Merging Multiple Views Using a Spherical Representation

K. Higuchi, H.Delingette, M. Hebert, K. Ikeuchi

The Robotics Institute
Carnegie Mellon University
Pittsburgh PA 15213

Abstract

*This paper proposes a new method for building a 3-D model from a set of range images. The method can merge data of **free-form** surfaces obtained from **arbitrary** viewing directions, with no prior knowledge of the poses. Our approach is based on matching the spherical representations of an object between the views. To obtain the spherical representation, we deform a discrete mesh to fit the object surface. A variation of the Gaussian curvature metric, which we call simplex angle, is computed at each node on the deformed mesh and mapped to a coordinate on the unit sphere. The transformation of the objects is computed by comparing the simplex angle measure at each node on the unit sphere. The transformation which produces the minimum errors is selected as the best match. We have implemented this method, applied the method to range images of objects from arbitrary viewpoints, and demonstrated the applicability for modeling from observation.*

1 Introduction

Most computer vision and graphics systems require three-dimensional models. Almost all the object recognition systems are designed as model-based systems. All the computer graphics systems generate appearances of an object from a three-dimensional model. For these applications, building accurate geometric models of objects, is a central issue.

Traditionally, a human operator manually represents objects using a CAD system. Conventional CAD systems, such as PADL or Vantage, represent a complicated object as a collection of much simpler primitive objects with set operations among them represented using a Constructive Solid Geometry (CSG) tree. Typically, CSG trees are built manually and are converted to boundary representations. The more complicated the object is, the more time and effort is necessary to build proper CSG trees. Worse, although this method can represent man-made objects efficiently, it cannot represent natural objects consisting of free-form surfaces.

Several attempts have been made to develop methods for building object models from observation. Martin and Aggawal [9] proposed a method to merge multiple range data taken from multiple *known* viewing directions. They represent a object as a collection of occupied cells. For the sake of memory efficiency, they employ the oct-tree representations. Their method needs precise calibration to determine the viewing directions since it assumes that they are known exactly. Therefore, it cannot handle range data obtained from arbitrary viewing directions.

Parvin and Medioni [10] segment range data into regions and represent one view as a graph of visible regions. By matching two graphs from two arbitrary viewing directions, they determined the transformation between the graphs. By using this obtained transformation, they merged range data from arbitrary multiple viewing directions. Since this method requires stable segmentation results, it is only suitable to handle *man-made* objects which consist of well-defined edges and planar faces; it is not appropriate to handle natural objects which consist of free form surfaces and no clear region boundaries. Other techniques, such as Kamgar-Parsi's [7] avoid the need for real geometrical features by defining virtual features from, for example, the iso-range contours of the object. Another example is Stein's approach [12] in which the virtual features are groups of surface normals around points on the surface. These virtual features are matched between two views of the object and the resulting correspondence is used for computing the best transformation between views.

Other techniques eliminate feature matching by formulating the registration problem as a non-linear minimization problem in which the objective function is the sum of the distances between the data points in one view and the transformed data points from the other view. For example, Champleboux [2] uses the Levenberg-Marquart algorithm to perform the minimization. As is the case with any minimization approach, this type of approach requires an initial estimate that is relatively close to the true transformation and it is not guaranteed to converge to a global minimum in general.

Besl [1] proposed a method for matching between *free-form* surfaces. Given a set of input points and a surface, his algorithm projects each point on the surface and computes the transformation that brings the input points as close as possible to their corresponding projections. These steps are repeated until the transformation between input points and surface converges. A similar approach was suggested by Chen and Medioni [3] and by Zhang [13]. Besl's approach has the advantage that it can handle arbitrary surface representations, including point sets, and that it does not require extracting features or establishing correspondences between features. However, because it is an iterative algorithm, it is sensitive to the transformation used for initializing the iter-

0-8186-5310-8/94 $03.00 © 1994 IEEE

ations. If the transformation is significantly far from the true value, the algorithm may converge to a local minimum which is far from the real transformation. Furthermore, due to its iterative nature, it is never guaranteed to converge to the best transformation.

