Rubber Band Algorithm
for Estimating the Length of Digitized Space-Curves

Thomas Bülow
Institute of Computer Science
Christian Albrechts University
Preusserstrasse 1–9, 24105 Kiel, Germany
tbl@ks.informatik.uni-kiel.de

Reinhard Klette
CITR Tamaki, The University of Auckland
Tamaki Campus, Building 731
Auckland, New Zealand
R.klette@auckland.ac.nz

Abstract

We consider simple digital curves in a 3D orthogonal grid as special polyhedrally bounded sets. These digital curves model digitized curves or arcs in three-dimensional euclidean space. The length of such a simple digital curve is defined to be the length of the minimum-length polygonal curve fully contained and complete in the tube of this digital curve. So far no algorithm was known for the calculation of such a shortest polygonal curve. This paper provides an iterative algorithmic solution, including a presentation of its foundations and of experimental results.

1. Introduction

The analysis of digital curves in 3D space is of increasing practical relevance in volumetric image data analysis. A digital curve is the result of a process (3D skeleton, 3D thinning etc.) which maps captured 'curve-like' objects into well-defined digital curves (see definition below). The length of a simple digital curve in three-dimensional euclidean space is based on the calculation of the shortest polygonal curve in a polyhedrally bounded compact set [5, 6]. This paper presents an algorithm for calculating such a polygonal curve with measured time complexity in $\mathcal{O}(n)$, where n denotes the number of grid cubes of the given digital curve.

We start with the definition of simple digital curves, see Fig. 1 for two examples. Any grid point $(i, j, k) \in R^3$ is assumed to be the center point of a *grid cube* with *faces* parallel to the coordinate planes, with *edges* of length 1, and *vertices*. *Cells* are either cubes, faces, edges or vertices. The intersection of two cells is either empty or a joint *side* of both cells. We consider a non-empty finite set K of cells such that for any cell in K it holds that any side of this cell is also in K. Such a set K is a special finite *euclidean complex*, and thus an *abstract cellular complex* [4] including straightforward definitions of a bounding relation for pairs of cells, and dimensions of cells.

Digital curves g in 3D space are defined as follows, for $n \geq 1$: A *cube-curve* is a sequence $g = (f_0, c_0, f_1, c_1, ..., f_n, c_n)$ of faces f_i and cubes c_i, for $0 \leq i \leq n$, such that faces f_i and f_{i+1} are sides of cube c_i, for $0 \leq i \leq n$ and $f_{n+1} = f_0$. It is *simple* iff $n \geq 4$, and for any two cubes c_i, c_k in g with $|i - k| \geq 2 \ (mod\ n)$ it holds that if $c_i \cap c_k \neq \emptyset$ then either $|i - k| = 2 \ (mod\ n)$ and $c_i \cap c_k$ is an edge, or $|i - k| = 3 \ (mod\ n)$ and $c_i \cap c_k$ is a vertex. A *tube* **g** is the union of all cubes contained in a cube-curve g. It is a polyhedrally-bounded compact set in R^3, and it is homeomorphic with a torus in case of a simple cube-curve. Analogously, *edge-curves* or *face-curves* may be defined in 3D space. This paper deals exclusively with simple cube-curves.

The cube-curve on the left of Fig. 1 is simple, and the cube-curve on the right is not. The latter example shows that the polyhedrally-bounded compact set **g** of a cube-curve g is not necessarily homeomorphic with a torus if each cube of this cube-curve g has exactly two bounding faces in g. A curve is *complete in* **g** iff it has a non-empty intersection with any cube contained in g. Following [5, 6], the *length*

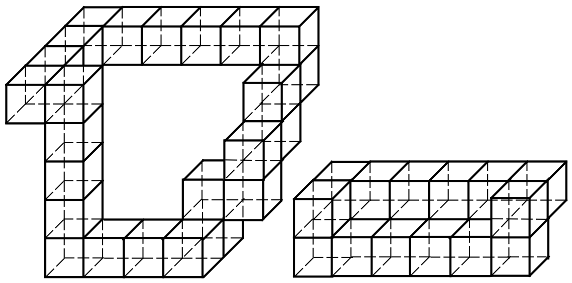

Figure 1. Two cube-curves in 3D space.

of a simple cube-curve g is defined to be the length $l(\mathcal{P})$ of a shortest polygonal simple curve $\mathcal{P}$ which is contained and complete in tube **g**. A simple cube-curve g is *flat* iff the center (grid) points of all cubes contained in g are in one plane parallel to one of the coordinate planes.

