Scale-space Based Feature Point Detection for Digital Ink

Tevfik Metin Sezgin and Randall Davis
MIT Computer Science and Artificial Intelligence Laboratory
The Stata Center 235
Cambridge MA, 02139
{mtsezgin,davis}@csail.mit.edu

Abstract

Feature point detection is generally the first step in model-based approaches to sketch recognition. Feature point detection in free-hand strokes is a hard problem because the input has noise from digitization, from natural hand tremor, and from lack of perfect motor control during drawing. Existing feature point detection methods for free-hand strokes require hand-tuned thresholds for filtering out the false positives. In this paper, we present a threshold-free feature point detection method using ideas from the scale-space theory.

Introduction

There is increasing interest in building systems that can recognize and reason about sketches. Among different approaches to sketch recognition, model-based recognition techniques model objects in terms of their constituent geometric parts and how they fit together (e.g., a rectangle is formed by four lines, all of which intersect at right angles at four distinct corners). In order to be able to match scene elements to geometric model parts, it is necessary to convert the free-hand strokes in the scene into geometric primitives, which results in a more concise and meaningful description of the scene compared to a raw representation only in terms of sampled pen positions. As described in (Sezgin *et al.* November 2001), feature point (i.e., corner) detection is a major part of generating such geometric descriptions.

Issues

The major issue in feature point detection is the noise in the data. We consider noise from two sources: imprecise motor control and digitization. We describe characteristics of each kind of noise with examples to make the distinction clear.

Imperfect motor control

Examples of noise due to imperfect motor control include line segments that were meant to be straight but are not, or corners that look round as opposed to having a precise turning point. This kind of noise gives sketches their "messy" appearance. Easiest way of characterizing this kind of noise is to ask if the noise would still be present if the user drew very carefully perhaps using a ruler. If the answer is negative, then the noise is due to imperfect motor control.

Digitization noise

Digitization noise is the kind of noise that cannot be removed even if one draws very carefully. Although visually less apparent, it hinders feature point detection because digitization corrupts curvature and speed data, which are primary sources of information in feature point detection. Digitization noise can be present in the (x, y) positions and in their timestamps. Source of the spatial digitization noise is the conversion to screen coordinates. For example, in the Acer C110 Tablet PC, the pen positions are digitized into a 1024x768 grid. Spatial digitization can be so poor that the point stream returned by digitization may occasionally have points with repeating (x, y) positions.

In the same platform, timestamps too have digitization noise. Because the concept of having digitization noise in timestamps is less intuitive, we illustrate the point with an example. Consider the stroke in Fig. 1 captured using an Acer c110 Tablet PC. In this platform, we know that the hardware samples points uniformly at a high resolution, digitizing the timestamps once. Then, the operating system digitizes the timestamps again at 100 Hz. Although the timestamps are good when read at the higher resolution directly using Microsoft's Tablet PC API, they get corrupted during digitization. For the stroke in Fig. 1, Fig. 2 shows the deviation of the digitized timestamps from their predicted ground truth values computed by a least squares linear regression line. The slope of the least squares regression gives us the hardware sampling rate (which is about 133 Hz). The difference in the sampling frequencies causes a skew to accumulate between the real timestamps of the points and those obtained after digitization. The timestamps are occasionally adjusted for the skew by repeating a timestamp, which occurs about every four points with a standard deviation of 0.53. Furthermore, although less frequent, the digitizer consistently returns a point which is 11ms apart from the previous one (as opposed to the more frequent 10ms time difference). This happens roughly once every 92 points ($\mu = 92.25$, $\sigma = 0.95$). If we consider that the time resolution at a sampling rate of 100Hz is 10 ms, the deviations in Fig. 2 which range between $(-8, 6)$ with $\sigma = 3.02ms$ is quite significant. Digitization noise of this nature causes

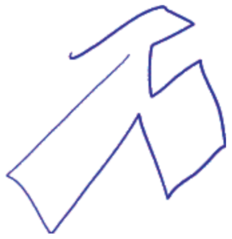
Figure 1: A free-hand stroke captured using Acer c110.

