3D Modeling Using Planar Segments And Mesh Elements [*]

Ioannis Stamos
Dept. of Computer Science
Hunter College / CUNY
New York, NY 10021
istamos@hunter.cuny.edu

Gene Yu and George Wolberg
Dept. of Computer Science
City College / CUNY
New York, NY 10031
wolberg@cs.ccny.cuny.edu

Siavash Zokai
Brainstorm Technology
New York, NY 10011
zokai@brainstormllc.com

Abstract

Range sensing technology allows the photorealistic modeling of large-scale scenes, such as urban structures. The generated 3D representations, after automated registration, are useful for urban planning, historical preservation, or virtual reality applications. One major issue in 3D modeling of complex large-scale scenes is that the final result is a dense complicated mesh. Significant, in some cases manual, post-processing (mesh simplification, hole filling) is required to make this representation usable by graphics or CAD applications. This paper presents a 3D modeling approach that models large planar scene areas of the scene with planar primitives (extracted via a segmentation pre-process), and non-planar areas with mesh primitives. In that respect, the final model is significantly compressed. Also, lines of intersection between neighboring planes are modeled as such. These steps bring the model closer to graphics/CAD applications. We present results from experiments with complex range scans from urban structures and from the interior of a large-scale landmark urban building (Grand Central Terminal, NYC).

1 Introduction

Our goal is the automated generation of coherent 3D models of large outdoor scenes by utilizing information gathered from laser range scanners and regular cameras. There is a clear need for highly realistic geometric models of the world for applications related to Virtual Reality, Telepresence, Digital Cinematography, Digital Archeology, Journalism, and Urban Planning. Recently, there has been a large interest in reconstructing models of outdoor urban environments. The areas of interest include geometric and photorealistic reconstruction of individual buildings or large urban areas using a variety of acquisition methods and interpretation techniques, such as ground-base laser sensing, air-borne laser sensing, ground and air-borne image sensing.

A typical 3D modeling system involves the phases of (1) individual range image acquisition from different viewpoints, (2) noise removal for each range image, (3) segmentation of each range image (i.e. extraction of lines, planes, etc.), (4) registration of all images into a common frame of reference, (5) transformation of each range image into an intermediate surface-based or volumetric-based representation, (6) merging of all range images into a common representation (3D model), (7) hole filling in the final 3D model, (8) simplification of the final 3D model, and (9) construction of CAD model of the scene. The third step (segmentation) is not utilized by all systems, but it is very important in our approach.

There has been significant progress in the area of 3D modeling from dense range scans. Representative 3D modeling systems include the following: [1, 2, 3, 4, 6, 7, 9, 14, 10]. We present a method for 3D modeling and mesh-simplification (sixth, and eighth task of a 3D modeling system) based on the segmentation results of each range image (third task). We also provide a hole filling algorithm in the final 3D model. Our goal is to retain the geometric details of the 3D model in areas where planar segmentation is not possible and to simplify the model in areas where planar segments from the segmentation module are available. This approach is beneficial in large-scale scenes that contain a large number of planar areas (such as urban environments). Our ultimate goal is the automated generation of a CAD model of the scene.

The fact that we are relying on the original segmentation results for modeling planar areas increases the time-efficiency and accuracy of our algorithms in these areas. Meshing the planar parts using the 3D range points increases the time- and space-complexity of the method. It is however much easier to model them as large planes. Also meshing algorithms are not able to correctly model sharp corner discontinuities due to noisy range measurements around corners. We, on the other hand, are able to robustly estimate the location of these sharp corners by the utilization of the planar segments. The major steps of our approach are as follows: (a) detection of planar segments in each range scan [13], (b) merging of planar components from overlapping range scans, (c) detection and representation of lines of intersection between planar components, (d) identification of the non-planar parts of the scene, (e) modeling of the non-planar parts with mesh primitives, and (f) hole filling.

2 Merging Planar Components

In this section we present a method that merges segmented planar areas of two or more overlapping range scans.

[*] Supported in part by NSF CAREER IIS-01-21239, NSF MRI/RUI EIA-0215962, ONR N000140310511, and NIST ATP 70NANB3H3056

We assume that the range-to-range registration problem has been solved, and that we are given a set of registered and segmented 3D scans. When two or more scans are registered, segmented planar areas (abbreviated as SPAs) that correspond to the same part of the scene but acquired from different viewpoints overlap. The overlapping SPAs do not match exactly in size, shape, and position due to (a) occlusions from other scene objects, (b) different fields of views (not seeing the same part of the scene), (c) different viewing angles and distances, (d) different amounts of noise (due to different grazing angles), and (e) errors introduced by the segmentation module. Our algorithm must detect the overlaps and then merge the overlapping regions into consistent polygonal areas. By doing this we expect to create segments that describe consistent 3D shapes (polygons in our case), and not only parts of the shapes seen from different points of view. Alternatively, a system could segment the final registered point cloud into SPAs. Such an approach would not require the merging of SPAs of different range scans. On the other hand, detecting planar segments along with their boundaries is a robust and efficient process if performed on each individual range scan. This is due to the fact that the structure (grid) of each range scan can be utilized for each individual image. That is why we independently segment each range scan and then merge the segments.