A non-flat simple cube-curve in R^3 specifies exactly one minimum-length polygonal simple curve (MLP) which is contained and complete in its tube [5]. The MLP is not uniquely specified in flat simple cube-curves. Flat simple cube-curves may be treated as square-curves in the plane, and square-curves in the plane are extensively studied, see, eg. [3]. It seems there is no straightforward approach to extend known 2D algorithms to the 3D case. Global information has to be taken into account for length calculation of digital curves independent upon the dimension to ensure multigrid convergence [3].

2. Simple cube-curves

This section contains fundamentals used in our algorithm for calculating the length of a simple cube-curve. Proofs are published in [1]. Let g be a simple cube-curve, and $\mathcal{P} = (p_0, p_1, ..., p_m)$ be a polygonal curve complete and contained in **g**, with $p_0 = p_m$.

Lemma 1 *It holds $m \geq 3$ for any polygon $\mathcal{P} = (p_0, p_1, ..., p_m)$ complete and contained in a simple cube-curve.*

The case $m = 3$ is possible. During a traversal along the curve $\mathcal{P}$ we leave cubes, and we enter cubes. The traversal is defined by the starting vertex p_0 of the curve and the given orientation. Let $\mathcal{C}_\mathcal{P} = (c_0, c_1, ..., c_n)$ be the sequence of cubes in the order how they are entered during this curve traversal. Because $\mathcal{P}$ is complete and contained in **g** it follows that $\mathcal{C}_\mathcal{P}$ contains all cubes of g, and no further cubes are in g.

Lemma 2 *For an MLP $\mathcal{P}$ of a simple cube-curve g it holds that $\mathcal{C}_\mathcal{P}$ contains each cube of g just once.*

Now we consider a special transformation of polygonal curves. Let $\mathcal{P} = (p_0, p_1, ..., p_m)$ be a polygonal curve contained in a tube **g**. A polygonal curve $\mathcal{Q}$ is a **g**-*transform* of $\mathcal{P}$ iff $\mathcal{Q}$ may be obtained from $\mathcal{P}$ by a finite number of steps, where each step is a replacement of a triple a, b, c of vertices by a polygonal sequence $a, b_1, ..., b_k, c$ such that the polygonal sequence $a, b_1, ..., b_k, c$ is contained in the same set of cubes of g as the polygonal sequence a, b, c. The case $k = 0$ characterizes the deletion of vertex b, the case $k = 1$ characterizes a move of vertex b within **g**, and cases $k \geq 2$ specify a replacement of two straight line segments by a sequence of $k + 1$ straight line segments, all contained in **g**.

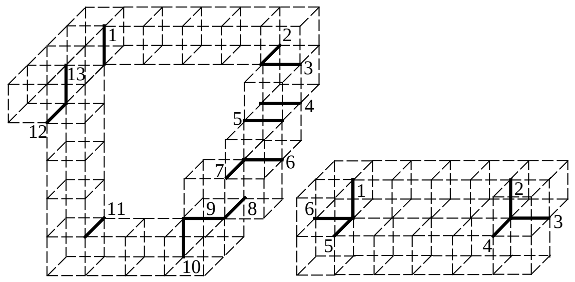

Figure 2. Critical edges of two cube-curves.

Lemma 3 *Let $\mathcal{P}$ be a polygonal curve complete and contained in the tube* **g** *of a simple cube-curve g such that $\mathcal{C}_\mathcal{P}$ is without repetitions of cells. Then it holds that any* **g**-*transform of $\mathcal{P}$ is also complete and contained in* **g**.

An edge contained in a tube **g** is *critical* iff this edge is the intersection of three cubes contained in the cube-curve g. Figure 2 illustrates all critical edges of the cube-curves shown in Fig. 1. Note that simple cube-curves may only have edges contained in three cubes at most. For example, the cube-curve consisting of four cubes only (note: there is one edge contained in four cubes in this case) was excluded by the constraint $n \geq 4$.

Theorem 1 *Let g be a simple cube-curve. Critical edges are the only possible locations of vertices of a shortest polygonal simple curve contained and complete in tube* **g**.

