Two Approximative Algorithms for Calculating Minimum-Length Polygons in 3D Space

Fajie Li and Reinhard Klette

CITR, Computer Vision Unit, Computer Science Department
The University of Auckland, Auckland, New Zealand.

Abstract

We consider simple cube-curves in the orthogonal 3D grid. The union of all cells contained in such a curve (also called the tube of this curve) is a polyhedrally bounded set. The curve's length is defined to be that of the minimum-length polygonal curve (MLP) fully contained and complete in the tube of the curve. So far no provable general algorithm is known for the approximative calculation of such an MLP. This paper presents two approximative algorithms for computing the MLP of a general simple cube-curve in $O(n^4)$ time, where n is the total number of critical edges of the given simple cube-curve.

Keywords: Critical edges, rubber-band algorithm, minimum-length polygon, shortest paths

1 Introduction

The analysis of cube-curves is important for 3D image data analysis (e.g., biomedical imaging), robotics (e.g., 3D path planning), and further areas using 3D regular grids for data representation. A cube-curve is, for example, the result of a digitization process which maps a curve-like object into a union S of face-connected closed cubes. The definition of length of a simple cube-curve in 3D Euclidean space can be based on the calculation of the minimal length polygonal curve (MLP) in a polyhedrally bounded compact set [3, 4].

The computation of the length of a simple cube-curve in 3D Euclidean space was a subject in [5]. But the method may fail for specific curves. [1] presents an algorithm (rubber-band algorithm) which intents to compute an approximating MLP in S, and this algorithm has measured time complexity $O(n)$, where n is the number of grid cubes of the given cube-curve. However, so far no proof is given that the rubber-band-algorithm actually always converges towards the correct MLP, and no proof is known for its worst-case time complexity.

The difficulty of the computation of the MLP in 3D may be illustrated by the fact that the Euclidean shortest path problem (i.e., find a shortest obstacle-avoiding path from source point to target point, for a given finite collection of polyhedral obstacles in 3D space and a given source and a target point) is known to be NP-complete [9]. However, there are some algorithms solving the approximate Euclidean shortest path problem in 3D with polynomial-time, see [10].

Recently, [6] developed an algorithm for calculation of the correct MLP (with proof) for a special class of cube-curves. The main idea is to decompose a cube-curve into arcs by finding "end angles" (see Definition 2 below).

Later, [7] constructed an example of a (special - see title of reference) simple cube-curve, and generalized this by characterizing the class of all of those cube-curves. In particular, it is true that these cube-curves do not have any end angle; and this means that we cannot use the MLP algorithm proposed in [6] which is provable correct. This was the basic importance of the result in [7]: we showed the existence of cube-curves which require further algorithmic studies.

More recently, [8] proved that the rubber-band algorithm is correct for first-class simple cube-curves.

The work of [6], [7] and [8] all focus on a special class of simple cube-curves which are called first-class simple cube-curves (see Definition 3 below). This paper presents two approximate algorithms for computing the MLP for a general simple cube-curve in worst-case time complexity $O(n^4)$, where n is the number of critical edges in the given cube curve.

The paper is organized as follows: Section 2 defines the notations and lists theoretical results used in this paper. Section 3 describes our algorithms and discusses their computational complexity. Section 4 gives the conclusions.

2 Basics

2.1 Definitions

Following [2], a grid point $(i, j, k) \in \mathbb{Z}^3$ 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* as 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 an alternating sequence $g = (f_0, c_0, f_1, c_1, \ldots, f_n, c_n)$ 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 $|i - k| \geq 2 \pmod{n+1}$, if $c_i \bigcap c_k \neq \phi$ then either $|i - k| = 2 \pmod{n+1}$ and $c_i \bigcap c_k$ is an edge, or $|i - k| \geq 3 \pmod{n+1}$ and $c_i \bigcap c_k$ is a vertex.

