The Binford-Horn LINEFINDER

VISION FLASH 16

B.K.P. Horn

Massachusetts Institute of Technology

Artificial Intelligence Laboratory

Vision Group

July 1971

ABSTRACT

This paper briefly describes the processing performed in the course of producing a line drawing from vidisector information.

THE LINE-FINDER:

Programs able to produce line-drawings of images of single convex polyhedra have existed for several years. It was thought that it should be easy to generalize these to handle more complex scenes. Only recently, however, has it become possible to reliably produce line-drawings of images of sets of polyhedra. There are two main problems. Firstly, the images of sets of polyhedra are not as simple as one might at first expect. Secondly, current scene-analysis programs demand a line-drawing complete with well-defined vertices.

One might conjecture that the image of a set of polyhedra ought to consist of polygonal areas of more or less uniform intensity separated by step-transitions in intensity at the lines corresponding to the projections off the edges of the objects. Due to mutual illumination, scattering, smudges, shadows, incomplete opacity and a number of instrument defects <1> this is not the case and usually the variations in intensity within one region (corresponding to a face or the visible portion of a face of an object) is often larger than the variation between adjacent polygons. Furthermore only the most obvious edges are associated with anything like a step-like transition of intensity, other transitions being roof-shaped or flat except for a small peak right on the

edge due to an edge-effect<2>. The most obvious image degrading effect of the sensing device is noise. In our case this was of the order of 1 % of the signal, there being little point in improving on this, since the surface visual noise is about this magnitude even for clean, evenly painted polyhedra.

Line-finders can be classified according to whether they use a raster scan or a scan pattern which follows lines. The first method is convenient from the point of view of asynchronously reading intensity values and programming simplicity. In some cases the image sensing device forces the use of a raster scan, particularly with integrating devices. Line-following on the other hand can be made to be more sensitive and accurate at the price of program complexity. Line-finders can also be classified according to whether the predicate applied to each small region to establish whether it belongs to an edge or not is linear or not. If the visual noise was spatially independent, a case could be made for an optimal linear predicate. This however is not the case, since smudges for example have a definitly non-random spatial distribution. Non-linear methods, while more complex have a clear advantage on real images.

EDGE-MARKING:

The line-finder here described consists of an edge-marker due to T.Binford and a line-drawer due to B.K.P.Horn. Input is obtained from a random access image disector camera. The edge-marker is a non-linear parallel line-follower. Several lines are followed simultaneously while the image is scanned in a raster fashion, thus combining the good features of raster-scan with those of line-following. The separation between scan-lines is larger than the spacing of points along one line and three lines are considered at any one time. The intensities are correlated with the three most common intensity transitions as described above; step, roof and peak. Up to here processing is linear. If any value exceeds a threshold calculated from the known signal-to-noise ratio, it is checked for a local maximum both along the line and in angular orientation.

Once declared a likely edge-point or feature point, it is checked whether it could form the continuation of a line already being followed. The test involves a check on proximity and a match of attributes such as direction and size of the intensity step. If it cannot form the continuation of an existing line and it is particularly strong and not too close to another line, it will be used as the starting point of a new line to be followed.

Otherwise it is ignored. So we have a list of lists of likely edge-points. The head of each such list contains various attributes such as the probability that the line is purely a noise effect.

If a line cannot be continued in this manner it is discarded unless it exceeds some minimal size. Note that there are certain adjacency effects; lines cannot encroach too closely. For this and other reasons connected with the correlation process, lines are usually not followed right up to the vertices. One scan consisting of succesive hoizontal lines picks up all edges within about 50 degrees of vertical. A second scan consisting of vertical lines does the same for edges within about 50 degrees of horizontal. In total about a million points are scanned for a typical scene and the whole process takes a few minutes (The scanning and first levels of correlation can be performed by an attached processor for extra speed).

We end up with a list of lists of tenative edge-points, also called feature points of inhomogeneous points. Some of these lists will represent more than one edge and some edges will beprepresented by more than one list. The lists will not usually contain points very close to vertices, except two-edge vertices.

Up to this stage not very many heuristics have to be invoked and consequently the possibility of corruption of the data is small. Numerous line-finders have been developed to this level, few however produce as clean a set of tentative edge-points as Binford's program. Very few line-finders have proceeded beyond this level of competence to create the cleaned up line-drawing demanded by current higher-level scene-analysis programs. In part this is often due to poor edge-marking, but more often the unexpected difficulties encountered in what at first sight seems the simple process of forcing this data into the form of a line-drawing with well-defined vertices.

