Proceedings of the SME Applied Machine Vision ‘96 Conference, Cincinnati Ohio, June 3-6, 1996

Efficient piecewise linear approximation of space curves using chord and arc length

John Albert Horst
Intelligent Systems Division
The National Institute of Standards and Technology
U.S. Department of Commerce
Bldg. 220 Rm. B-124
Gaithersburg, MD 20899
internet: horst@cme.nist.gov
ftp: isdftp.cme.nist.gov/pub/horst
voice: (301)975-3430

and

Isabel Beichl
Applied and Computational Mathematics Division
The National Institute of Standards and Technology
U.S. Department of Commerce
Bldg. 820 Rm. 365
Gaithersburg, MD 20899
internet: isabel@cam.nist.gov
voice: (301)975-3821

abstract

An efficient, accurate, and simple method for doing suboptimal piecewise linear approximation of open or closed space curves is described. The algorithm guarantees approximation within a deviation threshold and is offered as an efficient alternative to the popular, but inefficient, split and merge approach. The several efficient alternatives this author encountered are defined only for planar curves. The approach we offer is also appropriate for space curves. In our approach, a monotonically increasing function of chord and arc length is used to form the initial set of approximating points followed by a merging of those points. A posterior least squares fitting technique is presented which further improves the approximation. Preliminary Gaussian smoothing can be done depending on the application. Analysis of this new approach on a variety of planar curves is presented and comparisons are made with other piecewise linear curve approximation algorithms.

1 introduction

Planar curve approximation methods have received much attention in the computer vision literature for a long period of time. Such approximations are useful for a variety of reasons:

- shape analysis algorithms, e.g., 2D template matching [Jain 96], rarely require a complete set of data [Sato 93]
- significant data reduction can be achieved, particularly for large input curves, depending on the accuracy of the approximation
- many real-time applications, e.g., display graphics, can realize significant speedups through curve data reduction via curve approximation

Piecewise linear planar curve approximation has been the focus of particular attention and is attractive largely because of the inherent simplicity of an iconic representation. For example, there is an ease and simplicity when doing correlations with model templates. More abstract curve representations (such as B-splines) may be harder to compare with models. The piecewise linear algorithms can be classified into the following types:

- optimal and suboptimal
- global and local (as to how the algorithm processes the data)
- efficient, i.e., $O(n)$, and inefficient, i.e., $O(n^2)$, for n sampled data points in the input curve
- can handle non-integer valued curves or just integer valued curves as input
- can handle open as well as closed curves
- can handle space curves or just planar curves as input

To some extent the diversity of methods arises out of a multiplicity of needs of those seeking to design and employ such algorithms:

- speed and determinism for use in real-time pattern recognition systems
- optimality in error performance where there is no need for on-line, real-time performance
- data reduction on integer valued planar curves, often defined as chain codes, gotten from thinned edge images which were, in turn, gotten from intensity images
- simplicity for ease of programming and debugging
- human-like, symmetry preserving results
- manufacturing applications requiring curve approximation for machine tool or measurement probe tip paths in 3-dimensional space
- combinations of the above

The method described in this paper is defined for curves in space ($\mathbf{R}^3$) and it is simple, fast, deterministic, and guarantees accuracy. We have in mind its use particularly for manufacturing applications such as those found in defining paths in $\mathbf{R}^3$ of machine tool cutting bits and coordinate measuring machine probe tips.

2 related research

Pavlidis [Pavlidis 77] and Dunham [Dunham 89] offer optimal approaches. An optimal piecewise linear curve approximation typically means that, if we desire to approximate with m points a curve which has n Š m points, the particular m points selected will give the approximation which minimizes the maximum deviation of the curve points from the approximating line segments. As one would expect, the computational cost of the optimal methods rises sharply with the number of input points [Dunham 89].

Human-like, symmetry-preserving approaches are found in several papers. Fischler and Bolles [Fischler 86] have developed a suboptimal approach that succeeds in copying human perceptual partitioning of planar curves: high level perception activity is accomplished, e.g., noisy segments are distinguished from non-noisy segments. However, the algorithm is inefficient and the input parameters are difficult to tune. Aoyama and Kawagoe [Aoyama 89] also focus their efforts on avoiding distortion and providing human-like performance more than assuring simplicity and efficiency, since the complexity of their algorithm is $O(n^2)$. They also limit themselves to digital planar curves. They classify linear approximation algorithms based on whether the data is accessed serially or globally. However, for many computing tasks, simplicity and efficiency are of the highest priority. Real-time object recognition systems for manufacturing are less concerned with human-like approximations and more with simplicity and efficiency. The algorithm we have developed is clearly biased towards these types of concerns, however, our approach creates a remarkably small amount of global distortion as will be shown in the figures.

Sklansky's planar curve approximation method [Sklansky 80] is shown by Dunham [Dunham 89] to be efficient and to find curve approximating points that are very close in number to those found by optimal algorithms. However, Sklansky's approach is defined for digitized, planar curves only.

Teh and Chin [Teh 89] have also defined an algorithm for finding dominant points on digital curves that has a reasonably good error performance. However, their method suffers from the following weaknesses:

- It is complex to describe and the concept of 'region of support' seems non-intuitive
- No parameters are supplied to control tightness of fit
- The algorithm can be performed in parallel, but is relatively slow if done serially
- It assumes closed, digital planar curves only.

Robergé [Robergé 85] defined an efficient algorithm for planar curves. This approach has a consistently shorter execution time compared to several other efficient algorithms [Dunham 86]. It is also not sensitive to quantization error. However, we noticed the following weaknesses:

- The true upper bound guaranteed by the algorithm, $\sqrt{5}d$, is noticably looser than our approach for the sample curves we investigated as shown in Figure 6
- For some curves (see Figure 6) Robergé's algorithm distorts the graphic significantly
- An additional parameter is required for input other than deviation

William's method [Williams 77] is not reliable since it cannot guarantee an upper bound on the approximation error (see Figure 6). This fact is enough to disqualify this approach for many manufacturing applications, in which deviation in approximation must be tightly controlled. However, it is conceivable that a minor modification to this approach would correct the problem.

