexible texture blending technique for generating high-quality photorealistic textures by fusing information from multiple optical sensor resources. The result is a photorealistic 3D representation of large-scale areas(city-size) of the real-world.

We have integrated the developed techniques to produce a complete model system, and have tested the proposed system extensively with many city-size datasets which confirms the validity and robustness of the approach. The reported results verify that the system is a consistent work flow that allows non-expert and non-artists to rapidly fuse aerial LiDAR and imagery to construct photorealistic 3D scene models.

The following sections will detail the developed techniques and system. Section II summarizes existing knowledge in the areas of geometry modeling and texture generation. Section III presents the overview structure of the system. Section IV and V introduce the techniques of automatic geometry modeling and texture composition. Section VI presents the experimental results, and finally the Section VII concludes the presented work.

II. Related Work

Below is a brief overview of the existing knowledge in the areas of 3D model reconstruction and texture generation methodologies.

A. 3D Model Reconstruction

The proposed system in [1] can deal with uncalibrated image sequences acquired with a hand-held camera. 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. The ambiguity on the reconstruction is restricted from projective to metric through self-calibration.

Nevatia et al [2], 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 verify paradigm they are able to extract impressive models from a small set of aerial images. The authors later extended their work in [3] to automatically estimate camera pose parameters from two or three vanishing points and three 3D to 2D correspondences.

In [6] a ground-based LiDAR scanner is used to record a rather complex ancient structure of significant cultural heritage importance. Multiple scans were aligned and merged together using a semi-automatic process and a complete 3D model was created of the outdoor structure. The reconstructed model is shown to contain high-level of details however the complexity of the geometry limits this approach to the reconstruction of single buildings rather than large-scale.

In a different approach, [4] proposed an interactive system which can reconstruct buildings using ground imagery and a minimal set of geometric primitives. More recently [5] extended this system to incorporate pointcloud 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 according to the application.

Another ground-based approach is presented in [8] 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, as is also the case with the proposed method in [9] which combines unregistered range and image sensing for reconstructing photorealistic 3D models.

A similar ground-based approach is presented in [7] 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 more recent primitive-based system [10] presented a method for the rapid reconstruction of photorealistic large-scale virtual environments using a minimal set of three primitives. In this case the authors sacrifice full automation to achieve high-fidelity and high-quality models.

In [11] 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. Therefore, a significant color difference between images will cause visible and non-smooth transitions between neighbouring polygons assigned to different texture images.

B. Texture Generation

A method for creating renders from novel view-points is presented in [12], where densely regularly sampled images are blended together. The image acquisition process is simple and reproducible for relatively small objects, however the complexity of the capturing process greatly increases for large objects and especially in cases where the object exists in an outdoor environment where the lighting conditions cannot be controlled as in a lab environment. In addition, the proposed method does not involve the use or generation of geometric information thus, it limits its use to applications for visualization purposes only.

The method introduced in [4] uses a small set of images to interactively reconstruct a 3D model of the scene using a set of parameterized primitives. A view-dependent texture mapping method is then employed for the computation of the texture maps of the model. Although this technique is sufficient to create realistic renderings of the scene from novel view-points its computation is still too expensive for real-time applications, like games or virtual reality and the novel viewpoints are constrained to be close to the initial camera positions. In addition, view-dependent texture mapping works seamlessly in cases where the images are taken at regularly sampled intervals which is not generally true in the context of large-scale areas since some images may come from satellites and aerial images thus having a wide baseline between them.

In [13] they 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.

The authors in [14] estimate a set of blending transformations that minimizes the overall color discrepancy in overlapping regions in order to deal with the unnatural color texture fusion due to variations in lighting and camera settings.

A different approach is proposed in [15] to seamlessly map a patchwork of texture images onto an arbitrary 3D model. By specifying a set of correspondences between the model and any number of texture images their system can create a texture atlas.

A slightly different approach for texture generation and extraction is proposed in [16]. Given a texture sample in the form of an image, they create a similar texture over an irregular mesh hierarchy that has been placed on a given surface, however this approach cannot capture the "actual" appearance of the models.

