PERGAMON

Pattern Recognition 36 (2003) 505–518

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Assessing the behaviour of polygonal approximation algorithms

Paul L. Rosin*

Department of Computer Science, Cardiff University, Queen's Buildings, Newport Road, PO Box 916, Cardiff, CF24 3XF, UK

Received 21 August 2001; accepted 25 March 2002

Abstract

In a recent paper, we described a method for assessing the accuracy of polygonal approximation algorithms (IEEE Trans. PAMI 19(6) (1997) 659). Here, we develop several measures to assess the stability of such approximation algorithms under two classes of variations, namely perturbations of the data and changes in the algorithms' scale parameters. In order to quantify the former, a break-point stability index is defined and tested. For the latter, two indices are introduced; (1) a monotonicity index is applied to analyse the change in the approximation error or the number of line segments against increasing scale, and (2) 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.

Keywords: Evaluation; Polygonal approximation; Curve; Breakpoint

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 [1]. 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 despite the continuing interest in developing new curve segmentation algorithms.[1] Some recent work by Ji and Haralick [7] 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. [8] 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 [9] 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,

* Tel.: +44-29-2087-5585; fax: +44-29-2087-4598.
E-mail address: paul.rosin@cs.cf.ac.uk (P.L. Rosin).

[1] For instance, in 1998 alone the approaches to polygonal approximation included: minimising rate distortion using dynamic programming [2], genetic algorithms [3], finite element analysis [4], scale-space analysis [5], mathematical morphology [6], and statistics [7].

0031-3203/02/$22.00
PII: S0031-3203(02)00076-6

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.

Unfortunately the quality and nature of the ground truth is problematic with the above approaches. Previous attempts to acquire an ideal set of breakpoints show that humans disagree over the location and number of breakpoints [9,10]. Zhang et al. label corners as correctly detected if they lie within five pixels of a ground truth corner (generated by only one human subject), but this does not appear to be a rigorous solution. The same issue had to be tackled by Rosin [11] who 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 [12]. 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. The two main types of performance measures in use are goal-directed or goal-independent. The advantage of being task-specific is that it is generally easier to accurately assess the quality of an algorithm's performance. However, the disadvantage is that the measures do not necessarily give any indication of how well the algorithms will perform when used to help achieve a different task. Thus the algorithms need to be retested for each task. We therefore prefer to develop general-purpose (i.e. task independent) performance indicators that measure specific behaviours of the algorithms. For any task some of these behaviours may be more important than others, and so an algorithm's suitability can be determined by identifying the behaviours relevant to a particular task and checking their individual performance ratings.

Our measures fall into two classes and can be categorised as measuring the effect of (1) changes in the data, and (2) changes in the algorithm. More specifically the data is altered by deleting points, simulating occlusion, while the algorithm is modified by changing its scale parameter. Perturbations of the data are commonplace in practice; it is desirable that an algorithm is insensitive to them. For instance, in many vision applications involving point correspondence such as model matching, pose estimation, and tracking, it is not just the accuracy but also the stability of corner or line detection algorithms that is extremely important. Altering an algorithm's parameters is likely to occur when it is being tuned by a user for a specific task. It is convenient for the user to assume that an algorithm's output can be easily predicted from previous experience on other data and with other parameters. Therefore, the consistency of an algorithm's behaviour under parameter changes is a useful indication of its ease of use in developmental stages.

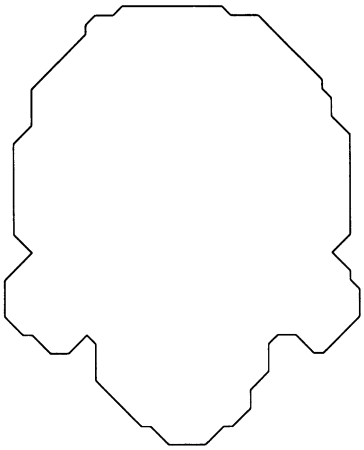

Fig. 1. Test curve from Teh and Chin [13].

2. Assessment measures

In this section the set of assessment measures are developed. Initially, we show their application to the output of polygonal approximation algorithms resulting from their application to a single curve (the well-known example in Fig. 1 taken from Teh and Chin [13]). This enables us to visually validate the correctness of the measures, while the next section concentrates on the numerical evaluation of the polygonal approximation algorithms applied to a larger data set.

We start by visualising the effects of changes in the algorithm's parameter values by generating scale-space type plots (see Fig. 2). These are formed by varying the algorithm's scale parameter[2] and plotting the resulting breakpoint indices produced by the algorithm. The scale parameter typically corresponds to a window size for performing operations such as smoothing, non-maximal suppression, etc. or is a size threshold such as maximum allowable deviation between the curve and its approximation. Sixteen

[2] In most cases the scale parameter range was set to 4–45.

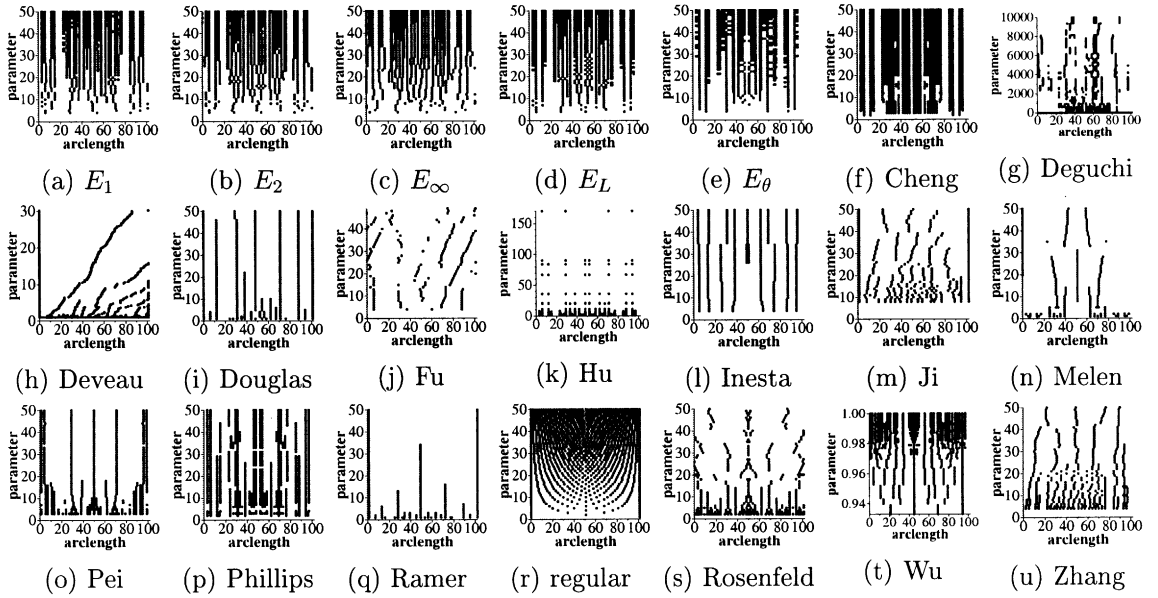

Fig. 2. Scale-space plots of output breakpoint positions versus input scale parameters generated from Teh and Chin's test curve by the various algorithms. The breakpoint positions are the indices of the curve's pixels (in the range 0, 102) while the scale parameters vary according to the algorithms.

algorithms from the literature[3] were used as well as five optimal algorithms [11]. These form a good cross section, ranging over 25 years and are based on a diversity of methods (see Table 1).