GENERATING LINES:

Since some lists may contain feature points of more than one edge, they have to be segmented. This is done recursively at the point of maximum distance from the line joining the end-points. A little subtlety is needed to cope with portions that are parallel to this line. Once segmented, least-squares lines are fitted to the lists. The partial results of this first fit are stored with the line to allow combining lines later on without loss of accuracy. The first lines to be combined are those that appear to overlap. A number of tests are applied to avoid combining unrelated lines. These tests check on proximity, relative

angle, perpendicular distance of the end-points from the combined line and so on. In a similar manner co-linear lines are combined if the gap between them is relatively small. Any short lines remaining at this stage are discarded.

The line-drawing is now fairly complete, lacking only vertices. The lines have been distorted very little in this process unless unrelated lines were combined. The more difficult and less conservative part is yet to come.

CONCOCTING VERTICES:

Some vertices are clearly indicated from the close convergence of lines. A first estimate of the location of the vertex is made from the centre of gravity of the end-points being considered (unless there are only two, in which case they are intersected). A search is then made for all lines whose extension passes close to this point and with one end near this point. The point with least-squares perpendicular distance from these lines is then declared the vertex and the lines connected to it.

At the next step, a search is made for the lines with end-points close to a line. These give rise to T-joints, while vertices close to existing lines are potential K-joints. Finally crossing lines give rise to X-joints. Numerous

heuristics inform this process; lines already connected at one end are treated preferentially to those still free on both ends for example. All vertices are now established and an attempt is made to extend unattached lines. Following this vertices which are very close together are conglomerated. The data-structure is then re-ordered for optimal plotting and ready for output in LISP-readable format.

The heuristics used depend on certain tolerances which are intially calculated in terms of the line-scan interval and the known resolution of the imaging device. These factors could be "tuned" to improve the performance and accuracy in critical cases. This is probably not worth the effort, the time would be better spent on designing new heuristics to direct the vertex creation phase. A large protion of this assembler language program is concerned with debugging and performing the required list-processing. To a large degree the various heuristics developed empirically and this program could not have been written without the aid of a display device (Digital Equipment Company 340) and a very effective time-sharing system (The Incompatable Time-Sharing System).

This second phase of the program makes use of four overlaying rectangular grids covering the image to help in determining proximity. This method is sometimes referred to as multi-entry

coding, since each point is entered into four buckets, and each line will appear in many. This makes for high speed (a few seconds) despite the need for a large number of sucessive applications of various heuristics to the whole data-structure.

PERFORMANCE:

No idea of how to produce a good line-drawing is any good unless it is embodied in a demonstratable program. This program produces excellent line-drawing of simple scenes. In more complicated scenes a number of short-comings can be observed. The simplest and easiest to deal with is the absence or incompleteness of some of the lines, due to lack of contrast between adjacent faces. Occasionally extra lines will be produced due to shadows, smudges and noise. There is a trade-off between these two effects, and since present scene-analysis programs can handle missing lines better than extra lines, the threshold is set to favour the production of the former. Occasionally too, a section of the line-drawing will be garbled, usually because of the combination of two unrelated vertices. This causes some distortion of the lines and at times makes the line-drawing uninterpretable. The incidence of all of these effects can be considerably reduced by using a finer raster at the cost of an increase in scan and calculation time.

For easy scenes the process is quite repeatable, coordinates of vertices always being very close on successive trials. In complicated scenes different lines may be missed at different times.

COMMENTS:

Some fof the ways in which images of sets of polyhedra differ from our simple model of equal intensity polygonal areas have important implications in other areas. Mutual illumination for example is going to make color-constancy a non-trivial problem. Further it should be noted that some of the edges missed by this program will also be missed by line-verifiers. Fortunately the better line-proposers are very conservative and hardly ever propose a line were there is none. One could tie the liberal proposers to conservative line-verifiers and let the conservative proposers work blind - that is just accept their judgment.

The only features that can be reliably determined from a corrupted image are those with significant spacial extension. Without such extension we cannot apply the integrative processes necessary to collect evidence for the existense of the feature. Vertices for this reason cannot be considered primitive elements of an image, but exist only as the intersection

of lines which have been shown to intersect by the higher-level scene-analysis programs. Letting the line-drawing program establish vertices throws out accuracy, because it may join up unrelated vertices.This is clearly a place were increased communication between the line-drawing program and the scene-analysis programs would pay off.

REFERENCES

(1) Berthold K. P. Horn, "The Image Dissector 'Eyes'" Artificial Intelligence Memo 178 (Cambridge, Mass.: Artificial Intelligence Laboratory, M.I.T., August 1969).

(2) Annette Herskovits and Thomas O. Binford, "On Boundary Detection," Artificial Intelligence Memo 183 (Cambridge, Mass.: Artificial Intelligence Laboratory, M.I.T., July 1970).