Each range scan R_i is represented as a two-dimensional array of 3D points $\{\mathbf{r}(k,l), k = 1 \dots N, l = 1 \dots M\}$[1]. The segmentation module [13] operates on each range scan R_i and produces a segmented range scan S_i. Each segmented scan S_i consists of a set of segmented planar areas (the terms SPA, segmented planar area, cluster, or planar cluster will be used interchangeably in the rest of the document). Each SPA C_i is a bounded plane with exterior and possible interior (i.e. holes) borders. It contains the following information:

1. The infinite plane $P(C_i)$ where C_i lies (unit normal and position of the plane),
2. The set of 3D points of scan R_i that lie on the SPA C_i.
3. A polygonal outer boundary of C_i as a sequence of range points $\mathbf{r}(k,l)$, and
4. Zero or more polygonal inner boundaries of C_i (holes).
5. A geometric center, or centroid.

The algorithm that merges SPAs between two overlapping scans is as follows (for details refer to Sec. 2.1):

1. The SPAs that do overlap between the two scans are detected. Two segmented planar areas C_i and C_j overlap iff they are co-planar (i.e. the lie on the same infinite plane) and their polygonal boundaries have a non-empty intersection. To test co-planarity the orientations and positions of the two infinite planes $P(C_i)$ and $P(C_j)$ need to be compared within an angle threshold a_{thresh} and distance threshold d_{thresh} respectively. In order to decide whether the co-planar polygonal boundaries of C_i and C_j have a non-empty intersection, the boundaries should first be rotated to become parallel to the plane $z = 0$. This allows us to perform 2D polygonal processing operations. In particular, we can intersect the rotated polygonal boundaries and decide whether the intersection is empty or not.
2. Overlapping $SPAs$ are united into a merged planar area. In order to achieve this the SPAs become parallel to the plane $z = 0$ through a rotation. Then the areas are united on the 2D space of the plane $z = 0$. Finally the united result is rotated back in 3D space by applying the inverse rotation.

In order to implement the aforementioned algorithm each SPA needs to maintain additional information. In particular, each SPA also includes the following members: *conformed* plane normal, *conformed* plane position, *conformed* set of outer/inner boundary points, and *conformed* centroid. The conformed data members play a key role. Initially they are equal to the actual plane normal/position, set of outer/inner boundary points, and centroid of the planar area. As the algorithm proceeds, the conformed data members diverge from the original values. In particular, when a set of $k \geq 2$ clusters $C_1, \dots, C_k$ are considered to be co-planar, then before any transformation takes place the (weighted) average normal and average position of all k clusters is calculated. The conformed normal and position of the k clusters is made equal to the average normal, and the inner and outer boundary points of each cluster is projected on the conformed plane (i.e. the average plane going between all k clusters). This projection constitutes the conformed inner and outer boundaries. Finally, the conformed centroid of all k clusters is made equal to the (weighted) average of the actual centroids of the clusters. Now, the conformed boundary points of the k clusters are rotated around their common conformed centroids. The fact that we are using the conformed boundary points means that all the rotated points lie on the exact same plane before and after the rotational transformation. If after the rotation the clusters do not overlap, then the algorithm reverts to the original plane positions, normals, and outer/inner boundary points.

Finally, the time complexity of the algorithm has been improved by the utilization of orthogonal bounding boxes around the boundaries of each planar cluster. If the bounding boxes of the boundaries of the clusters do not overlap then no further consideration of planar overlap is needed. A result of the merging algorithm for one building is shown in Fig. 4.

2.1 Skeleton of the algorithm

In this section we present the merging algorithm in more detail.

INPUT: A set $\mathbf{S_1} = \{C_1^1, C_2^1, \dots, C_N^1\}$ of N planar clusters (SPAs) from scan R_1, and a set $\mathbf{S_2} = \{C_1^2, C_2^2, \dots, C_M^2\}$ of

[1]The indices k, l define the position and orientation of the laser-beam which produces the 3-D point $\mathbf{r}(k,l)$.

M planar clusters (SPAs) from scan R_2.
OUTPUT: A set $\{C_1^u, C_2^u, \ldots, C_L^u\}$ of L united planar clusters , and K unchanged (due to no overlap) clusters $\{C_1^3, C_2^3, \ldots, C_K^3\}$.

STAGE 1: For each pair of clusters (C_i^1, C_j^2) between scans R_1 and R_2, decide if there is an actual overlap between them. No merging is taking place at this stage. The boolean variable $Overlap(C_i^1, C_j^2)$ becomes true iff an overlap exists.

Testing for overlap between two SPAs is achieved as follows:
If the bounding boxes of the two planar clusters overlap and their infinite planes are co-planar within an angle threshold a_{thresh} and distance threshold d_{thresh}, then:

1. *Conformed* plane normals/positions, centroids, and boundary points for both clusters are computed.
2. A common *conformed* centroid (weighted average based on the number of boundary points) for both clusters is computed.
3. Both clusters are rotated around the common *conformed* centroid so that they become parallel to the $z = 0$ plane.
4. The transformed boundary points are translated into generalized LEDA [8] polygons. The boolean intersection operator between the two LEDA polygons is applied. Finally, the two clusters overlap if and only if the intersection of their LEDA polygons is not empty.

