Computer Vision and Image Understanding **88,** 94–118 (2002)
doi:10.1006/cviu.2002.0963

Geometry and Texture Recovery of Scenes of Large Scale

Ioannis Stamos

Computer Science Department, Hunter College and Graduate Center, City University of New York, New York, New York

and

Peter K. Allen

Computer Science Department, Columbia University, New York, New York
E-mail: istamos@hunter.cuny.edu, allen@cs.columbia.edu

Received March 16, 2002; accepted May 31, 2002

This paper presents a systematic approach to the problem of photorealistic 3-D model acquisition from the combination of range and image sensing. The input is a sequence of unregistered range scans of the scene and a sequence of unregistered 2-D photographs of the same scene. The output is a true texture-mapped geometric model of the scene. We believe that the developed modules are of vital importance for a flexible photorealistic 3-D model acquisition system. Segmentation algorithms simplify the dense datasets and provide stable features of interest which can be used for registration purposes. Solid modeling provides geometrically correct 3-D models. Finally, the automated range to an image registration algorithm can increase the flexibility of the system by decoupling the slow geometry recovery process from the image acquisition process; the camera does not have to be precalibrated and rigidly attached to the range sensor. The system is comprehensive in that it addresses all phases of the modeling problem with a particular emphasis on automating the entire process interaction.

1. INTRODUCTION

The recovery and representation of 3-D geometric and photometric information of the real world is one of the most challenging and well-studied problems in computer vision and robotics research. 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

1077-3142/02 $35.00

reconstructing models of outdoor urban environments [29]. 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, and 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 databases for GIS (geographical information systems), and a combination of reconstructed areas with existing digital maps.

Recent developments in range sensing have made possible the acquisition of accurate 3-D scans of outdoor scenes. Range sensors have proven their effectiveness in controlled laboratory environments (e.g., [16, 44]). Taking these methods out of the controlled laboratory environment and using them on large geometrically complex outdoor scenes poses many difficult challenges. In this paper, we present an integrated system for creating geometrically and photometrically correct 3-D models of large outdoor structures.

The problem we attack can be described as follows: Given a set of dense 3-D range scans of a complex real scene from different viewpoints and a set of 2-D photographs of the scene, (a) create the 3-D solid model which describes the geometry of the scene, (b) recover the positions of the 2-D cameras with respect to the extracted geometric model, and (c) photorealistically render the scene by texture-mapping the associated photographs on the model.

The integrated system we developed for the production of photorealistic geometric models of large and complex scenes is described in Fig. 1. We start with a set of range and brightness images which cover the measured site. The range images are first segmented, and 3-D features of interest are extracted (Section 3). After all range images are expressed in the same coordinate system a volumetric solid geometric model which expresses the geometry of the scene is computed (Section 4). Finally, the relative positions of the brightness

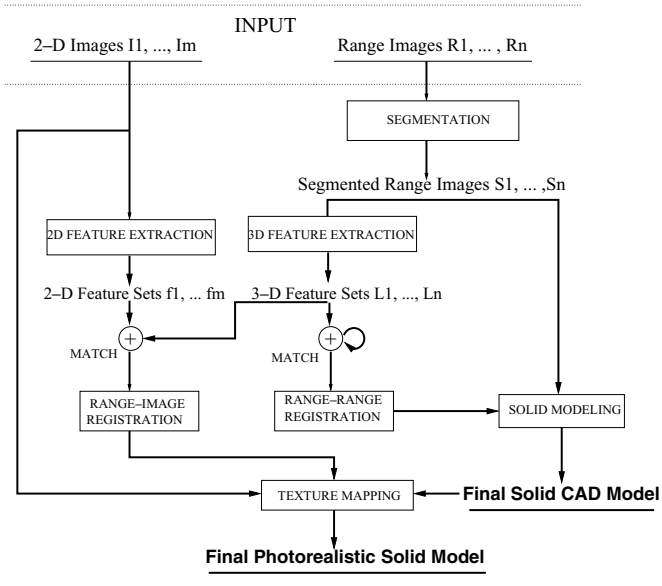

FIG. 1. System for building geometric and photometric correct solid models.

cameras with respect to the 3-D model are automatically recovered and the photographs are mapped on the geometric model in order to provide a photorealistic view of the scene (Section 5).

2. RELATED WORK

There are two major approaches in the photorealistic reconstruction of urban 3-D scenes: purely geometric (extraction of dense geometry via range sensing or extraction of sparse and irregular geometry via stereo techniques) and image-based rendering methods (extrapolating geometry in the rendering phase via resampling the captured light field of the scene). Representative systems whose goal is the photorealistic reconstruction of real scenes by the utilization of 2-D imagery only are [6, 17, 46]. In those cases the necessary human interaction and the a priori geometric constraints imposed by the human operator lead to a lack of scalability with respect to the number of processed images of the scene and to the computation of simplified geometric descriptions of the scene. Teller's approach [52] addresses the limitations of the previously described methods by acquiring and processing a large amount of pose-annotated high-resolution spherical imagery of the scene. The end result consists of vertical facades with associated textures [15] along with relief estimation [51]. The automatic computation of the transformation between nearby mosaics [2, 3] is achieved. The whole project is very promising; methods based on images alone, however, are not able to capture highly detailed architectural environments.

Systems which extract dense and regular geometry *must* rely on accurate range measurments. Representative approaches include the VIT group [7, 20, 54], the Digital Michelangelo project [37], the Pietá project [8], Fitzgibbon *et al.* [23], Zhao [57], and Sequiera [45].

Zisserman's group in Oxford [24] works toward the fully automatic construction of graphical models of scenes when the input is a sequence of closely spaced 2-D images (video sequence). Their system couples the matching of 2-D point features in triples of consecutive images with the computation of the fundamental matrices between pairs of images and trifocal tensors between triples of images (projective reconstruction). This work shows how far purely image-based methods have gone but also points out the following inherent limitations: (a) Sparse depth estimates which depend on the texture and geometric structure of the scene, and (b) The resulting CAD model which is a crude approximation in the areas which do not support 3-D measurements. In more recent work of the same group, a priori knowledge regarding the scene is utilized in the 3-D extraction phase [4, 38].

3. SEGMENTATION AND 3-D FEATURE DETECTION

The individual range images which the range sensor provides are the result of dense sampling of visible surfaces in large urban scenes. Using a Cyrax laser scanner, we get 1K by 1K range samples ($\sim$1 million range samples) with a spatial resolution of a few centimeters. Smoothly varying parts of the scene (e.g., planar or cylindrical surfaces) are sampled with the same rate as nonsmooth surfaces (e.g., parts of the scene with orientation discontinuities). If we are able to identify those smoothly varying parts then we can represent them with a fewer number of parameters. In this section, we formulate the range

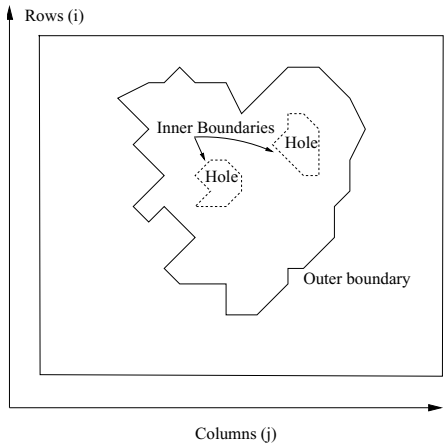

FIG. 2. Cluster boundaries defined as sequence of points on the rectangular grid over which the range image is defined.

segmentation and 3-D feature detection problems. Further details on these methods can be found in [48, 49].

3.1. Segmenting Dense Range Data

We follow the formulation introduced by [9]. Our goal is to segment the range image $\{r(i, j), i = 1 \dots N, j = 1 \dots M\}$[1] into a set of clusters $\{C_{null}, C_1, \dots, C_n\}$. Each cluster $C_i, i \geq 1$ is defined over a connected domain $\{r(i, j)\}$ of 3-D points and it corresponds to a smoothly varying surface S_i of the object. Also, no two clusters overlap; that is, $C_i \cap C_j = \emptyset, \forall i, j : i \neq j$. Finally the number of points that support each cluster is larger than a user defined threshold T_{size}. The special symbol C_{null} corresponds to the cluster of 3-D points which cannot be classified to any surface.

Each cluster $C_i, i \geq 1$ is represented by a set of parameters $\mathcal{P}(S_i)$ which define the surface S_i of infinite extent where the points of the cluster lie and by the sequence of those range points $\{r(i, j)\} \subset C_i$ which define the outer and inner boundaries of the cluster (see Fig. 2). A range point belongs to the boundary of the cluster if at least one of its 8-neighbors in the 2-D grid is not a member of the cluster. The outer boundary $\mathcal{O}(S_i)$ is the one that encloses all range points of the cluster, whereas the inner boundaries $\mathcal{I}_k(S_i)$ are holes inside the cluster. So, a cluster is represented as follows:

$$C_i = (\mathcal{P}(S_i) \mid \mathcal{O}(S_i), \mathcal{I}_0(S_i), \dots \mathcal{I}_K(S_i)).$$

Our final goal is to extract 3-D curves of finite extent at the intersections of adjacent surfaces S_i. That is, our goal is to generate a list of curves

$$\mathbf{L} = \{\mathcal{L}(S_{i0}, S_{j0}), \mathcal{L}(S_{i1}, S_{j1}), \dots, \mathcal{L}(S_{iW}, S_{jW})\}.$$

The symbol $\mathcal{L}(S_{iK}, S_{jK})$ corresponds to the curve of finite extent which is the intersection of the surfaces S_{iK} and S_{jK}. Those surfaces have defined boundaries according to the formulation described above.

[1] The indices i, j define the position and orientation of the laser-beam which produces the 3-D point $r(i, j)$.