Our goal is to eliminate two major limitations of these techniques. First, we want to compute accurately the transformations between views of an object even in the absence of any prior estimate of the transformations. We want to compute the transformations in a deterministic way so that we are guaranteed to find the best transformations. Second, we want to eliminate the requirements for feature extraction or feature matching, both of which typically limit the class of shapes that can be modeled. To this end, we have developed a surface representation which is translation-invariant so that a simple algorithm can be used to compute the rotation between views. The representation is computed from range data directly without any feature extraction or analytic surface fitting. Specifically, we propose a novel method to merge multiple free-form surfaces taken from unknown viewing directions. The method is based on an intrinsic representation of 3-D surfaces, the Simplex Angle Image (SAI). The SAI is a descendant of the Extended Gaussian Image (EGI) and Complex Extended Gaussian Image (CEGI) [6][8]. The common characteristic of representations in this family is that they all achieve translation invariance by storing Gaussian curvatures on the unit sphere. The SAI representation differs from its ancestors in that it preserves the relative spatial relations among surface patches in the representation while EGI and CEGI do not. Because of this property, an SAI constitutes a unique representation of a non-convex object. Another consequence is that it can be used for representing partial views of objects, whereas the other spherical representations can be used only for complete objects in general.

Our approach to building SAIs from range data and to merging them to form complete object models is as follows. We first deform a semi-regularly tessellated geodesic dome onto an object surface while maintaining the local regularity constraint. Then, at each vertex of the tessellation, we calculate Gaussian curvature. The resulting distribution of Gaussian curvature is mapped on the unit sphere. The resulting curvature distribution on the sphere is referred to as SAI. By comparing the two distributions computed from two different views of an object, we determine the rotation between two SAIs. From this rotation, we recover the transformation between the two viewing directions, and merge the data using the obtained transformation. The final step is to generate a surface model for the complete object.

Section 2 briefly explains the concept of SAI, and discusses the method for handling partial surface representations on SAI. The algorithms for constructing surface representations from a single range image are described in Section 3. The algorithm for matching SAIs obtained from different views is described in Section 4. Some of the experimental results are shown in Section 5. For reasons of space, we discuss only the application of the SAI representation to model building; see [4][5] for a more complete presentation of deformable meshes and of the spherical mapping and its application to object recognition.

2 Symplex Angle Images

In this section, we briefly introduce the concept of an SAI. First, we explain how to tessellate an arbitrary surface into a semi-regular mesh, homeomorphic to a tessellated sphere, and how to calculate a simplex angle, a variation of Gaussian curvature. Finally, we discuss how to handle partial views.

2.1 Semi-Regular Tessellation

A natural discrete representation of a surface is a graph of points, or tessellation, such that each node is connected to each of its closest neighbor by an arc of the graph. It is desirable for many algorithms to have a constant number of neighbors at each node. We use a class of tessellations such that each node has exactly three neighbors. Such a tessellation can be constructed as the dual of a triangulation of the surface.

Let us first consider tessellations of the unit sphere. A regular tessellation would be a tessellation covering a complete spherical surface such that the distance between vertices is constant and that each node has exactly three neighbors. It is well known that only approximate global regularity can be achieved. Specifically, the approach that we use is to first build a triangulation by subdividing each triangular face of a 20-face icosahedron into N^2 smaller triangles. The final tessellation is built by taking the dual of the $(20N^2)$ faces triangulation, yielding a tessellation with the same number of nodes. This tessellation of a sphere, a geodesic dome, is the starting point of our technique.

In order to obtain a mesh of a complete object surface, we deform a tessellated surface until it is as close as possible to the object surface while maintaining the following local regularity constraint: Let P be a node of the tessellation, P_1, P_2, P_3 be its three neighbors, G be the centroid of the three points, and Q be the projection of P on the plane defined by P_1, P_2, and P_3 (Figure 1). The local regularity condition simply states that Q coincides with G. The local regularity condition is invariant by rotation, translation, and scaling because it is purely local and involves only relative positions of the nodes with respect to each other, not absolute distances.

This local regularity constraint is the generalization to three dimensions of the regularity condition on two dimensional discrete curves which simply states that all segments are of equal lengths.

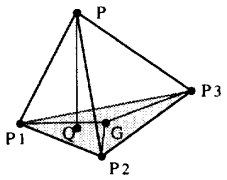

Figure 1: Local Regularity

2.2 Discrete Curvature Measure: Simplex Angle

The last step in building a discrete surface representation is to define a measure of curvature that can be computed from a tessellation with the appropriate regularity properties.

