Compressed Piecewise-Circular Approximations of 3D Curves

Alla Safonova and **Jarek Rossignac**
College of Computing / GVU Center
Georgia Institute of Technology
<asafonov@andrew.cmu.edu> and <jarek@cc.gatech.edu>

ABSTRACT

*We propose a **compact approximation** scheme for **3D** curves. Consider a **polygonal curve** P, whose **n vertices** have been generated through adaptive (and nearly minimal) sampling, so that P approximates some original 3D curve, O, within tolerance ε_0. We present a practical and efficient algorithm for computing a continuous 3D curve C that approximates P within tolerance ε_1 and is composed of a chain of **m circular arcs**, whose end-points coincide with a subset of the vertices of P. We represent C using 5m+3 scalars, which we compress within a carefully selected quantization error ε_2. Our approximation uses a total of less than **7.5n bits**, when O is a typical surface/surface intersection and when the error bound $\varepsilon_1+\varepsilon_2$ is less than 0.02% of the radius of a minimal sphere around O. For less accurate approximations, the storage size drops further, reaching for instance a total of **n bits** when $\varepsilon_1+\varepsilon_2$ is increased to 3%. The storage cost per vertex is also reduced when ε_0 is decreased to force a tighter fit for smooth curves. As expected, the compression deteriorates for jagged curves with a tight error bound. In any case, our representation of C is always more compact than a polygonal curve that approximate O with the same accuracy. To guarantee a correct fit, we introduce a new **error metric** for ε_1, which prevents discrepancies between P and C that are not detected by previously proposed Hausdorff or least-square error estimates. We provide the details of the algorithms and of the geometric constructions. We also introduce a conservative speed-up for computing C more efficiently and demonstrate that it is sub-optimal in only 2% of the cases. Finally, we report results on several types of curves and compare them to previously reported polygonal approximations, observing **compression ratios** that vary between **15:1** and **36:1**.*

1 INTRODUCTION AND OVERVIEW

Three-dimensional (3D) curves play a fundamental role in many 3D applications. For example, they may represent surface/surface intersections in CAD/CAM, blood vessel lines in medical 3D imaging, point trajectories in 3D graphics and animation, or flow-lines in engineering simulation. We focus on the efficient computation of a compact approximation of such curves. Most applications represent 3D curves as an ordered set of sample points, assuming an agreed upon interpolation or approximation scheme. A polygonal interpolation is often chosen for its simplicity.

When the original curve *O* is **known**—or at least when an error bound may be estimated—the samples are usually spaced adaptively along *O,* so as to avoid over-sampling, while guaranteeing that the resulting polygonal approximation *P* lies within some prescribed error bound ε_0 from *O*. The samples generated in this manner are the vertices of *P*.

When the original curve is **unknown**, or when the estimation of the maximum deviation between *P* and *O* is too expensive, a much larger than necessary set of samples may be generated initially. Then a polynomial curve *P* that passes sufficiently close to all the samples is generated. Its vertices are usually selected from the initial samples. An adaptive selection process may be used in an attempt to minimize the number *n* of such vertices, while preserving closeness between *P* and all the initial samples.

In either case, the resulting polygonal approximation *P* may be represented using *3nB* bits, where B is the number of bits needed to represent each quantized coordinate of each one of its *n* vertices. To report our results more precisely, we will not take into account additional compression that could possibly be achieved through vertex prediction or vector quantization [40], because these gains vary too much with the size of the sampling and with the smoothness of the curve, and because these more general compression techniques may be applied to most representations in a post-processing step.

Thus, we address the specific problem of computing a piecewise-circular approximation (abbreviated PCA), called *C*, for a polygonal curve *P*. The error between *P* and *C* is bounded by an *a priori* defined tolerance $\varepsilon_1+\varepsilon_2$. Here, ε_1 is the maximum deviation between *P* and an initial version of *C* computed with full precision. The additional error ε_2 is introduced by quantizing the representation of *C* for better compression.

Note that our construction procedure may be used either to find directly a very tight fit to an over-sampled approximation *P* of an unknown curve *O* or to produce a compressed approximation of a polygonal curve *P*, which itself may have been derived as a nearly optimally sampled approximation of some original curve *O*. In order to report our compression results in a fair manner, we cannot describe them in terms of compression ratios between a representation of *C* and a representation of an over-sampled polygonal approximation of the original curve. Therefore, we report our results by comparing the storage of *C* to the storage of a nearly optimally sampled polygonal approximation *P* of *O*.

We report our results for three types of curves: (a) the typical surface/surface intersections commonly found in CAD/CAM models, (b) jagged 3D curves that may for example form the bounding loop of a set of triangles selected from the boundary of a 3D model, and (c) helixes, which are simple 3D curves that may be easily reproduced and can serve as a benchmark for

comparing 3D fitting and compression techniques. We have tested our approach on a spectrum of these curves to ensure that the reported results are representative of the results that should be expected in practice.

We represent C using 5m+3 scalars: 3 for the coordinates of each vertex and 2 to encode how each arc deviates from a straight line. We compress them through **quantization** within an error bound ε_2. We select ε_2 and ε_1 so as to minimize the total storage of C, while keeping $\varepsilon_1+\varepsilon_2$ below a prescribed threshold.

We have found that our approximation uses less than ***7.5n* bits,** when O is a typical surface/surface intersection and when the error bound $\varepsilon_1+\varepsilon_2$ is less than 0.02% of the radius of a nearly minimal sphere containing O. As expected, the storage drops further for less accurate approximations. For example, it reaches ***1.0n* bits**, when $\varepsilon_1+\varepsilon_2$ is increased to 3%. In other words, we take a polygon P whose n vertices have been generated though an adaptive sampling of a 3D curve O. P requires $3nB$ bits of storage. We generate an approximation C of P and compress it to $1.0n$ bits. In practice, this *3B:1* **compression ratio** varies between ***15:1*** and ***36:1***, when B varies between *5* and *12* bits per coordinate.

It should be noted that, for smooth curves, better compression is achieved when we force a tighter fit between P and O. This observation may be explained by understanding that we usually require more arcs to provide a tight fit C to a less smooth polygonal approximation P of a curve O. As expected, the compression deteriorates for jagged or noisy polygonal curves with a tight error bound. Nevertheless, our representation of C is **always more compact** than P and hence there is never a storage penalty when using a PCA, rather then a polygonal approximation.

The proposed approach is based on a new algorithm for **finding**, if it exists, **a circular arc** A that has as end-points two vertices, V_i and V_j, of P and that lies within a prescribed tolerance ε from the subset S of P bounded by these two vertices. There is a two-parameter family of circular arcs joining V_i and V_j. We have developed a fast conservative test for deciding whether such an arc exists and a method for finding the arc in a time proportional to the number of vertices of S. The test is based on the computation of the plane that is the bisector of the wedge of all planes that pass through V_i and V_j and are closer than ε to all vertices of S. If the wedge is empty, we can guarantee that no arc exists. Otherwise, if the bisector plane exists but does not contain an acceptable arc, we consider that no such arc exists. This choice corresponds to a wrong (conservative) decision in only 2% of the cases, and thus leads to always valid, although sometimes (rarely) slightly sub-optimal approximations.

To guarantee a correct fit, we introduce a new **error metric**, which prevents discrepancies between P and C that are not detected by previously proposed Hausdorff or least square error estimates. Our error metric combines two conditions:

- The first condition requires that the arc be **inside the tube** $S\uparrow\varepsilon$. The notation $S\uparrow\varepsilon$, introduced in [41], refers to a grown version of S, defined as the set of points at distance no more than ε from S.
- The second condition requires that there exist an **ordered sequence of points** along the arc such that the first point be closer than ε from V_i, that the second point be closer than ε from the first vertex of S after, V_i and so on.

We present and compare two different algorithms for **selecting** the set of **vertices** of P that will be **interpolated** by C and will serve as the end-points of the arcs of C.

- The first algorithm is based on an efficient **greedy** process. It has $O(n\log m)$ time complexity, where m is the maximum number of points approximated by a single arc. However, it does not always produce an optimal result.
- The second algorithm is **optimal**, but more expensive. It has time complexity $O(n^3)$. It is inspired from an approach developed by Imai and Iri [13] for computing polygonal approximations, which we have extended to circular arcs.

We provide experimental comparisons of the compression results and report running times for both the optimal and the greedy algorithms.

The rest of the paper is organized as follows. First we review the most relevant prior art in curve approximation, fitting, and compression. Then, we discuss the strategies for selecting the interpolated vertices, compare error measures, present the details of our algorithm, and explain the geometric constructions that it uses. Finally, we present our results and compare them to Douglas and Peucker's [7] and Barequet et al [39] polygonal fit algorithms.

2 RELATED WORK ON CURVE FITTING

Curve fitting through the set of given points has been studied in several application areas. Published approaches can be grouped into **interpolating** and **approximating**, depending on whether the resulting curve passes through all of the data points or not. Non-linear curves (e.g. splines, b-splines, minimal energy splines, bi-arcs) have been used for this purpose [1, 2]. In general, approximation approaches minimize the error between the approximating curve and the set of given points.

We focus on fitting a curve to a given polyline. By bounding the error between the approximating curve C and the given polyline P (rather than the vertices alone), we guarantee that C does not exhibit undesirable wiggles between the points. Furthermore, if P lies within some error ε_0 from an original curve O and if C lies within error ε from P, we guarantee that C approximates O within error $\varepsilon_0+\varepsilon$.

Polygonal approximations of polylines have been studied extensively in 2D for GIS and other applications. They have been surveyed in [3-6]. The algorithms in this category can be roughly divided into **adaptive refinement** algorithms [7-9], **decimation** algorithms [10,11], and global **optimization** algorithms [12-21]. Barequet et al [39], give an optimal solution for

polygonal approximations of polylines in three and higher dimensions. By optimal, we mean that the approximating polyline *C* has the minimum number of vertices, while not deviating from *P* by more that a prescribed error. We focus on fitting **circular arcs** to a given polyline, as we believe that circular arcs provide **better compression** for many types of curves. They are especially good for approximating smooth 3D curves, like the typical surface/surface intersections found in CAD/CAM. Our technique also performs well on smooth curves with small noise, when the error tolerance exceeds the magnitude of the noise.

A significant amount of work has also been devoted to fitting **arcs** and **bi-arcs** through points in 2D [23-33]. An overview of these methods can be found in [25]. **None** of the above methods, however, guarantee a **fit to a polyline**, and none is easily **extendable to 3D**. All approaches that we surveyed minimize the error (maximum or average) between vertices in the given polyline and the approximating curve, and thus do not guarantee an error bound between *P* and *C*.

We only know of a few publications reporting techniques for **fitting non-linear curves to polylines**. Saux and Daniel [22] propose an algorithm that uses a B-spline curve to approximate the original polyline. They define a new error measure that guarantees the error bound between *P* and *C*. A B-spline curve, has a second-degree **continuity**, and thus can be advantageous to some applications. However, in applications, where G^1 and G^2 continuity is not as important as data reduction, our approach is preferable. The compression rate shown by Saux and Daniel is close to the compression rate of Douglas and Peucker's [7] polygonal approximation algorithm. The results that we present in this paper show that **our PCA approximation provides improvement in the compression rate** over the polygonal approximations reported in [7], and hence over storage requirements of the B-splines produced by the technique of [22].

Instead of fitting a non-linear curve, *C*, to the polyline, *P*, produced by sampling the original curve *O*, we could **approximate the original curve directly** with a non-linear curve such as our PCA. Typical examples of such approaches are discussed in [34-37]. Previously reported approaches of this type are either computationally prohibitive, or specific to a particular type of parametric curve model, or both. A notable exception is the work reported by Tseng and Chen's [30], which offers a general algorithm for approximating 3-dimensional curves by bi-arcs with an approximation error defined by the largest allowed deviation distance between the curve and bi-arcs. Their algorithm fits a bi-arc between two points, u_s and u_e, on the original curve and then finds the largest **deviation** distance between the bi-arc and a set of **samples** uniformly spaced along the original curve between u_s and u_e. Such an error estimate ignores the shape of the curve between the samples and fails to detect folds in *O,* as shown in Fig. 1e. The limitations of the various error measures are discussed below in Section 4.

3 STRATEGIES FOR SELECTING INTERPOLATED VERTICES

Before drilling down to the details of error estimation and arc fitting, we present in this section the overall strategy for building a PCA, because it is independent of these details.

Given a 3D polyline, *P*, with vertices P1, P2…Pn and a tolerance factor, ε, we construct a PCA, *C*, consisting of circular arcs, whose end-points coincide with an ordered subsequence of vertices of *P*. These endpoints are called the **interpolated vertices**. We want to minimize the number of arcs in *C*, while guaranteeing that the error between *P* and *C* does not exceed the prescribed tolerance factor ε.

