Assessing the Behaviour of Polygonal Approximation Algorithms

Paul L. Rosin
Department of Information Systems and Computing
Brunel University
Middlesex
UK
email: Paul.Rosin@brunel.ac.uk

Abstract

In a recent paper we described a method for assessing the accuracy of polygonal approximation algorithms [16]. Here we develop several measures to assess the stability of such approximation algorithms under variations in their scale parameters. A monotonicity index is introduced that can be applied to analyse the change in the approximation error or the number of line segments against increasing scale. A consistency index quantifies the variation in results produced at the same scale by an algorithm (but with different input parameter values). Finally, the previously developed accuracy figure of merit is calculated and averaged over 21 test curves for different parameter values to obtain more reliable scores.

1 Introduction

Recently there has been a surge of interest in the performance evaluation of computer vision algorithms and systems. Whereas in previous years comparative analysis often consisted solely of the visual inspection of various algorithms, nowadays there is more emphasis on quantitative assessment. This can take the form of analytic or Monte Carlo error propagation and sensitivity analysis, or various indices describing general or specific characteristics of the algorithm or task.

For the problem of segmenting curves into straight lines there has been relatively little development or application of performance evaluation. Some recent work by Ji and Haralick [9] on a curve segmentation algorithm based on intersecting two lines uses error propagation to determine the effect that noise on the data has on the orientation of the fitted lines. In addition, they manually determined a groundtruth set of edge data which enabled them to calculate breakpoint misdetection and false alarm rates. These rates were then plotted against the algorithm parameters (window length and angle threshold) so that the sensitivity of the algorithm's performance relative to changes in these parameters could be assessed. Previously Zhang *et al.* [18] also systematically varied the parameters of the data so that the algorithm's accuracy could be measured against the arc length and the subtended angle of the corner.

Kadonaga and Abe [10] also incorporated human assessment; theirs consisted of 1/ evaluation of the algorithms' results by a human panel, and 2/ a set of target segmentations obtained by 18 human subjects. The first directly produced an assessment value while for the second the breakpoints produced by the various algorithms were scored according to how similar they were to human selected breakpoints. The score took into account (in a rudimentary manner) the number of humans that selected similar breakpoints, and the degree of certainty of the human's selection. To measure the algorithms' robustness their performance was measured on input data which was transformed by rotation, scaling, and reflection.

Rosin [16] compared various algorithms according to criteria such as integral squared error (ISE), maximum deviation, etc. Previously a problem with this approach was that if the algorithms produced different numbers of breakpoints their errors could not be meaningfully compared. This was circumvented by comparing all the algorithms against the optimal segmentation for the appropriate number of lines rather than directly against each other. The assessment value consisted of the geometric mean of two factors: fidelity and efficiency. Fidelity measured how well the suboptimal polygon fitted the curve relative to the optimal polygon in terms of the approximation error. Efficiency measured how compact the suboptimal polygonal representation of the curve was, relative to the optimal polygon which incurred the same error.

The work presented here was originally motivated by a new class of methods for finding polygonal approximations of curves. Although the results were initially promising, several anomalies showed up. This led to the development of some performance measures in order to quantify these problems. In this context section 2 describes the measures and section 3 applies them to some well established algorithms from the literature.

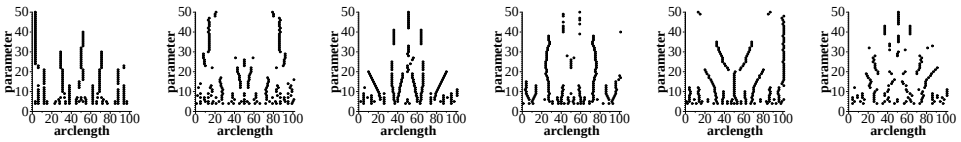
2 Assessment Measures

To verify the robustness of the algorithms their behaviour under various conditions needs to be determined. There are two factors that can be altered: the algorithm parameter values (a chord length, i.e. window size), and the data. To visualise the first we generated scale-space type plots which are formed by plotting for each parameter value tested the breakpoint indices produced by six versions of the algorithm. Observing these plots (figure 1) we note several types of behaviour of the algorithms' selection of breakpoints:

- oscillation – The location of some breakpoints moves back and forth as the chord size is gradually increased (e.g. figure 1a and 1e).
- systematic drift – The location of some breakpoints systematically moves in one direction as the chord size increases (e.g. figures 1c and 1e).
- general shift – Other breakpoints move with increasing chord size, but their shift is neither constrained to one direction, nor regularly oscillating (e.g. figure 1d).