We propose a definition in terms of angular variation between neighbors in the tessellation. Let P be a node of the tessellation, P_1, P_2, P_3 its three neighbors, O the center of the sphere circumscribed to the tetrahedron (P_1, P_2, P_3), Z the line passing through O and through the center of the circle circumscribed to (P_1, P_2, P_3). Now, let us consider the cross section of the surface by the plane P containing Z and P. The intersection of P with the tetrahedron is a triangle. One vertex of the triangle is P, and the base opposite P is in the plane (P_1, P_2, P_3) (Figure 2). We define the angle φ as the angle between the two edges of the triangle intersecting at P. By definition, φ is the discrete curvature measure at node P. We call φ the simplex angle at P to emphasize the analogy with the angle between consecutive segments that is commonly used for representing 2-D curves.

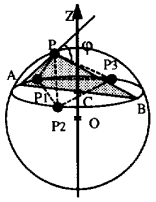

Figure 2: Definition of the Simplex Angle

The simplex angle varies between $-\pi$ and π. The angle is 0 for a flat surface, and is large in absolute value if P is far from the plane of its three neighbors. The simplex angle is negative if the surface is locally concave, positive if it is convex, assuming that the set of neighbors is oriented such that the normal to the plane they form is pointing toward the outside of the object. This behavior of the simplex angle corresponds to the intuitive notion of local "curvature" of a surface. Another desirable property is that the simplex angle is sphere-invariant in that φ remains the same no matter where P is located on the circumscribed sphere. In particular, this implies that if the nodes of the tessellation are on a surface whose curvature is constant in a region, then the simplex angle will also be constant in this region no matter what the distribution of the points is. Finally, it is clear that the simplex angle is invariant by rotation, translation, and scaling. In the remainder of the paper, we will denote by g the function that maps a node to its simplex angle; the simplex angle φ at a node P will be denoted by $g(P)$.

It is important to note that other definitions of the quantity $g(P)$ are possible as long as the definition guarantees that $g(P)$ is invariant by rigid transformations and by scaling. In particular, any invariant curvature measure could be used instead of our simplex angle. We selected this definition because it is easy to compute and it is reasonably stable with respect to small variation of the surface (see [5] for a more complete discussion of alternate definition of φ).

2.3 Spherical Mapping

In this section, we define a canonical mapping between a surface tessellation and a standard tessellation of the unit sphere. Let M be a tessellation of points on a surface such that it has the topology introduced in Section 2.1. Let S be a reference tessellation with the same number of nodes on the unit sphere. We can establish a one-to-one mapping h between the nodes of M and the nodes of S. The mapping h depends only on the topology of the tessellation and the number of nodes. Specifically, for a given size of the tessellation, $K = 20N^2$, where N is the frequency of the tessellation (Section 2.1), we can define a canonical numbering of the nodes that represents the topology of any K-tessellation. In other words, if two nodes from two different K-tessellation have the same index, so do their neighbors. With this indexing system, $h(P)$, where P is a node of the spherical tessellation, is the node of the object tessellation that has the same index as P.

Given h, we can store at each node P of S the simplex angle of the corresponding node on the surface $g(h(P))$. The resulting structure is a tessellation on the unit sphere, each node being associated with a value corresponding to the simplex angle of a point on the original surface. By analogy with the EGI, we call this representation the Simplex Angle Image (SAI). In the remainder of the paper, we will denote by $g(P)$ instead of $g(h(P))$ the simplex angle associated with the object tessellation node $h(P)$ since there is no ambiguity.

If the original tessellation M satisfies the local regularity constraint, then the corresponding SAI has several invariance properties. First, given K, the SAI is invariant by translation and scaling of the original object. This condition is satisfied because the simplex angle itself is invariant by translation and scaling (Section 2.3), and because M still satisfies the local regularity condition after translation and scaling (Section 2.1).

The fundamental property of the SAI is that it represents an object unambiguously up to a rotation. More precisely, if M and M' are two tessellations on the same object with the same number of nodes, then the corresponding SAIs S and

S' are identical up to a rotation of the unit sphere. Strictly speaking, this is true only as the number of nodes becomes very large because the nodes of one sphere do not necessarily coincide with the nodes of the rotation version of the other sphere. One consequence of this property is that two SAIs represent the same object if one is the rotated version of the other.

From this definition of the mapping h, we can now easily see the origin of the property of connectivity conservation mentioned in the Introduction. If two nodes P_1 and P_2 are connected on the spherical tessellation, then the two corresponding nodes $M_1 = h(P_1)$ and $M_2 = h(P_2)$ on the object tessellation are also connected by an arc of the object tessellation. The property holds because of the definition of h which depends only on the topology of the tessellation, not on the positions of the nodes.