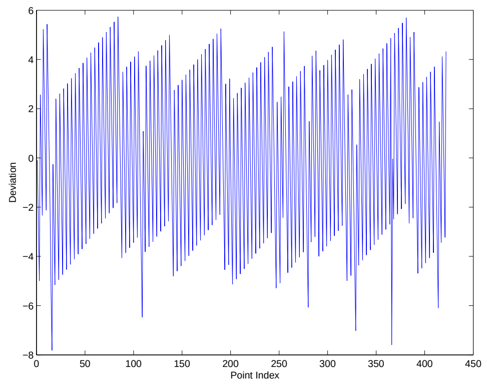

Figure 2: This graph shows the deviation of the timestamps from their linear regression line measured in milliseconds.

speed data computed by taking the time derivative of position to be noisy.

Similar digitization noise behavior is also present in the HP tc1100. Although different in nature, digitization noise is also present for mouse based interfaces, digitizing tablets such as the Wacom PL-400 and the Mimio, a whiteboard capture hardware. The case of Acer c110 and HP tc1100 is more interesting in part because there is a two layer digitization process.

The mainstream approach to dealing with noise is to use filtering criteria based on thresholds preset either by hand or learned from labeled calibration data.

Another approach to dealing with noise is down-sampling points in an effort to achieve a less noisy signal, but such methods throw away potentially useful information when they use fewer than all the points. Furthermore free-hand sketching data is already sparse[1]. Here, we describe a feature point detection system that doesn't depend on preset thresholds or constants, and uses all the points in the stroke.

System Description

Feature Point Detection

Feature point detection is the task of finding corners (vertices) of a stroke. We want to be able to find corners of piecewise linear strokes. For strokes that have curved parts (complex shapes), we want to be able to identify points where curved and straight segments connect. Our technique works for piecewise linear shapes and complex shapes. Requiring the ability to handle complex shapes complicates the problem significantly and rules out well studied piecewise linear approximation algorithms. [2] For strokes with curved portions, we would like to avoid picking points on the curved regions resulting in a piecewise linear approximation of the curved regions.

Our approach takes advantage of the availability of point timestamps during online sketching and combines information from both curvature and speed data, while avoiding a piecewise linear approximation.

Feature points are indicated by maxima of curvature[3] and the minima of pen speed. The strategy of corner detection through local extrema in curvature and speed data would work perfectly in an ideal noiseless setup. In practice it results in many false positives, because local extrema due to the fine scale structure of the noise and those due to the high level structure of the stroke get treated the same way.

One could try setting parameters to filter out these false positives but selecting a priori parameters has the problem of not lending itself to different scenarios where object features and noise may vary. Our experience with the average based feature detection method in (Sezgin *et al.* November 2001) is that its parameters need adjustment for different stroke capture hardware and sometimes for different users. For example, some people tend to make corners more rounded than others. This requires adjusting the parameters of the system for different conditions, a tedious task for the user who must supply data on each platform for calibration purposes, and for the programmer who should find a good set of parameters for each case. Our aim is to remove this overhead by removing the dependence of our algorithms on preset thresholds.

As indicated by our experiments, the extrema due to noise disappear if we look at the data at coarser scales while those due to the real feature points persist across coarser scales. We base our feature point detection technique on our observation that features due to noise and real features exist at different scales. We use the scale-space framework to derive coarser and smoother versions of the data and use the way the number of feature points evolves over different scales to select a scale where the extrema due to noise don't exist. We give details of how we achieve this after a brief introduction to the scale space concept.

Scale-space representation

An inherent property of real-world objects is that they exist as meaningful entities over a limited range of scales. The classical example is a tree branch. A tree branch is meaningful at the centimeter or meter levels, but looses its meaning at very small scales where cells, molecules or atoms make

[1]Data sampled using a traditional digitizing tablet or a Tablet PC may have resolution as low as 4-5 dpi as opposed to scanned drawings with up to 1200-2400 dpi resolution. This is because sometimes users draw so fast that even with high sampling rates such as 100Hz only few points per inch can be sampled.

[2]Vertex localization for piecewise linear shapes is a frequent subject in the extensive literature on graphics recognition. (e.g., (Rosin 1996) compares 21 methods).

[3]Defined as $|\partial\theta/\partial s|$ where θ is the angle between the tangent to the curve at a point and the x axis and s is the cumulative curve length.

sense, or at very large scales where forests and trees make sense.

A technique for dealing with features at multiple scales is to derive representations of the data through multiple scales. The scale-space representation framework introduced by Witkin (Witkin 1983) allows us to derive such multi-scale representations in a mathematically sound way.