A *tube* g is the union of all cubes contained in a cube-curve g. A tube is a compact set in $\mathbb{R}^3$, its frontier defines a polyhedron, and it is homeomorphic with a torus in case of a simple cube-curve. A curve in $\mathbb{R}^3$ is *complete* in g iff it has a nonempty intersection with every cube contained in g. Following [3, 4], we define:

Definition 1 *A* minimum-length polygon (MLP) *of a simple cube-curve g is a shortest simple curve P which is contained and complete in tube g. The length of a simple cube-curve g is defined to be the length $l(P)$ of an* MLP *P of g.*

It turns out that such a shortest simple curve P is always a polygonal curve, and it is uniquely defined if the cube-curve is not only contained in a single layer of cubes of the 3D grid (see [3, 4]). If it is contained in one layer, then the MLP is uniquely defined up to a translation orthogonal to that layer. We speak about *the* MLP of a simple cube-curve.

A *critical edge* of a cube-curve g is such a grid edge which is incident with exactly three different cubes contained in g.

Definition 2 *Assume a simple cube-curve g and a triple of consecutive critical edges e_1, e_2, and e_3 such that $e_i \perp e_j$, for all $i, j = 1, 2, 3$ with $i \neq j$. If e_2 is parallel to the x-axis (y-axis, or z-axis) implies that the x-coordinates (y-coordinates, or z-coordinates) of two vertices (i.e., end points) of e_1 and e_3 are equal, then we say that e_1, e_2 and e_3 form an* end angle*, and g* has an end angle*, denoted by $\angle(e_1, e_2, e_3)$; otherwise we say that e_1, e_2 and e_3 form a* middle angle*, and g* has a middle angle.

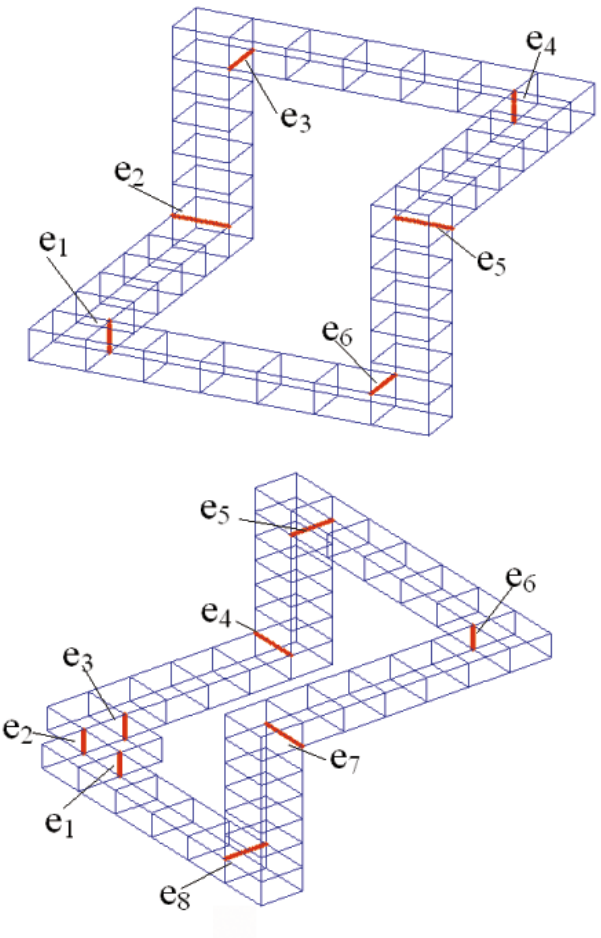

Figure 1: Top: a first-class simple cube-curve. Bottom: a non-first-class simple cube-curve.

Figure 1 (top) shows a simple cube-curve which has four end angles $\angle(e_1, e_2, e_3)$, $\angle(e_3, e_4, e_5)$, $\angle(e_4, e_5, e_6)$, $\angle(e_6, e_1, e_2)$, and two middle angles $\angle(e_2, e_3, e_4)$, $\angle(e_5, e_6, e_1)$.

Definition 3 *A simple cube-curve g is called* first-class *iff each critical edge of g contains exactly one vertex of the MLP of g.*

Figure 1 (top) is also a first-class simple cube-curve; a non-first-class simple cube-curve is shown in Figure 1 (bottom). For the latter one, the vertices of the MLP are in e_1, e_3, e_4, e_5, e_6, e_7 and e_8. In other words, the critical edge e_2 does not contain any vertex of the MLP of this simple cube-curve.

