Efficient Model Creation of Large Structures based on Range Segmentation *

2nd International Symposium on 3D Data Processing, Visualization & Transmission, September 2004, Thessaloniki, Greece.

Ioannis Stamos
Department of Computer Science
Hunter College and Graduate Center
City University of New York
istamos@hunter.cuny.edu

Marius Leordeanu†
The Robotics Institute
Carnegie Mellon University
mleordea@andrews.cmu.edu

Abstract

This paper describes an efficient 3D modeling method from 3D range data-sets that is utilizing range data segmentation. Our algorithm starts with a set of unregistered 3D range scans of a large scale scene. The scans are being preprocessed for noise removal and hole filling. The next step is range segmentation and the extraction of planar and linear features. These features are utilized for the automatic registration of the range scans into a common frame of reference [13]. A volumetric-based algorithm is used for the construction of a coherent 3D mesh that encloses all range scans. Finally, the original segmented scans are used in order to simplify the constructed mesh. The mesh can now be represented as a set of planar regions at areas of low complexity and as a set of dense mesh triangular elements at areas of high complexity. This is achieved by computing the overlaps of the original segmented planar areas on the generated 3D mesh. The example of the construction of the 3D model of a building in the NYC area is presented.

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. The ultimate goal is the reconstruction of detailed models of urban sites (digital cities). The creation of digital cities drives other areas of research as well: visualization of very large data sets, creation of model data-bases for GIS (Geographical Information Systems) and combination of reconstructed areas with existing digital maps.

* Supported in part by NSF CAREER IIS-01-21239, NSF MRI/RUI EIA-0215962, and CUNY Institute of Software Design and Development award.

† Work done when the author was at Hunter College, City University of New York.

A typical 3D modeling system involves the phases of

1. Individual range image acquisition from different viewpoints.
2. Noise removal and hole filling 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. Simplifying the final 3D model.
8. Construction of CAD model of the scene.

There has been significant progress in the area of 3D modeling from dense range scans. Typical 3D modeling systems have been developed by Allen and colleagues [9, 3], by Levoy's group [5], by the NRC's VIT group [2], by Bernardini and colleagues [4], by Zakhor's group [6], by Zhao's group [14], and by Sequira's group [10].

This paper presents a mesh-simplification method of the final 3D model (seventh task of a 3D modeling system) based on the segmentation results of each range image (third task). The range segmentation of each scan is based on our previous work presented in [12]. For the automated range-registration between the range scans (fourth task) we are using the methods developed by our group (see [13]). Finally, for the 3D modeling part (fifth and sixth task) we are using the algorithm of [5]. Note, that our 3D simplification method does not depend on the 3D modeling method used. 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. Our ultimate goal is the automated generation of a scene CAD model. The fact that we are relying on the original segmentation results for simplification increases the accuracy of our algorithms, since the final 3D model may diverge from the original scans due to mis-registrations or averaging.

2 Preprocessing and Hole Filling

Our goal is to retain from each range scan only the data that confidently substitute a large and connected cluster of range measurements. That means that we want to automatically remove objects such as cars, traffic lights, people, trees, etc. that occlude the actual measured object. Objects of this type have the following two properties:

1. They are small in size if they are compared with the large structures we measure.

2. They constitute unorganized point sets if we compare them with the regular structures we measure.

We wish to utilize these two properties in order to identify all external objects and remove them from the scene. In the main object scanned the distance between adjacent 3D points is not large (it is usually within a meter). Our clustering algorithm starts by collecting all adjacent 3D points that lie within a threshold distance from each other and thus form clusters of connected points in 3D space. We take advantage of the adjacency of 3D points, gathered in a regular two-dimensional grid by the range scanning equipment, in order to speed up the clustering process. Every point that does not belong to this cluster (that is it lies further than a distance $d_{thresh1}$ from it) is likely to belong to external objects. Therefore, we keep **only** the main cluster of 3D points. Since the threshold for forming these $d_{thresh1}$ clusters is relatively large (in the order of meters), it is still possible that the main cluster will contain noisy points which do not belong to the main surface that we want to scan. This is particularly true in the case of windows that appear as clouds of points with no clear underlying structure, sparse in space, but close enough to the main cluster to pass the previous filtering. However, what this windows points have in common is that they are pretty far from each other, so they are unlikely to represent consistent surfaces. Even when they do, the surfaces they cover are very small. Therefore, we re-cluster the main cluster by lowering the threshold to the order of centimeters (usually 10 cm). We believe that the new clusters will represent actual successfully scanned surfaces.

