Computer Science Department of The University of Auckland
CITR at Tamaki Campus (http://www.citr.auckland.ac.nz)

CITR-TR-91 June 2001

Approximation of 3D Shortest Polygons in Simple Cube Curves

Thomas Bülow[1] and Reinhard Klette[2]

Abstract

One possible definition of the length of a digitized curve in 3D is the length of the shortest polygonal curve lying entirely in a cube curve. In earlier work the authors proposed an iterative algorithm for the calculation of this minimal length polygonal curve (MLP). This paper reviews the algorithm and suggests methods to speed it up by reducing the set of possible locations of vertices of the MLP, or by directly calculating MLP-vertices in specific situations. Altogether, the paper suggests an in-depth analysis of cube curves.

[1] Department of Computer and Information Science University of Pennsylvania, Philadelphia, USA
[2] Center for Image Technology and Robotics Tamaki Campus, The University of Auckland, Auckland, New Zealand r.klette@auckland.ac.nz

Approximation of 3D Shortest Polygons in Simple Cube Curves

Thomas Bülow [1] and Reinhard Klette [2]

[1] Department of Computer and Information Science
University of Pennsylvania, Philadelphia, USA
[2] CITR, University of Auckland, Tamaki Campus, Building 731
Auckland, New Zealand

Abstract. One possible definition of the length of a digitized curve in 3D is the length of the shortest polygonal curve lying entirely in a cube curve. In earlier work the authors proposed an iterative algorithm for the calculation of this minimal length polygonal curve (MLP). This paper reviews the algorithm and suggests methods to speed it up by reducing the set of possible locations of vertices of the MLP, or by directly calculating MLP-vertices in specific situations. Altogether, the paper suggests an in-depth analysis of cube curves.

1 Introduction

The analysis of digital curves in 3D space is of increasing practical relevance in volumetric image data analysis, see cited examples of applications in [5, 6]. A digital curve is the result of a process (path on a 3D surface, 3D thinning etc.) which maps captured 'curve-like' objects into well-defined digital curves (see definition below). The length of such a simple digital curve may be defined based on local chain-code configurations in digital space [6], or on (global) curve approximations in Euclidean space. The latter approach has the potential of multi-grid convergence of estimated length toward the true length assuming that a digitization model such as cube intersection [6] is used for obtaining test data by digitizing curves of known length. For global length estimation algorithms there are (at least) two options, the use of

1. the length of a 3D NSS (naive straight segment) approximation [5] of a 26-connected curve, or
2. the length of the 3D MLP (minimum-length polygonal) curve fully contained and complete in the tube of a simple 6-connected closed curve [9, 10].

In earlier work the authors started with studying an algorithmic solution for the 3D MLP approach and proved that the only possible positions of MLP-vertices are the so-called *critical edges* which are incident with three cubes of the simple cube-curve [8], and presented an iterative algorithm for approximating the MLP of a simple cube-curve in 3D [1]. It is expected [5] that the length estimation behavior of 3D NSS compares to 3D MLP similarly as that of DSS to MLP

in the two-dimensional case [7]. However, to support such experiments we have to continue to improve the time complexity of the 3D MLP approximation as specified so far in [1] for making statistically relevant performance evaluations feasible.

The difficulty of the subject may be illustrated by the fact that the *Euclidean shortest path problem* (given a finite collection of polyhedral obstacles in 3D space, and a source and a target point, find a shortest obstacle-avoiding path from source to target) is known to be NP-hard [3]. However, there are polynomial-time algorithms solving the *approximate Euclidean shortest path problem* in 3D, see [4]. Our iterative algorithm in [7] is not yet known to be always convergent to the exact 3D MLP, or whether it may only approximate in well-defined cases, up to some error etc. the correct MLP. All our experiments so far suggest that the algorithm presented in [1] is always convergent to the correct 3D MLP, and time measurements also support the hypothesis that its run-time behavior is asymptotically linear in the number of input cubes even if a very small threshold is used for termination of the algorithm.

In this paper we review this algorithm and discuss three ways to speed it up: 1) Not all critical edges contain MLP-vertices. We identify a subset of these irrelevant critical edges which can then be excluded from further calculation. 2) Certain critical edges can be identified which can only contain MLP-vertices – if at all – at their end-points. 3) The positions of MLP-vertices belonging to flat arcs of the MLP can under certain conditions be calculated in closed form. Therefore the present paper may also be seen as a step towards an in-depth analysis of the geometry of 3D MLP's in simple cube curves.

