Automatic Extraction of Invariant Features for Object Recognition

Ellen L. Walker and Kenji Okuma
Mathematics & Computer Science Department
Hiram College
Hiram, OH 44234
walkerel@hiram.edu
http://hirame.hiram.edu/~walkerel

Abstract

A powerful technique for three-dimensional object recognition has been the use of geometric invariants: measurable relationships between geometric objects that are invariant to transformations such as projection. Because of the invariance, these measurements will be the same whether measured on the actual three-dimensional object, or in the image. Therefore, objects in the image can be recognized if the same invariant can be found. In this paper, we investigate the automatic extraction of cross-ratio invariant features for object recognition. We show that clustering is a promising technique in this extraction, because it reduces the dependency on tuning parameters in the image processing phase.

1. Introduction

Geometric invariants are measurable relationships between geometric objects that are invariant to transformations such as projection. Because of the invariance, these measurements will be the same whether measured on a three-dimensional object, or in an image of that object. Therefore, objects can be recognized in an image if their known invariants can be found [4, 8]. Generally, invariants are based on collections of feature points extracted from an image, such as corners, line intersections, or circle centers.

One invariant that has been widely used in computer vision is the *cross-ratio*, based on the distances between four collinear points [4]. Labeling the points as A, B, C and D, the cross ratio is simply (AB)(CD) / (AC)(BD). If the same four collinear points can be found in different images of the same object, their ratios will be the same as the cross ratio measured. As an example, Figure 1 shows an image of three floppy disks, and Figure 2 shows the edges in the image, with collinear point sets for two cross-ratio invariants identified. In this image, point features were chosen at line segment intersections (image corners), as well as at intersections between line segments and circles.

The goal of this work is to develop techniques for automatically extracting invariant features for recognition of objects from images of those objects. Invariants that are automatically extracted from a set of training images of a given object can then be used to recognize that object in complex images. Using geometric invariants, we will extend our previous work on two-dimensional object recognition by clustering [5, 6, 7] to three dimensions.

Figure 1: Floppy Disks at Different Orientations

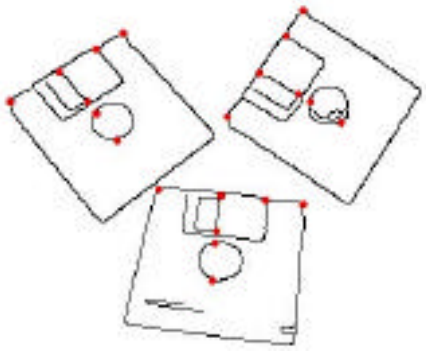

Figure 2: Feature points for computing cross ratios

All feature extraction must begin with image processing. Problems with traditional image processing techniques include both sensitivity to noise and occlusion, and dependence on parameters such as smoothing coefficients and thresholds that greatly affect the extraction outcome.

Tuning these parameters is a crucial and difficult step in ensuring reliable feature extraction. If the parameters are

too restrictive, then important image features will be missed, causing the objects that contain those features to also be missed. To ensure that the necessary features are extracted, the parameters must be less restrictive, but then many additional points that are not directly related to the desired features are also extracted. In the case of invariant-based object recognition, this means that many extra invariants, not related to any objects in the image, will be computed.

In this work, we hope to show how fuzzy clustering can reduce the dependency of feature detection on arbitrary tuning parameters, making it more robust and automatic. We specifically address the extraction of invariant point sets for cross ratios.

2. Feature Extraction Overview

The process of feature extraction has three main parts: image preprocessing, line detection, and invariant extraction. Image preprocessing reduces the amount of data in the image by locating significant image boundaries. Line detection determines the relevant straight lines formed by these boundaries, and invariant extraction finds sequences of feature points along the extracted lines.

2.1 Image Preprorocessing

The goal of image preprocessing is to generate simple line-drawing images such as the one in Figure 2. Our implementation uses the Canny edge detector [1] for this extraction. While the extracted edges are generally good, they include many short, incorrect, (noise) edges as well as the correct boundaries. These can be seen in the thresholded edge result in Figure 3. Noise edges are removed through a two-step process: first, connected components are extracted from the thresholded edge image, and then the smallest components, those with the fewest edge pixels, were eliminated. After noise removal, the resulting edges are quite clean, as can be seen by comparing the edges in Figure 4 with the middle disk in Figure 3.

The parameters for edge detection and the thresholds for edges and significant components were chosen empirically, as is common practice. Since all of the image preprocessing steps operate on a pixel-by-pixel basis, they have been carried out on the three-disk image. Line detection and invariant extraction, however, are carried out on single-disk subimages.