Note that this theorem also covers flat simple cube-curves with a straightforward corollary about the only possible locations of MLP vertices within a simple square-curve in the plane: such vertices may be convex vertices of the inner frontier or concave vertices of the outer frontier only because these are the only vertices incident with three squares of a simple square-curve.

3. Rubber-band algorithm

Our algorithm is based on the following physical model: Assume a rubber band is laid through the tube **g**. Letting it move freely it will contract to the MLP which is contained and complete in **g** (assumed the band is slippery enough to slide across the critical edges of the tube). The algorithm consists of two subprocesses: at first **(A)** an initialization process defining a simple polygonal curve $\mathcal{P}_0$ contained and complete in the given tube **g** and such that $\mathcal{C}_{\mathcal{P}_0}$ contains each cube of g just once (see Lemma 2), and second **(B)** an iterative process (a **g**-transform, see Lemma 3) where each completed run transforms $\mathcal{P}_t$ into $\mathcal{P}_{t+1}$ with $l(\mathcal{P}_t) \geq l(\mathcal{P}_{t+1})$, for $t \geq 0$. Thus the obtained polygonal curve is also complete and contained in g.

(A) The initial polygonal curve will only connect vertices which are end points of consecutive critical edges. For curve initialization, we scan the given curve until the first pair (e_0, e_1) of consecutive critical edges is found which are not parallel or, if parallel, not in the same grid layer (see Fig. 2 (right) for a non-simple cube-curve showing that searching for a pair of non-coplanar edges would be insufficient in this case). For such a pair (e_0, e_1) we start with vertices (p_0, p_1), p_0 bounds e_0 and p_1 bounds e_1, specifying a line segment p_0p_1 of minimum length (note that such a pair (p_0, p_1) is not always uniquely defined). This is the first line segment of the desired initial polygonal curve $\mathcal{P}_0$.

Now assume that $p_{i-1}p_i$ is the last line segment on this curve $\mathcal{P}_0$ specified so far, and p_i is a vertex which bounds e_i. Then there is a uniquely specified vertex p_{i+1} on the following critical edge e_{i+1} such that p_ip_{i+1} is of minimum length. Length zero is possible with $p_{i+1} = p_i$; in this case we skip p_{i+1}, ie. we do not increase the value of i. Note that this line segment p_ip_{i+1} will always be included in the given tube because the centers of all cubes between two consecutive critical edges are colinear. The process stops by connecting p_n on edge e_n with p_0 (note that it is possible that a minimum-distance criterion for this final step may actually prefer a line between p_n and the second vertex bounding e_0, ie. not p_0). See Table 1 for a list of calculated vertices for the cube-curve on the left in Figs. 2 and 4. The first row lists all the critical edges shown in Fig. 2. The second row contains the vertices of the initial polygon shown in Fig. 4 (initialization = first run of the algorithm). For example, vertex b is on edge 2 and also on edge 3, so there is merely one column for (2/3) for these edges.

This initialization process calculates a polygonal curve $\mathcal{P}_0$ which is always contained and complete in the given tube. Note that traversals following opposite orientations or starting at different critical edges may lead to different initial polygons. For example, a 'counterclockwise' scan of the cube-curve shown in Fig. 2 (left), starting at edge 1, selects edges 11 and 10 to be the first pair of consecutive critical edges, and the generated 'counterclockwise' polygon would differ from the one shown in Fig. 4.

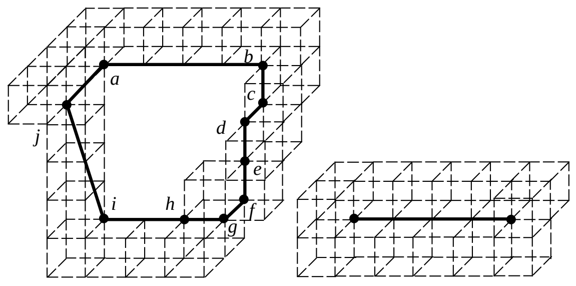

Figure 3. Curve initializations ('clockwise').

E	1	2/3	4	5	6/7	8	9	10	11	12/13
$R1$	a	b	c	d	e	f	g	h	i	j
$R2$	a	b	D	D	e	D	D	h	i	j

Table 1. Calculated points on edges.