Although we are not able yet to provide a closed form solution for a 3D MLP algorithm of a given simple cube-curve, these three steps lead in this direction by replacing parts of the iterative procedure by direct computation.

In the following section the basic notions are introduced. We define the length of a simple cube-curve. In Section 3 we summarize our previous algorithm for the calculation of this length. In Section 4 the three items mentioned above are elaborated.

2 The Length of Simple Cube Curves

We start with the definition of simple cube curves, see Fig. 1 for two examples. Any grid point $(i, j, k) \in \mathbb{R}^3$ 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* at its corners. *Cells* are either cubes, faces, edges or vertices. The intersection of two cells is either empty or a joint *side* of both cells. 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* $\mathbf{g}$ is the union of all cubes contained in a cube-curve g. It is a polyhedrally-bounded compact set in $\mathbb{R}^3$,

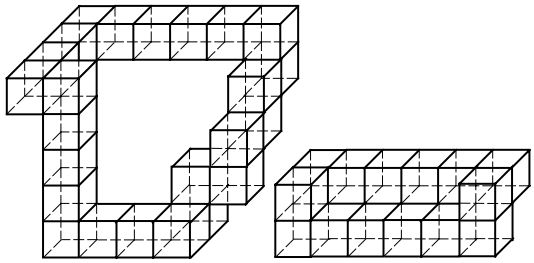

Fig. 1. Two cube-curves in 3D space.

and it is homeomorphic with a torus in case of a simple cube-curve. The cube-curve on the left of Fig. 1 is simple, and the cube-curve on the right is not. Analogously, *edge-curves* or *face-curves* may be defined in 3D space. This paper deals exclusively with simple cube-curves. A curve $\mathcal{P}$ in 3D Euclidean space is *complete in* $\mathbf{g}$ iff it has a non-empty intersection with any cube contained in g. Following [9,10], the *length* 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 $\mathbf{g}$. 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 $\mathbb{R}^3$ specifies exactly one minimum-length polygonal simple curve (MLP) which is contained and complete in its tube [10]. 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, e.g. [7]. It seems there is no straightforward approach to extend known 2D MLP algorithms to the 3D case.

3 The Iterative Algorithm

This section contains fundamentals used in our algorithm presented in the conference paper [1] for calculating the length of a simple cube-curve. Let g be a simple cube-curve, and $\mathcal{P} = (p_0, p_1, ..., p_m)$ be a polygonal curve complete and contained in $\mathbf{g}$, 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 $\mathbf{g}$ 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 $\mathbf{g}$. A polygonal curve $\mathcal{Q}$ is a $\mathbf{g}$-*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 $\mathbf{g}$, 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 $\mathbf{g}$.

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

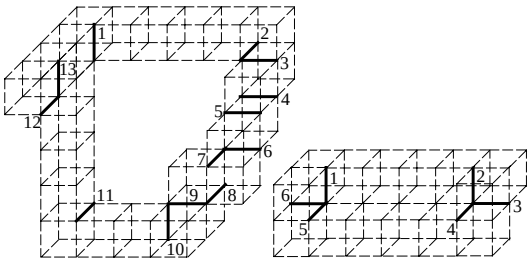

Fig. 2. Critical edges of two cube-curves.

An edge contained in a tube $\mathbf{g}$ 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 $n \geq 4$. Based on these lemmata it was possible to prove the following theorem [8]:

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 $\mathbf{g}$.*

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.

Our algorithm is based on the following model: Assume a rubber band is laid through the tube $\mathbf{g}$. Letting it move freely it will contract to the MLP which

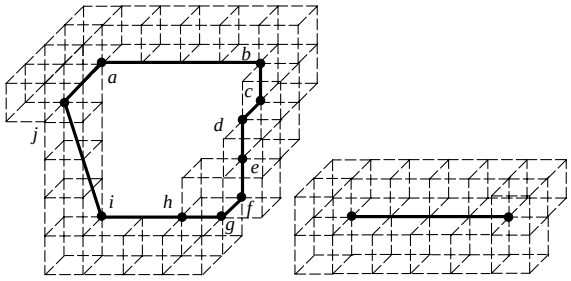

Fig. 3. Curve initializations ('clockwise').

is contained and complete in **g**. 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}, i.e. 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 collinear.