- discontinuities – In several cases (e.g. figures 1b, 1d, and 1f) breakpoints suddenly jump to a new distant position when the chord size is incremented slightly.
- new breakpoints – Sometimes additional breakpoints are created as the chord size increases (figures 1c and 1f). This is counter to our expectation that increasing the scale parameter should generally decrease the number of detected features.

Quantifying these behaviours might prove laborious, requiring tracking points through scale-space. Moreover, in some cases it is unclear whether the behaviour is necessarily undesirable or could be just a natural outcome from the specific data set. For instance, does systematic drift indicate improved localisation or merely algorithm instability? Likewise, discontinuities might indicate instability of the algorithm or might occur due to a transition between natural scales [16]. Therefore to simplify matters we collapse the scale-space plots into 1D graphs enabling some computationally simple and unambiguous measures to be extracted.



(a) (b) (c) (d) (e) (f)

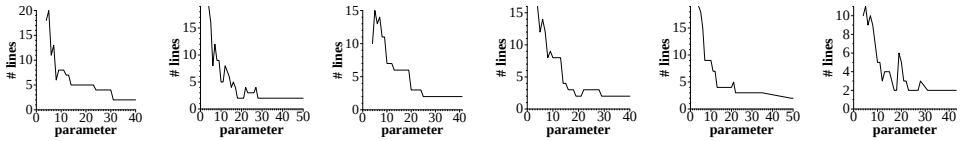
Figure 1: Scale-space plots of breakpoints

2.1 Monotonicity Measure

The input parameter of the algorithms evaluated above is effectively a scale parameter. Therefore, if the number of lines (or equivalently the number of breakpoints) produced by an algorithm is plotted against the input parameter then we would generally expect to see a monotonically decreasing curve since at larger scales there are fewer dominant points. Likewise, for some error measure (e.g. ISE) the error value should monotonically decrease as the input parameter increases. For the six algorithms these two curves are plotted in figure 2 and figure 3 where it can be seen that in fact they are not monotonic. This can pose a problem when using an algorithm since it makes it difficult to select appropriate parameters (either manually or automatically) if the effect of changing these parameters is not predictable.

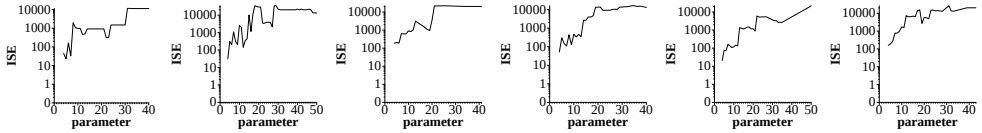
To quantify the degree of this irregularity we introduce a monotonicity measure. It involves calculating the amount of decrease T_- over the plot which we determine by integrating the gradient over all negative gradient values. Likewise, we integrate the gradient over all positive gradient values to determine the amount of increase T_+. For the discrete plot (x_i, y_i) this effectively becomes

$$\triangle y_i \quad = \quad y_i - y_{i-1}$$



(a) (b) (c) (d) (e) (f)

Figure 2: Number of lines versus chord size



(a) (b) (c) (d) (e) (f)

Figure 3: Integral square error versus chord size

$$T_- = -\sum_{\forall \triangle y_i < 0} \triangle y_i$$
$$T_+ = \sum_{\forall \triangle y_i > 0} \triangle y_i.$$

However, since we would expect the error plot to be steeper when the error is larger we normalise the values by their height h_i

$$h_i = \frac{y_i + y_{i-1}}{2}$$
$$T_- = -\sum_{\forall \triangle y_i < 0} \frac{\triangle y_i}{h_i}$$
$$T_+ = \sum_{\forall \triangle y_i > 0} \frac{\triangle y_i}{h_i}.$$

