Local Matching of Surfaces Using Critical Points

Gerald Radack

Norman I. Badler
University of Pennsylvania, badler@seas.upenn.edu

Local Matching of Surfaces Using Critical Points

Abstract

The local matching problem on surfaces is: Given a pair of oriented surfaces in 3-space, find subsurfaces that are identical or complementary in shape. A heuristic method is presented for local matching that is intended for use on complex curved surfaces (rather than such surfaces as as cubes and cylinders).

The method proceeds as follows: (1) Find a small set of points-called "critical points" -on the two surfaces with the property that if p is a critical point and p matches q, then q is also a critical point. The critical points are taken to be local extrema of either Gaussian or mean curvature. (2) Construct a rotation invariant representation around each critical point by intersecting the surface with spheres of standard radius centered around the critical point. For each of the resulting curves of intersection, compute a "distance map" function equal to the distance from a point on the curve to the center of gravity of the curve as a. function of arc length (normalized so that the domain of the function is the interval [0,1]). Cll the set of contours for a given critical point a "distance profile." (3) Match distance profiles by computing a "correlation" between corresponding distance contours. (4) Use maximal compatible subsets of the set of matching profiles to induce a transformation that maps corresponding critical points together, then use a cellular spatial partitioning technique to find all points on each surface that are within a tolerance of the other surface.

Disciplines

Computer Engineering | Computer Sciences

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-84-13.

LOCAL MATCHING OF SURFACES USING CRITICAL POINTS

Gerald M. Radack
Norman I. Badler

MS-CIS-84-13

Department of Computer and Information Science
Moore School/D2
University of Pennsylvania
Philadelphia, PA 19104

May 1984

[*] Acknowledgement: NASA grant NAS9-16634 and NAS9-17239.

Local Matching of Surfaces Using Critical Points

Gerald M. Radack

Norman I. Badler

May 1984

Technical Report MS-CIS-84-13

Department of Computer and Information Science
School of Engineering and Applied Science / D2
University of Pennsylvania
Philadelphia, PA 19104-3897

This work was supported in part by NSF under grants MCS78-07466 and MCS82-19196, and by NASA under contract NAS9-16634.

Local Matching of Surfaces Using Critical Points

Abstract

The local matching problem on surfaces is: Given a pair of oriented surfaces in 3-space, find subsurfaces that are identical or complementary in shape. A heuristic method is presented for local matching that is intended for use on complex curved surfaces (rather than such surfaces as as cubes and cylinders).

The method proceeds as follows: (1) Find a small set of points—called "critical points"—on the two surfaces with the property that if p is a critical point and p matches q, then q is also a critical point. The critical points are taken to be local extrema of either Gaussian or mean curvature. (2) Construct a rotation invariant representation around each critical point by intersecting the surface with spheres of standard radius centered around the critical point. For each of the resulting curves of intersection, compute a "distance map" function equal to the distance from a point on the curve to the center of gravity of the curve as a function of arc length (normalized so that the domain of the function is the interval [0,1]). Call the set of contours for a given critical point a "distance profile." (3) Match distance profiles by computing a "correlation" between corresponding distance contours. (4) Use maximal compatible subsets of the set of matching profiles to induce a transformation that maps corresponding critical points together, then use a cellular spatial partitioning technique to find all points on each surface that are within a tolerance of the other surface.

1. Introduction

The local matching problem for surfaces is: Given two surfaces in 3-space, find subsurfaces (one from each surface) that are similar in shape. Of course, there are many possible interpretations of "similar in shape." In this paper, we are concerned with rigid matching of noisy data. Thus, we will take "similar in shape" to mean that for each pair of matching regions there exists a rigid motion that can be applied to one region that brings all points of each region to within a fixed tolerance of the other region. We will present a heuristic method for local matching that will work on a broad class of surfaces, but is geared toward irregularly shaped surfaces.

2. Motivation

Local surface matching is important in analyzing how objects fit together. Here are some possible applications:

2.1. Biochemistry

Interaction of two complex organic molecules (e.g. protein) occurs when geometrically complementary surface sections come together [7]. Although the molecules are not always completely rigid, they assume certain well-defined shapes and the rigid model remains useful [11]. Although other factors may influence the interaction, the area of the matching surfaces is important in predicting the likelihood of a reaction mode. Thus, local matching could be used to screen pairs of molecules for possible reactions and find the most likely configurations of complexes.