The authors in [17] propose a method for computing the

blending weights based on a local and global component. This results in smooth transitions in the target image in the presence of depth discontinuities.

Another line of research uses video resources for the dynamic texturing of large-scale environments. Neumann et al [18], incorporate the dynamic input of multiple video cameras and generate projective textures for the models.

III. System Overview

The system overview consists of two modules: Building Generation and Texture Composition as shown in Fig. 1(a).

Initially, in the Building Generation module, 3D models representing the scene are generated from the unstructured input data in two steps:

1) Preprocessing. The unstructured data is subdivided into space and memory manageable parts which are converted into our internal format representation.
2) Building Extraction. An automatic segmentation groups neighbouring points of similar geometric properties into regions representing the buildings' roofs. The regions are then used to derive a set of noisy 2D roof boundaries which are refined by a novel boundary refinement method based on Gaussian Mixture Models. Finally, 3D models are extruded from the refined 2D roof boundaries.

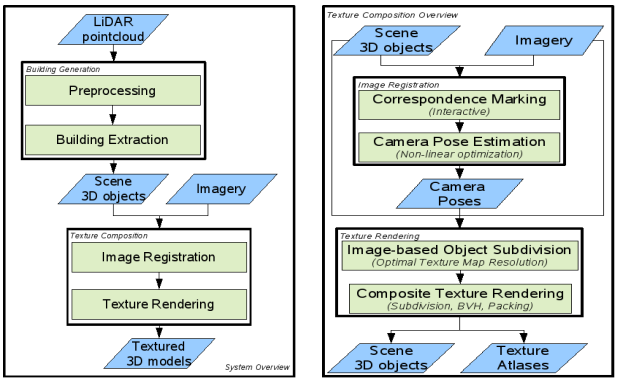
Next, in the Texture Composition module, the 3D models of the scene and imagery capturing their real-world appearance are combined together to create composite texture atlases for the realistic appearance of the 3D models in two steps:

1) Image Registration. A set of correspondences are interactively specified between the 3D geometry representing the scene and the imagery capturing the appearance of the same geometry. A non-linear optimization is then employed to recover the camera poses.
2) Texture Rendering. The geometry is subdivided into smaller primitives. A non-linear blending function which ensures a smooth and seamless transition between the different images, is then used to compute the composite texture maps. Finally, the composite textures are packed into compact texture atlases.

IV. Building Generation

A. Preprocessing

In this step, the data is subdivided into smaller, space/memory manageable parts and are represented by a set of 2D XYZ maps. The XYZ maps reduce the building extraction problem from a 3D to a 2D, therefore allowing all subsequent processing to be performed entirely in 2D and allowing the use of fast image processing techniques such as hole filling to be performed, which recovers any missing information from the local neighbourhood. This

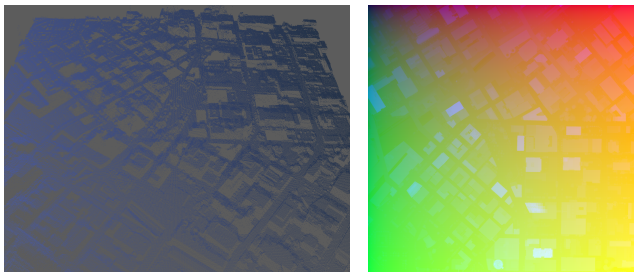


(a) (b)

Figure 1. (a) System Overview. (b) Texture Composition Overview

greatly reduces the computational complexity and improves the computational time of the system.

Fig. 2(a) shows the unstructured LiDAR data for a downtown area of Atlanta. The corresponding resampled, hole-filled 2D XYZ map for the same area is shown in Fig. 2(b). The R, G, B channels of the image correspond to the X, Y, Z axes respectively.



(a) Unstructured LiDAR data. (b) Resampled 2D XYZ map.

Figure 2. Preprocessing. The unstructured pointcloud is resampled into manageable components represented by 2D XYZ maps.