We have explored two algorithms for **selecting** the **interpolated vertices**, while trying to minimize the number of acceptable arcs. They offer a tradeoff between compression and complexity. Several similar optimal and sub-optimal algorithms have been previously developed for selecting interpolated vertices for fitting polylines, B-spines, or other non-linear curves. An overview of these algorithms may be found in [25, 26].

3.a Optimal Algorithm - OPCA

Our **optimal** approach, abbreviated **OPCA**, follows the approach of Imai and Iri [13] for approximating curves with polygonal lines. We first build a directed graph, *G,* whose nodes correspond to the vertices of *P*. Two vertices V_i and V_j are connected by a directed edge, E_{ij} in *G*, if and only if an acceptable arc *A* starting at vertex V_i and ending at vertex V_j exists. The time complexity for building *G* is $O(n^3)$, since the test of the existence of an acceptable arc between two vertices V_i and V_j has complexity $O(n)$, and since we have to test all possible pairs of vertices V_i and V_j, where $i<j$. Then, we use a breadth-first search to find the shortest path from vertex V_1 to vertex V_n. This path gives us an optimal PCA. The time complexity of the complete OPCA approach is $O(n^3)$, since it takes $O(n^3)$ to build *G* and $O(n^2)$ to find shortest path using breadth-first search.

3.b Doubling Algorithm - DPCA

Our more **practical** approach, abbreviated **DPCA**, is based on a greedy process, which first attempts to find an acceptable arc from V_1 to V_2. If an arc exists, it looks for an acceptable arc from V_1 to V_4. It continues in this manner, each time doubling the distance (number of edges) to the end vertex of the arc until it fails to find an acceptable arc. Then, a binary search on the last interval identifies the endpoint, V_j, of the longest acceptable arc. We then proceed to fit arcs to the remaining portion of *P*, starting at V_j. This greedy algorithm does not produce an optimal result and has a time complexity of $O(n\log k)$, where *k* is the maximum number of points approximated by a single arc. For a complete analysis of the running time of this algorithm please refer to [26]. In section 7, we show that both algorithms produce similar results and thus we advocate DPCA.

The following sections focus on computing, if there exists, one arc between two candidate vertices V_i and V_j.

4 COMPARING ERROR MEASURES

Consider a single arc, *A,* in *C* connecting two interpolated vertices V_i and V_j. Define the polyline *S* to be the subset of *P* between vertices V_i and V_j. Let $S_1, S_2 \ldots S_K$ denote the vertices of *S*. Thus $V_i=S_1, V_{i+1}=S_2 \ldots V_j=S_k$. An arc *A* is an **acceptable arc** if the error between *A* and *S* is less that ε. We discuss here how we define the **error between *A* and *S***.

Several error measures have been proposed. In this section we illustrate their shortcomings and introduce a new, more precise error measure, which is used by our approach.

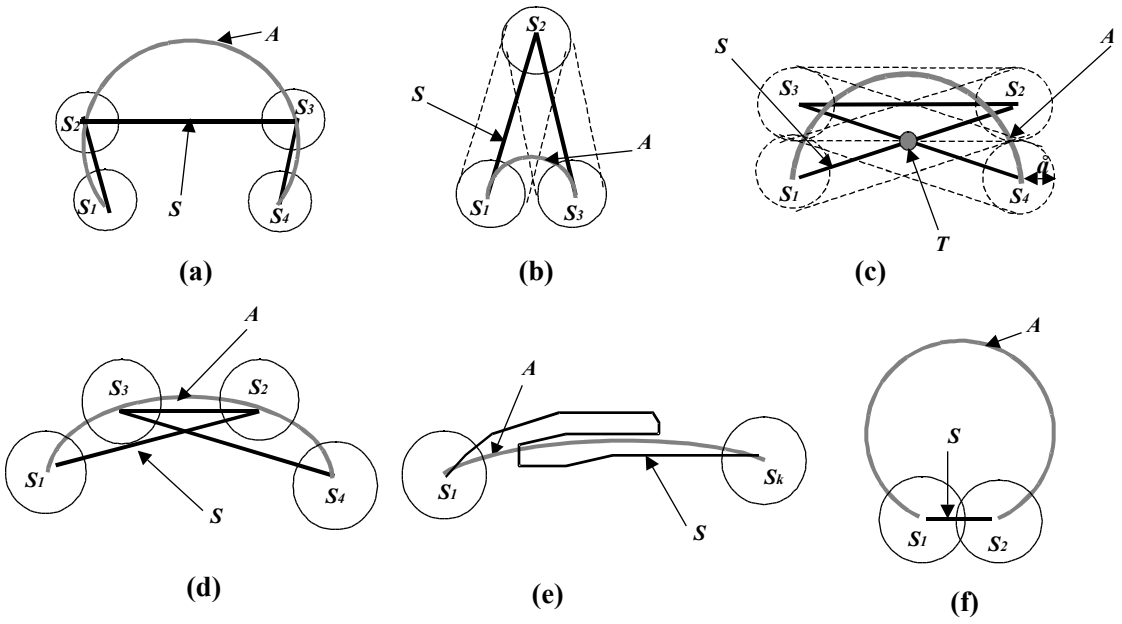


Fig 1: *(a) Drawbacks of expressing the error as the maximum distance to vertices. (b) Drawbacks of the tolerance zone error measure. An error between vertex V_2 and A is much more than* ε. *(c) Example showing that the Hausdorff error measure is not the same as error measure in 4.4. (d and e) Drawbacks of the combined maximum-norm and tolerance-zone error measure, and of the Hausdorff distance. (f) Drawback of the error measure defined in 4.5.*

4.a Least square error measure

The **least square** error measure is defined as $\frac{1}{k}\sum_{i=1}^{k} d(A,S_i)^2$, where $d(A, S_i)$ is a shortest distance between an arc and a point.

The use of a least square error measure makes it easy to compute the arc *A* for which it is minimal. However, the resulting solution does not guarantee a bound on the maximum error between the vertices of *S* and *A*. Furthermore, it does not guarantee an error bound on the portions of *S* between the vertices.

4.b Maximum distance to vertices

Under the **maximum distance to vertices** error measure, *A* is acceptable if the maximum **distance** between *A* and the **vertices of *S*** does not exceed ε. However, it still does not restrict the distance between *A* and points of *S* other than the vertices (Fig. 1a).

4.c Tolerance zone error measure

Under the **tolerance zone** error measure, *A* is acceptable if it lies in the **tube** defined as the Minkowski sum, $S \uparrow \varepsilon$, of *S* with a ball of radius ε centered at the origin. However, *S* may be further than ε from sections of *A*, as illustrated in Fig. 1b.

4.d Combined tolerance zone and maximum distance to vertices

Combining the **maximum norm** error with the **tolerance zone** error solves the problems illustrated in Fig. 1a and 1b. Unfortunately, an unacceptable deviation of the approximating curve from the original polyline may still occur (Fig. 1d and 1e). Such situations may happen, because, under this combined error, both curves are viewed as unordered sets of points. The problem shown in Fig. 1e has been previously pointed out by Yang and Du [26]. To deal with this case, they introduced the notion of circular monotone 2D curves, which have a polar representation with respect to the arc midpoint. They used it in conjunction with the maximum norm error measure. Consequently, we conclude that their approach suffers from the shortcomings discussed above (in 4.2).

4.e Reversed tolerance zone

Under the **reverse tolerance zone** error measure, A is acceptable if S lies in the tube $A\uparrow\varepsilon$. However, portions of A may be further from S as illustrated in Fig. 1f.

4.f Hausdorff distance

This **Hausdorff distance** error measure combines the error measures defined in 4.c and 4.f into a symmetric error measure, where: $S\subset A\uparrow\varepsilon$ and $A\subset S\uparrow\varepsilon$. It still allows an unacceptable deviation between A and S (Fig. 1d and 1e). Note that the Hausdorff error measure is not the same as the error measure in 4.d. Fig. 1c illustrates the difference between the Hausdorff distance and the combined measure defined in Subsection 4.d: A and S are within error ε according to 4.4, but not according to the Hausdorff error, since point T on S is outside the tolerance zone around A.

4.g Fréchet distance

To overcome the drawbacks illustrated above, we advocate a **new error measure**, $\delta_A(A, S)$, inspired by [21] and defined as follows: Let B_i denote a **ball** of radius ε with center at S_i, for each vertex, S_i, of S. The **caplet** T_i is the **convex hull of two consecutive balls** B_i, B_{i+1}. Let $A(t)$ be a point on arc A for $t\in[0,1]$. We orient the arc A so that $A(0) = S_1$ and $A(1) = S_k$ (Fig. 2).

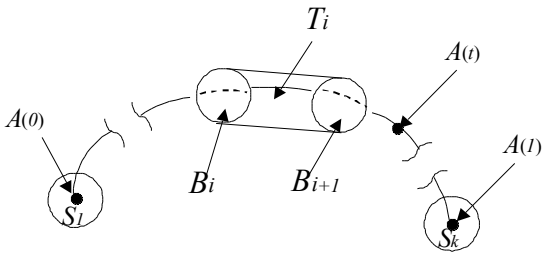


Fig 2: *B_i is a ball of radius ε with center at S_i. A caplet T_i is a convex hull of two consecutive balls B_i, B_{i+1}.*

Definition 1. We say that $\delta_A(A, S) \le \varepsilon$, if the following two requirements are satisfied:

- The **order requirement**: there exists a sequence of non-decreasing parameters $t_1, t_2 \dots t_k$ such that $A(t_i)\in B_i$ for $i=1\dots k$.
- The **tolerance requirement**: $A(t)\in T_i$ for $t\in[t_i, t_{i+1}]$.

This metric adds the order requirement to the Hausdorff error measure, thus eliminating the problems shown in Fig. 1.

Let $\delta_F(A, S)$ denote the Fréchet distance between curves P and C, as defined in [17, 37, 38]. $\delta_F(P,C) \le \varepsilon$, if there exist two monotone (non-decreasing) functions, α and β, of [0, 1] onto itself, such that $d(\alpha(t),\beta(t)) \le \varepsilon \quad t\in[0,1]$. In APPENDIX A, we prove that **our error measure is equivalent to the Fréchet metric**, for the case of an arc and a polyline. We use our formulation, δ_A, instead of the Fréchet formulation, for its computational advantage.

In the next two sections, we present an algorithm for finding an acceptable arc, A, if it exists, or for deciding that no acceptable arc exists for a given polyline S. A is acceptable if $\delta_A(A, S) \le \varepsilon$.

5 REDUCTION TO A 2D PROBLEM

In this section, we explain how to reduce our search for an acceptable arc A to a two-dimensional problem.

5.a Finding a stabbing plane for a single arc

Consider a family of **planes** passing through S_1 and S_k. Each plane $W(\alpha)$ in this family is defined by an **angle parameter**, α, which specifies its orientation around a line L through S_1 and S_k. Each ball B_i, for $1<i<k$, of radius ε around S_i defines an interval $[\alpha_1,\alpha_2]$, in which values of α represent planes $W(\alpha)$ that intersect B_i. Let $W(\alpha_0)$ be a plane passing through the center of B_i (Fig. 3). The **interval** $[\alpha_1,\alpha_2]$ is $[\alpha_0-\beta, \alpha_0+\beta]$, where $\beta=arcsin(\varepsilon/D)$ and D is the distance from the center of B_i to L.

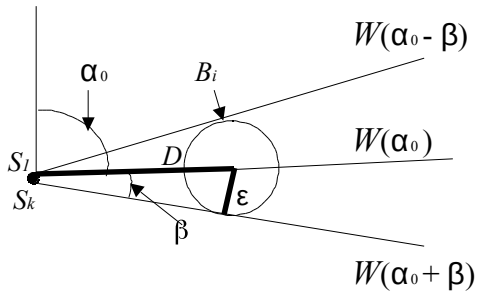


Fig 3: *Finding stabbing interval $[\alpha_1, \alpha_2]$ for ball B_i.*

We compute the **intersection** of all these **intervals**. If the intersection is empty, there is no plane that stabs all the balls and thus no acceptable arc. If the intersection is not empty, we pick its **central value**, α^*, and reduce our search for A to $W(\alpha^*)$.

We pick this bisecting plane $W(\alpha^*)$, because, according to our analysis and to the statistical data that we have collected on the set of all curves listed in Section 7, it has the **highest probability** of containing an acceptable arc A (Fig. 4). According to our experiments, the conditional probability that there be a plane containing an acceptable arc, given that $W(\alpha^*)$ does not, is less than 2%. Thus, if $W(\alpha^*)$ does not contain an acceptable arc, we declare that no such arc exists, making a wrong, although **conservative** decision in only 2% of the cases.