2.2. Jigsaw puzzles

A system for assembling three-dimensional jigsaw puzzles clearly requires local surface matching in order to find how pairs of pieces can fit together. Of course, additional higher level algorithms would be needed to control the puzzle assembly. See [9] and [5] for papers on two-dimensional jigsaw puzzle matching.

2.3. Scene analysis

Only portions of objects in a scene may be visible in a single picture. This limits the methods that can be used to recognize objects. One approach is to use color or texture to recognized objects. However, in many applications this information is not available or not unique enough to disambiguate objects. Another approach is use silhouettes to recognize the objects. Often, however, silhouettes will not be sufficient for recognition. A more general approach is to use surface geometry (obtained through stereo, shape from shading or structured light) for recognition. One approach to the identification problem is to integrate multiple views of an object [3, 4] in order to obtain a complete object shape description that can then be matched using any traditional matching method [2] (for example, moments). Often, it may not be possible or convenient to obtain multiple views of a scene, and objects may still be occluded. Local matching can be used to recognize objects when only parts of the surfaces can be reconstructed.

3. Previous work

Oshima and Shirai [8] described a method for recognizing objects that uses local matching. Structured light was used to obtain three-dimensional coordinates from a TV picture. The resulting points were segmented into planar or quadric surface regions. A graph was then built describing the regions and relations between them. This graph was matched against graphs that were constructed while the system was in a "learning mode" in order to obtain a match.

This method assumes that the surfaces can be decomposed into well defined patches and that patches will correspond on two matching surfaces. Thus, the method is more suited for matching surfaces of objects that are the same, and would not be suitable for applications in which a surface fit is to be found (for example fitting three-dimensional jigsaw puzzle pieces).

Watson and Shapiro [10] presented a method for local matching of images of three-dimensional objects, but they worked with line drawings rather than surface data.

4. Surface representation

There are several representations in common use for surfaces [1], including quadric patches, spline patches and polygonal networks (sometimes called "polyhedral surfaces"). We have chosen to represent surfaces by networks of polygons—in particular triangles—because of the ease with which geometric operations necessary to our matching method can be done on triangles. Any surface can be represented by triangles to as close a tolerance as desired.

5. Matching method

Each surface to be matched is processed in order to extract certain features that can be used in matching and construct a rotation and translation independent representation. The actual matching can then proceed quickly. This preprocessing is particularly advantageous when a surface has to be compared against a number of other surfaces.

Our matching method works as follows:

Preprocessing

1. Identify a small set of points on each surface ("critical points") with the property that points in this set should match other points in the set.
2. Construct a representation of surface shape in the neighborhood of each critical point (called a "distance profile") which is rotation and translation invariant.

Matching

1. Compare all pairs of distance profiles from the two surfaces for matches.
2. Find maximal compatible sets of pairs of matching profiles. (A set is compatible if there is a single transformation which, when applied to one surface, brings together all the corresponding critical points from the set.)
3. For each set of profile pairs, apply the transformation mentioned in step 2 and find all points on each of the two surfaces that are within a tolerance

of the other surface.

5.1. Critical points

Since curvature at a point on a surface depends only on the shape of the surface in a small neighborhood of a point, the curvatures at corresponding points on matching regions should be the same. In particular, maxima and minima of curvature should occur at corresponding points on two matching regions. Thus, we can use maxima and minima of curvature as the points around which the distance profile representation is constructed. We shall call maxima and minima of curvature "critical points."

There are several "curvatures" defined on surfaces, including Gaussian curvature and mean curvature. If mean curvature is used, then saddle points on a surface will not be identified as critical points. If Gaussian curvature is used, then other points which would have been identified as critical points by mean curvature will be missed. Therefore, it is best to use minima and maxima of both mean and Gaussian curvature for critical points.

5.2. Finding critical points

We have defined critical points to be maxima or minima of Gaussian or mean curvature. To find critical points, we compute an approximation to curvature at each vertex of a triangulated surface. Then any vertices that have a curvature higher than all vertices within a fixed neighborhood (whose size is a parameter selected by the user) are identified as critical points.