B. Building Extraction

The automatic extraction of buildings and the automatic creation of the 3D models is performed in two steps: Region segmentation and Boundary Extraction and Refinement.

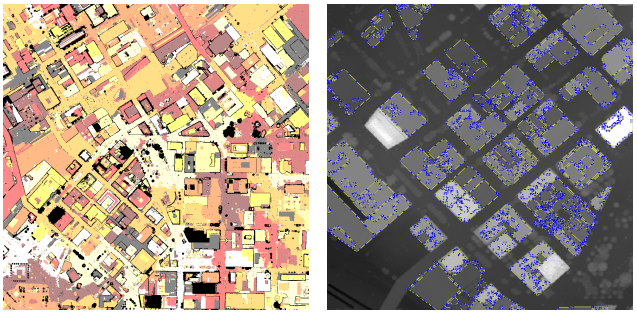
1) Region Segmentation: An automatic segmentation is performed on the resampled 2D XYZ maps to group neighbouring points of similar geometric properties into disjoint regions. Initially, a 1-D Gaussian distribution G_d and a 3-D Gaussian distribution $G_{\vec{N}}$ is created for each region R_i to describe the distributions of the depth and normals of all the points $P \in R_i$, respectively. The segmentation then begins by initializing a region R_0 with a starting point $P_{(x,y)} \in M$ in the XYZ map M. Candidate points in the 8-neighbourhood system are considered and are added iff the likelihood, of the candidate point's depth d_p and normal $\vec{n}_p$, belonging to the Gaussian distributions G_d and $G_{\vec{N}}$

describing the region being processed, is above an adaptive threshold as given by equations 1 and 2,

$$Pr(d_p) \geq Pr(\tau \times \mu_{G_d}) \quad (1)$$

$$Pr(\vec{n}_p) \geq Pr(\tau \times \vec{\mu}_{G_{\vec{N}}}) \quad (2)$$

where μ_{G_d} and $\mu_{G_{\vec{N}}}$ is the mean depth and the mean normal of the two distributions respectively, and $\tau = 0.5$ is a parameter which controls how similar the candidate point's properties have to be to be added to the region. Successful candidate points are added to the region and the Gaussian distributions describing that region are updated to reflect the change. This process is iteratively performed and a new region is initialized each time a region has considered all its neighbouring points and no change has occurred. The result is a set of disjoint regions shown with different colors in Fig. 3(a).



(a) (b)

Figure 3. (a) Color-coded segmentation result for downtown Atlanta. (b) The extracted roof boundaries after the refinement process using GMMs. The blue points indicate a boundary point and the yellow lines are edges connecting those points.

2) Boundary Extraction and Refinement: Next, the roof boundaries are extracted by computing the contour enclosing each segmented region using Suzuki's algorithm [19]. The result is a closed-contour boundary corresponding to each segmented region, which consists of a dense number of points and may contain artifacts primarily due to the noise in the original data.

Boundaries which are spatially close(within one pixel) to each other are grouped into the same *element*, and further processing is performed on the entire set of boundaries contained in each element. The reason for processing neighbouring roof boundaries together is because they are more likely to have similar orientations(or perpendicular) which helps with the determination of the principal orientations of the buildings. In the example of Fig. 3(b) the boundaries are grouped into 47 elements based on a one pixel proximity. Examples of elements are shown in Fig. 4.

A novel boundary refinement process is then applied in order to remove the artifacts and linearize the dense boundaries. A Gaussian Mixture Model(GMM) is used to classify the boundary points into different orientations. A GMM is a superposition of K Gaussian densities of the form,

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \quad (3)$$

where each Gaussian density (e.g. component of the mixture) $N(x|\mu_k, \Sigma_k)$ has its own mean μ_k and covariance Σ_k. The parameters π_k are the mixing coefficients for which $\pi_k \geq 0$ and $\sum_{k=1}^{K} \pi_k = 1$. The calculation of the parameters $\pi = \{\pi_1, ..., \pi_k\}$, $\mu = \{\mu_1, ..., \mu_k\}$ and $\mathbf{\Sigma} = \{\Sigma_1, ..., \Sigma_k\}$ is performed using an expectation maximization(EM) algorithm which maximizes the log of the likelihood function given by,

$$ln \quad p(\mathbf{X}|\pi, \mu, \mathbf{\Sigma}) = \sum_{n=1}^{N} ln\{\sum_{k=1}^{K} \pi_k N(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k\} \quad (4)$$

where $\mathbf{X} = \{x_1, ..., x_N\}$ are the data samples.