The virtues of the scale-space approach are twofold. First, it enables multiple interpretations of the data. These interpretations range from descriptions with a fine degree of detail to descriptions that capture only the overall structure of the stroke. Second, the scale-space approach sets the stage for selecting a scale or a set of scales by looking at how the interpretation of the data changes and features move in the scale-space as the scale is varied.

The basic idea behind the scale-space representation is to generate successively higher level descriptions of a signal by convolving it with a filter. As our filter, we use the Gaussian defined as:

$$g(s, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-s^2/2\sigma^2}$$

where σ is the smoothing parameter that controls the scale. A higher σ means a coarser scale, describing the overall features of the data, while a smaller σ corresponds to finer scales containing the details. The Gaussian filter does not introduce new feature points as the scale increases. This means that as scales get coarser, the number of features (obtained by extrema of the data in question) either remains the same or decreases (i.e., neighboring features are merge causing a decrease in the total number of feature points). The Gaussian kernel is unique in this respect for use in scale-space filtering as discussed in (Yuille & Poggio 1986) and (Babaud *et al.* 1986).

In the continuous case, given a function $f(x)$, the convolution is given by:

$$F(x, \sigma) = f(x) * g(x, \sigma) = \int_{-\infty}^{\infty} f(u) \frac{1}{\sigma\sqrt{2\pi}} e^{(x-u)^2/2\sigma^2}\, du$$

We use the discrete counterpart of the Gaussian function which satisfies the property:

$$\sum_{i=0}^{n} g(i, \sigma) = 1$$

Given a Gaussian kernel, we convolve the data using the following scheme:

$$x_{(k,\sigma)} = \sum_{i=0}^{n} g(i, \sigma) x_{k - \lfloor n/2+1 \rfloor + i}$$

There are several methods for handling boundary conditions when the extent of the kernel goes beyond the end points. In our implementation, we assume that for $k - \lfloor n/2+1 \rfloor + i < 0$ and $k - \lfloor n/2+1 \rfloor + i > n$ the data is padded with zeroes on either side.

Scale selection

The scale-space framework provides a concise representation of the behavior of the data across scales, but doesn't tell us what scale(s) to attend to. In our case, we would like to know what scale to attend to for separating noise from real features. The next two sections explain how we used the feature count for scale selection for curvature and speed data.

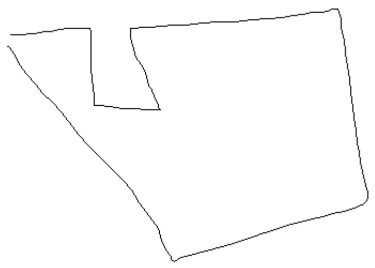
Figure 3: A freehand stroke.

Application to curvature data

We start by deriving direction and curvature data, then derive a series of functions from the curvature data by smoothing it with Gaussian filters of increasing σ. We build the scale-space by finding the zero crossings of the curvature at various scales.

Scale-space is the (x, σ)-plane where x is the dependent variable of function $f(.)$ (Witkin 1983). We focus on how maxima of curvature move in this 2D plane as σ is varied.

Fig. 3 shows a freehand stroke and Fig. 4 the scale-space map corresponding to the features obtained using curvature data. The vertical axis in the graph is the scale index σ (increasing up); the horizontal axis ranges from 0 to 178 indicating which of the points in the original stroke is calculated to be a feature point. The stroke in question contains 179 points. We detect the feature points by finding the negative zero-crossings of the derivative of absolute value of the curvature. We do this at each scale and plot the corresponding point (σ, i) for each index i in the scale-space plot. An easy way of reading this plot is by drawing a horizontal line at a particular scale index, and then look at the intersection of the line with the scale-space lines. The intersections give us the indices of the points in the original stroke indicated to be feature points at that scale.

As seen in this graph, for small σ (near the bottom of the scale-space graph), many points in the stroke are classified as vertices, because at these scales the curvature data has many local maxima, most of which are caused by the noise in the signal. For increasing σ, the number of feature points decreases gradually.

Our next step is to choose a scale where the false positives due to noise are filtered out and we are left with the real vertices of the data. We want to achieve this without having any particular knowledge about the noise[4] and without having preset scales or constants for handling noise.