Our experiments showed that this approach works well and cleans the scanned building parts. We used constant thresholds for all scans of the Thomas Hunter Building. However, this approach does not handle the case of smaller objects that are connected to the main structure, but are not part of it (for instance flag-poles, etc.). The removal of such object requires manual intervention.

The next problem to attack is that of hole filling. Holes are created due to the following reasons:

1. Occlusion from other scene objects (i.e. trees, cars, etc.). The removal of these objects generates holes, i.e. missing range points.

2. Glass windows. The laser is not reflected by the glass. The result is that the laser beams pass through the glass and measure points in the interior of the buildings (ceilings, bookcases, etc.)[1].

In the case of windows the generated holes are interior, meaning that they are not connected to the exterior of the 3D range scans. It is easy then to find out which holes belong to windows by finding all interior holes. We can easily fill these holes by linear interpolation. Finally, after filling the window holes, we can put back the removed (due to clustering) points that are close enough to the filled part and then refill the windows by using the recovered points for interpolation. In this way we can recover points that belong to the window frames (or other thin elements) that were mistakenly removed due to the inherently sparse and thin structure of the frames.

3 3D Model Generation and Simplification

In this section we describe the algorithms used for the generation of a simplified 3D models starting from raw range scans. First we introduce notations that will be used throughout the document.

Each range scan R_i is represented as a two-dimensional array of 3D points $\{\mathbf{r}(k,l), k = 1 \ldots N, l = 1 \ldots M\}$[2]. The segmentation module [12] 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, that are abbreviated as $\mathbf{SPA}$. Each $\mathbf{SPA}$ is a bounded plane with exterior and possible interior (i.e. holes) borders (see Figures 1c through 1f for some examples). The j_{th} segmented planar area of scan S_i is represented with the symbol $\mathbf{SPA}_{i,j}$. For simplicity we drop the subscripts in the remaining of this document. We are using the variable name $\mathbf{F_s}$ for an arbitrary $\mathbf{SPA}$. Each $\mathbf{SPA}$ $\mathbf{F_s}$ contains the following information:

1. The infinite plane $P(\mathbf{F_s})$ where $\mathbf{F_s}$ lies (unit normal and position of the plane),

2. The set of 3D points (along with their indices) of scan S_i that lie on the $\mathbf{SPA}$ $\mathbf{F_s}$:
$\{\text{3D points } \mathbf{r}(k,l) \in R_i : \mathbf{r} \text{ lies on } \mathbf{F_s}\}$,

3. A sequence of indices (k,l) that traverse the outer boundary of $\mathbf{F_s}$, and

4. A sequence of indices (k,l) that traverse each inner boundary of $\mathbf{F_s}$.

After range scan acquisition and segmentation, all segmented range scans S_i are registered in the same frame of reference using the method described in [13]. This method automatically matches linear border features between pairs of segmented scans. After the registration process the 3D points $\mathbf{r}(k,l)$ of each $\mathbf{SPA}$ $\mathbf{F_s}$ are transformed to a common coordinate system (the indices (k,l) are not affected

[1] With high probability these elements (ceilings, etc.) will be removed in the preprocessing stage.

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

by this transformation). Finally the infinite plane $P(\mathbf{F_s})$ of each **SPA** $\mathbf{F_s}$ is updated and it is now expressed with respect to the common coordinate system. In the remaining of the document the symbols $\mathbf{r}(k,l)$ and $P(\mathbf{F_s})$ are expressed with respect to this common coordinate system.

A triangulated mesh surface M, that approximates the surface generated by the n registered scans R_i, is produced. We are using the method described in [5] (voxelization followed by the marching cubes algorithm) for the creation of the mesh. The mesh surface M consists of a set of 3D points $\mathbf{V} = \{\mathbf{v_m}, m = 1, \ldots, L\}$ along with connectivity information that generates a triangulation of the set $\mathbf{V}$. Note, that our simplification method is not based on the mesh generation method used.

Our goal is to identify a subset of the generated mesh vertices that lie on planar areas in the segmented range scans S_i. This subset of mesh vertices will allow us to simplify the reconstructed mesh. We thus want to identify the 3D points $\mathbf{v_m}$ of the mesh that lie on at least one **SPA** of the segmented scans S_i. Connected sets of planar 3D points (i.e. points that lie on at least one **SPA**) of the mesh along with their boundaries need to be computed. These sets are abbreviated as **MPA**. We use the variable name $\mathbf{F_m}$ to represent an arbitrary **MPA**. Thus $\mathbf{F_m} = \{\mathbf{v_{m1}}, \ldots, \mathbf{v_{mN}}\} \subset \mathbf{V}$.

