Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes

Ajmal S. Mian, Mohammed Bennamoun, and Robyn Owens

Abstract—Viewpoint independent recognition of free-form objects and their segmentation in the presence of clutter and occlusions is a challenging task. We present a novel 3D model-based algorithm which performs this task automatically and efficiently. A 3D model of an object is automatically constructed offline from its multiple unordered range images (views). These views are converted into multidimensional table representations (which we refer to as tensors). Correspondences are automatically established between these views by simultaneously matching the tensors of a view with those of the remaining views using a hash table-based voting scheme. This results in a graph of relative transformations used to register the views before they are integrated into a seamless 3D model. These models and their tensor representations constitute the model library. During online recognition, a tensor from the scene is simultaneously matched with those in the library by casting votes. Similarity measures are calculated for the model tensors which receive the most votes. The model with the highest similarity is transformed to the scene and, if it aligns accurately with an object in the scene, that object is declared as recognized and is segmented. This process is repeated until the scene is completely segmented. Experiments were performed on real and synthetic data comprised of 55 models and 610 scenes and an overall recognition rate of 95 percent was achieved. Comparison with the spin images revealed that our algorithm is superior in terms of recognition rate and efficiency.

Index Terms—Multiview correspondence, registration, 3D object recognition, segmentation, 3D representation, shape descriptor, geometric hashing.

1 Introduction

The aim of object recognition is to correctly identify objects in a scene and estimate their pose (location and orientation). Object recognition in complex scenes in the presence of clutter (due to noise and the presence of unwanted objects) and occlusions (due to the presence of multiple objects) is a challenging task. Object recognition from 2D images is an appealing approach due to the widespread availability of cameras. However, 2D recognition techniques are sensitive to illumination, shadows, scale, pose, and occlusions. Three-dimensional object recognition on the other hand, does not suffer from these limitations. An important paradigm of 3D object recognition is model-based (e.g., [22], as opposed to view-based, e.g., [18]) whereby 3D models of objects are constructed offline and stored in a model library using a suitable representation. During online recognition, a range image of the scene is converted into a similar representation and matched with the models of the database in order to recognize library objects.

A 3D model of a free-form object [3] is constructed by acquiring its range images from multiple viewpoints so that its surface is completely covered. These views are then registered in a common coordinate basis. Registration is performed in two steps, namely, coarse and fine registration [6]. Coarse registration can be performed manually or automatically through system calibration or feature matching [6]. We will focus on automatic coarse registration using feature matching, also known as correspondence identification [34]. Coarse registration is followed by fine registration, using, for example, the Iterative Closest Point (ICP) algorithm [4]. After fine registration, the views are integrated and reconstructed to form a seamless 3D model.

The main challenge in 3D modeling is the automatic establishment of correspondences between overlapping views. This problem becomes more challenging when the views are *unordered* (i.e., the order in which the views were acquired is unknown and, hence, there is no a priori knowledge about which view pairs overlap). A *pairwise* correspondence algorithm is not practical in such cases because it must exhaustively search for correspondences between $\frac{N(N-1)}{2}$ view pairs ($O(N^2)$, where N is the total number of views). In the case of unordered views, a multiview correspondence algorithm is more suitable. We define *multiview correspondence* as a one-to-many correspondence approach ($O(N)$) whereby a single view is simultaneously matched with multiple views. Our major contribution in the model database construction is a novel multiview correspondence algorithm which is an extension of our pairwise correspondence algorithm [34].

Existing correspondence techniques such as the RANSAC-based DARCES [8], bitangent curve matching [42], spin image matching [22], geometric histogram matching [1], three-tuple matching [9], and SAI matching [19] are all pairwise correspondence techniques and, therefore, cannot be efficiently applied to solve the multiview correspondence problem [33]. Huber and Hebert [20] proposed a framework for automatic 3D modeling from unordered views. Their framework is, however, based on an exhaustive search ($O(N^2)$) to find correspondences between all possible pairs of views in order to initialize a graph of relative pose

• *The authors are with the School of Computer Science and Software Engineering, The University of Western Australia, 35 Stirling Highway, Crawley, WA 6009, Australia.*
E-mail: {ajmal, bennamou, robyn}@csse.uwa.edu.au.

Manuscript received 7 Mar. 2005; revised 26 Jan. 2006; accepted 3 Feb. 2006; published online 11 Aug. 2006.
Recommended for acceptance by R. Basri.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0126-0305.

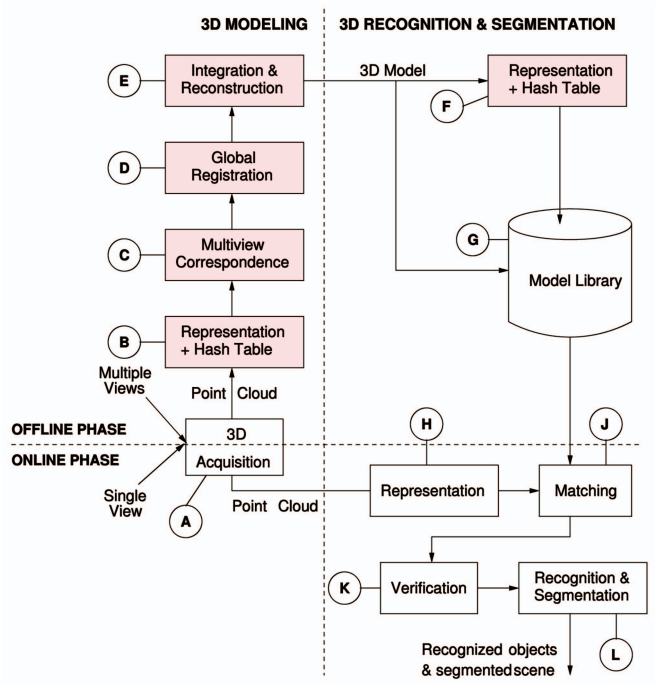


Fig. 1. Block diagram of our 3D model-based object recognition and segmentation algorithm. The shaded blocks are exclusive to the 3D modeling algorithm, whereas the unshaded blocks constitute the recognition and segmentation algorithm.

estimates. The exhaustive search is followed by the multiview surface matching algorithms which search for discrete edges (pairwise correspondences) in the graph and verify each pairwise correspondence by performing global (multiview) fine registration of all the views. This adds to the computational complexity of the already expensive exhaustive search.

Our multiview correspondence algorithm automatically establishes correspondences between the **unordered** 2.5D views of a free-form object by performing a one-to-many correspondence search ($O(N)$) using a 4D hash table (Fig. 1, Modules B and C). The result is a spanning tree of relative transformations between the unordered views used to coarsely register them in a common coordinate basis (Section 3). The registration is refined using multiview fine registration (Module D) followed by the integration and reconstruction of the views into a seamless 3D model (Module E). To the best of our knowledge, our algorithm (initially proposed in [30]) is the only automatic multiview correspondence algorithm available. Note that our contribution is in the area of multiview coarse registration and should not be confused with multiview fine registration algorithms (e.g., the work of Oishi et al. [35]), which assumes that the views are coarsely registered and, hence, one can easily find overlapping views and their amount of overlap. Moreover, the work of Avidan et al. [2] is related to multiview correspondence between 2D images as opposed to range images.

Our second major contribution is in 3D object recognition. Existing techniques have various limitations mainly in terms of their applicability to free-form objects, accuracy, efficiency, robustness to clutter and occlusions, and the discriminating capability of the used feature representation. An excellent survey of free-form object representation and recognition techniques can be found in [6]. A brief survey is presented here for completeness. Flynn and Jain [14] used CAD models to build large databases for automatic 3D object recognition. Recently, the use of scanned 3D models has become popular due to the decreasing cost of 3D scanners. COSMOS [13] requires the calculation of principal curvatures which are not only sensitive to noise but also require the underlying surface to be smooth and twice differentiable. Moreover, COSMOS cannot be used for the recognition of occluded objects. The splash representation [39] makes assumptions about the shape of the objects and is also sensitive to noise and occlusions. Recognition techniques based on matching B-Spline curves such as [40], remain sensitive to the unsolved knot problem. The B-Spline representation is not unique since the location of the knots cannot be accurately calculated and is not repeatable for a given object. SAI matching [17] is limited to only those surfaces which are topologically equivalent to a sphere and cannot differentiate between similar objects having different scales. The derivation of point signatures [10] is vulnerable to surface sampling and noise and, therefore, may result in ambiguous representations [33]. CEGI-based techniques such as [24] cannot be applied to free-form objects. The recognition based on HOT curves [23] relies on the accurate localization of inflection points, which is sensitive to noise. Johnson and Hebert proposed a recognition algorithm based on a novel spin image representation [22]. However, their representation is sensitive to the resolution and sampling of the models and scene. Moreover, spin images have a low discriminating capability because they map a 3D surface to a 2D histogram [34], which leads to many ambiguous matches. An improvement to this technique [7] overcomes the sensitivity of the spin images to resolution and sampling. However, the problems of the low discriminating capability of the representation and the inefficiency of the algorithm remain unsolved. Most of the above techniques use a one-to-one matching strategy resulting in a recognition time that grows linearly with the size of the model library. Recently, the spin image representation was used in a batch RANSAC algorithm for rapid 3D vehicle recognition [37]. It was also used for automatic clustering and classification of 3D vehicles [12], [21] in order to handle large databases and classify unknown but similar vehicles. Some pairwise correspondence techniques (e.g., [1], [8]) have also been applied to 3D object recognition.

We present a fully automatic 3D model-based free-form object recognition [31] and segmentation [32] algorithm based on our robust multidimensional table representation (which we refer to as tensors) [34] in order to overcome the deficiencies of the existing algorithms. Fig. 1 shows the block diagram of our complete algorithm, including the offline 3D modeling and the online recognition and segmentation phases. The combination of the strengths of our representation [34] and the customized use of a 4D hash table for matching constitutes the basic ingredients of this novel algorithm. Our algorithm is fully automatic and requires no user intervention at any stage, including the offline 3D modeling, the online object recognition, and scene segmentation. Briefly, the online recognition and segmentation part of our algorithm proceeds as follows: The point cloud of a scene is converted into a triangular mesh which is decimated for performance reasons. Next, a pair of vertices is randomly selected from this mesh to construct a tensor (Module H). This tensor is then simultaneously matched with the tensors of the 3D models in the library by casting votes to the tuples (model number, tensor number) using a 4D hash table (Module J). The tuples that receive fewer votes than a threshold are discarded and a similarity measure is calculated between the scene tensor and the tensors of the remaining

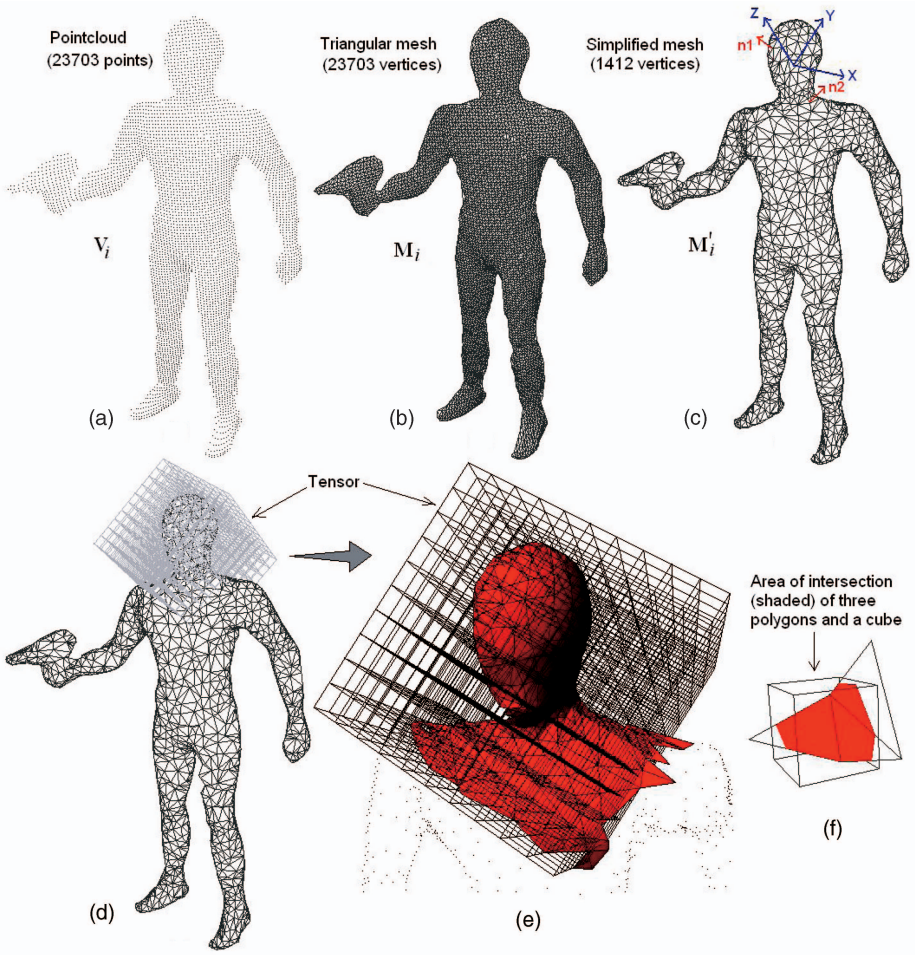


Fig. 2. Illustration of the tensor computation. (c) Two points and their normals n1 and n2 (red) are used to define a 3D coordinate basis (blue). (d) and (e) A $10 \times 10 \times 10$ grid defined at the origin. (f) The surface area intersecting each bin of the grid is the value of the corresponding tensor element. (Most figures in this paper, including this one, are best viewed in color.)

tuples. The tuple with the highest similarity is hypothesized to be present in the scene. This hypothesis is verified by transforming the 3D model to the scene (Module K). If the model aligns accurately with an object in the scene, that object is recognized and segmented (Module L). The segmented region is removed from the scene and the above process is repeated until the scene is completely segmented or no further library objects are present in the scene. Experiments were performed using a model library of 55 objects and 610 scenes and an overall recognition rate of 95 percent was achieved. Our results indicate that our algorithm is independent of the amount of clutter in the scene and can effectively handle up to 82 percent occlusion. Experimental comparison with the spin images [22] revealed that our algorithm is superior in terms of accuracy and efficiency.