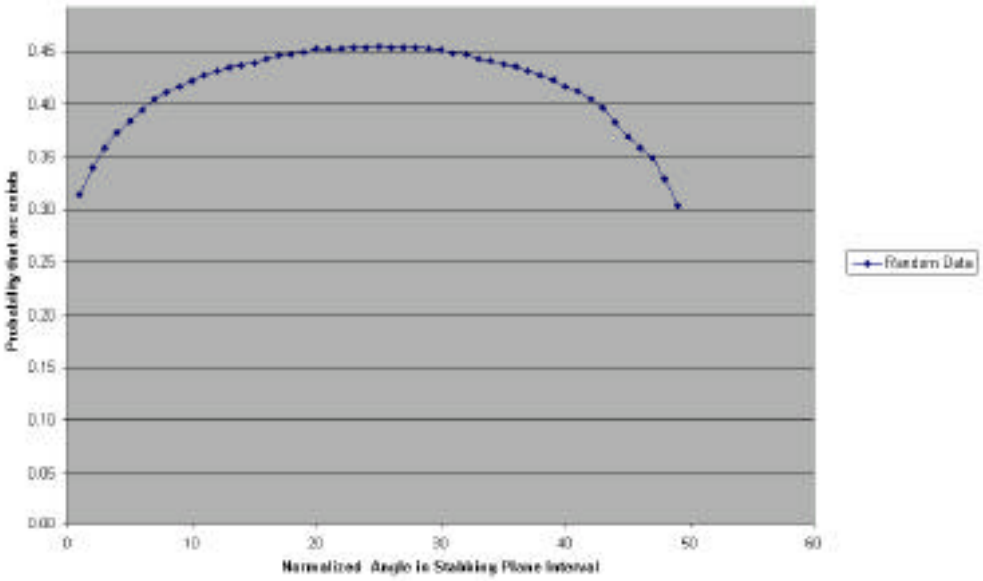


Fig 4: *Probability that an arc exists in the stabbing plane interval.*

By focusing on $W(\alpha^*)$, we have reduced our search for an acceptable arc A to a **2D problem**. In the next sub-section, we formally state the 2D problem. In Section 6, we present our approach for solving it.

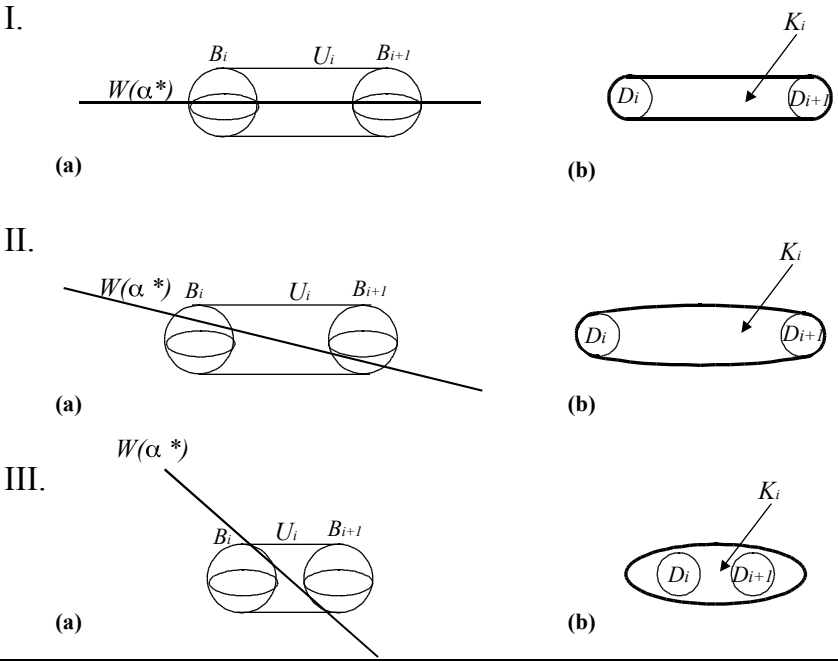


Fig 5: *Three configurations for the intersection of the caplet T_i with the plane $W(\alpha^*)$. The right column (b) corresponds to the cross-section by the plane $W(\alpha^*)$, shown on the left (a).*

5.b Intersecting the tolerance zone with the stabbing plane

Recall that **caplet** T_i is the convex hull of two balls of radius ε. It is bounded by a portion of a cylinder and by two hemispheres. The **intersection**, K_i, of **plane** $W(\alpha^*)$ with each caplet T_i, is convex and bounded by a single smooth closed curve made of at most four arcs that are portions of ellipses, lines, or circles. We use the term **2D-caplet** to refer to this intersection. The intersection of $W(\alpha^*)$ with each ball B_i, is a **disk** D_i. Note, that the disks D_i and D_{i+1} lie inside K_i.

Depending on the angle between the plane $W(\alpha^*)$ and the central line of a caplet T_i, several cases for the boundary of K_i are possible. Fig. 5 illustrates three of them: **Case 1:** When plane $W(\alpha^*)$ is equidistant from the centers of both disks, the boundary of K_i consists of two lines and two semi-circles. (When $W(\alpha^*)$ is tangent to B_i and B_{i+1}, this boundary degenerates to a line segment.) **Case 2:** Let U_i be a tube of radius ε around the line through S_i and S_{i+1}. K_i is bounded by two sections of the ellipse $E_i=W(\alpha^*)\cap U_i$, by one section of D_i , and by one section of D_{i+1}. **Case 3:** K_i is the ellipse $E_i=W(\alpha^*)\cap U_i$. In the other two cases (not shown), K_i is bounded by one elliptic and one circular arc, combining the situations of case 2 and case 3 on each end.

Given disks $D_1 \ldots D_k$, and 2D-caplets $K_1 \ldots K_k$, we want to find an arc, A, that satisfies the order and the tolerance requirements imposed by our metric (Definition 1). To satisfy the order requirement, there must exist a sequence of non-decreasing parameters $t_1,t_2 \ldots t_k$ such that $A(t_i)\in D_i$ for $i=1 \ldots k$. To satisfy the tolerance requirement, $A(t)$ must stay in K_i, for $t\in[t_i, t_{i+1}]$.

6 SEARCHING FOR AN ACCEPTABLE ARC IN THE PLANE

In this section, we present our algorithm for finding an acceptable arc in $W(\alpha^*)$. The arc A must interpolate the two end-vertices, S_1 and S_k, and is thus defined by a single parameter, h, which may for instance be chosen to represent the angle between the vector S_k–S_1 and the tangent to A at S_1 (see Fig. 6a). An acceptable arc $A(h)$ must satisfy a series of conditions, each constraining the interval H of acceptable values for h.

Our approach may be broken down into the following three phases (discussed below):

1. Check that we stab all disks
2. Check that we respect the stabbing order
3. Check that we stay in the tolerance zone between the disks

6.a Finding the planar pencil of arcs that stab all balls

The interval $[h_m, h_M]$ of values of angle parameter h for which $A(h)$ stabs a single disk D may be computed from the center D^1 and radius r of D by finding centers of arcs A_m and A_M that pass through S_1 and S_k and are tangent to D on each side (Fig. 6b). The details are provided in Appendix B. Given A_m and A_M, we obtain two angle parameters h_m and h_M.

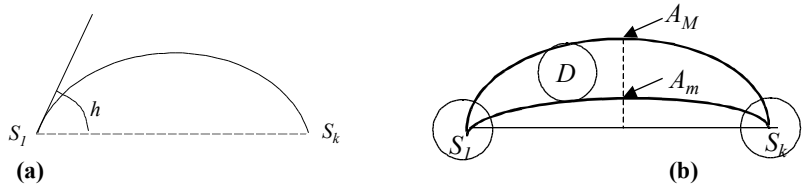


Fig 6: *All arcs passing through points S_1 and S_k are defined by a single parameter, h (a), which may for instance be chosen to represent an angle between a line passing through S_1 and tangent to the arc with the line segment joining S_1 and S_k. Fig. (b) shows the extremes of a family of arcs starting at vertex S_1 and ending at vertex S_k that stub disk D.*

To guarantee that all acceptable arcs stab all disks, we define H as intersection of all such individual intervals $[h_m,h_M]$. We compute H progressively, starting with an infinite interval and replacing it by its intersection with each individual interval.

6.b Enforcing the stabbing order

An arc that stabs all the disks, does not necessarily stab them in an order compatible with their indices.

When the intersections of the arc with the disks are **disjoint**, the order is clearly defined. All acceptable arcs either stab them in the right order or not. Consequently, compliance with the stabbing order requirement may be easily checked, in this case, by considering a single stabbing arc. Thus, when all disks are disjoint, we select A^*, defined as $A(\min(H))$, compute the arc-length parameters s_i of the closest point along A^* to each disk S_i, and check that they respect the order ($i>j \quad s_i>s_j$).

If some disks **overlap**, we consider a single pair (D_i,D_j) at a time. If D_i and D_j are disjoint, we use the above procedure to decide whether the pair is correctly stabbed by all $A(h)$ with $h\in H$, or whether it is stubbed in the reverse order and thus reduces H to the empty set. If D_i and D_j are **not disjoint**, they define an interval H_{ij}, such that: $h\in H_{ij} \Leftrightarrow A(h)$ stabs D_i and D_j in the correct order (see Fig. 7). We compute H_{ij} and replace H by $H\cap H_{ij}$, as detailed in Subsection 6c.

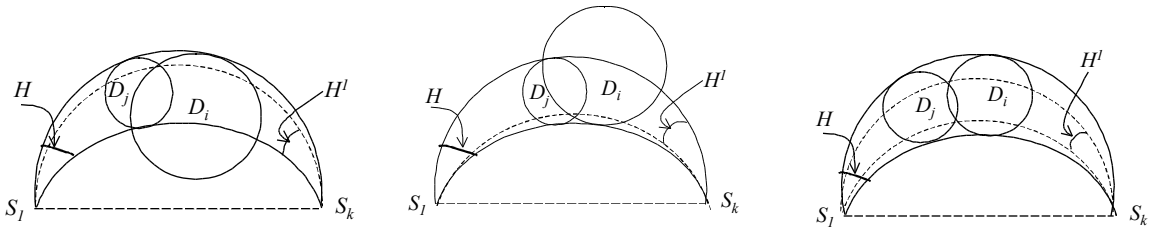


Fig 7: *Two intersecting disks D_i and D_j reduce the interval H of acceptable arcs from above (a), from bellow (b) and both (c). H^1 is the reduced interval.*

If we were to test all pairs of disks, the above procedure would have $O(n^2)$ complexity. We have chosen to reduce this time-complexity to a **linear complexity** in exchange for a slight possibility of not finding a solution when one exists. We have compared the n^2 and our linear solutions and have so far found **no real case where** the **linear approach misses** a stabbing arc.

Our linear solution considers only pairs (D_i,D_j), when $j–i$ is less than a given constant z. If it returns an empty interval, we correctly conclude that no acceptable arc exists. Otherwise, we pick the shortest arc, A^*, defined as $A(\min(H)$.

A^* may not stab all pairs in order. For example, in Fig. 8a, if $z=2$, we only test the pairs (D_1, D_2) and (D_2,D_3). The resulting interval, H, is bounded by the two dashed arcs. Not all arcs in H intersect disks D_1 and D_3 in order. In particular, the shortest one, A^*, does not. Thus, we must if for compliance with the stabbing order against all disks, as explained in Subsection 6d.

If A^*t fails the stabbing-order test we give up, concluding that no solution exists. Although this conservative conclusion may theoretically be wrong, we found that, for a small constant z, it proved correct for all the real cases we have tested. If A^* passes the stabbing test, we know that no other arc with a smaller h passes the stabbing test. Thus, if we discover that A^* is not contained in the tolerance zone, we correctly conclude that there is no solution to our 2D problem. If A^* passes both the order-compliance test and the tolerance zone test, we have found an acceptable arc.

(a) **(b)**

Fig 8: *(a) If k = 2, our reduced test will only test pairs (D_1,D_2) and (D_2,D_3). The resulting interval, H, is shown by two arcs. Not all arcs in interval H will intersect disks D_1 and D_3 in order. (b) Checking the order requirement for an arc A**

6.c Constraining H to satisfy the stabbing order for a pair of intersecting discs

Consider a pair (D_i,D_j) of intersecting disks. Assume without loss of generality that $j>i$. Their bounding circles intersect at two points (which may be coincident for singular configurations where the circles are tangent to each other). We compute the parameters h for the arcs passing through these intersection points and use them to split H into sub-intervals. We test each sub-interval for compliance with the stabbing order requirement by generating a bisector arc (for the average h in the sub-interval). If the intersections of the bisector arc with the two disks overlap or are order-compliant, we declare this sub-interval compliant with the stabbing order for D_i and D_j. Otherwise, we declare this sub-interval not compliant. We return the union of the sub-intervals for which the stabbing order is correct.