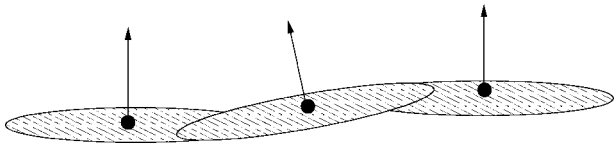

FIG. 3. Neighborhood of points inside a cluster. The local planes fit around each point are very close with respect to each other.

The outline of our segmentation algorithm is the following:

Point classification: A local plane is fit in the $k \times k$ neighborhood of every 3-D point. The normal $\mathbf{n_p}$ of the computed plane corresponds to the smallest eigenvector of the 3 by 3 matrix $A = \Sigma_{i=1}^{N}((\mathbf{v_i} - \mathbf{m})^{\mathrm{T}} \cdot (\mathbf{v_i} - \mathbf{m}))$ where $\mathbf{m}$ is the centroid of the set of $k \times k$ vertices $\mathbf{v_i}$. The smallest eigenvalue (least-squares fit) of the matrix A expresses the deviation d of the points $\mathbf{v_i}$ from the fitted plane; that is, it is a measure of the quality of the fit. If the fit is acceptable the point is classified as *locally planar* (when the deviation d is below a user-specified threshold P_{thresh}); otherwise is classified as *nonplanar*. Finally if the number of sensed points in the $k \times k$ neighborhood is not enough to produce a reliable fit the point is classified as *isolated*.

Cluster initialization: Create one cluster for every *locally planar* point.

Cluster merging: Merge clusters from the initial cluster list. Each cluster in the final list is defined as a set of 3-D points which are connected and lie on the same surface. Formally, the points of every final cluster represent segmented surfaces that can be approximated by small local patches of similar position and orientation at the point-level. That is, the local planes of every pair of neighboring points are very close with respect to each other (see Fig. 3).

Surface fit: Fit a plane (using the same algorithm described in the point classification phase) to the points of each cluster.

Boundary extraction: Extract the boundaries of each cluster.[2] Each boundary is thus a sequence of 2-D grid points: $\mathbf{B} = \{(i_0, j_0), (i_1, j_1), \ldots, (i_m, j_m)\}$. The actual 3-D boundary is just $\mathbf{B3D} = \{r(i_0, j_0), r(i_1, j_1), \ldots, r(i_m, j_m)\}$ where $r(i, j)$ is the 3-D position which corresponds to the grid point (i, j). We are also computing the 3-D axis-aligned bounding box **BOUND3D** of the set **B3D**. The bounding box is used for a fast estimation of the extension of the 3-D boundary in space and is used in the algorithms for 3-D line extraction (Section 3.2).

The *merging* of neighboring clusters is driven by a metric of conormality and coplanarity of two planar patches. Two adjacent *locally planar* points are considered to lie on the same planar surface if their corresponding local planar patches have similar orientation and are close in 3D space. Figure 4 displays two local planar patches which have been fit around the points P_1 and P_2 (point classification). The normals of the patches are $\mathbf{n_1}$ and $\mathbf{n_2}$, respectively. The points P_i' are the projections of the points P_i on the patches. The two planar patches are considered to be part of the same planar surface if the following two conditions are met:

Conormality measure: The first condition claims that the patches should have identical orientation (within a tolerance region); that is, the angle $\alpha = \cos^{-1}(\mathbf{n_1} \cdot \mathbf{n_2})$ is smaller than a threshold α_{thresh}.

[2] A range point belongs to the boundary of the cluster if at least one of its 8-neighbors is not a member of the cluster.

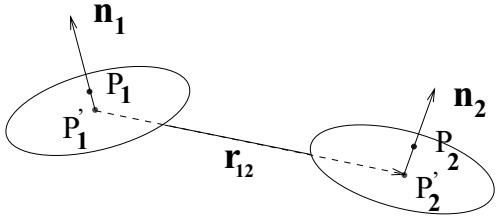

FIG. 4. Coplanarity measure. Two planar patches fit around points P_1 and P_2 at a distance $|\mathbf{r_{12}}|$.

Coplanarity measure: The second condition is that the patches lie on the same infinite plane. The distance between the two patches is defined as $d = \max(|\mathbf{r_{12}} \cdot \mathbf{n_1}|, |\mathbf{r_{12}} \cdot \mathbf{n_2}|)$, where $\mathbf{r_{12}}$ is the vector connecting the projections of P_1 and P_2 on their corresponding local planes (see Fig. 4). This distance should be smaller than a threshold d_{thresh}.

Thus, we defined the predicate *CoPlanar* $(P_1, P_2 \mid \alpha_{thresh}, d_{thresh})$ which decides whether two points P_1 and P_2 could be part of the same planar surface within a tolerance defined by the thresholds α_{thresh} and d_{thresh}.

The cluster-merging is a sequential-labeling algorithm [5] of the 3-D points into 8-connected regions, where the metric of similarity between neighboring points is the predicate defined in the previous paragraph. This algorithm has complexity $O(N)$ where N is the total number of range points (in our experiments $N \sim 10^6$).

The segmentation algorithm is automatic. The user, though, has to provide three thresholds. The threshold P_{thresh} is used to decide whether a 3-D point can be part of a smoothly varying surface (*Point Classification* phase). Points, whose local planes produce fitting errors[3] above this threshold, are consider to lie on a local discontinuity. Thus, they are rejected from further consideration. The thresholds α_{thresh} and d_{thresh} are used to decide whether two planar patches have similar orientation and position in space (*cluster-merging* phase). Local patches whose angular distance is greater than α_{thresh} or whose positional distance is greater than d_{thresh} are not considered to be parts of the same smoothly varying surface. Thus, by changing those thresholds we are able to achieve different segmentation results.

3.2. Extracting 3-D Linear Features from Segmented Range Images

At a second level of abstraction the 3-D range dataset is represented as a set of 3-D curves. Those curves are the result of an intersection of neighboring bounded 3-D surfaces which have been extracted by the range segmentation module. We implemented the extraction of 3-D lines as a result of planar surface intersections. Those 3-D features are used for the registration between 3-D datasets and between 3-D datasets and 2-D images.

The extraction of 3-D lines involves three steps:

1. Intersection of neighboring 3-D planes to produce 3-D lines of infinite extent[4] (Fig. 5a).

2. Verification of the infinite 3-D lines. This step involves the computation of the distance between the bounded surfaces and the produced 3-D line (Fig. 5b).

[3] Minimum least square errors.

[4] A measure of closeness which utilizes the 3-D bounding boxes of the planes is used.

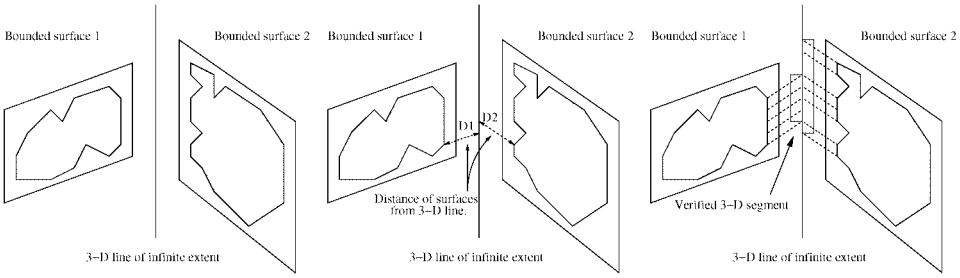

FIG. 5. (a) Infinite 3-D line as intersection of neighboring planes, (b) distance of bounded surfaces from line, and (c) verified 3-D linear segment.

3. Generation of 3-D linear segments out of the infinite 3-D lines. This is done by keeping the parts of the infinite 3-D lines which are verified by the range dataset (Fig. 5c).

Thus the 3-D line extraction algorithm works as follows: For every pair (S_i, S_j) of neighboring 3-D surfaces generate the infinite 3-D lines $L(S_i, S_j)$ as the intersection of those surfaces. Then verify the existence of that line by computing its distance from the two surfaces S_i and S_j. Finally, extract the verified 3-D linear segment $LS(S_i, S_j)$ out of the infinite 3-D line (for more details see [47, 48]).

3.3. Segmentation Results

The segmentation algorithms have been tested on range scans of urban structures. We have chosen four buildings. Two of them, the Casa Italiana and Teacher's College Building, are part of the Columbia University campus in New York City and are typical urban structures. These are buildings with planar façades and regular patterns of windows and doors. We also scanned the front of the Guggenheim Museum in New York City,[5] a one-of-a-kind building with conical façades. Our final scanned building is the Flat-Iron Building, a trademark of New York's early 20th century architecture.

The range and segmented scan of one view of Casa Italiana (a photograph of the building can be seen in Fig. 18a) and segmented scans of two additional views of the same building are shown in Fig. 6. Range and segmented scans of Teacher College are shown in Fig. 7, whereas photographs and range and segmented scans of the Guggenheim museum and of the Flat-Iron Building are shown in Figs. 8 and 9, respectively. Each segmented surface is displayed with a different color. The points which failed the initial *Point Classification* step are displayed in red.

The segmentation algorithm correctly extracts planar regions. In the case of Casa Italiana, all major walls have been extracted as well as small bricks and window borders. The same is true in the case of Teachers College, where we are able to extract parts of the roof and window shades. The first view of the Flat-Iron Building has been segmented into two major planar regions and a large number of window borders have been identified. In the second view one major wall has been extracted. Finally, in the case of the Guggenheim Museum the segmentation algorithm is able to extract conical façades. Thus, the algorithm can extract slowly varying smooth surfaces and not exclusively planes. This is because in the Cluster-Merging phase we are using a local region-growing decision which does not force the extracted regions to lie on a plane.

[5] Designed by Frank Lloyd Wright, one of the most famous architects of the 20th century.

FIG. 6. Casa Italiana. Top row: Range and segmented images (first view). Bottom row: Segmented images (second and third view). Each segmented surface is displayed with different color. The points which failed the initial *Point Classification* step are displayed in red (the large segmented wall on the second view is not red). Color versions of all images of this paper can be found at http://www.cs.columbia.edu/robotics/CVIU.