In many cases, we may not have mathematical descriptions of the surfaces to be matched in a form from which the curvature can be computed directly. We assume that only polyhedral approximations to the surfaces are available. Thus, we must approximate curvature of an underlying surface from the triangulated surface. To compute curvature at vertex V, we fit a polynomial surface to vertices within a cylinder whose axis is the normal at V (where AIWID is a user-selectable parameter) and whose radius is AIWID. (The cylinder is used because a function of the form $z = f(x,y)$ is fitted.) To find these vertices, a region growing procedure is used. First we convert to a coordinate system in which the normal at V is parallel to the z axis. Then we initialize region R to contain an arbitrary triangle having V as a vertex. We add triangles adjacent to R as long as they share an edge with a triangle already in R and have at least one vertex whose (x,y) coordinates are within AIWID of the (x,y) coordinates of V. The parametric surface is then fit to the set P of all triangle vertices in R that have (x,y) coordinates within AIWID of the (x,y) coordinates of V.

Given a set P of points in 3-space, we fit a surface by finding the coefficients a_{ij} $(0 \leq i + j \leq 2)$ that minimize

$$\sum_{p \in P} [p_z - f(p_x,p_y)]^2$$

where $f(x,y) = \Sigma\, a_{ij} x^i y^j$.

The curvature can then be computed from the coefficients of the polynomial f.

5.3. Constructing the profiles

After a surface's critical points have been found, the distance profile representation of the surface can be computed. Each distance profile represents the local geometry in the neighborhood of a critical point. Recall that a profile is made up of a collection of distance contours. A distance contour is the locus of all points on the surface that are a fixed distance from a point called the *center point* of the contour. The center point of each contour is a critical point. In our implementation, we restrict the distances from the center point that are used for computing contours to be integral multiples of a real number called the radius increment (RINCR). The set of points on the surface at distance k·RINCR from a center point p is called "contour level k of the profile around p." Note that this is equivalent to the intersection between the the surface and a sphere with radius k·RINCR and center p.

A contour of intersection between a sphere and a triangulated surface can be decomposed into curves that are either closed or start and end on the boundary of the surface.

An encoded surface can be thought of as an array of profiles, one for each critical point. A profile contains the location of the center point and an array of contour levels, normally ranging from 1 to some value less than 15. Each level is either the empty set (if there is no intersection at that level) or a linked list of connected curves. Note that if contour level k is empty, then all contour levels greater than k must also be empty. This follows from the fact that the surfaces are connected and the distance function is continuous. Each curve in a contour is represented as a set of points which when connected by line segments approximate the true curve of intersection.

We will now discuss the algorithm used to compute the contours. The problem is to generate a set of curves that represents the locus of all points on a triangulated surface that are distance D (where $D = k \cdot \text{RINCR}$ for some k) from a point p on the surface. The algorithm for contour generation is:

```
initialize list Q to empty;
for each triangle T in the database
   if T intersects the sphere with radius $D$ and center $p$ then
      add T to list Q;
while Q is not empty do
   begin
   choose an arbitrary triangle A from Q;
   generate a connected curve on the contour that goes through
    triangle A, removing from Q each triangle T through which
    the curve passes if all curves through T have been generated[1];
   end;
```

[1]There can be up to three curves of intersection between a triangle and a sphere.

5.4. Matching profiles

Profiles are used to represent the shape of neighborhoods of critical points in a way that allows easy comparison for shape similarity. Two profiles P and Q are said to match if

1. The curvatures at the center points of the profiles are the same to within a tolerance.
2. A specified number of contours on one profile match contours on the other profile.

5.5. Contour matching

Any matching method that will work on space curves may be used to match contours. If we assume that contours are simple closed curves, then a representation we call the "distance map array" (a modified version of Freeman's centroidal profile [6]) may be used. The distance map array for a contour C is defined by

$$\mathrm{dmaC}[i] = d(C(i\cdot(\mathrm{length}(C)/\mathrm{NDISMP}),\bar{C}) \qquad 1 \leq i \leq \mathrm{NDISMP}$$

$$\mathrm{dmaC}[i+\mathrm{NDISMP}] = \mathrm{dmaC}[i]$$

where $\bar{C}$ is the center of gravity of the contour:

$$\bar{C} = \int C(s)\, ds$$

NDISMP is the size of the distance map array. We typically use 50.