The gradient of the line versus chord size also decreases with increasing chord size, and so we apply this normalisation for both plots. For all but the worst cases the amount of decrease will exceed the amount of increase, and so we combine the values as

$$M_D = \left(1 - \frac{T_+}{T_-}\right) \times 100.$$

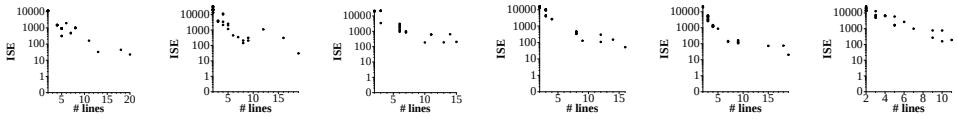
This will generally produce a value in the range $[0, 100]$ (but will be negative for extremely bad instances), where perfect monotonicity scores 100. As well as measuring monotonic decrease we can measure monotonic increase too

$$M_I = \left(1 - \frac{T_-}{T_+}\right) \times 100.$$

A useful property of the measure is that it is independent of scaling of abscissa. This enables it to be applied to different algorithms whose parameter values have different scales, even if they are related by some non-linear function.

2.2 Consistency Measure

We have seen in figures 2 and 3 how the number of lines and the approximation error can increase even as the chord size increases. Another aberration we have noted is that when different parameter values produce polygons with the same number of lines these polygons may differ. This applies both to adjacent and non-adjacent parameter values. Since these different approximations tend to have different errors we can show this inconsistency by plotting the errors against parameter values; see figure 4.



(a) (b) (c) (d) (e) (f)

Figure 4: ISE versus number of lines

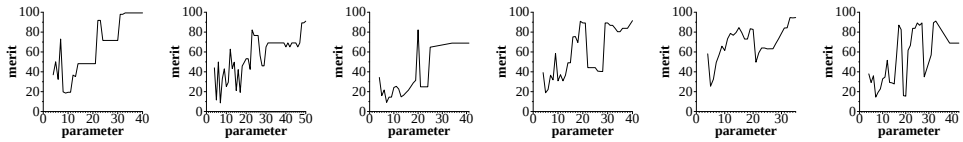
We propose a measure to quantify this inconsistency. It uses the range of error values r_i produced by the algorithm for a given number of lines i in the approximating polygon. Again, like the monotonicity measure, the value is normalised. It is divided by the height of the midpoint of the span h_i. The final consistency measure is summed over all numbers of lines produced by the algorithm

$$C = \sum_i \frac{r_i}{h_i}$$

2.3 Other Measures

2.3.1 Figure of merit

In our previous work a figure of merit was generally calculated for an algorithm at just one or two parameter settings. However, when merit is plotted against parameter value for the six new algorithms we see that the graph fluctuates considerably (figure 5). In other words, careful selection of the algorithm's parameter values can produce a misleadingly favourable rating. This suggests that the mean and variance of the figure of merit over a range of scales would provide a more meaningful assessment.



(a) (b) (c) (d) (e) (f)

Figure 5: Merit versus chord size

(a) Douglas & Peucker (b) Ramer

Figure 6: Breakpoint stability under progressive curve deletion

2.3.2 Endpoint shift stability

So far we have just discussed measuring the effects of varying the algorithms' parameters. Obviously we can also perturb the data and measure the stability of the number of breakpoints, ISE, breakpoint location, merit, etc. We will not pursue this course of action in this paper, but will give just one example demonstrating the effect of systematically deleting increasing amounts off the beginning of a curve. While this will produce limited effects on the algorithms that operate on local windows, others based on sequential or recursive subdivision are potentially sensitive to shifts in the endpoints. For a fixed threshold (set at 2) the plots of breakpoints for the Douglas & Peucker [4] and Ramer [14] algorithms are shown (figure 6). All points to the left of the diagonal have been deleted. In this case oscillation or shift is clearly undesirable, and therefore would be an appropriate measure of endpoint stability. At each level of deletion d the shift s_b^d at each breakpoint b is measured. Let the length of the curve after deletion be l_d, the number of breakpoints at this level be n_d, and the number of levels of deletion be m. Then we calculate the shift measure as

$$S = \left(1 - \frac{1}{m}\sum_d \sum_b \frac{s_b^d}{n_d l_d}\right) \times 100.$$

Since Douglas & Peucker's algorithm shows several large shifts it only scores 89.85 while Ramer's scores 99.75 as only small shifts occurred. Note that the shift measure does not take into account the frequent change in numbers of breakpoints that Ramer's algorithm displays with increasing levels of deletion.