This paper is organized such that the different modules of Fig. 1 are described in a sequential order except for 3D acquisition (Module A), which is outside the scope of this paper.

2 Tensor Representation and Hash Table Construction (Module B)

The input views (or point clouds Fig. 2a) $\mathbf{V}_i$ (where V_i is an $n_i \times 3$ matrix of 3D coordinates) are converted into triangular meshes $\mathbf{M}_i$ $(i = 1, \ldots, N)$ (Fig. 2b). Each $\mathbf{M}_i$ is decimated [16], for performance reasons, to get $\mathbf{M}'_i$ (Fig. 2c). Normals are then calculated for each vertex and triangular face of $\mathbf{M}'_i$. If $\mathbf{V}_i$ contain the entire object and completely cover its surface, its approximate dimensions $\mathbf{D}$ can be calculated using (1) [34].

$$\mathbf{D} = [D_x, D_y, D_z] = \max_{xyz}\left(\max_{xyz}(\mathbf{V}_i\mathbf{P}_i) - \min_{xyz}(\mathbf{V}_i\mathbf{P}_i)\right), \quad (1)$$

where $\mathbf{P}_i$ is the rotation matrix that aligns $\mathbf{V}_i$ with its principal axes. The function $\max_{xyz}(\mathbf{V}_i)$ takes the maximum values in each column of $\mathbf{V}_i$. Tensors are then calculated for all $\mathbf{M}'_i$ as described in Section 2.1 and a hash table is built as described in Section 2.2.

2.1 Tensor Computation

The vertices of each $\mathbf{M}'_i$ are paired such that they satisfy a distance and an angle constraint. The distance constraint allows pairing of only those vertices whose mutual distance is between d_{min} and d_{max}. If d_{min} and d_{max} are chosen too large, there will be slim chances that both vertices of the pair will be chosen from the overlapping region of the two views. If d_{min} and d_{max} are too small, the resultant coordinate basis (explained below) will be sensitive to noise. If $d_{max} - d_{min}$ is too small, very few vertices will satisfy this constraint and if $d_{max} - d_{min}$ is too large, too many vertices will satisfy this

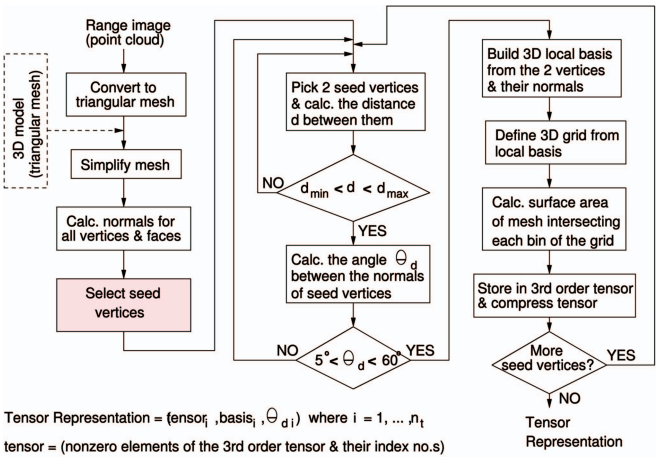


Fig. 3. Flow chart of the tensor representation. The input is either a range image or a 3D model.

constraint. We defined d_{min} and d_{max} according to (2) on the basis of our extensive experiments [34].

$$d_{min} = \frac{\text{mean}(D_x, D_y, D_z)}{6} \quad d_{max} = \frac{\text{mean}(D_x, D_y, D_z)}{4}. \quad (2)$$

The angle constraint allows pairing of only those vertices for which the angle θ_d between their normals is greater than 5 degrees. A single vertex is paired with the nearest three vertices which satisfy the above constraints. These constraints not only avoid the C_2^n combinatorial explosion of vertex pairs, but also increase the chances that both vertices in a pair are selected from the region of overlap of the two views.

A 3D coordinate basis is then defined for each pair of vertices as follows: The center of the line joining the vertices makes the origin. The average of the normals of the two vertices makes the z-axis. The cross product of the two normals makes the x-axis and the cross product of the z-axis and the x-axis makes the y-axis (see Fig. 2c). This coordinate basis is used to define a 3D grid at the origin (Figs. 2d and 2e). The size of the grid determines the degree of locality of the representation, whereas the size of each bin determines the level of granularity at which the surface is represented. The bin size was automatically derived as $b_s = \frac{d_{min}}{5}$ and the grid size was selected as 10^3 on the basis of our pairwise correspondence experiments [34].

Once the grid is defined, the surface area of intersection of the mesh with each bin of the grid is recorded in a third order tensor (Figs. 2d and 2e). Each element of the tensor is equal to the surface area of the mesh intersecting the grid bin (Fig. 2f) corresponding to that tensor element. Hodgman's polygon clipping algorithm [15] is used to efficiently find the area of intersection of a bin and the mesh. Our tensor representation is simply a multidimensional table whose entries correspond to the surface area of a local patch of an object. Since most of the bins of the 3D grid are likely to be empty (Fig. 2e), the resulting tensor will have many zero elements. Therefore, the tensor is compressed to a sparse form in order to cut down on memory utilization by approximately 85 percent. The flow chart of the tensor representation is given in Fig. 3. Note that the selection of seed vertices (shaded block) is mainly performed to reduce the number of possible vertex pairs when representing a complete 3D model at high resolution (Section 4).

For reasons of efficiency in terms of time and memory, it is desirable to represent each view with the minimum number of required tensors n_t. We found that a minimum of $n_t = 300$ per view is required for correct pairwise correspondences [34] as well as multiview correspondence (see Section 5.2). Therefore, n_t valid vertex pairs are randomly selected to compute tensors.

2.2 Hash Table Construction

Multiview correspondence is highly dependent on the efficiency of the used matching algorithm. Mamic and Bennamoun [27] carried out a review of the different matching approaches. These include hypothesis and test, matching relational structures, pose clustering, interpretation trees, registration, and geometric hashing. Of these approaches, geometric hashing [25] appears to be the most efficient and appropriate due to the insensitivity of the matching time to the number of views or models in the database. However, geometric hashing, in its crude form, has some drawbacks. First, the hash table must be built for all combinations of four points of each view which has a complexity of $O(n^4N)$ (where n is the number of points per view/model and N is the total number of views/models). Second, the hash table is built with surface data points which makes the matching process sensitive to the resolution and surface sampling. We adopted a variant of the geometric hashing for multiview tensor matching. In our variant, the hash table is efficiently constructed from the n_tN tensors only without going into the combinatorial explosion of the data points. Moreover, the tensors represent local surface patches of the views instead of data points. This makes the hash table and, hence, the matching process independent of the resolution and surface sampling of the views.

Once all the views have been represented with tensors, these tensors are used to fill up a 4D hash table. Three dimensions of the hash table correspond to the i, j, k indices of the tensor elements, whereas the fourth dimension is defined by the angle θ_d of the tensors. θ_d is quantized into bins of $\Delta\theta_d$. The appropriate value for $\Delta\theta_d$ was empirically estimated and was found to be 5 degrees (Section 5). The bins of the 4D hash table are filled up as follows: For each tensor $\mathbf{T}_b$ of each view $\mathbf{M}_a$ (where a, b are index numbers), the tuple (a, b) entry is made in all the bins of the hash table corresponding to the i, j, k indices of the nonzero elements of the tensor and the θ_d of the tensor. Fig. 4a shows the flow chart of the hash table construction.

3 Automatic Multiview Correspondence (Module C)

We present two efficient variants of our automatic multiview correspondence algorithm, i.e., the connected graph and the hypergraph algorithm. The first variant (Section 3.1) makes a single multilevel spanning tree, whereas the second (Section 3.2) variant makes disjoint and unilevel spanning trees first and then connects them to form a multilevel spanning tree hypergraph.

3.1 Connected Graph Algorithm

Fig. 5a shows the trace of the connected graph algorithm for the 33 views of the *Hasi* (Fig. 7). This algorithm selects the mesh $\mathbf{M}_R$ with the maximum surface area as the root node to initialize a spanning tree graph (Fig. 5a, Step 1). The tensors of $\mathbf{M}_R$ are then matched with the tensors of the remaining meshes (nodes) in the search space (Fig. 5a). Details of the matching process are given in Section 3.3. Matching tensors

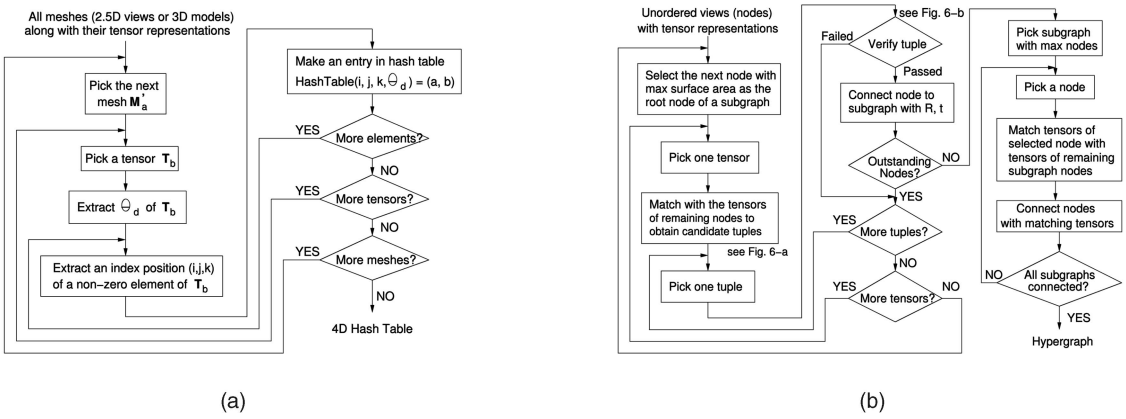


Fig. 4. Flow chart of (a) the 4D hash table construction and (b) the hypergraph algorithm.

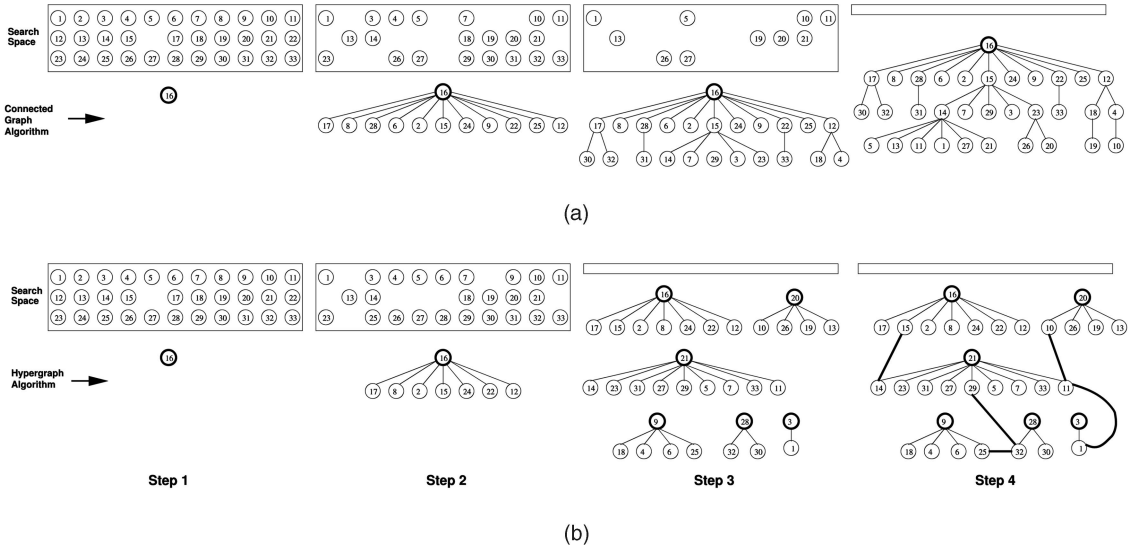


Fig. 5. (a) Trace of the connected graph and (b) hypergraph multiview correspondence algorithms for the 33 views of *Hasi* (Fig. 7). The root nodes and the arcs joining two subgraphs (in the case of hypergraph algorithm) are drawn in bold.

are used to calculate the rigid transformation between their respective nodes (see Section 3.4 for details). The node with the matching tensor is removed from the search space and connected to $\mathbf{M}_R$ with an arc which represents the rigid transformation between the two nodes. Once all the tensors of $\mathbf{M}_R$ have been matched (Fig. 5a, Step 2), another node is selected from the spanning tree and its tensors are matched with the tensors of the remaining nodes in the search space (Fig. 5a, Step 3). This process continues until all the nodes have been added to the spanning tree (Fig. 5a, Step 4). The search space is reduced each time a new node is added to the spanning tree. This provides further efficiency to the algorithm.