Another way to look at these properties of SAIs is in terms of unicity of representation. A given SAI defines a tessellation size and a distribution of simplex angles. The unicity property states that an SAI represents a unique object tessellation up to rotation, translation, and scale. The unicity property holds even in the case of arbitrary non-convex objects because of the connectivity conservation property.

2.4 SAI for Partial Views

In the previous section, we have discussed a technique to represent a complete surface of an object. In order to merge multiple views, we need to address now the problem of representing partial views of an object.

We can distinguish between regions of the tessellation that are close to input data points, and parts that are interpolated between input data. The first type of region is the visible part of the tessellation, and the second type is the hidden part. The number of vertices of the spherical model corresponding to the visible area varies depending upon how the hidden area is interpolated. In order to compare SAIs computed from different views, we need to adjust the number of nodes because the relative sizes of the visible and hidden areas vary depending on the viewing direction. We perform this adjustment using the following procedure.

Let us consider the problem of merging two views, V_1 and V_2. Let S_1 and S_2 be the number of nodes that would be visible from V_1 and V_2 if we had a complete model of the object. Let the visible areas of the object surface be A_1 and A_2 for V_1 and V_2, respectively (Figure 3). The ratio of the number of visible SAI nodes to the total number of SAI nodes, S_0 is equal to the ratio of the visible area to the entire object area, A_0.

$$\frac{S_1}{S_0} = \frac{A_1}{A_0} \qquad \frac{S_2}{S_0} = \frac{A_2}{A_0}$$

However, we do not know A_0 since we have only partial views of the object, but we can estimate A_1 and A_2 from each of the views. Eliminating S_0 from these equations, we obtain:

$$S_2 = S_1 \frac{A_2}{A_1} \tag{1}$$

This equation enables us to modify the SAIs from different views so that the distribution of nodes in the visible area is consistent between views. More precisely, we compute the scale factor A_2/A_1 from the estimated visible areas from each of the images, and move the nodes of the SAI from V_2 so that Eq.(1) is satisfied.

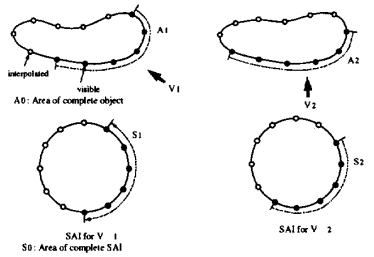

Figure 3: SAI Scaling Algorithm

The key in this procedure is the connectivity conservation property of the SAI. Specifically, if a connected patch of the surface is visible, then its corresponding image on the SAI is also a connected patch on the sphere. This property allows us to bring the two connected patches into correspondence using a simple scaling.

3 Extracting the SAI from 3-D Data

In the previous sections, we have defined the basic concept of SAI. In this section, we describe the detailed algorithm for computing such a mesh from an input object.

The general approach is to first define an initial mesh near the object and to slowly deform it by moving its nodes until the mesh satisfies two conditions: It must be close to the input object, and it must satisfy the local regularity condition. The first condition ensures that the resulting mesh is a good approximation of the object, while the second condition ensures that a valid SAI can be derived from the mesh. Section 3.1 describes the basic algorithm for deforming the mesh. Section 3.2 describes the algorithm for converting the final mesh to a spherical representation.

3.1 Mesh Deformation

The problem now is to deform the mesh such that all the nodes satisfy two fundamental properties:

- Mesh nodes must be as close as possible to the original surface.
- Mesh nodes must satisfy the local regularity constraints: a node is on the line parallel to the normal vector of the plane formed by its three neighbors and passing by the center of the neighbors.

These two conditions ensure that the mesh is a good approximation of the surface while guaranteeing that it is an intrinsic representation. The formalism of deformable surfaces [4] is applied to deform the mesh until it satisfies these criteria. Specifically, each node is subject to two types of forces. The first type of forces brings a node closer to the input surface, while the second type forces the node to satisfy the local regularity constraint. Let F_o be the force of the first type applied at a given node N, and F_g be the force of the second type at the same node. Node P is iteratively moved according to those forces. If P_{t+1}, P_t, and P_{t-1} are the positions of node P at three consecutive iterations, the update rule is defined as:

$$P_{t+1} = P_t + F_o + F_g + D(P_t - P_{t-1}) \quad (2)$$

This expression is simply the discrete version of the fundamental equation describing a mechanical system subject to two forces and to a damping coefficient D. D must be between 0 and 1 to ensure convergence. As long as it is within these bounds, D affects only the rate of convergence. A typical value is $D = 0.5$. Theoretically, the combination of forces brings the mesh to a state such that $F_o \approx 0$ and $F_g \approx 0$. In practice, the iterative update of the mesh is halted when the relative displacements of the nodes from one iteration to the next are small enough.