Definition 4 *Let $e_1, e_2, ..., e_n$ be all consecutive critical edges of g. Let $c_1, c_2, ..., c_N$ be all consecutive cubes of g. c_j is called* after *c_i if $j = i + 2$ (mod N).*

Definition 5 *Let e be a critical edge of g. Let c_i, c_j and c_k be three consecutive cubes of g such*

that $e \in c_i, c_j$ and c_k. If c_k is after c_i, then c_i is called the first cube *of e in g, c_k is called* the third cube *of e in g.*

Definition 6 *Let e, f be two critical edges of g. Let c_i be the third cube of e in g, c_j be the first cube of f in g. If the arc $\{c_i, c_{i+1}, ..., c_{j-1}, c_j\}$ (mod N) contains the cube which is after c_i, then $\{c_i, c_{i+1}, ..., c_{j-1}, c_j\}$ (mod N) is called* the arc *between e and f in g.*

The rubber-band algorithm is published in [1, 2], and we do not recall it here. Basically, an initial set of vertices on critical edges is modified in each run of the algorithm such that vertices can be either deleted, or moved into another position on a critical edge (if the resulting curve is of shorter length than the one from the previous iteration step).

2.2 Known Theorems

We recall a few theorems relevant for the discussion of our algorithms:

Theorem 1 ([4]) *A nonflat simple cube-curve in $\mathbb{R}^3$ specifies exactly one minimum-length polygonal simple curve (MLP, minimum-length polygon) which is contained and complete in its tube.*

Theorem 2 ([1]) *Let g be a simple cube-curve. Critical edges are the only possible locations of vertices of the MLP of g.*

Theorem 3 ([11]) *For any directed finite graph G, the all-pairs shortest paths can be computed in $O(n^3(log\ log\ n/log\, n)^{5/7})$ time, where n is the number of vertices of G.*

Theorem 4 ([8]) *The rubber-band algorithm is correct for first-class simple cube-curves. The computational complexity is $O(n)$, where n is the number of critical edges of the simple cube-curve.*

2.3 New Theorems

Let a, b, c, d, e and $f \in \mathbb{R}^3$ with $a \leq b, c \leq d$ and $e \leq f$. Let

$$f(x, y, z) = \sqrt{x^2 + y^2 + z^2}$$

where $(x, y, z) \in [a, b] \times [c, d] \times [e, f]$. If there is a real number $M > 0$ such that $|f(x_1, y_1, z_1)| \geq M$, where $(x_1, y_1, z_1) \in [a, b] \times [c, d] \times [e, f]$, then we have the following:

Lemma 1 *there is a real number $\delta > 0$ such that for each point $(x_2, y_2, z_2) \in [a, b] \times [c, d] \times [e, f]$ with $|x_2 - x_1| < \delta, |y_2 - y_1| < \delta, |z_2 - z_1| < \delta$, and $|f(x_2, y_2, z_2)| > 3M/2$.*

Proof: Since $f(x, y, z)$ is uniformly continuous at each point of $[a, b] \times [c, d] \times [e, f]$, so for $\varepsilon_0 = M/2 > 0$, there is a real number $\delta > 0$ such that for any two points $(x_1, y_1, z_1), (x_2, y_2, z_2) \in [a, b] \times [c, d] \times [e, f]$ with $|x_2 - x_1| < \delta, |y_2 - y_1| < \delta$ and $|z_2 - z_1| < \delta$, we have

$$|f(x_2, y_2, z_2) - f(x_1, y_1, z_1)| < M/2$$

That is, $f(x_1, y_1, z_1)| - M/2 < f(x_2, y_2, z_2) < f(x_1, y_1, z_1)| + M/2$. Since $|f(x_1, y_1, z_1)| \geq M$ implies $-f(x_1, y_1, z_1) \geq -M$ or $-f(x_1, y_1, z_1) \leq M$. Therefore, $f(x_2, y_2, z_2) > -M - M/2 = -3M/2$ or $|f(x_2, y_2, z_2)| < M + M/2 = 3M/2$. That is $|f(x_2, y_2, z_2)| > 3M/2$. [End of proof]

Let $M_1 = \max\{a, b, c, d, e, f\} > 0$. If $|f(x_1, y_1, z_1)| \geq M_2$, then for any two points (x_1, y_1, z_1) and $(x_2, y_2, z_2) \in [a, b] \times [c, d] \times [e, f]$ with $|x_2 - x_1| < \delta, |y_2 - y_1| < \delta$ and $|z_2 - z_1| < \delta$, we have the following:

Lemma 2 $|f(x_2, y_2, z_2) - f(x_1, y_1, z_1)| < 2M_1\delta/M_2$.

