Eliminating False Positives During Corner Finding by Merging Similar Segments

Aaron Wolin, Brandon Paulson and **Tracy Hammond**
Sketch Recognition Laboratory
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112
awolin@cs.tamu.edu, bpaulson@cs.tamu.edu, (979) 845-7143
hammond@cs.tamu.edu, (979) 862-4284

Abstract

We present a new corner finding algorithm based on merging like stroke segmentations together in order to eliminate false positive corners. We compare our system to two benchmark corner finders with substantial improvements in both polyline and complex fits.

Sketch recognition is an emerging field that utilizes pen-based interaction with computers. Handwriting recognition software in the modern operating systems allows users to write naturally, and applications have been created to recognize sketches in domains such as UML diagrams (Hammond & Davis 2002) and family trees (Alvarado & Davis 2004).

In an attempt to perform free-sketch recognition, which allows users to draw as they would naturally without training or being trained by the system, certain geometric sketch recognition systems require a shape to be defined by a set of primitives (Hammond & Davis 2007).

Individual strokes can be classified as primitive shapes using Paulson (Paulson & Hammond 2008). However, we would like to allow users to draw multiple primitives in a continuous stroke as they would naturally.

Corner detection allows a user to draw in their own style while still allowing a user's drawn shapes to benefit from sketch recognition systems that rely on primitives. In a corner detection system, algorithms automatically break a user's drawn stroke into primitive lines and arcs. This task can be completed during stroke preprocessing, and the resulting primitives can then be sent to a geometric recognizer for stroke classification.

We propose MergeCF, a corner detection algorithm that utilizes the stroke's curvature and the user's drawing pen speed in order to find the corners of a stroke. MergeCF then eliminates false positives by removing similar corners, merging like stroke segments together, and examining stroke segments' direction values. Our corner finder is powerful and improves upon current state-of-the-art techniques using two different accuracy measures.

Implementation

MergeCF utilizes curvature and speed differences within a stroke to obtain an initial corner segmentation for our stroke. We then repeatedly merge smaller stroke segments with longer segments, and, if the fit for the merged segment is below a certain threshold, we eliminate the corner between the two segments.

Curvature and Speed values

Our curvature and speed values are based on the equations given by (Stahovich 2004). The chord length between two points is the euclidean distance between the points, and the path length across a series of points $p_a, p_{a+1}, \ldots, p_b$ is taken to be the sum of the euclidean distances between each pair of points. Speed at a point p_i is calculated as the path length change divided by the time difference across the point's neighbors, p_{i-1} and p_{i+1}.

Initial Fit

After MergeCF computes the curvature and speed values for each point, the system finds an initial set of corners by taking points that are above a curvature threshold and below a speed threshold. MergeCF also checks for corners that are close together in proximity and removes the corner with the smallest curvature from the initial fit.

Merging Segments

MergeCF's initial corner fit tends to contain a few extraneous points that overfit the stroke. The main algorithm involved with our corner finding system is designed to eliminate these false positives.

The algorithm works on the assumption that corners surrounding the smallest stroke segments are likely to be false positives overfitting the data. The algorithm to eliminate these false positives first finds the smallest stroke segment, checks if the segment can be merged with any of its neighbors, and then merges the segment with the "best" neighboring segment. The best segment is determined to be the segment that has the least primitive fit error (either line or arc) when combining the two segments.

For example, Figure 1 shows a symbol with an initial corner fit containing three false positives (the circled points) and numbered stroke segments. Merging segment 5 with segment 4 would still result in an arc fit error that is not too much higher than the either segment 4 or 5's original error. Yet, merging segment 5 with segment 6 would produce a very high primitive error for either lines or arcs. Therefore,

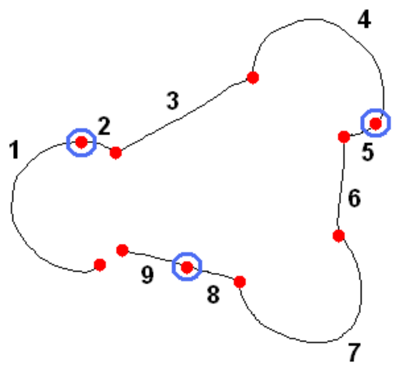

Figure 1: Initial set of corners found for a stroke, which would split the stroke into 9 primitive lines and arcs. False positives are circled.

the best segments to merge are 4 and 5, and the circled point in between the two segments is removed from the corner set. It is important to note that no merging occurs if the error when merging two segments is much higher than the sum of the original errors of the segments. We calculate line and arc errors using the primitive recognizer presented in (Paulson & Hammond 2008).