2.2 Line Detection

To extract collinear point sets, one must first extract significant straight lines from the image. These lines correspond to major linear features such as the edges of the disks and of the metal slider. We chose the Hough transform[3] for line detection, because it operates globally on the image rather than locally, and prominent long lines are the best basis for cross ratio invariants. The Hough transform works by allowing each edge point in the image

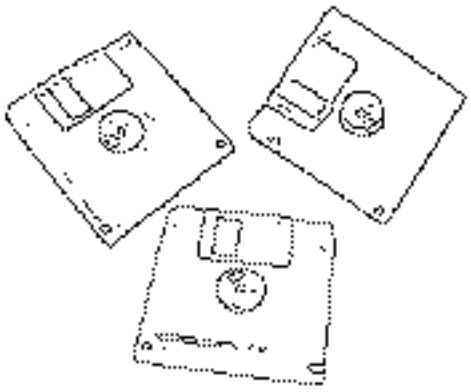

Figure 3: Edges after Thresholding

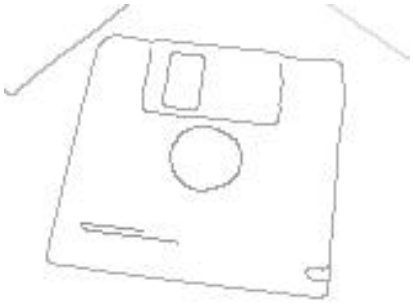

Figure 4: Final Edges, Middle Disk

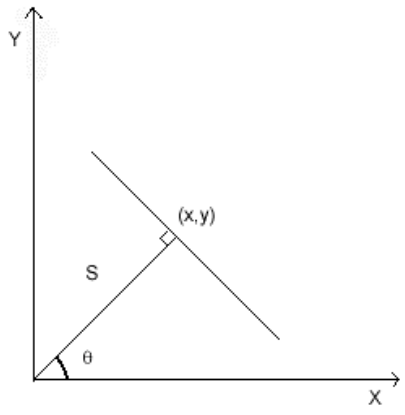

Figure 5: Parameter Space for a Line

to vote for all lines that pass through the point, and then selecting the lines with the most votes.

Lines are represented as points in a *parameter space*, whose axes are S and θ. Each S , θ pair represents a unique line with the equation $S = x \cos \theta + y \sin \theta$. (See Figure 5).

The parameter space is discretized; in our case, both the range of distances (r) and angles (θ) are divided into 360 equal subranges. For each point, for each angle, the distance from that point to the origin is determined, and the parameter table for that distance/angle pair is incremented. After all edge points are considered, the peaks in the parameter space indicate which lines are supported by the most points from the image. Figure 6 shows the parameter space for the middle disk, whose edges are shown in Figure 4. The darker spots indicate peaks that correspond to image edges. Finally, Figure 7 shows the edges that correspond to values greater than 20 in the parameter space. Darker edges have more support than lighter edges.

The discretization of the parameter space is one of the hidden tuning parameters of the Hough transform. With non-overlapping bins, all lines within a bin are reported as the single line at the center of the bin. When the bins are too large, the reported lines do not overlap the actual lines in the image. For precision, we must err on the side of bins that are too small.

When small bins are used, the result is another problem. Because the line segments are digitized, not every point on a segment yields exactly the same line. Thus, strong lines tend to respond to several neigboring bins. This explains the thick bundles of lines in Figure 7. Any threshold large enough to pick up the desired lines will also pick up many extraneous lines due to this effect.

What is needed is a way to recognize one strong line for each bundle, or cluster, in the parameter space. Thus, we add a clustering step after thresholding using a relatively low threshold.

For these experiments, we use K-Means clustering[2], setting K to 25 (approximately twice as large as the number of desired line segments). The cluster centers are initialized by evenly partitioning the sorted data points. The effect of this method is to distribute the initial centers roughly evenly across the occupied portion of the parameter space. Each cluster center is calculated as the sum of its (S,θ) vectors, weighted according to the number of votes in the parameter space. After cluster centers are calculated, each point is moved to the closest cluster and the centers are recalculated. The Euclidean geometric distance ($S^2+\theta^2$) is used to determine the closest cluster. After each iteration, any empty clusters were dropped. The clustering ends when no points change clusters during an iteration.