The approach we take is to keep track of the number of feature points as a function of σ and find a scale that preserves the tradeoff between choosing a fine scale where the data is too noisy and introduces many false positives, and choosing a coarse scale where true feature points are filtered

[4]The only assumption we make is that the noise is at a different scale than the feature size.

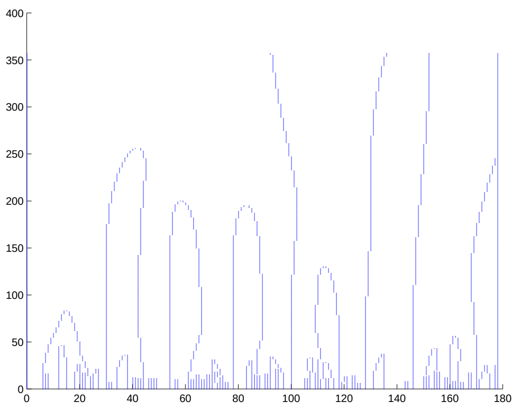
Figure 4: The scale-space for the maxima of the absolute curvature for the stroke in Fig. 3. This plot shows how the maxima move in the scale-space. The x axis is the indices of the feature points, the y axis is the scale index.

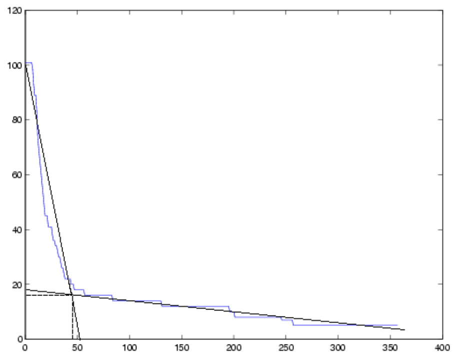
Figure 5: This plot shows the drop in feature point count (y axis) for increasing σ (x axis) and the scale selected by our algorithm for the stroke in Fig. 3.

out. For example, the stroke in Fig. 3, has 101 feature points for $\sigma = 0$. On the coarsest scale, we are left with only 5 feature points, two of which are end points. This means 4 actual feature points are lost by the Gaussian smoothing. Because the noise in the data and the shape described by the true feature points are at different scales, it becomes possible to detect the corresponding ranges of scales by looking at the feature count graph.

Fig. 5 gives the feature count graph for the stroke in Fig. 3. In this figure, the steep drop in the number of feature points that occurs for scales in the range $[0, 40]$ roughly corresponds to scales where the noise disappears, and the region $[85, 357]$ roughly corresponds to the region where the real feature points start disappearing. Fig. 6 shows the scale-space behavior during this drop by combining the scale-space with the feature-count graph. In this graph, the x, y, axis z, respectively correspond to the feature point index [0,200], σ [0,400], and feature count [0,120]. We read the graph as follows: given σ, we find the corresponding location in the y axis. We move up parallel to the z axis until we cross the first scale-space line.[5] The z value at which we

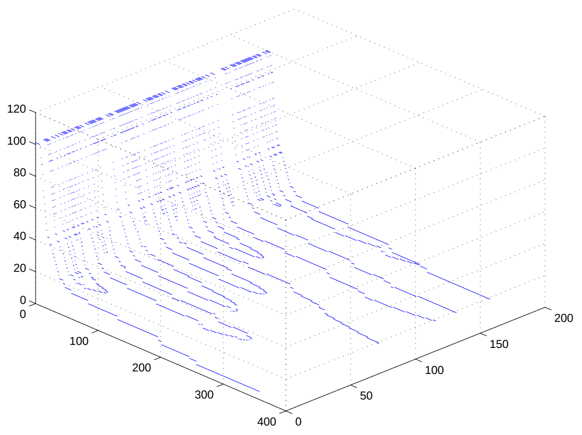
Figure 6: Joint scale-space feature-count graph for the stroke in Fig. 3, simultaneously showing feature point movements in the scale-space and the drop in feature point count for increasing σ.

cross the first scale-space line gives the feature count at scale index σ. Now, we draw a line parallel to the x axis. Movements along this line correspond to different feature indices, and its intersection with the scale-space plot corresponds to indices of feature points present at scale index σ. The steep drop in the feature count is seen in both Fig. 5 and Fig. 6.