STAGE 2: The pairwise $Overlap$ relation between pairs of clusters in $\mathbf{S_1} \times \mathbf{S_2}$ defines an equivalence relation EQ among the clusters in the set $\mathbf{S_1} \cup \mathbf{S_2}$. The relation EQ is defined as follows: (a) $(C_i, C_i) \in EQ$ for all clusters $C_i \in \mathbf{S_1} \cup \mathbf{S_2}$. (b) If $Overlap(C_i, C_j)$ then $(C_i, C_j) \in EQ$, for all cluster $C_i \in \mathbf{S_1}$ and $C_j \in \mathbf{S_2}$. (c) If $(C_i, C_j) \in EQ$ then $(C_j, C_i) \in EQ$, for all clusters in $\mathbf{S_1} \cup \mathbf{S_2}$. (d) If $(C_i, C_j) \in EQ$ and $(C_j, C_k) \in EQ$ then $(C_i, C_k) \in EQ$ for all clusters in $\mathbf{S_1} \cup \mathbf{S_2}$. In other words this equivalence relationship partitions the clusters of $\mathbf{S_1} \cup \mathbf{S_2}$ into subsets that define common planar surfaces[2].

In this stage the set $\mathbf{S_1} \cup \mathbf{S_2}$ of clusters is partitioned into the equivalence classes defined by EQ. All clusters in each equivalence class are part of the same extended planar surface.

STAGE 3: This is the last stage, where all clusters in the same equivalence class (i.e. clusters that are part of the same planar extended surface) are being united. Note that the partition has been computed in the previous step. Now for each set $\{C_1, \ldots, C_m\}$ of an equivalence class (with $m \geq 2$) the union C_u is computed as follows:

1. *Conformed* plane normals/positions, centroids, and boundary points for all m clusters are computed.
2. A common *conformed* centroid (weighted average based on the number of boundary points) for all clusters is computed.
3. All clusters are rotated around the common *conformed* centroid so that they become parallel to the $z = 0$ plane
4. The transformed boundary points are translated into generalized LEDA polygons and the boolean unification operator between the two LEDA polygons is applied. A new cluster C_u is created. The resulted generalized polygon is transformed into the outer and inner boundaries of C_u.
5. Finally, the inverse of the rotation computed in step 3 is applied to the boundaries of C_u, so that it is now expressed in the original coordinate system.

STAGE 4: The final output consists of all the united clusters C_u and all clusters $C_i \in \mathbf{S_1} \cup \mathbf{S_2}$ that do not overlap with any other cluster in the data set.

[2]For instance if $Overlap(C_i, C_j)$ and $Overlap(C_j, C_k)$ then clusters C_i, C_j and C_k are all part of an extended planar surface.

3 Producing straight planar borders

One of the drawbacks of automated 3D model creation from range datasets is the inability to capture clean straight boundaries at sharp normal discontinuities (corners) due to noisy measurements around the corners. An example of this problem is shown in Figures 6(a) and 6(b). Creating straight borders from the measured boundaries is the second step (after planar segmentation) of automatic CAD model generation. Our approach is based on the automated computation of lines of intersection between neighboring planar clusters. These lines are very accurately and robustly computed. That is why they can become the missing straight 3D borders. A detail of our results is shown in Figures 6(c) and 6(d).

4 Identifying range points not on SPAs

The computation of the merged planar areas from a set of range scans, naturally leads to the need to combine the planar with the non-planar areas of the 3D scene. Consider two range scans R_1 and R_2 that have been placed into the same frame of reference after registration. Let us call $\mathbf{S_1}$ the set of segmented planar areas of the first scan and $\mathbf{S_2}$ the set of segmented planar areas of the second scan. Our merging module merges the segmented planar areas $\mathbf{S_1}$ and $\mathbf{S_2}$ (by uniting overlapping areas) resulting to the set $\mathbf{S_m}$ of planar areas (see section 2). Let us call $\mathbf{NP_1}$ the set of unsegmented points from the first scan. These are the range points of R_1 that are not in any segmented planar area of $\mathbf{S_1}$ (this information is given by the segmentation module).

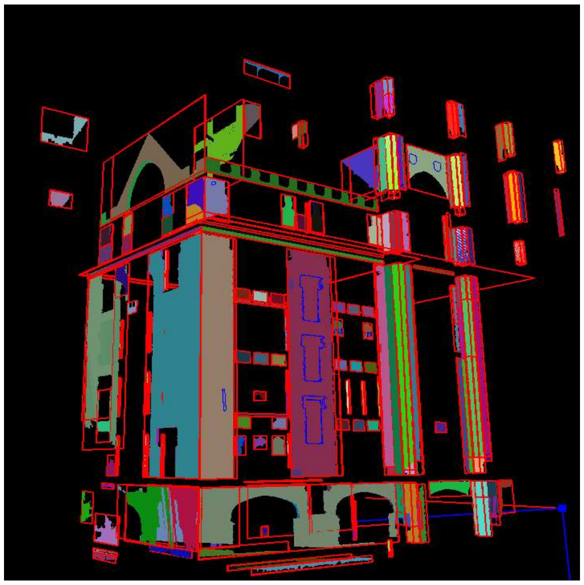


Figure 2: Axes-aligned bounding boxes of planar areas with respect to the coordinate system that conforms to the building structure (shown in red), with tightly enclosed planar areas (shown in various colors). The scene is seen from a view that is different that the view used in figures 1(a) and 1(b).