F_o is defined by calculating the point P_c from the original surface that is closest to the node, that is:

$$F_o = kPP_c \quad (3)$$

where k is the spring constant of the force which must be between 0 and 1. The effect of the force is negligible if the node is already very close to the surface. Conversely, the force pulls nodes that are far from the surface, the strength of the force increasing with distance. When the points are far away, it is desirable to limit the strength of the force to avoid unstable situations in which a node would move toward the surface too quickly and overshoot the optimal position by a large distance. In practice, k varies between 0.01 at the beginning of the iterations to 0.4 at the end of the iterations, that is, when the nodes of the mesh have reached a stable position.

The curvature force F_g is calculated by computing the point P_g that is on the line normal to the triangle formed by the three neighbors of P and containing G, and such that the mesh curvature at P and P_g are the same: $g(P_g) = g(P)$. Those two conditions ensure that P_g satisfies the local regularity condition while keeping the original mesh curvature. F_g is defined as a spring force proportional to the distance between P and P_g:

$$F_g = aPP_g \quad (4)$$

In practice, other terms must be added to Eq.(2) in order to guarantee smoothness of the resulting mesh; seeSection [4] for a complete description of the smoothness constraints.

3.2 From Surface Tessellation to SAI

Once a locally regular tessellation is created from the input data, a reference tessellation with the same number of nodes is created on the unit sphere. The value of the angle at each node of the surface is stored in the corresponding node of the sphere.

Figure 4(a) and (b) show the intensity image of a partial view of a bust, and the corresponding range image. In this example, we use the dual of the 9th subdivision of a 20-face icosahedron, (1620 faces) as the initial mesh as shown in Figure 5(a). This initial mesh is deformed using Eq.(2) and reaches the stable state shown in Figure 5(b). The corresponding SAI data is shown in Figure 5(c). In the SAI display, the distance from each vertex to the origin is proportional to the simplex angle.

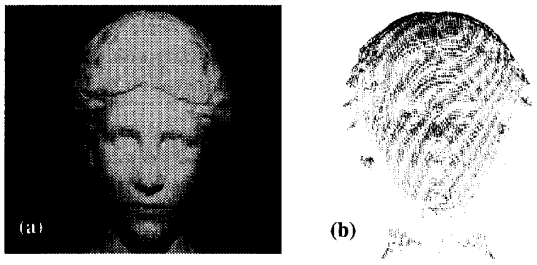

Figure 4: Input data; (a) Intensity image; (b) Range data

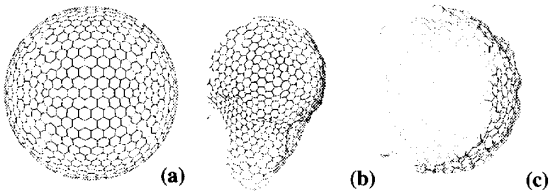

Figure 5: (a) Initial mesh; (b) Deformed mesh; (c) SAI represented on the unit sphere.

4 Matching Multiple Views

We now address the matching problem: Given two SAIs, determine the rotation between them, and then find the rigid transformation between the two original sets of points. The representations of a single object with respect to two different viewing directions are related by a rotation of the underlying sphere. Therefore, the most straightforward approach is to compute a distance measure between two SAIs. Once the rotation yielding minimum distance is determined, the full 3-D transformation can be determined.

4.1 Finding the Best Rotation

In the following discussion, we will consider only the vertices of the SAIs that correspond to visible parts of the surface. Let S and S' be the SAIs of two views. Denoting by $g(P)$ and $g'(P)$, the value of the simplex angle at a node P of S and S', respectively. S and S' are representations of the same area of the object if there exists a rotation R such that:

$$g'(P) = g(RP) \tag{5}$$

For every point P of S'. Since the SAI is discrete, $g(RP)$ is not defined because in general RP will fall between nodes of S'. We define a discrete approximation of $g(RP)$, $G(RP)$, by interpolating the values of g at the four nodes of S' nearest to RP, P_1 to P_4. Formally, $G(RP)$ is a weighted sum of $g(P_i)$, the *ith* weight being a function of the distance between P_i and RP.