Given that the problem of segmenting curves appears at least superficially or intuitively to be a straightforward task this would suggest that out of the enormous number of algorithms available in the literature many are likely to be mere variations on each other. However, the scale-space plots shown here display sufficient variety to counter that argument (at least for the tested algorithms).

We note several types of behaviour of the algorithms' selection of breakpoints:

- Oscillation—The location of some breakpoints moves back and forth as the scale varies (e.g. Figs. 2l and u).
- Systematic drift—The location of some breakpoints systematically moves in one direction as the scale increases. (e.g. Figs. 2h and j).
- Discontinuities—In several cases (e.g. Figs. 2m,n and u) breakpoints suddenly jump to a new distant position when the scale parameter is incremented slightly.
- New breakpoints—Sometimes additional breakpoints are created as the scale increases (Figs. 2c and n). This behaviour is counter to our expectation that increasing the scale parameter should generally decrease the number of detected features.

Of course, by their very definition some methods (e.g. Ramer, Fig. 2q) are guaranteed to avoid any scale-induced breakpoint shift. At the opposite end of the spectrum the simple method of subdividing the curve into evenly sized segments (labelled "regular") inevitably displays continual breakpoint shift while Deveau's algorithm (Fig. 2h) exhibits severe systematic drift. There are obviously similarities between the optimal algorithm scale-space plots, despite the fact that very different criteria are being optimised.

Quantifying these behaviours could 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 [11]. Therefore, to simplify matters we collapse the scale-space plots into 1D graphs enabling some computationally simple and unambiguous measures to be extracted.

[3] Some changes to the algorithms were made: (1) In Cheng's algorithm [14] a slight modification had to be made when calculating $W(d)$ to cater for $d = \pm 4$. (2) Random selection was used to optimise Deguchi and Aoki's criterion [15], so that only suboptimal results are reported. (3) For Deveau's algorithm [16] the parameter TOL was selected as the scale parameter while the others were fixed to ANG = 110 and NW = 20. (4) Only the first stage of Phillips and Rosenfeld's algorithm [17] was implemented.

Table 1
Polygonal approximation methods tested

Authors	Method
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
Cheng and Hsu [14]	Curvature approximation
Deguchi and Aoki [15]	Regularisation
Deveau [16]	Tolerance band
Douglas and Peucker [18]	Sequential subdivision
Fu et al. [21]	Curvature approximation
Hu and Yan [24]	Local pattern rewrite rules and curvature
Inesta et al. [22]	Symmetry
Ji and Haralick [7]	Statistical hypothesis test
Melen and Ozanian [26]	Curvature and first derivative of curvature
Pei and Horng [27]	Spatial shift from smoothing
Phillips and Rosenfeld [17]	Arc/chord distances
Ramer [19]	Recursive subdivision
Regular	Subdivision into evenly sized segments
Rosenfeld and Johnston [20]	Curvature approximation
Wu and Levine [23]	Simulation of electric charge density distribution
Zhang et al. [8]	Bayes theory

2.1. Monotonicity measure

If the number of lines (or equivalently the number of breakpoints) produced by an algorithm is plotted against the input scale 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 increase as the input parameter increases. For the sixteen algorithms these two curves are plotted in Figs. 3 and 4, 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

$$\Delta y_i = y_i - y_{i-1}, \quad T_- = -\sum_{\forall \Delta y_i < 0} \Delta y_i, \quad T_+ = \sum_{\forall \Delta y_i > 0} \Delta y_i.$$

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

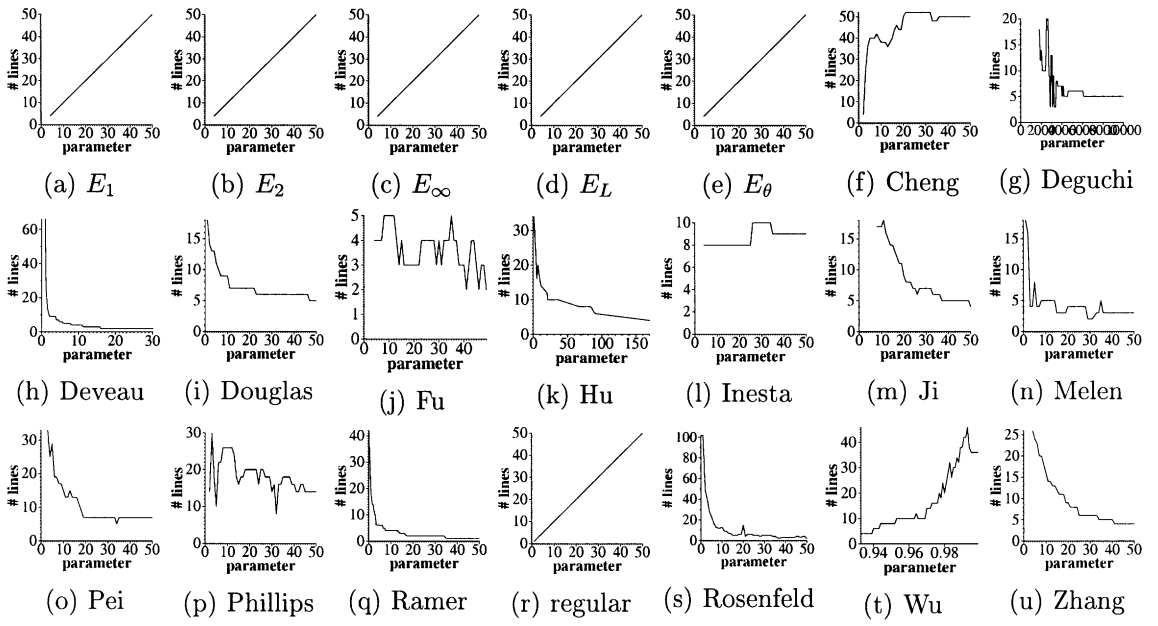

Fig. 3. Number of output lines produced by various algorithms plotted against the input scale parameter.

