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

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

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

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

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

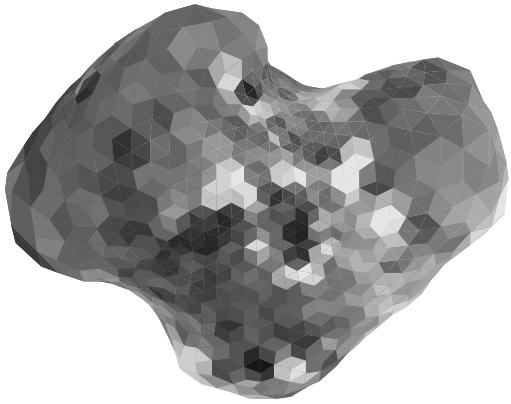

Figure 23: Error distribution on the final model built using views 1, 2, 7, and 10 of Figure 9; surface shading is proportional to the surface error.

5. Conclusion

We 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. The matching and registration algorithms may be used directly for building complete models, or they can be used front-end to more precise registration algorithms that use the range data itself but require accurate initial estimates of the viewing poses.

This approach eliminates two major limitations of conventional model building systems. First, it enables us to convert the matching problem to a straightforward correlation of spherical images. As a result, the approach is able to deal with arbitrary transformations between views and to operate without requiring an initial estimate of the transformation. Second, it does not require any feature extraction or feature matching. As a result, the SAI matching approach can handle general curved surfaces. We have used the SAI as a way to store curvature. 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, albedo or texture could be stored. This appearance information can be used to augment the definition of the distance between SAIs by adding additional terms Adding appearance information will make the approach more effective by providing a more discriminating measure of distance between shapes. Another research direction is in the determination of the best sequence of views to be used for a particular model. In the examples presented in this paper, the number of images was small enough and the overlap between them was large enough that it did not matter in which order the images are processed. In general, however, it is important to compare the pairs of images that are most likely to yield the most accurate matching results.

tions, (1,2), (2,7), (7,10), and (10,7) using the SAI matching algorithm. The transformations are combined in order to transform all the data points in a common reference frame, after which a new mesh is fit to the data. A mesh frequency of 7, corresponding to 980 nodes, is used both for matching and for building the final mesh. The SAIs used for the matching and the corresponding meshes are shown in Figure 11.

Figure 21(a) shows a 3-D view of the set of data points obtained by merging the four data sets using the transformation computed from the SAI matching. Figure 21(b) shows two views of the resulting mesh fit to the complete object model. It is important to note that, once the data sets are registered, other surface fitting algorithms could also be used. In that case the deformable surfaces and SAI matching are used as tools for registering curved surfaces. Also, the mesh is shown at frequency 7 in order to be consistent with the models of the individual views of Figure 11 but, in practice, a higher frequency should be used for finer resolution on the model.

Figure 22 shows the surface error distribution on the model of Figure 21. In this figure, the surface shading is proportional to the surface error. The maximum errors, shown as bright faces, occur at the center of the object. This is because it is the region of the object with the smallest density of data points in the four views used in this experiment.

Those two examples show that the SAI matching can successfully match and register partial views of 3-D objects, even in the case of relatively little curvature information (Figure 20) or in the case of a complex object with large concavities (Figure 21). As we mentioned earlier, the remaining problem is to automatically select the best views to use in the model-building or, alternatively, to integrate all the views at once. This is the subject of on-going research.

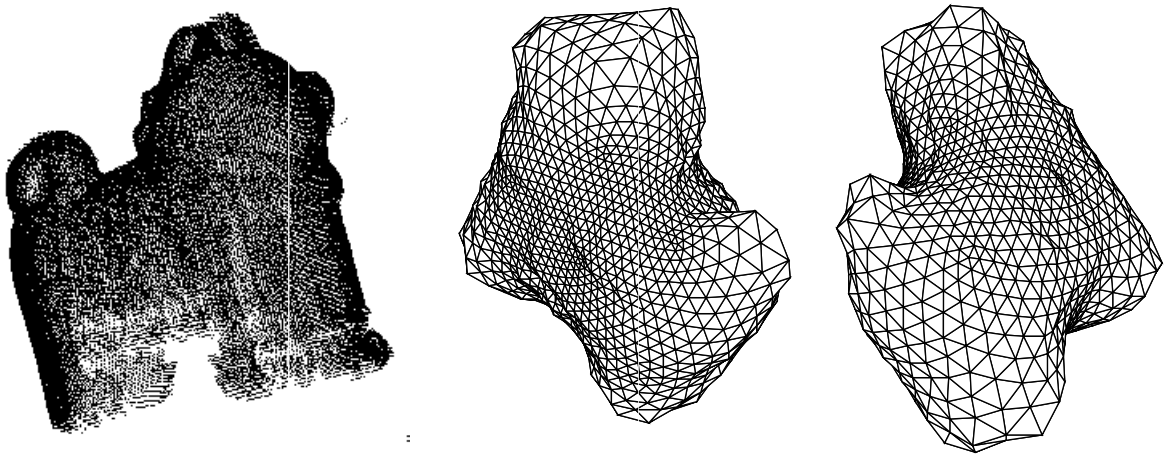

Figure 22: Final model built by combining views 1, 2, 7, and 10 of Figure 9: (a) Registered set of data points; (b) Two views of the final model mesh.

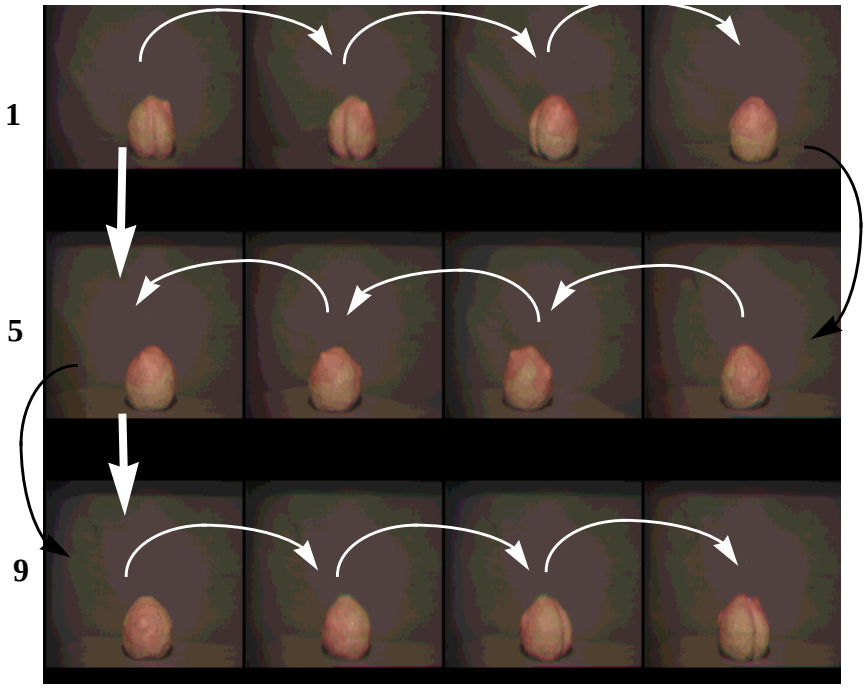

Figure 19: Twelve views of an object and computed poses.

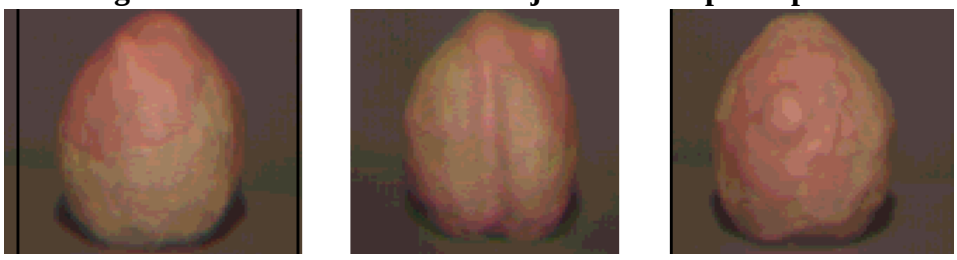

Figure 20: Three views with sufficient overlap.

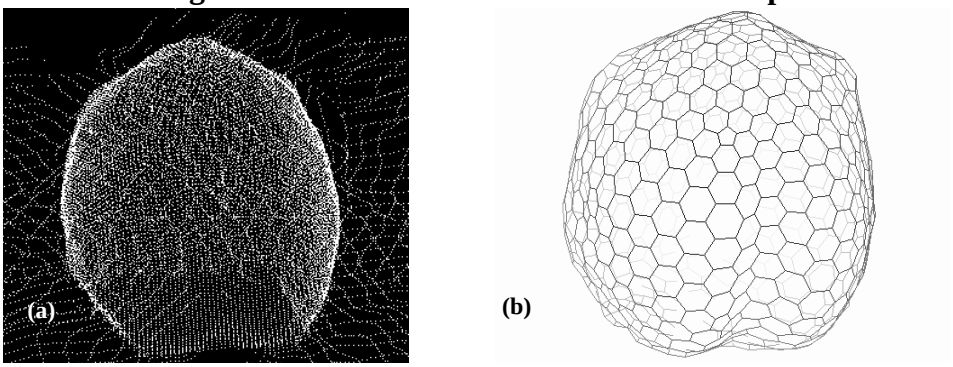

Figure 21: Complete 3-D model; (a) Combined set of data points from registered range data; (b) Surface model.