Similarly, $\mathbf{NP_2}$ is the set of unsegmented points from the second scan. We wish to identify the set of points $\mathbf{NP} \subset \mathbf{NP_1} \cup \mathbf{NP_2}$ that lie within the planar areas of $\mathbf{S_m}$. Then the set of points $\mathbf{NP_1} \cup \mathbf{NP_2} - \mathbf{NP}$ contains only points not on any planar area of $\mathbf{S_m}$

We first compute the axes-aligned bounding boxes of the planar areas in the set $\mathbf{S_m}$. This will help us to perform a fast[3] interiority test. That means that we can efficiently decide which unsegmented points lie within the bounding boxes of the planar areas. The points that are interior to the bounding boxes may or may not overlap the planar areas, and further testing is necessary. However, points that are exterior to the bounding boxes do not overlap any planar area; no further testing is therefore required for them. Since the range scans are not aligned with the axes of the coordinate system, the computed bounding boxes have large extent, making most points being interior to them (see Fig. 1(a)). That is why the major 3D directions of the 3D scene are calculated. The range points are then rotated in a way that makes the three Cartesian axes parallel to the three major directions of the scene. The bounding boxes are now much smaller and tightly enclose the planar areas (see Figs. 1(b) and 2). In this way most of the unsegmented points that do not overlap the planar areas fall outside of the bounding boxes and are excluded from further consideration.

The second step is to identify which unsegmented points in $\mathbf{NP_1} \cup \mathbf{NP_2}$ that are interior to the bounding boxes actually lie on planar areas. This step is necessary because a point that is classified as unsegmented from one view may happen to lie on a segmented planar area of another view. For every unsegmented point P_i that lies within the bounding box of a planar area A_j the following processing is performed:

1. The unsegmented point and the planar area are rotated so as for the planar area to be on plane $z = 0$.

2. The z-coordinate of the rotated point $P_i' = [x_i, y_i, z_i]^T$ is the vertical distance of the point from the planar area. If this distance is larger than a threshold $z_{th} = 0.1m$ the point is still considered as unsegmented.

3. Otherwise, the projection $[x_i, y_i]^T$ of the unsegmented point on the plane $z = 0$ is considered. If this projection lies within the polygonal boundary of the rotated planar area, then the point actually lies on the area. In this case the status of the point changes from unsegmented to planar. This interiority test is an expensive operation implemented by the LEDA library. Our algorithm, however, is computationally feasible due to the fact that most of the points have been discarded from further consideration in the first step (bounding box interiority step).

In summary an unsegmented point of the set $\mathbf{NP_1} \cup \mathbf{NP_2}$ can change its status (i.e. become planar) if and only if all the following apply:

- The point is interior to one bounding box.

- The distance of the point from the planar area of the bounding box is smaller than $z_{th} = 0.1m$ (step 2 of the previous test).

- The projection of the point on the planar area lies within the boundaries of the planar area (step 3 of the previous test).

Fig. 3 shows unsegmented points correctly classified as lying on planar areas or not. From a total of $1,717,185$ points from two scans $943,265$ (54.9% of the total) are classified as planar (i.e. the produce the segmented planar areas of the two scans) by the segmentation phase. Of the remaining $773,920$ unsegmented points: 1) $54,351$ points (0.03% of the total) lie on the planar areas and thus become planar points. 2) $719,569$ points (41% of the total) retain their status (i.e. they do not lie on any planar area). Finally, the unsegmented points can be further classified by calculating their distance from the bounding boxes. Out of the $719,569$ unsegmented points that retained their status, $528,807$ (30.8%) are within 1 meter from at least one bounding box, and $190,762$ (11% of the total) points are further away.

[3] $O(NM)$ time complexity, where N is the number of unsegmented points and M is the number of planar areas. Note that M is much smaller than the number of range points.