Figure 10 displays the results of the line extraction algorithm for the first views of the Casa Italiana and Teachers College. It is clear that major linear features (borders of large walls) as well as borders of windows have been extracted correctly.

The segmentation algorithms are very efficient. The surface extraction algorithm has a complexity of $O(N)$ where N is the total number of range points. The feature extraction algorithm has a complexity of $O(M^2)$ where M is the number of extracted 3-D planes.

Once we have extracted the segmented surfaces from the original dense range scan, we can use them as input to the full 3-D CAD modeling phase of the system (Section 4). We can also use the extracted 3-D linear features for range-to-range and range-to-image registration (Section 5).

4. SOLID MODELING

In general, 3-D modeling systems are based either on mesh-based or on volumetric approaches. In the mesh-based approaches, each range image is transformed into a mesh of

FIG. 7. Teachers College. Range (top) and segmented (bottom) image of the scene. Each segmented surface is displayed with a different color. The points which failed the initial *Point Classification* step are displayed in red.

triangular faces and all range images are merged by averaging surface elements on the mesh level. The final result is a triangular mesh which approximates the outer surface of the sensed object ([53] is a representative approach). Volumetric approaches, on the other hand, combine individual range images into a 3-D volume. The underlying representation

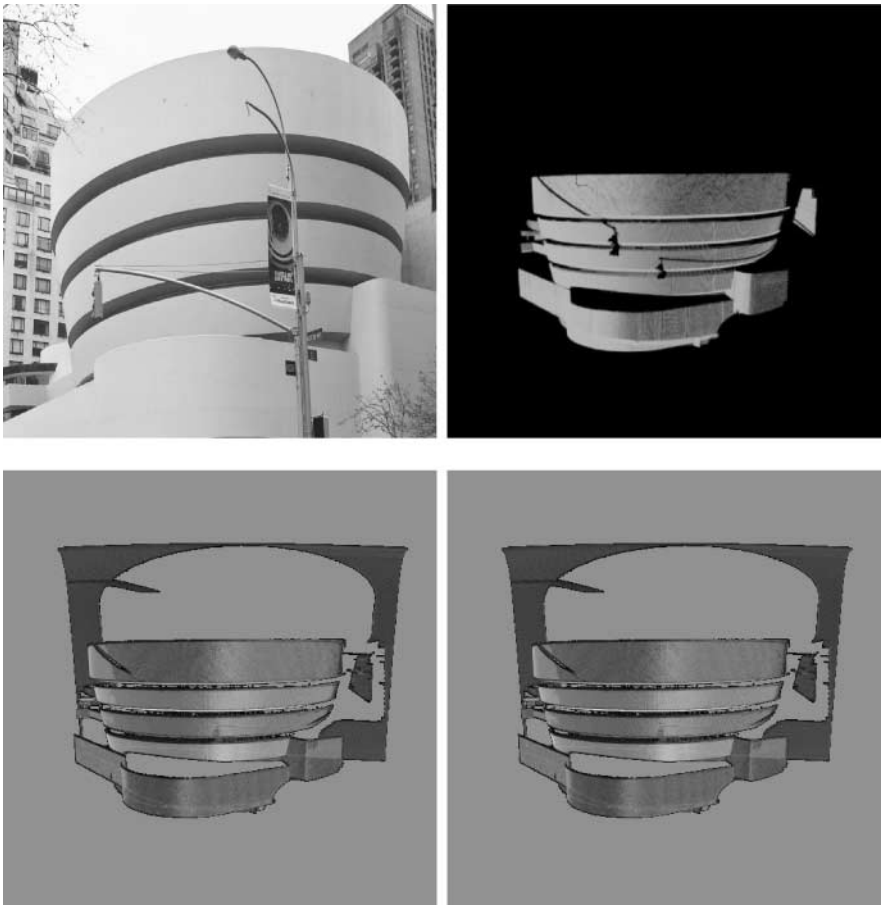

FIG. 8. Guggenheim Museum. Top row: Two-dimensional photograph and dense range scan (first view). Bottom row: Segmented range scans (first and second view). Each segmented surface is displayed with a different color. The points which failed the initial *Point Classification* step are displayed in red.

is the 3-D volume which approximates the actual volume the sensed object occupies (see [16, 43] for representative approaches). Volumetric approaches are considered superior to mesh-based methods, since they can model and fill holes in the final models. Holes can destroy the photorealistic appearance of the scene and thus are highly undesirable.

Our solid modeling system involves the following:

1. Individual range-image acquisition from different viewpoints.
2. Registration of all images into a common frame of reference.
3. Transformation of each range image into an intermediate volumetric-based representation using a sweeping operation.
4. Merging of all swept range images into a common representation.

4.1. Modeling System

We have extended the modeler which is based on an earlier work by Reed and Allen [43]. The innovative principle of this approach is the representation of each individual range image

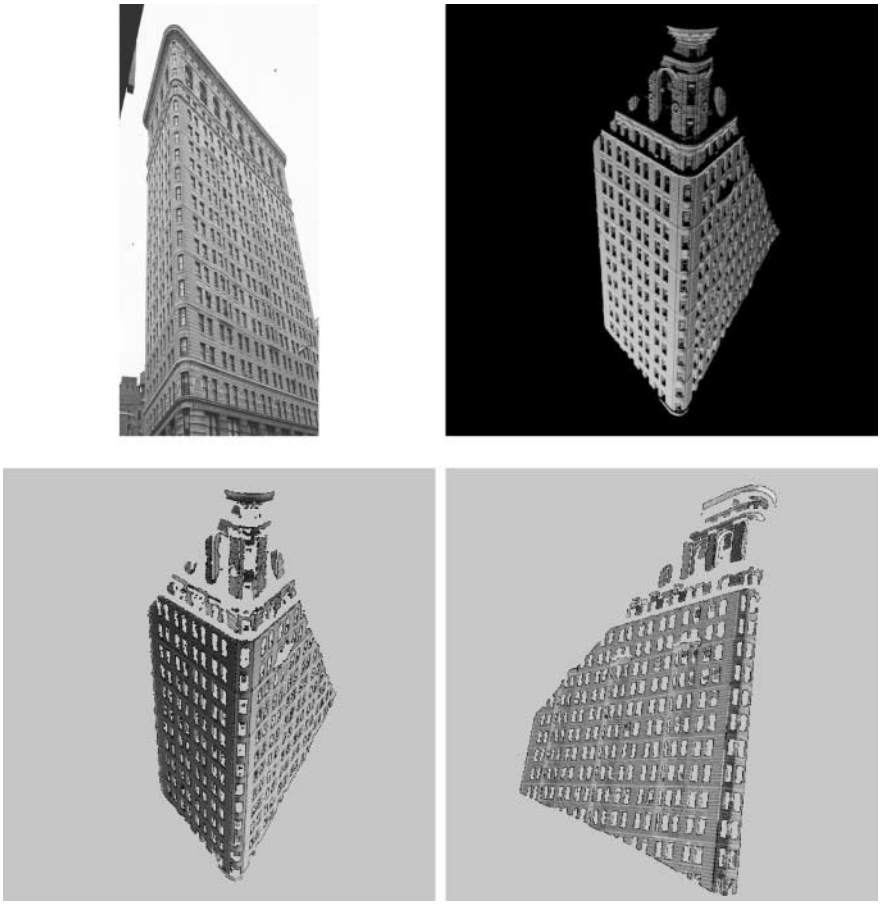

FIG. 9. Flat-Iron Building. Top row: Two-dimensional photograph and dense range scan (first view). Bottom row: Segmented range scans (first and second view). Each segmented surface is displayed with a different color. The points which failed the initial *Point Classification* step are displayed in red.

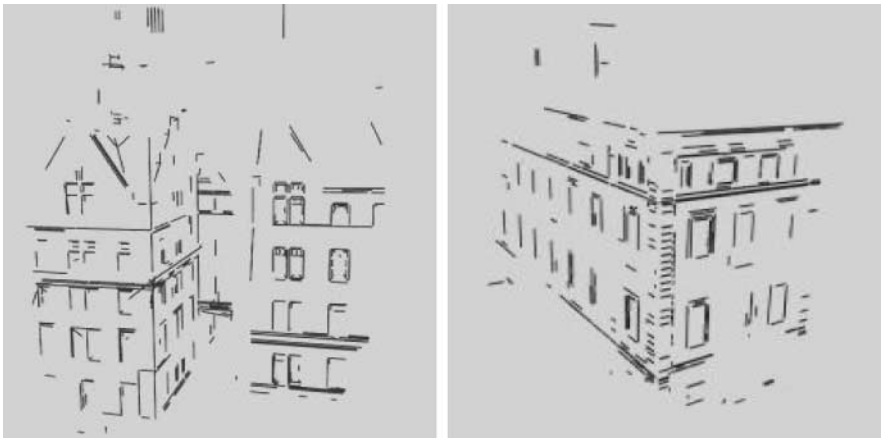

FIG. 10. 3-D lines produced by intersection of neighboring 3-D planes. Major features (such as wall and window borders) have been captured. Teachers College (left) and first view of Casa Italiana (right).