Proof: Without loss of generality, suppose that $x_1 < x_2, y_1 < y_2$ and $z_1 < z_2$. By the mean-value theorem, $|f(x_2, y_2, z_2) - f(x_1, y_1, z_1)| = |[f(x_2, y_2, z_2) - f(x_1, y_2, z_2)] + [f(x_1, y_2, z_2) - f(x_1, y_1, z_2)] + [f(x_1, y_1, z_2) - f(x_1, y_1, z_1)]| = |f_x(\xi_x, y_2, z_2)(x_2 - x_1) + f_y(x_1, \eta_y, z_2)(y_2 - y_1) + f_z(x_1, y_1, \zeta_z)(z_2 - z_1)| =$

$$\left| \frac{\xi_x}{\sqrt{{\xi_x}^2 + {y_2}^2 + {z_2}^2}}(x_2 - x_1) + \frac{\eta_y}{\sqrt{{x_1}^2 + {\eta_y}^2 + {z_2}^2}}(y_2 - y_1) + \frac{\zeta_z}{\sqrt{{x_1}^2 + {y_1}^2 + {\zeta_z}^2}}(z_2 - z_1) \right|$$

where $\xi_x \in (x_1, x_2), \eta_y \in (y_1, y_2)$ and $\zeta_z \in (z_1, z_2)$. By Lemma 1, we have $|f(x_2, y_2, z_2) - f(x_1, y_1, z_1)| < 2M_1\delta/(3M_2) + 2M_1\delta/(3M_2) + 2M_1\delta/(3M_2) = 2M_1\delta/M_2$. [End of proof]

Let $e_1, e_2, ..., e_n$ be all the consecutive critical edges of g. Let the points $p_i(x_i, y_i, z_i), p'_i(x'_i, y'_i, z'_i) \in e_i$, with $|x'_i - x_i| < \delta, |y'_i - y_i| < \delta$ and $|z'_i - z_i| < \delta$, where $i = 1, 2, ..., n$. Let

$$d_e(i) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}$$

$$d'_e(i) = \sqrt{(x'_{i+1} - x'_i)^2 + (y'_{i+1} - y'_i)^2 + (z'_{i+1} - z'_i)^2}$$

where $i = 1, 2, ..., n-1$,

$$d_e(n) = \sqrt{(x_1 - x_n)^2 + (y_1 - y_n)^2 + (z_1 - z_n)^2}$$

and

$$d'_e(n) = \sqrt{(x'_1 - x'_n)^2 + (y'_1 - y'_n)^2 + (z'_1 - z'_n)^2}$$

Let $d = \sum_{i=1}^{n} d_e(i)$ and $d' = \sum_{i=1}^{n} d'_e(i)$. Let $M_1 = \max\{a, b, c, d, e, f\} > 0$. By Lemma 2, we have the following:

Theorem 5 *If $|d_e(i)| \geq M_2$, where $i = 1, 2, ..., n$, then*

$$|d - d'| < 4nM_1\delta/M_2$$

Proof: Let $\delta_{x_{i+1}} = x_{i+1} - x'_{i+1}, \delta_{x_i} = x'_i - x_i, \delta_{y_{i+1}} = y_{i+1} - y'_{i+1}, \delta_{y_i} = y'_i - y_i, \delta_{z_{i+1}} = z_{i+1} - z'_{i+1}, \delta_{z_i} = z'_i - z_i, \delta_x = \delta_{x_{i+1}} + \delta_{x_i}, \delta_y = \delta_{y_{i+1}} + \delta_{y_i}$, and $\delta_z = \delta_{z_{i+1}} + \delta_{z_i}$. Then we have

$$|\delta_x| \leq 2\delta, \ |\delta_y| \leq 2\delta, \text{ and } |\delta_z| \leq 2\delta$$