Our final goal is to discard the interior points of the **MPA**s, and to keep only their boundary points. Note, that due to the fact that the mesh M is constructed by the integration of all range images, the 3D points $\mathbf{v_m}$ that define the triangles of the mesh M are not the exact same points $\mathbf{r}(k,l)$ that are measured in each individual raw range image R_i. The 3D points $\mathbf{v_m}$ are produced by the mesh generation algorithm that introduces some type of weighted averaging on the original 3D points $\mathbf{r}(k,l)$. Also the spatial resolution of the output mesh is not the same as the resolution of the raw range scans.

Before continuing we need to specify that a mesh point $\mathbf{p_m}$ on the final integrated mesh M is considered to lie on the **SPA** $\mathbf{F_s}$ of a scan S_i, if and only if the distance between $\mathbf{p_m}$ and the infinite plane $P(\mathbf{F_s})$ is within a distance threshold *Point-to-Plane-Dist* and the angle between the normal of the local mesh surface around $\mathbf{p_m}$ and the normal of the plane $P(\mathbf{F_s})$ is within an angle threshold *Point-to-Plane-Angle*. Moreover, there must exist at least one point $\mathbf{p_s}$ on the 3D segmented scan S_i that belongs to the **SPA** $\mathbf{F_s}$ such that the distance between the points $\mathbf{p_s}$ (on S_i) and $\mathbf{p_m}$ (on M) is within *Point-to-Point-Distance*.

A good initial estimation of the **MPA**s is to find all connected sets of 3D points from the mesh M that lie on one **SPA** of some 3D scan. However it is possible that one mesh point is contained in several **SPA**s. This could happen because these **SPA**s represent the same planar feature on the object scanned from different viewpoints (i.e. the **SPA**s are overlapping planar areas measured from different scans). That is why we need to keep track of the overlapping **SPA**s that give rise to a common mesh planar region **MPA** on M. We are providing the following definition:

Definition 1:

The **SPA**s $\mathbf{F_{s1}}$ and $\mathbf{F_{s2}}$ of scans S_1 and S_2 are part of a common **MPA** $\mathbf{F_m}$ on M if and only if:

1. The infinite planes of the segmented areas are the same, i.e. $P(\mathbf{F_{s1}}) = P(\mathbf{F_{s2}})$ and
2. There is at least one point $\mathbf{p_m}$ on the mesh M that lies on both $\mathbf{F_{s1}}$ and $\mathbf{F_{s2}}$, or lies on $\mathbf{F_{s1}}$ (or $\mathbf{F_{s2}}$) and on another **SPA** $\mathbf{F_{s3}}$, where $\mathbf{F_{s3}}$ and $\mathbf{F_{s2}}$ (or $\mathbf{F_{s1}}$) are both part of a common **MPA** according to Definition 1. [3]

In practice if $\mathbf{F_{s1}}, \mathbf{F_{s2}}, \ldots, \mathbf{F_{sk}}$ represent the *complete set* of **SPA**s that constitute the **MPA** $\mathbf{F_m}$, then the equality of their infinite planes has to be established within some threshold[4]. Also, there is a slight difference between the plane $P(\mathbf{F_m})$[5] and the average of the corresponding planes $P(\mathbf{F_{s1}}), P(\mathbf{F_{s2}}), \ldots, P(\mathbf{F_{sk}})$. This is due to the fact that the **MPA** $\mathbf{F_m}$ is formed by points from the mesh (reconstructed surface) while all **SPA**s matching that $\mathbf{F_m}$ contain points lying on the original raw scans. This difference can be the result of:

- Registration errors between scans, and
- The surface reconstruction algorithm used. All surface reconstruction algorithms do introduce some form of weighted averaging in order to represent overlapping cloud of points as one consistent mesh surface.

3.1 Estimation of Mesh Planar Areas (MPAs)

Input: a surface mesh M approximating an object O, a set of 3D scans R_i of the same object O, and the corresponding segmented 3D scans S_i (note that all scans are registered in the same coordinate system as mesh M).

Operation: All mesh planar areas of M are computed according to **Definition 1**.

Output: The output of the algorithm is a triangulated surface mesh which contains non-planar vertices (i.e. vertices not lying on any **SPA** of any scan S_i) unchanged. The planar vertices are partitioned into disjoint **MPA**s.