Our experiments suggest that this phenomena (i.e., the drop) is present in all hand drawn curves, except in singular cases such as a perfectly horizontal or perfectly vertical line drawn at a constant speed. We model the feature count - scale graph by fitting two lines and derive the scale where the noise is filtered out using their intersection. Specifically, we compute a piecewise linear approximation to the feature count - scale graph with only two lines, one of which tries to approximate the portion of the graph corresponding to the drop in the number of feature points due to noise, and the other that approximates the portion of the graph corresponding to the drop in the number of real feature points. We then find the intersection of these lines and use its x value (i.e., the scale index) as the scale. Thus we avoid extreme scales and choose a scale where most of the noise is filtered out.

Fig. 5 illustrates the scale selection scheme via fitting two lines l_1, l_2 to the feature count - scale graph. The algorithm to get the best fit simply finds the index i that minimizes $OD(l_1, \{P_j\}) + OD(l_2, \{P_k\})$ for $0 \leq j < i$, $i \leq k < n$. $OD(l, \{P_m\})$ is the average orthogonal distance of the points P_m to the line l, P is the array of points in the feature count - scale graph indexed by the scale parameter, and $0 \leq i < n$ where n is the number of points in the stroke. Intuitively, we divide the feature count - scale graph into two regions, fit an ODR line to each region, and compute the orthogonal least squares error for each fit. We search for the division that minimizes the sum of these errors, and select the scale corresponding to the intersection of the lines for which the division is optimal (i.e., has minimum error).

Interestingly enough, we have reduced the problem of stroke approximation via feature detection to fitting lines to

[5]The first scale-space line corresponds to the zeroth point in our stroke, and by default it is a feature point and is plotted in the scale space plot. This remark also applies to the last point in the stroke.

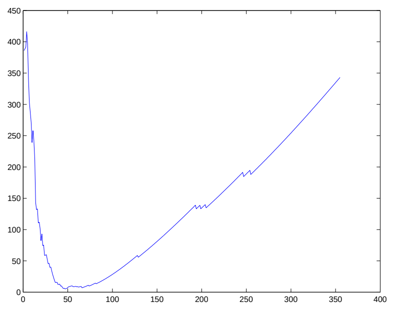

Figure 7: The summed error for the two lines fit to Fig. 5 during scale selection for the stroke in Fig. 3.

the feature count graph, which is similar in nature to the original problem. However, now we know how we want to approximate the data (i.e., with two lines). Therefore even an exhaustive search for i corresponding to the best fit becomes feasible. As shown in Fig. 7 the error as a function of i is U shaped. Thus, if desired, the minima of the summed error can be found using gradient descent methods, by paying special attention to not getting stuck in the local minima. For the stroke in Fig. 3, the scale selected by our algorithm is 47.

While we try to choose a scale where most of the false maxima due to noise are filtered out, feature points at this scale may still contain some false positives. This problem of false extrema in the scale space is also mentioned in (Rattarangsi & Chin 1992), where these points are filtered out by looking at their separation from the line connecting the preceding and following feature points. They filter these points out if the distance is less than one pixel.

The drawback of the filtering technique in (Rattarangsi & Chin 1992) is that the scale-space has to be built differently. Instead of computing the curvature for $\sigma = 0$ and then convolving it with Gaussian filters of larger σ to obtain the curvature data at a particular scale, they treat the stroke as a parametric function of a third variable s, path length along the curve. The x and y components are expressed as parametric functions of s. At each scale, the x and y coordinates are convolved with the appropriate Gaussian filter and the curvature data is computed. It is only after this step that the zero crossings of the derivative of curvature can be computed for detecting feature points. The x and y components should be convolved separately because filtering out false feature points requires computing the distance of each feature point to the line connecting the preceding and following feature points, as explained above. This means the Gaussian convolution, a costly operation, has to be performed twice in this method, compared to a single pass in our algorithm.

Because we convolve the curvature data instead of the x and y coordinates, we can't use the method mentioned above. Instead we use an alternate 2-step method to minimize the number of false positives. First we check whether there are any vertices that can be removed without increasing the least squares error between the generated fit and the original stroke points. The second step in our method takes the generated fit, detects consecutive collinear[6] edges and combines these edges into one by removing the vertex in between. After performing these operations, we get the fit in Fig. 8.

[6]Measure of collinearity is determined by the task in hand. We consider lines with $|\Delta\theta| \leq \pi/32$ to be collinear.