6.d Testing a candidate arc for correct stabbing order

Let I_i be the interval of intersection of disk D_i with arc A^*. Let b_i and e_i represent the arc-length parameters along A^* for the starting and the ending points of I_i (Fig. 8b). An order requirement for arc A^* is satisfied if, for each i in $[1...k]$, $e_i \le \max_{t<i} b_t$. Our extended notion of order is consistent with the one proposed in [21] for stabbing lines. It states that the segment (line or arc) intersects the disks in order if there exists a sequence of points P_j on the segment, each associated with disk D_j, such that each point P_j lies inside D_j and is not further away along the segment than all P_i for $i>j$. As stated by Guibas and colleagues [21] this is equivalent to saying that "no later interval ends before all earlier one begins".

6.e Testing the candidate arc for containment in the tolerance zones between the disks

To satisfy the tolerance requirement, we must check that for some sequence of increasing values t_i, for each interval $[t_i, t_{i+1}]$, the arc A^* stays inside the 2D-caplet K_i. It was pointed out in Section 5 that disks D_i and D_{i+1} are in K_i. On intervals $[t_i, e_i]$ and $[b_{i+1}, t_{i+1}]$, A^* lies inside disks D_i and D_{i+1} respectively, and thus inside the 2D-caplet K_i (see Fig. 9). Thus, we only need to check that arc A^* stays inside the 2D-caplet K_i on the interval $[e_i, b_{i+1}]$, for all $i=1...k$.

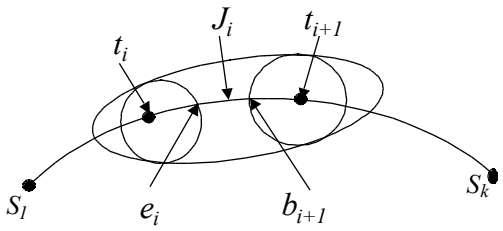


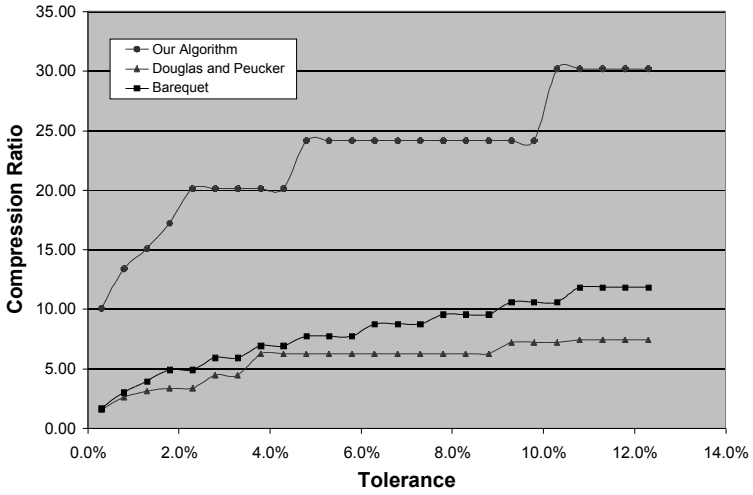
Fig 9: *Checking tolerance requirement for arc A*.*

Let J_i be the part of arc A^* between points e_i and b_{i+1} (Fig. 9). The tolerance requirement for J_i is satisfied if J_i stays within the corresponding ellipse E_i (or rectangle). We first compute the arc length parameters of the intersection between A^* and E_i. We do this by substituting the parametric form of the circle into the implicit equation of the ellipse and finding the roots of the

resulting fourth degree polynomial. Alternatively, one could use the implicit equation of the cylinder and solve the problem in 3D (see [31] for the details on this approach). Then, we test whether these intersection parameters line inside $[e_i, b_{i+1}]$. If so, J_i intersects the ellipse and A^* is invalid. **If A^* intersect the ellipse, then all arcs in the interval H intersect the ellipse**, and thus no acceptable arc exist. For a proof, remember that the two endpoints of J_i lie inside the ellipse and so does the straight edge, E, joining them (because the ellipse is convex). The region bounded by E and any acceptable arc A' in H, larger than A^*, will contain A^*, and thus will intersect with the region interior to the ellipse and with its complement. The boundary of that region will therefore intersect the ellipse. Since the intersection does not happen along E, it must occur along A'.

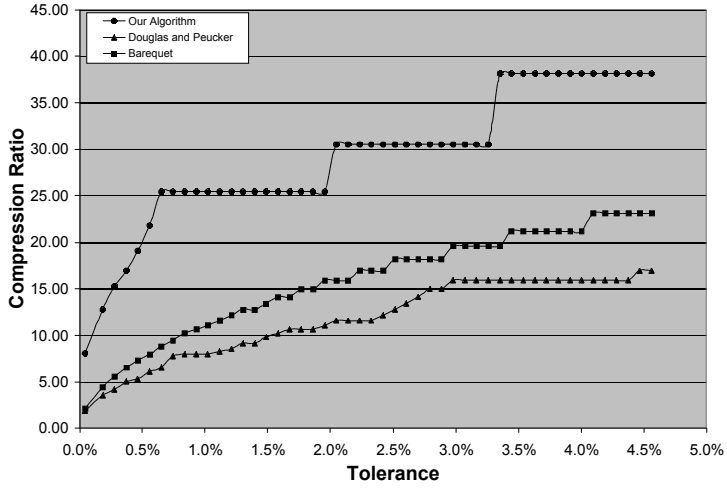
7 RESULTS

In this section we present experimental results of running our DPCA algorithm on a series of sample curves. The choice of those curves was based on a desire to cover different areas of application. We have used helices because they are simple 3D curves that are easy to reproduce. We have used cone-cone intersection curves since they illustrate curves found in CAD/CAM applications. We have also used rugged boundaries of triangle meshes, as an example of piecewise polygonal curves that are used in graphics and may correspond to patch boundaries.



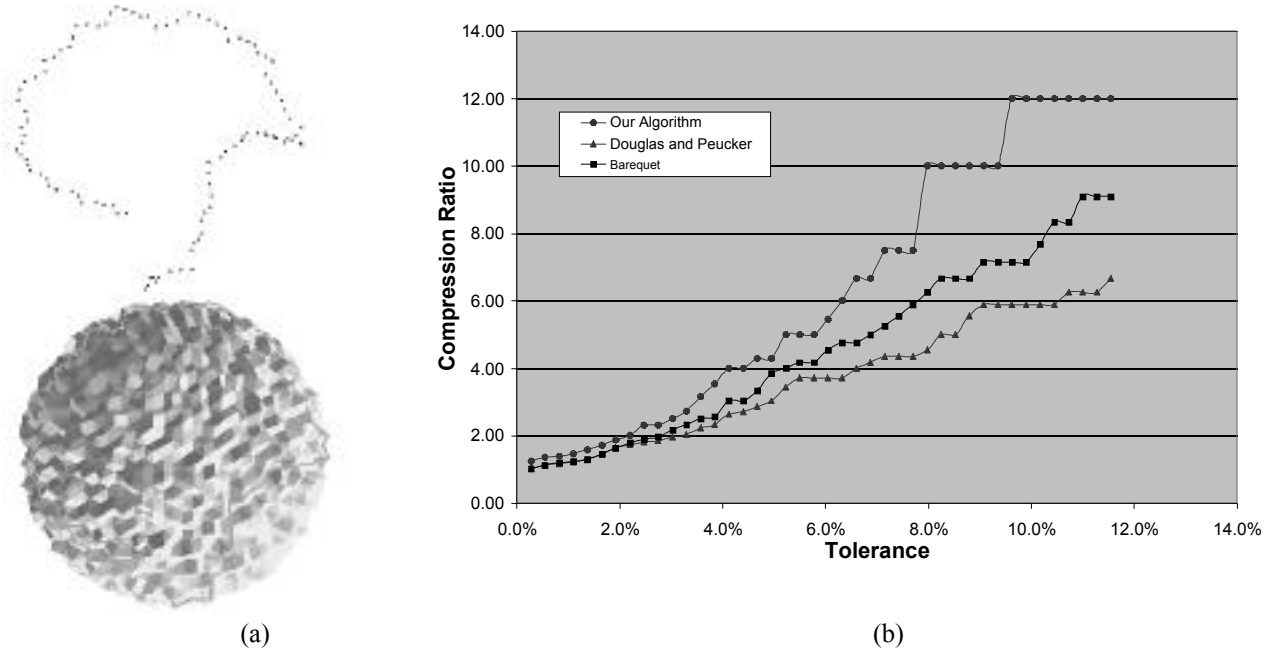
(a) (b)

Fig 10 : *Given the polyline P (**helix**) (a), the graph (b) compares compression ratio of our algorithm with the polygonal algorithms of [7] and [39]. We have plotted the comparison for curves corresponding to 25 different values of the error tolerance.*



(a) (b)

Fig 11: *Given the polyline P that approximates a **cone-cone intersection** (a), the graph (b) compares compression ratio of our algorithm with polygonal algorithms of [7] and [39] for 25 different cases of defined by different values of tolerance.*



(a) (b)

Fig 12: *Given the polyline P that bounds a region of a rugged polygonal surface (a), the graph (b) compares compression ratio of our algorithm with polygonal algorithms of [7] and [39] at different tolerance levels.*

We have tested our optimal PCA (OPCA) and doubling PCA (DPCA) algorithms on a series of sample curves and compared their compression ratio with the compression ratios achieved with the algorithm of Douglas and Peucker's [7], which is commonly used in GIS, and with the optimal algorithm of Barequet et al [39]. Both of these algorithms fit polylines rather than PCAs. We have also compare the performance and running times of OPCA and DPCA algorithms.

7.a Comparisons with polygonal fits

Fig. 10 and 11 compare the performance of polygonal algorithms to our algorithm on typical helix and cone-cone intersection curves, respectively. The helix curve was created with an error $\varepsilon_0 = 0.3\%$ of the radius of the helix. The cone-cone intersection curve was created with an error $\varepsilon_0 = 0.01\%$ of the radius of a nearly minimal bounding sphere. Fig. 12 compares the performance of polygonal algorithms to our algorithm on a rugged boundary of an artificially complex polygonal surface.

Since the representation of the polyline requires 3 parameters per edge, and the representation of the PCA requires 5 parameters per arc we **report the ratio between the number of scalar values** in the original polyline P and the number of scalar values needed to store the approximating PCA or polyline.

Our algorithm consistently outperforms both polygonal algorithms of [7, 39] for any error tolerance. The **compression improvements range from 20% to 300%.** For noisy data, the degree to which our approach outperforms polygonal algorithms increases as the tolerance is relaxed (see Fig. 12b). This behavior can be explained by the fact that the larger error tolerance allows the PCA curve to approximate more of the general shape of the curve rather than its noise.

7.b Comparing our doubling (DPCA) and optimal (OPCA) algorithms

E	#Points in P	OPCA algorithm	DPCA algorithm	Ratio OPCA/DPCA
0.1%	254	14	14	1
0.3%	254	9	9	1
0.5%	254	8	8	1
0.7%	254	6	6	1
0.9%	254	6	6	1
1.1%	254	6	6	1
1.3%	254	6	6	1
1.5%	254	6	6	1
1.7%	254	6	6	1
1.9%	254	6	6	1

(a)

E	#Points in P	OPCA algorithm	DPCA algorithm	Ratio OPCA/DPCA
0.1%	201	17	17	1
0.3%	201	12	12	1
0.5%	201	10	10	1
0.7%	201	9	9	1
0.9%	201	9	9	1
1.1%	201	8	8	1
1.3%	201	8	8	1
1.5%	201	7	7	1
1.7%	201	7	7	1
1.9%	201	7	7	1

(b)

Table 1: *We compare the number of arcs produces by our OPCA algorithm to the number of arcs produces by our DPCA algorithm for the intersection of two cones (a).and for a helix (b).*

Even though our DPCA algorithm does not produce an optimal result, the experimental data that we have collected shows that, on smooth curves, the DPCA algorithm perform as well as the OPCA algorithm. See tables 1(a) and 1(b). Our experimental data on curves with noise shows that the DPCA algorithm produces optimal solutions in most cases (Table 2a). Table 2b shows running times for both algorithms for the same helix curve as in Table 2.

E	#Points in P	OPCA algorithm	DPCA algorithm	Ratio OPCA/DPCA
0.9%	100	43	43	1.00
1.3%	100	38	38	1.00
1.7%	100	35	35	1.00
2.1%	100	31	31	1.00
2.5%	100	26	26	1.00
2.9%	100	26	26	1.00
3.3%	100	22	22	1.00
3.7%	100	20	21	0.95
4.1%	100	15	[16]	0.93
4.5%	100	15	15	1.00
4.9%	100	14	14	1.00

(a)

E	#Points in P	OPCA Time	DPCA Time
0.1%	201	3.7	0.1
0.3%	201	6.4	0.1
0.5%	201	8.3	0.1
0.7%	201	9.6	0.1
0.9%	201	11.4	0.1
1.1%	201	12.8	0.1
1.3%	201	13.5	0.1
1.5%	201	14.8	0.1
1.7%	201	15.8	0.1
1.9%	201	16.4	0.1

(b)

Table 2: *The number of arcs produced by OPCA is compared (a) to the number produced by DPCA for curves with noise. The execution time of the two algorithms for a cone-cone intersection curve is given (b) in seconds on a 700MHz PC.*

In conclusion, our OPCA algorithm is between 30 and 150 times slower than the DPCA algorithm and has no benefit except for noisy curves with a relatively high error tolerance compared to the noise magnitude.