A similar example of model generation from multiple views is shown in Figure 21 and Figure 22. Figure 21 shows the result of merging views number 1, 2, 7, and 10 from the sequence shown in Figure 9. The views are matched by computing the pairwise transforma-

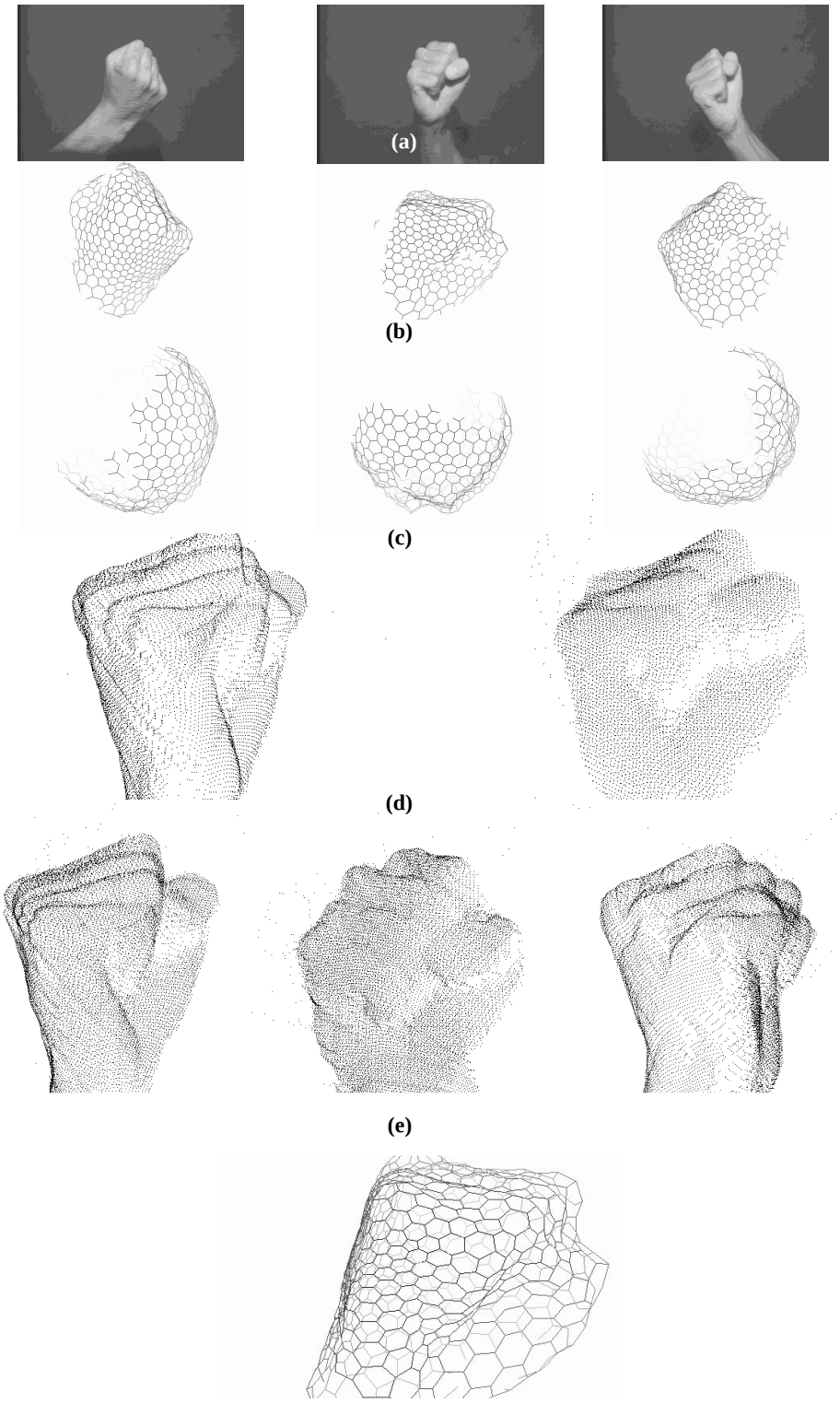

Figure 18: Building a complete model of a human hand; (a) Intensity images; (b) Deformed mesh; (c) SAIs; (d) Data points after pairwise registration; (e) Three views of the data points after full registration; (f) Complete model.

dual of the 7th subdivided icosahedron containing 980 faces as initial mesh. In each image, only about 30% of the object is visible; the remaining 70% of the representation is interpolated and is ignored in matching the SAIs.

Figure 17(d) shows the result of pairwise image registration. Each of the two displays shows a 3-D view of the set of data points obtained by combining the points from two views using the transformation computed from the matching. The errors of the registration of these models are 0.86 and 0.97 mm RMS distance, respectively. Figure 17(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 17(f) shows the complete surface model obtained by applying the deformable surface algorithm [4] to the entire data set.

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 viewpoints are not known a priori but are recovered accurately by the algorithms. Of course, in this approach the accuracy is limited by the resolution of the mesh. Increased accuracy can be achieved by going back to the original data points and using a more precise registration algorithm which uses the pose from the SAI matching as a starting point. Second, the matching algorithm does not require any feature extraction or feature matching. This is an important characteristic that enables us to handle arbitrary curved objects for which reliable feature extraction is difficult, such as the hand in Figure 17.

In the example of Figure 17, there is good surface overlap between all the views and there is no ambiguity as to which transformations should be used to generate the final model. Figure 18 shows a different situation that is typical of a model building application in which we have a larger number of image. From considerations of surface coverage, views 1,5 and 9 are sufficient to reconstruct the model (Figure 19). In fact, it would be preferable to use only those views instead of the 12 views to speed up reconstruction and to minimize errors. However, the transformations between these three views, indicated by the thick vertical white arrows, cannot be computed directly because there is very little overlap between the corresponding surfaces. Therefore, the only way to compute the transformations is to compute the intermediate transformations, indicated by the curved arrows, using SAI matching between consecutive views. These “elementary” transformations are compounded to form the two desired transformations. Data points from images 1,5, and 9 are converted to a common coordinate system as shown in Figure 20(a). A 980-node surface model is then computed by fitting a deformable surface as shown in Figure 20(b).

It is clear that we could have matched different views to recover the relative transformations and that this particular selection may not be optimal. Finding the optimal combination of views is still an open issue. What this example shows, however, is that the matching algorithm provides us with the basic tool for performing registration between surfaces in a general manner. It also shows that the individual transformations computed by the matching algorithm are accurate enough that they can be compounded over long sequences into transformations which accurate enough for building a complete model.

View number	Average area/node (mm^2)	Standard deviation of area/node (mm^2)	Percentage Variation (%)
1	13.33	0.25	1.9
2	13.76	0.31	2.3
3	14.16	0.23	1.6
4	17.77	0.41	2.3
5	16.64	0.32	1.9
6	16.16	0.28	1.7
7	17.05	0.25	1.5
8	14.49	0.27	1.9
9	15.50	0.22	1.4
10	10.32	0.21	2.0
11	13.45	0.29	2.2
12	13.18	0.27	2.0

Figure 17: Density of nodes in the meshes produced from the twelve views of Figure 9 expressed as the average surface area per mesh node and the absolute and relative variation of density over the mesh.

4. Building a Complete Model

We have described so far an algorithm for matching two pieces of the surface of an object computed from two unregistered range images. We now consider the case of merging multiple views into a single model. The basic approach is to match the surfaces from the images in a pairwise manner, to combine the transformations obtained from the matching into transformations between each image and a single model reference frame, to convert all the data points from all the range image into this common coordinate system, and to fit a deformable mesh to this data set in order to obtain a smooth surface model.

Figure 17 shows an example of model building from three views. In this experiment, 3-D range data is obtained using a commercial light-stripe range-finder [9] which can acquire registered range and intensity images. Figure 17(a) shows the intensity images of a human hand from three different arbitrary views. Figure 17(b) and (c) show the tessellation of the visible part of the hand and the corresponding SAI for each view, respectively. We use the

they are interpolated and not fit to the data. The errors were computed from 998 nodes out of a total of 1620 nodes in this example. The ratio of number useful of useful nodes to total number of nodes is lower in the case of the face because only partial views are used.

3.5.3. Mesh Density

A property of the mesh model that is critical for the matching to work at all is that the density of nodes on the surface be approximately uniform. We claimed that this uniform distribution is achieved by incorporating a constraint of local regularity in the deformable surface algorithm. In this section, we verify experimentally that the density is indeed approximately uniform.

Measuring the density of nodes is somewhat difficult because we use an approximation of the surface, the mesh, rather than the true surface. In order to estimate node density, we take advantage of the fact that each node of the mesh is associated with a unique triangular face because the mesh is always the dual of a triangulation of the surface. Consider the node marked ***P*** in the fragment of mesh shown in Figure 15: This node corresponds to the triangle ***T*** joining the centers of the three cells intersecting at ***P***. This triangle belongs to the triangulation dual of the mesh tessellation. The area of the triangle $a(\boldsymbol{T})$ is an estimate of the surface area associated with ***P***. Because node density may be defined equivalently as the average number of nodes per unit area or as the average surface area per node, the density can be approximated by the average area of the triangles evaluated over the mesh. Of course, this is only an approximation of the density but a large variation of node density using this approximate definition would still indicate that our assumption of uniform distribution would be violated.

Figure 16 shows the average area per node in mm^2 for the meshes built from each of the twelve views of Figure 9 and the variation of the area per node over all the nodes of each mesh. Since it is difficult to interpret the absolute value of the variation of area/node, a more meaningful measure of uniformity is the variation relative to the average value. The relative variations are shown as percentages in the third column of Figure 16. Those numbers are rounded-off to the first decimal. The meshes have frequency 7 (980 nodes) in this case, although only the nodes that correspond to actual, i.e., non-interpolated nodes, are taken into account in the calculation of the average area per node.