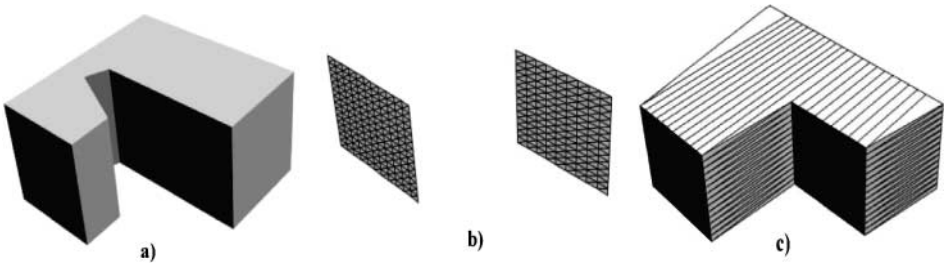

FIG. 11. Solid modeling concept. (a) Solid block being sensed, (b) Mesh generated by the measured 3-D points. (c) Partial solid model construction formed by sweeping mesh surfaces along the sensing direction.

with a solid volume. Each cloud of range points is transformed into a triangular surface mesh, and that mesh is then *swept* or extruded to a solid volume. The volumes elegantly capture the sensed and "yet to be explored" parts of the scene. The difficult problem of merging individual registered range images transforms to the computation of volumetric boolean set intersections between the partial solid volumes which represent each individual view of the scene. The modeling concept is described in Fig. 11. A triangular surface mesh (Fig. 11b) is constructed from the raw range image of the sensed object (Fig. 11a). That mesh is swept toward the sense direction and the resulting solid model is shown in Fig. 11c. A part of the reconstructed triangular mesh of an object is shown in Fig. 12a. Each triangular mesh element is extruded along the sensing direction creating a triangular prism (Fig. 12b). The final swept volume is the result of the boolean unification of the individual triangular prisms (Fig. 12c).

4.2. Modeling Outdoor Structures

Reed's method works well in a carefully calibrated laboratory setting, where objects placed on a turntable are scanned from accurately calibrated range-sensor positions. In order to extend the system to outdoor environments we need to provide a method for the registration of the individual range scans. Indoor model recovery typically uses calibrated stations which we do not have access to outdoors. However, since urban scenes provide very reliable 3-D linear features (see Section 3 for our 3-D line extraction method), we decided to use those linear features for range registration. The registration (computation of the rotation matrix R and translation vector $\mathbf{T}$) between the coordinate systems of the nth and first range image is possible if at least two line[6] matches are given. The rotation is calculated according to the closed-from solution described in [21]. After the rotation is known a linear system can be formed for the computation of the translation [47]. Currently, the match between features is manual. The automation of this process could be possible by the utilization of the RANSAC framework described in Section 5 in the context of range-to-image registration. Another issue which we need to consider is the fact that parts of the model may be missing due to transparent objects in the scene (such as windows). Those parts need to be interpolated from neighboring samples. Finally, the simplification of the dense mesh surfaces becomes important for efficient 3-D modeling and efficient rendering of the scene. A key contribution of our work is the incorporation of segmentation as the necessary stage of early 3-D modeling. The generation and unification of every individual

[6] Infinite 3-D line without endpoints.

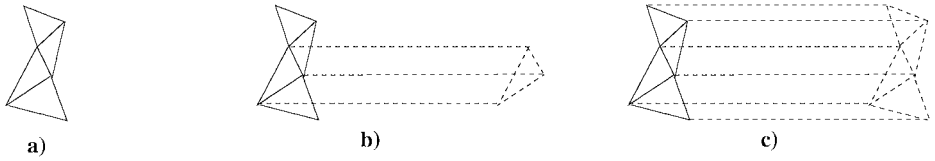

FIG. 12. Sweeping triangular mesh elements. (a) Triangular mesh surface, (b) one triangular element swept towards sensing direction, (c) all triangular elements swept.

triangular prism in order to create the final solid model (see Fig. 12) can be avoided with the use of polygonal prisms in the areas where planar surfaces have been extracted. Instead of creating one triangular prism for each individual mesh element, we can create prisms for the large planar polygonal areas extracted by the segmentation process. Parts of the scene which cannot be segmented into planar regions are treated as triangular prisms swept toward the sensing direction. That means that nonplanar parts, such as columns and cupolas, are kept as dense swept meshes.

Figure 13 demonstrates the result of sweeping the outer polygonal, hole, and triangular elements in the sensing direction. In the top row a presentation of sweeping when no holes exist is shown. Polygonal and triangular prisms are unified in order to create a partial solid model. The extent of the sweep operation is determined by the computation of an adequate far plane distance that will envelope the building's extent. For the Italian house experiment we swept each volume back by 120 m.[7] A complication arises in the presence of hole elements. This complication is demonstrated on the bottom row of Fig. 13. Holes inside large polygonal elements must be handled differently; the prisms created by sweeping the holes can be thought of as negative prisms. That means that a hole defines a void space. This is naturally implemented by the subtraction of the swept volume generated by the holes from the final partial solid model.

4.3. 3-D Modeling Results

We have tested this method on a building on Columbia University's campus. The acquired range scans are segmented, registered, and finally transformed to a nonredundant volumetric solid representation. The segmentation of each range scan is done by means of the segmentation algorithm described in Section 3. We manually match pairs of automatically extracted 3-D lines in order to align the scans on the same coordinate system (see Section 4.2).

The original range scans of the building Casa Italiana are subsampled and then segmented into polygonal planar regions. The unification of those two types of sweeps (sweeps produced by polygonal and sweeps produced by triangular faces) provides the final solid sweep for the first view of the building. The final solid sweeps for the three views of Casa Italiana and their volumetric boolean intersections which result in a topologically correct and geometrically accurate solid model of the scene are shown in Fig. 14.

Table 1 shows the reduction on the number of polygons used in the solid modeling phase. That reduction and increased efficiency is the result of the involvement of the segmentation

[7] The spatial extent of the building can be easily computed from the registered 3-D point clouds. The 3-D bounding box of the cloud is computed (by finding the extreme 3-D points of the set in the X, Y, and Z direction). The maximum dimesion of this box is used as the sweeping distance.

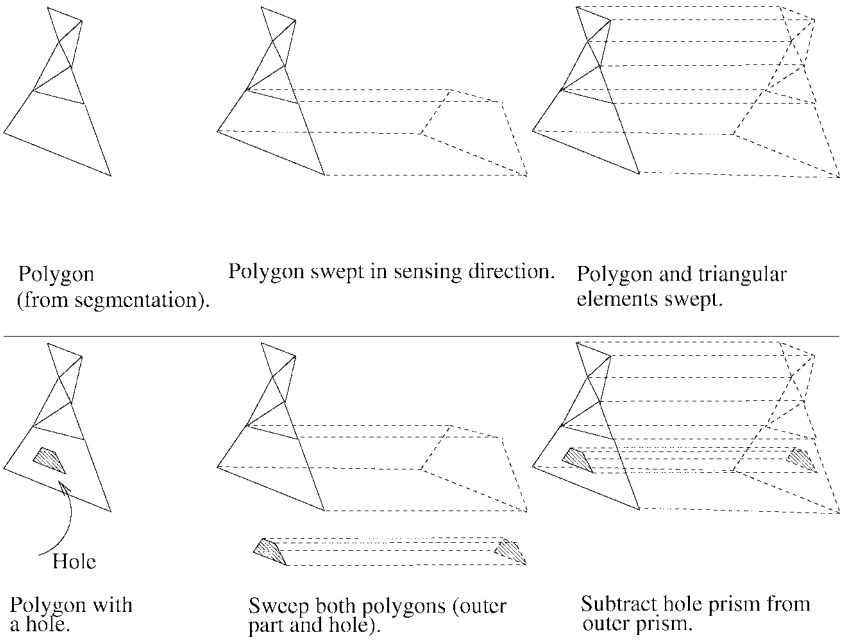

FIG. 13. Extension of solid modeler. Top row: (left) Polygonal face (without holes) and triangular mesh elements. (middle) One polygonal element swept toward the sensing direction. (right) All elements swept. The union of all prisms is the resulted partial solid volume. Bottom row: same concept. Now, though, the polygonal face has holes.

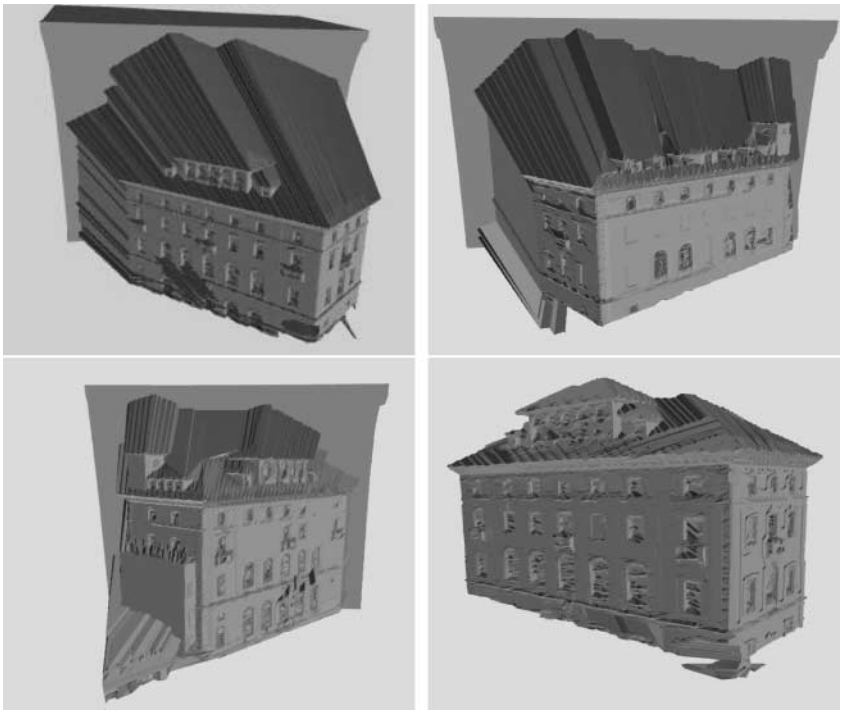

FIG. 14. Modeling Casa Italiana. Three volumetric sweeps of the segmented range scans. The volumetric set intersection of the three sweeps leads to the final composite solid model (bottom right image).