Clustering cuts down the number of extraneous segments, as is seen in Figure 8. Comparing Figure 8 to the original image in Figure 4, the four boundary edges of the disk can be seen among the strongest (darkest) lines detected. The dark segment cutting the corner at the upper left corresponds to the edge of the left disk that was not cropped out of this image. (This is a good example of unavoidable

Figure 6: Parameter Space f or Figure 4

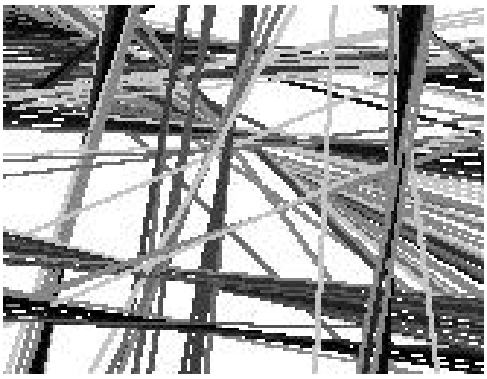

Figure 7: Strongest Lines from Figure 6

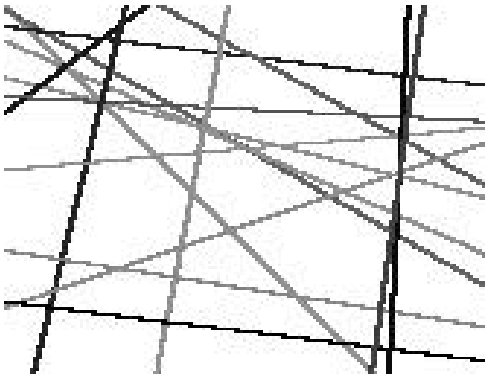

Figure 8: Lines After Clustering (K=25)

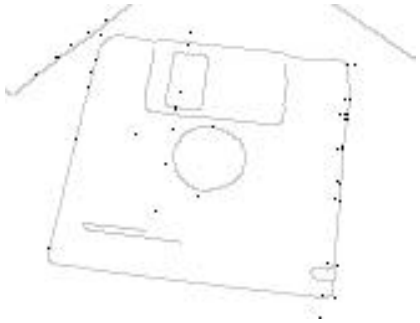

Figure 9: Invariant Points (Middle Disk)

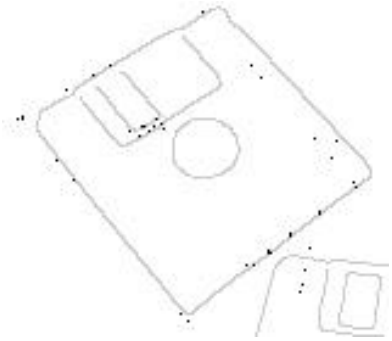

Figure 10: Invariant Points (Left disk)

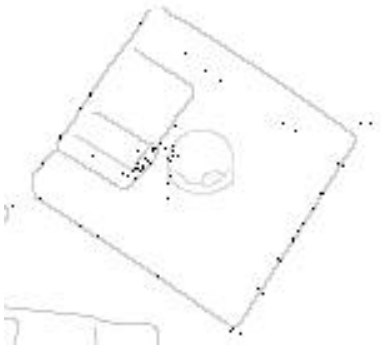

Figure 11: Invariant Points (Right Disk)

background clutter). Some of the diagonal edges arise from accidental alignments of the center circle with portions of other edges.

2.3. Extracting Invariants

Each detected edge is a possible line along which cross ratios can be computed. Currently, we use a brute-force method for finding point sets. In general, every intersection point between every pair of lines is computed, and any line that has four or more intersections with other lines yields one or more invariants.

In reality, not every intersection is considered. Because of numerical instability, it is difficult to accurately determine the intersection of near-parallel lines. Thus, intersections within 30 degrees of parallel are ignored. Intersections that fall outside the boundaries of the image are also ignored. Finally, we believe that most useful intersections will be on or near image edges. Therefore, intersections that do not have an image edge within a threshold distance (currently 10 pixels) are ignored. (These parameters were determined empirically on the center disk subimage, and used without modification on the other subimages.)

Figures 9, 10, and 11 show the points extracted for each of the three disks, overlaid on the original image. In Figures 10 and 11, the points are not aligned with the right edge of the disk. This is a negative effect of clustering; the lines from the disk's border and the edge of the slider were averaged together to get a parallel line between the two. Even with these problems, the point sets look good, qualitatively. The boundary edges of the disk seem to have similar selected points, for example the five points across the top (with the slider) boundary of the disks in Figures 10 and 11, and the points down each of the sides of the disks. These collections of similar points should yield similar invariants.

For every set of four points on an extracted line segment, one invariant is calculated. Even though the number of intersections seems limited, the number of invariants is still large. For the middle disk (Figure 9), there are 518 invariants. While some of those are useful in detecting disks, many of them are spurious. The question that remains to be answered is: which of these invariants can be relied upon to recognize floppy disk objects?

