Towards Corner Matching for 2D and 3D

T.J. Jankun-Kelly,[*] Bernd Hamann, Kenneth I. Joy
{kelly,hamann,joy}@cs.ucdavis.edu
University of California, Davis

Samuel P. Uselton
uselton@llnl.gov
Lawrence Livermore National Laboratory

Image resectioning is the process of taking 2D images of a 3D model and mapping those images back onto the model so the images can be viewed in 3-space. Consider photographic images of oil-slick tests on an airplane wing; image resectioning allows a scientist to take the images of the wing, shot at different locations and orientations, and visualize them as a continuous, 3-dimensional model. We aim to develop algorithms to automate the image resectioning problem with as little human interaction as possible. This abstract describes one of the steps toward that goal—finding points in the image and model that we can use for matching.

Finding Corner Points

Our algorithm focus on corner points as the key feature to find correspondences. Finding corner points in 2D images is a classical computer vision and image processing problem; we use standard techniques to find corners in images (see [2, 4, 1]). Given the 2D image, the algorithm produces a set of corner points from the decimated boundary (Figure 3) by merging points along the boundary until only lines with certain corner angles are left.

The second part of our algorithm calculates feature corners in the 3D model. The definition and extraction of corner/edge information in 3D triangulated surfaces is based on the method described in [5]. Starting with a triangular mesh (either provided directly or obtained as a result of a triangulation of an analytical surface), we define features of interest as follows:

Feature Edge. A feature edge is present between two adjacent triangles when the angle between the outward normals exceeds a given threshold value.

Boundary Edge. A boundary edge is an edge that belongs to only one triangle.

Corner Point. A corner point occurs at vertex where three or more feature edges meet. Optionally, any vertex of a boundary edge can be considered a corner point.

Figure 1 illustrates these definitions. For a given vertex v

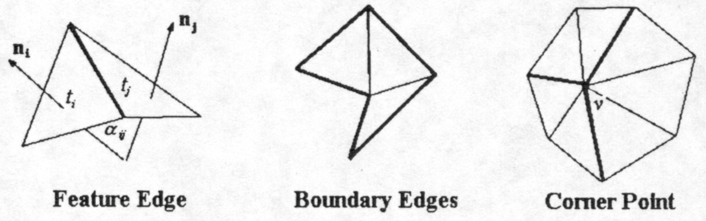


Figure 1: Feature edge, border edge, and corner point definitions.

in a mesh, we examine each edge emanating from v. If two triangles share the edge, we compare the angle between the outward, unit triangle normals; if the angle is greater than a specified "corner angle", the edge is added to a feature edge list for that vertex. After examining all edges, we declare v a corner point if and only if there are three or more feature edges emanating from v. This process is repeated for each vertex in the mesh until all corners (and, optionally, all feature and border edges) are extracted.

The algorithms we have presented are efficient. For the 2D case, once an initial boundary pixel is discovered, the algorithm described for extracting the boundary is linear in the number of boundary pixels, while line merging complexity is quadratic. For the 3D case, each vertex is examined only once and each edge is examined at most twice (once for each end-point).

Implementation and Analysis

We have implemented the corner extraction algorithm for both 2D and 3D in Java using Java3D. Figures 2, 3 and 4 demonstrate the corner finding algorithms. The model is a triangulation of a non-uniform rational B-Spline surface (NURBS) [3].

The 2D corner finding algorithm is sufficient for our needs. However, the 3D corner finder is not yet robust

[*]This was supported by NASA Ames Research Center through a summer internship in 1998 via NASA contract NAS2-14303. At UC Davis, this work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award), the Office of Naval Research under contract N00014-97-1-0222, the Army Research Office under contract ARO 36598-MA-RIP, the NASA Ames Research Center through an NRA award under contract NAG2-1216, the Lawrence Livermore National Laboratory through an ASCI ASAP Level-2 contract under W-7405-ENG-48 (and B335358, B347878), and the North Atlantic Treaty Organization (NATO) under contract CRG.971628 awarded to the University of California, Davis. We also acknowledge the support of Silicon Graphics, Inc., and thank the members of the Visualization Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis and the members of the Numerical Aerospace Simulation group (NAS) at NASA Ames.

enough for image resectioning. Though it finds corners easily for objects with sharp edges (i.e. cubes), it fails on objects with smoother variation. Even more important, the corners found by this method may have no correspondences to the points found by the 2D algorithm. For example, the corner points belonging to the nadir of the Figure 2 are not found by the 2D algorithm (Figure 3) as they are internal to the boundary. In addition, 3D corners that are obscured by faces or vertices lying in front of them are not found by the 2D algorithm since they are not visible from the camera position.

Conclusions

We have described a method for corner finding for automated image resectioning of multi-source data. Though we can automatically find corners in 2D images, classifying 3D corners is still problematic. Currently, our solution requires a user to intervene and position a given 3D model in such a way that our 2D algorithm can be used to find corners in projected image. If an efficient search of possible model orientations can be found, then an automated solution to our problem may be feasible. Using a 2D algorithm for both model projections and images has the advantage that the generated points should be similar enough to facilitate matching for projecting the images back into 3-space. The approach we have presented is a possible path toward more automation to deal with general multi-source data problems.