Initializations and notations: Let us call $\mathbf{G}$ the set of generated **MPA**s. Initially $\mathbf{G} = \emptyset$.

The function *getMPA* returns the mesh planar area $\mathbf{F_m}$ of mesh point $\mathbf{p_m}$. Thus, $\mathbf{F_m}$ =*getMPA*($\mathbf{p_m}$). This function returns **nil** if $\mathbf{p_m}$ does not lie on any mesh planar area.

Each **MPA** $\mathbf{F_m}$ contains a pointer to the list of overlapping segmented planar areas $(\mathbf{F_{s1}}, \mathbf{F_{s2}}, \ldots, \mathbf{F_{sk}})$ that generates it. Let us call this list $L(\mathbf{F_m})$. Whenever a new **MPA** is created this list is empty.

[3] In other words we are defining an equivalence relation $\mathcal{R}$ in the set of all **SPA**s of all scans, where two **SPA**s are equivalent if they lie on the same geometric plane and if at least one mesh point $\mathbf{p_m}$ lies on both **SPA**s. This equivalence relation partitions the set of all **SPA**s into subsets. There is an one-to-one correspondence between these subsets and the **MPA**s.

[4] Planes P_1 and P_2 are considered equal if and only if the distance between them is within *Plane-to-Plane-Dist* and the angle between their normals is within *Plane-to-Plane-Angle*.

[5] This is the plane generated by the 3D points of the **MPA** $\mathbf{F_m}$.

ALGORITHM: Consider a 3D point $\mathbf{p_m}$ on the mesh M and a 3D scan S_i.

1. If there is a new pair (point $\mathbf{p_m}$ from the mesh, scan S_i) go to step 2. Otherwise (have exhausted all pairs of mesh points and 3D scans) go to step 6.

2. Project the point $\mathbf{p_m}$ on the image bin of S_i through a ray that connects $\mathbf{p_m}$ with the center of projection of scan S_i. This ray hits the image bin at the position (i_0, j_0). Perform a local search on the image bin around (i_0, j_0) to find the closest point $\mathbf{p_s} = \mathbf{r}(i_{closest}, j_{closest})$ from the scan S_i to $\mathbf{p_m}$. Note that this search is really fast since the 3D points of each range scan are organized in a two-dimensional grid. If $\mathbf{p_s}$ is within *Point-to-Point-Distance* from $\mathbf{p_m}$ and it is part of some $\mathbf{SPA}$ $\mathbf{F_s} \in S_i$ then go to step 3. Otherwise (the point $\mathbf{p_m}$ is not part of any segmented planar area on S_i) go back to step 1.

3. If the mesh vertex $\mathbf{p_m}$ lies on $P(\mathbf{F_s})$ (within the thresholds mentioned before) go to step 4. Otherwise (that means that $\mathbf{p_m}$ does not lie on $\mathbf{F_s}$[6]) go back to step 1.

   Steps 2 and 3 insure that there exists at least one $\mathbf{SPA}$ $\mathbf{F_s}$ *in* S_i *on which* $\mathbf{p_m}$ *lies.*

4. Let $\mathbf{F_m}$ =*getMPA*($\mathbf{p_m}$).

   If $\mathbf{F_m}$ =**nil** then create a new $\mathbf{MPA}$ $\mathbf{F_m} = \{\mathbf{p_m}\}$ and add it to the set $\mathbf{G}$. Also initialize the list $L(\mathbf{F_m})$ of associated segmented planar areas to contain the $\mathbf{SPA}$ $\mathbf{F_s}$.

   If $\mathbf{F_m} \neq$**nil** then if the infinite planes of the associated $\mathbf{SPA}$s in the list $L(\mathbf{F_m})$ are not equal to $P(\mathbf{F_s})$, go to step 1 (we want to ensure that all the $\mathbf{SPA}$s in the list L lie on the same infinite plane). Otherwise go to step 5.

5. Add the point $\mathbf{p_m}$ to the $\mathbf{MPA}$ $\mathbf{F_m}$ and add $\mathbf{F_s}$ to the list $L(\mathbf{F_m})$. Go back to step 1.

6. (Optional) Fit infinite planes on the vertices of the $\mathbf{MPA}$s $\mathbf{F_m} \in \mathbf{G}$.

