THREE DIMENSIONAL DISPLAY OF OBJECTS FROM PLANAR CONTOURS

J. Lawrence Paul, Department of Pathology, University of British Columbia

Keywords: 3-D reconstruction, computer graphics

The problem is: given a set of object contours which describe the intersections of an object with a series of equi-distant parallel planes, to display a "solid" representation of the object. There are two major approaches to this problem: 1) Fuchs et al. (1977), working with pairs of contours, represented the surface between the contours with triangular "tiles" such that the surface had the minimum possible area. The tiles are then displayed using standard computer graphics techniques. The major drawbacks to this scheme are: First, it is complicated and slow; second, and more seriously, only bodies with a single closed contour at every intersection plane can be reconstructed. 2) Herman and Liu (1979) and Udupa (1981) try to display every point of the body as a cube. The cube covers a number of points of the display area, and is oriented according to the viewing position. The points of the cube are displayed if they are closer to the viewer than any other point of any other cube. This requires keeping two arrays the size of the display, one to store the distance to the nearest point, and one to record the intensity. The resulting image is then blurred to produce a more pleasing image. This technique works very quickly, but has a number of drawbacks: i) Each object point is a cube, which means that the distance between sections has to be equal to the horizontal/vertical distance between points of the contour. ii) In order to get the best images, the object must be rotated such that three sides of each object cube are clearly visible. This limits possible viewing angles. iii) The object points must be scarcer than the display points, since each object cube occupies a number of display pixels. This limits the object resolution. iv) The postfiltering of the image is a bit clumsy.

The method we developed is based on the latter technique. It too tries to display each point within a contour, using a distance array to determine whether a point is visible or not. The main differences between our procedure and the previous are: the resolution of the contours is changed to match that of the display image; each object point is represented as a point or line in the image plane, rather than a cube; the displayed image is determined from the distance array itself.

The contour points A,B,C are mapped into theoretical image points A',B',C' (Figs. 1,2), where the coordinates of A,B,C are integers, those of A',B',C' are reals. The coordinates include image location and height. The thickness between sections, Δz, is also represented in Fig. 2 as $\Delta z'$. The theoretical image points A',B',C' must be represented in the actual output image as points A*,B*,C*, where the coordinates of these points are integers. In Figs. 3 and 4, the nearest neighbour is used to determine the integer values. Depending on the size of the output image grid and the location of the coordinates, adjacent points i) may be interpolated between, eg. between A*,B* of Fig. 3 ii) may coincide, eg. A*,B* of Fig. 4, or iii) may be 8-neighbours, eg. B*,C* of Fig. 4. Thus we are able to convert a closed contour in the object domain to a closed contour at the output display resolution, each point having a height associated with it.

With this new closed contour, we can find left/right boundaries of object sections for each row of the output image. The information is used to update the distance array. For each point between these boundaries, we interpolate the height. If the point is closer to the viewer than the previous point, we update the height stored in the distance array at that location. We perform the same height test for each point on the 3-D $\Delta z'$ vector originating at that point.

We thus create a 2-D array of the same

dimensions at the output display image, containing heights. This is converted to an image by calculating the local surface direction at every point of the image, and calculating the brightness as

$$b(x,y) = \cos O(x,y) \quad d(x,y)$$

where x,y are the coordinates of the point, O(x,y) is the angle between the surface normal at (x,y) and the illuminatuon direction, and d(x,y) is the given height at (x,y). The local surface direction is calculated by fitting a plane onto the 3x3 points centered at (x,y). This also slightly blurs the surface, making it smoother.