8 COMPRESSING PCA REPRESENTATIONS

8.a Representation for arcs

A single arc, *A*, can be represented by its two endpoints, *S* and *E,* and a vector ***V*** between the midpoint *P = (S + E)/2* and the midpoint *M* of the arc (Fig. 13). By construction, ***V*** is orthogonal to ***SE*** and can be represented with two parameters, provided that we establish a convention that always defines a suitable coordinate system (i.e. a reference vector ***W*** around *SE*).

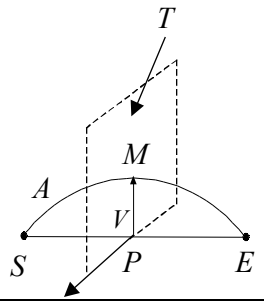


Fig 13: *Representation of a single arc A.*

To encode a PCA with *m* arcs, we need to store (*5m+3*) values (3 values for a first endpoint of the first arc and 5 values for each arc: 3 represent the ***SE*** vector and 2 represent the ***V*** *vector*). We chose to represent ***V*** by its absolute length and an angle it forms with the reference vector ***W*** in the plane *T*. To find the reference vector ***W*** for each arc we project vectors $\{(1,0,0)^T$, $(0,1,0)^T$, $(0,0,1)^T\}$ onto the plane *T* orthogonal to ***SE***, and choose the longest projection as a reference vector (ties are broken using a simple convention).

8.b PCA quantization

Our PCA approximates the original polyline within some error ε_1. Since we are focusing on compression, we use a quantization technique that reduces the storage of the PCA without exceeding an additional tolerance ε_2. To guarantee that the compressed PCA remains within error ε from polyline *P*, we first compute a PCA, *C*, with a tolerance ε_1 and then derive a quantization representation *Q* of *C*, with a quantization error bounded by ε_2, such that $\varepsilon_1+\varepsilon_2 \le \varepsilon$. To compute the **minimal storage** representation of *Q*, we consider several choices (candidate solutions) and select the smallest in size. Each candidate solution is defined by an error ε_1 in the interval $[0..\varepsilon]$. This choice imposes a quantization error $\varepsilon_2 = \varepsilon - \varepsilon_1$. We then use ε_1 to run our PCA construction algorithm (as explained above) and ε_2 to select the minimal number of bits for storing the representation *Q* of *C*, as explained in the next section, while guaranteeing that the error between *C* and *Q* does not exceed ε_2.

8.c Selecting the number of bits for PCA representation

Quantizing the five scalars that represent an arc will result in quantization errors that are bounded by the following formulae:

$$E_P = \frac{R_P}{2^{B_P}-1},\ E_V = \frac{R_V}{2^{B_V}-1} \text{ and } E_A = \frac{2\pi}{2^{B_A}-1}. \qquad (1)$$

Where, E_P, E_V, E_A denote, respectively, the error due to the quantization of vertex coordinates, the absolute length of vector $\boldsymbol{V}$, and the angle between vector $\boldsymbol{V}$ and the reference vector $\boldsymbol{W}$. B_P, B_V, B_A denote the number of bits used for these entities, and R_P, R_V, 2π denote their ranges of their values.

For an arc A with an absolute length of vector $\boldsymbol{V}$ less than the length of vector $\boldsymbol{SE}$ (Fig. 13) the maximum error ε_{max} between A and its quantized representation is bounded by the following formula (see Appendix B):

$$\varepsilon_{\max} \le 2|V|\sin(E_A/2) + 7\sqrt{3}E_P + E_V \qquad (2)$$

If the absolute length of $\boldsymbol{V}$ is greater than the length of $\boldsymbol{SE}$, instead of encoding arc A, we encode arc A', which complements A to a full circle. With each arc, we store one bit specifying whether A or A' was encoded.

In order to uniformly control the quantization error, we choose B_P and B_V to be the same for all arcs and B_A to be dependent on the radius of the arc. As shown in the equations (2), the larger the radius is, the larger B_A must be for that arc.

Given error ε_2, we want to select B_P, B_V and B^i_A, $i=1..m$, that minimize the storage requirement for C, while guaranteeing that the error between C and Q will not exceed ε_2. The storage size of Q with m arcs is defined by the following formula:

$$mB_V + 3mB_p + \sum_{i=1}^{m} B^i_A \qquad (3)$$

where B_P, B_V and B^i_A, $i=1..m$, are the $m+2$ unknowns.

To guarantee that the error between C and Q will not exceed ε_2 we require for each arc that the maximum error ε_{max} defined by equation (2) be less than ε_2. Substituting (1) into (2), we get m inequalities, one for each arc:

$$\varepsilon_{\max} \le 2|V|\sin\frac{2\pi}{2*(2^{B_A}-1)} + 7\sqrt{3}\,\frac{R_P}{2^{B_P}-1} + \frac{R_V}{2^{B_V}-1} \le \varepsilon_2 \qquad (4)$$

Thus, to find B_P, B_V and B^i_A, $i=1..m$, we need to solve a non-linear optimization problem with $m+2$ unknowns and m constrains. We solve it by selecting the number of bits for B_P and B_V on an interval [5..16] and then solving for B^i_A, $i=1..m$ from (4). We choose a solution that minimizes the size of Q defined by (3).

The complexity of the Doubling PCA algorithm without quantization was $O(nlogk)$, where n is the number of vertices in P and k is the maximum number of vertices of P approximated by a single arc. With the quantization procedure defined in this section, the complexity is limited by $C_1*(nlogk + C_2*m)$, where C_1 and C_2 are constants: C_1 is the number of values we try for ε_1 on an interval [0.. ε], $C_2 = 144$ is the number of values we try for B_P times the number of values for B_V and m is the number of arcs in the PCA.

8.d Compression results

The number of bits per vertex in the original polyline as a function of allowed error tolerance is given in Fig. 14a, 14b, and 14c, each for a different curve. For example, the compressed representation of a typical cone-cone intersection (Fig. 14b), requires about $7n$ bits for a total error $\varepsilon = 0.02\%$ of the radius of the minimal sphere around the curve and less than $1n$ bits for $\varepsilon = 3\%$, where n is the number of vertices in a polyline which approximates a cone-cone intersection curve with error $\varepsilon_0 = 0.01\%$.

9 CONCLUSION

We have shown that piecewise circular curves have advantage over polygonal and B-spline curves, when used to produce compact approximations of 3D curves. We have presented the details of an efficient approach for computing such piecewise circular approximations and for compressing them. For example let P be a polyline with 250 vertices that approximates the intersection curve between two quadric surfaces within a tolerance ε_0 of 0.01% of the size of the curve. Then in about two seconds, our algorithm computes a piecewise circular curve that approximates P with a tolerance ε of 0.02% and can be stored using 7x250 bits: a 5-to-1compression ratio. The compression ratio improves as the original tolerance ε_0 is relaxed.

10 ACKNOWLEGEMENTS

Research on this project was supported in part by U.S. National Science Foundation grant 9721358 and by a GTE student fellowship.

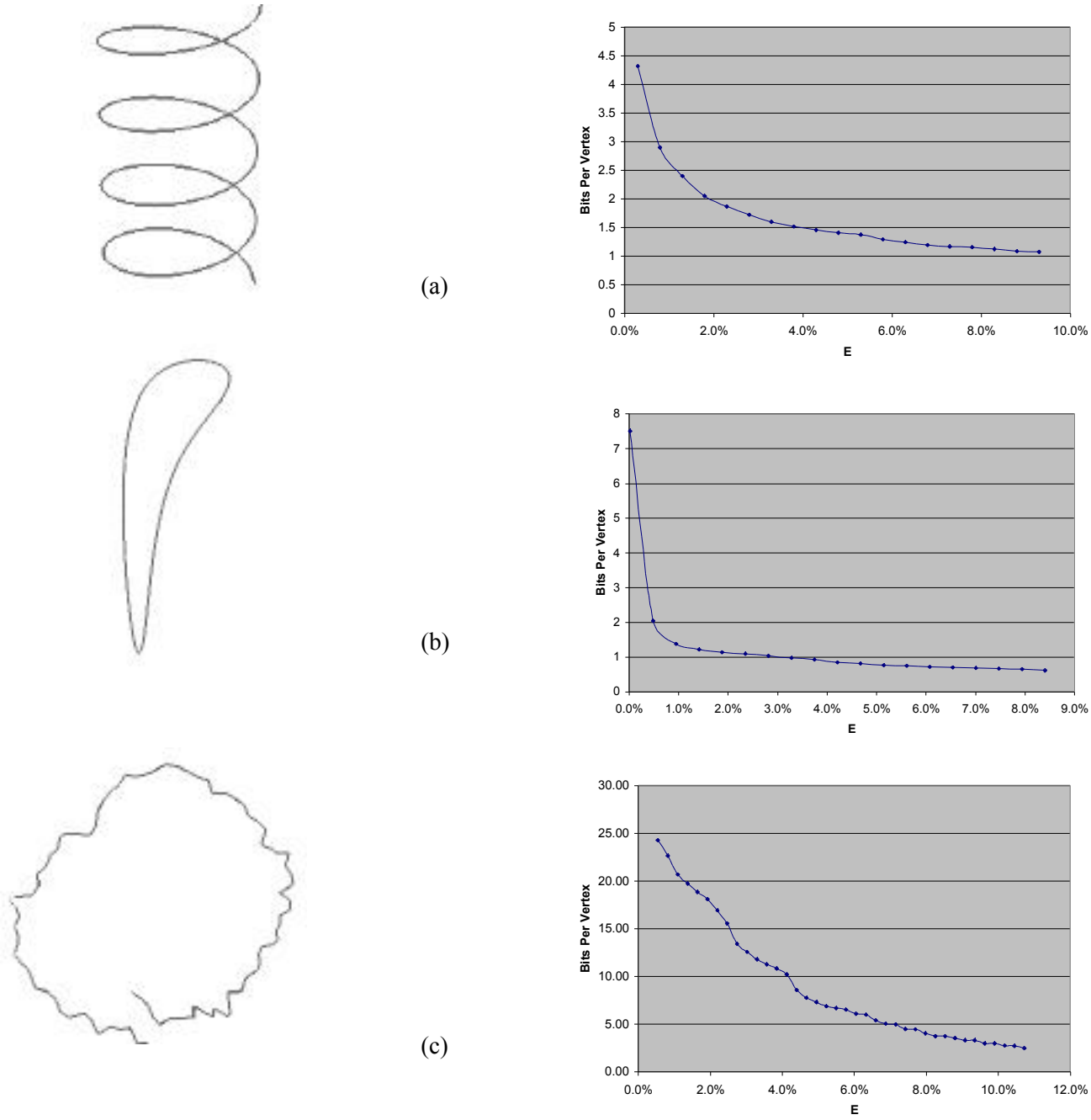


Fig 14: *Number of bits per vertex of original polyline as a function of total error* ε. *(a) Helix, created with* ε_0 *= 0.3% of the radius of the helix curve (b) cone-cone intersection, created with* ε_0 *= 0.01% of the radius of nearly minimal bounding sphere (c) Rugged boundary.*

11 REFERENCES

1. G. Farin, Curves and Surfaces for Computer Aided Geometric Design, A practical guide, Second edition.
2. P.J. Laurent, A.L. Mehaute, L.L. Schumaker, Curves and Surfaces, New York: Academic Press 1991.
3. Paul Heckbert and Michael Garland, Survey of Polygonal Surface Simplification Algorithms, SIGGRAPH 97 course on Multiresolution Surface Modeling.
4. R.B. McMaster, The integration of simplification and smoothing algorithms in line generalization, Cartographica 1989;26(1):101-121.
5. K. Beard, Theory of the cartographic line revisited, Cartographica 1991;28(4):32-58.
6. R.G. Cromley, Campbell GM, Integrating quantitative and qualitative aspects of digital line simplification, The Cartographic Journal, 1992;29(1):25-30.
7. D.H. Douglas and T.K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Canadian Cartographer, 10(2):112-122, 1973.
8. Dana H. Ballard, Strip Trees: A hierarchical representation for curves, Communications of the ACM, 24(5):310-321, 1981.