3 Experiments

We apply the measures to 16 algorithms from the literature as well as five optimal algorithms [16]; all incorporate a parameter that effectively controls scale. The algorithms form a good cross section, ranging over 25 years and based on a diversity of methods (see table 1). By their very definition some methods (e.g. Ramer) are guaranteed to avoid any breakpoint shift. At the opposite end of the spectrum the simple method of subdividing the curve into evenly sized segments inevitably displays continual breakpoint shift while Deveau's algorithm exhibits severe systematic drift. The plots of number of lines against parameter also show a wide range of behaviour, and table 2 details their monotonicity values. Since the input parameter for some algorithms corresponds to the desired number of output lines

Authors	**Method**
Cheng & Hsu [1]	curvature approximation
Deguchi & Aoki [2]	regularisation
Deveau [3]	tolerance band
Douglas & Peucker [4]	sequential subdivision
Inesta *et al.* [8]	symmetry
Ji & Haralick [9]	statistical hypothesis test
Fu *et al.* [6]	curvature approximation
Hu & Yan [7]	local pattern rewrite rules and curvature
Melen & Ozanian [11]	curvature and first derivative of curvature
Pei & Horng [12]	spatial shift from smoothing
Phillips & Rosenfeld [13]	arc/chord distances
Ramer [14]	recursive subdivision
regular	subdivision into evenly sized segments
Rosenfeld & Johnston [15]	curvature approximation
Wu & Levine [17]	simulation of electric charge density distribution
Zhang *et al.* [18]	Bayes theory
E_1	optimal minimum E_1 error
E_2	optimal minimum E_2 (ISE) error
E_∞	optimal minimum E_∞ (maximum deviation) error
E_L	optimal minimum length
E_θ	optimal minimum difference in orientation

Table 1: Polygonal approximation methods tested

then their graphs are linear (i.e. the optimal and "regular" algorithms). Other algorithms also produce perfectly monotonic plots (Deveau, Douglas & Peucker, and Ramer) although the correspondence between input parameter and number of lines is non-linear. The other extreme is shown by Phillips & Rosenfeld's algorithm which shows strong fluctuations. Fu *et al.* and Melen & Ozanian's algorithms also appear unstable, but to a lesser degree.

In a similar manner the monotonicity of ISE against parameter is measured and given in table 2. This time only the optimal ISE minimising algorithm maintains continuous reduction in ISE with continuous change of input parameter. In part this is because many of the other algorithms use the maximum deviation as an error criterion rather than ISE. Nonetheless, many algorithms appear nearly monotonic (e.g. Deveau, Douglas & Peucker, Hu & Yan, and Ramer) while others show extreme fluctuations (e.g. Phillips & Rosenfeld, Deguchi & Aoki, and Melen & Ozanian).

The consistency of errors plotted against numbers of lines is given in table 2. The optimal algorithms and others such as Douglas & Peucker, Hu & Yan, and the simple algorithm which show no or little vertical doubling of points achieve good (i.e. low) scores. On the other hand, Cheng & Hsu, Deguchi & Aoki, Rosenfeld & Johnston, and others with considerable vertical duplication receive poor (large) scores.

Finally, many plots of merit against parameter show substantial variations. We can see for example from the mean and standard deviations in table 3 that E_1 has a good score for a range of parameter values while the Rosenfeld & Johnston algorithm has a mediocre score with considerable standard deviation, whereas