Of approaches that are suboptimal, the split and merge method has arisen as perhaps the most popular [Chen 79, Duda 73, Jain 89, Grimson 90]. Its popularity is due to its

- simplicity, since it is controlled by a single, physically meaningful parameter, and it is easy to state and code
- guaranteed error performance, since it directly measures error
- ability to preserve visual features
- successful use as a preliminary step in optimal approaches [Pavlidis 77]
- ability to handle closed as well as open curves
- ability to handle analog space curves as well as digital planar curves

However, there are several known weaknesses:

- it is suboptimal
- it is inefficient, i.e., $O(n^2)$, for n sampled data points in the input curve, because it analyzes the data recursively
- there is also a problem with its sensitivity to the choice of initial breakpoint [So 93]
- the compute time varies depending on the degree of approximation required (i.e., 'tightness' of fit) which could be a problem for some real-time applications (see Figure 5)
- several attempts to improve the efficiency of the split and merge approach come with an increase in complexity [Nevatia 80, So 93]

The split and merge is also an off-line algorithm (non-real-time) in the sense that it requires that all the data be collected before processing. This is not desirable for many real-time applications. For example, in the collection of position data from a machine tool, one would like to eliminate redundant data as it is being collected. However, one algorithm often used that processes the data serially is the cone intersection algorithm [Williams 77]. However, the amount of global distortion in the Williams method may be unacceptable [Aoyama 89].

Many algorithms assume digital curves (often defined by Freeman chain codes [Teh 89]), implicitly arguing that it is so common in image processing that allowing for the more general case is unnecessary. Algorithms that only deal with digital curves, e.g., [Aoyama 91] and [Teh 89], are limiting because one may want to perform Gaussian smoothing first. Errors in a digital curve such as quantization error, missed points, or redundant points argue for a method that can allow curves with such anomalies.

One might ask why not just check the error from the chord lines directly and serially until that error is exceeded? This is a non-recursive serial algorithm and, therefore, the compute cost is proportional to the square of the number of data points. This is because errors must be formed again and again for the same points for each approximating line segment. Also, because it processes the data locally, it hasn't the global feature preserving qualities of the split and merge algorithm.

3 the chord and arc length algorithm

We now describe the chord and arc length (CAL) method for piecewise linear space curve approximation below. Steps one and three describe CAL. Step two is an optional preliminary Gaussian smoothing step; step four is an optional posterior merging step; steps five through seven describe an optional posterior least squares fitting step.

1. determine whether the curve is open or closed
2. do local Gaussian smoothing on the raw input curve, as needed, to reduce local error (e.g., quantization error)
3. starting anywhere on the closed curve (at the first point on the open curve), compute chord length, C, and arc length, S, for each successive point and when $(1/2)\sqrt{S^2 - C^2}$ is greater than the maximum deviation parameter, declare the previous point to be a dominant point
4. merge dominant points by testing if each dominant point can be eliminated without exceeding the threshold on deviation
5. compute a (parameterized) least squares line to the points on the curve between and including the most recent two dominant points computed
6. find the point on the previous and current least squares fit lines that are closest to the previous dominant point
7. choose the midpoint between these two closest points as the latest approximating point

For many applications, steps one, three, and four will be all that is needed. Both Gaussian smoothing and least squares fitting require significant additional computation, but still do not lead to inefficient computational costs.

Gaussian smoothing is sometimes useful because, for example, in image processing, an edge thinning algorithm may produce thinned edge pixels having redundant neighbors[1] for example, one four-neighbor and one eight-neighbor . Local smoothing can ameliorate local noise such as quantization noise. Gaussian windows of size five were used in our simulations.

Posterior least squares fitting just provides a tighter approximation, however, digital points would then be replaced by real valued points. It also ameliorates the inscribing problem found in the split and merge method (inscribing is also true with CAL less the posterior least squares method). It provides a better fit to smoothly curving data (e.g., circles) because it computes least squares lines and uses them to compute the approximating points.

The only additional complexities over the split and merge method are the addition of the least squares line computation and Gaussian smoothing, both of which are optional.

3.1 advantages and disadvantages of curve approximation using chord and arc length

This algorithm we've described has the following advantages:

- simplicity, since it is easy to state and code, and it is controlled by a single physically meaningful parameter, the maximum deviation parameter.
- efficiency, i.e., $O(n)$, for n sampled data points in the input curve.
- guaranteed error performance, proven in the appendix
- error performance is good compared to the popular but inefficient split and merge method
- excellent ability to preserve visual features when compared to some other efficient algorithms
- it is relatively insensitive to the initial choice of starting points
- performance times are constant with respect to the deviation threshold
- ability to handle closed as well as open curves
- ability to handle non-integer valued points in space ($\mathbf{R}^3$) as well as integer valued planar curves

1. such a set of neighbors is not necessarily redundant (e.g., at a corner).

None of the efficient methods we encountered are defined to handle space curves. The popular split and merge method can handle space curves but it is inefficient, which means that, for very large input curves, compute time can be forbidding. The simple statement of this new method requires only one serial pass through the data and the compute time is only linearly dependent on the number of data points. This fact is of interest to certain real-time applications in which the dominant points need to be selected on-line, i.e., before all the data has been collected. In addition, the compute time is essentially constant with respect to the required degree of approximation. The latter advantage is again important to certain real-time systems. Deterministic performance is accomplished, since execution time is nearly constant with respect to the tightness of fit. Efficiency is gained by *indirectly* measuring approximation error through the monotonic function of chord and arc length, $(1/2)\sqrt{S^2 - C^2}$.