3.2 Growing and Merging $\mathbf{MPA}$s

At the previous section a complete set of **MPA**s is computed by calculating the overlaps of a number of **SPA**s on the mesh M according to **Definition 1**. The recovered **MPAs**, however, are usually proper subsets of the real mesh planar areas. Usually misclassified points lie around the boundaries of the computed **MPA**s. This problem arises because the planes found during segmentation are slightly smaller than the real planar regions. Also the thresholds chosen for segmentation of the raw range scans can cause the fragmentation of planar features into several neighboring planar pieces. This could cause the generation of several neighboring **MPA**s that actually belong to a common planar surface. Finally, the accuracy of the mesh model as well as the choice of modeling thresholds could greatly influence the recovery of **MPA**s.

By utilizing the borders of the $\mathbf{MPA}$s computed in the previous section, $\mathbf{MPA}$s grow, shrink, or merge according to the following rules. We are referring to 3D points that do not lie on any $\mathbf{MPA}$ as non-planar, and to points that lie on one $\mathbf{MPA}$ as planar. Let us consider all mesh edges $(\mathbf{p_{m1}}, \mathbf{p_{m2}})$:

1. If $\mathbf{p_{m1}}$ is non-planar, $\mathbf{p_{m2}}$ is planar, and $\mathbf{p_{m1}}$ lies on the infinite plane $P(\mathbf{p_{m2}})$ of $\mathbf{MPA}(\mathbf{p_{m2}})$ then $\mathbf{p_{m1}}$ is marked as planar and is added to the $\mathbf{MPA}(\mathbf{p_{m2}})$ ($\mathbf{MPA}$ is growing).

2. If both $\mathbf{p_{m1}}$ and $\mathbf{p_{m2}}$ are planar and their infinite planes are equal, i.e. $P(\mathbf{p_{m1}}) = P(\mathbf{p_{m2}})$ then $\mathbf{MPA(p_{m1})}$ and $\mathbf{MPA(p_{m2})}$ are being merged.

3. If both $\mathbf{p_{m1}}$ and $\mathbf{p_{m2}}$ are planar, but $\mathbf{MPA}(\mathbf{p_{m1}}) \neq \mathbf{MPA}(\mathbf{p_{m2}})$, and point $\mathbf{p_{m1}}$ (or $\mathbf{p_{m2}}$) is closer to the plane of the point $\mathbf{p_{m2}}$ (or $\mathbf{p_{m1}}$ respectively) then the point $\mathbf{p_{m1}}$ (or $\mathbf{p_{m2}}$ respectively) is removed from its $\mathbf{MPA}$ and it is added to the $\mathbf{MPA}$ of the other point (shrinking and growing).

This loop is repeated through all edges until no change takes place.

3.3 Finding the Borders of $\mathbf{MPA}$s

Once all planar points are grouped into $\mathbf{MPA}$s (at sections 3.1 and 3.2) we are ready to simplify the mesh. This can be achieved by finding the borders of the $\mathbf{MPA}$s. The knowledge of border points allow us to remove points that are interior in the $\mathbf{MPA}$s. In this way we should are able to recover the $\mathbf{MPA}$s' interior surface without any loss of detail.

These borders must be simple cycles that contain every point exactly once as they are traversed in a particular direction. What we do not know at this point is whether the $\mathbf{MPA}$s found at the previous steps have simple cycled borders. The borders of an $\mathbf{MPA}$ $\mathbf{F_m}$ consist of all the points $\mathbf{v_m} \in \mathbf{F_m}$ that are connected through an edge with a point $\mathbf{v'_m} \notin \mathbf{F_m}$. Now, the borders of $\mathbf{F_m}$ are simple cycles if all border points have exactly two other border points of the same kind as neighbors. Two points are of the same kind if they are both non-planar or if they both belong to the same $\mathbf{MPA}$. The running time of this step (identifying border points) is O(N), where N is the total number of points.

3.4 Elimination of Border Points

After finding the neighbor border points for every border point we look for those with more or with less than two neighboring border points and we mark them as non-planar points (our goal is to eliminate points that introduce self-intersecting borders). By eliminating them from the set of planar points (thus eliminating them from their respective

[6] Due to averaging the mesh point $\mathbf{p_m}$ has moved far away from the raw range points.

$\mathbf{MPA}$) we may eliminate or introduce other border points. That is why after finishing this step we have to relabel all border points with exactly two neighbors and repeat the step described in section 3.3. This step is performed only if the number of planar points decreases. Since this number is finite the loop between the previous step (3.3) and the current step (elimination of border points) will terminate. In practice it terminates after a few iterations and the number of planar points is not significantly reduced. The running time of this step is also O(N).