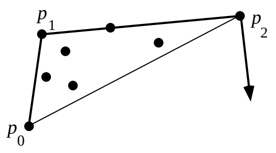

Figure 4. Intersection points with edges.

Initialization results are shown in Fig. 3, and the curve on the right is already an MLP for this non-simple cube-curve. In case of flat cube-curves the process will fail to determine the specified first pair of critical edges, and in this case a 2D algorithm may be used to calculate the MLP of a corresponding square-curve.

(B) In this iterative procedure we move pointers addressing three consecutive vertices of the (so far) calculated polygonal curve around the curve, until a completed run $t + 1$ does only lead to an improvement which is below an a-priori threshold τ i.e. $l(\mathcal{P}_t) - \tau < l(\mathcal{P}_{t+1})$. In all our experiments the algorithm converges fast for a practically reasonable value of τ.

Assume a polygonal curve $\mathcal{P}_t = (p_0, p_1, ..., p_m)$, and three pointers addressing vertices at positions $i - 1$, i, and $i + 1$ in this curve. There are three different options that may occur which define a specific **g**-transform.

$(\mathbf{O_1})$ Point p_i can be deleted iff $p_{i-1}p_{i+1}$ is a line segment within the tube. Then subsequence (p_{i-1}, p_i, p_{i+1}) is replaced in our curve by (p_{i-1}, p_{i+1}). In this case we continue with vertices $p_{i-1}, p_{i+1}, p_{i+2}$.

$(\mathbf{O_2})$ The closed triangular region $\triangle(p_{i-1}p_ip_{i+1})$ intersects more than just the three critical edges of p_{i-1}, p_i and p_{i+1} (see Fig. 4), ie. simple deletion of p_i would not be sufficient anymore. This situation is solved by calculating a convex arc (note: a convex polygon is the shortest curve encircling a given finite set of planar points [2]) and by replacing point p_i by the sequence of vertices q_1,...,q_k on this convex arc between p_{i-1} and p_{i+1} iff the sequence of line segments $p_{i-1}q_1, \ldots, q_kp_{i+1}$ lies within the tube. Because the vertices are ordered we may use a fast linear-time convex hull routine in case of $(\mathbf{O_2})$. Barycentric coordinates with basis $\{p_{i-1}, p_i, p_{i+1}\}$ may be used to decide which of the intersection points is inside the triangle or not.[1] In this case we continue with a triple of vertices starting with the calculated new vertex q_k. If $(\mathbf{O_1})$ and $(\mathbf{O_2})$ do not lead to

[1]In the majority of such cases we found $k = 1$, i.e. p_i is replaced by q_1.

any change, the third option may lead to an improvement.

$(\mathbf{O_3})$ Point p_i may be moved on its critical edge to obtain an optimum position p_{new} minimizing the total length of both line segments $p_{i-1}p_{new}$ and $p_{new}p_{i+1}$. That's a *move on a critical edge* and an $\mathcal{O}(1)$ solution is given below. Then subsequence (p_{i-1}, p_i, p_{i+1}) is replaced in our curve by $(p_{i-1}, p_{new}, p_{i+1})$. In this case we continue with vertices $p_{new}, p_{i+1}, p_{i+2}$.

In Table 1 we sketch the fact that the second run starts with the polygonal curve $\mathcal{P}_1 = (a, b, c, d, e, f, g, h, i, j)$. For the first triple (a, b, c) we have: none of the cases $(\mathbf{O_1})$ to $(\mathbf{O_3})$ leads to a better location of b. For triple (b, c, d) we delete c according to $(\mathbf{O_1})$ (symbol 'D'). Then triple (b, d, e) leads to the deletion of d, etc., finally (j, a, b) does not delete or move a. In the following round nothing changes. There are no other possibilities besides $(\mathbf{O_1})$...$(\mathbf{O_3})$. In our experiments we chose the threshold $\tau = l(\mathcal{P}_1) \cdot 10^{-6}$. Figure 5 shows another example. The initial polygon $\mathcal{P}_1$ is dashed. The solid line represents the final polygon. The short line segments are the critical edges of the given tube.