Figures 5,6,7 and 8 are examples produced by this program. These images are about 128 x 120 pixels, of 13 grey levels, displayed on a DEC VT-100 terminal upgraded with Selanar Corp.'s Graphics 100 system. The contours required to produce Figs. 5,6 and 7 were computer generated as 20 equidistant planes through hemispheres of radius 10,50 and 100 respectively. The intervals between the sections have been exaggerated in order that the detail may be more visible, resulting in a hemi-ellipsoid being displayed. In Fig. 5, the object contours are much coarser than the display contour, and the interpolated points, similar to Fig. 3, are clearly visible. In Fig. 6, the object contours are about the same resolution as the display. In Fig. 7, the object contours have much higher re solution than the display. Fig. 8 is a torus produced from a series of 13 hand-drawn contours. The contours start as a single curve, divide into two, then unite into a single curve again. Of these four examples, the method of Fuchs et al could not reconstruct the torus, and the method of Herman and Liu would be ineffective for Figures 6 and 7.

This technique is intermediate in complexity between the two previous by mentioned techniques, and has almost none of the drawbacks associated with them. The exception is that there are some viewing angle restrictions. Looking at an object head on will display the object as a series of flat contours, each contour a different density from the next, with sharply sloped sides between the contours. Looking at the object from 90° will show a series of parallel sections. In either case, the true shape of the object is difficult to determine.

Acknowledgement: This work was funded by British Columbia Health Care Research Foundation grant # 75(81-2).

References

Fuchs, H., Kedem, Z.M. and Uselton, S.P. (1977). Optimal surface reconstruction from planar contours. Comm ACM 20, 693-702.

Herman, G.T. and Liu, H.K (1979). Three-dimensional display of human organs from computed tomograms, Computer Graphics and Image Processing 9, 1-21.

Udupa, J.K. (1982). Interactive segmentation and boundary surface formation for 3-d digital images, Computer Graphics and Image Processing 18, 213-235.

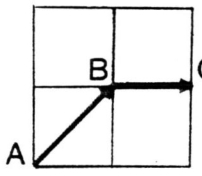

Fig. 1

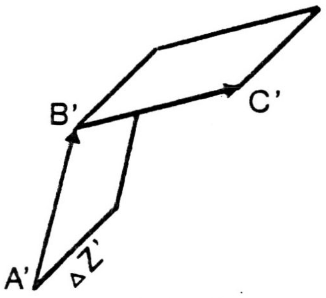

Fig. 2

Fig. 1. Three points on object contour.
Fig. 2. Theoretical view of 3 points in desired orientation.

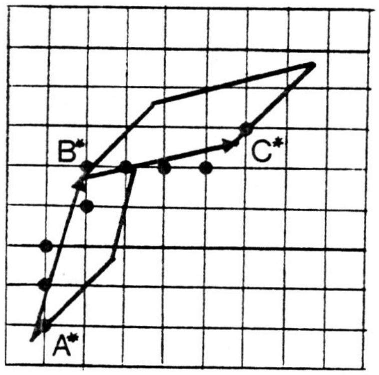

Fig. 3

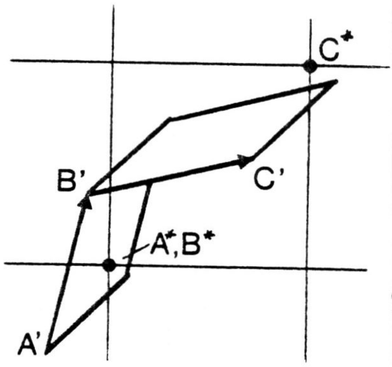

Fig. 4

Figures 3,4. Points A*,B*,C* are nearest grid points to A',B',C'. On fine grid (Fig. 3), points between A* and B* are interpolated. On coarse grid (Fig. 4), A*, B* coincide, and A*,C* and B*,C* are 8-neighbours.