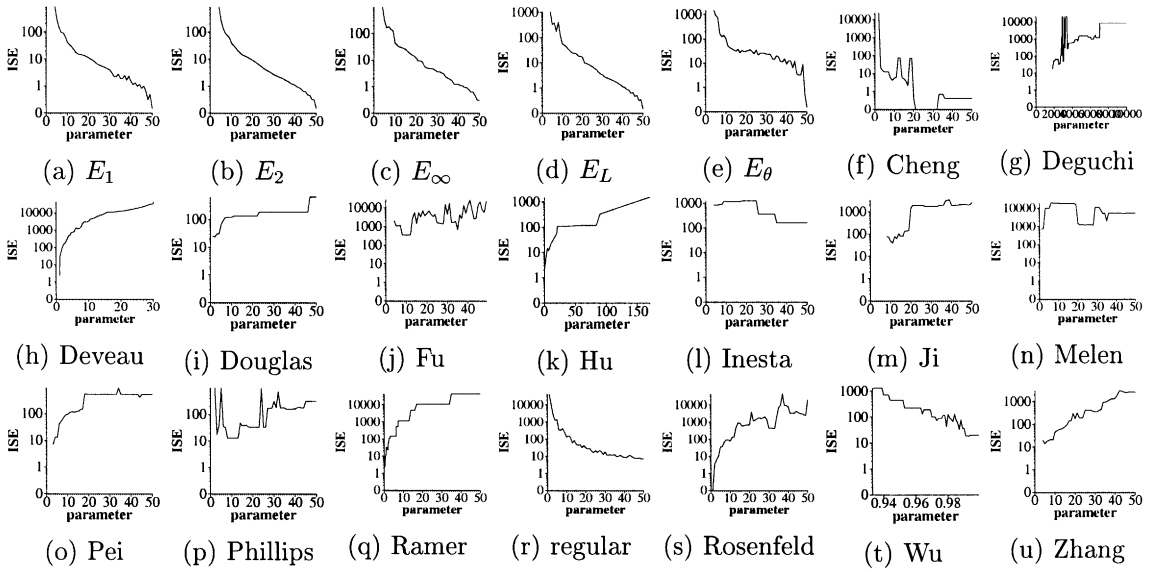

Fig. 4. Integral square error (ISE) produced by various algorithms plotted against the input scale parameter.

height h_i

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

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.

Checking the calculated monotonicity values in Table 2 we can see that they match our expectations gained from a visual analysis of the plots in Figs. 3 and 4. For instance, the perfectly monotonic results (e.g. Figs. 3a and q) receive top scores, algorithms producing mostly monotonic results (e.g. Figs. 3f and 4k) receive high scores, and finally those algorithms which are very irregular (e.g. Figs. 3j and 4p) are given low scores.

2.2. Consistency measure

We have seen in Figs. 3 and 4 how both the number of lines and the approximation error can decrease even as the scale parameter 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 the number of lines; see Fig. 5. This is obviously undesirable behaviour. It would be more convenient for a user of the algorithm to know that, given the limited number of lines used, an output segmentation generated according to an input parameter is the best representation of the data available from the algorithm.

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, measuring the error in consistency, and so the ideal value is zero

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

Errors are plotted against numbers of lines in Fig. 5. Again the consistency results in Table 2 verify our visual impressions. For example, algorithms which show no or little vertical doubling of points achieve good (i.e. low) scores (e.g. Figs. 5a and k) while those with considerable

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

Method	M (Line)	M (ISE)	Consistency error	Merit μ	σ	Stability error
Optimal E_1	100	81	0.00	87	8.7	2.39
Optimal E_2	100	100	0.00	100	0.0	2.09
Optimal E_∞	100	98	0.00	83	10.6	3.45
Optimal E_L	100	87	0.00	82	11.8	1.99
Optimal E_θ	100	66	0.00	39	19.1	2.15
Cheng and Hsu [14]	94	49	5.88	44	19.4	5.10
Deguchi and Aoki [15]	20	35	4.09	28	17.05	9.05
Deveau [16]	100	96	4.46	44	9.5	2.02
Douglas and Peucker [18]	100	99	0.00	78	9.8	2.30
Fu et al. [21]	20	8	2.83	63	18.0	4.15
Hu and Yan [24]	91	96	0.00	70	11.7	2.34
Inesta et al. [22]	53	79	0.38	30	14.7	4.21
Ji and Haralick [7]	87	58	1.64	25	8.3	—
Melen and Ozanian [26]	38	25	4.53	48	28.3	3.95
Pei and Horng [27]	71	81	1.10	34	5.4	2.55
Phillips and Rosenfeld [17]	4	1	4.57	27	13.4	1.94
Ramer [19]	100	95	0.00	80	15.4	3.77
Regular	100	50	0.00	33	14.9	0.00
Rosenfeld and Johnston [20]	58	53	5.62	47	16.5	2.63
Wu and Levine [23]	69	54	2.27	29	17.6	3.44
Zhang et al. [8]	100	71	2.86	49	7.2	2.31

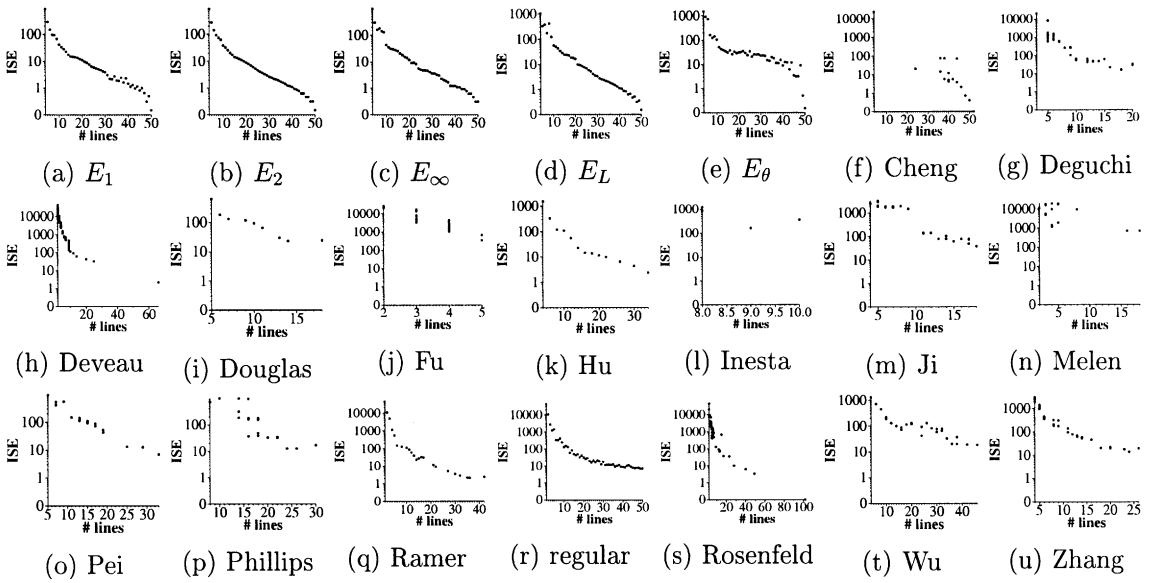

Fig. 5. The integral square error (ISE) of the output of various algorithms plotted against the number of lines they output.

vertical duplication receive poor (i.e. large) scores (e.g. Figs. 5f and p).