3.2 Hypergraph Algorithm

The output of this algorithm is also a spanning tree. However, the construction approach is different. Fig. 4b shows the flow chart of the hypergraph algorithm, whereas Fig. 5b shows its trace. This algorithm initially makes separate subgraphs which are also spanning trees. Each subgraph is made by selecting a root node and connecting other nodes to it by matching the tensors of the root node with the tensors of the remaining nodes in the search space. Details of the matching process are given in Section 3.3. When all the tensors of a root subgraph node have been matched with the remaining nodes in the search space, another subgraph root node is selected from the remaining nodes. Root nodes are selected on the basis of maximum surface area. This process continues until subgraphs have been constructed from all the nodes and there are no more nodes in the search space (Fig. 5b, Step 3). Next, a hypergraph is constructed from these subgraphs as follows: The subgraph with the maximum number of nodes is selected as the root subgraph and the tensors of its nodes are matched with the tensors of the remaining subgraphs as described in Section 3.3. Consequently, all subgraphs are connected to the root subgraph by a single arc (shown bold in Fig. 5b, Step 4) resulting in a hypergraph. Our experiments in Section 3.3 show that the hypergraph algorithm is more efficient.

3.3 Multiview Matching

To simultaneously match a tensor $\mathbf{T}_m$ of a model mesh/node $\mathbf{M}'_m$ with the tensors of the remaining views, the i, j, k indices of its nonzero elements and its θ_d (the angle between the normals of the two vertices used to define $\mathbf{T}_m$) are used to cast votes to all the tuples (a, b) (where a and b are the index numbers of a view and tensor respectively) present at the corresponding index positions in the 4D hash table (Fig. 6a). The tuples that receive fewer votes than half the number of

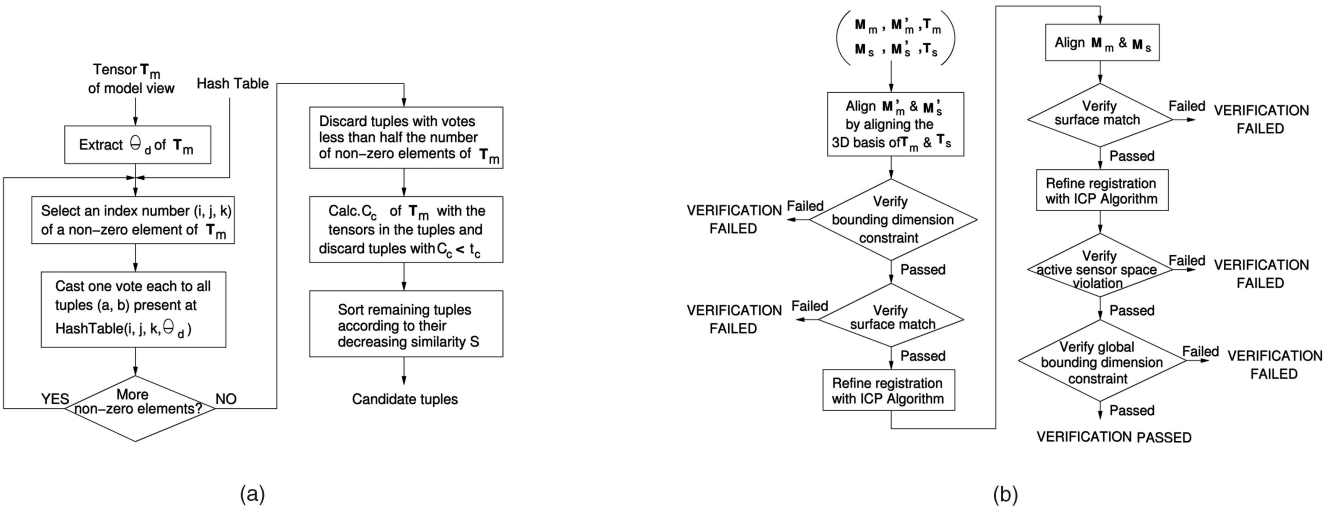


Fig. 6. Flow chart of (a) the multiview matching and (b) the correspondence verification.

nonzero elements of $\mathbf{T}_m$ are dropped. Next, the correlation coefficient C_c is calculated between $\mathbf{T}_m$ and each tensor $\mathbf{T}_s$ of all the remaining tuples using (3). C_c is calculated in the region of overlap (i.e., the elements of $\mathbf{T}_m$ which have a corresponding element in $\mathbf{T}_s$) of the two tensors in order to compensate for occlusions.

$$C_c = \frac{n_q \sum_{i=1}^{n_q} p_i q_i - \sum_{i=1}^{n_q} p_i \sum_{i=1}^{n_q} q_i}{\sqrt{n_q \sum_{i=1}^{n_q} p_i^2 - \left(\sum_{i=1}^{n_q} p_i\right)^2}\sqrt{n_q \sum_{i=1}^{n_q} q_i^2 - \left(\sum_{i=1}^{n_q} q_i\right)^2}}, \tag{3}$$

where p_i and q_i $(i = 1 \ldots n_q)$ are the respective elements of $\mathbf{T}_m$ and $\mathbf{T}_s$ in their region of overlap. Tuples whose $C_c < t_c$ ($t_c = 0.5$, on the basis of our extensive experiments [34]) are discarded and the similarity measure S (4) of the remaining tuples with $\mathbf{T}_m$ is calculated.

$$S = n_q C_c. \tag{4}$$

The remaining tuples are considered as potential correspondences to the tensor $\mathbf{T}_m$ and are verified (details in Section 3.4), starting from the tuple with the maximum value of S. A pair of verified corresponding tuples (say, $\mathbf{M}_m, \mathbf{T}_m$ from the graph and $\mathbf{M}_s, \mathbf{T}_s$ from the search space) is used to calculate the rigid transformation between the two meshes. The new mesh $\mathbf{M}_s$ is removed from the search space and added to the graph by connecting it with an arc to the mesh $\mathbf{M}_m$. The arc represents the transformation (see (5) and (6)) between the two meshes.

3.4 Correspondence Verification

Fig. 6b shows the flow chart of the correspondence verification. A match between a tensor $\mathbf{T}_m$ of model view/node $\mathbf{M}'_m$ and a tensor $\mathbf{T}_s$ of the scene view/node $\mathbf{M}'_s$ in the tuple is verified as follows: First, $\mathbf{M}'_s$ is transformed to the coordinates of $\mathbf{M}'_m$ to form $\mathbf{M}'_{ms}$ (7), using the rotation matrix $\mathbf{R}$ (5) and the translation vector $\mathbf{t}$ (6).

$$\mathbf{R} = \mathbf{B}_s^{\top}\mathbf{B}_m, \tag{5}$$

$$\mathbf{t} = \mathbf{O}_m - \mathbf{O}_s\mathbf{R}, \tag{6}$$

$$\mathbf{M}'_{ms} = \mathbf{M}'_m \cup (\mathbf{M}'_s\mathbf{R} + \mathbf{t}). \tag{7}$$

$\mathbf{B}_m$ and $\mathbf{B}_s$ are the matrices of coordinate basis used to define the model and scene tensors respectively. $\mathbf{O}_m$ and $\mathbf{O}_s$ are their corresponding origins. In (7), $\mathbf{M}'_{ms}$ is the union of the two meshes after the vertices of $\mathbf{M}'_s$ are rotated and translated using $\mathbf{R}$ and $\mathbf{t}$. The bounding dimensions $\mathbf{D}'_{ms}$ of $\mathbf{M}'_{ms}$ are then calculated using (8).

$$\mathbf{D}'_{ms} = \max_{xyz}(\mathbf{V}'_{ms}\mathbf{P}_{ms}) - \min_{xyz}(\mathbf{V}'_{ms}\mathbf{P}_{ms}). \tag{8}$$

In (8), $\mathbf{V}'_{ms}$ is the matrix of the data points (or vertices) of $\mathbf{M}'_{ms}$ and $\mathbf{P}_{ms}$ is the rotation matrix that aligns $\mathbf{V}'_{ms}$ along its principal axes. The dimensions $\mathbf{D}$ of the *object* are then subtracted from $\mathbf{D}'_{ms}$. If the maximum difference[1] between the two is less than a specified tolerance $3b_s$, $\mathbf{M}_m$ and $\mathbf{M}_s$ are also registered (using $\mathbf{R}$ and $\mathbf{t}$) and points on the two meshes that are within a distance of $2d_{res}$ (where d_{res} is the resolution of $\mathbf{M}_i$ and is equal to its mean edge length) are turned into correspondences. If the number of correspondences is more than n_c, the transformation is refined with a variant of the ICP algorithm [36] ($n_c = \frac{\min(n_m, n_s)}{4}$, where n_m and n_s are the number of vertices of $\mathbf{M}_m$ and $\mathbf{M}_s$, respectively). Once again, pairs of points on the two meshes that are within a distance of d_{res} are turned into correspondences. If the number of correspondences is more than $2n_c$, the combined bounding dimensions $\mathbf{D}_{ms}$ of $\mathbf{M}_m$ and $\mathbf{M}_s$ are calculated in a similar way to (8). If $\max(\mathbf{D} - \mathbf{D}_{ms})$ is less than $2d_{res}$, the algorithm proceeds to verify active sensor space violation (see Section 7.3). If any of the above local verification steps fail, the algorithm proceeds to verify the next tuple with the highest value of S.

During the global verification, the combined bounding dimensions $\mathbf{D}_L$ of all the registered meshes in the current spanning tree and the new mesh to be added are calculated in a similar way to (1). If $\max(\mathbf{D}_L - \mathbf{D}) < 4d_{res}$, the new mesh is added to the spanning tree otherwise the algorithm proceeds to test the next tuple with the highest value of S.

1. The combined dimensions of two or more correctly aligned views should not exceed the dimensions of the object.

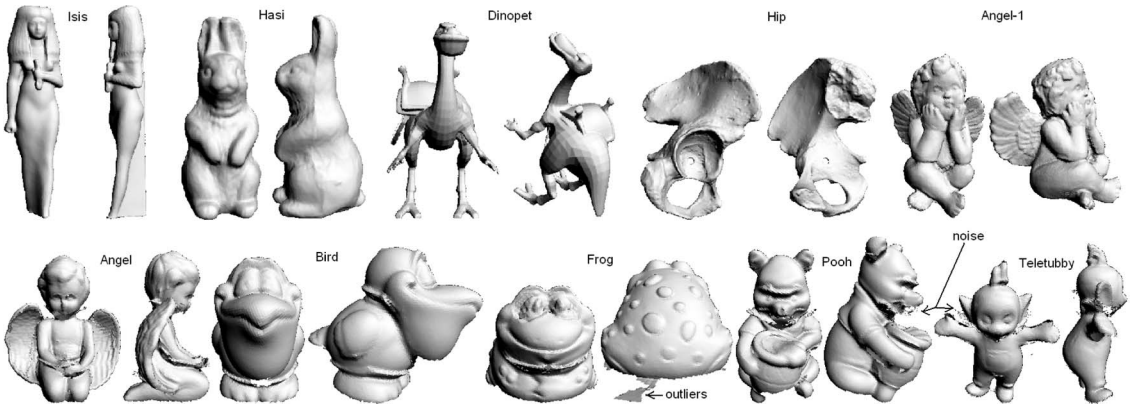


Fig. 7. Three-dimensional modeling results of our algorithm. The noise and outlier points are the result of the propagated sensor errors.