The disadvantages of CAL are:

- it is sensitive to quantization error, however, if such error is present in a curve, preliminary Gaussian smoothing and/or posterior merging will ameliorate this problem
- Adding the posterior merging step to ameliorate quantization error adds significantly to the total execution time of the algorithm and in some pathological cases would cause the algorithm to be inefficient.
- the split and merge method produces less perceivable deformity of the original curve than CAL (however, CAL produces less deformity than Williams (see Figure 6))
- it requires a costly square root calculation at each step (this could be approximated)
- none of the curve approximation methods studied (including CAL) minimize ***total*** deviation error, but only minimize ***maximum*** deviation error. Sometimes the maximum error can be small but the total error large, which can produce global distortion

CAL establishes an indirect measure of error by creating a natural upper bound on the deviation error of the approximation. This simple step brings efficiency without significant loss in error performance. Using language in [Aoyama 91], CAL accesses the data sequentially, the split and merge algorithm accesses the data globally. Global access (for the split and merge method) means that all the data in the curve is examined by the algorithm during each pass. However, while other algorithms seem to gain from global data access (e.g., [Aoyama 91]), the split and merge algorithm suffers from a significant loss of efficiency without significant appreciable gain. CAL (particularly with open curves but also with closed curves) almost completely avoids the problem on the choice of initial breakpoint from which the split and merge suffers [So 93]. However, because CAL measures error indirectly, the maximum error will deviate from the threshold value more in CAL than in split and merge depending on the particular shape of the curve. Moreover, the amount of this deviation in CAL is based on the ‘sharp corners’ in the curve, i.e., a smoothly varying curve will have greater deviation. CAL more naturally accommodates closed and open curves whereas many approaches seem more naturally suited to one or the other (e.g., split and merge is more suited to open and Teh-Chin to closed).

In the appendix we prove the claim that the CAL algorithm guarantees that the original curve points will never be farther than a user defined threshold away from the line segment used to approximate the curve segment containing that original point. The CAL algorithm can be considered an efficient, albeit indirect, form of the split part of the split and merge approach.

3.2 analysis

We now describe the chord and arc length algorithm more precisely. Let $\alpha(i)$, $i = 1, 2, \ldots, n$, be a sequence of points defining a curve in $\mathbf{R}^3$. An example of such a curve is illustrated in Figure 1. If $\alpha(1) \approx \alpha(n)$, the curve is closed. We seek a sequence of dominant points, $\alpha(i_j)$, $j = 1, 2, \ldots, m \le n$, such that, for $i_j < i < i_{j+1}$

$$\mathtt{max}\{\mathtt{dist}(\alpha(i), \mathtt{lineseg}(\alpha(i_j), \alpha(i_{i+1})))\} \le d \quad \text{(EQ 1)}$$

for a given maximum deviation value, d. $\mathtt{lineseg}(\alpha(i_j), \alpha(i_{i+1}))$ is the set of points along the chord from $\alpha(i_j)$ to $\alpha(i_{j+1})$. The distance function, `dist`, computes the shortest distance from a point on the curve in $\mathbf{R}^3$ to points on the chord. These definitions are illustrated in Figure 1.

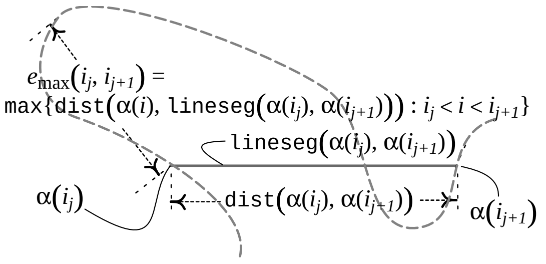
$e_{\max}(i_j, i_{j+1}) =$
$\mathtt{max}\{\mathtt{dist}(\alpha(i), \mathtt{lineseg}(\alpha(i_j), \alpha(i_{j+1}))) : i_j < i < i_{j+1}\}$
$\mathtt{lineseg}(\alpha(i_j), \alpha(i_{j+1}))$
$\alpha(i_j)$
$\mathtt{dist}(\alpha(i_j), \alpha(i_{j+1}))$
$\alpha(i_{j+1})$

Figure 1: The max distance from points on an arc in $\mathbf{R}^3$ to the line segment joining the endpoints of the arc.

With $\alpha(i_j)$ as the most recently determined dominant point, to find $\alpha(i_{j+1})$, the next dominant point, we form expressions for chord length, $c(i_j, i)$, and arc length, $s(i_j, i)$, between the points, $\alpha(i_j)$ and $\alpha(i)$, for $i_j < i$,

$$c(i_j, i) = \|\alpha(i) - \alpha(i_j)\| \qquad \text{(EQ 2)}$$

and

$$s(i_j, i) = \sum_{k = i_i}^{-1} \|\alpha(k+1) - \alpha(k)\| \qquad \text{(EQ 3)}$$

respectively, until, for some $i > i_j$

$$d(i_j, i) = \frac{1}{2}\sqrt{s^2(i_j, i) - c^2(i_j, i)} > d \qquad \text{(EQ 4)}$$

Then we choose $\alpha(i-1)$ (the previous point) as the new dominant point and we assign the indice, $i_{j+1} = i - 1$. Therefore, since (EQ 4) is monotonically increasing,

$$d(i_j, i_{j+1}) \le d \qquad \text{(EQ 5)}$$

If the curve is closed, all operations are mod *n*; if open, choose $\alpha(1)$ and $\alpha(n)$ as the first and last dominant points.

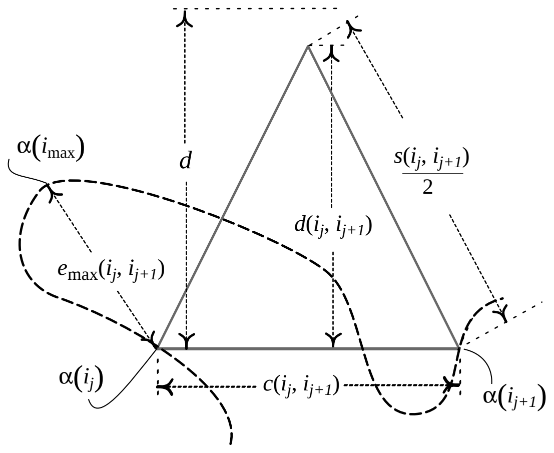

Figure 2: The isosceles triangle formed from the length of the arc in Figure 1.

