Submitted to MIT Oxygen 2001 Workshop

Free-Hand Stroke Approximation for Intelligent Sketching Systems

Tevfik Metin Sezgin MTSEZGIN@AI.MIT.EDU
MIT Artificial Intelligence Laboratory, 200 Technology Square, Cambridge MA, 02139 USA

Randall Davis DAVIS@AI.MIT.EDU
MIT Artificial Intelligence Laboratory, 200 Technology Square, Cambridge MA, 02139 USA

1. Introduction

Free-hand sketching is used extensively during the early design phases as an important tool for conveying ideas, guiding the thought process, and serving as documentation (Ullman et al., 1990). It also provides a natural way of interaction desirable in the context of E21 and intelligent design spaces. Unfortunately there is little computer support for sketching. The first step in building a sketch understanding system is generating more meaningful descriptions of free-hand strokes. We describe a system that takes strokes described by an array of points along with timing data, and generates such descriptions in terms of geometric primitives such as circles, polylines, curves and their combinations.

2. System Description

2.1 Issues

Loss of information due to digitization, noise in the data, imprecision due to freehand sketching and sparseness of sampled data points – which may be as low as 4-5 dpi as opposed to scanned drawings that may have 1200-2400 dpi – complicate the stroke approximation task.

2.2 Feature Point Detection

Stroke processing starts by looking for vertices. We want to avoid picking points on the curved regions as much as possible[1]. Vertex localization is a frequent subject in the extensive literature on graphics recognition (e.g., (Rosin, 1996) compares 21 methods). However these methods produce piecewise linear approximations.

Our approach takes advantage of the interactive nature of sketching by combining information from both curvature and speed data for detecting corners while avoiding a piecewise linear approximation. Feature points are indicated by the maxima of curvature[2] data and the minima of the pen speed. Both speed and curvature data are noisy and simply picking the extrema introduces many false positives. Below we describe two methods for selecting the extrema in each case.

2.2.1 Average Based Filtering

The basic idea in average based filtering is confining our search for the vertices to only those regions where the curvature/speed data is above/below a threshold computed by the mean of the data. Then we search for the global extremum within each region to find the feature points.

2.2.2 Scale Space Filtering

Our experiments show that the above method is satisfactory for settings where noise is relatively small but doesn't deal with extremely noisy data as in Fig. 1 very well. Because it doesn't differentiate between vertices due to the fine scale structure of the noise and those due to the high level structure of the stroke. The scale space based method deals with this problem by looking at the number of feature points present at different scales in the scale space. As seen in Fig. 2, the feature count graph has two distinct regions where the feature count drops with different rates. This behavior is typical for freehand strokes.

Our task is selecting a scale where most of the noise is filtered out. This is done by modeling the feature count graph by fitting two lines to it: one to the region with the steep drop corresponding to places where the feature points due to noise disappear and the other to the flatter region where real feature points disappear. We take the scale corresponding to the intersection of these two lines as our scale.

The results obtained by applying this technique to curvature and speed data are in Fig. 3.

[1] Piecewise linear approximation algorithms don't satisfy this requirement.

[2] From this point on, by curvature we will refer to the absolute value of the curvature data defined as $|\partial d/\partial s|$ where d is the angle between the tangent to the curve at a point and the x axis and s is the distance traveled along the curve.

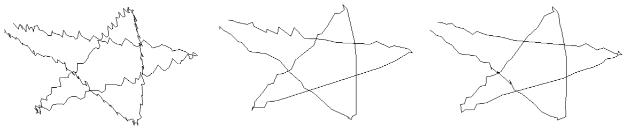

Figure 1. A very noisy stroke along with the fits generated using curvature and speed data with average based filtering. Both fits have picked many false positives due to noise.

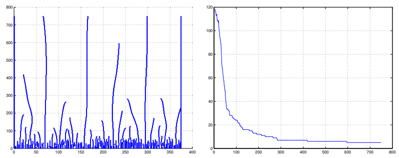

Figure 2. The scale space and feature count graph for the noisy stroke in Fig. 1.

2.2.3 GENERATING HYBRID FITS

The above methods separately may miss feature points, so we combine them to generate hybrid fits. We start the hybrid fit generation by letting the intersection of the generated fits to be our initial hybrid fit. At each iteration, we form a list of candidate vertices by picking the vertex with the highest certainty from each of the original fits. We augment the most recent hybrid fit by the candidate from this list that reduces the least squares error of the fit by the largest amount.

2.3 Handling Curves

Our system also supports strokes that may contain curves. First we detect curved regions by comparing the curve length l_1 between the consecutive detected feature points to the Euclidean distance l_2 between the two points. l_2/l_1 is significantly larger than 1 in curved regions. The curved regions are then approximated by Bézier curves.

3. Evaluation

For purposes of evaluation, we built a higher level recognizer that takes the output of our system and interprets strokes in the mechanical engineering domain. Fig. 4 shows a number of free hand sketches and the output of the recognizer[3].

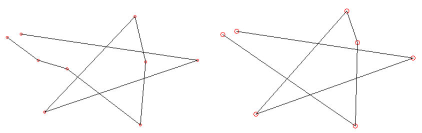

Figure 3. Curvature and speed fits generated using the scale space approach. These fits contain 9 vertices for the curvature and 7 vertices for the speed fit compared to 69 and 82 vertices for the fits obtained using average based filtering in Fig. 1.

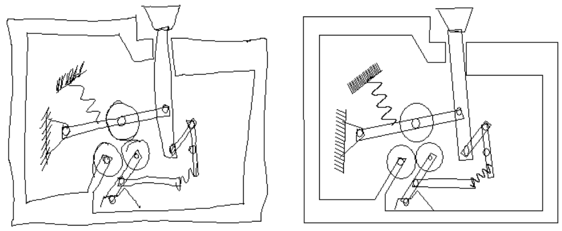

Figure 4. Input-output figures for the rough sketch of the direction reversal mechanism of a walkman.

4. Related Work

Related work can be found in (Landay & Myers, 2001; Eggli, 1994; Zeleznik, 1996; Rattarangsi & Chin, 1992) but these systems either don't support drawing arbitrary shapes or don't do automatic vertex detection. Some of them require drawing modes making them unnatural free-hand sketching interfaces.

5. Future Work

Future directions include potential improvements, user studies and integration with other systems including systems that can learn and classify strokes, patterns or objects taking the concise representation of strokes generated by our system as inputs.

References

Eggli, L. (1994). Sketching with constraints. Master's thesis, University of Utah.

Landay, J. A., & Myers, B. A. (2001). Sketching interfaces: Toward more human interface design. *IEEE Computer, vol. 34, no. 3, March 2001, pp. 56-64.*

Rattarangsi, A., & Chin, R. T. (1992). Scale-based detection of corners of planar curves. *IEEE Transactionsos Pattern Analysis and Machine Intelligence*, *14*, 430–339.

Rosin, R. (1996). Techniques for assessing polygonal approximations of curves. *7th British Machine Vision Conf., Edinburgh*.

Ullman, D. G., Wood, S., & Craig, D. (1990). The importance of drawing in the mechanical design process. *Computers and Graphics*, *14*, 263–274.

Zeleznik, R. (1996). Sketch: An interface for sketching 3d scenes. *Proceedings of SIGGRAPH'96* (pp. 163–170).

[3] Some domain specific objects such as springs and ground symbols are recognized by the recognizer.