3.5 Finding Clusters of Connected Planar Points

At this stage all border points are connected to exactly two neighboring border points, meaning that all borders are non-intersecting cycles. If we want to find the border on which a particular border point lies all we have to do is to start from it and move along the border by going to its neighbor border points, while making sure that we never return to the previous border point. The cycle terminates when the starting point is reached.

Two types of borders exist: interior and exterior. Interior are the borders that surround non planar regions (i.e. they define a hole in the $\mathbf{MPA}$), while exterior borders are the ones that surround planar areas. It is possible to have an $\mathbf{MPA}$ with more than one exterior border that surround a number of interior ones. For this reasons we need to subdivide the $\mathbf{MPA}$s in mesh planar faces, named $\mathbf{MPF}$s, that satisfy the additional property that they have exactly one exterior border and a number (≥ 0) of interior borders.

The computation of the $\mathbf{MPF}$s from the $\mathbf{MPA}$s is done by partitioning the $\mathbf{MPA}$s into connected components, based on the following definition of connectivity. Two mesh points $\mathbf{p_{m1}}$ and $\mathbf{p_{m2}}$ belong to the same mesh planar face $\mathbf{MPF}$ $\mathbf{P_f}$ if and only if:

1. $\mathbf{p_{m1}}$ and $\mathbf{p_{m2}}$ lie on the same $\mathbf{MPA}$, and
2. If there is a direct edge between $\mathbf{p_{m1}}$ and $\mathbf{p_{m2}}$ then the points must lie on at least one triangle with all its three vertices belonging to the same $\mathbf{MPA}$, and
3. If $\mathbf{p_{m1}}$ and $\mathbf{p_{m2}}$ are not directly connected by one edge then there must be at least one point $\mathbf{p_{m3}} \in \mathbf{P_f}$ such that the $(\mathbf{p_{m1}}, \mathbf{p_{m3}})$ and $(\mathbf{p_{m2}}, \mathbf{p_{m3}})$ belong to the same $\mathbf{P_f}$.

The running time of this step is O(N), where N is the total number of mesh points.

3.6 Combining Exterior and Interior Borders

For any extracted mesh planar face it is possible to connect all its exterior and interior borders into one simple non-self-intersecting cycle. Traverse the borders of each mesh planar face as follows:

1. Traverse the exterior border clockwise and jump from the exterior border point $\mathbf{p_j}$[7] to the closest point of the closest interior border. Starting from that point on the interior border traverse the border counterclockwise and stop at the point right before the starting point. Then jump back to the exterior border at the point right after $\mathbf{p_j}$. Finally, continue traversing the exterior border clockwise until the start is reached. Make this cycle the new exterior border and remove the closest interior border from the set of interior borders. Go to step 2.
2. Now the closest interior border is part of the exterior border. Repeat step 1 if there is any interior borders left. If there is none, stop.

In this way all borders (interior and exterior) are grouped into one simple cycled border corresponding to their planar face. The algorithm guarantees that the final border will be a simple cycle which does not cross itself. This border can be easily triangulated using Open Inventor libraries [1]. A small problem remains: at the point of jump from the exterior to the interior, a small rectangular area will be left uncovered. Since we know exactly where we jump from border to border, we can cover these small holes by creating separately their corresponding triangulated rectangles.

4 Results

Our algorithms have been tested on a dataset of 17 range scans of the Thomas Hunter Building in New York City. This is an intersecting building with a number of planar and non-planar areas. It is a building located on a densely populated urban area. That means that holes due to people, moving traffic, or traffic lights need to be filled. Some of these holes are filled by range scans acquired from different viewpoints.

The range scans were acquired with a Leica Geosystems Cyrax 2500 laser range scanner. Each range scan consists of one million 3D points. Figure 1a shows a photograph of the building, and Figure 1b shows a dense 3D mesh created from one range scan (north view). Note that such a 3D mesh is very heavy for efficient rendering due to the fact that all points are used. Note that some of these points do not add anything to rendering quality since they are interior to a perfectly flat surface. Rendering of all 17 scans without simplification is computationally prohibitive.

The next step is the segmentation of each range scan into segmented planar areas ($\mathbf{SPA}$s) using our previously developed algorithms [11, 12]. This step generates the $\mathbf{SPA}$s along with their border points for each scan. Four segmented range scans are shown in Figures 1c-1f. Each $\mathbf{SPA}$ is displayed with different color for clarity. Interior borders in each $\mathbf{SPA}$ are shown in blue.