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, i.e. not p_0). This initialization process calculates a polygonal curve $\mathcal{P}_0$ which is always contained and complete in the given tube.

(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-

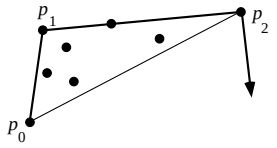

Fig. 4. Intersection points with edges.

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), i.e. 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 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 $p_{new}, p_{i+1}, p_{i+2}$.

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

$$|\boldsymbol{p}_{opt} - \boldsymbol{p}_{i-1}| + |\boldsymbol{p}_{i+1} - \boldsymbol{p}_{opt}| = \min_{\boldsymbol{p} \in l_e} L(\boldsymbol{p})$$

with

$$L(\boldsymbol{p}) = (|\boldsymbol{p} - \boldsymbol{p}_{i-1}| + |\boldsymbol{p}_{i+1} - \boldsymbol{p}|).$$

If $\boldsymbol{p}_{opt}$ lies on the closed critical edge e we simply replace $\boldsymbol{p}_i$ by $\boldsymbol{p}_{opt}$. If it does not, we replace $\boldsymbol{p}_i$ by that vertex bounding e and lying closest to $\boldsymbol{p}_{opt}$.

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

We give a slightly simpler solution for finding the point $\boldsymbol{p}_{opt}$ then provided in [1]. W.l.o.g. let us assume that l_e is parallel to the x-axis:

$$l_e = \{(t, y_e, z_e)^T | t \in \mathbb{R}\}$$

for some fixed y_e and z_e. If $x_{i-1} = x_{i+1}$ we find $\boldsymbol{p}_{opt} = (x_{i-1}, y_e, z_e)^T$. Otherwise

$$\frac{\partial L}{\partial x}(\boldsymbol{p}_{opt}) = 0$$

leads to a quadric equation in x_{opt},

$$(\alpha_{i+1} - \alpha_{i-1})x_{opt}^2 + 2(\alpha_{i-1}x_{i+1} - \alpha_{i+1}x_{i-1})x_{opt} + \alpha_{i+1}x_{i-1}^2 - \alpha_{i-1}x_{i+1}^2 = 0 \ ,$$

with $\alpha_i = (y_e - y_i)^2 + (z_e - z_i)^2$.

Figure 5 illustrates a given simple cube curve by its critical edges only, the initial polygon and the correct MLP as calculated by our algorithm. Note that some of the vertices subdivide critical edges in a rational ratio specified by the heights of consecutive linear cube segments.

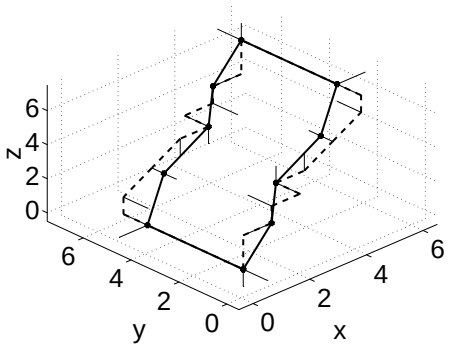

Fig. 5. Initial polygon (dashed) and correctly calculated MLP. Critical edges are shown as short line segments. The rest of the tube is not shown.

We illustrate the performance of the algorithm with respect to length measurement on circles in $\mathbb{R}^3$. We generate circles by

$$\boldsymbol{c}(t) = R_x(\phi_1)R_z(\phi_2)R_x(\phi_3)\begin{pmatrix} r\cos(t) \\ r\sin(t) \\ 0 \end{pmatrix}, \quad t \in [0, 2\pi[.$$

Here, R_x and R_z are the 3×3 rotation matrices about the x- and the z-axis, respectively. The angles ϕ_1, ϕ_2 and ϕ_3 are randomly chosen from a uniform distribution in the interval $[0, \pi]$. The radius r is randomly chosen from the interval $[0.5, 1]$. In each experiment a circle is generated and digitized on a 3D grid. The