9. R. B. McMaster, The geometric properties of numerical generalization, Geographical Analysis, 19(4):330-346, Oct. 1987.
10. Laurence Boxer, Chun-Shi Chang, Russ Miller, and Andrew Rau-Chaplin, Polygonal approximation by boundary reduction, Pattern Recognition Letters, 14(2):111-119, February 1993.
11. J.G. Leu and L.Chen, Polygonal approximation of 2-D shapes through boundary merging, Pattern Recognition Letters, 7(4):231-238, April 1988.
12. H. Imai and M. Iri, Computational-geometric methods for polygonal approximations of a curve, Computer vision, Graphics, and Image Processing, 36:31-41, 1986.
13. H. Imai and M. Iri, Polygonal approximations of a curve-formulations and algorithms, Computational Morphology, 71-86, North-Holland, Amsterdam, 1988.
14. A. Melkman and J. O'Rourke, On polygonal chain approximation, Computational Morphology, 87-95, North-Holland, Amsterdam, 1988.
15. W.S. Chan and F. Chin, Approximation of polygonal curves with minimal number of line segments or minimum error, Int. J. of Computational Geometry and Applications, 6(1):59-77, 1996.
16. D.Z. Chen and O. Daescu, Space-efficient algorithms for approximating polygonal curves in two-dimensional space, Manuscript, 1997.
17. K.R. Varadarajan, Approximating monotone polygonal curves using the uniform metric, Proc. 12th Ann. Acm Symp. On Computational Geometry, Philadelphia, PA, 106-112, 1996.
18. I. Ihm and B. Naylor, Piecewise linear approximations of digitized space curves with applications, in N.M. Patrikalakis (ed.), Scientific Visualization of Physical Phenomina, Springer-Verlag, Tokyo,545-569,1991.
19. D.Eu, G.T. Toussaint, On approximation of polygonal curves in two and three dimensions, CVGIP: Graphical Models and Image Processing, 56(3):231-246, 1994.
20. K.R. Varadarajan, Approximating monotone polygonal curves using the uniform metric, Proc. 12th Ann. ACM Symp. On Computational Geometry, Philadelphia, PA, 106-112, 1996.
21. L.J. Guibas, J.E. Hershberger, J.S.B. Mitchell, and J.S. Snoeyink, Approximating polygons and subdivisions with minimal link paths, Int. J. of Computational Geometry and Applications, 3(4):383-415, 1993.
22. E. Saux, M.Daniel, Data reduction of polygonal curves using B-splines, Computer Aided Design, vol. 31, no. 8, p. 507-515, 1999.
23. S.C. Pei and J.H. Horns, Optimum approximation of digital planar curves using circular arcs, Pattern Recognition, vol. 29, no. 3, p. 383-388, 1996.
24. S.C. Pei and J.H. Horns, Fitting digital curves using circular arcs, Pattern Recognition, vol. 28, no.1, p.107-116, 1995.
25. S.N. Yang, W.C. Du, Numerical methods for approximating digitized curves by piecewise circular arcs, Journal of Computational and Applied Mathematics, vol. 66, p. 557-569, 1996.
26. S.N. Yang, W.C. Du, Piecewise Arcs Approximation for Digitized Curves, Proceedings of the Computer Graphics International 1994 (CG194). Insight Through Computer Graphics, p. 291-302, 1996.
27. P.L.Rosin, G.A.W. West, Segmentation of edges into lines and arcs, Image and Vision Computing, vol. 7, no. 2, p. 109-114, 1989.
28. G.A.W. West, P.L.Rosin, Techniques for segmenting image curves into meaningful description, Pattern Recognition, vol. 24, no. 7, p. 643-652, 1991.
29. J.R.Rossignac, A.A.G. Requicha, Piecewise-circular curves for geometric modeling, IBM Journal of Research and Development, vol. 31, no. 3, p. 296-313, 1987.
30. Y.J. Tseng, Y.D.Chen, Three dimensional biarc approximation of freeform surfaces for machining tool path generation, International Journal of Production Research, vol. 38, no. 4, p. 739-763, 2000.
31. J. Rossignac, "Blending and Offsetting Solid Models", PhD Thesis, University of Rochester, NY, June 1985, Advisor: Dr. Aristides Requicha.
32. D.S. Meek and D.J. Walton, Approximation of discrete data by G^1 arc splines, Computer Aided Design vol. 24, no. 6, p. 301-306, 1992.
33. J.Hoschek, Circular splines, Computer Aided Design, vol. 24, no. 11, p. 611-618, 1992.
34. D.J. Walton, D.S. Meek, Approximation of quadratic Bezier curves by arc splines, Journal of Computational and Applied Mathematics, vol. 54, no.1, p. 107-120, 1994.
35. D.S. Meek, D.J. Walton, Approximating quadratic NURBS curves by arc splines, Computer Aided Design, vol. 25, no.6, p. 371-376, 1993.
36. M. Eck, J. Hadenfeld, Knot removal for B-spline curves, Fachbereich Mathematik, Technische Hochschule Darmstadt, D-64289 Darmstadt, FRG, 1994.
37. H. Alt, M. Godau, Measuring the resemblance of polygonal curves, Proceedings of the Eighth Annual Symposium on Computational Geometry, p. 102-109, 1992.
38. M. Godau, Die Fréchet-Metric für Polygonzüge - Algorithmen zur Abstandsmessung und Approximation, Diplomarbeit, Fachbereich Mathematik, FU Berlin, 1991.
39. G. Barequet, D.Z.Chen, O. Daescu, M.T. Goodrich, J. Snoeyink, Efficiently Approximating Polygonal Paths in Three and Higher Dimensions, Proceedings of the Fourteenth Annual Symposium on Computational Geometry, p. 317-326, 1998.
40. G. Taubin & J. Rossignac, Geometric Compression through Topological Surgery, ACM Transactions on Graphics, vol. 17, no. 2, pp. 84-115, April 1998.
41. J. Rossignac & A. Requicha, Offsetting Operations in Solid Modeling, vol. 3, p. 129-148, 1986.

12 APPENDIX A: Proof of equivalence with the Fréchet Metric

We prove here that our error measure is equivalent to the Fréchet Metric [37, 38] for the case of one arc and a polyline. Please refer to section 4.7 for the definitions of our error estimate and of the Fréchet error metrics. The notations δ_A and δ_F refer respectively to our error metrics and to the Fréchet error metrics.

12.a First we prove that $\delta_F(A, S) \le \varepsilon$ implies $\delta_A(A, S) \le \varepsilon$

By definition, $\delta_F(A, S) \le \varepsilon$ implies that there exists two monotone parameterizations, α and β, of the arc A and of the polyline P, respectively, which are both functions from [0,1] onto itself, such that $d(\alpha(t),\beta(t)) \le \varepsilon, \quad t \in [0,1]$. We need to prove that both the order and the tolerance requirements imposed by our metric are satisfied. This direction of the proof is similar to the one given in [21] for polylines in 2D, and has two parts.

The order requirement: We need to prove that there exists a sequence of non-decreasing parameters $t_1 \dots t_k$ such that $A(t_i) \in B_i$, for $i=1 \dots k$. Consider parameters t_i, for $i=1 \dots k$, such that $\beta(t_i)$ are vertices of the polyline S. Since $d(\alpha(t),\beta(t)) \le \varepsilon \quad t \in [0,1]$, $\alpha(t_i) \in B_i \quad$ for $i=1 \dots k$.

The tolerance requirement: We need to prove that $A(t) \in T_i \quad$ for $t \in [t_i, t_{i+1}]$. Since parameterizations α and β are monotone, $d(\alpha(t),\beta(t)) \le \varepsilon$ for $t \in [t_i, t_{i+1}]$ and thus $A(t) \in T_i$ for $i=1..k-1$.

12.b Second we prove that if $\delta_A(A, S) \le \varepsilon$ then $\delta_F(A, S) \le \varepsilon$.

By definition $\delta_A(A, S) \le \varepsilon$ implies that there exists a sequence of non-decreasing parameters $t_1, t_2 \dots t_k$, such that $A(t_i) \in B_i \quad$ for $i=1 \dots k$ and $A(t) \in T_i \quad$ for $t \in [t_i, t_{i+1}]$. We need to prove that there exist monotone parameterizations α and β of arc A and polyline P respectively, such that $d(\alpha(t),\beta(t)) \le \varepsilon \quad t \in [0,1]$. We do this by providing such a monotone parameterization on each of the intervals $[t_i, t_{i+1}]$, namely, we provide monotone parameterizations α and β such that $d(\alpha(t),\beta(t)) \le \varepsilon \quad t \in [t_i, t_{i+1}]$ for $i=1 \dots k-1$. We choose $\beta(t_i)$ to be an i^{th} vertex of the polyline S, $\alpha(t_i)$ to be some point on arc A, is inside of ball B_i. We know that $\alpha(t_i)$ exists from the initial conditions (Fig 15). Thus, $d(\alpha(t_i),\beta(t_i)) \le \varepsilon$, for $i=1 \dots k$. Clearly, if there are such parameterizations for each of the intervals, then the monotone parameterization for interval $t \in [0,1]$ is just a concatenation of the parameterizations for each of the intervals (Fig. 15).

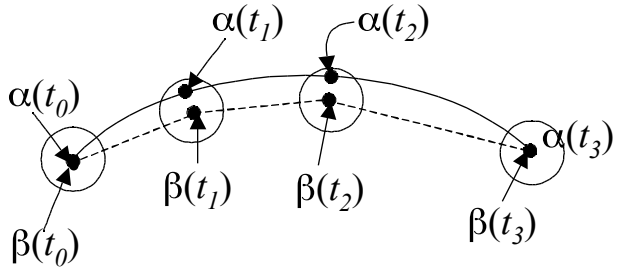


Fig 15: *We show that there exist monotone parameterizations α and β such that $d(\alpha(t),\beta(t)) \le \varepsilon \ \forall t \in [0,1]$ by showing that there exist monotone parameterizations on each of the intervals $[t_i, t_{i+1}]$. We choose $\beta(t_i)$ to be an i^{th} vertex of the polyline S, $\alpha(t_i)$ is some point on arc A, which is inside of ball B_i.*

First, we give a formal definition of the parameterization we propose. Then we intuitively explain what it means. Finally, we give the formal proof that this parameterization is correct.

We define a parameterization for a particular interval $[t_i, t_{i+1}]$ as follows: We split the interval $[t_i, t_{i+1}]$ into $N+M$ sub-intervals, where $N \to \infty$ and $M \to \infty$. We then give a definition for parameterizations α and β recursively (Fig. 16a):

- $\alpha(u_0) = A(t_i)$, $\alpha(u_N) = A(t_{i+1})$
- $\alpha(u_p)$ is monotonically increasing with uniformly distributed samples as p changes from 0 to N.
- $\alpha(u_{N+1}) = \alpha(u_{N+2}) = \dots = \alpha(u_{N+M}) = A(t_{i+1})$.
- $\beta(u_0) = S_i$.
- $\beta(u_p) = \max(\beta(u_{p-1}), \beta^*(u_p))$, for $p=1..N$, max is defined as a point farthest from S_i and β^* is defined below.
- $\beta(u_p)$ is monotonically increasing with uniformly distributed samples from $\beta(u_N)$ to $\beta(u_{N+M})=S_{i+1}$, for p in $[N, N+M]$.

$\beta^*(u_p)$ is the point on S_iS_{i+1} closest to S_i for which the distance between this point and $a(u_p)$ is less than or equal to ε (Fig. 16b). This point always exists since by initial conditions $A(t_i)$, for $i \in [t_i, t_{i+1}]$, belong to the convex hull of balls B_i and B_{i+1}.

To explain this parameterization intuitively, consider a boy walking along the arc segment $A(t)$, for $t \in [t_i, t_{i+1}]$, holding, on a leash of length ε, a dog that walks along the straight line segment S_iS_{i+1}. The boy starts walking at $A(t_i)$, while the dog is at S_i. Until the leash extends to its full length, ε, the dog remains at S_i. Afterwards, as the boy keeps on walking along the arc the dog lags behind as much as the leash permits, namely, it is always at distance ε from the boy. The dog never goes backward, even if the leash permits it, due to the maximum operation in the parameterization formula that we gave. As the boy reaches the end of the arc $(p=N)$, he stops and waits until the dog moves to the endpoint of the line segment, S_{i+1}. The same process is repeated for the next segment.

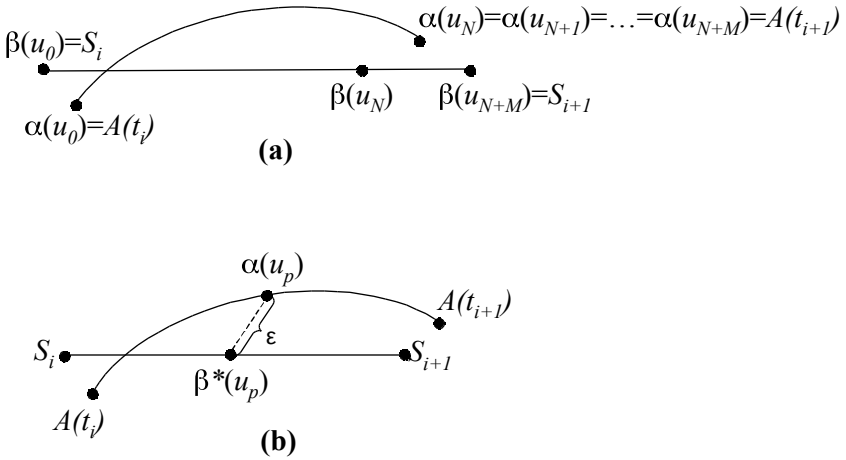


Fig 16: *Definition of the parameterization.*

Next we prove that such a parameterization is monotonically increasing, continuous, and $d(\alpha(u_p),\beta(u_p)) \le \varepsilon$ for $p=0\ldots N+M$.