This experiment shows that the variation of area per node is on average 1.9% over any given mesh. This shows that our claim that the local regularity constraint does enforce uniform distribution of nodes on the surface is satisfied in practice.

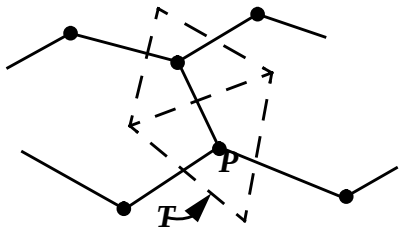

Figure 16: The triangle area associated with a mesh node.

mean error is on the order of 0.1mm which is also the maximum resolution of the range sensor. The standard deviation is on the order of 0.2mm, reflecting the fact that the error is distributed in a relatively uniform manner. The large maximum error is due to “border effects”. Specifically, a node at the edge of the visible part of the mesh may not overlap exactly with a region of the data set, thus causing a large error to be reported. This occurs only at a few isolated nodes at the border. Finally, the minimum error is very small, on the order of 0.01mm, but this is really meaningless because it occurs only at a very few isolated points and is the result of accidental alignment between mesh nodes and data points.

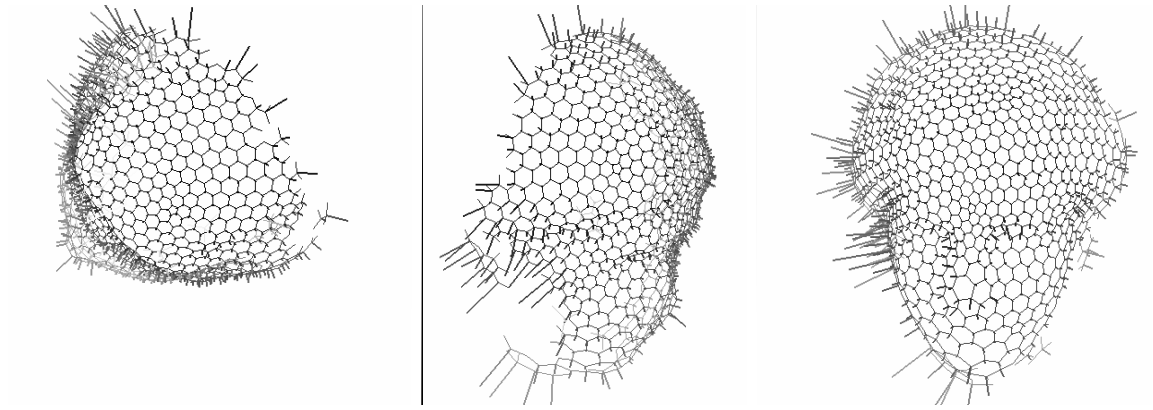

Figure 14: Distribution of errors in the registration example of Figure 7 displayed as a needle map; the length of the needles is proportional to the error.

	Face
Min	0.006
Max	2.46
Mean	0.167
Standard deviation	0.215
Number of points	998

Figure 15: Matching and pose estimation error statistics for the examples of Figure 13. The error values are expressed in millimeters.

These numbers show that the overall behavior of the registration error is on the order of the resolution of the sensor, in this case 0.1mm. This shows, in particular, that the node correspondences found through SAI matching are correct and the estimation of the pose based on the correspondences is basically as accurate as it can be given the finite sensor resolution.

Only the nodes of the mesh that are visible, as determined by the geometry of the sensor, are actually used in the error computation. The errors at the other nodes is meaningless since

Pair number	Rotation angle	Absolute Error (degrees)	Axis error (degrees)	Point transformation error (mm)
1-2	30.12	0.12	2.96	0.69
2-3	30.02	0.02	1.53	0.68
3-4	32.86	2.86	3.72	0.80
4-5	30.63	0.63	2.71	0.71
5-6	30.15	0.15	2.79	0.69
6-7	29.85	0.15	4.60	0.68
7-8	29.69	0.31	2.61	0.74
Mean values	30.47	0.60	2.99	0.71
Mean values without pair 3-4	30.08	0.23	2.87	0.70

Figure 13: Pairwise rotation angles recovered from the views of Figure 9 using SAI matching. The true rotation angle is 30^{o}.

3.5.2. Surface Error

The surface error is somewhat more difficult to estimate because of the difficulty in getting ground truth. Our approach is to compute the transformations between partial views, to transform all the data points from all the views into a common reference frame and to compute the distances between the nodes of the mesh fit to the combined data set and the input data points.

Figure 13 shows an example of the surface error distribution for the face of Figure 7. This figure shows views of the mesh used for performing the registration of Figure 7 in three different orientations. The error at each node of the mesh, that is, the distance between the node and the closest point of the data set, is displayed as a needle, the length of which is proportional to the error. This display shows that the error is reasonably uniform across the mesh. The largest values of the error occur at the edge of the mesh. This is because there is poor overlap between mesh and data at those points.

For a more quantitative evaluation of the quality of the registration, Figure 14 lists error statistics computed on the example of Figure 13. The table lists the minimum, maximum, average, and standard deviation of the registration error at the nodes of the mesh. The registration error is defined as the distance between a mesh node and the closest data point after registration. The errors are listed in millimeters in the table. In both examples, the

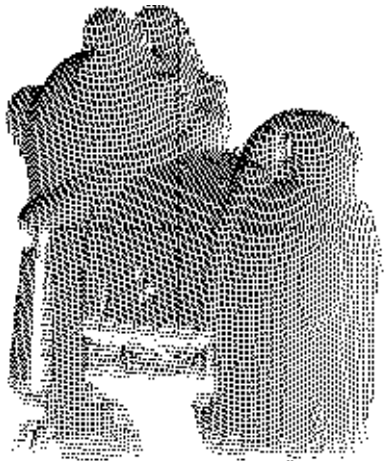

Figure 11: Range data points on image 1 of the sequence of Figure 9.

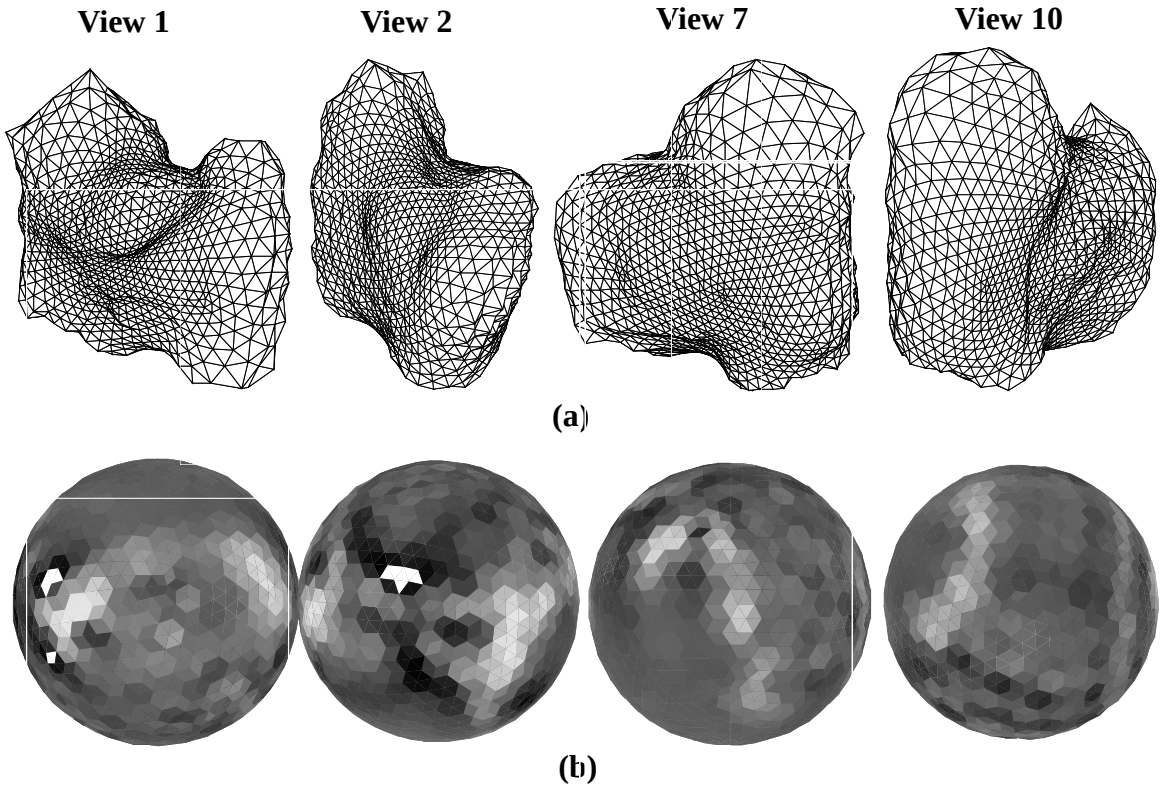

Figure 12: Data and surface representation used for the views 1, 2, 7, and 10 of Figure 9: (a) Mesh fit to the data points using the deformable surface algorithm; (b) Corresponding SAI representations displayed as shaded spheres.

achieved by computing the distance between a point transformed by the ground truth transformation and the same point transformed by the computed transformation. Any point could be used as long as it is near the surface of the object. We have computed this error, as reported in the last column of Figure 12 by using a set of points placed at regular intervals on a cube of side 100mm, which the typical size of the objects used in the experiments.