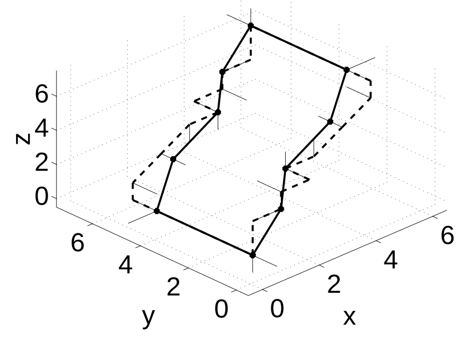

Figure 5. Initial polygon (dashed) and MLP.

4. Move on a critical edge

We consider situation $(\mathbf{O_3})$, ie. p_i lies on a critical edge, say e, and is not colinear with $p_{i-1}p_{i+1}$. Let l_e be the line containing the edge e. First, we find the point $p_{opt} \in l_e$ such that

$$|p_{opt} - p_{i-1}| + |p_{i+1} - p_{opt}| = \min_{p \in l_e}(|p - p_{i-1}| + |p_{i+1} - p|).$$

If p_{opt} lies on the closed critical edge e we simply replace p_i by p_{opt}. If it does not, we replace p_i by that vertex bounding e and lying closest to p_{opt}.

We describe how the point p_{opt} can be found. The set of all points $p \in R^3$ for which

$$|p - p_{i-1}| + |p_{i+1} - p| = k \ , \tag{1}$$

for some real constant $k \geq |p_{i+1} - p_{i-1}|$ is a prolate spheroid $\mathcal{S}$, ie. an ellipsoid, the two shorter principle axes of which have the same length. In the following we use three Cartesian coordinate systems of R^3 which are interrelated by rigid motions. (**I**) The coordinates in the coordinate system used up to now are denoted by lower case letters. (**II**) Coordinates in the coordinate system attached to $\mathcal{S}$ are denoted by upper case letters $(X, Y, Z) \in R^3$. In these coordinates $\mathcal{S}$ is represented by

$$\frac{X^2}{c^2 + \varepsilon} + \frac{Y^2 + Z^2}{\varepsilon} - 1 = 0, \tag{2}$$

with $c := |p_{i+1} - p_{i-1}|/2$ and $\varepsilon := k^2/4 - c^2$. (**III**) The coordinate system (x', y', z') is chosen such that l_e is identical to the z'-axis.

Depending upon the choice of ε, the line l_e has up to two intersection points with $\mathcal{S}$. If l_e is tangent to $\mathcal{S}$, the tangent point is p_{opt}, since there is no point $q \in l_e$ lying inside $\mathcal{S}$. Thus, we first have to find the value of ε such that l_e is tangent to $\mathcal{S}$. Afterwards we identify the intersection point of l_e and $\mathcal{S}$ as p_{opt}.

Representing (2) in coordinate system (**III**) and intersecting $\mathcal{S}$ with l_e by setting $x' = y' = 0$ yields a quadric equation in z':

$$a(\varepsilon){z'}^2 + b(\varepsilon)z' + c(\varepsilon) = 0 \ . \tag{3}$$

The coefficients a and b contain ε linearly, while c is quadratic in ε. The constraint that l_e be tangent to $\mathcal{S}$ is identical to the condition that (3) has a double solution. This is the case iff

$$a(\varepsilon)c(\varepsilon) - b^2(\varepsilon) = 0 \ . \tag{4}$$

Equation (4) is cubic in ε and has one real positive and two real negative solution. Since geometrically only the positive solution makes sense, we have a unique solution. Replacing ε into (3), solving that equation, and representing the solution in coordinate system (**I**) yields p_{opt}.

5. Time complexity and convergence of algorithm

We give an estimate of the complexity of the rubber-band algorithm in dependency of the number of cubes n of the given cube-curve.

The algorithm completes each run in $\mathcal{O}(n)$ time. The described move of point p_i on a critical edge requires constant time. The given value of τ ensured that the measured time complexity is in $\mathcal{O}(n)$, ie. the number of runs does not depend upon n. Figures 6 and 7 show the time needed until the algorithm converges in dependency of the number of cubes n and in dependency of the number of critical edges, respectively. The test set contained 70 randomly generated cube-curves. In Fig. 6 each error bar shows the mean convergence time and the standard deviation for a set of 10 digital curves.