TABLE 1
Results from Modelers Extension: Reduction on Number of Triangular Prisms

Viewpoint	Before segm. Triang. prisms	After segmentation Triang. prisms	After segmentation Polyg. prisms	Reduction
1	79,596	41,864	32	52.6 %
2	58,116	24,480	19	42.1 %
3	81,192	41,245	34	50.8 %

process in the modeling phase. The reduction in the number of triangular faces as a result of the segmentation is shown for the three range scans. The second column presents the number of triangular prisms that would be used for modeling if segmentation was not used. The third and fourth column show the number of triangular and polygonal prisms that are used for modeling after the segmentation of the range scans (triangular prisms are still needed for the unsegmented parts of the scene). The last column shows the amount of reduction on the number of triangular prisms due to the segmentation process. The reduction in the spatial complexity of the sweeps is very large (on the order of 50%). That reduced complexity greatly simplifies the task of boolean intersection. It also increases the time-efficiency of the solid modeling phase.

The data simplification and segmentation of each individual range image *before* merging translates to simplified intermediate volumetric representations. Merging those intermediate representations is the *hardest* part of modeling. Thus, the decreased complexity of the individual volumes results in increased efficiency in the solid modeling phase. This is exactly the kind of efficiency that is highly desirable in the context of modeling large outdoor scenes. The method scales well with the increased sampling density of measured 3-D scenes and is thus appropriate for handling the complexity of scenes of large scale.

5. 2-D IMAGE REGISTRATION WITH RANGE DATA

Our next goal is to enhance the geometric model with photographic observations taken from a freely moving 2-D camera by automatically recovering the camera's position and orientation with respect to the model of the scene and by automatically calibrating the camera sensor. Most systems which recreate photorealistic models of the environment by a combination of range and image sensing [41, 45, 54, 57] solve the range to image registration problem by fixing the relative position and orientation of the camera with respect to the range sensor (that is the two sensors are rigidly attached on the same platform). The major drawbacks of this approach are (i) lack of 2-D sensing flexibility, since the limitations of range sensor positioning (standoff distance, maximum distance) translate to constraints on the camera placement, and (ii) static arrangement of sensors, which means that the system cannot dynamically adjust to the requirements of each particular scene (the camera sensor is precalibrated off-line). Also, the fixed approach cannot handle the case of mapping historical photographs on the models, something our method is able to accomplish.

We provide a solution to the automated pose determination of a camera with respect to a range sensor without placing artificial objects in the scene and without a static arrangement of the range-camera system. This is done by solving the problem of automatically *matching* 3-D

and 2-D features from the range and image datasets. Our approach involves the utilization of parallelism and orthogonality constraints that naturally exist in urban environments in order to extract 3-D rectangular structures from the range data and 2-D rectangular structures from the 2-D images.

The problems of pose estimation and camera calibration are of fundamental importance in computer vision and robotics research since their solution is required or coupled with stereo matching, structure from motion, robot localization, object tracking, and object recognition algorithms. There are numerous approaches for the solution of the pose estimation problem from point correspondences [18, 19, 22, 39, 42] or from line correspondences [14, 27, 33]. Works in automated matching of 3-D with 2-D features include [13, 26, 28, 30, 32, 36, 40, 55] whereas in [56] the automated matching is possible when artificial markers are placed in the scene. Finally, Ikeuchi [34] utilizes the reflectance response of the laser range sensor in order to provide an automated solution to the problem.

5.1. 2-D Camera Pose Estimation

The camera pose estimation involves the following stages:

1. Extraction of two feature sets F_{3D} and F_{2D} (3-D and 2-D linear segments from the range and image datasets) (see Section 3).

2. Grouping of the 3-D and 2-D feature sets into clusters of parallel 3-D lines L_{3D} and converging 2-D lines[8] L_{2D} (Section 5.1.1).

3. Computation of an initial pose estimate P_0 (rotation and internal camera parameters) by utilizing the directions defined by the sets L_{3D} and L_{2D} (Section 5.1.2).

4. Grouping of the 3-D and 2-D line segments into higher level structures of 3-D and 2-D rectangles R_{3D} and R_{2D} and extraction of 3-D and 2-D graphs G_{3D} and G_{2D} of rectangles (repetitive pattern of scene and image rectangles) (Section 5.1.3).

5. Automatic selection of a matched set of rectangular features C^o and computation of a pose $P^o = \mathcal{A}(C^o \mid P_0)$ by running a pose estimator algorithm $\mathcal{A}$ (computation of a *coarse* pose estimate). Refinement $P^R = R(P^o, L_{3D}, L_{2D})$ of the estimated pose P^o by using all available information computed so far (computation of a *fine* pose estimate) (Section 5.1.4).

5.1.1. Vanishing Point Extraction and Clustering 3-D Lines

In the 2-D domain the extraction of vanishing points provides a natural clustering of lines into sets which correspond to parallel 3-D lines whereas in the 3-D domain the clustering into sets of parallel 3-D lines is direct. The end result of this module is the extraction of major vanishing points $\mathbf{VP} = \{v_1, \ldots, v_n\}$. Each vanishing point is supported by a set of 2-D lines[9] and the desired clustering $\mathbf{L_{2D}} = \{L_{2D_1}, \ldots, L_{2D_n}\}$ has been accomplished. If the number of major vanishing points N_{vps} is known a priori then we can select the N_{vps} largest clusters from the set $\mathbf{L_{2D}}$ as our result. Computing the number N_{vps} is an easy task (it is equivelant to identifying the major modes of the 1-D histogram of directions of 2-D lines on the plane [35]). There are many methods for the automatic computation of the major image vanishing points (see [3, 12, 35]). Our approach involves the computation of all pairwise

[8] Those lines define vanishing points on the image space.

[9] We are using a Canny edge detector with hysteresis thresholding [11] for extracting 2-D edges. Those edges are grouped into linear segments via orthogonal regression.

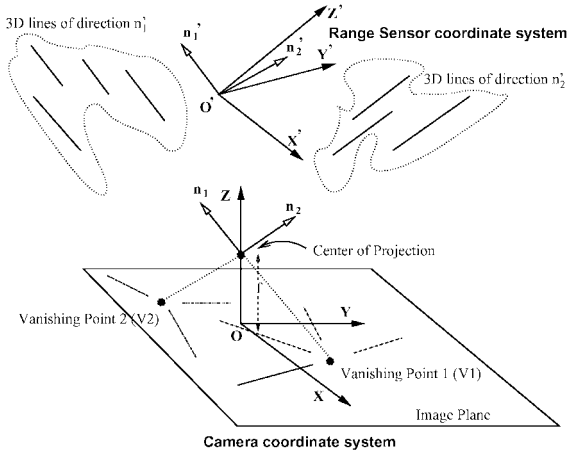

FIG. 15. Two vanishing points. The 2-D lines which correspond to parallel 3-D lines of direction $\mathbf{n_i}$ intersect at a common vanishing point V_i on the image plane.

intersections between the extracted image lines and the creation of a 2-D histogram of those intersections. For details see [47, 50].

For the clustering of the extracted 3-D lines into sets of parallel lines we are using a classic unsupervised nearest neighbor clustering algorithm [31]. The N_{vps} larger clusters of 3-D lines provide the desired grouping of 3-D lines into clusters of parallel lines $\mathbf{L_{3D}} = \{L_{3D_1}, \ldots, L_{3D_{N_{vps}}}\}$ along with the average 3-D direction of each cluster $\mathbf{U_{3D}} = \{V_{3D_1}, \ldots, V_{3D_{N_{vps}}}\}$.

5.1.2. Computing Camera Rotation and Internal Camera Calibration

The rotation computation is based on the fact that the relative orientation between two 3-D coordinate systems O and O' can be computed if two matching directions between the two systems are known. In this case there is a closed-form solution for the rotation [21] and we can write $R = R(\mathbf{n_1}, \mathbf{n'_1} \mid \mathbf{n_2}, \mathbf{n'_2})$, where $\mathbf{n_i}$ and $\mathbf{n'_i}$ are corresponding orientations expressed in the coordinate systems O and O'. In our case, the direction of the 3-D lines which produce the vanishing point v_i is the unit vector $\mathbf{n_i} = (v_i - COP)/\|(v_i - COP)\|$ (COP is the center of projection of the camera), expressed in the coordinate system of the camera sensor. This direction can be matched with a scene direction $\mathbf{n'_i}$ which is expressed in the coordinate system of the range sensor and which has been provided by the 3-D clustering module (Section 5.1.1). So, the rotation computation is reduced to the problem of finding two pairs of matching 3-D directions and 2-D vanishing points (see Fig. 15).

The camera center of projection (principal point and focal length) can be computed by three such pairs of directions[10] (see [6, 12]).

5.1.3. Computing Translation by Extracting 3-D and 2-D Rectangles

Calculating the translation requires the exact matching of local 3-D and 2-D features and global properties alone are not enough. Since 3-D points are hard to localize in

[10] The camera calibration is performed before the computation of the rotation, since the rotation computation assumes a calibrated camera.

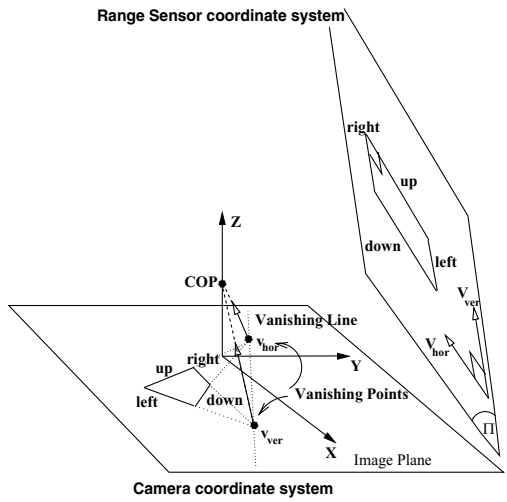

FIG. 16. 3-D rectangle formed by lines parallel to the scene directions V_{ver} and V_{hor} and its corresponding 2-D quadrangle formed by 2-D lines which meet at the image vanishing points v_{ver} and v_{hor}.

the 3-D dataset and since we have already developed a method for the reliable and accurate extraction of 3-D lines [48] we will match 2-D with 3-D linear features. In order to reduce the search-space of possible matches we move up in the feature hierarchy and group the 3-D and 2-D lines into graphs of rectangular and quadrangular structures.