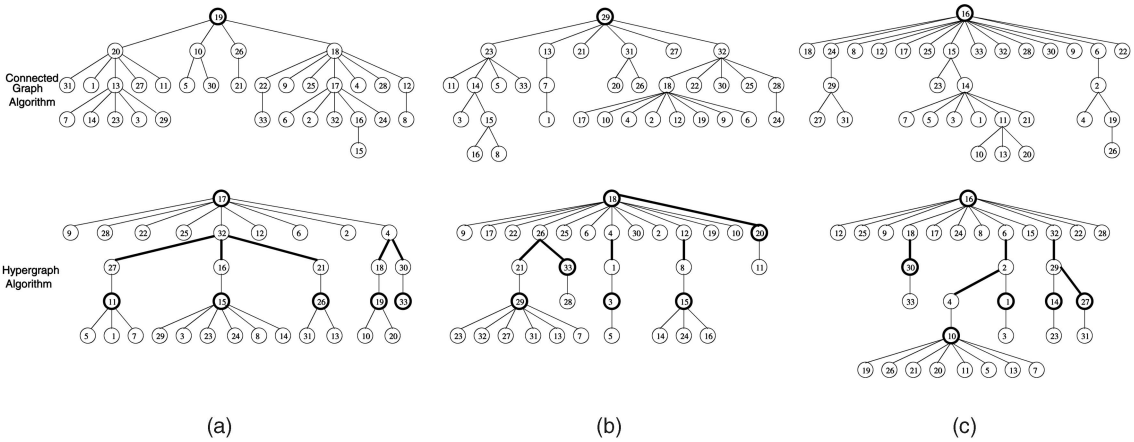


Fig. 8. Output of our connected graph and hypergraph algorithm. The root nodes and the arcs joining two subgraphs are drawn in bold. Timing information of the hypergraph algorithm is given for the Matlab implementation on a 2.4 GHz machine.(a) Isis (4.76 mim), (b) Dinopet (3.48 min), and (c) Hip (6.21 min).

4 Global Registration, Integration, and Reconstruction (Modules D and E)

The spanning tree is used to register all the views in the coordinate basis of the root mesh M_R by concatenating transformations. To avoid the accumulation of registration errors, which may result in seams between distant views of the spanning tree, the registration is refined with a modified version of Williams and Bennamoun's global registration algorithm [41]. Finally, the registered views are merged into a seamless 3D model using VripPack (Volumetric Range Image Processing Package) [38] which uses the volumetric integration algorithm by Curless and Levoy [11] for integration and the marching cubes algorithm [26] for reconstruction.

5 Three-Dimensional Modeling Results and Analysis

Fig. 7 shows some example models resulting from our automatic 3D modeling algorithm. In the case of the *Isis*, *Hasi*, *Dinopet*, and *Hip*, 33 unordered range images of each object were fed to our connected graph algorithm and hypergraph algorithm for multiview correspondence. Fig. 5 and Fig. 8 show the output graphs constructed by the two variants of our algorithm in each case. Notice that the graphs constructed by the connected graph algorithm have a maximum of only four levels, starting from the root node (e.g., node 19 in Fig. 8a) at level zero to the end nodes at level four (e.g., node 15 in Fig. 8a). However, in the case of the *Hip* using the hypergraph algorithm, the maximum number of levels was five. The number of levels is low in either case, but the hypergraph algorithm is more efficient. The advantages of a small number of levels are twofold. First, the spanning tree is quickly and efficiently constructed since the nodes are quickly removed from the search space. Second, the registration errors are only accumulated through a small number of branches. The following sections give a quantitative analysis of our automatic multiview correspondence algorithm according to four important criteria.

5.1 Robustness to the Feature Resolution of the Views

We tested the performance of our multiview correspondence algorithm with varying resolution of the meshes in order to demonstrate its robustness. These experiments were performed on the unordered views of five objects, namely, the *Angel*, *Bird*, *Frog*, *Pooh*, and *Teletubby* (Fig. 7, second row). There were 18 views each of the first four objects and 20 views of the *Teletubby*. In each case, we reduced the resolution of the meshes [16] and represented them with a constant number of 400 tensors per view. Next, we tested if our hypergraph algorithm can automatically establish correspondences between the unordered views of individual objects. The results

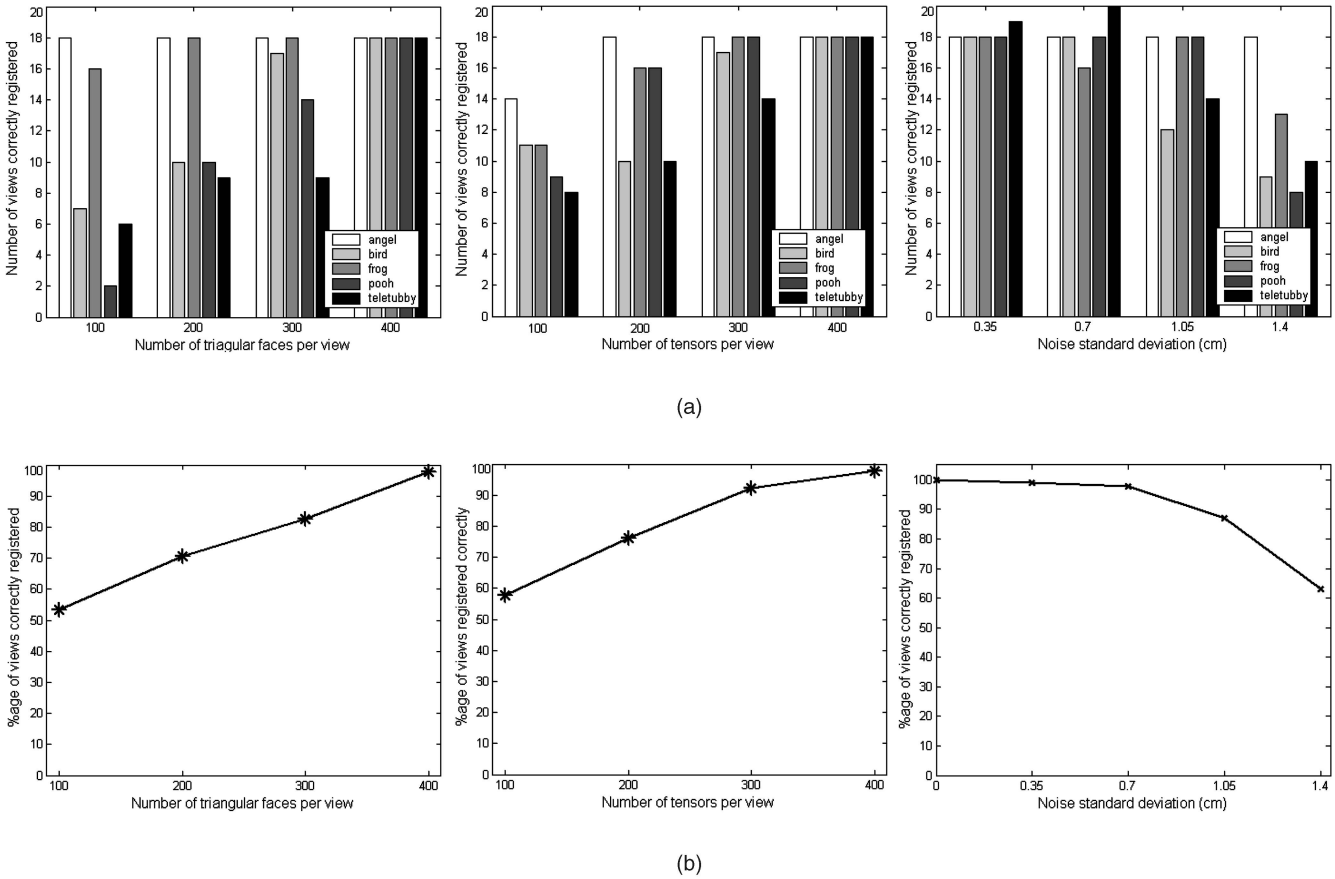


Fig. 9. (a) Individual and (b) combined performance of our multiview correspondence algorithm.

of our experiments were quantified on the basis of the number of views/nodes correctly registered to the root node of the root subgraph (Fig. 9, column one). Our results show that our algorithm has 98 percent performance at a low resolution of 400 faces per view. The failures occur mainly due to insufficient overlap (< 50 percent) between the views [34]. Below 400 faces, the performance of our algorithm degrades gracefully. Note that the views which are not connected to the root subgraph are connected among themselves, forming one or more subgraphs. In such a case, only one arc per subgraph is needed to register them to the solution hypergraph.

5.2 Robustness to the Required Number of Tensors per View

It is desirable to represent each view with a minimum number of tensors for computational efficiency. Therefore, we investigated the minimum required number of tensors per view for a successful multiview correspondence. We performed this experiment on the same data set used in Section 5.1 while keeping their resolution constant at 400 faces per view. The number of tensors per view n_t was varied from 100 to 400 and our hypergraph algorithm was used to automatically register the views. Fig. 9 (column two) shows the number of nodes correctly registered to the root node of the root subgraph as a function of n_t. Note that our algorithm reaches 98 percent performance at only 400 tensors per view. Below this value, the performance of our algorithm degrades gracefully.

5.3 Robustness to Noise

The range images of the five objects in Fig. 7, second row, were acquired with the Minolta Vivid scanner and seemingly contained some inherent noise as well as many outlier points (see Fig. 7). Nonetheless, these views were still correctly registered by our multiview correspondence algorithm. To further test the robustness of our algorithm, we introduced additional Gaussian noise in these range images along the scanner viewing direction (Fig. 10). The spatial resolution of the range images of the *Angel* was $1.4\ cm$, whereas that of the remaining data set was $0.7\ cm$. The average size (mean $\mathbf{D}$) of the *Angel* was $120.7\ cm$, whereas that of the remaining objects was $58.8\ cm$. Next, these views were converted into our tensor representation at 400 faces per view and 400 tensors per view. Our hypergraph algorithm was then used for multiview correspondence between the unordered views of each object. Fig. 9 (column three) shows the performance of our algorithm as a function of increasing Gaussian noise. Note that the performance of our algorithm is almost unaffected by noise up to $\sigma = 0.7$. Any further increase in the noise level degrades the performance of our algorithm; however, the algorithm does not catastrophically fail.

5.4 Robustness to the Number of Views

These experiments were performed on the views of the *Isis*, *Hasi*, *Dinopet*, and *Hip* as the number of views was comparatively large in their case. The views were converted into our tensor representation at 400 faces per view and 300 tensors per view. The results are reported in Fig. 11. They represent the total number of tensors that had to be matched to register the views versus the number of input views. This

(a) (b)

Fig. 10. Example views of the five objects after adding Gaussian noise with (a) $\sigma = 0.7\ cm$ and (b) $\sigma = 1.4\ cm$.

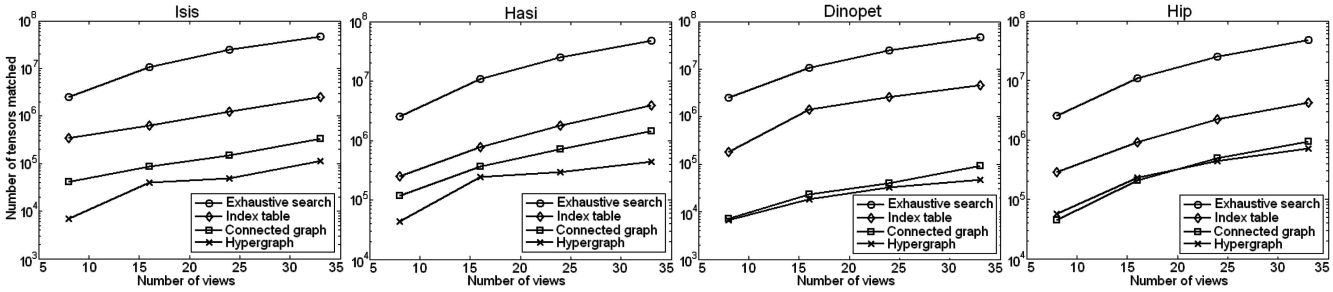


Fig. 11. Comparison of the exhaustive search and the three variants of our multiview correspondence algorithm. Note: The vertical scales are logarithmic.

criterion was chosen in order to compare different variants of our multiview correspondence algorithm with an exhaustive search, in which case, the total number of tensor pairs λ that must be matched is given by $\lambda = C_2^N n_t^2$ (where C stands for combinations and n_t is the number of tensors per view, $n_t = 300$ in this experiment). All three variants of our multiview correspondence algorithm show a significant improvement compared to the exhaustive search. The hypergraph algorithm, however, is the most efficient, with an average improvement factor of 301.5 over the exhaustive search. The improvement factor of the connected graph algorithm was 172.9, whereas that of the index table-based algorithm [29] was only 12.4.

6 The 3D Model Representation and Model Library (Modules F and G)

The model library was built by reconstructing 3D models of free-form objects from their partial views using our automatic algorithm (Fig. 1). This library was then augmented by available 3D models and CAD models, resulting in a model library of 50 objects (Fig. 12). The use of CAD models becomes mandatory when the object to be recognized is unavailable for scanning. However, CAD models generally contain internal details that are irrelevant to our recognition algorithm. Therefore, we generated synthetic views of each CAD model from 62 uniformly distributed viewing directions and merged them again using VripPack [38] to form a seamless 3D model which was comprised of only the visible (external) surface of the CAD model.