Method	**M(Line)**	**M(ISE)**	**Consistency**	**Merit**	
				μ	σ
Cheng & Hsu	94	49	5.88	44	19.4
Deguchi & Aoki	20	35	4.09	28.9	17.05
Deveau	100	96	4.46	44	9.5
Douglas & Peucker	100	99	0.00	78	9.8
Fu *et al.*	20	8	2.83	63	18.0
Inesta *et al.*	53	79	0.38	30.1	14.7
Hu & Yan	91	96	0.00	70	11.7
Ji & Haralick	87	58	1.64	25	8.3
Meloza & Ozanian	38	25	4.53	48	28.3
Pei & Horng	71	81	1.10	34	5.4
Phillips & Rosenfeld	4	1	4.57	27	13.4
Ramer	100	95	0.00	80	15.4
regular	100	50	0.00	33	14.9
Rosenfeld & Johnston	58	53	5.62	47	16.5
Wu & Levine	69	54	2.27	29	17.6
Zhang *et al.*	100	71	2.86	49	7.2
optimal E_1	100	81	0.00	87	8.7
optimal E_2	100	100	0.00	100	0.0
optimal E_∞	100	98	0.00	83	10.6
optimal E_L	100	87	0.00	82	11.8
optimal E_θ	100	66	0.00	39	19.1

Table 2: Assessment of various algorithms applied to Teh and Chin's test curve

Pei & Horng receive a poorer mean score but are more consistent over different parameter values.

These same measures were calculated for 21 test curves by running them over a range of parameter settings – in most cases $[4 - 45]$. The mean and standard deviations calculated over the full set are given in table 3. Of course, since all the algorithms are effectively being compared against the E_2 metric for all algorithms of performance then only the optimal E_2 algorithm can receive across the board top marks. Not surprisingly, the similar E_1 algorithm does well. Several other algorithms such as Douglas & Peucker, possibly Hu & Yan, and especially Ramer behave well most of the time although their merit scores lag the optimal E_2 results considerably at times. It should be noted that all the algorithms exhibit large variations in merit over varying parameter values and data sets.

4 Discussion

We have described several quantitative criteria for measuring the stability of the performance of polygonal approximation algorithms. They are simple to compute and require no parameters except the range of tested algorithm parameters. In most cases this will be determined by the algorithms themselves. For example, for Ramer's algorithm the window within which the straight line is fitted must be greater than two to allow for some deviation from that line, and cannot be larger than the number of points in the curve. The criteria implemented are not

Method	**M(Line)**		**M(ISE)**		**Consistency**		**Merit**	
	μ	σ	μ	σ	μ	σ	μ	σ
Cheng & Hsu	72	13.2	28	15.5	6.1	2.64	16	11.3
Deguchi & Aoki	17	10.8	18.5	23.0	6.5	5.37	32	17.15
Deveau	100	0.0	89	13.4	2.3	0.62	41	24.2
Douglas & Peucker	99	2.8	98	5.6	0.1	0.27	64	12.4
Fu *et al.*	8	19.5	1	19.7	2.6	0.80	51	17.3
Hu & Yan	86	13.9	89	11.9	0.0	0.00	56	13.4
Inesta *et al.*	4	53.3	0	48.8	1.6	0.60	37	13.3
Ji & Haralick	81	25.1	68	18.0	1.7	1.32	22	13.9
Meloza & Ozanian	83	13.7	61	18.4	1.9	1.27	58	20.0
Pei & Horng	73	15.7	74	12.9	2.1	1.43	40	15.2
Phillips & Rosenfeld	29	27.1	26	17.8	5.5	1.81	30	12.6
Ramer	100	0.0	100	1.1	0.0	0.00	74	19.7
regular	100	0.0	66	8.8	0.0	0.00	27	9.0
Rosenfeld & Johnston	85	13.4	71	9.1	2.6	1.27	52	25.0
Wu & Levine	59	31.0	52	19.6	2.1	0.95	32	15.6
Zhang *et al.*	96	5.6	60	13.9	2.3	1.07	36	14.2
optimal E_1	100	0.0	97	2.8	0.0	0.00	88	9.0
optimal E_2	100	0.0	100	0.0	0.0	0.00	100	0.0
optimal E_∞	100	0.0	95	3.1	0.0	0.00	77	10.4
optimal E_L	100	0.0	80	14.2	0.0	0.00	63	13.6
optimal E_θ	100	0.0	77	16.4	0.0	0.00	40	18.9

Table 3: Assessment of various algorithms applied to 21 test curves

exhaustive. Several others were discussed with reference to the scale-space plots. However, in these cases it was difficult to consistently interpret them.