The geometry of the projection of a 3-D rectangle on a 2-D image quadrangle is shown in Fig. 16. Three-dimensional rectangles which are formed by pairs of lines of directions (V_{ver}, V_{hor}) have corresponding 2-D quadrangles which are formed by pairs of 2-D lines which converge to the vanishing points (v_{ver}, v_{hor}). That means that in order to extract corresponding 3-D rectangles and 2-D quadrangles we need to utilize the extracted clusters of 3-D and 2-D lines. For the following discussion we will call one of the two scene directions *vertical* (V_{ver}) and the other one *horizontal* (V_{hor}). Analogously we call v_{ver} and v_{hor} the vanishing points which correspond to the directions V_{ver} and V_{hor}.

We can formulate the 3-D and 2-D rectangle extraction problem as follows: The input is two pairs of 3-D directions V_{ver}, $V_{hor} \in \mathbf{U_{3D}}$ and 2-D vanishing points v_{ver}, $v_{hor} \in \mathbf{VP}$ along with the 3-D L_{3D_0}, $L_{3D_1} \in \mathbf{L_{3D}}$ and 2-D L_{2D_0}, $L_{2D_1} \in \mathbf{L_{2D}}$ (Section 5.1.1) clusters that support them. The output is a set of 3-D rectangles and 2-D quadrangles R_{3D} and R_{2D} and two corresponding graphs G_{3D} and G_{2D} describing the spatial relationship among structures in R_{3D} and R_{2D}, respectively.

Following this notation, a 3-D rectangle is a planar 3-D structure whose sides can be tagged as l_{up} or l_{down} if they are parallel to the V_{hor} direction and as l_{left} or l_{right} if they are parallel to the V_{ver} direction (Fig. 16). Also we can define three relationships between rectangles which lie on the same scene plane: *right of*, *top of*, and *in* or *out of*. The exact same representation can be used for the 2-D quadrangles. In order to use the same notation and define spatial relationships between 2-D quadrangles we need to transform them to 2-D rectangles. This can be done if we rotate the two vanishing points v_{ver} and v_{hor} (and similarly transform all 2-D lines which support them) such that they are parallel to the image plane. More details on the rectangle-extraction algorithm can be found in [47].

5.1.4. Matching 3-D and 2-D Rectangles

The last part of the pose computation module is the calculation of the camera translation with respect to the range sensor by matching local 3-D and 2-D features between the range and image datasets. In Section 5.1.2 3-D scene directions are matched with 2-D image vanishing points in order to solve for the camera rotation.

Matching between a set of 3-D and 2-D rectangles will provide us a *coarse* pose estimate. Exploring every possible combination of matches is an intractable problem since we need to consider an exponentially large number of possibilities. In order to solve the problem we follow the RANSAC framework introduced in [22]. Instead of considering all possible matches we are randomly sampling the search space $(R_{3D_1} \times R_{2D_1}) \cup (R_{3D_2} \times R_{2D_2}) \cup \ldots \cup (R_{3D_M} \times R_{2D_M})$ of 3-D and 2-D rectangular structures.[11]

Each sample C_{ran} consists of a fixed number n_{ran} of pairs of 3-D and 2-D rectangles, where n_{ran} is the minimum number of matches that can produce a reliable pose-estimate. Every sample C_{ran} produces a pose estimate which is verified by projecting the 3-D rectangles on the 2-D image. A matching score Q_{match} is computed, and we select as correct the match which produces the maximum score. Our algorithm sets the score Q_{match} to equal the number of 3-D rectangles which map (when projected to the image) to an extracted 2-D quadrangle (larger is better). What remains to be defined is how we decide when two 2-D rectangles are close with respect to each other.[12] This decision is based on an adaptive threshold which depends on the relative size of pairs of rectangles [47]. Finally, the pose estimation algorithm $\mathcal{A}$ from a set of matched 3-D and 2-D lines (we can view each rectangle as a set of four lines) is described in detail in [33]. In the implementation of the RANSAC procedure the pose estimator $\mathcal{A}$ optimizes only with respect to the translation since the rotation is already known to us (Section 5.1.2).

If we want to ensure with probability Pr that at least one of our random selections corresponds to a valid match then the maximum number of steps is $N_{max} = \log(1 - Pr)/\log(1 - b)$ where b is the probability of randomly selecting a sample of n_{ran} *correct* matches [22]. If we assume that in our scene there are K pairs of 3-D and 2-D rectangles that can be correctly matched then $b = (K/L)^{n_{ran}}$ and $L = |(R_{3D_1} \times R_{2D_1}) \cup (R_{3D_2} \times R_{2D_2}) \cup \ldots \cup (R_{3D_M} \times R_{2D_M})|$ is the number of all possible pairs of 3-D and 2-D rectangles. K is unknown, so we set it equal to $1/3L$ for our experiments. Note that the lower the probability of correct matches b the larger the number of required steps N_{max}.

The coarse estimate computed using the RANSAC method is very important because it provides an initial solution which can be subsequently refined to yield a *final* pose estimate. The refinement involves the projection of all 3-D lines of the extracted clusters $\mathbf{L_{3D}}$ on the 2-D image assuming the coarse pose estimate P^o and so a set of projected 3-D lines $\mathcal{P}(\mathbf{L_{3D}})$ is formed. Each individual projected cluster is compared with the groups of extracted 2-D lines $\mathbf{L_{2D}}$ and new line matches among the 3-D and 2-D datasets are verified. The increased number of line matches results in better pose estimation.

[11] N matches $(\mathbf{n}'_\mathbf{i}, \mathbf{n}_\mathbf{i})$ between scene and image directions produce $M = \binom{N}{2}$ pairs of the form $((\mathbf{n}'_\mathbf{i}, \mathbf{n}_\mathbf{i}), (\mathbf{n}'_\mathbf{j}, \mathbf{n}_\mathbf{j}))$. In Section 5.1.3 we described a method to compute 3-D and 2-D rectangles (R_{3D_k}, R_{2D_k}) from clusters of 3-D and 2-D lines, and pairs of the above pairs of matched scene and image directions.

[12] Note that 2-D quadrangles are transformed into 2-D rectangles when we extract the vanishing points which produce them.

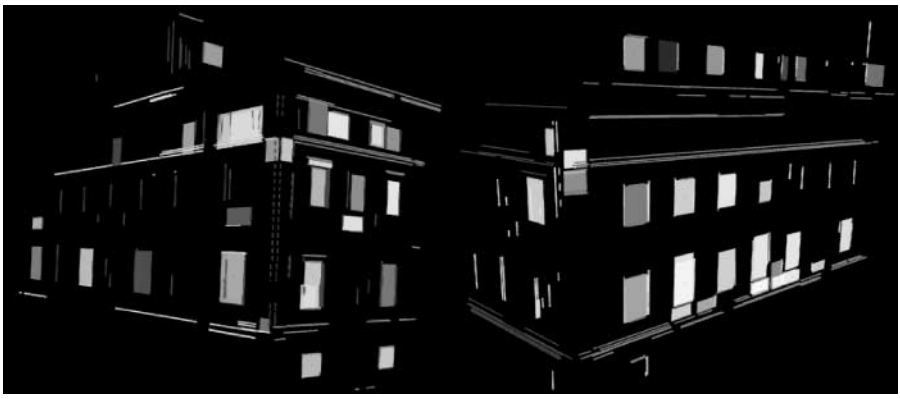

FIG. 17. (a,b) Clusters of 3-D lines (color encodes different directions) and extracted 3-D rectangles (rectangles are rendered as solids of different color for clarity). Two different views of the building.

5.2. Range-Image Matching Results

The results of the 3-D line and rectangle extraction from two range scans of Casa Italiana (Fig. 6) are shown in Figs. 17a and 17b. For clarity, different rectangles are rendered with different colors. Also there are three major clusters of parallel lines (encoded with the colors red, green, and blue). In the 2-D domain, the three major vanishing points and clusters of 2-D lines (for two views of the building) are shown in Figs. 18a and 18b. The automatically computed principal point of the cameras is also shown; it is the point of intersection of vanishing point directions on the image. The next set of figures (18c, 18d) displays the results of extracting 2-D quadrangles from the 2-D images. The extracted quadrangles from two different views are overlaid on the 2-D images. Notice that in some cases we introduce false boundary edges to complete the quadrangles. However, these false boundaries do not affect the matches, since RANSAC filters out correlations between 3-D and 2-D rectangles which do not produce a valid pose estimate.[13]

The outcome of the coarse pose estimation algorithm (RANSAC algorithm) is presented next in Figs. 18e and 18f. The extracted 2-D rectangles (red in color or black in black-and-white printing) are shown overlaid with the projection (green in color or white in black-and-white printing) of those 3-D rectangles which produce the maximum matching score Q_{match} (Q_{match} is 9 for the first view and 8 for the second view). The final pose (Section 5.1.4) is visually verified in Figs. 19a and 19b where the extracted 3-D lines shown in Figs. 17a and 17b respectively are projected on the 2-D images. The extracted 2-D lines are shown in red and the projected 3-D lines in green. As you can see the projected 3-D lines are very well aligned with the 2-D data-sets, which means that both the registration and the feature extraction algorithms produce accurate results. The number of samples the RANSAC algorithm tried was 8457 (6.0 s on an SGI Onyx2) for the first view and 223831 (2 min and 29 s) for the second view.

Finally, Figs. 20a and 20b present the final texture-mapped 3-D models using the computed calibration parameters and pose estimate on the two views of the model. The texture map visually verifies the accuracy of our method and presents the true geometric model of the scene enhanced with the photometric observations.