Each library model was aligned with its principal axes and scaled so that its mean x, y, z dimension equals $100\ mm$. This made the recognition task more challenging as it is relatively difficult to differentiate between objects of the same size. Our recognition algorithm is not scale invariant (alike the spin images [22]) and can differentiate between similar objects having different scales. To this point, all the 3D models were uniformly sampled since their surfaces were reconstructed with the marching cubes algorithm [26] (using VripPack [38]). In most cases, a 3D model does not require the same high level of detail throughout its surface. Therefore, the rescaled models were decimated for performance reasons at a quadric error of 0.1 [16]. As a result, the library models ended up with different resolutions ranging from 1.4 to $3.0\ mm$.

The simplified 3D models were then represented with tensors (Section 2.1). However, it was not feasible to construct tensors using all the vertices of a model since the average number of vertices per model was 8,000. Therefore, we selected seed vertices on each model by decimating it once more at a quadric error of 30 [16] to get a lower resolution mesh and then selecting the nearest vertices of the original 3D model to the vertices of the lower resolution mesh. The seed vertices are then paired according to the constraints discussed in Section 2.1 as well as the following additional constraints: An upper bound is placed on the angle θ_d between the normals of a pair of vertices, i.e., $5^o < \theta_d < 60^\circ$, in order to increase the chances of both vertices in any pair being visible from a wide range of viewing directions.[2] Moreover, while calculating the tensors, only those triangular faces whose normals make an angle of less than 90 degrees with the z-axis are considered. This ensures that the triangular faces from the back side of the 3D model (which are not visible from the z-axis viewing direction) are not included in the tensor computation. Finally, n_t (the number of tensors per model) is increased to 2,000 in the case of full 3D models because they have more surface area compared to their single views. Increasing the value of n_t also improves the online recognition time. The size of each tensor was $15 \times 15 \times 15$ and the size of each bin was $5\ mm$. The size of the tensor was increased to achieve higher discriminating capability required for 3D object recognition.

Fig. 12 shows our 3D model library,, which is diverse in the sense that it contains a wide variety of 3D models, including free-form objects (e.g., animals) and CAD objects (e.g., vehicles). Some models are very similar, e.g., 37 and 38. Moreover, models 17, 19, and 31 have missing surfaces at their bottoms because they were reconstructed from insufficient views.

2. Pairing of a vertex from the front of the model to a vertex on its back, for example, is not useful for recognition as these vertices are not visible from any single viewing direction.

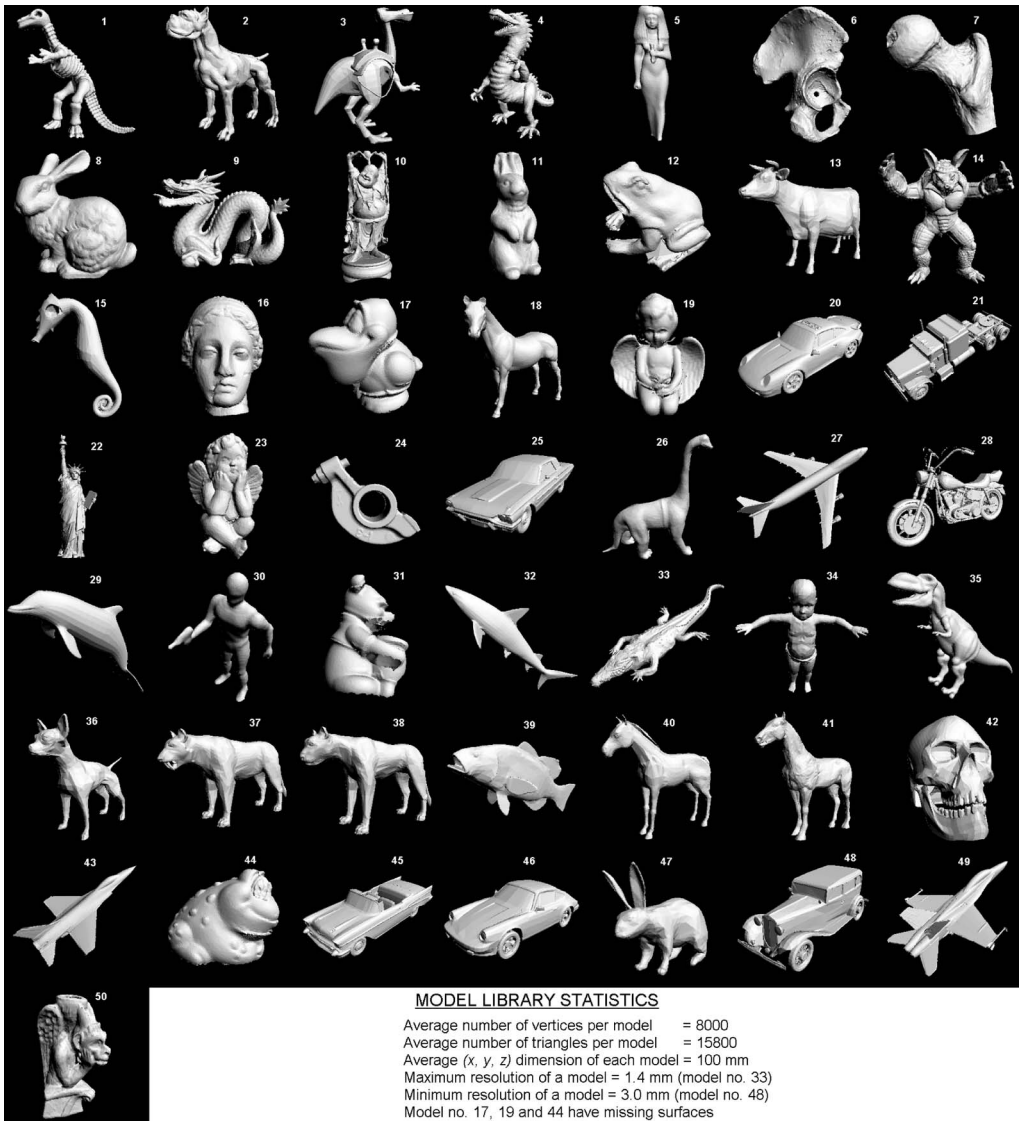


Fig. 12. Three-dimensional model library.

7 Three-Dimensional Object Recognition Algorithm

7.1 Scene Representation (Module H)

The input to our 3D object recognition and segmentation algorithm is a range image (point cloud) of a scene and the output is the recognition and segmentation of library objects in the scene. Fig. 13 shows the flow chart of the recognition and segmentation algorithm. Module H of Fig. 13 shows the flow chart of the scene representation. The input range image is first converted into a triangular mesh $\mathbf{M}_s$. $\mathbf{M}_s$ is decimated at a quadric error of 0.1 [16] to get an optimal mesh $\mathbf{M}'_s$. $\mathbf{M}'_s$ is decimated [16] at a quadric error of 30 to get a low resolution mesh $\mathbf{M}''_s$. $\mathbf{M}''_s$ is used to select seed vertices in $\mathbf{M}'_s$ which are then paired according to the distance and angle constraints (Section 2.1) to get a list of valid vertex pairs L_v. The same values of distance constraint (d_{min} to d_{max}) and angle constraint (5^o to 60^o) must be used for the pairing of vertices during offline 3D model representation and during online representation of a scene. Next, a pair is randomly selected from L_v and used to construct a tensor $\mathbf{T}_s$ from $\mathbf{M}'_s$ (Section 2.1).

7.2 Matching (Module J)

A pair of vertices can be ordered in two ways each resulting in a different coordinate basis. These coordinate bases are rotated 180 degrees on their z-axis with respect to each other. As a result, we calculate $\mathbf{T}_s$ for only one ordering of the vertices and rotate $\mathbf{T}_s$ by 180 degrees on its z-axis to get the tensor for the second ordering of the vertices. Each $\mathbf{T}_s$ is matched with the model library using the multiview matching scheme (Section 3.3). The result of each matching is a list of candidate tuples (a, b) (where a and b are the index numbers of a model and tensor, respectively) sorted according to their degree of similarity S (4) with $\mathbf{T}_s$. Each candidate tuple is a hypothesis that model a is present in the scene. The hypothesis is verified (Section 7.3) in a hierarchical fashion, keeping the most expensive verification step to the last. A confidence measure is calculated and continuously updated at each verification step. For a hypothesis to be finally accepted, it must pass each individual verification step as well as maintain an overall acceptable value of confidence.

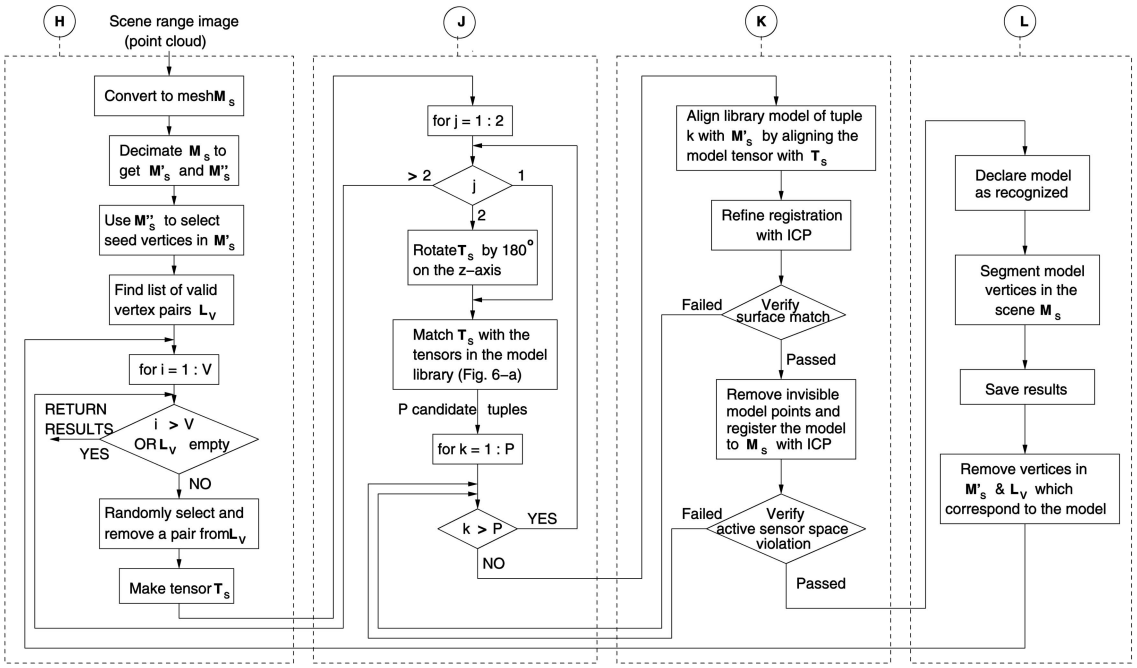


Fig. 13. Flow chart of the 3D object recognition and segmentation algorithm. Modules H, J, K, and L of Fig. 1 have been expanded in this flow chart.

7.3 Hypothesis Verification (Module K)

During the first verification step, model a from the library is transformed to the scene mesh $\mathbf{M}'_s$ by aligning the coordinate frame of its tensor b with the coordinate frame of $\mathbf{T}_s$ (Section 3.4). This transformation is refined with the ICP algorithm [4] and the surface match between the model and the scene is verified. If the model aligns accurately with a portion of the scene surface, the algorithm proceeds to the next stage. Otherwise, the next tuple is selected for verification. The accuracy of alignment α is calculated according to (9).

$$\alpha = \frac{\text{corresponding vertices of model with } \mathbf{M}'_s}{\text{total vertices of model}}, \tag{9}$$

$$f_1 = \alpha S. \tag{10}$$

In (9), a model vertex and $\mathbf{M}'_s$ vertex are considered corresponding if their mutual distance is less than twice the resolution of the model. In (10), f_1 is the confidence measure and S is the similarity value (4) between $\mathbf{T}_s$ and the tensor b. The confidence value f_1 of a hypothesis is proportional to the similarity measure between the tensors and the accuracy of alignment resulting from the tensors. A hypothesis is allowed to proceed to the next verification step only if α and f_1 satisfy their respective minimum required thresholds. Selecting a high threshold for α will reject a correct hypothesis for objects which are highly occluded in the scene, whereas a low value will allow many incorrect hypotheses to proceed to the next verification step. In our experiments, we selected two thresholds for α. For values of $\alpha < t_{\alpha 1}$, all hypotheses were rejected and, for values of $\alpha > t_{\alpha 2}$, all hypotheses were allowed to proceed to the next step. Hypothesis with values of $t_{\alpha 1} < \alpha < t_{\alpha 2}$ were allowed to proceed to next step only if $f_1 > t_{f1}$. We selected $t_{\alpha 1} = 0.15$, $t_{\alpha 2} = 0.3$, and $t_{f1} = 20$ on the basis of over 650 verification trials using a labeled independent training set.