The error listed in Figure 12 is the average value over the set of points of $\|\boldsymbol{T}_g \boldsymbol{M} - \boldsymbol{T}_c \boldsymbol{M}\|$, where $\boldsymbol{T}_g$ is the actual transformation and $\boldsymbol{T}_c$ is the computed transformation. The error is expressed in millimeters. The maximum resolution of the sensor is 0.1mm. As anticipated, the worst error is for the outlier pair 3-4, although it is on the same order as the errors for the other views because of the nearly symmetrical shape.

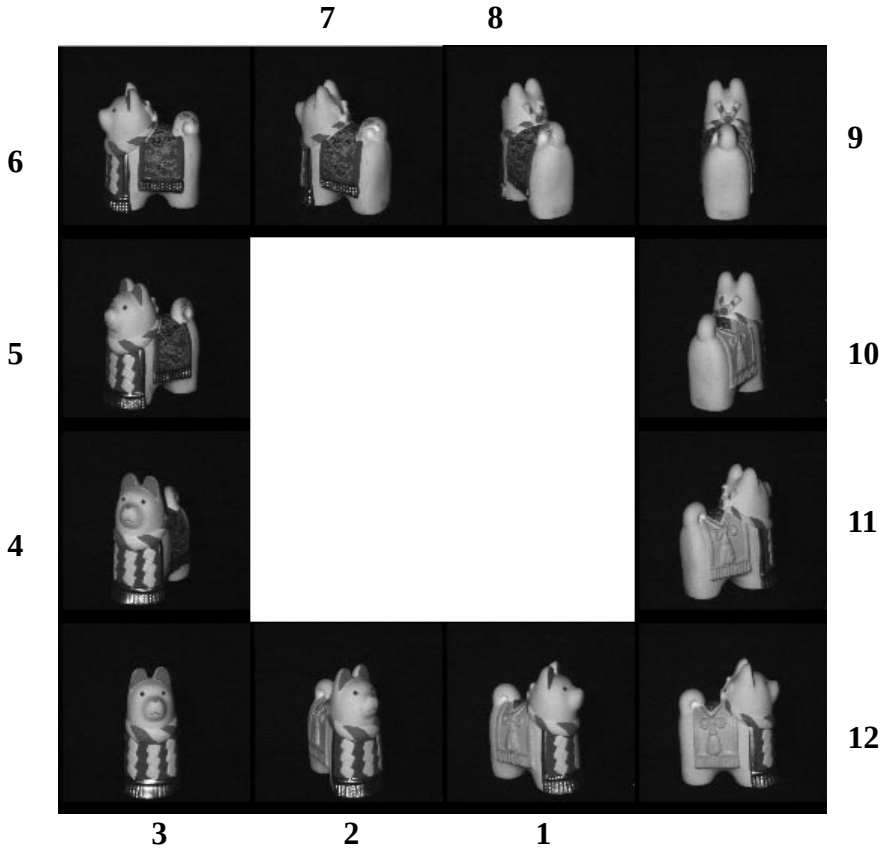

Figure 10: A sequence of 12 images of a toy dog.

shown in Figure 9; the range data points from the first view are shown as 3-D points in Figure 10. The object was placed on a precision turntable with its axis aligned with the Z-axis of the coordinate system in which the data points are measured. The object was rotated 30^o between images, thus providing the ground truth on the transformation. We were able to align the rotation axis with the Z-axis because the data coordinate system is attached to a calibration target that can be placed arbitrarily. In this experiment, we placed the calibration target such that the Z-axis would coincide with the rotation axis. We verified that the alignment was correct by rotating the table by a known amount and by comparing the rotation computed from the points on the calibration targets with the true rotation.

We computed the SAI representation on all the views of the object. Figure 11(a) shows the mesh fit to the data points using the deformable surface algorithm for views 1, 2, 7, and 10. Parts of the mesh are interpolated and do not correspond to valid data because only partial data is available from each individual view. Figure 11(b) shows the SAIs built from those meshes. The SAIs are displayed as shaded spheres with the shading proportional to the simplex angle value.

In order to evaluate the transformation error in the SAI matching algorithm, we matched pairs of consecutive views and compared the recovered transformation with the true transformation. A summary of the error statistics on the computed rotation angles is shown in Figure 12. The first column of the table shows the actual recovered angle; the second column of the table shows the absolute difference between the computed angle and 30^o, the actual rotation angle. The angular error is well below 0.5^o except for the pair 3-4. After analysis of the data, it turns out that the poor recovery of rotation is due to the fact that view number 3 consists mostly of the back of the dog which has a rotational symmetry about the Z-axis. Consequently, the resulting similarity surface produced by the SAI matching is very flat around the minimum, thus leading to the poor recovery of the rotation angle. We listed this pair for the sake of completeness although this condition can be identified by selecting only the views that have a sufficient SAI similarity measure for actual model building.

The average computed angle and the average absolute error are listed at the bottom of the table both with and without using pair 3-4. This result shows that, for all the views in which there is sufficient geometric information, the rotation is recovered within a fraction of a degree.

The third column of the first table of Figure 12 shows the error in the orientation of the computed axis, in degrees. This error is computed by taking the arc cosine of the Z component of the computed axis. The orientation of the rotation axis is computed correctly for pair 3-4 even though the rotation angle is not. This is because most of view 3 is rotationally symmetrical about the Z-axis so that the rotation angle is undefined whereas the axis is well-defined. The average error in the orientation of the axis is also listed at the bottom of the table.

Although errors in angles and axis and rotation do provide a quantitative basis for describing the error on the computed transformation, it is difficult to compare the full transformation with ground truth becomes of the coupling between the rotation and translation components. More precisely, an error in the rotation angle may be compensated by an error in the position of the axis, i.e. the translation component. A better error evaluation could be

The key in this rescaling 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 spherical scaling.

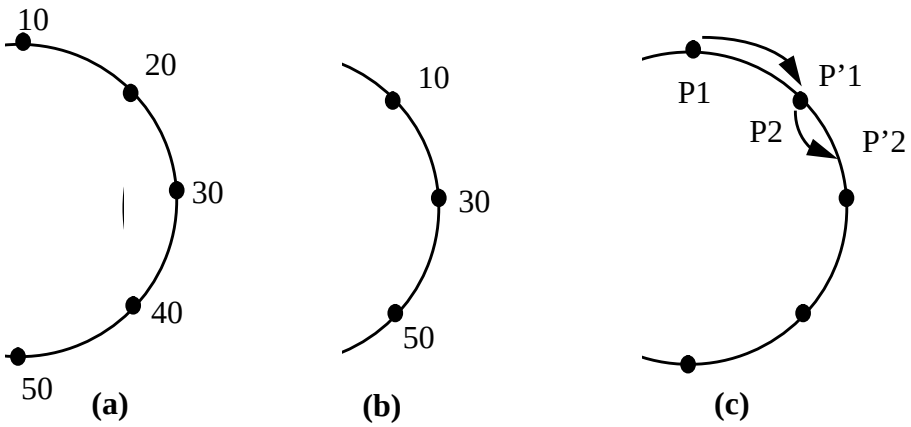

Figure 9: Simple illustration of the rescaling algorithm; (a) Five nodes are visible in view 1; (b) View 2 is the same SAI with a lower density of points on the surface, i.e., k = 2; (c) The SAI of view 2 is rescaled using a scale factor of 1/k = 1/2; for example, point P_1 is assigned the value of P'_1 in the original SAI.

3.5. Experimental Evaluation

There are two ways to evaluate the performance of the matching algorithm. The first approach involves comparing the transformation between views computed using the SAI matching with ground truth in order to compute the *transformation error.* This approach evaluates how well the transformation is recovered, not how well the mesh built after registration fits the data. The second approach addresses the latter: Two sets of data points are transformed to the same coordinate system and the mesh built from the combined data set is compared with the data points. More precisely, the distances between the mesh nodes and the data points are used for computing an error measure, which we call the *surface error.*

In addition to quantifying the quality of the registration from SAI matching, we need to verify our claim that the local regularity constraint defined in Section 2.1. does indeed produce a distribution of nodes with uniform density across the mesh. This is critical because the rescaling algorithms of Section 3.4. are entirely based on this assumption.

In the remainder of this Section, we evaluate the two matching errors and we verify the claim of uniform distribution through a set of examples.

3.5.1. Transformation Error

Figure 9 to Figure 12 illustrate one experiment designed for evaluating the transformation error. In this experiment, we took twelve images of a toy dog. The intensity images are

In this equation, k_1 and k_2 are the density of nodes, i.e., the number of nodes per unit surface area, for view 1 and view 2 respectively. We know k_i because we know the number of nodes visible in view *i*, S_i, and we can estimate the visible area A_i by taking the sum of the areas of the triangles that make up the surface triangulation.

From Equation (4), we see that in order for the two views to have the same density of nodes, we need to move the nodes of V_2 by a scale factor:

$$k \approx k_1 / k_2 \tag{5}$$

Once the nodes of V_2 are moved by this scale factor, the new density of nodes k'_2 satisfies $k_2' \approx k_1$ and the SAIs of the two views can be compared. In practice, the rescaling is implemented as follows: For each node ***P*** of the SAI of V_2, the corresponding node ***P'*** obtained by scaling ***P*** on the sphere by a factor 1/*k* is computed. The attribute value at ***P*** in the new SAI is then set to the attribute value of ***P'*** in the original SAI. The net effect is that the region on the sphere corresponding to the visible part of V_2 is expanded or contracted depending on the value of *k*. In this algorithm, only the nodes that are non-interpolated are taken into account since only those are used in comparing SAIs. More precisely, the value of ***P*** in the new SAI is updated only if the corresponding node ***P'*** is a valid node of the original SAI.