[13] A false match results to a pose estimate which is not verified when the 3-D rectangles are projected on the 2-D image, since most 3-D rectangles will be projected far away from their 2-D counterparts.

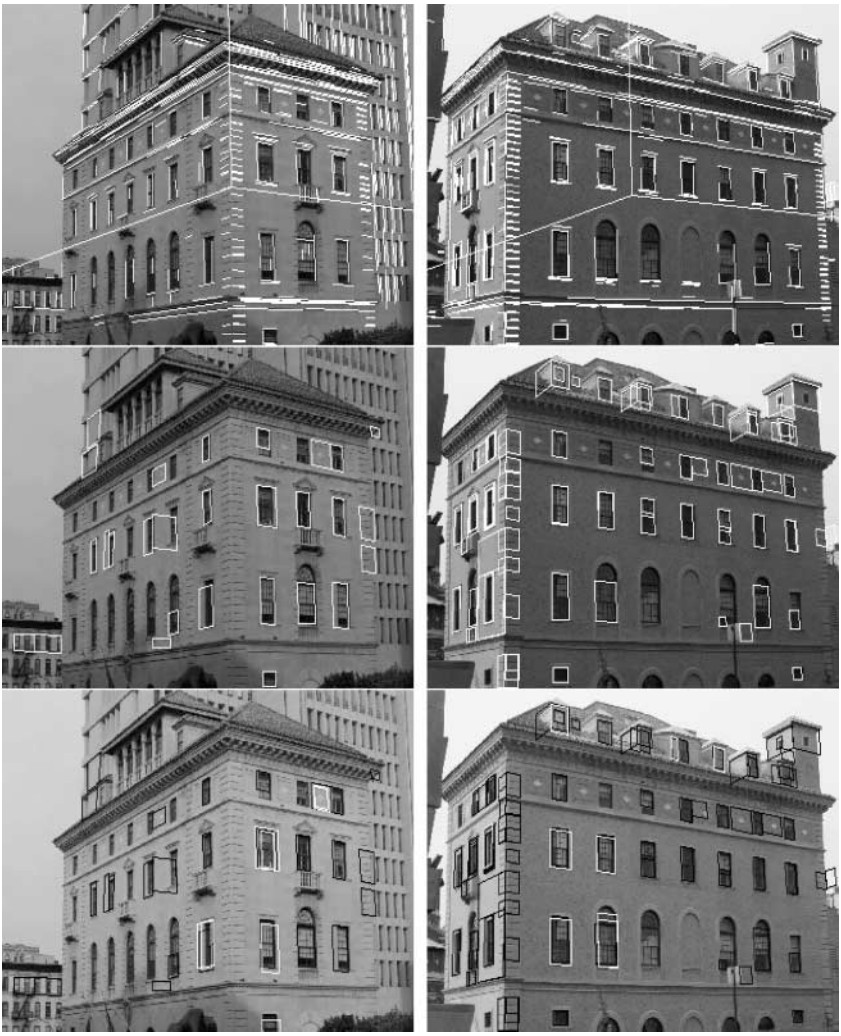

FIG. 18. Results. (Top row) 2-D images and clusters of 2-D lines, where different colors correspond to different vanishing points (two views of the building shown). (Middle row) Extracted 2-D quadrangles. (Bottom row) Extracted 2-D quadrangles (shown in red in color or black in black-and-white printing) and Q_{match} matched 3-D rectangles projected on images after coarse pose estimation (shown in green in color or white in black-and-white printing).

FIG. 19. Results. Projected 3-D lines on the images after final pose estimation (shown in green in color or white in black-and-white printing). The extracted 2-D lines are shown in red in color or black in black-and-white printing.

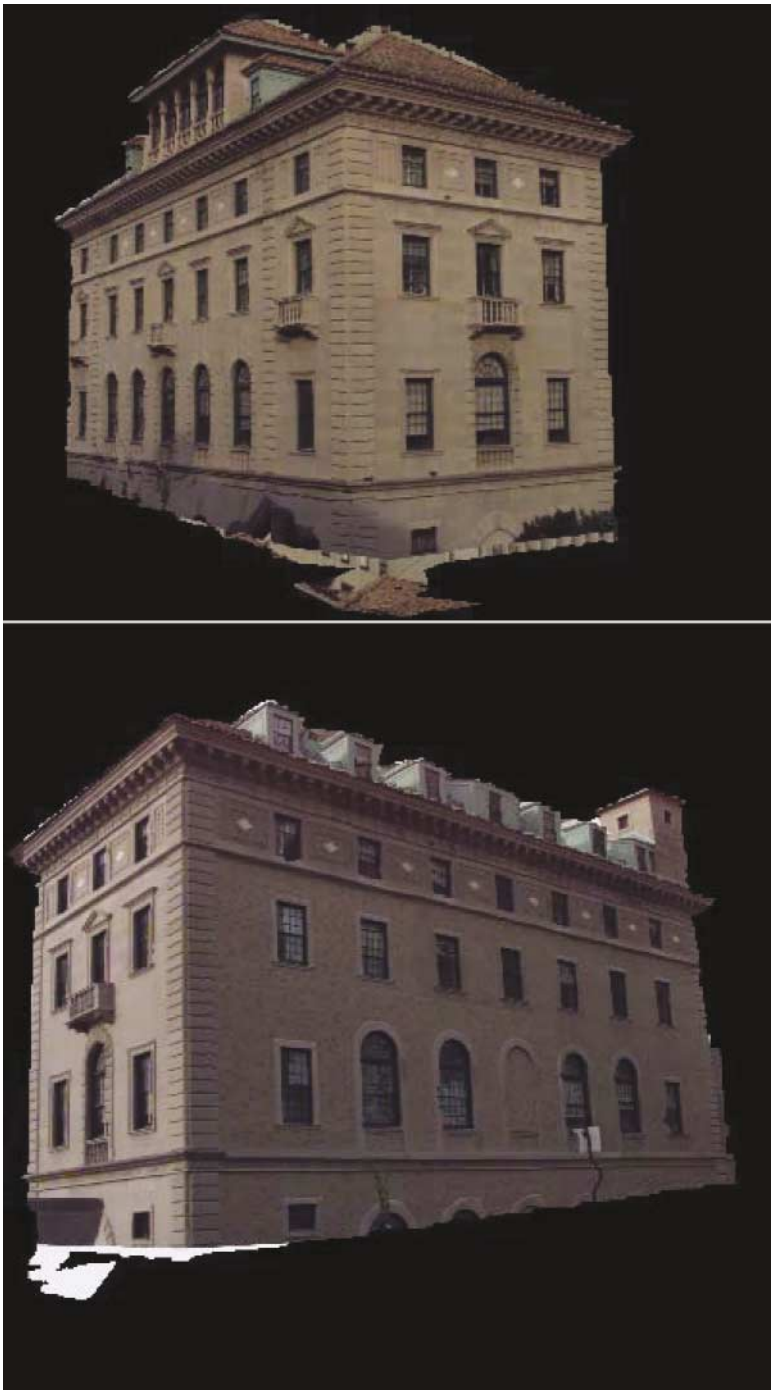

FIG. 20. Results. Images texture-mapped on 3-D model assuming final pose (two views of the model). The texture map, visually verifies the accuracy of our method and presents the true geometric model of the scene enhanced with the photometric observations.

6. CONCLUSIONS

Our system addresses one of the most difficult problems in computer vision and robotics research in a unique and effective manner. We believe that the developed modules are of vital importance for a flexible photorealistic 3-D model acquisition system. Segmentation algorithms simplify the dense datasets and provide stable features of interest which can be used for registration purposes. The solid modeling provides geometrically correct 3-D models. The automated range-to-image registration can increase the flexibility of the system by decoupling the slow geometry recovery process from the image acquisition process; the camera does not have to be precalibrated and rigidly attached to the range sensor. The system is comprehensive in that it addresses all phases of the modeling problem with a particular emphasis on automating the entire process without manual interaction.

Due to the scope of the system, there are still a number of open technical issues that need to be addressed: (i) The segmentation routines fit planes to the extracted clusters of points. Fitting of general smoothly varying surfaces is needed in nonplanar parts of the scene. Also a number of thresholds have to manually be set by the user in order to customize the segmentation. It is unclear how the current system will work with scenes that do not contain large amounts of planar surfaces. (ii) The solid modeling part requires very accurate registration between the range images. Also the whole object must be in the field of view of the sensor at each scanning operation. This problem can be attacked if we replace the boolean intersection of the solid sweeps with the unification of the complements of the sweeps. (iii) Our algorithm operates in scenes which contain linear features with strong orthogonality constraints. We are able to exploit the unique features found in urban environments; it is yet to be determined how it will extend to more general 3-D scenes. (iv) Currently we do not merge multiple images (textures) on the 3-D model but we texture-map only one brightness image per view. There are methods to attack the problem (i.e., view-dependent texturing [17], statistical texture estimation [15], or the unstructured lumigraph method [10]). All approaches implement heuristics though and the research problem is still open.

Our future work includes investigation in the area of sensor planning along the lines of [44]. In order for this module to be operational on outdoor scenes, a navigational module in a partial known world is needed. This module is currently under development in our lab [25] and it runs on a mobile robot where the laser range scanner is located. The integrated project (called AVENUE) [1] contains all the necessary modules (mobile robot navigation, partial 3-D and 2-D maps of the environment of Columbia University's campus, sensor planning, and site modeling) which can enable a completely autonomous site exploration system.

ACKNOWLEDGMENTS

This work was supported in parts by an ONR/DARPA MURI award ONR N00014-95-1-0601, DURIP award N00014-98-1-0267 and NSF grants CDA-96-25374 and EIA-97-29844. The first author was also supported by the Foundation of Ioannis S. Latsis, Greece. The authors thank Michael Reed for his help in creating the solid models.