Note that this representation does not change if the contour is rotated or translated. Therefore, if two contours represent the same shape curve, then their distance map arrays can only differ by a cyclic shift. Thus, two distance map arrays dmaC and dmaD represent the same shape curve only if there exists a *k* such that

$$\mathrm{dmaC}[i] - \mathrm{dmaD}[i+k] < \mathrm{DMTOL} \qquad 1 \leq i \leq \mathrm{NDISMP}$$

(DMTOL is another parameter.) We assume here that the contours match globally. The contours should match globally within a small enough neighborhood of a critical point unless the critical point is on the boundary of a matching region. However, it would be better to match the contours locally.

5.6. Finding compatible profile matches

Corresponding to each pair of profiles that have been matched, there is a number which represents the goodness of the match. (The smaller the number is, the better the match.) We arrange all pairs of profiles whose "goodness" is below a certain threshold into a table called the "match table."

We then use a backtracking search of the table to find all maximal compatible sets of pairs. A set of pairs $\{P_i, Q_i\}$ of profiles is compatible if there exists a transformation which, when applied to the center points of the Q_i, will bring the center points P_i and Q_i together (to within a tolerance) for all i. From each set we can derive a transformation, and if a set contains three or more noncollinear pairs of points, there will be a unique transformation that minimizes the mean distance between corresponding points. Once we have this transformation, we can apply it to the second surface and find all points on each surface that are within a given tolerance (called NEARTOL) of the other surface.

5.7. Finding matching regions

Once we have a transformation that brings together some pairs of points on two surfaces (points which happen to be critical points), we wish to find *all* points on each surface that are within a tolerance of the other surface. Assume that the transformation has been applied to the second surface. If we consider each point p to be represented by a box with side length NEARTOL and center p, then two points can only be within NEARTOL of each other if their boxes overlap (i.e., if there exists a point contained in both boxes). Therefore, if we enter a point into the list of points for each cell through which its box passes, then two points can only be within NEARTOL of each other if there exists a cell with both points in its list of points.

The algorithm for finding the regions is:

```
initialize CHAIN to empty;
for each point $p$ in $S_1$ and $S_2$ do
   for each cell $C$ which intersects the box of $p$ do
      begin
      if $C$ is not in the hash table then
         begin
         add $C$ to the hash table as an empty cell;
         add $C$ to list CHAIN;
         end;
      add $p$ to the list of points in $C$
      end;
for each cell $C$ in list CHAIN do
   for each point $p$ of $S_1$ in $C$ do
      for each point $p'$ of $S_2$ in $C$ do
         if $d(p,p')$ < NEARTOL then
            begin
            add $p$ to $R_1$;
            add $p'$ to $R_2$;
            end;
```

If the average distance between neighboring vertices of a triangulated surface is greater than NEARTOL, we can make the cells cubes with width equal to the average distance between neighboring vertices. This means that there will be an average of one vertex from each surface per cell. Since there can be no more nonempty cells than there are vertices on the two surfaces, the cost of the "for each cell" loop is $O(|S_1|+|S_2|)$. Therefore, the total cost of the above algorithm is $O(|S_1|+|S_2|)$.

6. Results

The matching method described in previous chapters was implemented and tested on a series of surfaces. Our first test was with the two surfaces shown in Figure 7-1. The first surface consists of a flat plane with a pyramid, a Gaussian hill and a "volcano" superimposed. The second surface is the same except that a polynomial hill is added. Figure 7-2 shows the result of matching the surfaces. The matching regions are shaded lighter than the nonmatching regions. In this case, the matching regions happen to cover most of the surfaces.

Our next test used similar surfaces, except the flat corner of the first surface was bent down. The result is shown in Figure 7-3.

Our third test was performed on the "plug" and "outlet" in Figure 7-4. The result of matching is shown in Figure 7-5.

7. Conclusions

We have presented a local matching method for surfaces.

The limitations of the method are:

- Matching regions must contain critical points in order to be recognized.
- Contours from two profiles must match globally in order for the profiles to be matched.

Topics for future research are:

- Combine the matching method described here with a method of matching large regions of constant curvature (which do not contain critical points).
- Find a way to do local matches on contours and handle contours that are not simple closed curves.
- Add domain-specific information to the matching process, for example color

or texture.

- Integrate the local matcher into a vision system.