The next step is to automatically register all 17 range scans using the method developed by our group (see [13]). The borders of segmented planar areas from different scans are automatically matched and all scans are placed on a common frame of reference. Each raw scan is then preprocessed for hole filling and noise removal (see section 2). Finally the method developed by Curless and Levoy (see [5]) is being used for the construction of the final mesh M. The 3D space is partitioned into voxels of volume 0.15 x 0.15 x 0.15

[7] The point $\mathbf{p_j}$ is the one closer to the closest interior border.

meters. A finer final mesh can be achieved by increasing the granularity of the voxelization. The 3D points from each range scan are deposited into voxels and a final mesh surface surface is being generated by the marching cubes algorithm [8]. Two views of this mesh are shown in Figures 2a and 2b. The mesh has the appearance of a solid model due to the hole filling process (see section 2). The interior of the windows have been filled by linear interpolation in each raw range scan; the range sensor does not provide any measurements on glass. Note that due to the weighted averaging of 3D range points at each voxel, areas of sharp discontinuities in the original range scans appear blurred in the final model.

The algorithm described in section 3 is executed next. The mesh planar areas (**MPA**s) are identified on the generated mesh M by the overlaps of the segmented planar areas. The borders of the areas are computed and connected components of co-planar points are generated on the mesh. Finally, points interior to the **MPA**s are discarded, and only the outer borders are kept. The generated **MPA**s combined with the unsegmented part of the mesh is shown in Figures 2c and 2d. These **MPA**s describe connected planar areas on the final mesh. Finally, Figures 2e and 2f show the final model after simplification, i.e. after the elimination of the interior points of the **MPA**s. The simplified 3D model appears extremely similar to the original mesh surface. This is due to the fact that only actual planar areas are simplified. This decision is based on the segmentation of the original raw scans, free of modeling or registration errors. Note that the resolution of the **MPA**s is coarser than the resolution of the segmented planar areas. This loss in resolution is due to the discretization of the 3D space needed by the mesh generation algorithm.

Out of a total of 248,119 points on the generated mesh M, 31,037 points lie on the borders of **MPA**s for our dataset. That means that 12% of all points lie on borders of planar areas. The simplification process takes almost five minutes for our dataset of 17 scans on a 2.8 GHz Xeon processor - 2 Gbytes RAM PC. The *Point-to-Point-Distance* threshold used was 0.15 meters, while the *Point-to-Plane-Distance* threshold was 0.1 meters, and the *Point-to-Plane-Angle* threshold was 10 degrees. The *Plane-to-Plane-Dist* threshold was 0.05 meters, and the *Plane-to-Plane-Angle* threshold was 4 degrees. For more information about the thresholds please see section 3.

5 Conclusions

This paper presents a mesh-simplification method that is integrated in a complete system for the generation of photo-realistic models of urban environments. Our main contribution is the utilization of planar segments in the model creation process. We not only want to produce a simplified version of the original mesh (for one of the many approaches in this direction see [7]) but to also generate the actual architectural structure of the buildings. Our goal is to automatically produce the CAD model of a structure from range measurements. This work brings us closer to this goal.

References

[1] SGI Open Inventor Library. http://oss.sgi.com/projects/inventor/.

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

[3] 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.

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

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

[6] 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.

[7] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH*, 1997.

[8] W. E. Lorensen and H. E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

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

[10] V. Sequeira and J. Concalves. 3D reality modeling: Photo-realistic 3D models of real world scenes. In *IEEE Int'l Symposium in 3D Data Processing Visualization and Transmission*, pages 776–783, 2002.

[11] I. Stamos and P. K. Allen. 3D model construction using range and image data. *IEEE Conf. Computer Vision and Pattern Recognition*, pages 531–536, June 2000.

[12] 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.

[13] I. Stamos and M. Leordeanu. Automated feature-based range registration of urban scenes of large scale. In *IEEE Conf. Computer Vision and Pattern Recognition*, volume II, pages 555–561, Madison, WI, June 2003.

[14] H. Zhao and R. Shibasaki. Reconstructing a textured CAD model of an urban environment using vehicle-borne laser range scanners and line cameras. *Machine Vision and Applications*, 14(1):35–41, 2003.