We must show that (EQ 1) is true if (EQ 4) is met throughout the sequence of points $\alpha(i)$. This can most easily be demonstrated by examination of Figure 2. Intuitively we can think of the function $d(i_j, i_{j+1})$ by imagining that, if the curve were a flexible, but not stretchable, string attached at the dominant points, $\alpha(i_j)$ and $\alpha(i_{j+1})$ and we pushed the string upwards with a stick, we can form an isosceles triangle as in Figure 2. We can prove that, for any curve in $\mathbf{R}^3$,

$$e_{\max}(i_j, i_{j+1}) \le d(i_j, i_{j+1}) \le d \qquad \text{(EQ 6)}$$

where $e_{\max}(i_j, i_{j+1})$ is the largest deviation from points on the curve from $\alpha(i_j)$ to $\alpha(i_{j+1})$ to points on the chord from $\alpha(i_j)$ to $\alpha(i_{j+1})$ as illustrated in Figure 1 and Figure 2. We prove (EQ 6) in the appendix.

A lemma and a theorem (stated and proved in the appendix) reveal that if we are given a threshold value, *d*, the basic chord and arc length algorithm we have described will guarantee that the deviation of each point on the curve from its appropriate approximating line segment (i.e., the chord line segment) will be less than or equal to *d*. This provides a physically meaningful upper bound on the approximation while maintaining efficiency.

The merge process operates as in the split and merge method, namely, if

$$\mathtt{max}\{\mathtt{dist}(\alpha(i), \mathtt{lineseg}(\alpha(i_{j-1}), \alpha(i_{i+1})))\} \le d \qquad \text{(EQ 7)}$$

for all *i* such that $i_{j-1} < i < i_{j+1}$, $\alpha(i_j)$ is eliminated as a dominant point.

Gaussian smoothing is not a necessary part of this approach, but can be useful with digital curves when we wish to eliminate quantization noise. Let, $\alpha(i)$, be the unsmoothed sequence of points in $\mathbf{R}^3$. For *w* odd, the Gaussian smoothing vector of length *w* is

$$\left(g_{-\frac{w-1}{2}}, \ldots, g_{-1}, g_0, g_1, \ldots, g_{\frac{w-1}{2}}\right), \qquad \text{(EQ 8)}$$

be integrations of appropriately chosen, normalized one-dimensional Gaussian functions[1]. Then the smoothed sequence of curve points in $\mathbf{R}^3$ is

$$\beta(i) = \sum_{j = i - \frac{w-1}{2}}^{i + \frac{w-1}{2}} \alpha(i) g(j - i) . \qquad \text{(EQ 9)}$$

We now describe a least squares fitting technique to further reduce approximation error and to help ameliorate the inscribing error problem found in all referenced approaches. Like Gaussian smoothing, least squares fitting is not a necessary part of this approach, but may be useful if computing time is available and non-integer valued points are acceptable. Each subset of points,

$$(\alpha(i_j), \alpha(i_j + 1), \ldots, \alpha(i_{j+1} - 1), \alpha(i_{j+1})) ,$$

on the original raw input curve between dominant points is used to fit a parameterized line , $l_2(t)$. We find

1. We used Gaussian windows of size three, five, and seven: (0.1586, 0.6827, 0.1586), (0.0228, 0.22978, 0.4950, 0.2297, 0.0228), (0.0062, 0.0606, 0.2417, 0.3829, 0.2417, 0.0606, 0.0062)

points, p_1 and p_2, which are the points on the previous least squares line, $l_1(t)$, and on the current least squares line, $l_2(t)$, that are the shortest distance from $\alpha(i_j)$ to each of the lines. The latest approximating point is then[1] chosen as the midpoint between p_1 and p_2.

3.3 algorithm performance

In this section we present some of the results of testing the CAL approach to curve approximation on a variety of types and sizes of curves and also testing its performance against some of the other curve approximation methods discussed in this paper.

3.3.1 Curve approximation algorithm performance metrics

There are several metrics that we have established for measuring performance:

1. How does the actual maximum deviation (of the original curve points from the approximating line segments) compare to the input deviation threshold? The closer the former is to the latter, the better the algorithm. It is generally not good if the former can exceed the latter.
2. For a given actual maximum deviation, how many approximating points where required? Fewer approximating points for a given deviation is better.
3. With all else being equal, what is the speed of execution?
4. What is the variability in the speed of execution with respect to the number of approximating points (controlled by the input deviation parameter)?
5. What is the variability in the speed of execution with respect to the size of the input curve?

We will now analyze the performance of the chord and arc length method (CAL) under these various metrics.

3.3.2 Performance of CAL under metric #1: actual deviation vs. input deviation

It is essential to examine the error between the approximating line segments and the smoothed curve points as a function of the number of approximating points for a variety of types of curves. We find that the chord and arc length algorithm performs better than both the split and merge and the Teh-Chin algorithm in the two example curves we used. We expect that the split and merge method would do slightly better when the starting point of a closed curve is chosen carefully [So 93]. However, this comes at a significant cost in additional complexity. We demonstrate the errors of several algorithms in Figure 4 for the digital closed curve of a chromosome in Figure 3. Particularly surprising is the performance of CAL with respect to the split and merge method as shown in Figure 8 for the digital closed curve of a leaf in Figure 6. Figure 6 shows the excellent performance of CAL by this metric against two other efficient algorithms, namely, Williams [Williams 77] and Robergé [Robergé 85]. Williams performance is particularly poor on this curve because it reveals that errors of approximation greater than threshold can occur. The performance of the Robergé algorithm shows that the actual maximum deviation (of the original curve points from the approximating line segments) is too much less than the guaranteed threshold value. We have also analyzed standard deviation errors and total deviation errors and have obtained results similar to that reported on maximum error in Figure 4 and Figure 8.

3.3.3 Performance of CAL under metric #2: number of approximating points for a given input deviation

In Figure 6 we see that for a given input deviation value, the number of approximating points varies rather widely over the three candidate algorithms. The Robergé algorithm gives a poor performance. Williams perform very well against CAL, mostly because of CAL's sensitivity to quantization error. CAL does better by this metric when posterior merging is performed as can be seen in Figure 6.