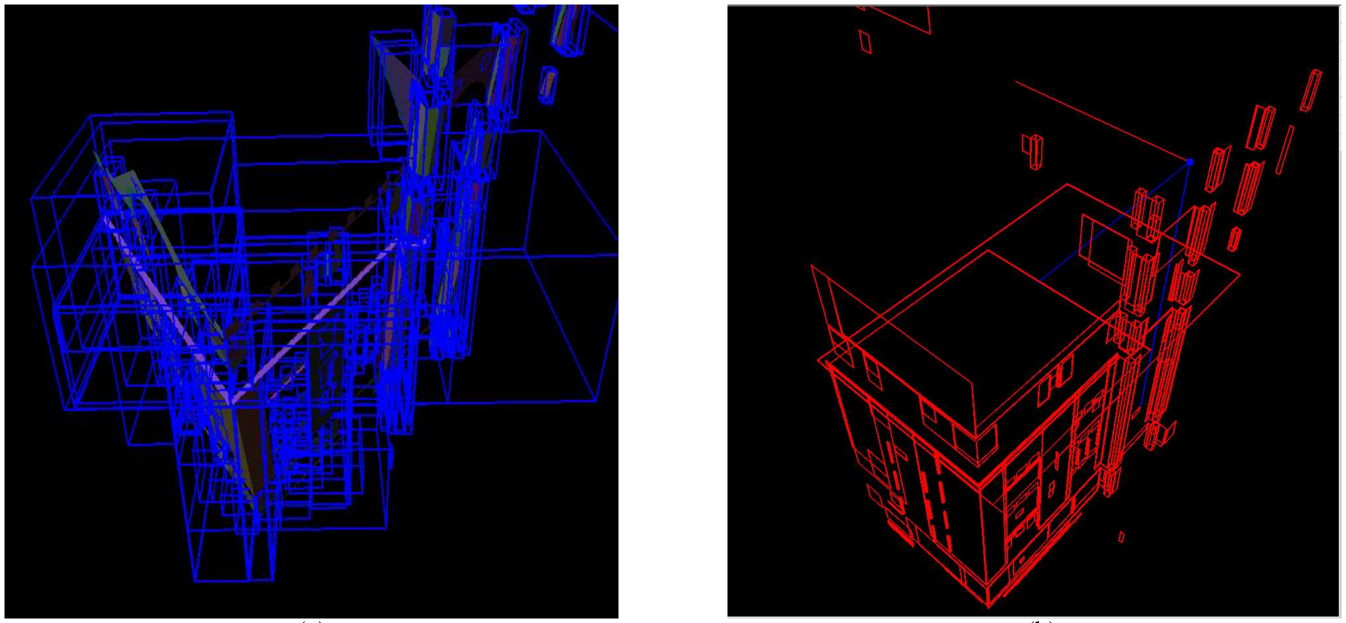

(a) (b)

Figure 1: **(a)** Axes-aligned bounding boxes of planar areas with respect to scanner's coordinate system (shown in blue). Note that the boxes have large extent since the measured surfaces are tilted with respect to the axes of the range scanner.**(b)** Axes-aligned bounding boxes of planar areas with respect to the coordinate system that conforms to the building structure (shown in red). The bounding boxes are now much smaller and tightly enclose the planar areas.

5 Meshing Algorithm

Parts of the scene that are not planar (as identified by the algorithm of the previous section) are modeled via an implementation of the ball pivoting algorithm [3]. This algorithm has a number of advantages: a) it is general, b) it is conceptually simple, c) it does not average the point measurements but faithfully follows the data, and d) it allows for space-efficient out-of-core implementation (i.e. the data can be processed in slices that fit in main memory). Its main disadvantage (wrt [4] for example) is that the selection of a ball radius ρ is critical, and that a large number of holes will appear in the final model. However, its natural out-of-core implementation makes it appropriate for large datasets.

The basic idea of the algorithm is as follows. The input is a set of registered range scans that are represented as 3D points with normals. The connectivity information within points of the same scan does not need to be maintained. Suppose that we are given a ball of radius ρ. This radius is a critical parameter that needs to be specified. First, from the set of 3D points a seed triangle of three points is detected. Three points define a seed triangle iff an *empty* ball of radius ρ passes through all three points. This seed triangle is the first triangle of the output mesh. The algorithm continues by rotating the initial ball around one of the edges until it hits another data point. Then a new triangle (there are special cases discussed in the original paper) will be added to the mesh. The ball continues to roll as long as new points are being considered. We should note that this ball is always in contact with exactly three points of the dataset and that is always *empty*. If the ball is not able to hit any other point, a search for a new seed triangle is initiated and the process of the rotating ball continuous. The algorithm terminates when no seed triangle can be detected. The selection of the radius ρ is thus very important, since a small ball will pass through the points and no mesh will be created, and a large ball will not be able to capture concavities of high curvature. Unfortunately, due to the inability to select one or more optimal ball sizes, and due to noise in the computation of normals, a number of holes appear in the data. A hole filling algorithm is thus essential.

6 Hole Filling

As mentioned in the previous section, the meshing algorithm produces a large number of holes. The sequence of boundary edges that surround these holes are identified during mesh generation. The hole-filling method of [5] considerably smooths the hole areas. It is thus appropriate for small holes and smooth data sets (such as statues) and less effective for large holes and scenes with sharp discontinuities. Here is the outline of our current hole-filling algorithm:

(1) The input mesh is segmented into connected components. This step helps to determine interior holes vs. exterior boundaries. Note that exterior boundaries should not be filled.