- Both parameterizations are monotonic by construction.
- In order to show that α and β are continuous, we have to show that as $N\to\infty$ and $M\to\infty$, $d(\alpha(u_{p-1}), \alpha(u_p))\to 0$ and $d(\beta(u_{p-1}), \beta(u_p))\to 0$. These conditions are clearly true for the parameterization α and for the parameterization β for $p=N\ldots N+M$ from their definitions. For the parameterization β for $p=0\ldots N$, we have to show that $d(\beta(u_{p-1}),\beta(u_p)) \to 0$. Consider a particular pair u_{p-1}, u_p. If $\beta(u_p)$ is not equal to $\beta^*(u_p)$ then $\beta(u_p)=\beta(u_{p-1})$ and we have nothing to prove. Assume that $\beta(u_p)= \beta^*(u_p)$. Then we need to show that $d(\beta^*(u_p),\beta(u_{p-1})) \to 0$. At the same time $\beta(u_{p-1}) \ge \beta^*(u_{p-1})$ thus, if we were to show that $d(\beta^*(u_p),\beta^*(u_{p-1})) \to 0$ then it would imply that $d(\beta^*(u_p),\beta(u_{p-1})) \to 0$. But this is trivial to see that as $N\to\infty$, $d(\beta^*(u_p),\beta^*(u_{p-1})) \to 0$ because the arc and the line segment are both continuous and differentiable (For two points A and B along the arc or the polyline segments, the notation $A \ge B$ means that A is located further along the segment than B).
- Last we have to show that $d(\alpha(u_p),\beta(u_p)) \le \varepsilon$ for $p=0\ldots N+M$.

For $p=N\ldots M$, $d(\alpha(u_p),\beta(u_p)) \le \varepsilon$ since $\alpha(u_N) = \alpha(u_{N+1}) = \alpha(u_{N+M})$, $d(\alpha(u_{N+M}),\beta(u_{N+M})) \le \varepsilon$ from initial conditions and $d(\alpha(u_N),\beta(u_N)) \le \varepsilon$ as we are going to prove next (Fig 17).

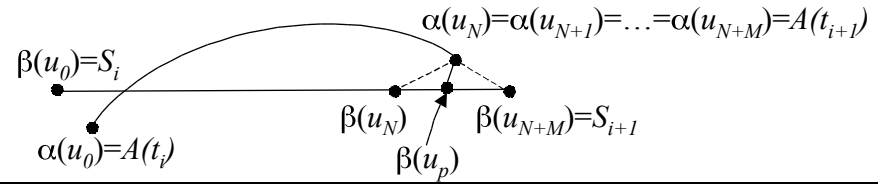


Fig 17: *$d(\alpha(u_p),\beta(u_p)) \le \varepsilon$ since $\alpha(u_N) = \alpha(u_{N+1}) = \alpha(u_{N+M})$, $d(\alpha(u_{N+M}),\beta(u_{N+M})) \le \varepsilon$ and $d(\alpha(u_N),\beta(u_N)) \le \varepsilon$.*

For $p=0\ldots N$, if $\beta(u_p) = \beta^*(u_p)$, then $d(\alpha(u_p),\beta(u_p)) \le \varepsilon$ by the definition of β^*. To prove that $d(\alpha(u_p),\beta(u_p)) \le \varepsilon$ if $\beta(u_p) \ne \beta^*(u_p)$ (and consecutively $\beta(u_p)= \beta(u_{p-1}) > \beta^*(u_p)$) we split the proof into two cases:

Case 1: The notation $\alpha^T(u_p)$ means the projection of $\alpha(u_p)$ onto line segment S_i, S_{i+1}. If the projection onto the line containing S_i, S_{i+1} lies to the left of S_i then $\alpha^T(u_p) = S_i$, if it is to the right of S_{i+1} then $\alpha^T(u_p) = S_{i+1}$. If $\alpha^T(u_p) \ge \beta(u_p)$, $d(\alpha(u_p),\beta(u_p)) \le \varepsilon$ since $d(\alpha(u_p), \alpha^T(u_p)) \le \varepsilon$ and $d(\alpha(u_p),\beta^*(u_p)) \le \varepsilon$ and $\beta^*(u_p)<\beta(u_p) \le \alpha^T(u_p)$ (Fig. 18).

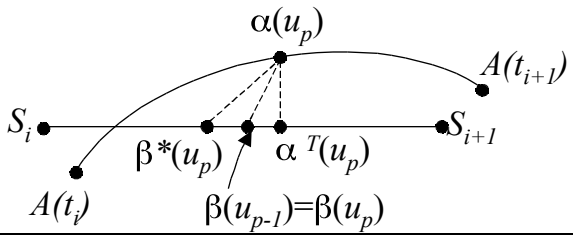


Fig 18: *If $\alpha^T(u_p) \ge \beta(u_p)$, $d(\alpha(u_p),\beta(u_p)) \le \varepsilon$ since $d(\alpha(u_p), \alpha^T(u_p)) \le \varepsilon$ and $d(\alpha(u_p),\beta^*(u_p)) \le \varepsilon$ and $\beta^*(u_p)<\beta(u_p) \le \alpha^T(u_p)$.*

Case 2: Now suppose, $\alpha^T(u_p)<\beta(u_p)$. We prove this case by contradiction. First, there exists $k<p$ such that $\beta(u_k)=\beta^*(u_k)=\beta(u_{p-1})$. Also, $d(\alpha(u_k), \beta^*(u_k))=\varepsilon$ since $\beta^*(u_k)=\beta(u_{p-1})>S_i$, because $\beta(u_{p-1})>\beta^*(u_p)\geq S_i$. Since $\beta^*(u_k)=\beta(u_{p-1})$, the distance $d(\alpha(u_k)\ \beta(u_{p-1}))$ is also equal to ε. Also, $\alpha^T(u_k)\geq\beta(u_{p-1})$ since $\beta^*(u_k)=\beta(u_{p-1})$ and β^* of any point $\alpha(u_i)$ is always less than or equal to $\alpha^T(u_i)$, because $d(\alpha^T(u_i),\alpha(u_i)\leq \varepsilon)$.

Consider the sphere C_{p-1} with center at $\beta(u_{p-1})$ and radius ε. $\alpha(u_k)$ lies on that sphere. Consider the plane L passing trough point $\beta(u_{p-1})$ perpendicular to the line segment S_iS_{i+1}. Since $\alpha^T(u_k)\geq\beta(u_{p-1})$, $\alpha(u_k)$ lies on or on the right side of plane L (Fig. 19).

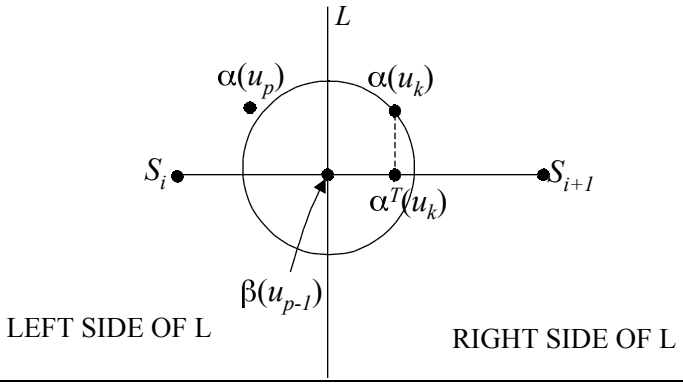


Fig 19: *Sphere C_{p-1} with center at $\beta(u_{p-1})$ and radius ε. $\alpha(u_k)$ lies on that sphere. Plane L passes trough point $\beta(u_{p-1})$ perpendicular to line segment S_i, S_{i+1}. Since $\alpha^T(u_k) \geq \beta(u_{p-1})$, $\alpha(u_k)$ lies on or on the right side of plane L Since $\alpha^T(u_p) < \beta(u_{p-1})$, $\alpha^T(u_p)$ lies on the left side of plane L*

Next we prove that $\alpha(u_p)$ cannot lie outside of sphere C_{p-1}, and therefore $d(\alpha(u_p),\beta(u_{p-1}))\leq\varepsilon$.

Since $\alpha^T(u_p)<\beta(u_{p-1})$, $\alpha^T(u_p)$ lies on the left side of plane L (Fig 19). We know that arc A first hits point $\alpha(u_k)$ and then hits point $\alpha(u_p)$, since $p> k$. We next show that point $\alpha(u_{k-1})$ lies outside of sphere C_{p-1}. If A is tangent to a sphere C_{p-1}, it is easy to see that $\alpha(u_{k-1})$ lies outside of circle C_{p-1} (this takes care of the case when $\alpha(u_k)$ lies on plane L). Suppose that A intersects C_{p-1}. If point $\alpha(u_{k-1})$ lies inside C_{p-1}, then A must exit sphere C_{p-1} , via the sequence of points: $\alpha(u_{k-1})$, $\alpha(u_k)$, $\alpha(u_{k+1})$, then enter it again to get to the other side of plane L, where $\alpha(u_p)$ lies (C_{p-1} is tangent to a cylinder in which A should stay to satisfy the tolerance requirement of our metric), and then exit C_{p-1} again to hit point $\alpha(u_p)$, which is impossible.

Thus, we proved that point $\alpha(u_{k-1})$ lies outside of sphere C_{p-1}. Also, it is easy to see that it lies to the right of plane L. Thus the distance between $\alpha(u_{k-1})$ and any point on line segment S_i, S_{i+1} smaller or equal to $\beta(u_{p-1})$ is larger than ε, which is impossible since point $\alpha(u_{k-1}) < \alpha(u_k)$ and therefore $\beta(u_{k-1}) \leq \beta(u_k) =\beta(u_{p-1})$.

13 Appendix B: Finding an arc passing through two points and tangent to a circle

In this section we show how to find two arcs A_m and A_M that pass through S_1 and S_k and are tangent to D (see Fig. 6b). D has a center D^1 and radius r.

To find arc A_m we solve the following system of equations (Fig. 20a):

$$\begin{cases} m^2 + t^2 = R^2 \\ x^2 + (y+t)^2 = (R+r)^2 \end{cases}$$

where,

- $\boldsymbol{x}$ is the absolute value of the DotProduct(D^1- M, $\boldsymbol{n}$), M is the midpoint of line segment passing through S_1 and S_k
- $\boldsymbol{n}$ is a unit vector in the direction of a line passing through S_1 and S_k,
- $\boldsymbol{y}$ is the absolute value of the DotProduct(D^1- M, $\boldsymbol{l}$),
- $\boldsymbol{l}$ is a unit vector in the direction perpendicular to $\boldsymbol{n}$,
- $\boldsymbol{m}$ is half of the length of line segment passing through S_1 and S_k,
- $\boldsymbol{t}$ and $\boldsymbol{R}$ are two unknowns, $\boldsymbol{t}$ is the distance from M to the center of A_m and $\boldsymbol{R}$ is the radius of A_m,

Given t, the center D^2 of A_m is equal to $M+l*t$.