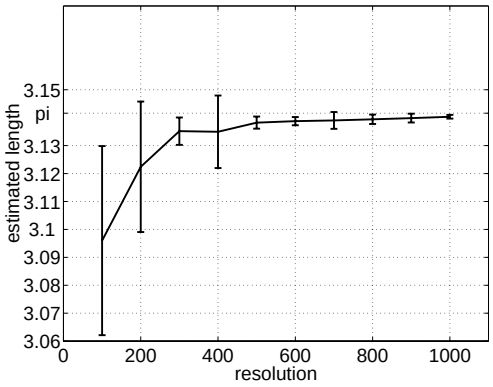

Fig. 6. Results of the length estimation of digitized circles with resolution 100^3 to 1000^3 using $\tau = l(\mathcal{P}_t) \cdot 10^{-7}$.

grid size varies between 100^3 and 1000^3. We performed experiments using different thresholds τ, and Fig. 6 illustrates the case of $\tau = l(\mathcal{P}_t) \cdot 10^{-7}$ and resolutions $(100 \cdot n)^3, n \in \{1, 2, \ldots, 10\}$. For each combination we estimated the length of 25 digitized circles. We normalized the results such that the true circumference of the underlying circles is π.

4 Analysis of Cube Curves

In it's current form the above algorithm is slower than necessary. It can be sped up by taking into account more of the available information about the possible positions of vertices of the MLP. Here we propose three ways of doing so: 1) Not all critical edges are relevant, i.e., not every critical edge can contain a vertex of the MLP. Removing irrelevant critical edges reduces the number of necessary computations. 2) In certain situations a critical edge can contain a vertex of the MLP only – if at all – at one of its endpoints. This information can be used for the initialization of the algorithm. 3) Consider a subsequence of the MLP which is contained in one fixed grid-layer. Assume the correct endpoints are known as well as the critical edges which contain the intermediate vertices of the MLP. Under these conditions the positions of the intermediate vertices can be calculated in closed form.

Considerations as started in this section may lead to a better understanding whether it is possible to derive a non-iterative, i.e. closed form of an algorithm calculating 3D MLPs within simple cube curves.

4.1 Removing Irrelevant Critical Edges

Consider a flat arc of a simple cube curve, i.e. an arc all of which cubes lie within one cube-layer. As shown in Fig. 7 this arc can be projected to a 2D plane, where the cube-curve becomes a face-curve. The 2D MLP algorithm reported in [7] may

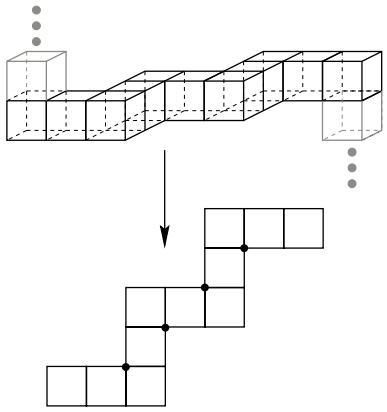

Fig. 7. A subsequence of a simple cube curve lying entirely in one cube-layer (top). Its projection to a 2D face-curve. The vertices marked by dots indicate the corresponding critical edges.

be used for such a planar segment of a simple cube curve. The following theorem also shows how in this case irrelevant critical edges can be identified.

Theorem 2. *Let $E = \{e_i, e_{i+1}, e_{i+2}, e_{i+3}, e_{i+4}\}$ be a sequence of consecutive, parallel critical edges. Let e_i, e_{i+2} and e_{i+4} be coplanar. Consider the projection of the cube-curve segment containing E as in Fig. 7. Then e_{i+2} contains no vertex of the MLP if line segment $e_i e_{i+4}$ intersects the boundary of the face-curve at e_{i+2} only.*

Figure 8 illustrates the situation and no further proof is needed.

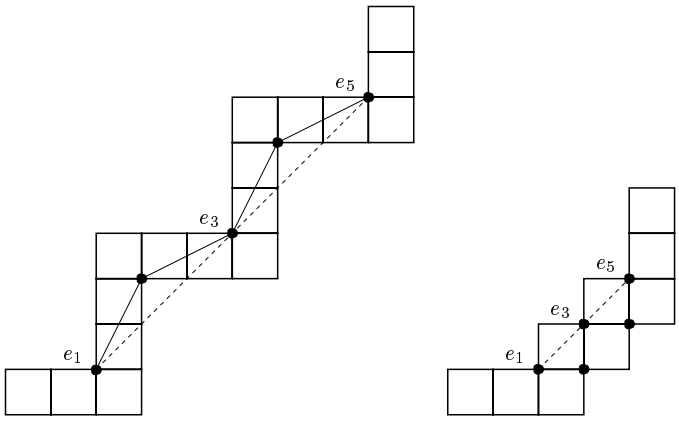

Fig. 8. Illustration of Theorem 2. Left: e_3 contains a vertex of the MLP. Right: e_3 does not contain a vertex of the MLP and can be removed from the list of critical edges.