References

[1] Dana H. Ballard and Christoper M. Brown. *Computer Vision.* Prentice Hall, 1982.

[2] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.

[3] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press, 4th edition, 1997.

[4] Theo Pavlidis. *Algorithms for Graphics and Image Processing.* Computer Science Press, 1982.

[5] W. Schroeder, J. Zarge, and W. Lorensen. Decimation of triangle meshes. *Computer Graphics (SIGGRAPH '92)*, 26(2), August 1992.

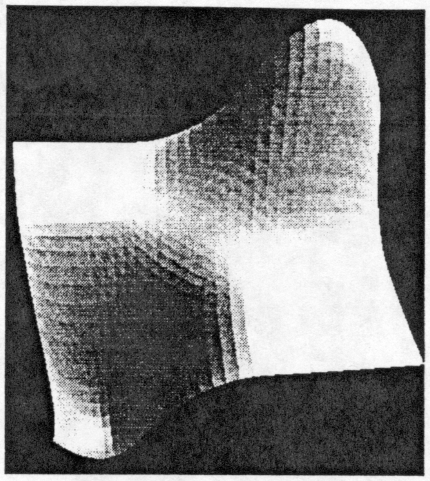


Figure 2: Original surface.

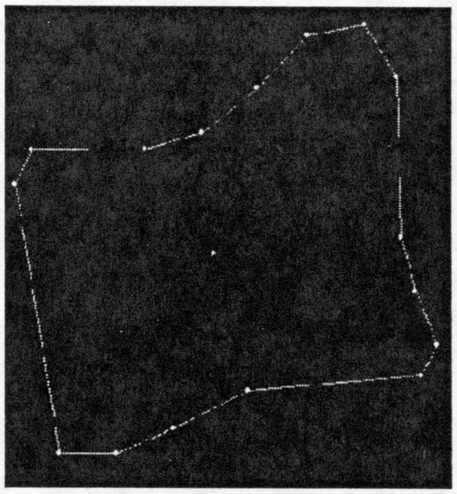


Figure 3: The 16 extracted 2D corners (white dots) and boundary (cyan/grey line).

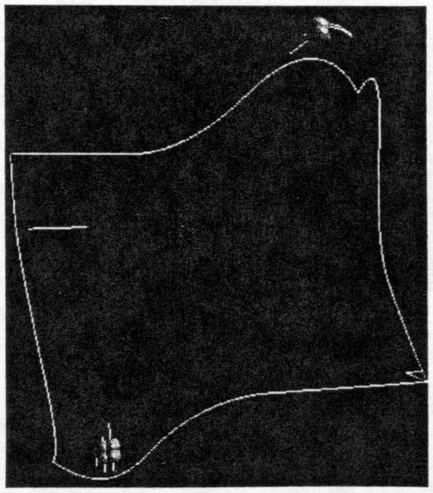
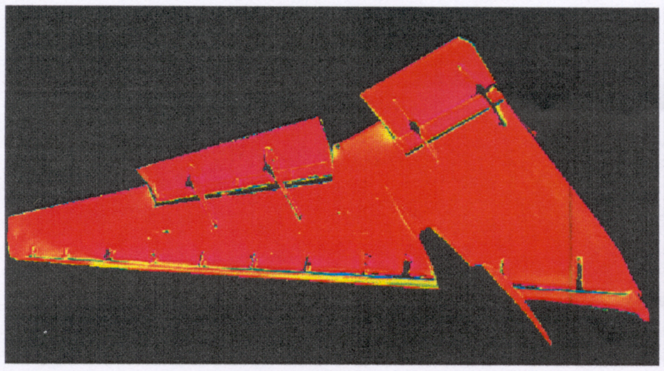
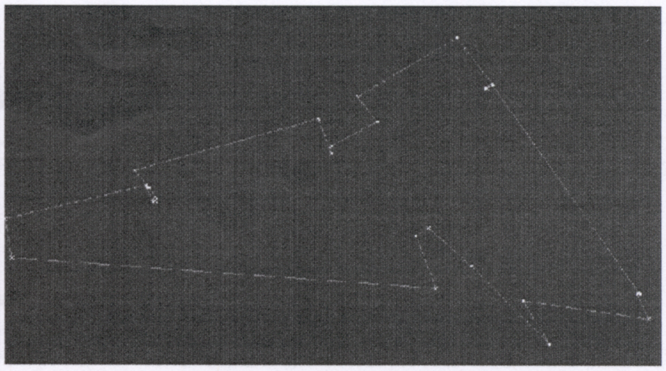


Figure 4: The eight extracted 3D corners (grey spheres) and feature edges (white lines).