(2) The holes are parametrized by edge tweaking [11]. Each hole is parameterized as follows:

- Starting at an arbitrary first point, traverse the hole in the order given. For each vertex v_i (a) Estimate the

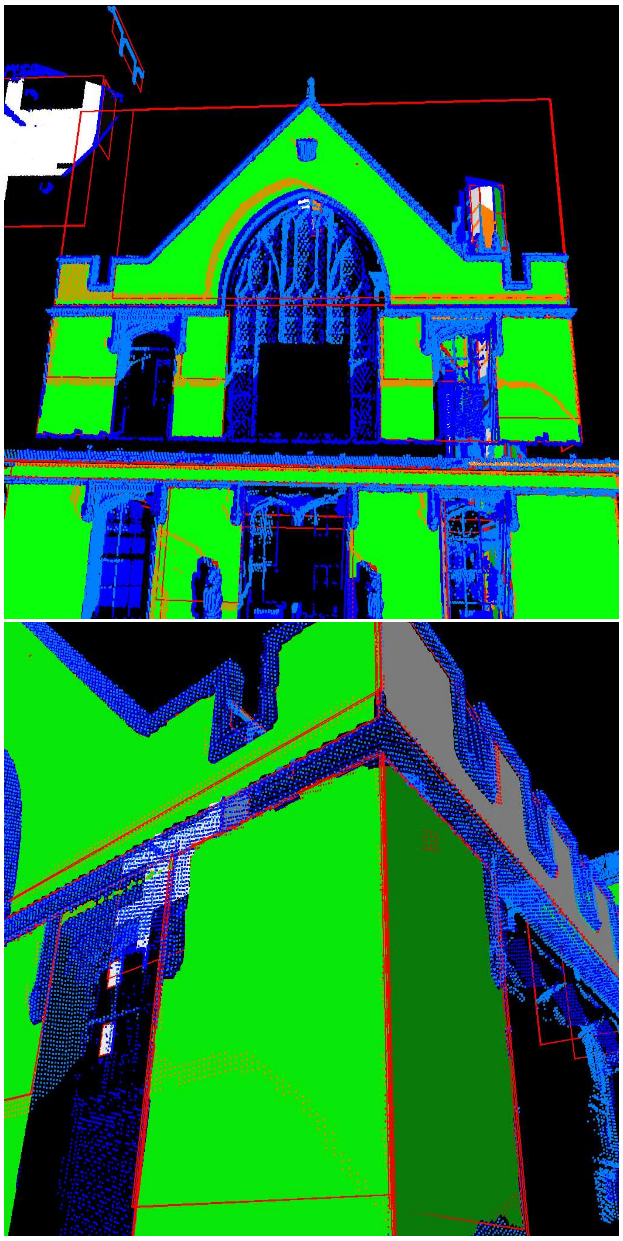


Figure 3: **(a) Top.** Merged segmented planar areas (shown in green) from two scans. Unsegmented points that do not lie on planar areas are shown in blue. Unsegmented points that are classified as planar are shown in red. These points become planar because they are in the bounding box of a planar area, their distance from the planar area is smaller than $z_{th} = 0.1m$, and their projection lies within the boundaries of the planar area (see Sec. 4). The unsegmented points will be used for the generation of a triangular mesh, whereas the planar points will become part of the planar areas on which they lie. **(b) Bottom.** Unsegmented points and planar areas. A different part of the scene is shown.

normal at the vertex, (b) Project the previous and next edge into the plane defined by the estimated normal, and compute the angle a_i between the edges. Finally, (c) Compute the length l_i of the previous edge.

- Transfer the 3D angle and length measurements to the plane as follows: start walking along the x-axis for a distance of l_0 units, then turn a_0 degrees. Next, walk l_1 units and turn a_1 degrees, etc.

- Transferring the angles and lengths into the plane introduces distortion, so that the starting point and the ending point of the loop will not meet in general. Edge tweaking closes the loop by defining an objective function that constrains the endpoints to meet. The objective function allows the edge lengths to vary, but does not allow the angles to vary.

- If the amount of distortion is too great, the parameterized curve will have self-intersections. Solving this problem is part of our future work.

- Eliminate surface borders by checking the sum of the angles around the loop. Outer boundaries can be detected because they have the opposite sign from holes. Large distortions may cause this test to fail. In practice, this test seems to generate false negatives but not false positives.

- Triangulate the parameterized loops using Delauney refinement [12].

- Lift the new points to 3D space.

We would also like to investigate the algorithm of [15] that seems promising.

7 Conclusions

We have presented a method that starting from a set of segmented range scans, produces large planar areas, along with dense mesh elements. Linear borders of intersection are also computed. One of our result is shown in Fig. 7. Our main contribution is that we provide a framework for automated CAD scene creation from range datasets.