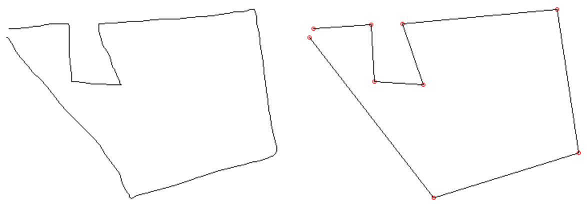

Figure 8: The input stroke (left) and the features detected by looking at the scale-space of the curvature (right).

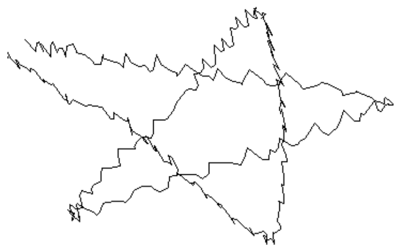

Figure 9: A very noisy stroke.

One virtue of the scale-space approach is that works extremely well in the presence of noise. In Fig. 9 we have a very noisy stroke. Figure 10 shows the feature-count and scale-space behaviors respectively. The output of the scale-space based algorithm is in Fig. 11. This output contains only 9 points. For comparison purposes, the output of the average based feature detection algorithm (Sezgin *et al.* November 2001) based on curvature is also given in Fig. 11. This fit contains 69 vertices. (The vertices are not marked for the sake of keeping the figure clean.)

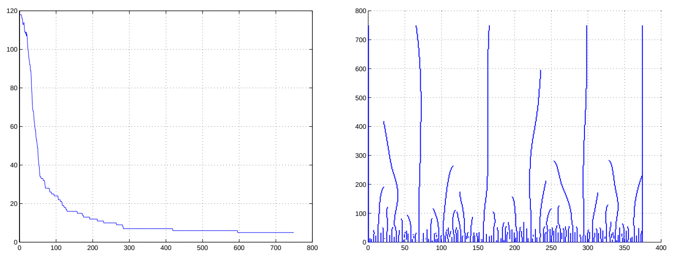

Figure 10: The feature count for increasing σ and the scale-space map for the stroke in Fig. 9. Even with very noisy data, the behavior in the drop is the same as it was for Fig. 3.

Application to speed change

We also applied the scale selection technique described above to speed data. The details of the algorithm for deriving the scale-space and extracting the feature points are similar to that of the curvature data except for obvious differences (e.g., instead of looking for the maxima, we look for the minima).

Fig. 12 has the scale-space, feature-count and joint graphs

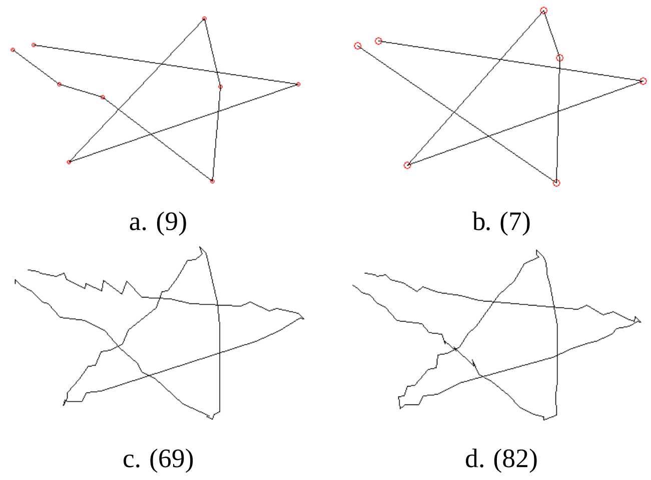

Figure 11: Above, curvature (a) and speed (b) fits generated for the stroke in Fig. 9 with scale-space filtering. Below, fits generated using average based filtering (c,d). For each fit, the number of vertices is given in parenthesis.

for the speed data of the stroke in Fig. 9. As seen in these graphs, the behavior of the speed scale-space is similar to the behavior we observed for the curvature data. We use the same method for scale selection. In this case, the scale index picked by our algorithm was 72. The generated fit is in Fig. 11 along with the fit generated by the average based filtering method using the speed data.

For the speed data, the fit generated by scale-space method has 7 vertices, while the one generated by the average based filtering has 82. In general, the performance of the average based filtering method is not as bad as this example may suggest. For example, for strokes as in Fig. 3, the performance of the two methods are comparable, but for extremely noisy data as in Fig. 9, the scale-space approach pays off when using curvature and speed data.