The second and third steps verify if the active space of the sensor is violated by the transformed model [10]. Two types of active space violations can occur: First, when part of the model is transformed into a region of the scene where the sensor does not detect any surface even though that region is visible to the sensor; second, when part of the model is transformed into the space between the sensor and a detected surface of the scene. The first type of active space violation can occur when the model is misaligned with the scene and in the presence of sensor errors (e.g., failure of the sensor to detect black surfaces, a common problem with structured light-based sensors). However, the severity of this error is generally high in the case of a misaligned model. The second type of violation, however, can only occur due to a misaligned model. A misaligned model is an indication that the hypothesis is incorrect and should therefore be rejected. The following paragraphs give the details of the second and third verification steps.

The second verification step proceeds as follows: The transformation resulting from the first step is used to align the model to the high resolution scene $\mathbf{M}_s$. Next, all the model vertices that are not visible to the sensor are removed. Such vertices include those which fall outside the field of view of the sensor and those which are self-occluded or occluded by other objects. The registration between the remaining model vertices and $\mathbf{M}_s$ is refined with ICP [4] and the alignment α_2 between the two is calculated using (11).

$$\alpha_2 = \frac{\text{corresponding vertices of model with } \mathrm{M_s}}{\text{total visible vertices of model}}. \tag{11}$$

Ideally, α_2 should be equal to 1.0 for a correct hypothesis since every visible model vertex should find a corresponding vertex in the scene. In practice, α_2 varies between 0.8 and 1.0 for a correct hypothesis. A low value of α_2 indicates that the hypothesis is incorrect. However, a high value does not guarantee that the hypothesis is correct. Therefore, if $\alpha_2 < 0.8$, the hypothesis is rejected. Otherwise, the visible vertices of the model and the vertices of $\mathbf{M}_s$ are mapped to the sensor's retinal plane (e.g., the xy plane by eliminating

their z components, in case of an orthographic sensor). Next, the 2D distances between the model vertices and their respective nearest scene vertices are calculated. These distances are sorted and the longest n_1 distances are used to calculate the active space violation measure β using (12).

$$\beta = \frac{1}{2n_1 d_s} \sum_{i=n-n_1}^{n} d_i, \tag{12}$$

$$f_2 = \frac{\alpha_2}{\beta}. \tag{13}$$

In (12), n is the number of remaining (visible) vertices of the model, $n_1 = \text{round}(n/10)$, d_s is the resolution of the scene, and d_i is the distance of the ith vertex of the model from its nearest vertex in the scene. Division by d_s is performed to make β scale independent and, hence, a constant threshold can be selected for it. The value of β is high when the active space of the sensor is violated by the transformed model. In (13), f_2 is the new confidence measure. The hypothesis is allowed to proceed to the next verification step only if $\beta < 1.0$ and $f_2 > 1.0$.

The final verification step proceeds as follows: The model and $\mathbf{M}_s$ are placed in a z-buffer and the number of vertices of the model that lie in the space between the sensor and $\mathbf{M}_s$ are calculated. Ideally, no model vertex should come in between the sensor and the scene for a correct hypothesis. However, due to sensor errors few vertices may still be found. We selected a conservative threshold of 30 for this verification step. A hypothesis was finally accepted if fewer than 30 vertices of the model were found between the sensor and the scene. Otherwise, it was rejected. If the hypothesis fails verification, the next tuple with the highest value of S is selected for verification. If none of the hypotheses (tuples) can pass all the verification steps, another pair of vertices is randomly selected from L_v to compute a tensor and the whole process of matching is repeated (see Fig. 13, Module K).

7.4 Recognition and Segmentation (Module L)

An accepted hypothesis (tuple) results in the identification, pose estimation, and segmentation of the library object (a of the tuple) in the scene $\mathbf{M}_s$. These recognition and segmentation results are saved. The vertices corresponding to the 3D model are removed from the decimated mesh of the scene ($\mathbf{M}'_s$) as well as from the list of valid vertex pairs L_v, which not only improves the efficiency of the algorithm but also increases the odds for the recognition of the remaining scene objects.

After the recognition and segmentation of an object from the scene, another vertex pair is randomly selected from the remaining list of valid vertex pairs L_v to compute a tensor and the matching process is repeated. This process continues until the complete scene has been segmented or no further library objects are present in the scene. There are two stopping conditions for the algorithm: first, when all the vertex pairs in L_v have been tested; second, when no library object is recognized after consecutively matching V number of tensors (V was set to 50 in all our experiments). In this case, the algorithm stops only if no hypothesis can make it to the computation of confidence measure f_2. Otherwise, the hypothesis with the maximum value of f_2 is accepted and the algorithm proceeds.

Note that, after the recognition and segmentation of one or more objects in the scene, further tensors $\mathbf{T}_s$ are calculated using the segmented scenes so that the vertices, as well as the surface meshes of the recognized objects, are not reconsidered. However, during the active space violation tests, the original unsegmented scene $\mathbf{M}_s$ is taken into consideration.

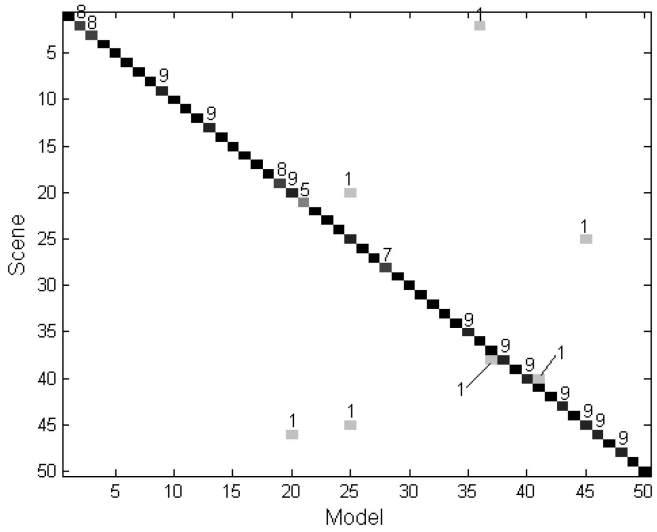


Fig. 14. Confusion matrix of 10 recognition trials per object. Recognition rate (TP) was 95 percent. FP = 1.2 percent and FN = 3.8 percent.

8 Three-Dimensional Object Recognition and Segmentation Results

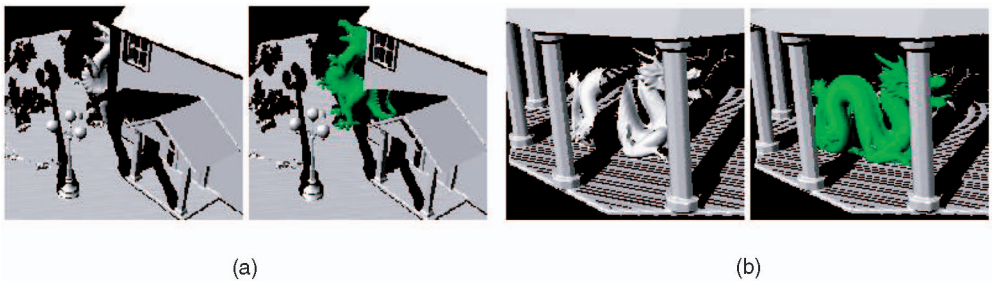
We performed our experiments on synthetic and real data. Results using synthetic data are presented in Sections 8.1 to 8.3. Results using real data and comparison with the spin image recognition [22] are presented in Section 8.4.

8.1 Scenes with a Single Object and No Clutter

In the first experiment, a single view of each library object (Fig. 12) was used for its recognition. The scenes contained only self-occlusions and no clutter. This experiment was repeated for 10 different views (range images) of each library object, resulting in 500 recognition trials. Ten views of each model were synthetically generated using VripPack [38]. Each time, a model was rendered from an arbitrary viewing direction and its visible surface was reconstructed on a volumetric grid using the marching cubes algorithm [26]. This resulted in a completely different triangular mesh of the model's visible surface. Note that some of these views suffered from a high level of self-occlusion. The confusion matrix of these 500 recognition trials is reported in Fig. 14. The darkness of each pixel (number of times recognized) is written on its top unless it is 10 or zero. There were no True Negatives (TN) in this experiment since every scene contained a library object. As a result, False Negatives (FN) were calculated using (14).

$$\text{FN} = \text{Recognition Trials} - \text{TP} - \text{FP}. \tag{14}$$

The overall recognition (TP True Positive) rate was 95 percent. The FP rate was 1.2 percent and the FN rate was 3.8 percent. Notice that the lowest number of TPs occurred for models 21 and 28. Both of these models have complex details causing a lot of discontinuities in their meshes, which was the main reason for their high FN rate. The seven FP are: Models 2, 20, 25, 45, 46, 38, 40 recognized as models 36, 25, 45, 25, 20, 37, 41, respectively. Notice that



(a) (b)

Fig. 15. Recognition of single occluded objects in cluttered scenes. The recognized objects have been superimposed by their 3D models (4 and 9) from the library.

every object (under high self-occlusion) was confused with another closely resembling object, e.g., 38 was confused with 37 which is visually similar especially when viewed from the top.

8.2 Scenes with Multiple Objects in the Presence of Clutter and Occlusions

In this experiment, we placed 3D models from the library of Fig. 12 with other nonlibrary models using Scanalyze [38] and reconstructed the visible surface of the scene from a single viewing direction using the approach described in Section 8.1, only this time, there were multiple 3D models (including library and nonlibrary models) occluding each other and causing clutter. The scene was then fed to our algorithm for the recognition and segmentation of library objects. Note that, in each trial, our recognition algorithm had to recognize any of the 50 library objects in a scene. This also included multiple occurrences of a particular library object. No prior knowledge was provided to the algorithm regarding which library objects were present in a scene. The purpose of our tests, as opposed to the spin images recognition [22] (see Section 8.4 for details), was not to detect the presence or absence of a prespecified object in a scene. Rather, it was to identify all objects that are present in a scene without any prior knowledge.

Fig. 15 illustrates the recognition of a single library object each in two different scenes. In each case, the object is occluded by nonlibrary objects. Fig. 16 shows the trace of our 3D object recognition and segmentation algorithm. The scenes in Fig. 16 contain four library objects. Each time an object is recognized in the scene, its 3D model from the library is registered to the scene and its relevant surface is segmented and removed, hence reducing the amount of clutter in the scene. Notice that the segmentation by recognition is very accurate and is also capable of segmenting discontinuous surfaces belonging to the same object (Fig. 16f). Once all the library objects have been recognized, their complete models are placed at their appropriate pose in the original scene (Figs. 16k and 16l).

Fig. 17 shows additional recognition and segmentation results. In Fig. 17d, object 37 and 38 were correctly recognized despite their high similarity. In Fig. 17b, object 1 is present four times, out of which three have been correctly recognized. This shows that our algorithm can differentiate between multiple instances of the same object in the scene.[3] Some of the objects are correctly recognized despite being highly occluded, for example, Fig. 17e object 4, Fig. 17a object 17, and Fig. 17c object 11. Object 2 could not be correctly recognized in Figs. 17c and 17f because of its high level of occlusion (87 percent accroding to (15)).

We tested 60 synthetic scenes containing multiple objects causing clutter and occlusions, and quantized the recognition results based on the level of occlusion and clutter for each object. For comparison reasons, we defined occlusion according to Johnson and Hebert's formula[4] [22] as given in (15). However, a more appropriate definition was used for clutter, as given in (16).

$$\text{occlusion} = 1 - \frac{\text{model surface patch area in scene}}{\text{total model surface area}}, \quad (15)$$

$$\text{clutter} = 1 - \frac{\text{model surface patch area in scene}}{\text{total surface area of scene}}. \quad (16)$$

Since our algorithm can also segment the recognized models from the scene, the value of occlusion and clutter was automatically calculated for correctly recognized objects. Those objects which could not be automatically recognized, due to a high value of occlusion were manually segmented in order to calculate their occlusion and clutter values. Note that false positives can occur on library or nonlibrary objects. In either case, we calculate the occlusion of the falsely recognized object using (15). The results of the recognition trials on 60 cluttered scenes are compiled in Figs. 18a and 18b with respect to occlusion and clutter, respectively. From Fig. 18a, it is clear that our algorithm has an average recognition rate (true positives) of 95 percent with up to 78 percent occlusion. For higher than 78 percent occlusions, the recognition rate drops rapidly. The small rise in the false positives occurred as a result of similarities between certain models in the presence of occlusions. For example, model 38 was confused with 37 and model 40 was confused with 41. These false positives are not unexpected as these models are highly similar and it is hard to differentiate between them from certain viewpoints. From the behavior of the false positives curve, one can conclude that the false positives rate is a function of both occlusions and similarities between the library objects. At lower levels of occlusion, objects with high similarities can also be differentiated. Objects with low similarities, on the other hand, can be differentiated even at greater levels of occlusions. The false negatives in Fig. 18a start occurring at 72 percent occlusion and rapidly grow beyond 78 percent occlusion. Note that there are no false negatives below 72 percent occlusion.