As mentioned previously a boundary B_i consists of a set of dense points. A three dimensional feature descriptor F_{P_j} is used to represent each boundary point $P_j^{B_i}$ and is defined as,

$$F_{P_j} = (\vec{T}_x, \vec{T}_y, \kappa_e) \quad (5)$$

where T_x, T_y is the local tangent orientation and κ is the local extrinsic curvature which is expressed in terms of the discrete first(') and second(") derivatives as,

$$\kappa_e = \frac{x^{'} y^{''} - y^{'} x^{''}}{(x'^2 + y'^2)^{3/2}} \quad (6)$$

In contrast to existing work, we do not assume that buildings always have four sides or parallel sides, thus we do not use GMMs of fixed order K, but instead we compute K using a Minimum Description Length(MDL) estimator criterion proposed by [20]. Hence, for each set of boundaries in an element we first determine the best number of components the GMM should have and then perform the fitting using the EM algorithm to minimize equation 4.

Fig. 4 shows the classification of the boundary points for three different elements. The color of each boundary point P_j indicates the component of the GMM of that element which maximizes the probability of the point's feature descriptor F_{P_j}. As it can be seen, the primary advantage of using a GMM for the classification of the boundary points, is that it can better separate the outlier points produced by noise and accumulating errors from the resampling process, thus removing the otherwise significant bias during the boundary refinement. Moreover, the use of MDL to determine the number of components K of the GMM model allows the application of this technique to linear as well as non-linear shapes.

Finally, the classification of the points is used to cluster sequential boundary points into groups which are then reduced to single lines using a least-square line fitting algorithm.

(a) Element 1. (b) Element 2. (c) Element 3.

Figure 4. Boundary grouping into elements and point classification using GMMs. The colors indicate the component of the GMM that maximizes the probability of the feature descriptor F_{P_j} of a point.

The refined and linearized boundaries are shown in Fig. 3(b). A significant reduction to the number of points and an improvement in the linear structure of the roof boundaries is clearly evident, however small artifacts still remain. This is primarily caused by the similar nature of the artifacts with the buildings' features, which makes it very difficult to discriminate between real features and features due to noise.

Finally, a plane fitting is performed to all the points enclosed by the refined boundaries. Each plane corresponding to a roof boundary is extruded in the vertical direction to form a water-tight 3D polygonal model. The reconstructed models for the downtown and surrounding area of a U.S city, covering a $16km^2$ area, is shown in Fig. 5.

Figure 5. Reconstructed polygonal 3D models for downtown Baltimore and surrounding areas ($16km^2$).

V. Texture Composition

The texture composition is performed in two steps: the Image Registration and Texture Rendering, shown in Fig. 1(b).

A. Image Registration

The input to this module is a set of 3D models representing the scene and a set of images capturing the appearance of the same scene. The output is the recovered camera poses corresponding to the images, as shown in Fig. 1(b).

A plethora of techniques have already been proposed for solving the problem of camera pose recovery. Firstly, we define the pinhole-camera model used to describe each camera. The extrinsic and intrinsic parameters of each camera are specified by the camera matrix C in equation 7,

$$C = \underbrace{\begin{bmatrix} \alpha & -\alpha cot(\theta) & u_0 \\ 0 & \frac{\beta}{sin(\theta)} & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{intrinsic} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}}_{extrinsic} \tag{7}$$

where $\alpha = kf_x$, $\beta = kf_y$, (f_x, f_y) is 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_{x-z} determine the camera's rotation and translation relative to the world.