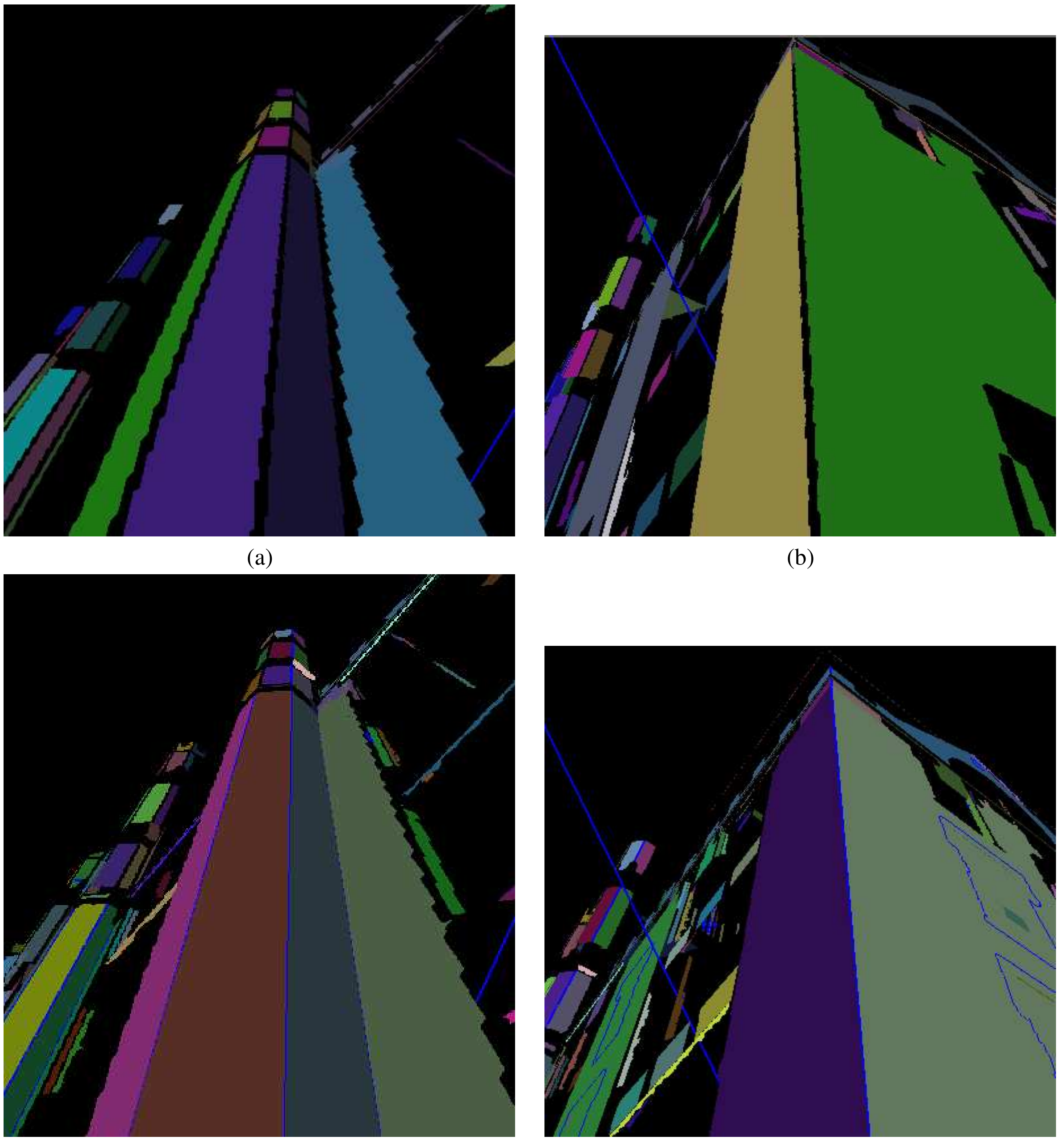
References

[1] Visual Information Technology Group, Canada, 2004. http://iit-iti.nrc-cnrc.gc.ca/about-sujet/vit-tiv_e.html.

[2] P. K. Allen, I. Stamos, A. Troccoli, B. Smith, M. Leordeanu, and S. Murray. New methods for digital modeling of historic sites. *IEEE Computer Graphics and Applications*, 23(6):32–41, 2003.

[3] F. Bernardini and H. Rushmeier. The 3D model acquisition pipeline. *Computer Graphics Forum*, 21(2):149–172, June 2002.

[4] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, pages 303–312, 1996.



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

Figure 6: (a) & (b) Two views of segmented model of a building before the creation of straight borders. Note that due to the finite number of range samples on the borders, it is not possible to have exact straight borders at the boundaries of planar clusters. (c) & (d) Two views of the same model after the application of the automated straightening process. The intersection lines (blue) are shown as well.

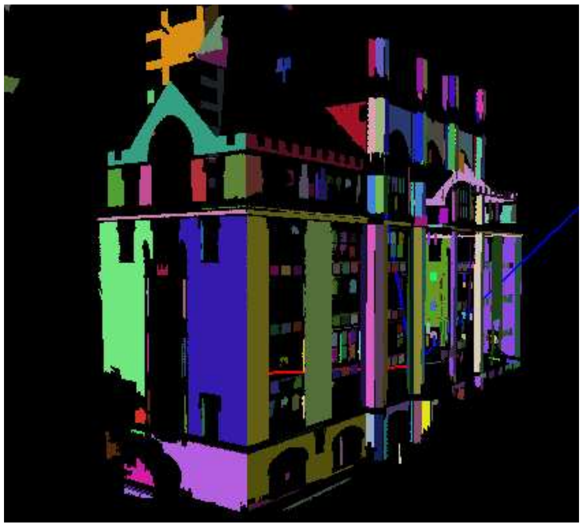


Figure 4: View of the final planar 3D model after all planar segments from ten registered segmented scans are merged. Again, different planes are represented with different colors for clarity.