Results

MergeCF was tested on a set of 501 unistroke symbols drawn by six users. The symbols in this set are based on the symbols found in (Kim & Kim 2006). The this set contained both polylines and complex shapes.

Results were also gathered on two other corner finders: Sezgin et al.'s algorithm (Sezgin, Stahovich, & Davis 2001) and Kim and Kim's algorithm (Kim & Kim 2006). We implemented both corner finders as presented in their respective papers, and we tested all of the corner finders on the same data sets. Table 1 displays our test results.

We used two different measures to determine the accuracy of a corner finder. The first accuracy measure is a "correct corners found" accuracy as presented by (Sezgin, Stahovich, & Davis 2001). This accuracy is calculated by dividing the number of correct corners found divided by the total number of correct corners perceived. Essentially, this accuracy measure does not discount false positives, and returning every point in a stroke would constitute a 1.00 accuracy since all of the correct corners have been found.

To counteract this issue, we also calculate an all-or-nothing accuracy for each corner finder. All-or-nothing implies that only the minimum number of corners to segment a figure are found (i.e. there are no false positives or negatives) in order for a stroke to be considered correctly segmented. This accuracy is calculated by taking the number of correctly segmented strokes divided by the total number of strokes.

	Merge	Sezgin	Kim
Correct Corners Found	0.971	0.822	0.783
All-or-Nothing Accuracy	0.667	0.298	0.194

Table 1: Results from the corner finding algorithms.

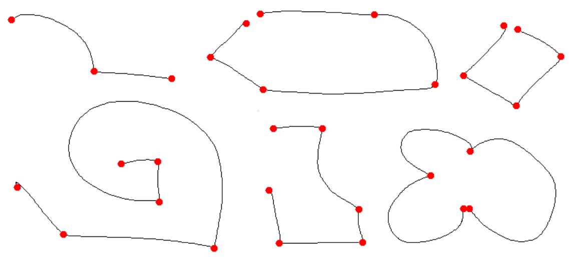

Figure 2: Examples of correctly classified symbols by MergeCF

Conclusion

We have presented a new corner finding technique that utilizes curvature and pen-speed values of a stroke to obtain an initial corner fit. After a fit is found, we eliminate false positives by examining small stroke segments and merging the segments with similar neighbors. Overall, our system greatly improves upon the existing benchmarks in two different accuracy measures.

Acknowledgements

This work funded in part by NSF IIS grant 0744150: Developing Perception-based Geometric Primitive-shape and Constraint Recognizers to Empower Instructors to Build Sketch Systems in the Classroom.

For more information, please see our website at: http://srl.csdl.tamu.edu/mergecf.shtml

References

Alvarado, C., and Davis, R. 2004. Sketchread: a multi-domain sketch recognition engine. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, 23–32. New York, NY, USA: ACM Press.

Hammond, T., and Davis, R. 2002. Tahuti: A geometrical sketch recognition system for uml class diagrams. *Papers from the 2002 AAAI Spring Symposium on Sketch Understanding* 59–68.

Hammond, T., and Davis, R. 2007. Ladder, a sketching language for user interface developers. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, 35. New York, NY, USA: ACM.

Kim, D., and Kim, M.-J. 2006. A curvature estimation for pen input segmentation in sketch-based modeling. In *Computer-Aided Design*, 238–248.

Paulson, B., and Hammond, T. 2008. Paleosketch: Accurate primitive sketch recognition and beautification. In *IUI (Intelligent User Interfaces)*.

Sezgin, T. M.; Stahovich, T.; and Davis, R. 2001. Sketch based interfaces: Early processing for sketch understanding. *Workshop on Perceptive User Interfaces, Orlando FL.*

Stahovich, T. F. 2004. Segmentation of pen strokes using pen speed. In *AAAI Fall Symposium Series 2004: Making Pen-Based Interaction Intelligent and Natural.*