2.3. Endpoint 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. In this section we demonstrate the effect of systematically deleting increasing amounts off the beginning of a curve. While this will have negligible effect on the algorithms that operate on local windows, others based on sequential or recursive subdivision are potentially sensitive to shifts in the endpoints.

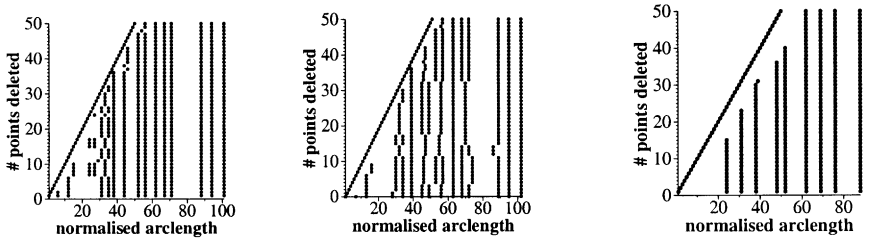

Fig. 6. Breakpoint positions under progressive curve deletion enabling their level of stability to be clearly seen.

Table 3
Breakpoint stability under progressive curve deletion

Authors	Endpoint stability	RMSE translation	RMSE orientation
Douglas and Peucker [18]	99.86	4.56	4.71
Ramer [19]	99.75	7.87	10.25
Rosenfeld and Johnston [20]	100.00	0.00	0.00

Altering the start point in this way will also produce similar effects to other variations of the data such as changing the start point on a closed curve or reverse ording the curve. We show just three examples of testing endpoint stability; for fixed parameter settings the plots of breakpoints for the Douglas and Peucker [18], Ramer [19] and Rosenfeld and Johnston [20] algorithms are plotted in Fig. 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 pixels at each breakpoint b (excepting the two corresponding to the start and finish of the curve) 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. We calculate the measure as the normalised average amount of shift

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

Since it operates on a local window Rosenfeld and Johnston's algorithm is very stable, as can be seen both from its breakpoint plot (Fig. 6a) and endpoint stability score (Table 3). Both Douglas and Peucker and Ramer's algorithm incur similar amounts of disruption, although they are distributed somewhat differently. To validate the measure we have compared it with the algorithms performances in a 2D pose estimation task. The model (i.e. the breakpoints detected on the complete curve) is matched against the 50 sets of breakpoints extracted from the shortened versions of the curve. Starting from the start of the curves a one-to-one correspondence between breakpoints is assumed; any excess of breakpoints on either of the curves is discarded. The expected pose estimate should be the identity transformation and is estimated using the least-squares method. The RMS errors calculated over the 50 matches are shown for the translation and orientation component in Table 3. They correlate well with the endpoint stability scores, verifying their effectiveness.

2.4. Data perturbation stability

To measure random perturbations along the curve a similar procedure is employed. First, several perturbed versions of the curve are generated. The breakpoints along the perturbed curves are detected, and their average displacement across the curves calculated as

$$S = \frac{1}{m}\sum_d \sum_b \frac{s_b^d}{n_d},$$

where $d = 1 \ldots m$ is the dth perturbed version of the curve. Like the consistency measure it measures the error in stability, and so the ideal value is zero. In order to simplify matters (i.e. retain an expected one-to-one correspondence between breakpoint indices) the perturbations were applied such that (1) the number of points in the curve remained constant, (2) the points remained in the integer grid, and (3) were eight-way connected. This was achieved by applying the two simple pixel rewrite rules shown in Fig. 7 along with their 90°, 180°, and 270° rotated versions. Fig. 8 shows their effect when they were applied ten times to Teh and Chin's test curve, where at each iteration they were applied to a random selection of 50% of the points. An important consideration is that the stability of an algorithm is likely to be related to the scale of analysis. That is, as a finer scale is applied then more breakpoints will be generated, whose average movement will be probably reduced. Therefore, it is important to attempt to compare the algorithms at similar scales. Our solution is to select each algorithm's scale parameters so as to produce as close as possible an average of eleven breakpoints per curve over the data set.

The plots of the sixteen segmentation algorithms' breakpoints generated from 40 perturbed versions of the curve are

Fig. 7. Pixel rewrite rules for perturbing curve data.

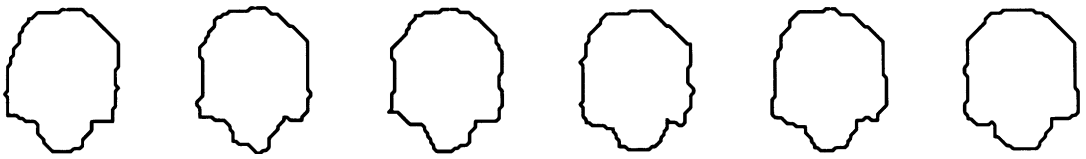

Fig. 8. Teh and Chin's test curve perturbed by applying rewrite rules.