3.3.4 Performance of CAL under metric #3: execution speed

Robergé is fastest as is reported by Dunham [Dunham 89] and which we also discovered as illustrated in Figure 6 (the unit of time in Figure 6 is relative). CAL is significantly slowed when posterior merging is done as shown in Figure 6.

3.3.5 Performance of CAL under metric #4: execution speed variability with respect to the number of approximating points

CAL performs particularly well against the split and merge algorithm as shown in Figure 5, which means that the timing curve for CAL is flat with respect to the number of approximating points.

2. Parameterized fitting easily allows fitting to arbitrary curves and minimizes perpendicular distance
1. Some minor changes to this general process are required at the beginning and end of each curve and for open versus closed curves.

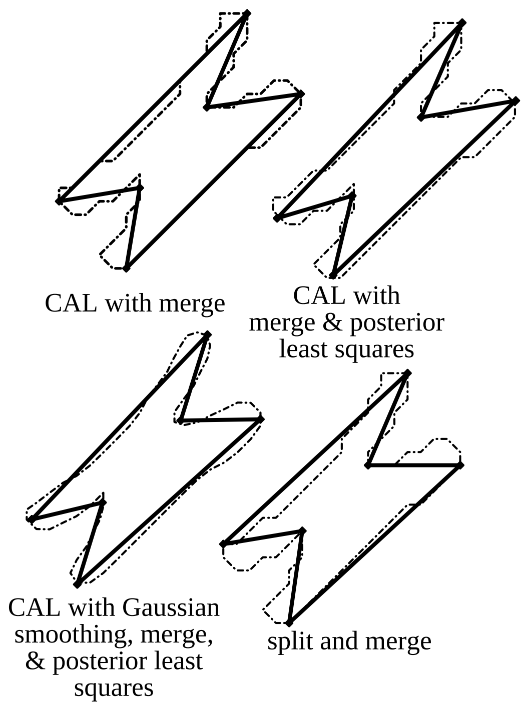
CAL with merge
CAL with merge & posterior least squares
CAL with Gaussian smoothing, merge, & posterior least squares
split and merge

Figure 3: A specific closed input curve (from Teh and Chen [Teh 89]) with curve approximation using CAL and split and merge.

3.3.6 *Performance of CAL under metric #5: execution speed and speed variability with respect to the size of the input curve*

All the efficient algorithms execute in linear time, $O(n)$, whereas the split and merge approach is $O(n^2)$, for n sampled data points in the input curve. However, it is with large curves that the main weakness in the CAL algorithm, namely sensitivity to quantization error, is evident. Figure 7 gives an example of CAL with and without merging on a large curve (approximately 2000 data points). Figure 6 shows that the time of execution of CAL is greatly increased when posterior merging is added.

4 *future work*

It is conceivable that the use of scale invariant forms of the CAL algorithm would be useful for some applications. For example, we could use a normalized function of chord and arc length, something like $(1/2)\sqrt{(S^2/C^2)-1}$, where C is chord length and S is arc length. This would allow us to define a system that doesn't require any input parameter.

Because the main weakness of CAL is the sensitivity to quantization error, and because the competing efficient algorithms effectively do an efficient type of merging, it would be worthwhile to investigate the combination of CAL and one of the other efficient approaches. It might also be fruitful to investigate an efficient but global feature preserving combination of the CAL approach and the split and merge method.

Even though optimal methods are known to be inefficient, it would also be helpful to use an optimal approach as a benchmark for testing the performance of CAL.

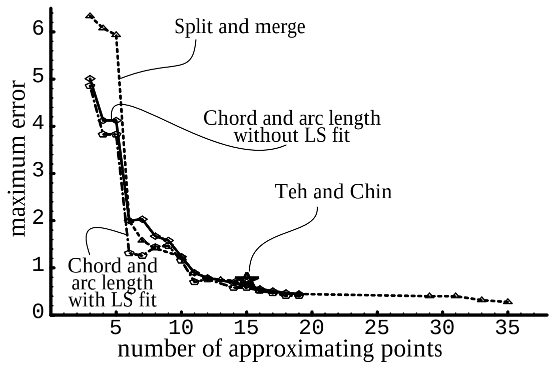

Figure 4: The maximum deviation error for the input curve of Figure 3.

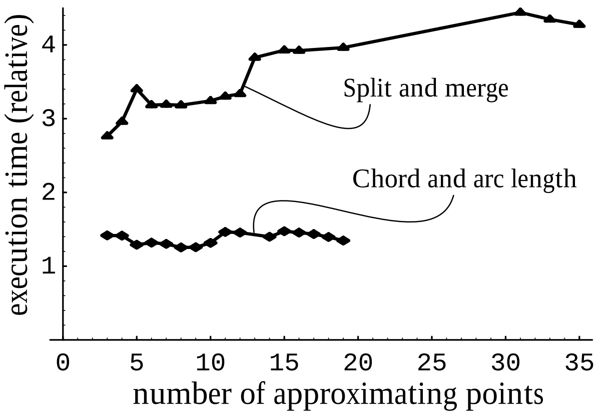

Figure 5: Timing performance as a function of the number of approximating points.