The problem now is to find this rotation using the discrete representation of S and S'. This is done by defining a distance $D(S, S', R)$ between SAIs as the sum of squared differences between the simplex angles at the nodes of one of the sphere and at the nodes of the rotated sphere. Formally, the distance is defined as:

$$D(S, S', R) = \sum_{S} (g'(P) - G(RP))^2 \tag{6}$$

The minimum of D corresponds to the best rotation that brings S and S' in correspondence. The simplest strategy is to sample the space of all possible rotations, represented by three angles (θ, ϕ, ψ), and to evaluate D for every sample value $(\theta_i, \phi_i, \psi_i)$. In practice, it would be too expensive to compute the distance for all possible rotations. Two approaches may be used to reduce the search space. First, additional knowledge, such as the rotation between the principal axis of the two views, can be used to reduce the size of the rotation space. Second, a coarse-to-fine approach can be used to locate the minimum by performing first a search for potential minima using a coarse discretization of the (θ, ϕ, ψ) space, and then searching only around these potential minima using a higher resolution. This approach has been shown to be very effective in pruning the search space [5].

Figure 6 shows the result of matching two views of a head. Figure 6(a) and (b) show the intensity images of the two views of the object, and the corresponding SAIs. Figure 6(c) shows the graph of the distribution of D as a function of two of the rotation angles, ψ and θ. The graph exhibits a sharp minimum corresponding to the best rotation between the two spherical maps.

It is important to note that the rotation in SAIs is not the rotation between the original objects; it is the rotation of the spherical representations. An additional step is needed to compute the actual transformation between objects as described below.

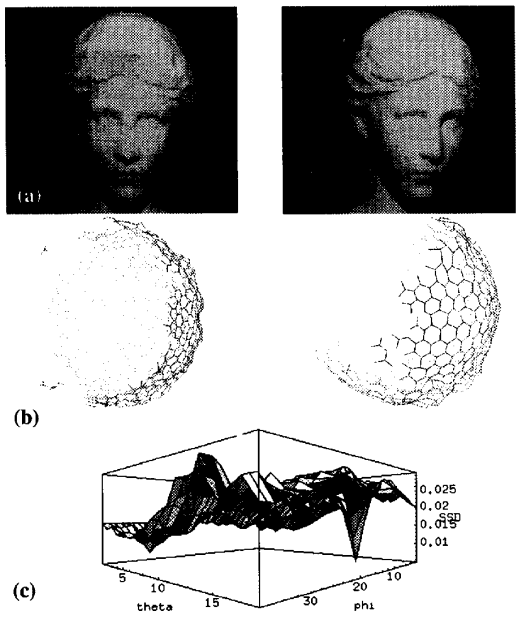

Figure 6: Matching two SAIs: (a) Intensity images; (b) SAIs; (c) Graph of the distance

4.2 Computing the Full Transformation

The last step in matching objects is to derive the transformation between the actual objects, given the rotation between their SAIs. The rotational part of the transformation is denoted by R_0, the translational part by T_0. Given a SAI rotation R, for each node P' of S' we compute the node P of S that is nearest to RP'. Let M, resp. M', be the point on the object corresponding to the node P of S, P'. A first estimate of the transformation is computed by minimizing the sum of the distances between the points M of the first object and the corresponding points $R_0M' + T_0$ of the second object. Formally, the expression to minimize is:

$$E_0(R_0, T_0) = \sum \| R_0M' - (T_0 - M) \|^2 \tag{7}$$

The sum in this expression is taken over the set of all the nodes of the mesh. The transformation derived by minimizing Eq.(7) is only an approximation because it assumes that the nodes from the two meshes correspond exactly. We use an additional step to refine the transformation by looking for the node M closest to M' for every node of the mesh and by computing again the minimum of Eq.(7).

Figure 7 shows the final result of computing the transformation between the two views of Figure 6. Figure 7(a) shows the superimposition of the data points from the two range images before computing the transformation. Figure 7(b) shows the same combined data set using the transformation computed using the algorithm above. This display shows

that the two views are registered correctly. In this experiment, no prior knowledge of the transformation was used.

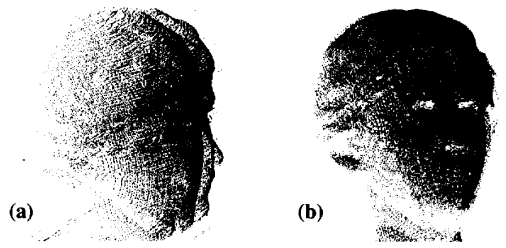

Figure 7: Merging two views; (a) Overlaid views before registration; (b) Overlaid views after registration.