Given a minimum of three 2D to 3D correspondences specified interactively by the operator the camera extrinsic and intrinsic parameters can be accurately estimated. The camera pose estimation is performed using a non-linear Levenberg-Marquardt optimization [21], [22] which minimizes the error function E_{C_k} for each camera C_k,

$$E_{C_k} = \frac{1}{n} \sum_{i=0}^{n} \sqrt{(I_x^i - P_x^i)^2 + (I_y^i - P_y^i)^2)} \tag{8}$$

where I^i is the ith image point, P^i is the projection of the ith 3D world point and n is the number of 2D to 3D correspondences.

An obvious problem of using an optimization to recover the camera extrinsic and intrinsic parameters is that it can get stuck in local minima and fail to converge to the global minimum solution. For example, varying the focal length (f_x, f_y) has the same effect as varying the translation t_z. Similarly, varying the principal point u_0, v_0 has the same effect as varying the translation t_x or t_y. In order to significantly decrease the likelihood of this problem occuring, we optimize the camera extrinsic and intrinsic parameters in an iterative, alternate process.

Firstly, the operator provides an initial estimate of the camera parameters. In the majority of the cases, we have found that just pointing the camera towards the object suffices and the optimization converges to a global minimum. Secondly, we perform the optimization in an iterative process. The extrinsic parameters are optimized first until the optimization converges followed by an optimization of the intrinsic parameters. This process is repeated until the optimization converges to a minimum solution.

B. Texture Rendering

The Texture Rendering module takes as input a set of scene 3D objects, the recovered camera poses and their associated images, and returns the resulting 3D objects along with a set of texture atlases. The generation of the resulting textured models involves two steps: Image-based Object

Subdivision and Composite Texture Rendering, as shown in Fig. 1(b).

Initially, the scene objects are decomposed into their corresponding faces which are then subdivided based on their image visibility. A bounding volume hierarchy is used as an efficient data structure for the representation of the subdivided objects in the scene. Next, the optimal texture map resolution is determined for each face and the composite texture maps are rendered. Finally, the composite texture maps are packed into one or more texture atlases.

1) Image-based Object Subdivision: In order to determine the optimal resolution of the texture map for each face, the face's vertices are projected into all the images using the recovered camera poses. The largest projected area for the face is defined as the optimal resolution. However, in some cases not all the projected points fall within the bounds of the image frame which can lead to problems when computing the texture map resolution. The problem is demonstrated by the example in Fig. 6(a) where the scene consists of three rectangles and two cameras having a white frame and a black frame respectively as shown in the insets. Fig. 6(b) and Fig. 6(c) show the projections of the objects into the image frame of the left and right camera respectively. The projected points which fall outside the bounds of the image frame can be heavily distorted due to the perspective nature of the projection and therefore cannot be used for the computation of the projected area.

To overcome this problem we perform an image-based visibility clipping where the 2D projections of the objects are clipped in image space and then backprojected to the 3D space where the objects are subdivided. This process is performed for all the cameras and all the objects in the scene and it guarantees that all faces in the scene are *entirely* visible in all the cameras. Fig. 6(d) and Fig. 6(e) shows the projections of the objects after the image-based object subdivision. Similarly, Fig. 6(f) shows the subdivided 3D objects produced by this process.

2) Composite Texture Rendering: The generation of the composite texture maps starts with the preparation of the scene for raytracing. Irregular faces are subdivided into the simple primitives used by the raytracer e.g. triangles and quadrilaterals. A bounding volume hierarchy is then used as the internal data structure for the efficient representation of the data.