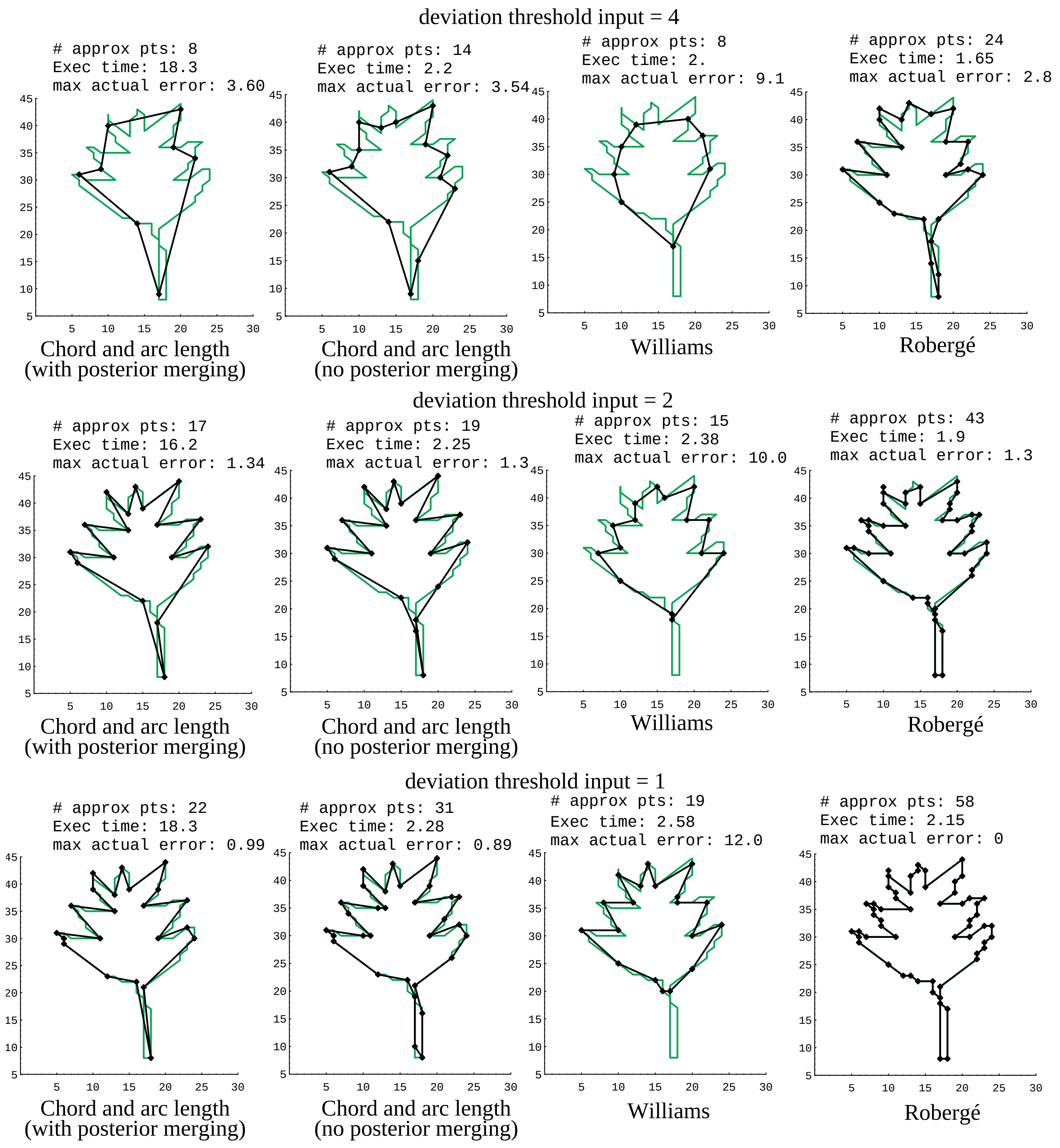

Figure 6: A specific closed input curve (from Teh and Chin [Teh 89]) with curve approximation using the same input deviation threshold with three efficient algorithms.

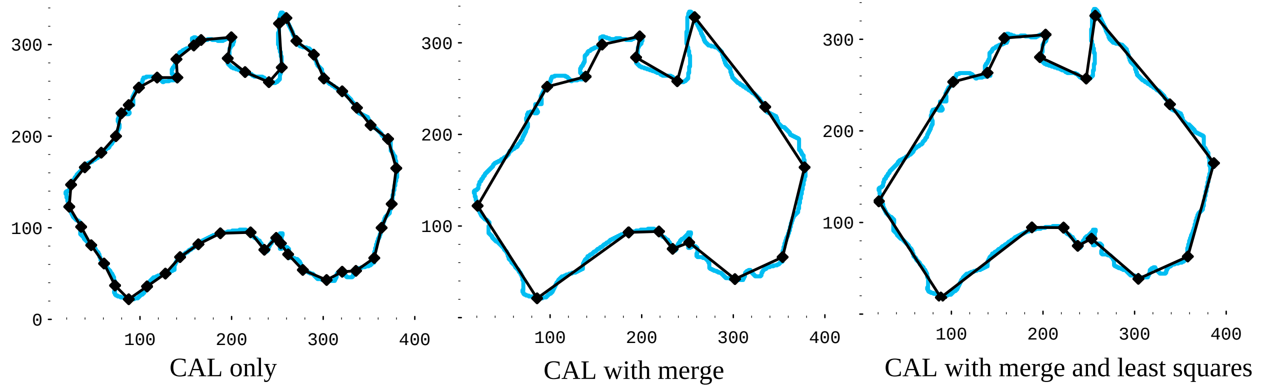

Figure 7: Curve approximation with CAL on an input curve (1708 points). Input deviation threshold = 12.

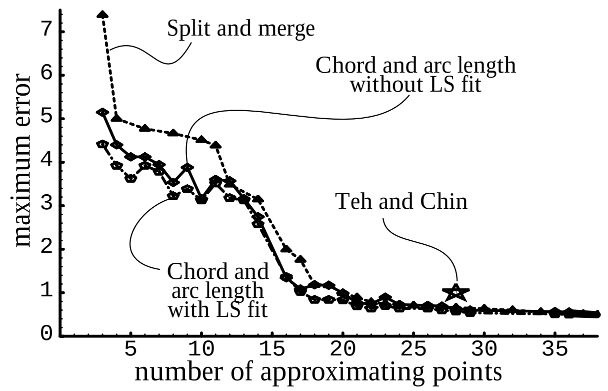

Figure 8: The maximum deviation error for the input curve of Figure 6.

5 conclusion

We have presented an efficient suboptimal method for piecewise linear curve approximation for space curves with integer or non-integer values.

It's excellent error performance (as measured by the maximum actual deviation error versus the number of approximating points), guaranteed error performance (proof in appendix), efficiency, and ability to handle non-integer valued space curves as well as integer valued planar curves are the key strengths of the approach presented. Because of its ability to handle space curves, we hope that this algorithm will not have application only to computer vision, but to other applications such as the representation of machine tool cutting paths.