The rescaling procedure is illustrated in Figure 8. The five nodes in the SAI of V_1 have value 10 to 50 (Figure 8(a)). The same SAI is generated from view V_2 except that the node density is half the density in V_1 (Figure 8(b)), that is, $k = k_1/k_2 = 2$. Figure 8(c) shows the effect of rescaling on two nodes $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$. The node $\boldsymbol{P'}_1$ corresponding to $\boldsymbol{P}_1$ through the rescaling has value 10 in the original SAI of V_2. Therefore, $\boldsymbol{P}_1$ is assigned a value of 10. The node $\boldsymbol{P}_1$ was interpolated in the original SAI but is now a valid node because we have "expanded" the SAI.

Similarly, the node $\boldsymbol{P'}_2$ corresponding to $\boldsymbol{P}_2$ is half-way between a node with value 10 and a node with value 30 in the original SAI. Therefore, $\boldsymbol{P}_2$ is assigned the interpolated value 20. After applying to all the nodes the same procedure as for $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, the new SAI has the same density of nodes as the SAI of V_1 and the SAI values have been moved accordingly on the sphere. This example is a 2-D illustration of the algorithm. The algorithm is identical in 3-D except that values need to be interpolated between three points in the cases where a node does not fall exactly on a node of the original SAI after rescaling, instead of interpolating between two points as in the case for $\boldsymbol{P}_2$ in Figure 8.

This scaling procedure is applied to the SAIs, not to the surface meshes. The correspondences between mesh nodes that are used for computing the final transformation are established using the original mesh nodes and no scaling is necessary.

The underlying assumption for this algorithm to work is that the density of nodes in the surface mesh is approximately constant over any given mesh. As we have mentioned earlier, the local regularity constraint was introduced in order to enforce this condition. In Section 3.5.3., we will show through experimental measurements that the variation in mesh density is on the order of 2% in practice which is sufficient for the rescaling algorithms to work.

tal results show that the SAI matching provides an accurate transformation in an efficient way without any prior knowledge of the transformation. High precision algorithms, such as ICP, which require accurate prior estimate of the transformation and which are more expensive, can be used with the transformation from the SAI matching as a starting point if more precision is required. The results presented in this paper do not include this refinement step since we are interested in evaluating the performance of the SAI matching itself.

Figure 7 shows the final result of computing the transformation between the two views of Figure 5. 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.

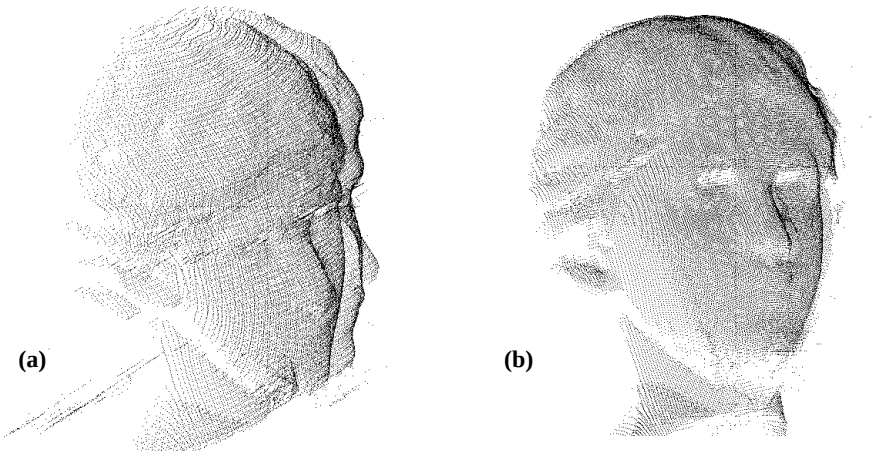

Figure 8: Merging two views; (a) Overlaid views before registration; (b) Overlaid views after registration.

3.4. Matching Partial Views

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. As a result, the density of mesh nodes in the common area of the object differs between the two views. Since we need to have approximately the same density of nodes in the two views in order to compare them using SAIs, we need to change the density of nodes in one of the views.

Let us consider two views, V_1 and V_2. Let S_1 and S_2 be the number of nodes that are visible from V_1 and V_2, and A_1 and A_2 be the surface area visible in V_1 and V_2, respectively. Let us assume for now that the node distributions in V_1 and V_2 are uniform, that is:

$$S_1 \approx k_1 A_1 \qquad S_2 \approx k_2 A_2 \tag{4}$$

Instead, we approximate the mapping by taking the closest nodes.

The second observation is that the goal of the alternative matching approach is to provide a meaningful way of discretizing the space of rotations. More precisely, in the previous approach, the space of rotations is discretized using constant increments in the three angles, ($\Delta\theta$, $\Delta\varphi$, $\Delta\psi$), which have no relation to the discretization of the sphere. As a result, those increments need to be set to very small values to ensure that the space of rotations is explored at the appropriate resolution, thus oversampling the search space. Using initial assignments of node to define the rotations gives us a way to define a discretization of the rotation space that is consistent with the discretization of the sphere and that eliminates the need for arbitrary increments. In particular, the number of rotations that need to be evaluated is directly proportional to the density of nodes.

In order to check that this version of the algorithm gives the same result as the previous version based on the exhaustive search in (θ, φ, ψ), we used both algorithms on the examples presented in this paper. In all cases, the rotation matrices corresponding to the best match between SAIs are different from the two algorithms. This is to be excepted since the first algorithm uses a finer discretization of the angles than the second. However, in all cases, the correspondences between nodes are exactly identical between the two algorithms. As a result, the final transformations are also identical.

3.3. 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 $\boldsymbol{R}_o$, the translational part by $\boldsymbol{T}_o$. Given a SAI rotation $\boldsymbol{R}$, we know the corresponding node $\boldsymbol{P'}$ of each node $\boldsymbol{P}$ of S. Let $\boldsymbol{M}$, resp. $\boldsymbol{M'}$, be the point on the object corresponding to the node $\boldsymbol{P}$ of S, resp. $\boldsymbol{P'}$. A first estimate of the transformation is computed by minimizing the sum of the distances between the points $\boldsymbol{M}$ of the first object and the corresponding points $\boldsymbol{R}_o\boldsymbol{M'}+\boldsymbol{T}_o$ of the second object. Formally, the expression to minimize is:

$$E(R_o, T_o) = \sum \left\| M - (R_o M' + T_o) \right\|^2 \tag{3}$$

The sum in this expression is taken over the set of all the nodes of the mesh. The optimum transformation for E can be computed in a non-iterative manner by using the standard quaternion-based techniques. The resulting transformation 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 $\boldsymbol{M}$ closest to $\boldsymbol{M'}$ for every node of the mesh and by computing again the minimum of $E(\boldsymbol{R},\boldsymbol{T})$ [5].

This last step is very similar to the ICP algorithm [1] in that $(\boldsymbol{R},\boldsymbol{T})$ is estimated by minimizing the distance between the points of one mesh and the corresponding closest points on the other mesh. As such, the final transformation is the best one that can be estimated given the resolution of the mesh. It should be noted that higher accuracy on the transformation could be achieved by using the ICP algorithm on the full data set instead of using the approximated mesh because of the much greater resolution of the original data set. The experimen-

other nodes, that is, there is a node $\boldsymbol{P'}_{ij}$ corresponding to a node $\boldsymbol{P}_i$ of *S*, given the initial correspondences. Moreover, there is only a small number of such initial correspondences, or, equivalently, there is a small number of distinct valid rotations of the unit sphere. In fact, the number of rotations is *3K* if *K* is the number of nodes.

Based on this observation, we developed an SAI matching algorithm decomposed into two stages: a pre-processing phase and a run-time phase. During pre-processing, we generate the data structure shown in Figure 6(b). The data structure is a two dimensional array in which each row corresponds to a possible rotation of the SAI and in which column *j* of row *i* is the index of the node $\boldsymbol{P}_{ij}$ corresponding to node $\boldsymbol{P}_j$ and correspondence number *i*. At run-time, the distance is evaluated for each row of the array:

$$D_i(S, S', R) = \sum (g(P_j) - g(P_{ij}))^2 \tag{2}$$

The row that produces the minimum D_i gives the best correspondence between nodes of the mesh, $\{(\boldsymbol{P}_j, \boldsymbol{P'}_{ij})\}$, which is used for computing the full transformation between the object meshes as described in the next section. It is important to note that this algorithm tries all possible rotations of the SAIs up to the resolution of the mesh. Consequently, it is guaranteed to find the global optimum of *D* and it does not require an initial estimate of the transformation. This validates our initial claims of global optimality and pose-independence of the algorithm. This is an efficient algorithm because all that is required at run time is to look up the correspondence table, to compute the sum of square differences of the corresponding nodes and to add them up. In our current implementation, the computation time is 7 sec. for *K* = 980 on a Sparc10 workstation.