Figure 1: **[TOP ROW] (a) Color Image of Thomas Hunter Building, New York City. (b) 3D mesh from raw range scan (one million 3D points) of the north view of the building. Pseudo-color corresponds to the intensity of the returned laser-beam. [MIDDLE ROW] (c) & (d) Segmented planar areas (SPAs) from two range scans (south and north views). Different colors correspond to different planes. The thick blue line that penetrates the scans represents the scanning direction. [BOTTOM ROW] (e) & (f) Segmented planar areas (east views) from two range scans. For color images please visit** `www.cs.hunter.cuny.edu/~ioannis`**.**

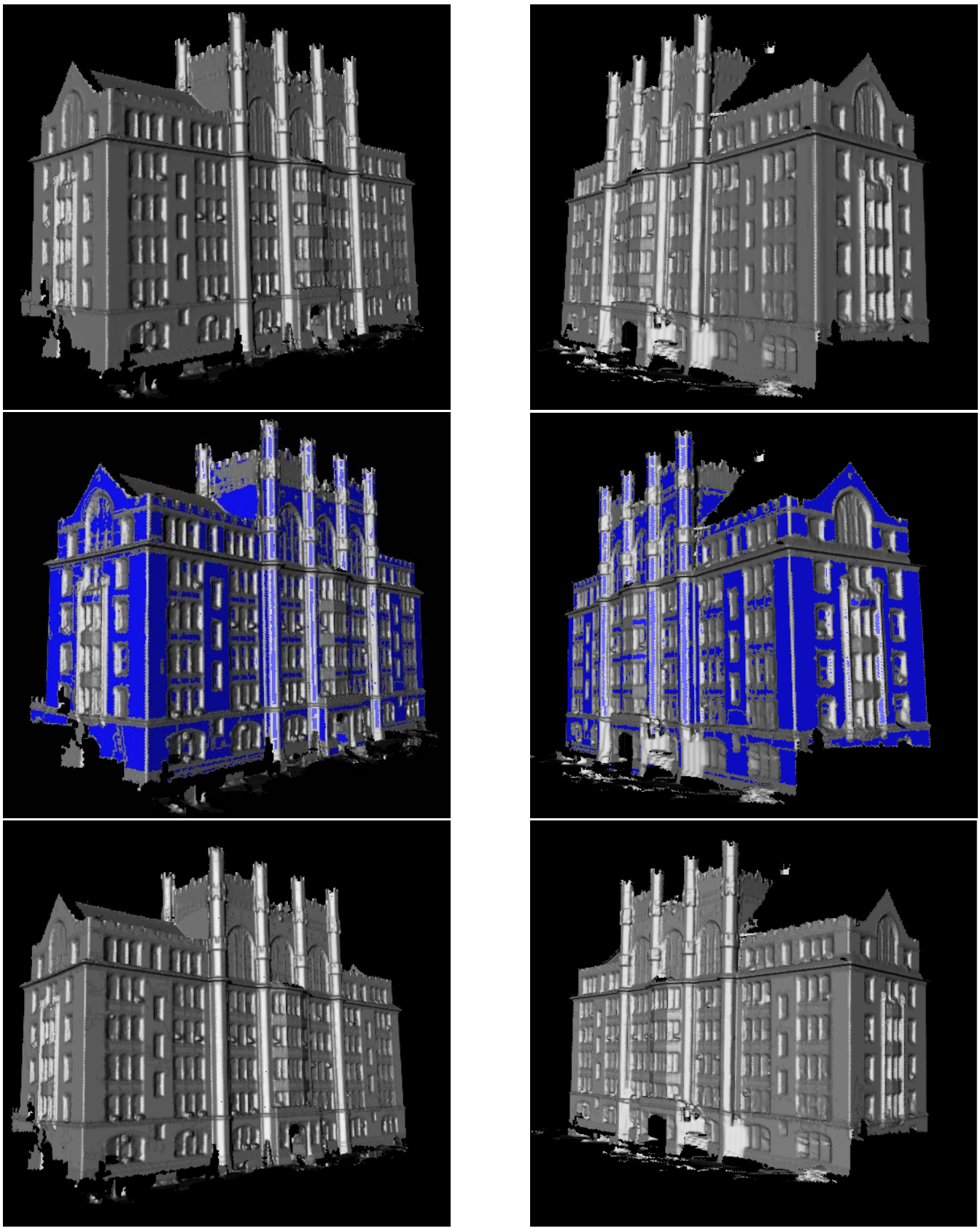

Figure 2: **[TOP ROW] (a) & (b) North and south views of the mesh model of the Thomas Hunter Building in New York City. The model was generated with the method of [5]. [MIDDLE ROW] (c) & (d) North and south views of the same model with mesh planar areas $\mathrm{MPA}s$ identified as blue. [BOTTOM ROW] (e) & (f) 3D model after simplification.**