The sensitivity to quantization error and the need, in such cases, for preliminary Gaussian smoothing and/or posterior merging, which adds significantly to the computation, is the major weakness of this approach. Perhaps a useful marriage between CAL and one of the other efficient methods would be profitable.

CAL plus posterior merging can be thought of as an efficient modification of the split and merge method in which the error is calculated indirectly instead of directly. CAL operating times are linearly proportional to the number of data points processed. Because split and merge operating times are proportional to the square of the number of data points, curves with many points can take forbiddingly long times to compute depending on how precise the approximation is, which implies that the compute time of the split and merge method varies with the size of the input deviation threshold (as shown in Figure 5). Alternatively, performance times of CAL are constant with respect to the deviation threshold. Another advantage of CAL is that it seems to be insensitive to the choice of the initial breakpoint, unlike split and merge [So 93]. The split and merge method chooses the same set of approximating points in each iteration, just adding more for increased accuracy. The approximating points chosen by CAL varies more widely. This is because the split and merge method accesses the data globally, CAL (and other efficient algorithms) access locally.

'C' code is available which realizes the CAL algorithm and can be accessed through anonymous ftp[1] at isdftp.cme.nist.gov. The necessary files are in the directory /pub/horst. The key files are main.c, curvefit.c, curvefit.h, v2d_math.c, and file_in. The input deviation parameter is user-specified within the file, curvefit.h. The input curve data must be in the same format as the data in the file called file_in.

1. For username type 'anonymous' and for the password enter your email address.

6 *appendix*

We now prove (EQ 6), but first we state some definitions. *Definitions*: Let $\alpha(i)$, $i = 1, 2, \ldots, n$, be a sequence of distinct points in the plane. Let $S = s(i_j, i_{j+1})$, the arc length function (EQ 3), $C = c(i_j, i_{j+1})$, the chord length function (EQ 2), $E_{\max} = e_{\max}(i_j, i_{j+1})$, the maximum distance from points $\alpha(i)$, $i = i_j, i_j + 1, \ldots, i_{j+1} - 1, i_{j+1}$, to the line segment formed by the points $\alpha(i_j)$ and $\alpha(i_{j+1})$, $\alpha(i_{\max})$, the point on the curve where the deviation from the chord line segment is at a maximum (illustrated in Figure 9 and Figure 10), $\xi_{\max}$, the distance from $\alpha(i_{\max})$ to the line formed by the points $\alpha(i_j)$ and $\alpha(i_{j+1})$, and $D = (1/2)\sqrt{S^2 - C^2}$.

Lemma: $\xi_{\max}$ ð D.

Proof: Construct a triangle formed by the points $\alpha(i_j)$, $\alpha(i_{\max})$, and $\alpha(i_{j+1})$ with base of length C and sides of lengths A and B. These definitions are illustrated for two different triangles in Figure 9 and Figure 10. Let b be the angle opposite the side of length B and let a be the angle opposite the side of length A. Note that $\xi_{\max} = E_{\max}$ only when a and b are less than or equal to $\pi/2$. In Figure 9 and Figure 10, $\xi_{\max}$ is the height of the triangle. Construct an isosceles triangle, as in Figure 11 and Figure 12, with base length and perimeter equal to those of the triangle in Figure 9 and Figure 10, respectively. Let h be the height of the isosceles triangle. Let $R = A + B$. Since the shortest distance between two points is the line segment connecting them, R ð S. From elementary geometry,

$$\xi_{\max} = h\sqrt{1 - \left(\frac{A - B}{C}\right)^2} \qquad \text{(EQ 10)}$$

which implies that $\xi_{\max}$ ð h. Since R ð S and $h = (1/2)\sqrt{R^2 - C^2}$, h ð D. Since $\xi_{\max}$ ð h ð D, $\xi_{\max}$ ð D.

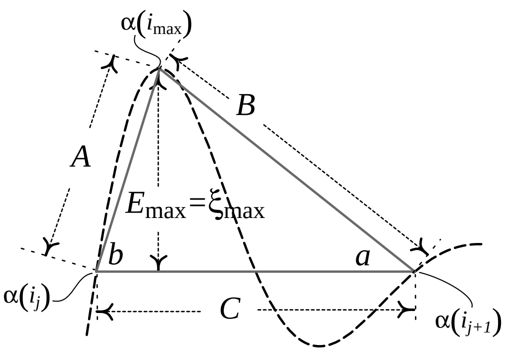

Figure 9: Triangle formed by the two endpoints of the arc and the point of maximum deviation in the case where a and b are less than $\pi/2$.

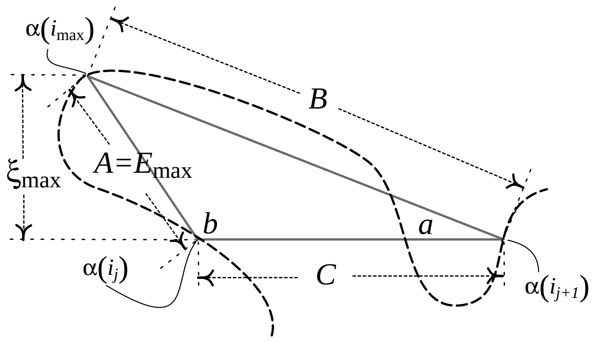

Figure 10: Triangle formed by the two endpoints of the arc and the point of maximum deviation in the case where $b > \pi/2$.

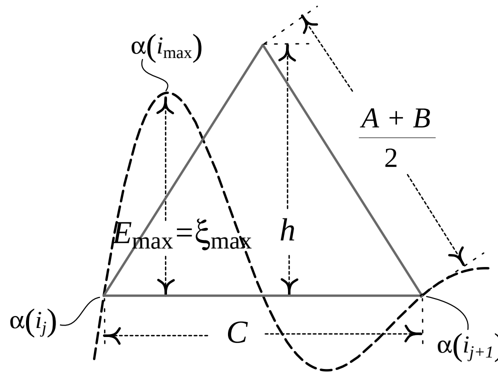

Figure 11: The isosceles triangle equivalent to the triangle of Figure 9.