The performance of our recognition algorithm with respect to clutter (Fig. 18b) indicates that our algorithm is

3. The same scene was tested with the spin images algorithm [22] which could detect only a single occurrence of the object.

4. The occlusion reported in Fig. 18a, Fig. 19b, and [22] should be normalized to $\frac{\text{occlusion}-50}{0.5}$ to measure occlusion relative to the maximally possible visible surface area of an object.

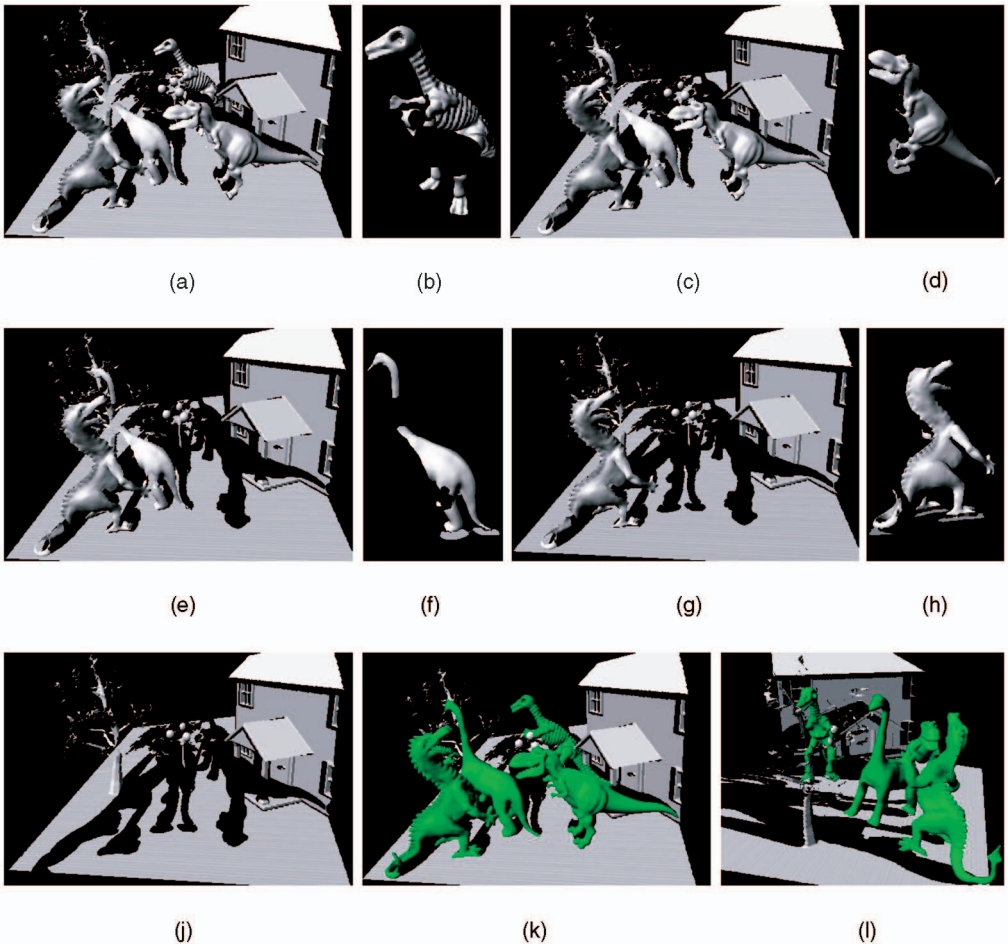


Fig. 16. Trace of our recognition algorithm. The objects (no. 1, 4, 26, and 35) are recognized one by one and segmented and finally superimposed by their complete 3D models (green) in (k). The recognized scene is shown from a different angle in (l).

basically independent of the amount of clutter present in the scene. The drop in the recognition rate beyond 90 percent clutter was mainly because all the objects which fell into this category were highly occluded.

8.3 Efficiency with Respect to Time

The time required by our algorithm for the recognition and segmentation of a cluttered scene is a function of many variables, including the size of the scene, the number of objects in the scene, the number of tensors matched before an object is recognized, and, finally, to some extent, the shape of the objects. If a scene contains objects which have local surface patches that are similar in 3D to the local surface patches of many library objects, their tensors would result in comparatively more candidate tuples which must be verified. This does not result in false positives but only adds a little to the computation time. In our experiments, the average time taken by a Matlab implementation of our algorithm to recognize and segment a single object in a cluttered scene was less than 2 minutes on a 2.4 GHz Pentium IV machine with 1 GB memory. We believe that this time will be significantly reduced with a C++ implementation.

The performance of our algorithm was tested by varying the library size and keeping the remaining parameters constant, i.e., each time the same scene was used and a single tensor was matched. Fig. 18c shows the recognition and segmentation time measured on a real clock for a single library object in a cluttered scene (Fig. 17b) versus the library size. The recognition time is not affected since our algorithm uses a hash table to perform a simultaneous one-to-many matching rather than a one-to-one matching (in which case, the matching time is linear with respect to the number of library models). Note that the recognition time of the final system may still slightly increase since a large number of candidate tensors need to be kept until the verification stage. An unavoidable consequence of an extremely large library size is that the chances of tensors from different objects being nearly similar are likely to increase causing collisions in the hash table. Therefore, as the hash table gets denser, the recognition time will begin to increase, though sublinearly with the library size.

8.4 Tests on Real Data and Comparison with Spin Image Recognition

We compared the performance of our algorithm with the spin image recognition algorithm [22] using real data acquired with the Minolta Vivid 910 scanner, which has a resolution of 640×480. The library of Fig. 12 was augmented by five more models of real objects increasing the library size to 55 models. Fig. 19a shows the 2D images of the five objects and their 3D models. We intentionally scanned the rhino from insufficient viewpoints[5] so that its 3D model contains large holes after reconstruction. This was done to test the robustness

5. The rhino was scanned from the sides and front only. It was not scanned from the top, bottom, and back.

Fig. 17. Recognition and segmentation results of our algorithm on six scenes containing clutter and occlusions.

of our algorithm and the spin images algorithm [22] to incomplete 3D models. As a consequence of insufficient scanning, the views of the rhino were required to be manually registered as they did not have sufficient overlap. However, the remaining four objects were scanned from sufficient viewpoints and their 3D models were constructed using our automatic 3D modeling algorithm. All five models were then represented with tensors and added to the model library as described in Section 4. The hash table was also updated accordingly. The ability of our system to learn and add new models without any major modification to the overall model library is a great advantage.

Next, 50 real scenes were generated by randomly placing four or five of the real objects together in a scene. Each scene was scanned from a single viewing direction by the Minolta scanner generating a dense point cloud and was fed to our algorithm for recognition and segmentation. Note that these scenes only contained the five real objects, but our algorithm was effectively trying to identify any of the 55 library objects during recognition. Fig. 20 shows six example scenes and their corresponding recognition results. As expected, our recognition and segmentation algorithm performs equally well on real data. Note that the tight placement of the objects has resulted in complex scenes and yet the recognition and segmentation are quite accurate.

For comparison, we tested the same 50 real scenes on the spin image recognition algorithm [22] (code available at [28]). For this purpose, the model library was processed to make the resolution of all the models equal. Moreover, recognition without compression was used so that the spin images algorithm can operate at its peak performance in terms of recognition rate [22]. After compiling the results, we observed that the spin images algorithm completely failed to recognize the rhino (Fig. 19a) in any of the 50 scenes. This was possibly because the 3D model of the rhino contained large holes due to insufficient scanning. These holes, however, did not cause

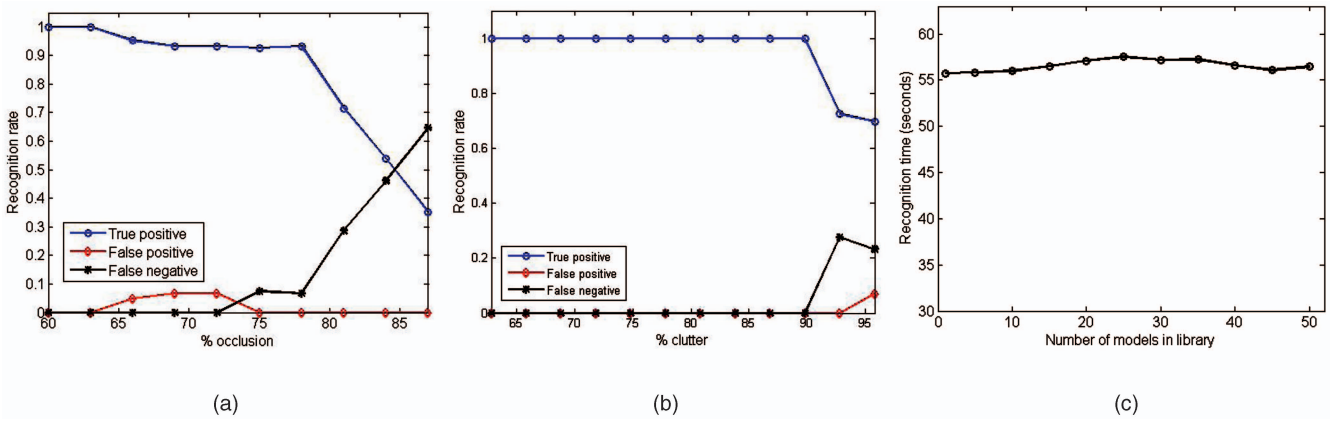


Fig. 18. Recognition rate of our algorithm versus (a) occlusion and (b) clutter. (c) Recognition time of our algorithm based on matching a single tensor from the scene versus the number of models in the library.

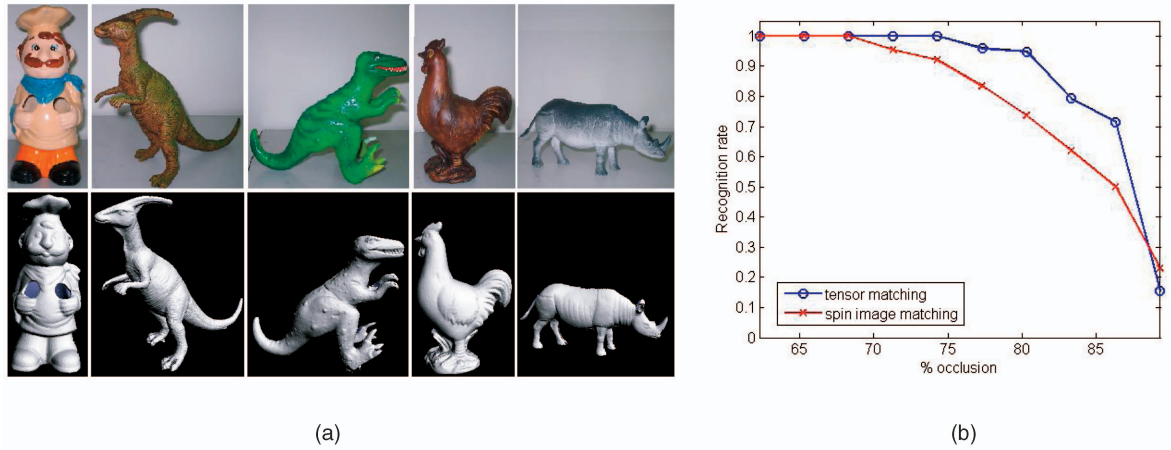


Fig. 19. (a) Two-dimensional images of real objects (first row) and their corresponding 3D models (second row). (b) Recognition rate versus occlusion (on 50 real scenes) showing that our algorithm performed better than the spin images [22].

any problem for our recognition algorithm as the rhino was correctly recognized in 24 out of 28 scenes (the rhino was more than 85 percent occluded in the four scenes where it could not be recognized). Therefore, we excluded the rhino from the recognition results of both algorithms and performed their quantitative comparison using the remaining results. Fig. 19b shows the recognition rates of the two algorithms as a function of occlusion. Our algorithm outperforms the spin images by maintaining a higher recognition rate as the amount of occlusion increases. The average recognition rate of our algorithm was 96.6 percent and that of the spin images was 87.8 percent with up to 84 percent occlusion. The recognition rates versus clutter are not reported as both algorithms are independent of the amount of clutter in scenes (Fig. 18b and [22]).