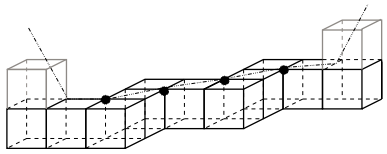

Fig. 9. Entering and leaving in same direction.

4.2 Vertices on Endpoints of Critical Edges

We again investigate flat arcs of the cube-curve. The following theorem shows that in a situation like the one shown in Fig. 9 the only possible positions of MLP-vertices are the endpoints of critical edges.

Lemma 4. *Let $E = (e_i, e_{i+1}, \ldots, e_{i+n-1})$ be a sequence of consecutive critical edges. If all these e_j's are pairwise parallel then they lie within the same grid layer (cube-layer).*

Proof. Moving into another grid-layer results in a critical edge in the plane between both grid layers, i.e. which is not parallel to the e_j's in the sequence E before.

It holds that $E = (e_i, e_{i+1}, \ldots, e_{i+n-1})$ is a *maximum-length sequence of parallel critical edges* iff all the e_j's contained in E are pairwise parallel and e_{i-1} and e_{i+n} are not parallel to e_i (and thus not parallel to any edge contained in E).

Theorem 3. *Let $E = (e_i, e_{i+1}, \ldots, e_{i+n-1})$ be a maximum-length sequence of parallel critical edges. If e_{i-1} and e_{i+n} are both in the upper (lower) face-layer, the only possible vertex positions on the edges $e_i, \ldots, e_{i-n-1}$ are the upper (lower) endpoints of these edges.*

Note, that by speaking about "upper" and "lower" face-layer, we assumed a horizontal layer. The same theorem holds if we replace "upper" and "lower" by "left" and "right" or by "front" and "back".

Proof. The shortest path connecting an edge-sequence $E = (e_i, e_{i+1}, \ldots, e_{i+n-1})$ lies within a horizontal plane, i.e., the positions of all vertices have the same z-coordinate (x-coordinate for "left" and "right", y-coordinate for "front" and "back"). Since the left and right neighbors of E are both above (below) the cube-layer, it is optimal to move the vertices to the upper (lower) endpoints of the edges $e_i, \ldots, e_{i-n-1}$.

4.3 Closed Form Evaluation of Vertex Positions

Let $(e_i, e_{i+1} \ldots, e_{i+n})$ be a sequence of consecutive parallel critical edges and $E = (e_i, e_j \ldots, e_k)$ the sequence which results from the former sequence by

removing all the critical edges containing no vertex of the MLP. W.l.o.g. we assume that E lies within a horizontal cube-layer.

Consider the sequence of the vertices $(p_a, p_{e_i}, p_{e_j}, \ldots, p_{e_k}, p_b)$ being a subsequence of the MLP. Here p_a and p_b are assumed to be known, and p_e denotes the vertex lying on e. Define

$$l = |p_a p_{e_i}| + |p_{e_i} p_{e_j}| + \ldots + |p_{e_k} p_b|,$$

and let $(p)_z$ be the z-coordinate of the vertex p.

Theorem 4. *For the specified situation it holds that the z-coordinates of $p_{e_i}, p_{e_j}, \ldots, p_{e_k}$ are given by linear interpolations*

$$(p_{e_i})_z = (p_a)_z + \frac{(p_b)_z - (p_a)_z}{l} |p_a e_i|,$$
$$(p_{e_j})_z = (p_{e_i})_z + \frac{(p_b)_z - (p_a)_z}{l} |e_i e_j|, \ \ldots$$

Proof. We represent e_i by the set of points in $\mathbb{R}^3$ which lie on e_i:

$$e_i = \{(x_i, y_i, z + \lambda_i) | \lambda_i \in [0,1]\}.$$

We introduce a function σ acting on the indices such that

$$E = (e_{\sigma(1)}, e_{\sigma(2)} \ldots, e_{\sigma(m)}) \ .$$

Further we denote the positions of the vertices p_a and p_b by

$$p_a = (x_{\sigma(0)}, y_{\sigma(0)}, \lambda_{\sigma(0)}) \quad \text{and} \quad p_b = (x_{\sigma(m+1)}, y_{\sigma(m+1)}, \lambda_{\sigma(m+1)}) \ .$$