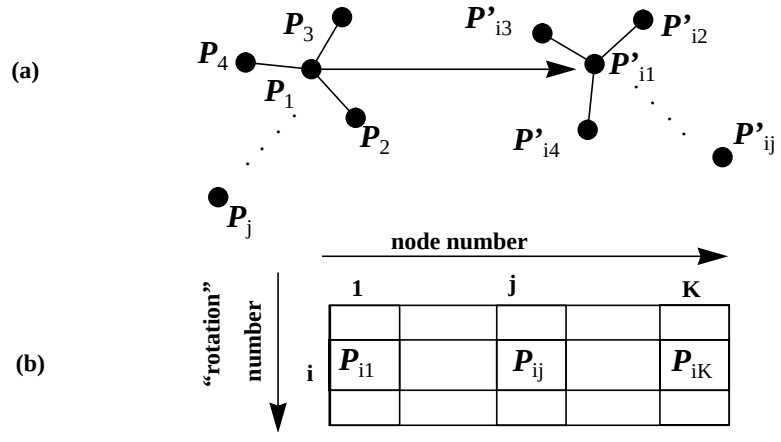

Figure 7: Efficient matching algorithm; (a) Valid correspondence between nodes; (b) Table of correspondences

There are two important observations that should be made about this algorithm. First, given a initial choice of nodes $\{\boldsymbol{P}_1,\boldsymbol{P}_2,\boldsymbol{P}_3\}$, the node $\boldsymbol{P'}_{ij}$ corresponding to $\boldsymbol{P}_i$ is not exactly at the same position, i.e. $\boldsymbol{P'}_{ij} = \boldsymbol{P}_i$ because the tessellation of the sphere is not perfectly regular. Instead, the node $\boldsymbol{P'}_{ij}$ is the one that is the *closest* to $\boldsymbol{P}_i$. In other words, the rotation defined by an initial assignment does not induce a 1-to-1 geometric mapping between the nodes.

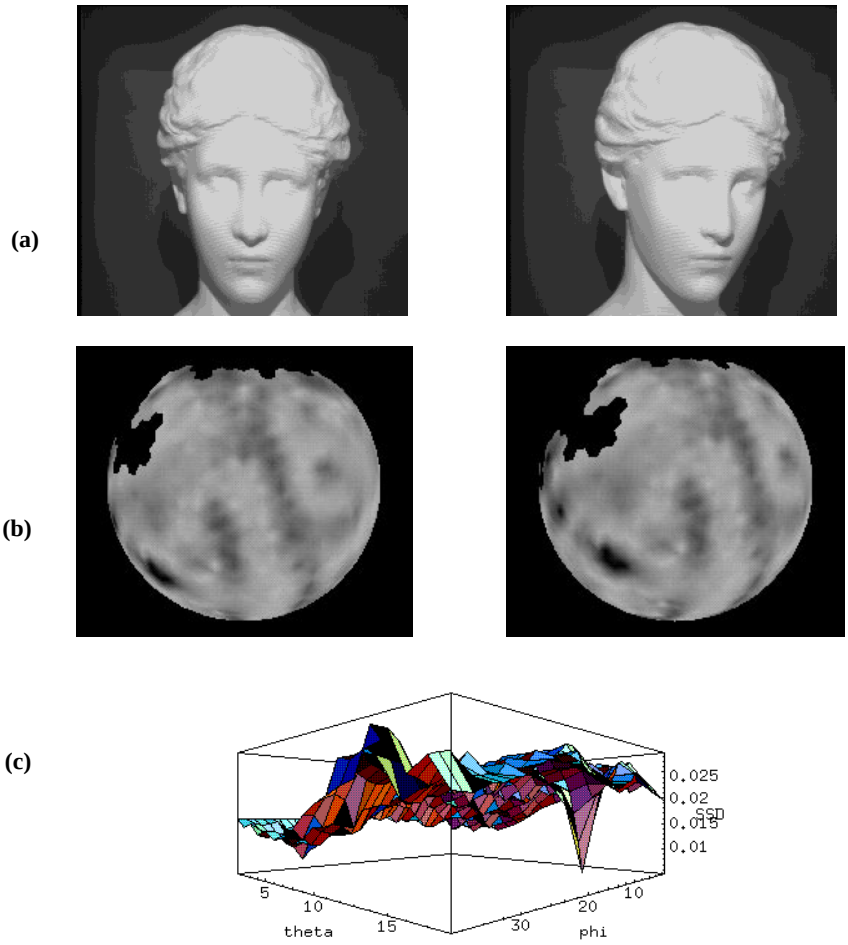

Figure 6: Matching two SAIs; (a) Intensity images of the views; (b) Corresponding SAIs with shading proportional to simplex angle; (c) Distance between two SAIs as function of two rotation angles φ and θ.

3.2. Efficient Matching

The graph of Figure 5 was obtained by sampling the space of all possible rotations, represented by three angles (θ, φ, ψ), and by evaluating D for every sample value (θ_i, φ_i, ψ_i). Although it is the approach that we used initially, it would be too expensive in practice to compute the distance for all possible rotations.

We developed an efficient SAI matching algorithm based on the observation that the only rotations for which $D(S, S', \boldsymbol{R})$ should be evaluated are the ones that correspond to a valid list of correspondences $\{(\boldsymbol{P}_i, \boldsymbol{P'}_j)\}$ between the nodes $\boldsymbol{P}_i$ of S and the nodes $\boldsymbol{P'}_j$ of S'. Figure 6(a) illustrates the idea of correspondences between nodes: Node $\boldsymbol{P}_1$ of the first SAI is put in correspondence with node $\boldsymbol{P'}_{i1}$ of S' and its two neighbors, $\boldsymbol{P}_2$ and $\boldsymbol{P}_3$, are put in correspondence with two neighbors of $\boldsymbol{P'}_{i1}$, $\boldsymbol{P'}_{i2}$ and $\boldsymbol{P'}_{i3}$, respectively. This set of three correspondences defines a unique rotation of the spherical image. It also defines an assignment for the

The problem now is to find this rotation using the discrete representation of S and S'. This is done by defining a distance $D(S, S', \boldsymbol{R})$ 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:

$$D(S, S', R) = \sum (g(P) - G(RP))^2 \quad (1)$$

The minimum of D corresponds to the best rotation that brings S and S' in correspondence.

Figure 5 shows the result of matching two views of a head. Figure 5(a) shows the intensity images of the two views of the object. Figure 5(b) shows the corresponding SAIs displayed as spheres shaded using the values of the simplex angle at the mesh nodes. Only the visible nodes of the mesh are used in generating this display. Visual inspection of the SAIs shown in Figure 5 shows that they are rotated versions of each other. Figure 5(c) shows the distribution of D as a function of two of the rotation angles, φ and θ. The graph exhibits a sharp minimum corresponding to the best rotation between the two spherical maps.

The rotation of the SAIs is not the same as the rotation of 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 in Section 3.3. below.

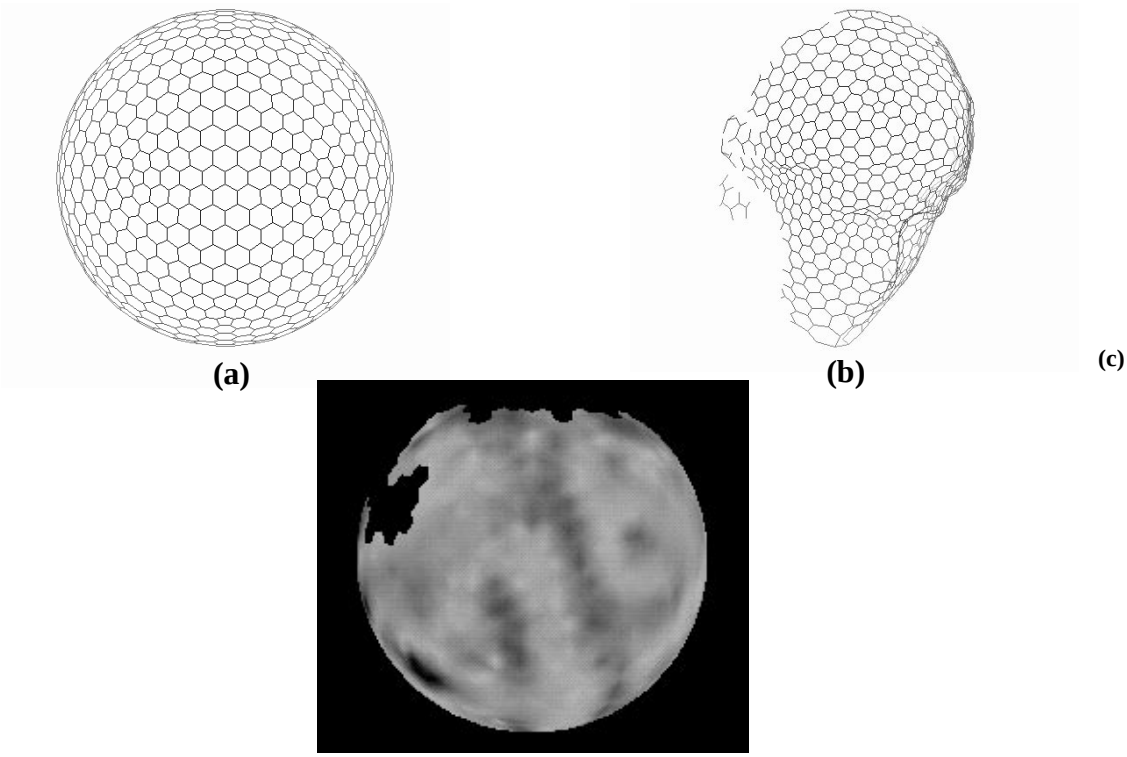

Figure 5: (a) Initial mesh; (b) Deformed mesh; (c) SAI represented on the unit sphere.

3. Registering Multiple Views

We now address the registration 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 between spherical images is determined, the full 3-D transformation can be determined.