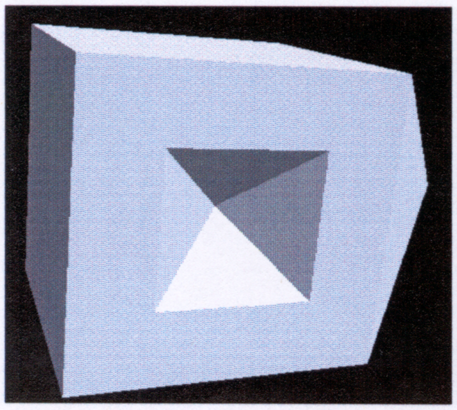


(a) Original PSP digital image.

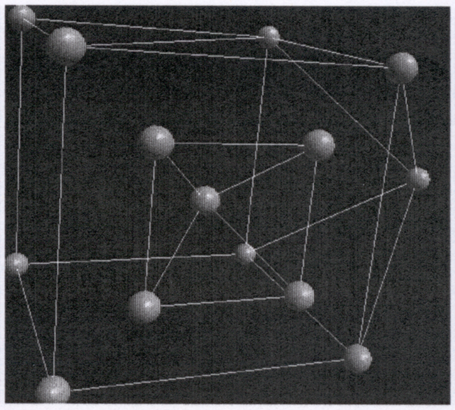


(b) Extracted corners obtained after edge decimation.

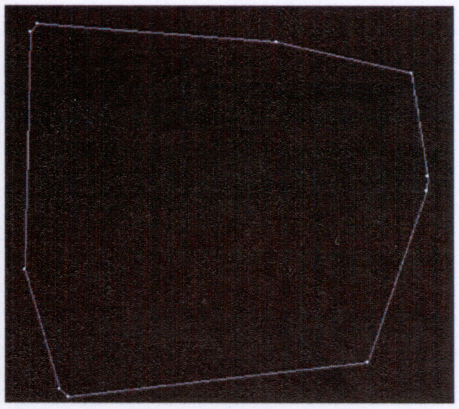
Figure 4: Results of 2D corner extraction algorithm for wind tunnel data.



(a) Original triangulated surface model.

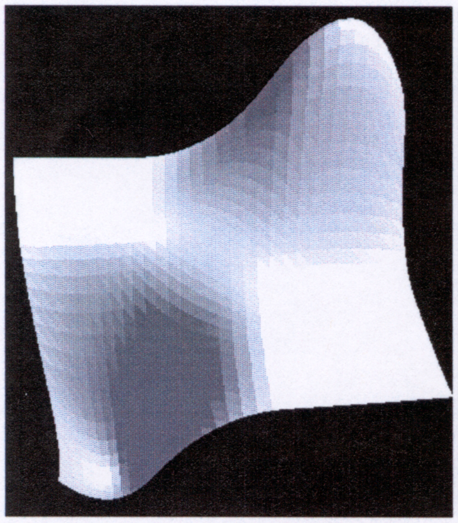


(b) Extracted 3D corners. 16 corners were found using a 45 degree corner angle.

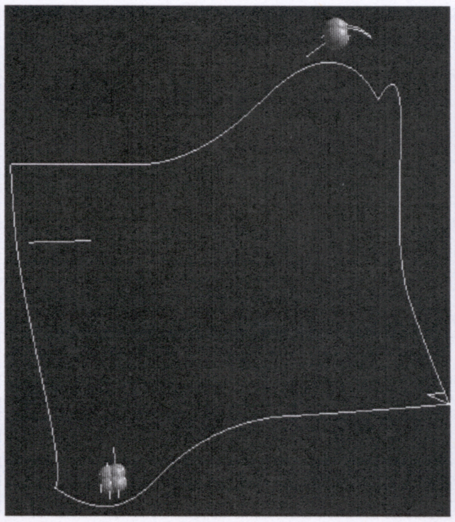


(c) Extracted 2D corners from image (a). 10 corners were found with a 135 degree corner angle.

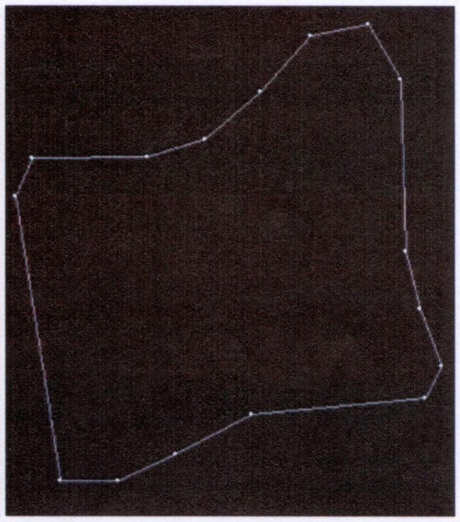
Figure 5: Results of 3D and 2D corner extraction for triangulated surface model.



(a) Original image of triangulated NURBS surface.



(b) Extracted 3D corners. Eight corners were found with a corner angle of 23 degrees.



(c) Extracted 2D corners from image (a). 16 corners were found with a 135 degree corner angle.

Figure 6: Corner extraction for a parametric NURBS surface.