Because the scale-space approach is computationally more costly[7], using average based filtering is preferable for data that is less noisy. There are also scenarios where only one of curvature or speed data may be noisier. For example, in some platforms, the system-generated timing data for pen motion required to derive speed may not be precise enough, or may be noisy. In this case, if the noise in the pen location is not too noisy, one can use the average based method for generating fits from the curvature data and the scale-space method for deriving the speed fit. This is a choice that the user has to make based on the accuracy of the hardware used to capture the strokes, and the computational limitations.

Combining information sources

Above, we described two feature point detection methods but didn't give a way of combining the results of each

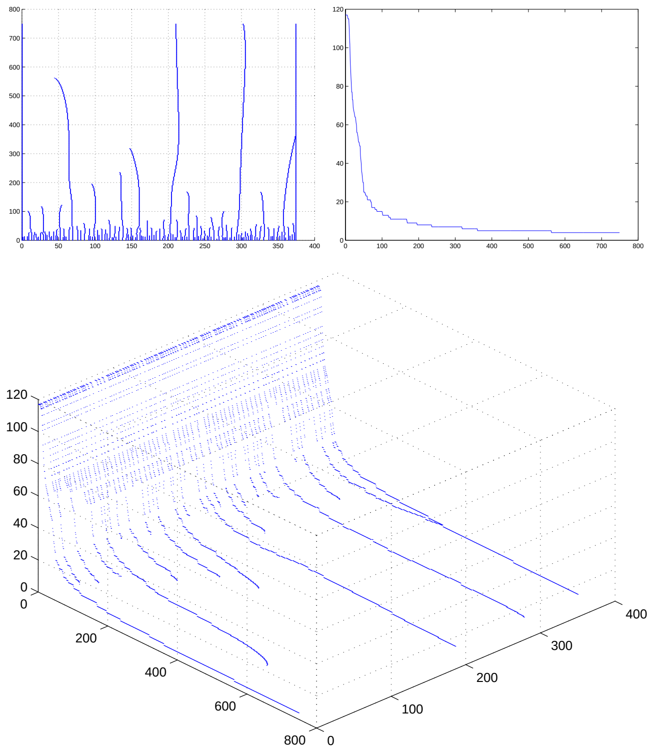

Figure 12: The scale-space, feature-count and joint graphs for the speed data of the stroke in Fig. 9. In this case, the scale selected by our algorithm is 72.

method. The hybrid fit generation method described in (Sezgin *et al.* November 2001) can be used to combine the results from two methods to utilize both information sources.

Handling complex strokes

As we mentioned earlier, we would like our method to work for strokes even if they have curved segments. In such cases, we would like to avoid piecewise linear approximations for the curved portions. In our framework, each curved region behaves like a big and smooth corner. Some arbitrary point on the curve (which happens to be the local extreme at the scale selected by our algorithm) gets recognized as a corner. This makes it possible to avoid a piecewise linear approximation of the curved segments. The curve detection method described in (Sezgin *et al.* November 2001) can be applied to detect the curved portions of a stroke.

Evaluation

We measured the performance of our scale-space based feature detection method on strokes from three different setups: Two Tablet PCs (an Acer c110 and an HP tc1100), and a Wacom digitizing LCD tablet PL-400. We chose the average based filtering method as our baseline method and compared our method's performance against it.

We collected data from 10 users. For each platform, the users were asked to draw three instances of 8 shapes. Six of these are shown in Fig. 13, the other two are rectangles rotated 45^o and -45^o. For each user on each platform, we counted the total number of errors (in our case either a

[7]Computational complexity of the average based filtering is linear with the number of points where the scale-space approach requires quadratic time if the scale index is chosen to be a function of the stroke length.

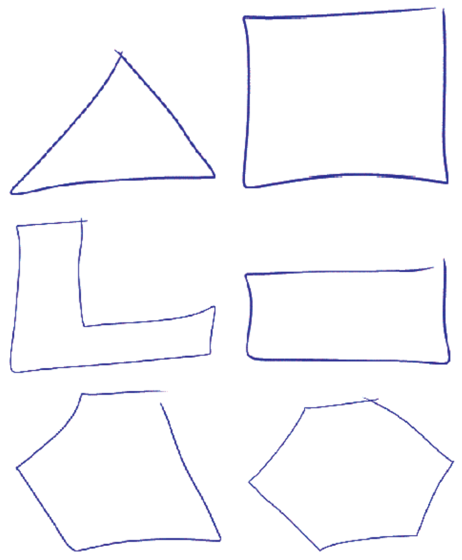
Figure 13: Shapes used in evaluation.