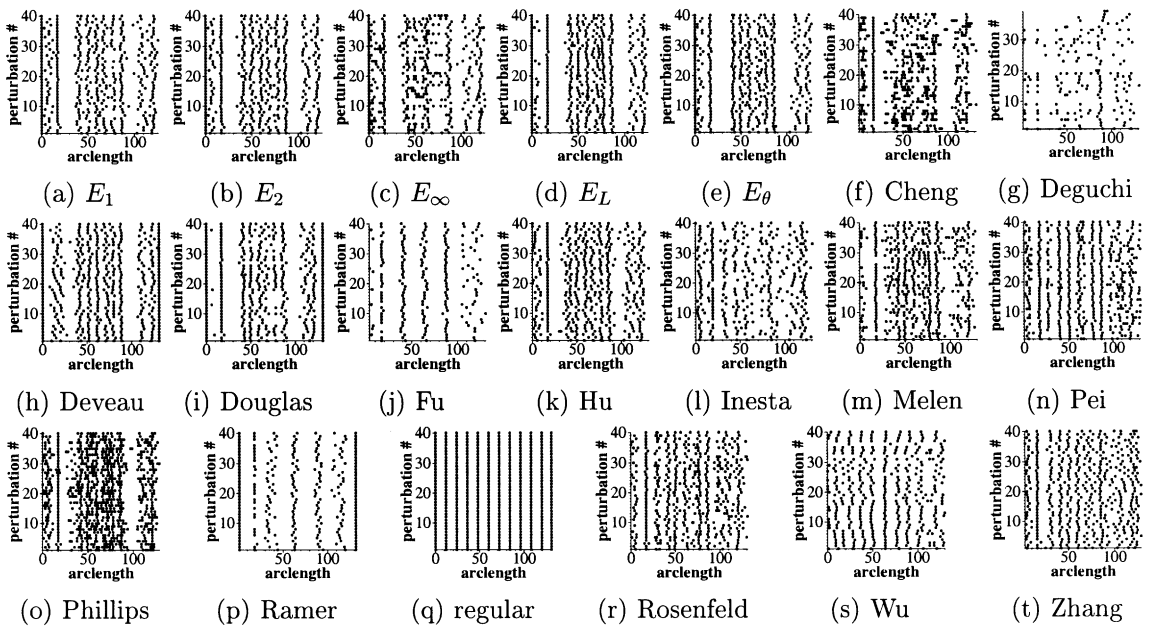

Fig. 9. Breakpoint shift induced by applying the rewrite rules to Teh and Chin's test curve.

shown in Fig. 9. It is rather more difficult to interpret these plots than the previous ones due to the variability of the results. For instance, in Figs. 9j and n we see that many of the breakpoints are stable while others are extremely unstable—overall this leads to a poor stability score in Table 2.

Again we test the validity of the stability measure by comparing it against the pose estimation task. While the perturbations were previously kept simple to enable the measure to be computed this is no longer necessary. Gaussian noise ($\sigma = 0.4$) is added to the co-ordinates of the original curve. The curve is then requantised and resampled to the image grid so that they are now of variable length. Some examples of the noisy curve are shown in Fig. 10; 40 were used in total. The RMS errors are given in Table 4 along with the stability scores. The three methods are identically ranked by stability and RMS orientation error. The stability value correctly identifies Ramer's algorithm as leading to the greatest RMS translation error, but swaps the ordering of the remaining two.

Table 4
Breakpoint stability under added noise

Authors	Noise stability	RMSE translation	RMSE orientation
Douglas and Peucker [18]	2.30	22.05	23.58
Ramer [19]	3.77	24.99	26.08
Rosenfeld and Johnston [20]	2.63	19.27	24.48

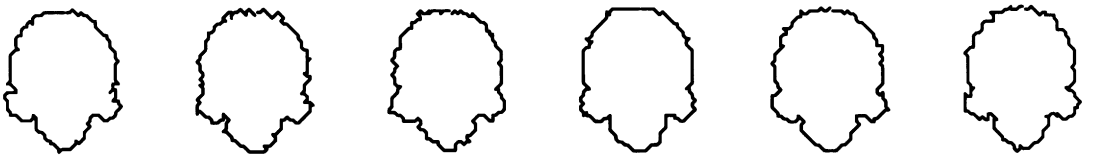

Fig. 10. Teh and Chin's test curve perturbed by adding Gaussian noise.

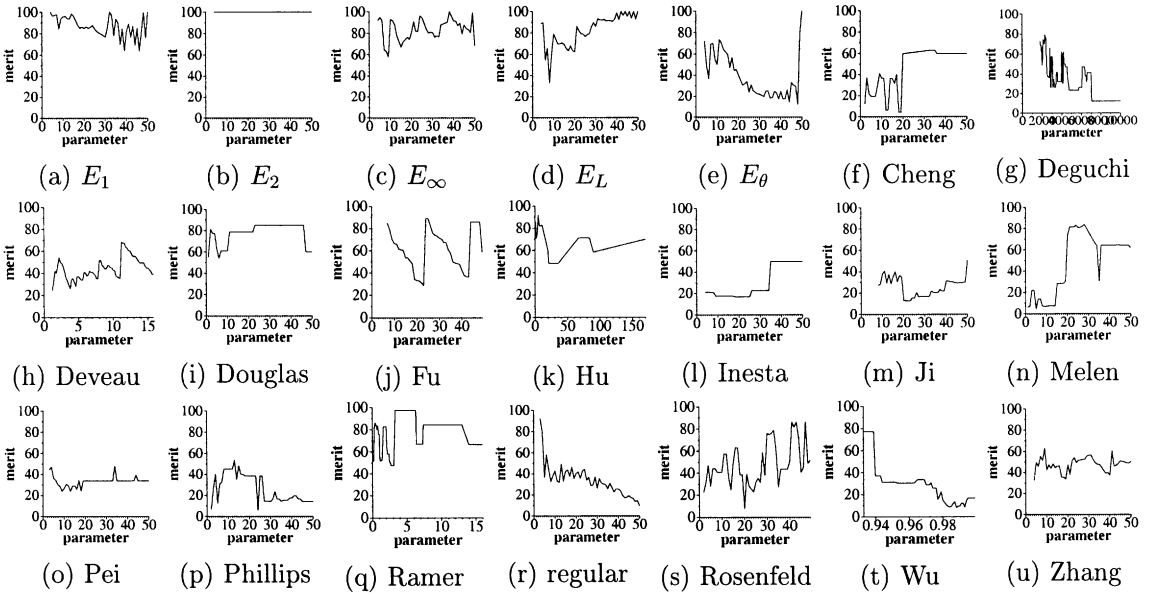

Fig. 11. Merit calculated for each algorithm's output plotted against the input scale parameter.

2.5. Figure of merit

In our previous work a figure of merit was generally calculated for an algorithm at just one or two parameter settings [11]. However, when merit is plotted against parameter value we see that the graph fluctuates considerably (Fig. 11). 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. Referring again to Table 2 we can verify that this enables us to distinguish between Rosenfeld and Johnston's algorithm (Fig. 11s) which has a mediocre score with considerable standard deviation, and Pei and Horng's algorithm (Fig. 11o) which received a poorer mean score but was more consistent over different parameter values.

3. Experiments

Now that the assessment measures have been described and their correctness has been verified to a reasonable degree by visual inspection and the pose estimation task we apply all the measures to all the algorithms for the 21 test curves shown in Fig. 12. The mean and standard deviations calculated over the full set are given in Table 5. Based on these values the various algorithms are reviewed.