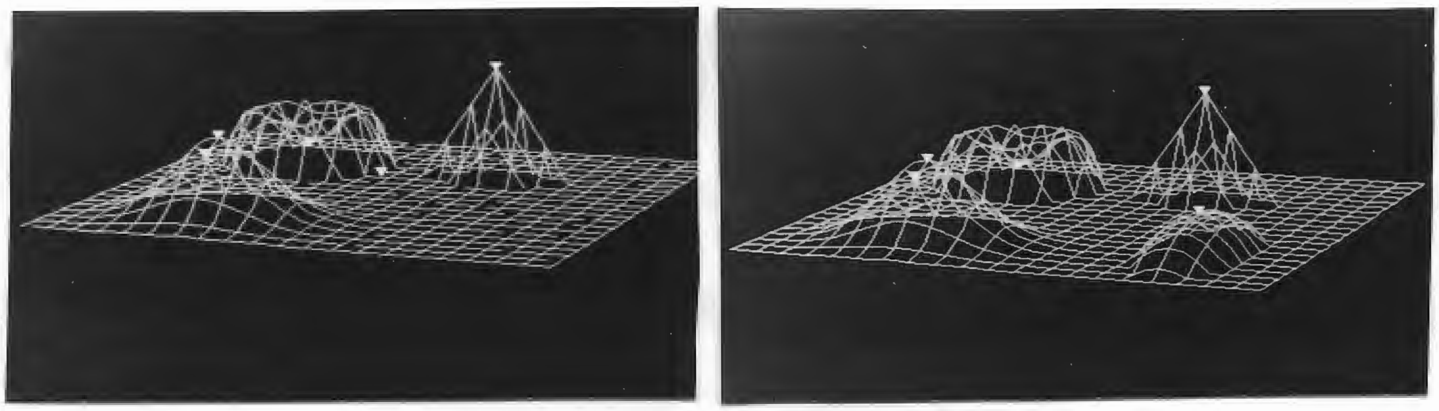

Figure 7-1: Surfaces with critical points
Adjacent lines in the picture differ by 1 in their x and y coordinates. Each square shown is actually divided into eight triangles. The critical points were computed using an AIWID value of 2, and a NBHRAD of 3.5. Critical points found by the program are shown marked with triangles.

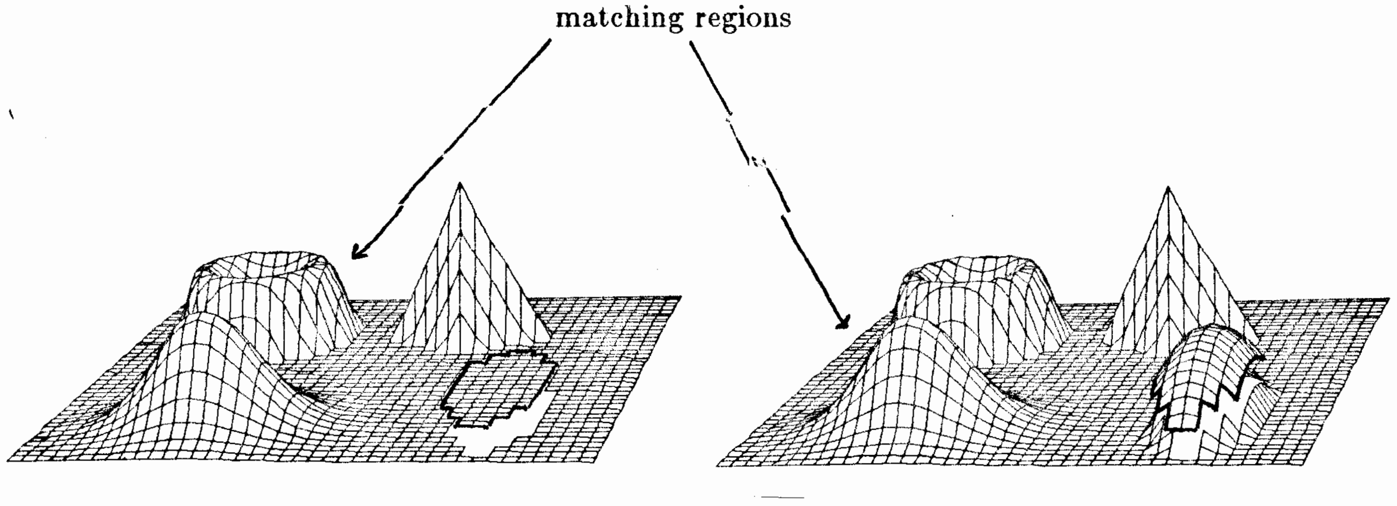

Figure 7-2: Surfaces with matching regions shown
The matching and nonmatching regions of the two surfaces are detached. The matching was done with a DMTOL of 5.0 and a NEARTOL of 0.5.