Next, a texture map is raytraced for each face. In order to minimize the information loss we exploit all the information available from all cameras. This is achieved by considering the color contribution of each image in which the face is visible. The integration of these contributions is performed by a blending function which ensures the smooth transition between multiple cameras and is expressed as a non-linear function of multiple parameters given by,

$$f_{blend} = \|\alpha W_1 \times \beta W_2 \times \gamma W_3 \times \delta W_4 \times \epsilon W_5\| \quad (9)$$

where $\alpha, \beta, \gamma, \delta, \epsilon$ are the importance factors of the five weights respectively and $W_{1..5}$ are the normalized weights for the following parameters:

1) W_1 is the weight of the resolution of the current image and is given by, $W_1 = \frac{(I_{width} \times I_{height})}{(I_{max_{width}} \times I_{max_{height}})}$. Images with higher resolution are given a higher weight than images with low resolution.
2) W_2 is the weight of the distance of the camera from the point on the face and is given by, $W_2 = \|T_{camera} - P\|^2$, where T_{camera} is the camera translation and P is the 3D position of the point. An image from a camera which is far away from the point e.g. an image from a satellite, is given a lesser weight than an image from ground or aerial camera.
3) W_3 is the weight of the distance of the projected point from the closest image border. This ensures the smooth transition (i.e. feathering) between multiple images at the image borders.
4) W_4 is the weight of the distance of the projected point from the principal point of the image. This is in order to account for the radial lens distortion which increases as a non-linear function of the distance to the principal point.
5) W_5 is the weight of the dot product between the camera's viewing direction and the face's normal. A camera which views a face at a grazing angle is given a lesser weight than a front-view camera.

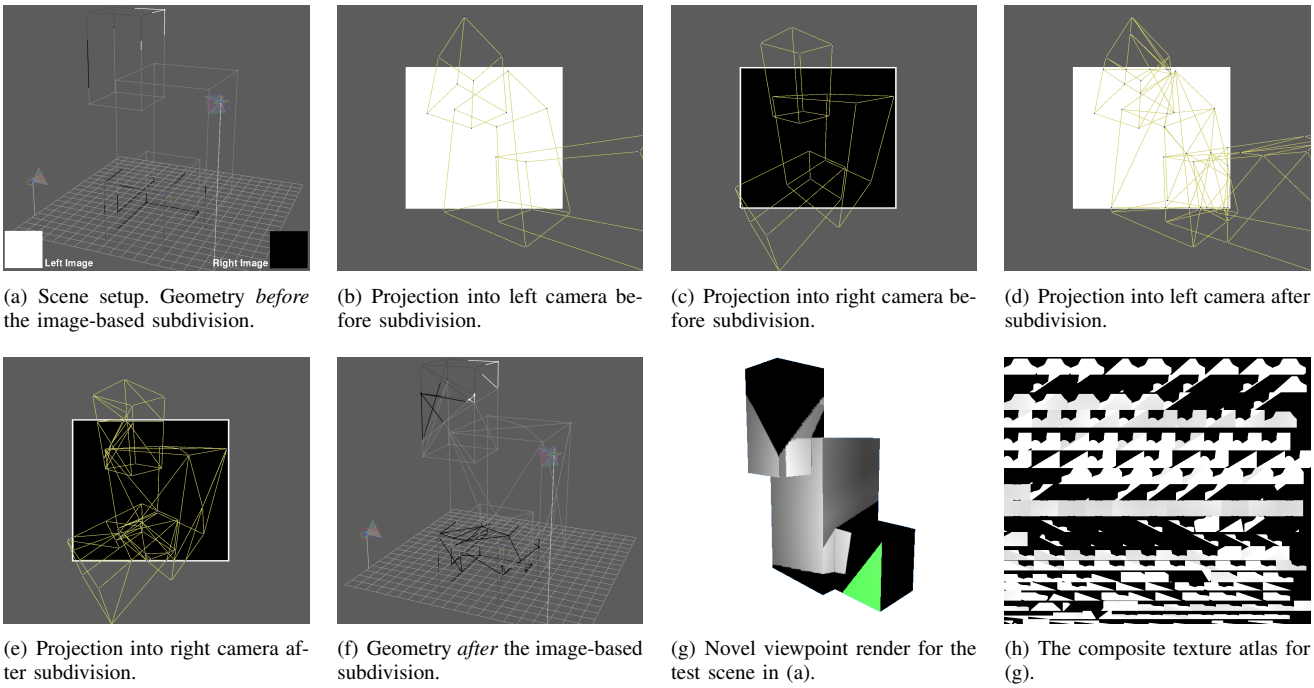
Hence, the resulting color of a point P on a face visible by n cameras is given by,

$$C_P = \sum_{i=0}^{n} f^i_{blend} \times I^i_{P_{proj}} \quad (10)$$

where f^i_{blend} is the combined weight computed by the blending function for the i'th camera, and $I^i_{P_{proj}}$ is the color at the projected point P_{proj} of the point P, in the image frame of the i'th camera.