[5] J. Davis, S. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *3DPVT*, June 2002.

[6] M. Fiocco, G. Bostrm, J. G. M. Gonalves, and V. Sequeira. Multisensor fusion for volumetric reconstruction of large outdoor areas. In *The 5th Intern. Conf. on 3-D Digital Imaging and Modeling*, Ottawa, 2005.

[7] C. Früh and A. Zakhor. Constructing 3D city models by merging aerial and ground views. *IEEE Computer Graphics and Applications*, 23(6):52–11, 2003.

[8] Leda computational geometry library. `http://www.mpi-sb.mpg.de/LEDA/`.

[9] M. Reed and P. K. Allen. 3D modeling from range imagery. *Image and Vision Computing*, 17(1):99–111, February 1999.

[10] C. Rocchini, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, and R. Scopigno. The marching intersections algorithm for merging range images. *The Visual Computer*, 20(2–3):149–164, May 2004.

[11] T. W. Sederberg, P. Gao, G. Wang, and H. Mu. 2-d shape blending: An intrinsic solution to the vertex path problem. In *SIGGRAPH*, pages 15–18, 1993.

[12] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1–3):21–74, May 2002.

[13] I. Stamos and P. K. Allen. Geometry and texture recovery of scenes of large scale. *Computer Vision and Image Understanding*, 88(2):84–118, Nov. 2002.

[14] I. Stamos and M. Leordeanu. Efficient model creation of large structures based on range segmentation. In *3DPVT*, September 2004.

[15] J. Wang and M. Oliveira. A hole-filling strategy for reconstruction of smooth surfaces in range images. In *SIBGRAPI*, 2003.

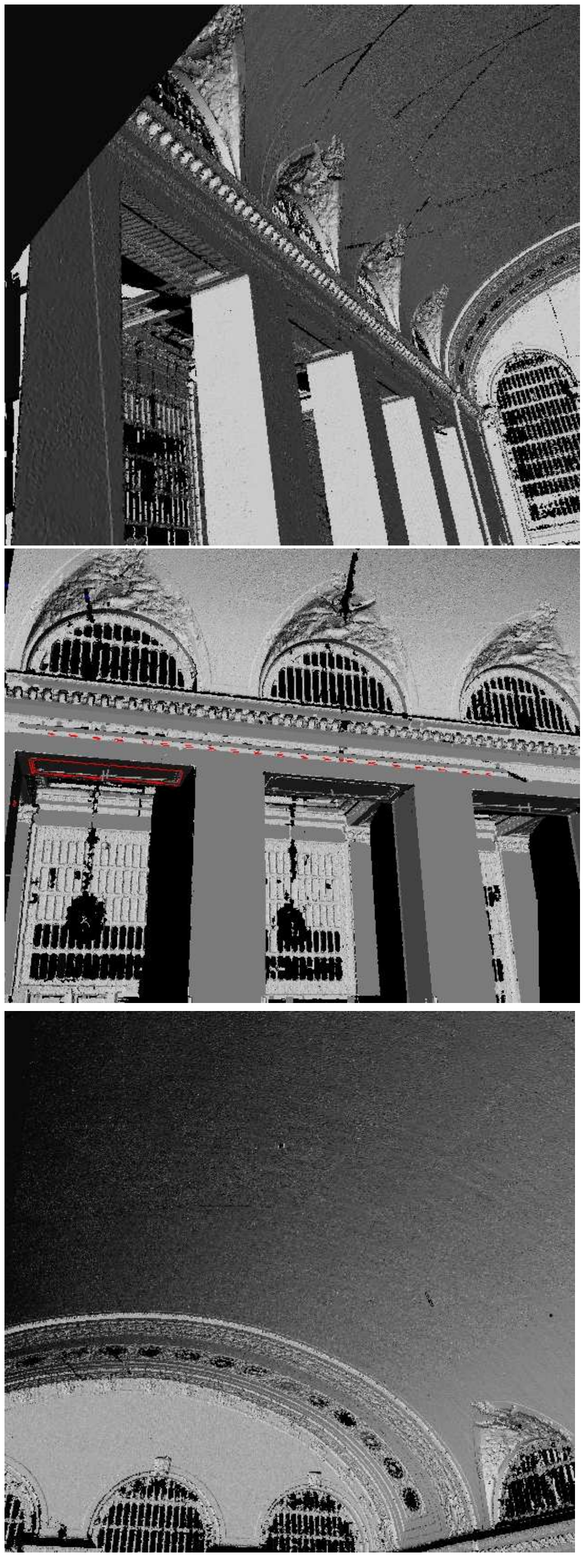


Figure 5: (Top) 3D model constructed via an implementation of the ball pivoting algorithm. Note that large planar areas around the columns are modeled as dense meshes. (Middle) Hybrid model: planar areas are modeled as large polygons (gray color for clarity), while non-planar areas are modeled via mesh. This representation is closer to a 3D CAD model. (Bottom) Part of the model that is modeled as mesh.