The length of the sequence $(p_a, p_{e_i}, p_{e_j}, \ldots, p_{e_k}, p_b)$ is thus given by

$$L(\lambda_{\sigma(1)}, \ldots, \lambda_{\sigma(m)}) = \sum_{i=1}^{m} \sqrt{(|e_{\sigma(i)} e_{\sigma(i+1)}|)^2 + (\lambda_{\sigma(i)} - \lambda_{\sigma(i+1)})^2} \ ,$$

where $(|e_{\sigma(i)} e_{\sigma(i+1)}|)^2 = (x_{\sigma(i)} - x_{\sigma(i+1)})^2 + (y_{\sigma(i)} - y_{\sigma(i+1)})^2$. Since L does not depend on the x_i and y_i directly but merely on the horizontal distances between consecutive critical edges $|e_{\sigma(i)} e_{\sigma(i+1)}|$ we can replace E by E' with edges

$$e'_i = \{(x'_i, 0, \lambda_i) | \lambda_i \in [0,1]\}$$

such that

$$x'_{\sigma(i)} - x'_{\sigma(i+1)} = |e_{\sigma(i)} e_{\sigma(i+1)}|$$

which leads to

$$L = \sum_{i=1}^{m} \sqrt{(x'_{\sigma(i)} - x'_{\sigma(i+1)})^2 + (\lambda_{\sigma(i)} - \lambda_{\sigma(i+1)})^2}.$$

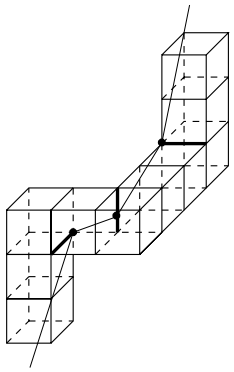

Fig. 10. An example for the application of Theo. 4: The position of the middle MLP-vertex can be calculated using the theorem. In this case $\lambda = 1/3$. An arc of the MLP is represented by the thin line. The bold edges of the cube curve are the critical edges.

Minimizing L with respect to $\lambda_{\sigma(1)}, \ldots, \lambda_{\sigma(m)}$ leads to

$$\lambda_{\sigma(i)} = \lambda_{\sigma(0)} + \frac{\lambda_{\sigma(m+1)} - \lambda_{\sigma(0)}}{x'_{\sigma(m+1)} - x'_{\sigma(0)}}(x'_{\sigma(i)} - x'_{\sigma(0)}).$$

Converting to the previous notation yields the formulae in the theorem.

Figure 10 shows a simple situation in which Theo. 4 can be applied. We finally present an example of a cube-curve with an MLP which can be calculated in closed form (see Fig. 11). The vertices in the left and right arcs, which lie in the Y-Z plane, can by Theo. 3 found to lie on the endpoints of critical edges. Which of the critical edges in these arcs are effective can be found by a 2D method for face-curves. The latter is also true for the two other arcs which lie in the X-Y plane. The locations of the vertices on the effective critical edges of these arcs can be found by Theo. 4.

5 Conclusion

In this paper we reviewed an algorithm for iterative approximation of 3D MLP of a simple cube curve and initiated a discussion whether it is possible or not to derive a closed-form, i.e. non-iterative algorithm for this computational problem. We have shown that certain irrelevant critical edges can be identified, and how the positions of some MLP-vertices can directly be calculated without an iterative procedure. These first steps towards a better understanding of simple cube curves are based on a segmentation of 3D cube curves into 2D face curves.