3.1. Line monotonicity

Line monotonicity checks the expectation that increasing the scale parameter should not increase the number of lines. Since the input parameter for some algorithms corresponds directly to the desired number of output lines (rather than indirectly via the window size for example) then they receive perfect monotonicity scores (i.e. the optimal and "regular" algorithms). Other algorithms also produce perfect (or near perfect) monotonic plots [9,16,18,19] although the correspondence between input parameter and number of lines is non-linear. The other extreme is shown by Fu et al. [21] Inesta et al. [22] Deguchi and Aoki [15] and Phillips and Rosenfeld's [17] algorithms which all perform poorly, showing strong fluctuations in the number of lines. Wu and Levine's [23] algorithm performs with only mediocre

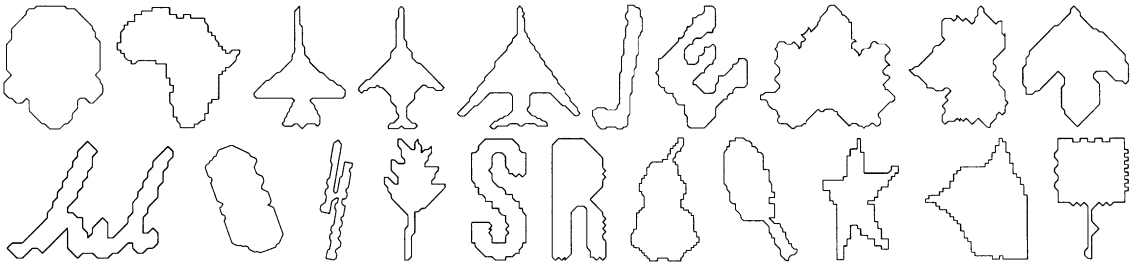

Fig. 12. Test curves.

Table 5
Assessment of various algorithms applied to 21 test curves

Method	M (Line)		M (ISE)		Consistency		Merit		Stability	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Optimal E_1	100	0.00	97	2.8	0.0	0.00	88	9.0	2.14	0.57
Optimal E_2	100	0.0	100	0.0	0.0	0.00	100	0.0	2.02	0.59
Optimal E_∞	100	0.0	95	3.1	0.0	0.00	77	10.4	2.75	1.72
Optimal E_L	100	0.0	80	14.2	0.0	0.00	63	13.6	1.67	0.68
Optimal E_θ	100	0.0	77	16.4	0.0	0.00	40	18.9	1.94	0.77
Cheng and Hsu [14]	72	13.2	28	15.5	6.1	2.64	16	11.3	4.86	1.87
Deguchi & Aoki [15]	17	10.8	18	23.0	6.5	5.37	32	17.15	5.63	2.19
Deveau [16]	100	0.0	89	13.4	2.3	0.62	41	24.2	1.87	0.48
Douglas & Peucker [18]	99	2.8	98	5.6	0.1	0.27	64	12.4	1.65	0.51
Fu et al. [21]	8	19.5	1	19.7	2.6	0.80	51	17.3	5.08	3.42
Hu & Yan [24]	86	13.9	89	11.9	0.0	0.00	56	13.4	2.37	0.52
Inesta et al. [22]	4	53.3	−1	48.8	1.6	0.60	37	13.3	3.48	1.24
Ji & Haralick [7]	81	25.1	68	18.0	1.7	1.32	22	13.9	—	—
Melen & Ozanian [26]	83	13.7	61	18.4	1.9	1.27	58	20.0	2.88	1.12
Pei & Horng [27]	73	15.7	74	12.9	2.1	1.43	40	15.2	2.14	0.77
Phillips & Rosenfeld [20]	29	27.1	26	17.8	5.5	1.81	30	12.6	2.03	0.90
Ramer [19]	100	0.0	99	1.1	0.0	0.00	74	19.7	2.08	1.56
Regular	100	0.0	66	8.8	0.0	0.00	27	9.0	0.00	0.00
Rosenfeld & Johnston [20]	85	13.4	71	9.1	2.6	1.27	52	25.0	2.40	0.79
Wu & Levine [23]	59	31.0	52	19.6	2.1	0.95	32	15.6	1.93	1.08
Zhang et al. [8]	96	5.6	60	13.9	2.3	1.07	36	14.2	2.15	0.30

monotonicity, while the remainder are all adequate. We note that although Ji and Haralick's [8] algorithm produces an overall high monotonicity score the high standard deviation indicates that it is variable over the test set.

3.2. ISE monotonicity

ISE monotonicity checks the expectation that decreasing the scale parameter should not increase the approximation error. 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, the plots of many algorithms appear nearly monotonic and achieve correspondingly high monotonicity values in Table 2 (i.e. [16,18,19,24]). Plots showing extreme fluctuations (i.e. [21,22]) receive very low scores. In particular, Inesta et al.'s [22] algorithm produced very variable results, leading to the high standard deviation in monotonicity. Cheng and Hsu [14], Deguchi and Aoki [15], and Phillips and Rosenfeld's [17] algorithms also performed poorly.

3.3. Consistency

The consistency measure checks that algorithms do not produce alternative segmentations with the same number of lines but different approximation errors. The optimal algo-

rithms are of course perfectly consistent, as are the Hu and Yan [24], Ramer [19], and regular algorithms. The Douglas and Peucker [18] algorithm is also close to perfectly consistent. The worst offenders are the Cheng and Hsu [14], Deguchi and Aoki, [15] and Phillips and Rosenfeld's [17] algorithms which also showed poor ISE monotonicity.

3.4. Merit

The figure of merit was developed to measure the performance of polygonalisation algorithms with respect to an optimal algorithm [11]. It combines two factors, fidelity (the accuracy of the approximation), and efficiency (the number of lines required to achieve a given accuracy). We have used the optimal E_2 algorithm (i.e. minimum ISE) as the baseline for measuring merit. Thus, relative to ISE, even the other optimal algorithms which minimise alternative criteria do not necessarily perform well, although, as expected the most similar criteria do best (i.e. E_1 and E_∞). From among the non-optimal algorithms Ramer's does by far the best, the next best being Douglas and Peucker's algorithm. Surprisingly, both Cheng and Hsu [14] and Ji and Haralick's [7] algorithm's received very poor scores, falling below the simple regular curve subdivision method.

3.5. Stability

Stability tests the sensitivity of the segmentation algorithm to perturbations in the data, and measures the degree the perturbations cause the breakpoints to shift. For the experiment testing stability under data perturbation, the algorithm's scale parameters were selected so that they produced as close to eleven breakpoints on average as possible. As Table 5 shows, the simple regular subdivision algorithm is naturally perfect. Of more interest we can see that the optimal algorithms are not necessarily best according to stability. As we might expect, the maximum error (E_∞) is particularly sensitive to perturbations. In fact, none of the algorithms achieve good scores, we note that the best is the Douglas and Peucker algorithm. We can however see that the following algorithms are particularly sensitive to perturbations in the data: Cheng and Hsu, Deguchi and Aoki,[4] Fu et al., and Inesta et al.