3.1. Finding the Best Rotation Between SAIs

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. S and S' are representations of the same area of the object if there exists a rotation $\boldsymbol{R}$ such that $g(\boldsymbol{P}) = g(\boldsymbol{RP})$ for every point $\boldsymbol{P}$ of S. Since the SAI is discrete, $g(\boldsymbol{RP})$ is not defined because in general $\boldsymbol{RP}$ falls between nodes of S'. We define a discrete approximation of $g(\boldsymbol{RP})$, $G(\boldsymbol{RP})$, by interpolating the values of g at the four nodes of S' nearest to $\boldsymbol{RP}$, $\boldsymbol{P}_1$ to $\boldsymbol{P}_4$. Formally, $G(\boldsymbol{RP})$ is a weighted sum of $g(\boldsymbol{P}_i)$.

introduced in [4].

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. These two conditions can be expressed as a set of forces acting between mesh nodes and data points and between the mesh nodes and their neighbors. Once a locally regular mesh is created from the input data, a reference tessellation with the same number of nodes is created on the unit sphere. We refer the reader to [4] for the details of the algorithm.

Figure 3(a) and (b) show an intensity image and the corresponding set of points from the range image. In this example, we use the dual of the 9th subdivision of a 20-face icosahedron, (1620 faces) as shown in Figure 4(a). This initial mesh is deformed and reaches the stable state shown in Figure 4(b). The corresponding SAI data is shown in Figure 4(c). The SAI is displayed as a shaded sphere in which the intensity level at each node is proportional to the simplex angle.

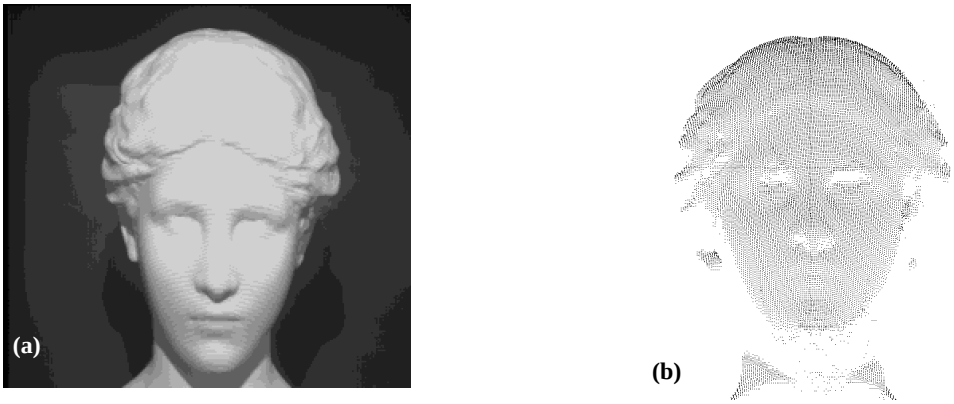

Figure 4: Input data; (a) Intensity image, (b) Range data.

tions.

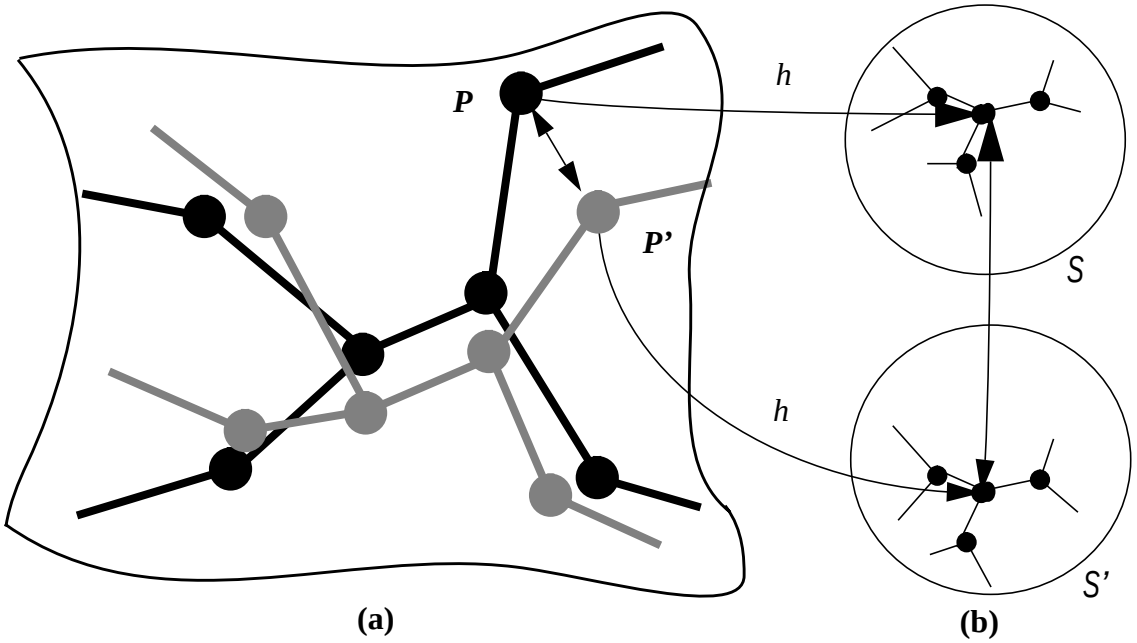

Figure 3: Illustration of the mapping between SAIs in the case of rotation between views; (a) A fragment of two meshes produced from two rotated copies of an object overlaid on the common surface patch; Node *P* corresponds to the closest node in the other mesh, *P'*, which has similar simplex angle; (b) SAI representation of the same meshes; The correspondence between *P* and *P'* induces a correspondence between the nodes of the SAIs.

The representation described so far is a complete mesh covering a close surface isomorphic to a sphere. In practice, however, we usually have only partial data on the object because only a few views are available from the sensor. Therefore, only a subset of the mesh correspond to actual data points. In order to differentiate those mesh nodes that correspond to actual data and those that are interpolated without reference to data points, we compute the data point that is the closest to each node of the mesh. Nodes for which the distance to the closest data point is greater than a threshold are marked as “interpolated” and are not used in the matching algorithms. Nodes that are close enough to data points are marked as “valid” and are the only used in the matching. The threshold used for differentiating between valid and interpolated points is relative to the average excepted size of the object so that the result is independent of scale.

2.4. Extracting the SAI from a Range Image

In the previous sections, we have described the basic approach to representing a mesh of points as a spherical image. The remaining problem is to compute the 3-D mesh given a set of points from a range image. We use directly the algorithm based on deformable surfaces

ing the model depend only on the relative position of the data points with respect to the mesh, not on their absolute positions. If the data points are all translated by the same amount, the initial mesh is also translated by the same amount and, therefore, the final mesh is unchanged, that is, the absolute coordinates of all the nodes of the mesh are translated by the same amount, but the relative positions of the mesh nodes are the same as they were before translation.

Second, the SAI is also invariant to scaling. This is because, in the deformable surface algorithm that we use, the parameters that control the convergence are computed relative to the size of the object. Therefore, if all the data points are scaled by the same amount, the mesh converges to the same mesh, that is, the absolute coordinates of all the nodes of the mesh are scaled by the same amount, but their relative positions are unchanged. Since the value of the simplex angle is independent of scale, the values stored in the SAI are the same as before scaling.

The case of rotation is a bit more difficult. The SAIs S and S' of two rotated copies M and M' of a given object are not identical but we can define a mapping between the two SAIs which the best mapping between the nodes of M and M' given the resolution of the meshes. Figure 3 illustrates the situation. Vertices and edges of the mesh M are drawn in black on a fragment of the object's surface. The vertices and edges of the mesh M' obtained by fitting a mesh after rotation of the object are drawn in grey on the same surface fragment. Assuming that the distribution of nodes over the surface is constant, the two meshes would overlap as shown in Figure 3. Although the nodes are not exactly at the same location on the surface, we can establish a correspondence between each node of M and the closest node of M'. For example, node $\boldsymbol{P}$ of M is associated node $\boldsymbol{P'}$ of M' in Figure 3.

Since the simplex angle is invariant by rotation, corresponding nodes have similar simplex angle values. The problem is now to find this correspondence and that is where we need to use the SAI representation: Each node of M (resp. M') corresponds to a node on the unit sphere, S (resp. S'.) Therefore the mapping that we defined between M and M' also defines a mapping between S and S'. This mapping puts in correspondences nodes of the SAIs that have similar simplex angle values. Since the SAIs are represented on the unit sphere, we can represent this mapping by a rotation of the unit sphere.

This reasoning will provide the basis for our matching algorithms: First find the best rotation of the SAIs that brings in correspondence points with similar attribute values, then use the corresponding mapping between mesh nodes in order to compute the transformation between the two views of the object.

This approach is entirely based on the fact that the distribution of nodes on the surface is nearly uniform. This is the motivation for using the local regularity constraint and we will evaluate experimentally the quality of the node distribution in Section 3.5.3.