We solve a similar system of equations to find arc A_M (Fig. 20b):

$$\begin{cases} m^2 + t^2 = R^2 \\ x^2 + (y+t)^2 = (R-r)^2 \end{cases}$$

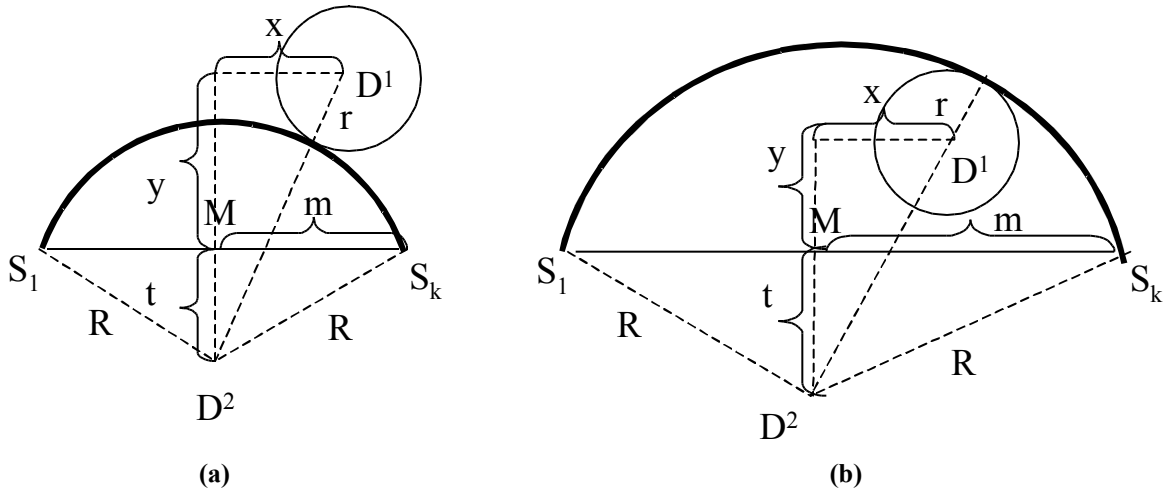


Fig 20: *Given points S_1 and S_k and a disk D with radius r and center D^1 we find an arc A_m passing through points S_1 and S_k and tangent to disk D from below (a) and arc A_M passing through points S_1 and S_k and tangent to disk D from above (b).*

14 Appendix C: The error bound for the quantized representation

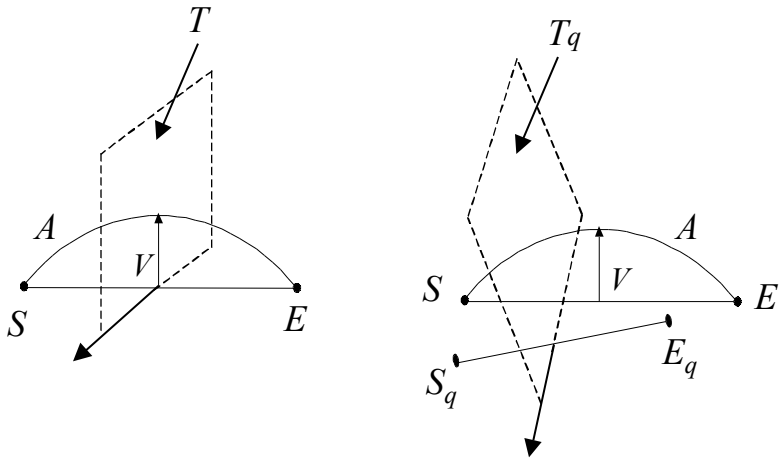


Fig 22: *(a) Representation of the arc, (b) Sq and Eq represent quantized vertices S and E respectively. Plane Tq is perpendicular to line segment Sq, Eq and passes through its center.*

In this section we describe our quantization procedure and prove that the maximum error ε_{max} between an arc A and its quantized representation A_q is bounded by the following formula: $\varepsilon_{\max} \le 2|V|\sin(E_A/2) + 7\sqrt{3}E_P + E_V$, where E_p, E_V and E_A denote, respectively, the error due to the quantization of vertices of PCA, the absolute length of vector ***V***, and the angle between vector ***V*** and the reference vector ***W***. We assume the of vector ***V*** is less than the length of vector ***SE*** (Fig 22a). See section 8 for the compete description of the arc representation.

The simple procedure of quantizing all parameters representing arc *A* will not produce good results. First, it is difficult to estimate an error between *A* and A_q, if such a quantization technique is used. Second, it may even result in an incorrect decompression for the reasons described next. Remember that an arc is represented by its two endpoints, *S* and *E,* and by a vector *V* in its bisecting plane (Fig 22a). Vector *V* is represented by its length and by the angle it forms with some reference vector ***W*** in the plane *T* which is perpendicular to ***SE***. To find the reference vector ***W***, we project vectors $\{(1,0,0)^T, (0,1,0)^T, (0,0,1)^T\}$ onto *T*, and choose the longest projection as a reference vector. Due to the quantization of endpoints of an arc *A*, during decompression, we will find a different reference vector ***W*** than the one used for compression since the plane *T* where we project vectors $\{(1,0,0)^T, (0,1,0)^T, (0,0,1)^T\}$ will be different. That introduces an error between *A* and A_q that is difficult to estimate. Also, since we use the longest of the projections as a reference vector ***W***, during decompression we may end up using a projection of a vector different than the one used for compression, which will result in an incorrect decompression.

An alternative would be to first quantize the endpoints of A and store a representation of V in a plane T_q that is perpendicular to the line segment connecting already quantized endpoints. But vector V does not necessarily lie in T_q (Fig 22b). Thus, we need to decide which vector V in plane T_q do we want to choose to represent our arc. There are different ways of choosing this vector. We propose one that allows us to easily compute the error between the original arc A and its quantized representation A_q.

To find vector V in plane T_q, we compute a transformation that maps S into S_q and E into E_q, where S_q and E_q represent quantized versions of S and E respectively. Let A^* be the image of A by this transformation. We perform this transformation in three steps and at each step estimate the maximum error we make. The resulting error between A and A^* is bounded by the summation of maximum errors made at each step. V is then a vector between the midpoint of line segment S_qE_q and the midpoint of the arc A^*. Next we compute a quantized representation of A, A_q, by quantizing the angle and radius of vector V. The resulting error between A and A_q is then bounded by the summation of error between A and A^* and between A^* and A_q.

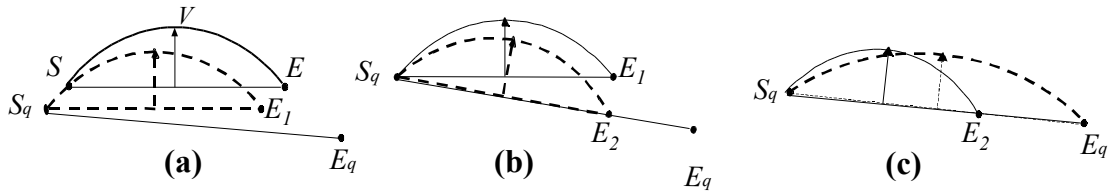


Fig 23: *Transformation of arc A into arc A*. Each figure shows a step of the transformation. Arc before a particular transformation step is drawn using solid line. An arc after each of the transformation steps drawn using dotted line.*

We first describe the process of transforming A into A^*.

1. First we transform A by a vector $(S_q - S)$. S is mapped into S_q, whereas E is mapped into some point E_1 (Fig 23a). The error between the original and transformed arcs is bounded by the length of vector $(S_q - S)$ and thus by a vertex quantization error $\sqrt{3}E_p$, where E_p is the quantization error of each coordinate of a vertex.

2. Then, we align the base of A with S_qE_q. In order to do that, we rotate A by an angle between vectors $(E_1–S_q)$ and $(E_q–S_q)$ around a vector L that is perpendicular to S_qE_q and to S_qE_1 (Fig 23b). E_1 is mapped into E_2. The resulting error is bounded by the distance between E_1 and E_2. Let notation $d(X,Y)$ represent the distance between vertices X and Y. We next show that $d(E_1, E_2) \le 4\sqrt{3}E_p$.

   - $d(E_1,E_2) \le d(E_1,E_q) + d(E_2,E_q)$, from triangle (E_1,E_2,E_q)
   - $d(E_1,E_q) \le d(E,E_q) + d(E,E_1) \le \sqrt{3}E_p + \sqrt{3}E_p \le 2\sqrt{3}E_p$, from triangle (E, E_1, E_q)
   - $d(E_2,E_q) \le 2\sqrt{3}E_p$ (see Figure 24)
   - Thus, $d(E_1,E_2) \le 4\sqrt{3}E_p$

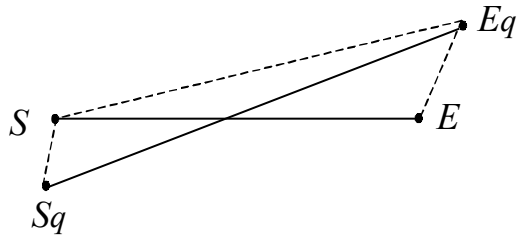


Fig 24 *Let L be the line segment from S to E .The quantization of the coordinates of its endpoints by less than E can change its length by at most* $2\sqrt{3}E_p$.

3. Finally, we stretch or shrink A into A^*, keeping the length and the direction of V constant. A^* only differs from A by its one endpoint. It is represented by the same vector V as A (Fig 23c). We show that the error between A and A^* is bounded by the distance between E_2 and E_q and thus (as was shown in Fig. 24) by $2\sqrt{3}E_p$.

We have shown how to transform A into A^* with error E_1 bounded by $7\sqrt{3}E_p$. The next step is to quantize the representation of V. We first compute the error due to the discretization of an angle of vector V and then the error due to the discretization of the length of vector V.

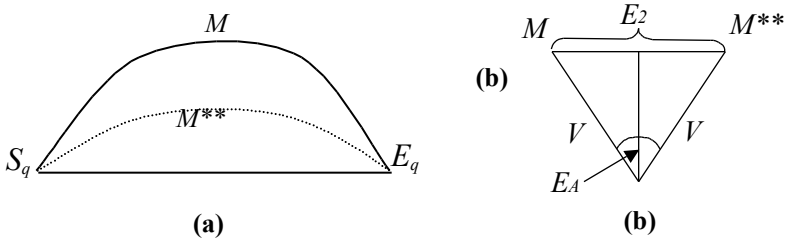


Fig 25 *Error due to quantization of an angle of vector V. Arc A** which results from the discretization is drawn as a dashed arc.*

The discretization of the angle of vector V introduces an error E_2 between A and A^{**}, which is limited by the distance between midpoints M and M^{**} of arcs A and A^{**} respectively (Fig. 25a). Thus, $E_2 \le 2|V|\sin(E_A/2)$ (Fig. 25b), where E_A is a quantization error in angle of vector V.

The Error due to the discretization of the length of vector V, $E_{3,}$, is bounded by the quantization error of length of vector V, E_V (Fig 26).

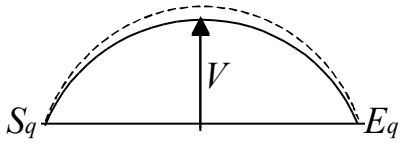


Fig 26 *Error due to the discretization of the length of vector V, $E_{3,}$, is bounded by the quantization error of length of vector V, E_V.*

The maximum error εmax between an arc A and its quantized representation Aq is bounded by the sum of errors E1, E2, E3 and thus the following formula: $\varepsilon_{max} \le 2|V|\sin(E_A/2) + 7\sqrt{3}E_P + E_V$.

Now, we consider the error between arcs A1 and A2 that only differ by one endpoint.

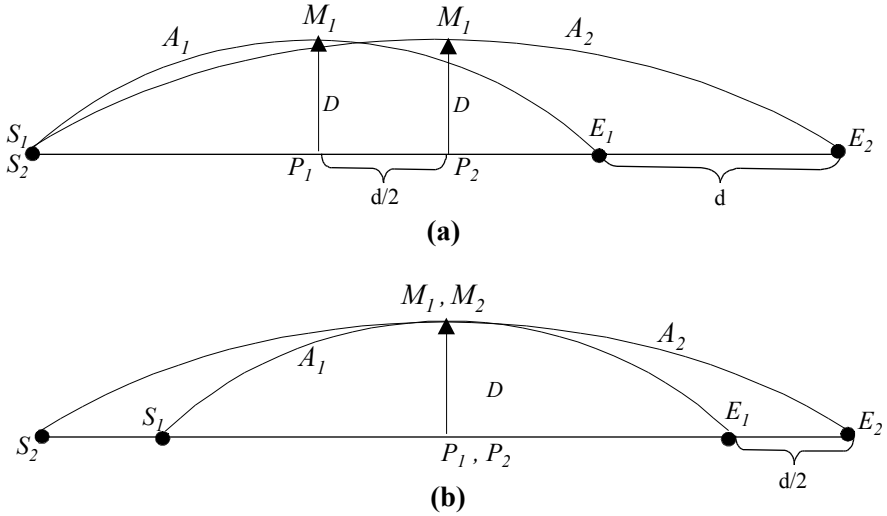


Fig 26: *(a) Two arcs A1 and A2 have the same first endpoints S1 and S2 , and the same deviation parameters of length D, second endpoints E1 and E2 lie on the same line but do not coincide, (b) Translate A1 along vector (E2 – E1) by distance d/2. Now midpoints P1 and P2 of arcs A1 and A2 coincide.*

Consider two arcs A_1 and A_2 in the plane. Each arc is represented by its two endpoints, S and E, and a deviation parameter D between the midpoint $P = (S + E)/2$ and the midpoint M of the arc. Arcs A_1 and A_2 have the same first endpoints S_1 and S_2, and the same deviation parameters of length D, second endpoints E_1 and E_2 lie on the same line but do not coincide (Fig. 26a). We prove that the error between A_1 and A_2 is bounded by the distance d between E_1 and E_2.

Without loss of generality, we assume that A_1 has smaller radius than A_2. Translate A_1 along vector $(E_2 – E_1)$ by distance $d/2$. Now, the midpoints P_1 and P_2 of arcs A_1 and A_2 coincide (Fig 26b). It is trivial to see that the error between the translated A_1 and A_2 is less than $d/2$. Thus the error between A_1 and A_2 is less than d.