5 Building a Complete Model

Figure 8 shows an example of model building from multiple views. Note that the transformations between views are unknown. In this experiment, 3-D range data is obtained using a commercial light-stripe range-finder [11]. Both intensity image and 3-D range image are able to be obtained simultaneously from this range-finder.

Figure 8(a) shows the intensity images of a human hand from three different arbitrary views. In this example, we use only the region of the fist. We discard the wrist region, because the subject cannot maintain the same wrist angle while image shooting.

Figure 8(b) and (c) show the tessellation of the visible part of the hand and the corresponding SAI for each view. Surface model and SAI are built using the approach described in Section 3.1 and Section 2, respectively. As the initial mesh, we use the dual of the 7th subdivided icosahedron containing 980 faces. In each image, only about 30% of the object is visible; the remaining 70% of the representation is interpolated and is ignored in the matching between the SAIs. In addition, while these images are the partial images of the complete object, we adjust the tessellations using the techniques described in Section 2.4. We then compute the correspondence of the SAIs between two pairs of views using the algorithms of Section 4.1. Finally, we compute the transformation of the rigid objects from the rotation between their SAIs.

Figure 8(d) shows the result of pairwise image registration. Each of the two displays in Figure 8(d) shows a 3-D view of the set of data points obtained by combining the points from two views using the computed transformation. The errors of the registration of these models are 0.86 and 0.97 mm RMS distance, respectively. Figure 8(e) shows several views of the set of points obtained by combining the points from all three images using the transformations computed by the SAI matching algorithm. Finally, Figure 8(f) shows the complete surface model obtained by applying the deformable surface algorithm [4] to the entire data set. This final representation was built using the 980-node tessellation. Higher resolution surface models can be obtained by increasing the number of nodes of the reference tessellation.

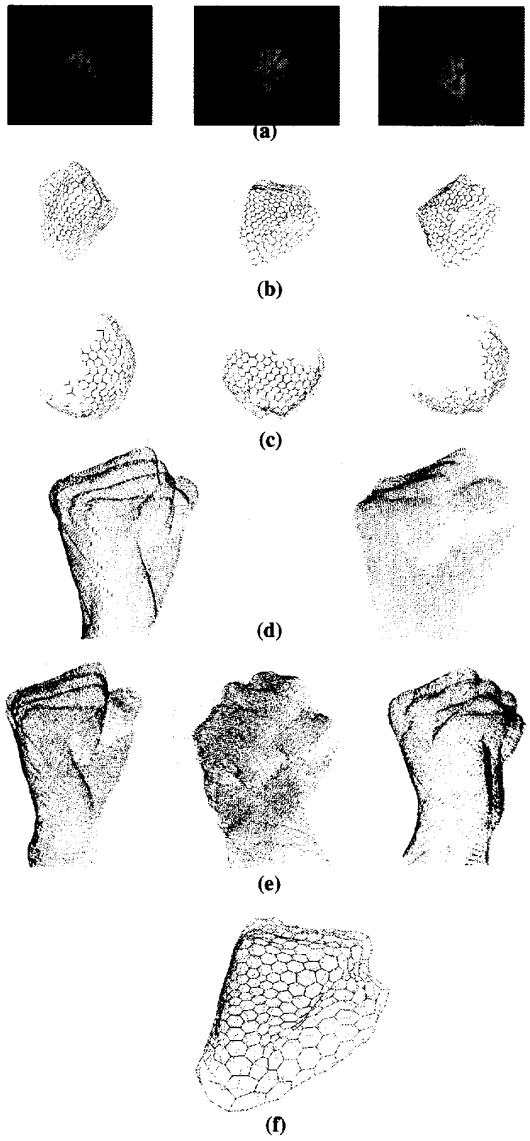

Figure 8: Building a complete model of a human hand; (a) Intensity images; (b) Deformed mesh; (c) SAIs; (d) Data points after pairwise registration; (e) Data points after full registration; (f) Complete model.

This experiment highlights some of the characteristics of the SAI matching approach. First, the viewpoints are arbitrary in that the transformations between them are not restricted to a single-axis rotation as is often the case in modeling systems. Furthermore, the transformations between the view-

points are not known a priori but are recovered accurately by the algorithms. Second, the matching algorithm does not require any feature extraction or feature matching. This is an important feature that enables us to handle arbitrary curved objects for which reliable feature extraction is difficult. Finally, the deformable surface algorithm used for building the initial tessellation of the surface has the property that it smoothes out most of the noise in the data while retaining the major features of the surface (see [4] for more details on this aspect of the algorithm). This property results in an accurate pose estimation even though the initial set of data points might be noisy.