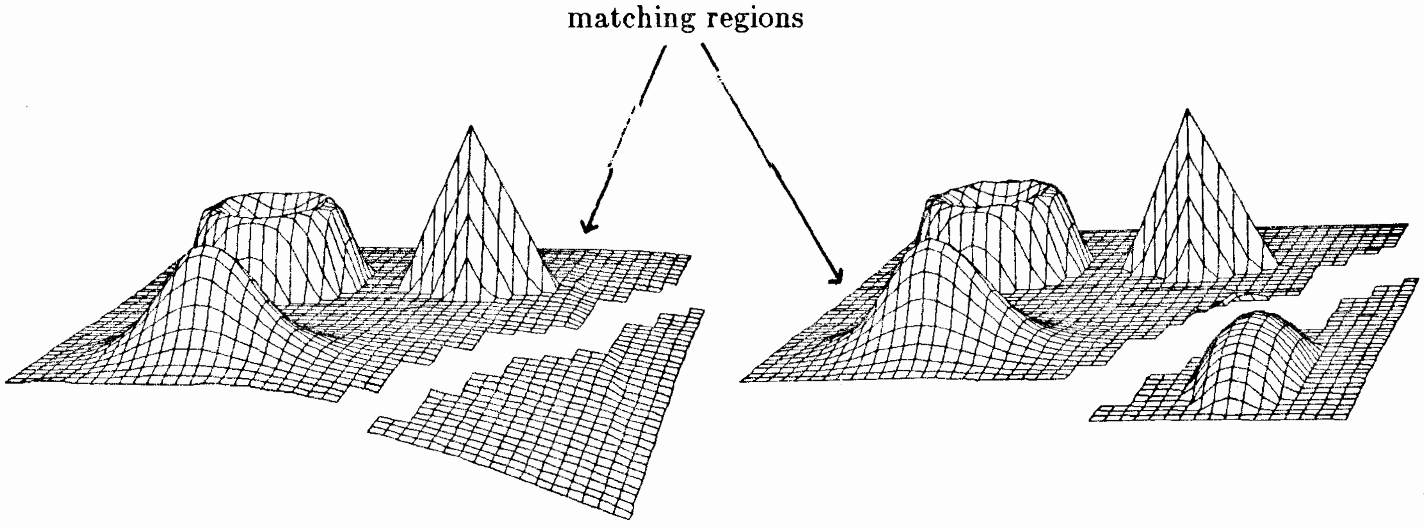

Figure 7-3: Effect on matching regions of bending first surface
A corner of the first surface has been bent down. Notice the larger nonmatching regions.

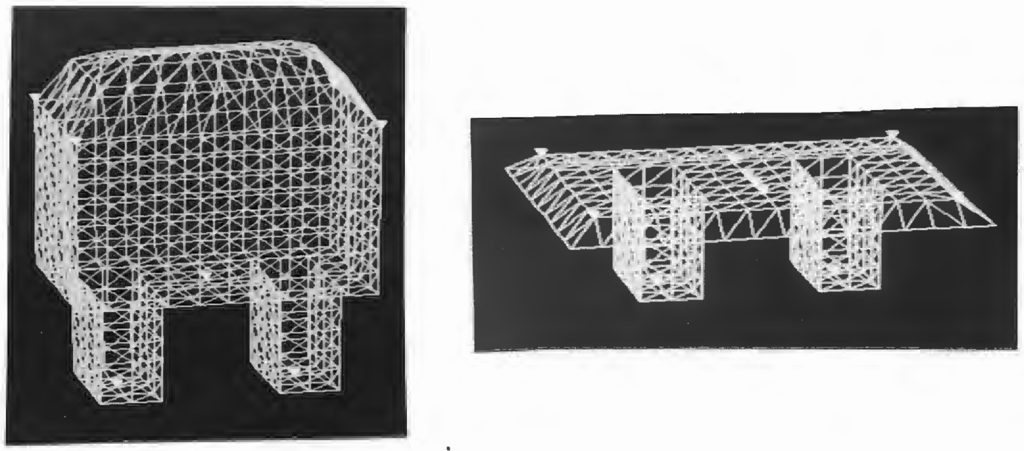

Figure 7-4: Plug and outlet with critical points
The triangles have sides of length 1, 1 and $\sqrt{2}$. The critical points were computed using an AIWID of 2 and a NBHRAD of 3.