Although the testing was extensive (over 24000 program runs were necessary) it is still incomplete on several counts. First, small test curves (100–200 points) were used to speed up the computation. This restricted the possible window sizes which leads to insufficient data for some algorithms to perform reliable statistical analysis. Second, some of the (standard) synthetic test curves have few or no perturbations which does not reflect the nature of many realistic image curves. Therefore additional testing is required to assess the stability and accuracy of the algorithms under increasing levels of noise. Third, only the scale parameter was varied. For those algorithms with multiple parameters these had to remain fixed for practical reasons.

Finally, the merit scores were calculated with respect to E_2 error. However, there is no *a priori* guarantee that this is the most appropriate error model – it just happens to be commonly used in practice. The other metrics tested with the optimal algorithm also generally appear visually reasonable, even though they only get a moderate mean merit rating. In fact some psychophysical evidence points to E_∞ as an appropriate measure [5] while specific tasks can be shown to favour particular choices of error [16].

Acknowledgements

I would like to thank the following for providing software, results, and advice: P. Cornic, T. Deveau, A. Fu, R. Haralick, J. Hu, J. Inesta, Q. Ji, M. Levine, D. Lutz, T. Melen, T. Ozanian, and K. Wu.

References

[1] F.-H. Cheng and W.-H. Hsu. Parallel algorithm for corner finding on digital curves. *Pattern Recognition Letters*, 8:47–54, 1988.

[2] A. Deguchi and S. Aoki. Regularized polygonal approximation for analysis and interpretation of planar contour figures. In *International Conference on Pattern Recognition*, volume 1, pages 865–869, 1990.

[3] T.J. Deveau. Reducing the number of points in a plane curve representation. In *Proc. AutoCarto VII*, pages 152–160, 1985.

[4] D.H. Douglas and T.K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10:111–122, 1973.

[5] D.H. Foster, D.R. Simmons, and M.J. Cook. The cue for contour-curvature discrimination. *Vision Research*, 33:329–341, 1993.

[6] A.M.N. Fu, H. Yan, and K. Huang. A curve bend function method to characterize contour shapes. *Pattern Recognition*, 30(10):1661–1671, 1997.

[7] J.M. Hu and H. Yan. Polygonal-approximation of digital curves based on the principles of perceptual organization. *Pattern Recognition*, 30(5):701–718, 1997.

[8] J.M. Iñesta, M. Buendía, and M.A. Sarti. Local symmetries of digital contours from their chain codes. *PR*, 29(10):1737–1749, 1996.

[9] Q. Ji and R.M. Haralick. Breakpoint detection using covariance propagation. *IEEE Trans. PAMI*, 20(8):845–851, 1998.

[10] T. Kadonaga and K. Abe. Comparison of methods for detecting corner points from digital curves. In *Int. Workshop on Graphics Recognition*, pages 3–12, 1995.

[11] T. Melen and T. Ozanian. A fast algorithm for dominant point detection on chain-coded contours. In *Proc. 5th Int. Conf. Computer Analysis of Images and Patterns*, pages 245–253, 1993.

[12] S.C. Pei and J.H. Horng. Corner point detection using nest moving average. *Pattern Recognition*, 27:1533–1537, 1994.

[13] T.Y. Phillips and A. Rosenfeld. A method of curve partitioning using arc-chord distance. *Pattern Recognition Letters*, 5:285–288, 1987.

[14] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer, Graphics and Image Processing*, 1:244–256, 1972.

[15] A. Rosenfeld and E. Johnston. Angle detection on digital curves. *IEEE Trans. Comp.*, 22:875–878, 1973.

[16] P.L. Rosin. Techniques for assessing polygonal approximations of curves. *IEEE Trans. PAMI*, 19(6):659–666, 1997.

[17] K.N. Wu and M.D. Levine. 2d shape segmentation: A new approach. *Pattern Recognition Letters*, 17(2):133–140, 1996.

[18] X. Zhang, R.M. Haralick, and V. Ramesh. Corner detection using the MAP technique. In *Proc. 12th Int. Conf. Pattern Recognition*, pages 549–551, 1994.