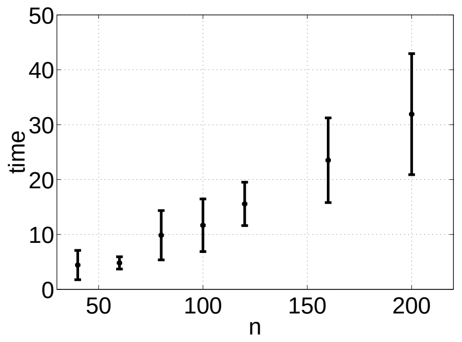

Figure 6. CPU-time in seconds in dependence of the number of cubes (n)

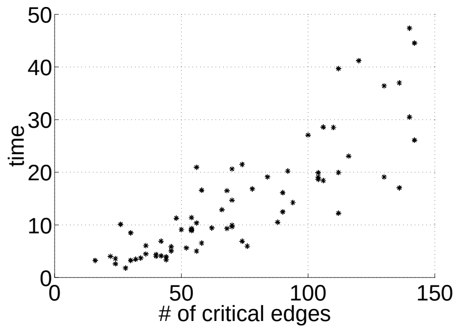

Figure 7. CPU-time in seconds in dependence of the number of critical edges

The algorithm is iterative and since by definition $l(\mathcal{P}_{t+1}) < l(\mathcal{P}_t)$ (otherwise, if equal, the algorithm stops) and there exists a lower bound for $l(\mathcal{P}_t)$ (namely the length of the MLP) the algorithm converges. However, the open problem remains: convergent towards which polygon? In all of our experiments it converges towards the MLP. Lemma 2 and Lemma 3 give a partial answer: always towards a polygon complete and contained in **g**.

Figure 8 shows the measured changes in the length of the polygons within the iterative process.

6. Conclusions

The given algorithmic solution provides a polygonal approximation and length measurement for simple cube-curves in 3D space with a measured time in $\mathcal{O}(n)$. Actually it was successfully used for a wider class of cube-curves also allowing cases such as the cube-curve on the right in Fig. 1: cube-curves g where each cube in g has exactly two bounding faces in g. This definition allows a simpler test of a given cube-curve, and also a simpler generation of test examples.

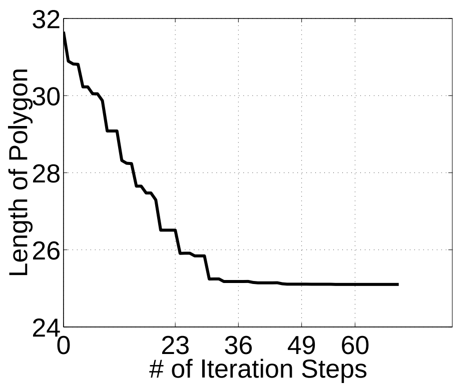

Figure 8. Length convergence for the curve shown in Fig. 5. The tick-marks on the iteration axis are set after each completed run.

The curve used in the initialization step **(A)** might be replaced by other curves such as sequences of straight segments between midpoints of consecutive cubes, or between midpoints of consecutive critical edges. These two curves would also be curves lying completely within a given tube of a simple cube-curve. However, the initial curve as defined in our algorithm leads to a much faster convergence in general compared to these two other options.

Two open problems are stated: the time complexity might be provable always in $\mathcal{O}(n)$, and the convergence might be provable always towards the MLP.

References

[1] T. Bülow and R. Klette. Digital curves in 3d space and linear-time length estimation. Technical Report CITR-TR-55, CITR Tamaki, Auckland University, December 1999.

[2] T. Busemann and W. Feller. Krümmungseigenschaften konvexer flächen. *Acta Mathematica*, 66:27–45, 1935.

[3] R. Klette, V. Kovalevsky, and B. Yip. On the length estimation of digital curves. In *Vision Geometry VIII*, number 3811 in SPIE Conference proceedings, pages 117–128, 1999.

[4] W. Rinow. *Topologie*. Deutscher Verlag der Wissenschaften, Berlin, 1975.

[5] F. Sloboda and Ľ. Bačík. On one-dimensional grid continua in R^3. Technical report, Institute of Control Theory and Robotics, Bratislava, 1996.

[6] F. Sloboda, B. Zaťko, and R. Klette. On the topology of grid continua. In *Vision Geometry VII*, number 3454 in SPIE Conference Proceedings, pages 52–63, 1998.