Then $(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2 = [(x'_{i+1} - x'_i) + (x_{i+1} - x'_{i+1}) + (x'_i - x_i)]^2 + [(y'_{i+1} - y'_i) + (y_{i+1} - y'_{i+1}) + (y'_i - y_i)]^2 + [(z'_{i+1} - z'_i) + (z_{i+1} - z'_{i+1}) + (z'_i - z_i)]^2 = [(x'_{i+1} - x'_i) + \delta_x]^2 + [(y'_{i+1} - y'_i) + \delta_y]^2 + [(z'_{i+1} - z'_i) + \delta_z]^2$. So, by Lemma 2, we have

$$|d_e(i) - d'_e(i)| < 2M_1 \times 2\delta/M_2 = 4M_1\delta/M_2$$

Therefore, $|d - d'| = |\sum_{i=1}^{n}(d_e(i) - d'_e(i))| \leq \sum_{i=1}^{n} |d_e(i) - d'_e(i))| < 4nM_1\delta/M_2$. [End of proof]

3 Algorithms and Their Complexity

3.1 Algorithms

Based on Theorems 1 and 2, we are now in a position to formulate the following algorithms.

3.1.1 Algorithm 1

1. Initialize an integer $m \geq 2$. For each critical edge e_i, let $P_{i_1}, P_{i_2}, ..., P_{i_{m+1}} \in e_i$ such that the Euclidean distance between P_{i_j} and $P_{i_{j+1}}$ is $\frac{1}{m}$, where $j = 1, 2, ..., m$ (P_{i_1} and $P_{i_{m+1}}$ are the end points of e_i).

2. Construct a weighted directed graph $G = [V, E]$, where

$V = \{P_{i_j} : P_{i_j} \in e_i, i = 1, 2, ..., n, j = 1, 2, ..., m\}$,

$E = \{P_{i_j}P_{k_l} : P_{i_j}P_{k_l}$ is completely contained in the arc between e_i and e_k in g $\}$.

The weight of $P_{i_j}P_{k_l}$ is defined as the Euclidean distance between P_{i_j} and P_{k_l}, where $i = 1, 2, ..., n, j = 1, 2, ..., m$.

3. For each $v \in V$, let $S(v) = \{u : uv \in V\}$, and do:

3.1. For each $u \in S(v)$, apply Dijkstra's algorithm to compute the directed shortest path between u and v. From this, find the local minimum-weight directed cycle which contains uv.

4. For each $v \in V$, do:

4.1. For each $u \in S(v)$, apply Dijkstra's algorithm to compute the directed shortest path between u and v. From this, find the local minimum-weight directed cycle which contains uv.

5. Compare the local minimum-weight directed cycle for each $v \in V$, the minimum one is the global minimum-weight directed cycle.

3.1.2 Algorithm 2

Steps 1 and 2 are the same as those in Algorithm 1.

3.1. Like step 3 in Algorithm 1, but now: for each $u \in S(v)$, find the local minimum-weight directed cycle which contains uv.

3.2. Let CE_1 be the set of critical edges such that each of the vertices of the local minimum-weight directed cycle is on one of CE_1.

3.3. Let CE_2 be the set of critical edges such that each of the vertices of the local second minimum-weight directed cycle is on one of CE_2, and CE_2 is not equal to CE_1.

4.1. For each $v \in V$, compute the local minimum-weight directed cycle which contains v and the associated set of critical edges $CE_1(v)$.

4.2. For each $v \in V$, compute the local directed cycle whose length comes next to the minimum-length cycle, and which contains v, and the associated set of critical edges $CE_2(v)$.

4.3. Compare the local minimum-weight directed cycle for each $v \in V$, the minimum one is the global minimum-weight directed cycle, denoted by $AMLP_1$. Compute also the associated set of critical edges CE_1.

5. Compare all local directed cycles whose length comes next to the minimum-length cycle, for each $v \in V$; the result is a global *second minimum-weight directed cycle*, denoted by $AMLP_2$. Also compute the associated set of critical edges CE_2.