Fig. 6 shows a render of the test scene introduced in Fig. 6(a) from a novel viewpoint using the composite texture maps. The integration and smooth transition between the black and white images is evident by the gray areas which are visible in both cameras. Non-visible faces for which no information is available are assigned a default green color. Similarly, backfacing faces are assigned a black color.

The composite texture maps produced for each face in the scene are of various sizes and shapes. To reduce the number of texture maps required for a particular scene, we pack the textures into larger collections of texture atlases. We employ a simple texture packing technique which first sorts the texture maps based on their width and height, and then copies each map sequentially in an atlas of a user-defined size. Additionally, a set of texture coordinates are computed and assigned to each face in the atlas. A variety of more advanced techniques have already been proposed [23], [24]



(a) Scene setup. Geometry *before* the image-based subdivision.

(b) Projection into left camera before subdivision.

(c) Projection into right camera before subdivision.

(d) Projection into left camera after subdivision.

(e) Projection into right camera after subdivision.

(f) Geometry *after* the image-based subdivision.

(g) Novel viewpoint render for the test scene in (a).

(h) The composite texture atlas for (g).

Figure 6. (a)-(f) Image-based object subdivision. (g)-(h) Texture composition result: 3D models and the composite texture atlas. The green color in (g) indicates non-visible faces for which no information is available.The blending of the black and white images results in the gray shades of the texture.

for efficiently packing textures into atlases and can be used instead. The result is a standalone 3D model with a set of associated texture atlases as shown in the example in Fig. 7.

Figure 7. A textured 3D model for a university campus. The green points indicate the interactively marked correspondences between the geometry and the imagery, which were used for the recovery of the camera pose.

VI. Experimental Results

Fig. 5 shows the result of the 3D reconstruction of a downtown and surrounding areas of a U.S. city. The generation of the 3D models was performed automatically in 11 hours and consisted of 36 XYZ maps of size 1Kx1K. In this example, the reconstructed models include all the data captured by the LiDAR scanner e.g. ground, roads, overhanging highways, bridges, trees, etc. A quantitative comparison between the original data and the resulting geometry indicates a reduction of 97.5% in the number of polygons(from 2088968 to 51442) and a reduction of 93.8% in the file size(measured in Mb of an OBJ ASCII file, from 151.4 to 9.3). Fig. 8 shows the textured 3D models using aerial oblique imagery.

Figure 8. A render of the textured 3D model for an area in downtown Baltimore.

VII. Conclusion

We have presented a complete and robust system for rapidly creating realistic virtual cities from LiDAR and imagery sensor data. Two significant components are developed for the system: a fully automatic technique for extraction of polygonal 3D models from LiDAR data, and a flexible texture blending technique for generation of photorealistic textures from multiple imagery data. Firstly, regions segmented by an automatic segmentation, are used to extract an

initial set of roof boundaries which are linearized and refined by a novel refinement process based on a Gaussian Mixture model classification and makes no particular assumptions about the shape of the boundaries, thus can be applied to linear as well as non-linear boundaries. The result is boundaries consisting of a reduced set of points which are used in combination with the elevation information of the enclosed points to generate light-weight, water-tight, 3D polygonal models representing the scene.

Secondly, imagery registered interactively to the geometry is used to recover the camera poses and texture atlases are generated by integrating multiple information together, therefore minimizing information loss. Moreover, the proposed non-linear blending function significantly reduces the appearance of artifacts in the composite textures. The texture composition process is independent of the 3D models, thus it can be applied in different contexts. The result is a *standalone 3D textured model*. Finally, we have presented our experimental results which verify the validity of our approach.

