A Simple Algorithm for Efficient
Piecewise Linear Approximation of Space Curves

John Albert Horst and Isabel Beichl
The National Institute of Standards and Technology
Gaithersburg, Maryland, USA
internet: {john.horst, isabel.beichl} @nist.gov
ftp: isdftp.cme.nist.gov/pub/horst
voice: (301)975-(3430, 3821)

Abstract

An on-line method for 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, on-line alternative to the split and merge approach. Other efficient methods operate only on planar curves, whereas the approach we offer is also appropriate for space curves. A simple function of chord and arc length is used to form the initial set of approximating points. Preliminary Gaussian smoothing, posterior merging and least squares fitting are optional and can be done depending on the application. Algorithm performance has been tested on a variety of planar curves and comparisons made with other piecewise linear curve approximation algorithms.

1 Introduction

Planar curve approximation methods have received much attention. 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 compression can be achieved, particularly for large input curves, depending on the accuracy of the approximation
- many real-time applications, such as graphics rendering, can realize significant speedups through curve data compression 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, correlations with model templates is simplified. 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, e.g., \(O(n^2)\), 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

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

2 Related research

Pavlidis [Pavlidis 77] and Dunham [Dunham 89] offer optimal approaches. 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 [Aoyama 89] [Fischler 86]. Fischler and Bolles have developed an 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. Some real-time systems might be less concerned with human-like approximations and more with simplicity and efficiency.

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 also defined for digitized, planar curves only.

Roberg [Roberg 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 noticeably loose compared to other approaches for the sample curves we investigated (see Figure 1).
- For some curves Roberg’s algorithm distorts the curve significantly.
An additional parameter is required for input other than deviation. Williams' method [Williams 77] is not reliable since it cannot guarantee an upper bound on the approximation error (see Figure 1). This fact is enough to disqualify this approach for many applications, in which deviation of approximation must be tightly controlled. However, it is conceivable that a minor modification to the Williams 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, Crimson 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 deviation 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 $n$ points in the raw 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)
- several attempts to improve the efficiency of the split and merge approach come with an increase in complexity [Nevatia 80, So 93]
- split and merge is an off-line (batch processing) algorithm

Many algorithms require input points with contiguous integer coordinates (often defined by Freeman chain codes [Teh 89]). However, there are applications where a more generic algorithm would be useful. Quantization error, missing or redundant points in the curve argue for a method that can process more generic curves.

3 The chord and arc length algorithm

We now describe the chord and arc length (CAL) method for piecewise linear space curve approximation. Steps one and three describe CAL. Step two is optional preliminary Gaussian smoothing; step four is optional posterior merging; steps five through seven describe optional posterior least squares fitting. These optional steps require significant additional computation, but can still be done in an online manner.

1. determine whether the curve is open or closed
2. do local Gaussian smoothing on the raw input curve, if required, 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 
   \[ \frac{1}{2} \sqrt{S^2 - C^2} \]
   is greater than the deviation threshold 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

Note that step three, which is the heart of CAL, does not require that the input curve be a set of discrete points. CAL will work well even if the input curve is represented as a continuous function.

It is straightforward to prove that CAL (steps one and three only) always approximates the input curve points to within the deviation threshold. This fact has been rigorously proven [Horst 96]. However, the following description should also be convincing. Consider the curve as a flexible but non-stretchable rope. If we fix any two points on the curve in space, grasp the rope at the midpoint between these two points, and stretch the rope to the limit, we see that a triangle is formed. The height of this triangle is equal to \( \frac{1}{2} \sqrt{S^2 - C^2} \) and is always greater than the distance between any point on the curve and the chord line segment.

Many algorithms require input points with contiguous integer coordinates (often defined by Freeman chain codes [Teh 89]). However, there are applications where a more generic algorithm would be useful. Quantization error, missing or redundant points in the curve argue for a method that can process more generic curves.
Figure 1: 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 2: The maximum deviation error for the input curve of Figure 1.

Figure 3: Relative execution times for the input curve of Figure 1.

4 Algorithm performance

We have done extensive testing of CAL against competing algorithms for a variety of different measures:

- the actual maximum deviation of the original curve points from each approximating line segment for a given input deviation threshold
- the number of approximating points required for a given actual maximum deviation (examples of this and the previous measure are given in Figures 1 and 2)
- the speed of execution
- the variability in the speed of execution with respect to the number of approximating points (example shown in Figure 3)
- the variability in the speed of execution with respect to the size of the input curve

We have analyzed the performance of CAL under these various metrics against several different approaches and it performs well in comparison. For example, as is seen in Figure 1, Robergé [Robergé 85] is fastest in execution time, but has more approximating points than CAL or Williams [Williams 77]. The split and merge algorithm has a variable execution time based on the number of approximating points whereas CAL is not as variable as is shown in Figure 3. Because CAL accumulates arc length, CAL is sensitive to quantization error and will sometimes select more dominant points than necessary as can be seen in Figure 1. This is particularly evident for integer-valued curves with a large number of points as is evident in Figure 4. In such cases, preliminary Gaussian smoothing and posterior merging, while adding significantly to the computational cost, will virtually eliminate the problem.
5 Conclusion

CAL’s good error performance (as measured by the maximum actual deviation error versus the number of approximating points), guaranteed error bound (approximation always within threshold), simplicity, 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. Depending on the application, the sensitivity to quantization error is a weakness of this approach. However, the problem can be ameliorated by preliminary Gaussian smoothing and/or a posteriori merging. Quantization error and its amelioration are shown in Figure 4.

6 References