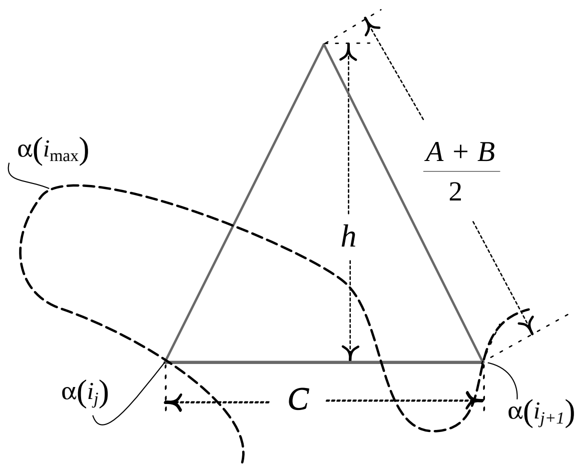

Figure 12: The isosceles triangle equivalent to the triangle of Figure 10.

The lemma shows that the distance from $\alpha(i_{\max})$ to the chord line is always less than $D = (1/2)\sqrt{S^2 - C^2}$. However, if *a* Š π/2, $E_{\max} = B$, and if *b* Š π/2, $E_{\max} = A$. In both these cases $E_{\max}$ Š $\xi_{\max}$ (see Figure 10). Therefore, the lemma alone is not sufficient to support the claim of (EQ 6). In other words, we need to prove that the distance from $\alpha(i_{\max})$ to the chord line **segment** is always less than *D* which is stated in the following theorem.

Theorem: $E_{\max}$ ð *D*.

Proof: There are only three cases to consider. We will now show that the theorem is established for each case.

Case 1: *a* < π/2 and *b* < π/2
Case 2: *b* Š π/2
Case 3: *a* Š π/2

Proof of Case 1: As in Figure 9, if *a* < π/2 and *b* < π/2, since $E_{\max} = \xi_{\max}$, the lemma implies that $E_{\max}$ ð *D*.

Proof of Case 2: If *b* Š π/2, as illustrated in Figure 10, $E_{\max} = A$. Construct an isosceles triangle (as in Figure 12). Let *h* be the height of this isosceles triangle. From the proof of the lemma, *h* ð *D*. By the Pythagorean relation,

$$2h^2 = A^2 + B^2 + 2AB - C^2, \qquad \text{(EQ 11)}$$

and by the law of cosines (from Figure 10),

$$0 = A^2 - B^2 - 2AC\cos b + C^2. \qquad \text{(EQ 12)}$$

If we add (EQ 11) and (EQ 12) we get,

$$2h^2 = 2A^2 + (2AB - 2AC\cos b). \qquad \text{(EQ 13)}$$

The term in parentheses in (EQ 13) Š 0, since $\cos b$ ð 0. Therefore, *A* ð *h* and since *h* ð *D*, $A = E_{\max}$ ð *D*.

Proof of Case 3: The exact same process presented in the proof for Case 2 is required to show that if *a* Š π/2, $A = E_{\max}$ ð *D*.

7 references

[Aoyama 91] Aoyama, Hiroshi and Kawagoe, Masahiro, "A piecewise linear approximation method preserving visual feature points of original figures," CVGIP: Graphical Models and Image Processing, Vol. 53, No. 5, (1991): 435-446.

[Chen 79] Chen, P. C., and Pavlidis, T., "Segmentation by texture using a co-occurrence matrix and a split-and-merge algorithm," *Computer Graphics, Image Processing*, **10** (1979): 172-182.

[Duda 73] Duda, R. O. and Hart, P. E., *Pattern Recognition and Scene Analysis*, New York: John Wiley, 1973.

[Dunham 89] Dunham, James George, "Optimum uniform piecewise linear approximation of planar curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, 1986, pp. 67-75.

[Fischler 86] Fischler, M. A. and Bolles, R. C., "Perceptual organization and curve partitioning," IEEE *Transactions on Pattern Analysis and Machine Intelligence*, January 1986.

[Grimson 90] Grimson, W. E. F., *Object Recognition by Computer: The Role of Geometric Constraints*, Cambridge, Massachusetts: The MIT Press, 1990.

[Jain 89] Jain, A. K., *Fundamentals of Digital Image Processing*, Englewood Cliffs, N.J.: Prentice Hall, 1989.

[Jain 96] Jain, A. K., "Object matching using deformable templates," IEEE *Transactions on Pattern Analysis and Machine Intelligence*, March 1996.

[Nevatia 80] Nevatia, R. and Babu, K. R., "Linear Feature Extraction and Description," *Computer Graphics and Image Processing* **13**, pp. 257-269.

[Pavlidis 77] Pavlidis, Theodosios, "Polygonal approximations by Newton's method," *IEEE Transactions on Computers* **C-26**, No. 8, August 1977.

[Ramer 72] Ramer, Urs, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing* **1**, 1972, pp. 244-256.

[Robergé 85] Robergé, James, "A data reduction algorithm for planar curves," *Computer Vision, Graphics, and Image Processing* **29**, 1985, pp. 168-195.

[Sato 93] Sato, Yukio, "Piecewise linear approximation of plane curves by perimeter optimization," *Pattern Recognition*, Vol. 25, No. 12, 1992, pp. 1535-1543.

[Sklansky 80] Sklansky, J. and Gonzalez, V., "Fast polygonal approximation of digitized curves," *Pattern Recognition*, Vol. 12, 1980, pp. 327-331.

[So 93] So, W. C. and Lee, C. K., "Invariant line segmentation for object recognition," *Proceedings of IECON '93*, Maui, Hawaii, Nov. 15-19, 1993, pp. 1352-1356.

[Teh 89] Teh, Cho-Huak and Chin, Roland T., "On the detection of dominant points on digital curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. II No. 8, 1989, pp. 859-872.

[Williams 77] Williams, Charles M., "An efficient algorithm for the piecewise linear approximation of planar curves," *Computer Graphics and Image Processing* **8**, 1978, pp. 286-293.