6. By Theorem 4, apply the rubber-band algorithm on CE_1, and compute an approximate minimum-weight directed cycle.

N	**n**	**m**	**time**	$AMLP_1$	CE_1	$AMLP_2$	CE_2
28	10	3	4.1090	17.1580	1,3-5,7-10	17.3229	1,3,4,6-10
20	12	3	8.1090	17.8533	1-3,5,6,9,10,12	17.8942	1,2,4-6,9,10,12
64	17	3	21.7490	50.0322	2, 5-7, 9, 11,14-17	50.0740	2, 4, 6,7, 9, 11,14-17
82	28	5	386.8290	59.1651	1,3-7,9,13-20,23-26,28	59.1697	1,3-8,13-20,23-26,28

Table 1: Computational complexity for Algorithm 2, where N is the number of cubes, n is the number of critical edges, m is the number of subdivision points on each critical edge; the time is in seconds.

3.2 Example and Time Complexity

Figures 2 to 4 show three examples of randomly generated simple cube curves, and corresponding graphs G providing the discrete input data for Algorithm 2 for selecting the approximate minimum-length polygon.

For Algorithm 1, the first three steps can be pre-processed. The main computation occurs in step 3.1, which is in $O((mn)^2)$, where n is the number of critical edges and m is the number of subdivision points on each critical edge. In step 3.1, $|S(v)| \leq mn$. So the time complexity of step 4 is $O((mn)^3)$. Since $|V| \leq mn$, so the time complexity of step 5 is $O((mn)^4)$.

For Algorithm 2, the time complexity is $O((mn)^4)$ as well.

It is clear that Algorithm 2 is slower than Algorithm 1. However, Algorithm 2 will find correct approximate MLP if m is sufficiently large while Algorithm 1 may fail to do so.

By Theorem 3, the time complexity of Algorithms 1 and 2 may be reduced to

$$O(n^3\ (log\ log\ n/log\ n)^{5/7})$$

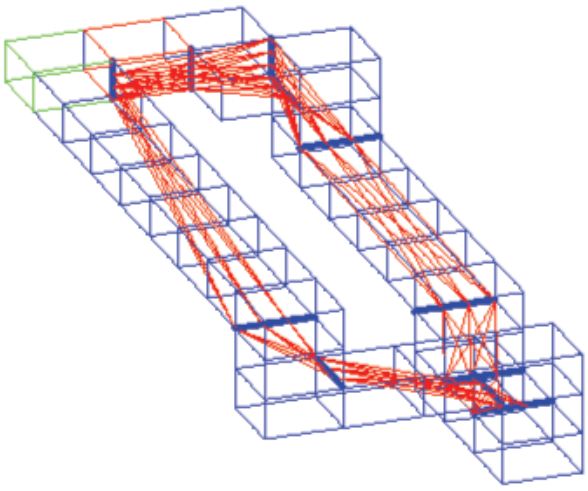

Figure 2: A randomly generated cube curve and the original graph G before starting Algorithm 2.

4 Conclusions

Given a simple cube-curve, there is only a finite number of critical edges. So there are always only a finite number of edges in CE_1 or CE_2 in Algorithm 2. Therefore there is a sufficiently large value of m (the number of subdivision points) such that for any CE which is not equal to CE_1, the weight of the global minimum-weight directed cycle ob-

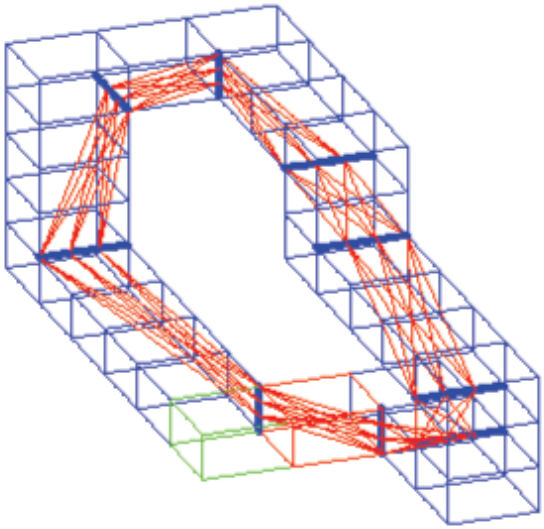

Figure 3: A randomly generated cube curve and the original graph G before starting Algorithm 2.