As with breakpoint stability under curve deletion the measure was validated by running the pose estimation task for each algorithm over all the noisy test curves. In Fig. 13a the graph of pose errors against stability displays reasonable correlation (Kendal's $\tau = 0.607$) although the error ellipses show that there is considerable variation in performance over the data set. We also see in Fig. 13b that the magnitude of variations in the pose and stability measures also correlate well (Kendal's $\tau = 0.681$), confirming the effectiveness of the stability measure.

[4] Of course, the random optimisation we used to achieve acceptable speed will have had a significant effect on the stability of the algorithm.

3.6. Overall assessment

It should be noted that most of the algorithms exhibit large variations in the measures over varying parameter values and data sets. To aid overall assessment of the algorithms the contents of Table 5 are presented again in Table 6. The entries now correspond to classifications into the upper ($\surd$), median ($=$), and lower ($\times$) quartiles. It is easy to see that:

- The optimal algorithms generally do well as would be expected, although the E_1 criterion is a little more sensitive than the others to data perturbations and the E_∞ criterion rather more so. Also the E_θ criterion is generally good but does not provide accurate approximations according to the E_2 criterion.
- Surprisingly, of the non-optimal algorithms from the literature, the two best are both simple techniques almost thirty years old, namely those by Douglas and Peucker and Ramer. Douglas and Peucker fares better on stability while Ramer rates higher on merit (i.e. providing more accurate and concise representations). Also of note are Deveau and Hu and Yan's algorithms which do quite well except for weaknesses in consistency and merit (Deveau) and line monotonicity and stability (Hu & Yan).

Thus if precise high quality approximations are required then the optimal algorithms with E_2 or E_L criteria are the natural choice. Otherwise Douglas and Peucker or Ramer provide more efficient algorithms with a moderate drop in performance.

4. Discussion

This paper has 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 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 40,000 program runs were necessary) it is still incomplete on several counts. First, only 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

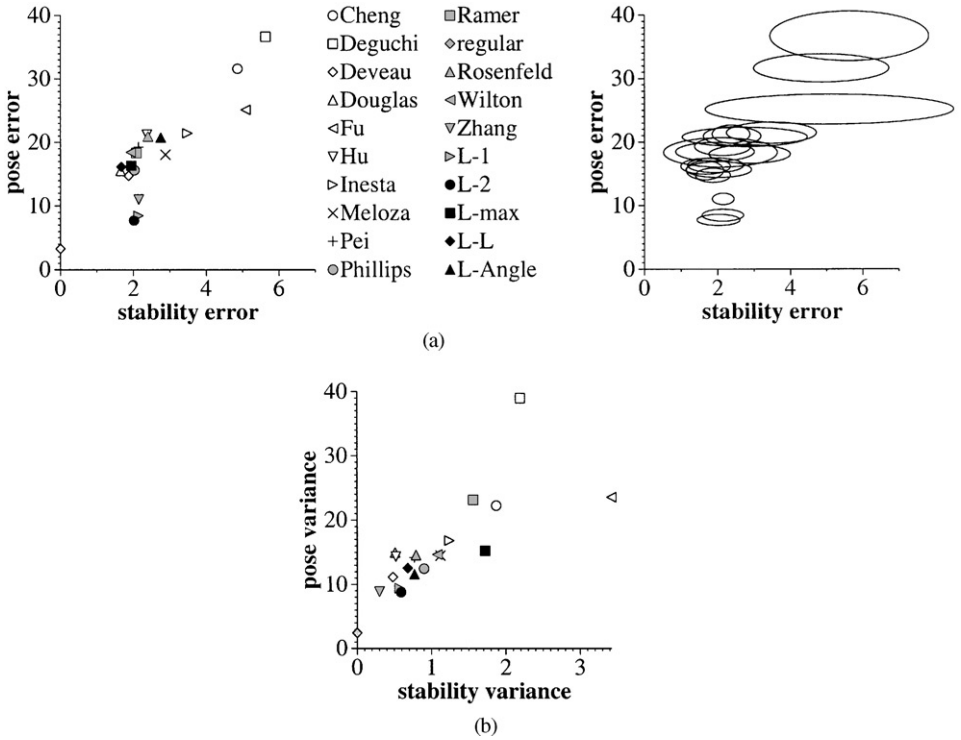

Fig. 13. Perturbation stability plotted against pose estimation error; error ellipses are shown 1σ on X-axis and 0.1σ on Y-axis.

Table 6
Overview of assessment of various algorithms classified into the upper ($\surd$), median($=$), and lower ($\times$) quartiles

Method	M (Line)	M (ISE)	Consistency	Merit	Stability
Optimal E_1	$\surd$	$\surd$	$\surd$	$\surd$	$=$
Optimal E_2	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Optimal E_∞	$\surd$	$\surd$	$\surd$	$\surd$	$=$
Optimal E_L	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Optimal E_θ	$\surd$	$\surd$	$\surd$	$=$	$\surd$
Cheng and Hsu [14]	$=$	$\times$	$\times$	$\times$	$\times$
Deguchi & Aoki [15]	$\times$	$\times$	$\times$	$\times$	$\times$
Deveau [16]	$\surd$	$\surd$	$=$	$=$	$\surd$
Douglas & Peucker [18]	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Fu et al. [21]	$\times$	$\times$	$\times$	$=$	$\times$
Hu & Yan [24]	$=$	$\surd$	$\surd$	$\surd$	$=$
Inesta et al. [7]	$\times$	$\times$	$=$	$=$	$\times$
Ji & Haralick [26]	$=$	$=$	$=$	$\times$	
Melen & Ozanian [27]	$=$	$=$	$=$	$\surd$	$\times$
Pei & Horng [17]	$=$	$=$	$=$	$=$	$=$
Phillips & Rosenfeld [17]	$\times$	$\times$	$\times$	$\times$	$\surd$
Ramer [19]	$\surd$	$\surd$	$\surd$	$\surd$	$=$
Regular	$\surd$	$=$	$\surd$	$\times$	$\surd$
Rosenfeld & Johnston [20]	$=$	$=$	$\times$	$=$	$=$
Wu & Levine [23]	$\times$	$\times$	$=$	$\times$	$\surd$
Zhang et al. [8]	$=$	$=$	$=$	$\times$	$=$

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. And fourth, there are many other algorithms which it would be nice to test.

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 [25] while specific tasks can be shown to favour particular choices of error [11].