	Acer c110	HP tc1100	Wacom PL-400
T	14	9	11.5

Figure 14: T values for the Wilcoxon matched-pairs signed-ranks test for our feature point detection method and the baseline with data collected using three different setups.

false positive or a false negative) using our feature detection method and the baseline method. For the baseline method we used hand-tuned parameters that gave the best possible fitting results. For each platform, we compared the total number of errors made by each method using the Wilcoxon matched-pairs signed-ranks test (Siegel 1956) with the null hypothesis that the feature detection methods have comparable performance. The T values we obtained for each platform is given in table 14.

Although we were unable to reject the null hypothesis for any platform with a significance of 5% for a two tailed test, in one case we obtained a T value of 9, very close to the value 8 required for rejecting H_0 in favor of our method with level of significance 2.5% for a one tailed test. Overall, our approach compared favorably to the average based filtering method, without the need to hand-tune thresholds for dealing with the noise on each platform.

Related and Future Work

Previous methods on feature point detection either rely on preset constants and thresholds (Sezgin *et al.* November 2001; Calhoun *et al.* 2002), or don't support drawing arbitrary shapes (Schneider 1988; Banks & Cohen 1990).

In the pattern recognition community (Bentsson & Eklundh 1992; Rattarangsi & Chin 1992; Lindeberg 1996) apply some of the ideas from scale-space theory to similar problems. In particular (Bentsson & Eklundh 1992; Rattarangsi & Chin 1992) apply the scale-space idea to detection of corners of planar curves and shape representation, though they focus on shape representation at multiple scales and don't present a scale selection mechanism. The work by (Lindeberg 1996) presents a way of normalizing operator responses (feature strengths) for different σ values such that values across scales become comparable. He presents a scale selection mechanism which finds maxima of the data across scales. Although this method has the merit of making no assumptions about the data, its merit is also its weakness because it doesn't use observations specific to a particular domain as we do for scale selection. It may be an interesting exercise to implement this method and compare its performance to our approach.

Acknowledgements

The authors would like to thank Prof. Fatin Sezgin from Bilkent University for his contributions on the statistical analysis of the evaluation data.

References

Babaud, J.; Witkin, A. P.; Baudin, M.; and Duda, R. O. 1986. Uniqueness of the gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8:26–33.

Banks, M., and Cohen, E. 1990. Realtime spline curves from interactively sketched data. In *SIGGRAPH, Symposium on 3D Graphics*, 99–107.

Bentsson, A., and Eklundh, J. 1992. Shape representation by multiscale contour approximation. *IEEE PAMI 13, p. 85–93, 1992.*

Calhoun, C.; Stahovich, T. F.; Kurtoglu, T.; and Kara, L. B. 2002. Recognizing multi-stroke symbols. *In AAAI 2002Spring Symposium Series, Sketch Understanding.*

Lindeberg, T. 1996. Edge detection and ridge detection with automatic scale selection. *ISRN KTH/NA/P–96/06–SE, 1996.*

Rattarangsi, A., and Chin, R. T. 1992. Scale-based detection of corners of planar curves. *IEEE Transactionsos Pattern Analysis and Machine Intelligence* 14(4):430–339.

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

Schneider, P. 1988. Phoenix: An interactive curve design system based on the automatic fitting of hand-sketched curves. Master's thesis, University of Washington.

Sezgin, T. M.; Stahovich, T.; and Davis, R. November 2001. Sketch based interfaces: Early processing for sketch understanding. *Proceedings of PUI-2001.*

Siegel, S. 1956. Nonparametric statistics: For the behavioral sciences. *McGraw-Hill Book Company*.

Witkin, A. 1983. Scale space filtering. *Proc. Int. Joint Conf. Artificial Intell., held at Karsruhe, West Germany, 1983, published by Morgan-Kaufmann, Palo Alto, California*.

Yuille, A. L., and Poggio, T. A. 1986. Scaling theorems for zero crossings. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8:15–25.