REFERENCES

1. P. K. Allen, I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer, AVENUE: Automated site modeling in urban environments, in *3rd Int. Conference on Digital Imaging and Modeling, 2001*, pp. 357–364.

2. M. Antone and S. Teller, *Automatic Recovery of Camera Positions in Urban Scenes*, Technical Report LCS TR-814, MIT, December 2000.
3. M. E. Antone and S. Teller, Automatic recovery of relative camera rotations for urban scenes, in *IEEE Conf. Computer Vision and Pattern Recognition, Hilton Head, NC, July 2000*, pp. 282–289.
4. C. Baillard and A. Zisserman, A plane-sweep strategy for the 3D reconstruction of buildings from multiple images, in *19th ISPRS Congress and Exhibition, July 2000*, Vol. 32, pp. 56–62.
5. D. Ballard and C. Brown, *Computer Vision*, Prentice Hall, Englewood Cliffs, NJ, 1982.
6. S. Becker, *Vision-Assisted Modeling from Model-Based Video Representations*, Ph.D. thesis, Massachusetts Institute of Technology, February 1997.
7. J.-A. Beraldin, L. Cournoyer, M. Rioux, F. Blais, S. F. El.-Hakim, and G. Godin, Object model creation from multiple range images: Acquisition, calibration, model building and verification, in *Intern. Conf. on Recent Advances in 3-D Dig. Imaging and Modeling, Ottawa, Canada, May 1997*, pp. 326–333.
8. F. Bernardini and H. Rushmeier, The 3D model acquisition pipeline, in *Eurographics 2000 State of the Art Report (STAR), 2000*.
9. P. J. Besl and R. C. Jain, Segmentation through variable-order surface fitting, *IEEE Trans. Pattern Anal. Mach. Intell.* **10**, 1988, 167–192.
10. C. Buehler, M. Bosse, S. Gortler, M. Cohen, and L. McMillan, Unstructured lumigraph rendering, in *SIGGRAPH 2001*, pp. 425–432.
11. J. Canny, A computational approach to edge detection, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**(6), 1986, 679–698.
12. B. Caprile and V. Torre, Using vanishing points for camera calibration, *Int. Conf. Comput. Vision* **4**, 1990, 127–140.
13. T. Cass, Polynomial-time geometric matching for object recognition, *Int. J. Comput. Vision* **21**, 1997, 37–61.
14. S. Christy and R. Horaud, Iterative pose computation from line correspondences, *Comput. Vision Image Understand.* **73**, 1999, 137–144.
15. S. Coorg and S. Teller, Extracting textured vertical facades from contolled close-range imagery, in *IEEE Conf. Computer Vision and Pattern Recognition, Fort Collins, Colorado, 1999*, pp. 625–632.
16. B. Curless and M. Levoy, A volumetric method for building complex models from range images, in *SIGGRAPH, 1996*, pp. 303–312.
17. P. E. Debevec, C. J. Taylor, and J. Malik, Modeling and rendering architecture from photographs: A hybrid geometry-based and image-based approach, in *SIGGRAPH, 1996*, pp. 11–20.
18. D. F. DeMenthon and L. S. Davis, Model-based object pose in 25 lines of code, *Int. J. Comput. Vision* **15**, 1995, 123–141.
19. M. Dhome, M. Richetin, J.-T. Lapresté, and G. Rives, Determination of the attitude of 3-D objects from a single perspective view, *IEEE Trans. Pattern Anal. Mach. Intell.* **11**, 1989, 1265–1278.
20. S. F. El-Hakim, P. Boulanger, F. Blais, and J.-A. Beraldin, A system for indoor 3-D mapping and virtual environments, in *Videometrics V, July 1997*, pp. 21–35.
21. O. Faugeras, *Three-Dimensional Computer Vision*, MIT Press, Cambridge, MA, 1996.
22. M. A. Fischler and R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Graphics Image Process.* **24**, 1981, 381–395.
23. A. Fitzgibbon, D. Eggert, and R. Fisher, High-level CAD model acquisition from range images, *Comput. Aided Design* **29**, 1997, 321–330.
24. A. W. Fitzgibbon and A. Zisserman, Automatic 3D model acquisition and generation of new images from video sequences, in *Proc. of European Signal Processing Conf. (EUSIPCO '98), Rhodes, Greece, 1998*, pp. 1261–1269.
25. A. Gueorguiev, P. K. Allen, E. Gold, and P. Blaer, Design, architecture and control of a mobile site modeling robot, in *Intern. Conf. on Robotics & Automation, San Francisco, April 2000*, pp. 3266–3271.
26. G. Hausler and D. Ritter, Feature-based object recognition and localization in 3D-space, using a single video image, *Comput. Vision Image Understand.* **73**, 1999, 64–81.
27. R. Horaud, F. Dornaika, B. Lamiroy, and S. Christy, Object pose: The link between weak perspective, para-perspective, and full perspective, *Int. J. Comput. Vision* **22**(2), 1997, 173–189.

28. D. Huttenlocher and S. Ullman, Recognizing solid objects by alignment with an image, *Int. J. Comput. Vision* **5**(7), 1990, 195–212.
29. Institute of Industrial Science(IIS), *Urban Multi-Media/3D Mapping Workshop*, University of Tokyo, Japan, 1999.
30. D. W. Jacobs, Matching 3-D models to 2-D images, *Int. J. Comput. Vision* **21**, 1997, 123–153.
31. A. Jain and R. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, 1988.
32. F. Jurie, Solution of the simultaneous pose and correspondence problem using gaussian error model, *Comput. Vision Image Understand*. **73**, 1999, 357–373.
33. R. Kumar and A. R. Hanson, Robust methods for estimating pose and a sensitivity analysis, *Computer Vision, Graphics Image Process.* **60**, 1994, 313–342.
34. R. Kurazume, K. Nishino, Z. Zhang, and K. Ikeuchi, Simultaneous 2D image and 3D geometric model registration for texture mapping utilizing reflectance attributes, in *The 5th Asian Conference on Computer Vision, January 2002*, pp. 454–461.
35. D. Liebowitz and A. Zisserman, Metric rectification for perspective images of planes, in *IEEE Conf. Computer Vision and Pattern Recognition, Santa Barbara, CA, 1998*, pp. 482–488.
36. D. Lowe, Robust model-based motion tracking through the integration of search and estimation, *Int. J. Comput. Vision* **8**, 1992, 113–122.
37. Digital Michelangelo Project, http://graphics.Stanford.edu/projects/mich/.
38. J. M. M. Montiel and A. Zisserman, Automated architectural acquisition from a camera undergoing planar motion, in *Int. Symp. on Virtual and Augmented Architecture (VAA01), Dublin, Ireland, June 2001*, pp. 207–218.
39. D. Oberkampf, D. DeMenthon, and L. Davis, Iterative pose estimation using coplanar feature points, *Comput. Vision Graphic Image Process.* **63**(3), 1996, 495–511.
40. C. Olson, Time and space efficient pose clustering, in *CVPR, Seattle, WA, 1994*, pp. 251–258.
41. K. Pulli, H. Abi-Rached, T. Duchamp, L. G. Shapiro, and W. Stuetzle, Acquisition and visualization of colored 3-D objects, in *Inter. Conf. on Pattern Recognition, Australia, 1998*, pp. 11–15.
42. L. Quan and Z. Lan, Linear N-point camera pose determination, *Pattern Anal. Mach. Intell.* **21**(7), 1999, 774–780.
43. M. Reed and P. K. Allen, 3-D modeling from range imagery, *Image Vision Comput.* **17**, 1999, 99–111.
44. M. Reed and P. K. Allen, Constrained-based sensor planning for scene modeling, *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 2000, 1460–1466.
45. V. Sequiera, K. Ng, E. Wolfart, J. Concalves, and D. Hogg, Automated reconstruction of 3D models from real environments, *ISPRS J. Photogrammetry Remote Sensing* **54**, 1999, 1–22.
46. H.-Y. Shum, M. Han, and R. Szeliski, Interactive construction of 3D models from panoramic mosaics, in *IEEE Conf. Computer Vision and Pattern Recognition, Santa Barbara, CA, June 1998*, pp. 427–433.
47. I. Stamos, *Geometry and Texture Recovery of Scenes of Large Scale*, Ph.D. thesis, Columbia University, 2001.
48. I. Stamos and P. K. Allen, 3-D model construction using range and image data, in *IEEE Conf. Computer Vision and Pattern Recognition, Hilton Head, SC, July 2000*, Vol. I, pp. 531–536.
49. I. Stamos and P. K. Allen, Integration of range and image sensing for photorealistic 3D modeling, in *Inter. Conf. on Robotics and Automation, San Fransisco, CA, May 2000*, pp. 1435–1440.
50. I. Stamos and P. K. Allen, Registration of 3D with 2D imagery in urban environments, in *Inter. Conf. on Computer Vision, Vancouver, Canada, July 2001*, pp. 731–736.
51. F. Taillandier, *Texture and Relief Estimation from Multiple Georeferenced Images*, Master's thesis, Ecole Polytechnique, September 2000.
52. MIT City Scanning Project, http://graphics.lcs.mit.edu/city/city.html.
53. G. Turk and M. Levoy, Zippered polygon meshes from range images, in *SIGGRAPH, 1994*, pp. 303–312.
54. Visual Information Technology Group, Canada, http://www.vit.iit.nrc.ca/VIT.html.
55. W. Wells, Statistical approaches to feature-based object recognition, *Int. J. Comput. Vision* **21**, 1997, 63–98.
56. Y. Yu, *Modeling and Editing Real Scenes with Image-Based Techniques*, Ph.D. thesis, UC Berkeley, 2000.
57. H. Zhao and R. Shibasaki, A system for reconstructing urban 3D objects using ground-based range and CCD sensors, in *Urban Multi-Media/3D Mapping workshop, Inst. of Industr. Sc., Univ. of Tokyo, 1999*.