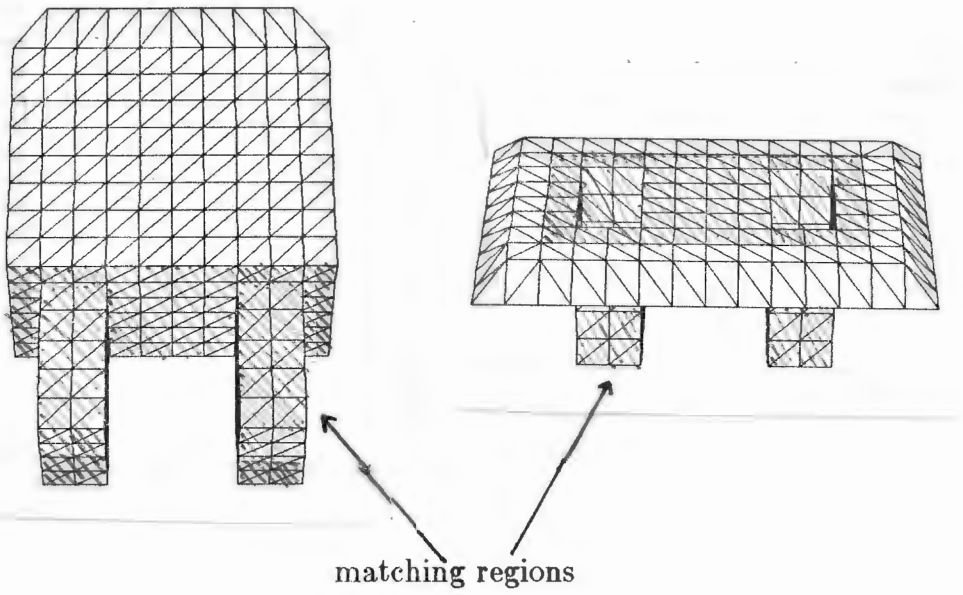

Figure 7-5: Plug and outlet matching regions
The matching regions are shown shaded. The matching was done with DMTOL and NEARTOL set to 0.1.

References

1. Badler, N., and R. Bajcsy. "Three-Dimensional Representations for Computer Graphics and Computer Vision." *Computer Graphics 12*, 3 (August 1978), 153-160.

2. Ballard, Dana H., and Christopher M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, 1982.

3. Dane, Clayton. *An Object-Centered Three-Dimensional Model Builder*. Ph.D. Th., University of Pennsylvania, May 1982.

4. Dane, Clayton, and Ruzena Bajcsy. A Three-Dimensional Object-Centered Model Builder. Proceedings of the Sixth International Conference on Pattern Recognition, International Association for Pattern Recognition, Munich, October, 1982, pp. 348-350.

5. Freeman, H., and L. Garder. "Apictorial Jigsaw Puzzles: The Computer Solution of a Problem in Pattern Recognition." *IEEE Transactions on Computers C-13*, 2 (April 1964), 118-127.

6. Freeman, Herbert. "Shape Description via the Use of Critical Points." *Pattern Recognition 10*, 3 (1978), 159-166.

7. Koshland, D. E., Jr. "Protein Shape and Biological Control." *Scientific American 229* (October 1973), 52-64.

8. Oshima, Masaki, and Yoshiaki Shirai. "Object Recognition Using Three-Dimensional Information." *IEEE Transactions on Pattern Recognition and Machine Intelligence PAMI-5*, 4 (July 1983), 353-361.

9. Radack, Gerald. "Jigsaw Puzzle Matching Using a Boundary-Centered Polar Encoding." *Computer Graphics and Image Processing 19*, 1 (May 1982), 1-17.

10. Watson, Layne T., and Linda G. Shapiro. "Identification of Space Curves from Two-Dimensional Perspective Views." *IEEE Transactions on Pattern Recognition and Machine Intelligence PAMI-4*, 5 (September 1982), 469-475.

11. Wodak, Shoshana J., and Joel Janin. "Computer Analysis of Protein-Protein Interaction." *Journal of Molecular Biology 124* (1978), 323-342.