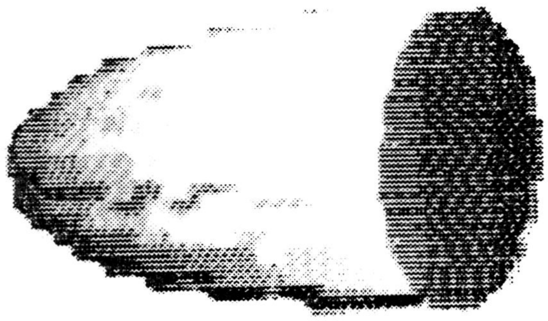

Fig. 5. Hemi-ellipsoid, 20 sections, r=10.

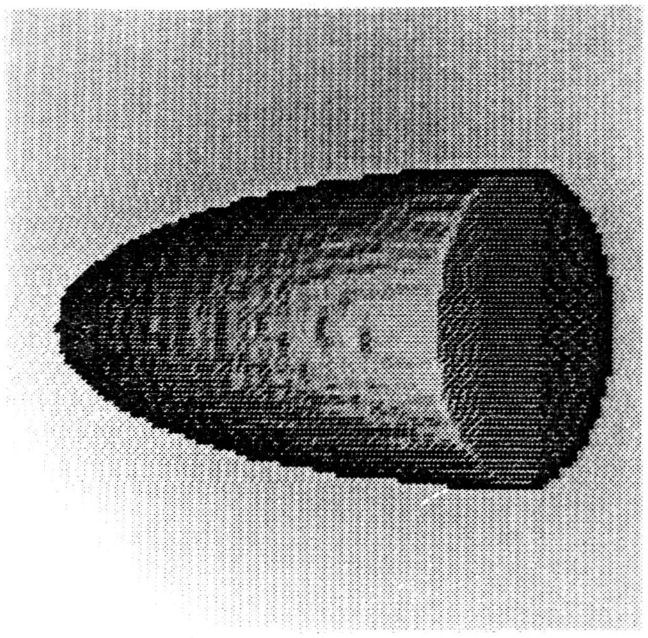

Fig. 6. Hemi-ellipsoid, 20 sections, r=50

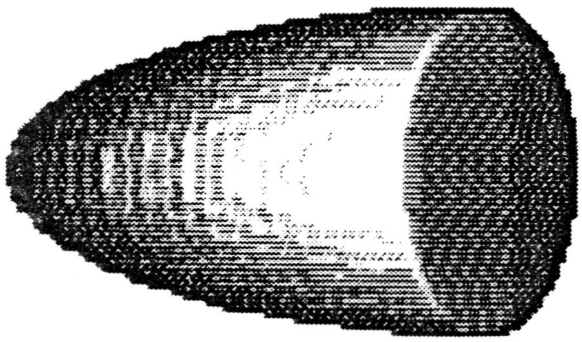

Fig. 7. Hemi-ellipsoid, 20 sections, r=100.

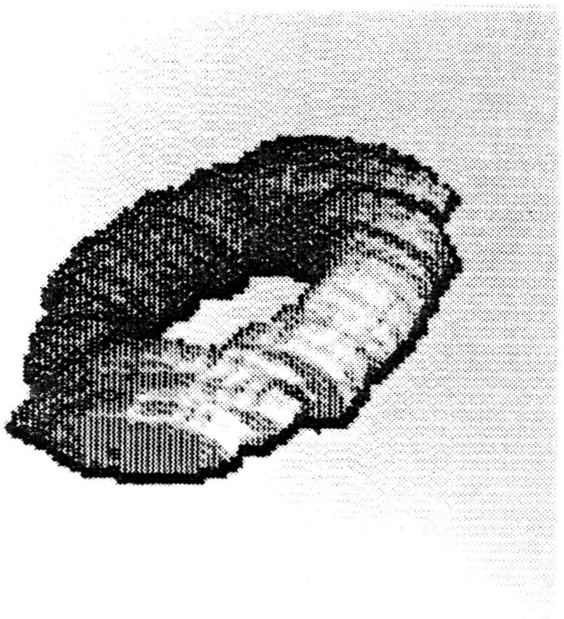

Fig. 8. Torus, 13 sections.