Triplets of run-length sequences may play a crucial role in analyzing simple cube curves. Starting with one cube of a simple cube curve we may assign one sequence of numbers to any of the three coordinate axes defined by consecutive

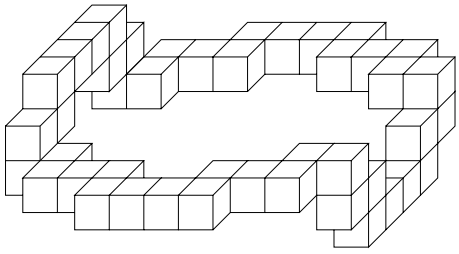

Fig. 11. The MLP of the shown cube-curve can be calculated in closed form using the presented methods. See text for details.

run length' parallel to this coordinate axis (i.e. 'how many consecutive cubes are in the same layer with respect to the coordinate axis?'). These sequences may be calculated during a first run through the given cube curve. For example, starting at the lower left cube of the simple cube curve shown in Fig. 1 on the left, and going to the right, we have sequences

$$1, 1, 1, 3, 6, 1, 1, 1, 1, 4, 4+$$

(4+ indicates that this run is in the same layer as the first cube) for the X-coordinate axis (left to right),

$$4, 4, 2, 7, 1, 1, 5+$$

(again continuation of the same layer as the first 4 cubes) for the foreground-to-background Y-axis, and

$$5, 2, 2, 2, 10, 1, 1, 1$$

for the bottom-to-top Z-axis. This shows that the longest run of 10 cubes appears in one Z-layer, matching the situation of Theorem 3, followed by a run of 9 cubes in one Y-layer. Such a longest run might be a good start for a 3D MLP algorithm. An analysis of these *run-length sequences* is also expected to allow a more direct calculation of (possible) positions of MLP vertices on critical edges. Figure 12 shows that these may be at irrational positions.

The evaluation of the practical relevance of algorithmic proposals (as discussed for the speed-up of the iterative algorithm) is particularly difficult, since the effect of these depends highly on the class of specific cube curves under consideration. For example, it is easily possible to construct curves with no irrelevant critical edges at one hand, and curves containing almost only irrelevant critical edges, on the other hand. Similarly, the applicability of Theorem 4 depends on the length of flat arcs in the cube-curve.

The effect of methods such as presented here on the complexity of the discussed iterative algorithm will be subject to future research, in close relation to methods modeling or generating specific classes of cube curves. For example, it will be worthwhile to investigate whether Theorem 3 allows the identification

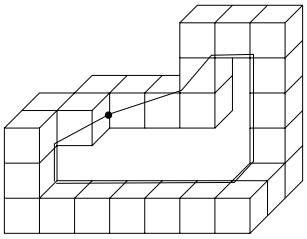

Fig. 12. A cube curve with an MLP-vertex at an irrational position: As a consequence of Theo. 4 the vertex marked by the dot is at $\lambda = \sqrt{2}/(2+\sqrt{2})$.

of at least two MLP-vertices for any given cube curve. This would allow a segmentation of a cube curve into two cube arcs, and the independent treatment of both arcs.

References

1. Th. Bülow and R. Klette. Rubber Band Algorithm for Estimating the Length of Digitized Space-Curves. In: Proc. 15th Intern. Conf. *Pattern Recognition*, 2000, vol. 3:551–555.
2. R. Busemann and W. Feller. Krümmungseigenschaften konvexer Flächen. *Acta Mathematica*, 66:27–45, 1935.
3. J. Canny and J.H. Reif. New lower bound techniques for robot motion planning problems. *IEEE Foundations of Computer Science*, 28:49–60, 1987.
4. J. Choi, J. Sellen, C.-K. Yap. Approximate Euclidean shortest path in 3-space. In: Proc. 10th ACM Conf. *Computat. Geometry*, 41–48, 1994.
5. D. Coeurjolly, I. Debled-Rennesson, O. Teytaud. Segmentation and length estimation of 3D curves. Chapter in this book.
6. A. Jonas, N. Kiryati. Length estimation on 3-D using cube quantization. *JMIV*, 8:215–238, 1998.
7. R. Klette, V. Kovalevsky, and B. Yip. On the length estimation of digital curves. In: Proc. SPIE Conf. *Vision Geometry VIII*, 3811:52–63, 1999.
8. R. Klette and Th. Bülow. Critical edges in simple cube-curves. In: Proc. *DGCI'2000*, Uppsala December 2000, LNCS **1953**, 467-478.
9. F. Sloboda, B. Zatko, and P. Ferianc. Minimum perimeter polygon and its application. in: *Theoretical Foundations of Computer Vision* (R. Klette, W.G. Kropatsch, eds.), Mathematical Research **69**, Akademie Verlag, Berlin, 59-70, 1992.
10. F. Sloboda and Ľ. Bačík. *On one–dimensional grid continua in* R^3. Report of the Institute of Control Theory and Robotics, Bratislava, 1996.