3. Choosing Relevant Invariants

Relevant invariants are those that are apparent in all images of a given object, but not in images of other objects. Our original plan was to cluster the invariants from multiple images of the same object to find large clusters that contained feature sets from all of the objects. Unfortunately, as Figure 12 shows, the invariants detected from each object cover a nearly continuous range of values (even when a log scale is used) rather than discrete subsets that can be easily clustered. Figure 13 shows a zoomed-in view of the first 100 values, which range only from 1 to 1.14, again with few significant gaps in value. When clustering is attempted, these insufficient gaps make the result overly dependent on the seed values.

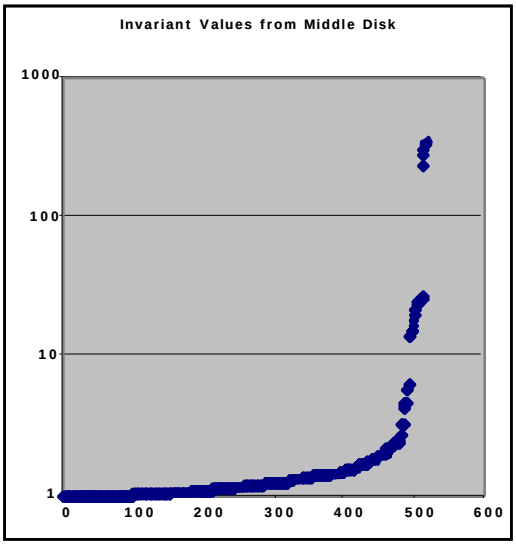

Figure 12: Sorted Invariant Values (Log Scale)

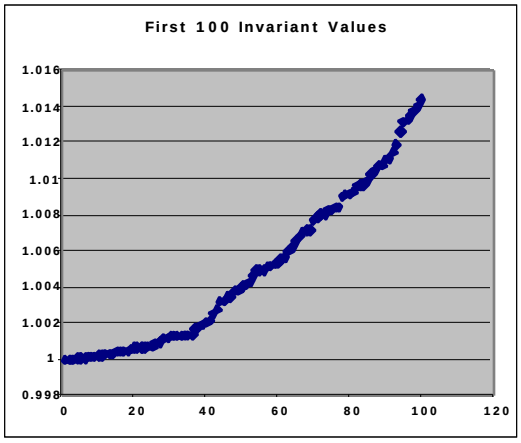

Figure 13: First 100 Values from Figure 12

Additional filtering of this invariant space is needed. Our initial hypothesis was that the invariant space would be filtered by intersecting the sets of invariants found in myltiple images. At least for our initial training set of three images, the invariant sets were so dense that little filtering by intersection is possible. Therefore, we now plan to look at methods of filtering on these small data sets before extending our work to larger training sets.

Therefore, we plan on further investigating both our initial feature extraction and our final step of extracting intersections to see where additional filtering can reduce the number of invariants without losing significant ones. One aspect that we will consider is whether filtering based cluster validity measures, both in the Hough parameter space clusters and the invariant-based clusters, can aid in the selection of relevant invariants.

Conclusion

We have made progress in automatically extracting four-point cross-ratio invariants automatically from images. A unique feature of our method is a clustering phase to reduce the dependency of line extraction on the discretization parameters of the Hough transform. Although our current method does not yet sufficiently filter the invariants, we are encouraged by our progress so far and plan to work further on the filtering steps.

References

1. Canny, J.F. "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, 1986, pp. 679-698.
2. J.C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," *J. Cybernetics*, vol. 3, 1973, pp. 32-57. Reprinted in *Fuzzy Models for Pattern Recognition*, J.C. Bezdek and S.K. Pal, eds., IEEE Press, 1992.
3. J. Illingworth and J. Kittler, "A Survey of the Hough Transform," Computer *Vision, Graphics and Image Processing*, vol. 44, 1988, pp. 87-116.
4. J.L. Mundy and A. Zisserman (eds.), *Geometric Invariance in Computer Vision*, MIT Press: Cambridge, MA, 1992.
5. E.L. Walker, "Fuzzy Relations for Feature-Model Correspondence in 3D Object Recognition," in *Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society*, June 1996, pp.28-32.
6. E.L. Walker "A Fuzzy Approach to Pose Determination" in *Proceedings of the 1997 Conference of the North American Fuzzy Information Processing Society*, Syracuse, NY, Sept. 1997, pp. 183-187.
7. E.L. Walker, "Combining Geometric Invariants with Fuzzy Clustering for Object Recognition" in *Proceedings of the 1999 Conference of the North American Fuzzy Information Processing Society*, New York, NY, June 1999, pp. 183-187.
8. I. Weiss, "Geometric Invariants and Object Recognition," *International Journal of Computer Vision*, vol. 10, no. 3, 1993, pp. 207-231.