6 Conclusion

This paper introduced a new approach for building models of curved objects from multiple arbitrary views. The basic representation is a mesh of nodes on the surface that satisfies certain regularity constraints. We introduced the notion of simplex angle as a curvature indicator stored at each node of the mesh. We showed how a mesh can be mapped into a spherical representation in canonical manner, and how object models can be generated by merging multiple views by computing the transformations among the views.

This approach eliminates two major limitations of conventional model building systems. First, by using a translation-invariant representation, the SAI, it enables us to decouple the determination of rotation and translation, and it makes the determination of rotation a simple problem which involves straightforward correlation of spherical images. This feature of the approach enables us to deal with arbitrary transformations between views and to deal with the worst case in which the transformation between views is not known at all. Most other approaches require some knowledge of the transformations between viewpoints or impose limitations on the allowable locations of the viewpoints. Even point-set matching techniques recently introduced [1] are minimization techniques that are not guaranteed to find the best transformation between views. The SAI matching is guaranteed to find the best rotation and translation even if no initial estimate is available. Second, it does not require any feature extraction or feature matching. This enables us to eliminate another limitation of model building systems, that is, the requirement that the object shape must be simple enough to be described by a set of simple geometric features. By contrast, the SAI matching approach can handle general curved surfaces.

We are working on extending the basic concept of SAI in order to achieve more robust matching and pose computation. In this paper, we have presented the SAI as a way to store curvature information in a translation-invariant manner while retaining the connectivity of the original surface. However, the concept of SAI is more general because other pieces of information may be stored at each node of the spherical image. For example, quantities computed from the intensity image, such as albedo or texture, can be stored at each node of the sphere. This information can be computed easily since range finders usually provide intensity data in addition to position at every data point. This appearance information can then be used to augment the definition of the distance between SAIs by adding a term that measures the difference of intensity values. Adding appearance information would make the approach more effective by providing a more discriminating measure of distance between shapes. In particular, the current matching algorithm cannot discriminate well between surfaces that are very smooth. In this case, the intensity distribution on the surface should be used since geometric information is limited.

Acknowledgments

This work was sponsored in part by National Science Foundation under IRI-9224521 and in part by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson, AFB, OH 45433-6543 under F33615-90-C-1465 and F33615-93-1-1282.

References

[1] Besl, P., Kay, N.D., A Method for Registration of 3-D Shapes, PAMI-14(2), 1992.

[2] Champleboux, G., Lavallee, S., Szeliski, R., Brunie, L., From Accurate Range Imaging Sensor Calibration to Accurate Model-Based 3-D Object Localization, Proc. CVPR-92, pp. 83-89, 1992.

[3] Chen, Y., and Medioni, G., Object Modelling by Registration of Multiple Range Images, Image and Vision Computing, 10(3), April 1992.

[4] Delingette, H., Hebert, M., Ikeuchi, K., Shape Representation and Image Segmentation Using Deformable Surfaces, Image and Vision Computing, 10(3), April 1992.

[5] Delingette, H., Hebert, M., Ikeuchi, K., A Spherical Representation for the Recognition of Curved Objects, ICCV'93, Berlin, 1993.

[6] Ikeuchi, K., Recognition of 3-D objects using the extended Gaussian image, IJCAI-81, pp. 595-600, 1981.

[7] Kamgar-Parsi, B., Jones, L.J., Rosenfeld, A., Registration of Muliple Overlapping Range Images: Scenes Without Distinctive Features, PAMI-13(9), 1991.

[8] Kang, S.B., Ikeuchi, K., Determining 3-D Object Pose Using the Complex Extended Gaussian Image, Proc.CVPR-91, 1991.

[9] Martin, W.N., Aggarwal, J.K., Volumetric Descriptions of Objects from Multiple Views, PAMI-5(2), 1983.

[10]Parvin, B., Medioni, G., B-rep from Unregistered Multiple Range Images, Proc. IEEE Robotics and Automation, 1992.

[11]Sato, K., Yamamoto, H., Inokuchi, S., Range Imaging System Utilizing Nematic Liquid Crystal Mask, Proc. ICCV-87, 1987.

[12]Stein, F., Medioni, G., TOSS-A System for Efficient Three Dimensional Object Recognition, Proc. DARPA Image Understanding Workshop, 1990.

[13]Zhang, Z., *Iterative Point Matching for Registration of Free-Form Curves*, Research Report 1658, INRIA, March 1992.