5. Summary

Recently, there has been an increased interest in the automatic and quantitative assessment of computer vision algorithms. This paper contributes by developing several measures to assess the stability of polygonal approximation algorithms under two classes of variations, namely perturbations of the data and changes in the algorithms' scale parameters. In order to quantify the former, a breakpoint stability index is defined and tested. For the latter, two indices are introduced; (1) a monotonicity index is applied to analyse the change in the approximation error or the number of line segments against increasing scale, and (2) a consistency index quantifies the variation in results produced at the same scale by an algorithm (but with different input parameter values).

Tests were applied to 16 algorithms drawn from the literature, ranging over 25 years and based on a diversity of methods. These showed that most of the algorithms exhibit large variations in the measures over varying parameter values and data sets. The optimal algorithms generally fared well as would be expected. More surprisingly, of the non-optimal algorithms, the two best are both simple techniques almost 30 years old, namely those by Douglas and Peucker and Ramer. Also of note are Deveau and Hu and Yan's algorithms which do quite well except for weaknesses in consistency and merit (Deveau) and line monotonicity and stability (Hu & Yan).

Acknowledgements

I would like to thank the following for providing software, results, and advice: Philippe Cornic, Terry Deveau, Alan Fu, Bob Haralick, Jianming Hu, Jose Inesta, Qiang Ji, Martin Levine, Dale Lutz, Trond Melen, Takouhi Ozanian, and Kenong Wu.

References

[1] R. Haralick, R. Klette, S. Stiehl, M. Viergever (Eds.), Evaluation and validation of computer vision algorithms, Dagstuhl-Seminar-Report 205, 1998.
[2] G. Schuster, A. Katsaggelos, An optimal polygonal boundary encoding scheme in the rate-distortion sense, IP 7 (1) (1998) 13–26.
[3] P. Yin, A new method for polygonal approximation using genetic algorithms, Pattern Recognition Lett. 19 (11) (1998) 1017–1026.
[4] T. Pham, H. Yan, An effective algorithm for the segmentation of digital plane curves: the isoparametric formulation, Pattern Recognition Lett. 19 (2) (1998) 171–176.
[5] A. Garrido, N. Perez De la Blanca, M. Garcia-Silvente, Boundary simplification using a multiscale dominant point detection algorithm, Pattern Recognition 31 (6) (1998) 791–804.
[6] R. Lin, C. Chu, Y. Hsueh, A modified morphological corner detector, Pattern Recognition Lett. 19 (3–4) (1998) 279–286.
[7] Q. Ji, R. Haralick, Breakpoint detection using covariance propagation, IEEE Trans. PAMI 20 (8) (1998) 845–851.
[8] X. Zhang, R. Haralick, V. Ramesh, Corner detection using the MAP technique, in: Proceedings of the 12th International Conference on Pattern Recognition, 1994, pp. 549–551.
[9] T. Kadonaga, K. Abe, Comparison of methods for detecting corner points from digital curves, in: International Workshop on Graphics Recognition, 1995, pp. 3–12.
[10] P. Cornic, Another look at the dominant point detection of digital curves, Pattern Recognition Lett. 18 (1) (1997) 13–25.
[11] P.L. Rosin, Techniques for assessing polygonal approximations of curves, IEEE Trans. PAMI 19 (6) (1997) 659–666.
[12] P.L. Rosin, Assessing the behaviour of polygonal approximation algorithms, Technical Report, Cardiff University, 2000.
[13] C. Teh, R. Chin, On the detection of dominant points in digital curves, IEEE Trans. PAMI 11 (1989) 859–872.
[14] F.-H. Cheng, W.-H. Hsu, Parallel algorithm for corner finding on digital curves, Pattern Recognition Lett. 8 (1988) 47–54.
[15] A. Deguchi, S. Aoki, Regularized polygonal approximation for analysis and interpretation of planar contour figures, in: International Conference on Pattern Recognition, Vol. 1, 1990, pp. 865–869.
[16] T. Deveau, Reducing the number of points in a plane curve representation, in: Proceedings on Auto-Carto, Vol. VII, 1985, pp. 152–160.
[17] T. Phillips, A. Rosenfeld, A method of curve partitioning using arc-chord distance, Pattern Recognition Lett. 5 (1987) 285–288.
[18] D. Douglas, T. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Canad. Cartographer 10 (1973) 111–122.
[19] U. Ramer, An iterative procedure for the polygonal approximation of plane curves, Comput. Graphics Image Process. 1 (1972) 244–256.
[20] A. Rosenfeld, E. Johnston, Angle detection on digital curves, IEEE Trans. Comp. 22 (1973) 875–878.
[21] A. Fu, H. Yan, K. Huang, A curve bend function method to characterize contour shapes, Pattern Recognition 30 (10) (1997) 1661–1671.
[22] J. Iñesta, M. Buendía, M. Sarti, Local symmetries of digital contours from their chain codes, Pattern Recognition 29 (10) (1996) 1737–1749.

[23] K. Wu, M. Levine, 2d shape segmentation: a new approach, Pattern Recognition Lett. 17 (2) (1996) 133–140.
[24] J. Hu, H. Yan, Polygonal-approximation of digital curves based on the principles of perceptual organization, Pattern Recognition 30 (5) (1997) 701–718.
[25] D. Foster, D. Simmons, M. Cook, The cue for contour-curvature discrimination, Vision Res. 33 (1993) 329–341.
[26] T. Melen, T. Ozanian, A fast algorithm for dominant point detection on chain-coded contours, in: Proceedings of the Fifth International Conference on Computer Analysis of Images and Patterns, 1993, pp. 245–253.
[27] S. Pei, J. Horng, Corner point detection using nest moving average, Pattern Recognition 27 (1994) 1533–1537.

About the Author—PAUL ROSIN received the B.Sc. degree in Computer Science and Microprocessor Systems in 1984 from Strathclyde University, Glasgow, and the Ph.D. degree in Information Engineering from City University, London in 1988. He was a research fellow at City University, developing a prototype system for the Home Office to detect and classify intruders in image sequences. He worked on the Alvey project "Model-Based Interpretation of Radiological Images" at Guy's Hospital, London before becoming a lecturer at Curtin University of Technology, Perth, Australia, and later a research scientist at the Institute for Remote Sensing Applications, Joint Research Centre, Ispra, Italy, followed by a return to the UK, becoming lecturer at the Department of Information Systems and Computing, Brunel University London, UK. Currently he is senior lecturer at the Department of Computer Science, Cardiff University.

His research interests include the representation, segmentation, and grouping of curves, knowledge-based vision systems, early image representations, machine vision approaches to remote sensing, and the analysis of shape in art and architecture. He is the secretary of the British Machine Vision Association.