ACKNOWLEDGMENTS

We thank the Airborn1 Inc for providing us with the USC campus LiDAR data. We acknowledge the members and Prof. Ulrich Neumann in the Computer Graphics and Immersive and Technologies (CGIT) lab of USC. We also thank the reviewers for their valuable comments and suggestions.

REFERENCES

[1] M. Pollefeys, L. J. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *International Journal of Computer Vision*, vol. 59, no. 3, pp. 207–232, 2004.

[2] R. Nevatia and K. E. Price, "Automatic and interactive modeling of buildings in urban environments from aerial images," in *ICIP (3)*, 2002, pp. 525–528.

[3] S. C. Lee, S. K. Jung, and R. Nevatia, "Automatic pose estimation of complex 3D building models," in *WACV*. IEEE Computer Society, 2002, pp. 148–152.

[4] P. E. Debevec, "Modeling and rendering architecture from photographs," Ph.D. dissertation, University of California, Berkeley, 1996.

[5] C. Poullis, A. Gardner, and P. Debevec, "Photogrammetric modeling and image-based rendering for rapid virtual environment creation," *Proceedings of ASC2004*, 2004.

[6] 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," USC, ICT, Technical Report, 2004.

[7] 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," in *ICCV*, 2007, pp. 1–8.

[8] C. Zach, T. Pock, and H. Bischof, "A globally optimal algorithm for robust TV-L1 range image integration," in *ICCV*, 2007, pp. 1–8.

[9] I. Stamos and P. K. Allen, "Geometry and texture recovery of scenes of large scale," *CVIU*, vol. 88, no. 2, pp. 94–118, 2002.

[10] C. Poullis, S. You, and U. Neumann, "Rapid creation of large-scale photorealistic virtual environments," in *VR*. IEEE, 2008, pp. 153–160.

[11] C. Früh and A. Zakhor, "Constructing 3D city models by merging ground-based and airborne views," in *CVPR*. IEEE Computer Society, 2003, pp. 562–569.

[12] M. Levoy and P. Hanrahan, "Light Field Rendering," in *ACM SIGGRAPH '96 Proceedings*, 1996, pp. 31–42.

[13] A. R. Martinez and G. Drettakis, "View-dependent layered projective texture maps," in *Pacific Conference on Computer Graphics and Applications*. IEEE Computer Society, 2003, pp. 492–496.

[14] N. Bannai, A. Agathos, and R. Fisher, "Fusing multiple color images for texturing models," The University of Edinburgh, Tech. Rep. EDIINFRR0230, Jul. 2004.

[15] K. Zhou, X. Wang, Y. Tong, M. Desbrun, B. Guo, and H.-Y. Shum, "Texturemontage," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1148–1155, 2005.

[16] G. Turk, "Texture synthesis on surfaces," in *SIGGRAPH*, 2001, pp. 347–354.

[17] R. Raskar and K.-L. Low, "Blending multiple views," in *Pacific Conference on Computer Graphics and Applications*. IEEE Computer Society, 2002, pp. 145–155.

[18] U. Neumann, S. You, J. Hu, B. Jiang, and J. W. Lee, "Augmented virtual environments (AVE): Dynamic fusion of imagery and 3D models," in *VR*. IEEE Computer Society, 2003, p. 61.

[19] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *CVGIP*, vol. 30, pp. 32–46, 1985.

[20] J. Rissanen, "A universal prior for integers and estimation by minimum description length," *Ann. of Statist.*, vol. 11, no. 2, pp. 416–431, 1983.

[21] K. Levenberg, "A method for the solution of certain non-linear probelms in least squares," *Quart. Appl. Math.*, vol. 2, pp. 164–168, 1944.

[22] D. W. Marquardt, "An algorithm for least-squares estimation of non-linear parameters," *Journal of the Society of Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[23] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," *ACM TOG*, vol. 21, no. 3, pp. 362–371, Jul. 2002.

[24] GRAPHITE, *http://www.loria.fr/ levy/Graphite/index.html*, 2003.