It is important to point out that, in [22], the spin images algorithm was searching for a **single known** object at a time[6] (see [22] Section 4.2). Therefore, the experiments in [22] can be termed as target detection, which is a much simpler task compared to recognition. Note that the recognition rate of our algorithm reported in Fig. 19b is higher than the target detection rate of the spin images reported in [22] as well. Another important point to note is that the recognition results of our algorithm are based on matching a maximum of 250 tensors per scene whereas those reported for the spin images (Fig. 19b) are based on matching an average of 4,000 spin images per scene. The average recognition time per scene of the spin images [22] was approximately 480 minutes, whereas that of our algorithm (including the segmentation of the scene) was less than 6 minutes on the same machine (using the same real scenes) despite the fact that the spin images algorithm was implemented in C++ and our algorithm was implemented in Matlab. The recognition time of the spin images linearly increases with the library size [22], whereas our algorithm's recognition time is comparatively less sensitive to the library size (Section 8.3).

6. This was confirmed through communication with A.E. Johnson, author in [22].

9 Conclusion

We presented a fully automatic 3D model-based free-form object recognition and segmentation algorithm. Our major contribution in the offline 3D modeling phase is a multiview correspondence algorithm which automatically registers *unordered* views of an object with $O(N)$ complexity. We demonstrated the robustness of this algorithm to numerous important criteria. Our major contribution in the online phase is an efficient algorithm for automatic 3D object recognition and segmentation in the presence of clutter and occlusions. Experiments were performed on synthetic and real data and an overall recognition rate of 95 percent was achieved. Comparison with the spin image recognition algorithm [22] revealed that our algorithm is superior both in terms of recognition rate and efficiency. We also demonstrated that

Fig. 20. Recognition results of our algorithm on six real scenes. These scenes do not have any background due of the limited range of the Minolta scanner. All objects are correctly recognized except for the chicken in (d) and the chef in (e).

our algorithm's recognition time is not sensitive to the size of the model library as opposed to the spin images [22].

Acknowledgments

The authors would like to acknowledge: CMU for providing range data (model 23 and 30 of Fig. 12), 3D model (13), the mesh reduction, and the spin image recognition code; Stanford University for providing 3D models (8, 9, 10, and 14), VripPack and Scanalyze softwares; the OSU for providing range data (model 17, 19, 31, and 44) [5]; Universität Stuttgart for providing range data (model 1 to 7, 11, 12, 15, 20, and 22); Cyberware for providing 3D models (16, 17, and 24); and Princeton University for providing 3D models (21, 25 to 29, 32 to 43, 45 to 50). They would also like to thank Ashley Chew for his help in executing the spin images code and Mark Walters for providing access to the Minolta scanner. They are also grateful to A.E. Johnson for responding to their queries regarding the spin images. This research is sponsored by ARC grant DP0344338.

References

[1] A.P. Ashbrook, R.B. Fisher, C. Robertson, and N. Werghi, "Finding Surface Correspondence for Object Recognition and Registration Using Pairwise Geometric Histograms," *Int'l J. Pattern Recognition and Artificial Intelligence,* vol. 2, pp. 674-686, 1998.

[2] S. Avidan, Y. Moses, and Y. Moses, "Probabilistic Multi-View Correspondence in a Distributed Setting with No Central Server," *Proc. European Conf. Computer Vision,* vol. 4, pp. 428-441, 2004.

[3] P. Besl, *Machine Vision for Three-Dimensional Scenes.* pp. 25-71, Academic Press, 1990.

[4] P.J. Besl and N.D. McKay, "Reconstruction of Real-World Objects via Simultaneous Registration and Robust Combination of Multiple Range Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 2, pp. 239-256, Feb. 1992.

[5] R.J. Campbell and P.J. Flynn, "A WWW-Accessible 3D Image and Model Database for Computer Vision Research," *Empirical Evaluation Methods in Computer Vision,* pp. 148-154, 1998.

[6] R.J. Campbell and P.J. Flynn, "A Survey of Free-Form Object Representation and Recognition Techniques," *Computer Vision and Understanding,* vol. 81, no. 2, pp. 166-210, 2001.

[7] O. Carmichael, D. Huber, and M. Hebert, "Large Data Sets and Confusing Scenes in 3-D Surface Matching and Recognition," *Proc. Int'l Conf. 3-D Digital Imaging and Modeling,* pp. 358-367, 1999.

[8] C. Chen, Y. Hung, and J. Cheng, "RANSAC-Based DARCES: A New Approach to Fast Automatic Registration of Partially Overlapping Range Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 11, pp. 1229-1234, Nov. 1991.

[9] C.S. Chua and R. Jarvis, "3D Free-Form Surface Registration and Object Recognition," *Int'l J. Computer Vision,* vol. 17, pp. 77-99, 1996.

[10] C.S. Chua and R. Jarvis, "Point Signatures: A New Representation for 3D Object Recognition," *Int'l J. Computer Vision,* vol. 25, no. 1, pp. 63-85, 1997.

[11] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," *Proc. SIGGRAPH Conf.,* pp. 303-312, 1996.

[12] R. Donamukkala, D. Huber, A. Kapuria, and M. Hebert, "Automatic Class Selection and Prototyping for 3-D Object Classification," *Proc. Int'l Conf. 3-D Digital Imaging and Modeling,* pp. 64-71, 2005.

[13] C. Dorai and A.K. Jain, "COSMOS: A Representation Scheme for 3D Free-Form Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 10, pp. 1115-1130, Oct. 1997.

[14] P.J. Flynn and A.K. Jain, "CAD-Based Computer Vision: From CAD Models to Relational Graphs," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 2, pp. 114-132, Feb. 1991.
[15] J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics-Principles and Practice.* Addison-Wesley, 1990.
[16] M. Garland and P.S. Heckbert, "Surface Simplification Using Quadric Error Metrics," *Proc. SIGGRAPH Conf.,* pp. 209-216, 1997.
[17] M. Hebert, K. Ikeuchi, and H. Delingette, "A Spherical Representation for Recognition of Free-Form Surfaces," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 7, pp. 681-690, July 1995.
[18] G. Hetzel, B. Leibe, P. Levi, and B. Schiele, "3D Object Recognition from Range Images Using Local Feature Histograms," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 394-399, 2001.
[19] K. Higuchi, M. Hebert, and K. Ikeuchi, "Building 3-D Models from Unregistered Range Images," *Proc. IEEE Int'l Conf. Robotics and Automation,* vol. 3, pp. 2248-2253, 1994.
[20] D. Huber and M. Hebert, "3D Modeling Using a Statistical Sensor Model and Stochastic Search," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 858-865, 2003.
[21] D. Huber, A. Kapuria, R. Donamukkala, and M. Hebert, "Parts-Based 3D Object Recognition," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 82-89, 2004.
[22] A.E. Johnson and M. Hebert, "Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 5, pp. 674-686, May 1999.
[23] T. Joshi, J. Ponce, B. Vijayakumar, and D. Kriegman, "Hot Curves for Modeling and Recognition of Smooth Curved 3D Objects," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 876-880, 1994.
[24] S.B. Kang and K. Ikeuchi, "The Complex EGI: A New Representation for 3D Pose Determination," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, pp. 707-721, 1993.
[25] Y. Lamdan and H. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognitin Scheme," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 238-249, 1988.
[26] W. Lorensen and H. Cline, "A High Resolution 3D Surface Construction Algorithm," *Proc. SIGGRAPH Conf.,* pp. 163-169, 1987.
[27] G. Mamic and M. Bennamoun, "Representation and Recognition of Free-Form Objects," *Digital Signal Processing,* vol. 12, pp. 47-76, 2002.
[28] "Mesh Tool Box," Vision and Mobile Robotics Laboratory, Carnegie Mellon Univ., http://www-2.cs.cmu.edu/~vmr/software/meshtoolbox/downloads.html, 2004.
[29] A.S. Mian, M. Bennamoun, and R.A. Owens, "From Unordered Range Images to 3D Models: A Fully Automatic Multiview Correspondence Algorithm," *Theory and Practice of Computer Graphics,* pp. 162-166, 2004.
[30] A.S. Mian, M. Bennamoun, and R.A. Owens, "Automatic Multiview Coarse Registration of Range Images for 3D Modeling," *Proc. IEEE Conf. Cybernetics and Intelligent Systems,* vol. 1, pp. 158-163, 2004.
[31] A.S. Mian, M. Bennamoun, and R.A. Owens, "A Novel Algorithm for Automatic 3D Model-Based Free-Form Object Recognition," *Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics,* vol. 7, pp. 6348-6353, 2004.
[32] A.S. Mian, M. Bennamoun, and R.A. Owens, "3D Recognition and Segmentation of Objects in Cluttered Scenes," *Proc. IEEE Workshop Applications of Computer Vision,* vol. 1, pp. 8-13, 2005.
[33] A.S. Mian, M. Bennamoun, and R.A. Owens, "Automatic Correspondence for 3D Modeling: An Extensive Review," *Int'l J. Shape Modeling,* 2005.
[34] A.S. Mian, M. Bennamoun, and R.A. Owens, "A Novel Representation and Feature Matching Algorithm for Automatic Pairwise Registration of Range Images," *Int'l J. Computer Vision,* vol. 66, no. 1, pp. 19-40, 2006.
[35] T. Oishi, R. Sagawa, A. Nakazawa, R. Kurazume, and K. Ikeuchi, "Parallel Alignment of a Large Number of Range Images," *Proc. Int'l Conf. 3-D Digital Imaging and Modeling,* pp. 195-202, 2003.
[36] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm," *Proc. Int'l Conf. 3-D Digital Imaging and Modeling,* pp. 145-152, 2001.
[37] Y. Shan, B. Matei, H.S. Sawhney, R. Kumar, D. Huber, and M. Hebert, "Linear Model Hashing and Batch RANSAC for Rapid and Accurate Object Recognition," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 121-128, 2004.
[38] "Software Packages," Stanford Computer Graphics Laboratory, http://graphics.stanford.edu/software/, 2005.
[39] F. Stein and G. Medioni, "Structural Indexing: Efficient 3-D Object Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 2, pp. 125-145, Feb. 1992.
[40] J. Wand and F.S. Cohen, "Part II: 3-D Object Recognition and Shape Estimation from Image Contours Using B-Splines, Shape Invariant Matching, and Neural Network," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 1, pp. 13-23, Jan. 1994.
[41] J. Williams and M. Bennamoun, "Simultaneous Registration of Multiple Corresponding Point Sets," *Computer Vision and Understanding,* vol. 81, no. 1, pp. 117-142, 2001.
[42] J.V. Wyngaerd, L.V. Gool, R. Koth, and M. Proesmans, "Invariant-Based Registration of Surface Patches," *Proc. IEEE Int'l Conf. Computer Vision,* vol. 1, pp. 301-306, 1999.

Ajmal S. Mian received the BE degree in avionics from the College of Aeronautical Engineering, NED University, Pakistan in 1993. He worked on a number of engineering and R&D projects related to radar data processing, communication jamming, and antijamming techniques before he was nominated for a masters degree. He received the MS degree in information security from the National University of Sciences and Technology, Pakistan, in 2003 and was awarded a PhD scholarship. He received the PhD degree in computer science from The University of Western Australia in 2006. He is currently a research fellow at the School of Computer Science and Software Engineering at The University of Western Australia. His research interests include computer vision, pattern recognition, multimodal biometrics, and information security.

Mohammed Bennamoun received the MSc degree from Queen's University, Kingston, Canada, in the area of control theory, and the PhD degree from Queen's/QUT in Brisbane, Australia, in the area of computer vision. He lectured in robotics at Queen's, and then joined QUT in 1993 as an associate lecturer. He then became a lecturer in 1996 and a senior lecturer in 1998 at QUT. In January 2003, he joined the School of Computer Science and Software Engineering at The University of Western Australia as an associate professor. He was also the director of a research center from 1998-2002. He is the coauthor of the book *Object Recognition: Fundamentals and Case Studies* (Springer-Verlag, 2001). He has published more than 100 journal and conference publications. He served as a guest editor for a couple of special issues in international journals, such as the *International Journal of Pattern Recognition* and *Artificial Intelligence*. He was selected to give conference tutorials at the European Conference on Computer Vision 2002 and the International Conference on Acoustics Speech and Signal Processing (ICASSP) in 2003. He organized several special sessions for conferences; the latest was for the IEEE International Conference in Image Processing (ICIP) held in Singapore in 2004. He also contributed in the organization of many local and international conferences. His areas of interest include control theory, robotics, obstacle avoidance, object recognition, artificial neural networks, signal/image processing, and computer vision.

Robyn Owens received the BSc (Hons) degree in mathematics from the University of Western Australia (UWA) in 1974 before going to Oxford University to complete the MSc degree (1976) and the PhD degree (1980), also in mathematics. She spent three years in Paris at l'Université de Paris-Sud, Orsay, continuing research in mathematical analysis before returning to UWA in 1982 to work as a research mathematician on the Automated Sheep Shearing Project. Since then, she has lectured in the Mathematics Department and, in 1986, joined the Department of Computer Science and Software Engineering after a six month visiting lectureship at the University of California, Berkeley. She is currently the dean of Graduate Studies at UWA. Her recent work has been on feature detection in visual images and shape measurement and representation.