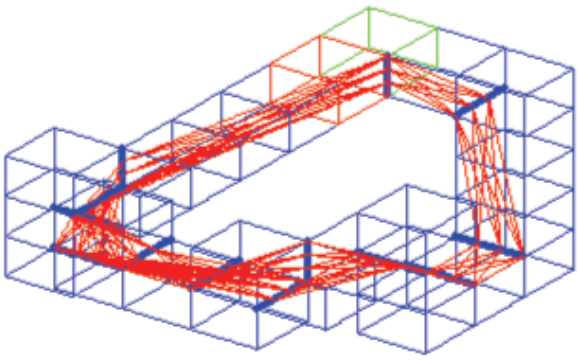

Figure 4: A randomly generated cube curve and the original graph G before starting Algorithm 2.

tained by applying the rubber-band algorithm on CE will be always greater than that of the global minimum-weight directed cycle obtained by applying the rubber-band algorithm on CE_1.

This implies that, mathematically, we can take a sufficiently large value of m to obtain the correct MLP of g. However, in practice, our experiments show that when m is 5, n is 28, running time will be about 386.8290 seconds (see Table 3.1.1) on a Pentium 4 PC using Matlab 7.04.

By Theorem 5, if better bounds than M_1 and M_2 can be found, then also a smaller value of m would result.

We have presented two approximate algorithms for computing the MLP of a general simple cube-curve in time $O(n^4)$, this may be reduced to $O(n^3(log\ log\ n/log\, n)^{5/7})$, where n is the number of its critical edges. Clearly, these algorithms can be used to study whether the rubber-band algorithm is correct for general simple cube-curves. So far, no counter-example has been found.

References

[1] T. Bülow and R. Klette, "Digital curves in 3D space and a linear-time length estimation algorithm," *IEEE Trans. Pattern Analysis Machine Intelligence*, no. 24, pp. 962–970, 2002.

[2] R. Klette and A. Rosenfeld. *Digital Geometry: Geometric Methods for Digital Picture Analysis.* Morgan Kaufmann, San Francisco, 2004.

[3] F. Sloboda, B. Zaťko, and R. Klette, "On the topology of grid continua," SPIE *Vision Geometry VII,* **3454**: 52–63, 1998.

[4] F. Sloboda, B. Zaťko, and J. Stoer, "On approximation of planar one-dimensional grid continua," In R. Klette, A. Rosenfeld, and F. Sloboda, editors, *Advances in Digital and Computational Geometry*, pp. 113–160. Springer, Singapore, 1998.

[5] A. Jonas and N. Kiryati, "Length estimation in 3-D using cube quantization," *J. Math. Imaging and Vision*, no 8, pp. 215–238, 1998.

[6] F. Li and R. Klette, "Minimum-length polygon of a simple cube-curve in 3D space," *In Proceedings IWCIA 2004*, LNCS3322: pp. 502–511, 2004.

[7] F. Li and R. Klette, "The class of simple cube-curves whose MLPs cannot have vertices at grid points," *In Proceedings DGCI 2005*, LNCS3429: pp. 183–194, 2005.

[8] F. Li and R. Klette, "Minimum-Length Polygons of First-Class Simple Cube-Curves," *In Proceedings CAIP 2005*, LNCS3691: pp. 321–329, 2005.

[9] J. Canny and J.H. Reif, "New lower bound techniques for robot motion planning problems," *Proc. IEEE Conf. Foundations Computer Science*, pp. 49–60, 1987.

[10] J. Choi, J. Sellen, and C.-K. Yap, "Approximate Euclidean shortest path in 3-space," *Proc. ACM Conf. Computational Geometry*, ACM Press, pp. 41–48, 1994.

[11] dynamics and pp. 1–42, 1991. Y. Han, "Improved algorithm for all pairs shortest paths," *Information Processing Letters*, no. 91, pp. 245–250, 2004.

[12] R. Seidel, "On the all-pairs-shortest-path problem," *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pp. 745–749, 1992.

[13] R. Seidel, "On the All-Pairs-Shortest-Path Problem in Unweighted Undirected Graphs," *Journal of Computer and System Science*, no. 51, pp. 400–403, 1995.