Another important consequence of the definition of h is that it preserves connectivity. More precisely, a connected patch of the surface maps to a connected patch of the spherical image. It is this property that allows us to work with non-convex objects and to manipulate models of partial surfaces, neither of which are possible with conventional spherical representa-

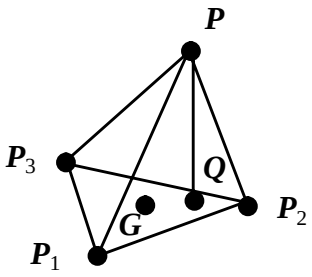

Figure 1: Local Regularity

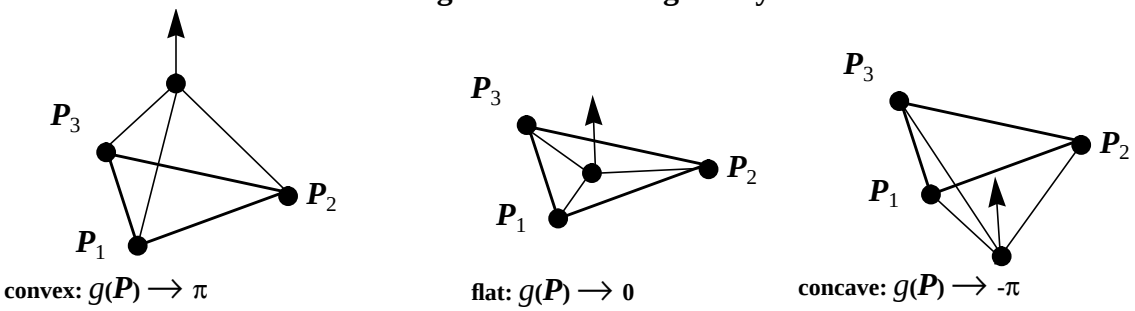

Figure 2: Definition of the Simplex Angle

2.3. Spherical Mapping

A regular mesh drawn on a closed surface can be mapped to a spherical mesh in a natural way. For a given number of nodes K, we can associate to each node a unique index which depends only on the topology of the mesh and which is independent of the shape of the underlying surface. This numbering of the nodes defines a natural mapping h between any mesh M and a reference mesh S on the unit sphere with the same number of nodes: $h(\boldsymbol{P})$ is the node of S with the same index as $\boldsymbol{P}$.

Given h, we can store at each node $\boldsymbol{P}$ of S the simplex angle of the corresponding node on the surface $g(h(\boldsymbol{P}))$. The resulting structure is a spherical image, that is, a tessellation on the unit sphere, each node being associated with the simplex angle of the corresponding point on the original surface. We call this representation the Spherical Attribute Image (SAI)[1]. In the remainder of the paper, we will denote by $g(\boldsymbol{Q})$ instead of $g(h^{-1}(\boldsymbol{Q}))$ the simplex angle associated with the sphere node $\boldsymbol{Q}$.

If the original mesh M satisfies the local regularity constraint, then the corresponding SAI has several invariance properties. First, for a given number of nodes, the SAI is invariant by translation and scaling of the original object. This can be shown by observing that the convergence of the deformable surface algorithm that we use is affected only by the distance between data points and the nodes of the mesh because the forces that are used for deform-

1. In previous papers on this subject, we used to refer to the SAI as the “Simplex Angle Image”. The new name reflects the fact that this representation may be used to store any attribute computed or measured on the surface, not just the simplex angle.

the data set and the mesh which, when combined, force the mesh to converge to the correct shape. The algorithm is described in detail in [4].

As we shall see, the key to our matching algorithms, in particular in the case of partial views, is to produce meshes in which the density of nodes on the object's surface is nearly uniform. Although perfectly uniform distribution is impossible, a simple local regularity constraint can enforce a very high degree of uniformity across the mesh. The local regularity constraint is defined as follows: Let $\boldsymbol{P}$ be a node of the tessellation, $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, $\boldsymbol{P}_3$ be its three neighbors, $\boldsymbol{G}$ be the centroid of the three points, and $\boldsymbol{Q}$ be the projection of $\boldsymbol{P}$ on the plane defined by $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, and $\boldsymbol{P}_3$ (Figure 1). The local regularity condition simply states that $\boldsymbol{Q}$ coincides with $\boldsymbol{G}$.

All the meshes used in this work are computed using a modified version of the deformable surface algorithm in which the local regularity constraint is incorporated as a force that tends to move each node $\boldsymbol{P}$ so that $\boldsymbol{Q} = \boldsymbol{G}$ with the notation of Figure 1. The implementation of the local regularity constraint is described in detail in [5].

This local regularity constraint is a generalization to three dimensions of the regularity condition on two dimensional discrete curves which simply states that all segments are of equal lengths. The difference between the two- and three-dimensional cases is that it is always possible to create a uniform discrete curves in 2-D, while only nearly uniform discrete surfaces can be generated in 3-D. We will show in Section 3.5.3. that, in practice, the density of mesh nodes on the surface varies in the order of 2%.

2.2. Discrete Curvature Measure

The next step in building a discrete surface representation is to define a measure of curvature that can be computed from a tessellation. Instead of estimating surface curvature by locally fitting a surface or by estimating first and second derivatives, we proposed in [5] a measure of curvature computed at every node from the relative positions of its three neighbors. We called this measure of curvature the simplex angle and we denote its value at node $\boldsymbol{P}$ by $g(\boldsymbol{P})$. Although $g(\boldsymbol{P})$ is not the curvature at $\boldsymbol{P}$, it behaves as a qualitative measure of curvature which is sufficient for matching purposes. Figure 2 illustrates the behavior of $g(\boldsymbol{P})$: The simplex angle varies between $-\pi$ and π. The absolute value of $g(\boldsymbol{P})$ is large in absolute value if $\boldsymbol{P}$ is far from the plane of its three neighbors and vanishes as $\boldsymbol{P}$ and its three neighbors are in the same plane. Finally, $g(\boldsymbol{P})$ is negative if the surface is locally concave, positive if it is convex. Given a configuration of four points, $g(\boldsymbol{P})$ is invariant by rotation, translation, and scaling[1] because it depends only on the relative positions of the points, not on their absolute positions.

1. Although the simplex angle is not formally the mean or Gaussian curvature of the surface, we frequently refer to it as the "curvature" of the surface at a node of a discrete mesh.

We describe the algorithms used for building SAIs from range images in Section 2. We first describe the concept of semi-regular meshes (Section 2.1.) and the measure of curvature (Section 2.2.) which are the basis for the surface representation. Then we introduce the mapping between surface mesh and spherical mesh in Section 2.3. Finally we describe the algorithm used for extracting representation from range data in Section 2.4. This discussion will show that there is no underlying assumption about the surface except that it is without topological holes, thus supporting our claim that our approach is suitable for *free-form surfaces*. In Section 3., we describe how two partial representations of the same object from two different poses can be registered. We first show how to compute a rotation of the spherical image in Sections 3.1. and 3.2.. We show in Section 3.2. that the search for the optimal rotation can be made very efficient, provided that some tables are pre-computed. The algorithm of Section 3.2. will validate our claim that the matching algorithm requires *no initial estimates* of the transformation and that it is guarantee to find the *best transformation* up to the resolution of the mesh. We show how to convert this rotation into a full 3-D transformation between surfaces in Section 3.3.. Since no assumption is made on the transformation and since no prior estimate is needed, we will show that the algorithm is able to match surfaces from *arbitrary poses*. We discuss the issue of matching partial views in Section 3.4. We evaluate the performance of the algorithms for pairwise view matching in Section 3.5. In addition to evaluating the accuracy of the transformations recovered by the SAI matching, we also verify a critical property of the representation, that is, the nodes are nearly uniformly distributed on the surface. Finally, we show how to build complete models in Section 4.

2. Spherical Attribute Images

In this section, we briefly introduce the concept of SAI. First, we explain how to tessellate an arbitrary surface into a semi-regular mesh, and how to calculate the simplex angle, a variation of curvature, at the nodes of the mesh, and how to map the mesh to a spherical image. Finally, we discuss how to handle partial views of 3-D objects.

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 neighbors by an arc of the graph. We use a type of mesh such that each node has three neighbors. Such a mesh can be constructed as the dual of a triangulation of the surface. Let us first consider tessellations of the unit sphere. We use a standard semi-regular triangulation of the unit sphere constructed 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 20 N^2-face triangulation, yielding a tessellation with the same number of nodes.

In order to obtain a mesh from an arbitrary surface, we use an algorithm based on the concept of deformable surfaces, in which we deform a tessellated surface until it is as close as possible to the object surface (Section 2.4.). This algorithm is based defining forces between

1. Introduction

Most computer vision systems require accurate three-dimensional models. The problem of building such models from observations consists in taking multiple range image of the object from different viewing positions and orientations, referred to as "viewing poses", to match the data in the different images in order to recover the relative poses, and to merge the data into a single model using the estimated poses. The approaches proposed so far suffer from two major limitations. First, they require accurate knowledge of the relative viewing poses. Second, they either require a complicated feature extraction algorithm to be applied to the range image or they restrict the class of shapes that can be modelled. Our goal in this paper is to eliminate these two restrictions in order to allow modelling of natural, free-form objects from arbitrary unknown viewpoints. Therefore, our goal is more in "generality" than in "accuracy". In particular, the techniques presented in this paper may be used as a front-end to high-accuracy registration techniques. However, we claim that those techniques cannot, by themselves, operate under our assumptions of general curved surface and of absence of initial pose estimate.

Examples of feature-based model building include the work of Parvin and Medioni [8] in which they 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 determine the transformation between the graphs. This method limits the class of shapes to which it can be applied since it requires stable segmentation results. Other techniques, such as Kamgar-Parsi's [6] 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 [9] in which the virtual features are groups of surface normals.

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. This type of approach requires an initial estimate of the relative viewing poses.

Besl [1] proposed an algorithm for matching between free-form surfaces. The algorithm is based on iterative projection of one surface on the other. A similar approach was suggested by Chen and Medioni [3] and by Zhang [10]. Besl's approach has the advantage that it does not require extracting features or establishing correspondences between features. However, because it is an iterative algorithm, it is very sensitive to the initial transformation.

In this paper, we propose a different approach to the model building problem. Our approach is based on the representation of free-form surfaces developed in [4][5]: A mesh of points is fit to an input set of data points from each view, a curvature measure is computed at every node of the meshes and map to a spherical image, the Spherical Attribute Image (SAI). The transformation between views is computed by comparing their SAIs. Finally, the data points from all the range images are merged into a single set using the estimated poses and a complete surface model is computed.