

The Curve Equipartition Problem

Costas Panagiotakis, George Georgakopoulos and George Tziritas

Computer Science Department, University of Crete, Greece, P.O. Box 2208

Abstract

In this paper, we analyze the general problem of a continuous curve partition into equal length segments, defined by any smooth distance metric, like the Euclidean distance. The goal is to locate $N - 1$ consecutive curve points, so that the curve can be divided into N equal length segments. First, it is proved that for any curve C and any number N there always exists at least one N -equipartition (EP) of C . The approach adopted for the proof can be used to compute at least one solution of the problem or all the solutions using a greedy version of this method. These methods have only theoretical value, as there appear algebraic expressions exponentially with increasing degree N . This can be explained by our proof that a version of the EP problem is **NP**-hard. So, the EP is possibly complete for the NP class. However, an $O(N)$ approximate proof based algorithm can solve the problem with high accuracy. Moreover, we propose two algorithms, an algorithm that computes approximately all the solutions and a steepest descent based method that converges to one of them. Some possible applications are also discussed.

1 Introduction

The curve segmentation problem is a challenging problem of computational geometry. A huge number of applications, like object recognition and tracking, signal summarization and compression, curve simplification and computer graphics applications, are based on curve segmentation. The main goal of these methods is to solve a curve segmentation problem into “homogenous” segments with uniform characteristics or to minimize a predefined error. The signal segmentation into uniform segments is applied mainly in the classification problem. Automatic sound [1] and image [2] segmentation and categorization can be solved using a suitable signal description and low cost segmentation algorithms. Many computer graphics applications are based on curve segmentation problem, like surface simplification and 3D modelling. Most polygonal surface simplification methods employ triangles as their approximating elements when constructing a surface [3]. One of the most popular triangulation

Email address: {cpanag, ggeo, tziritas}@csd.uoc.gr (Costas Panagiotakis, George Georgakopoulos and George Tziritas).

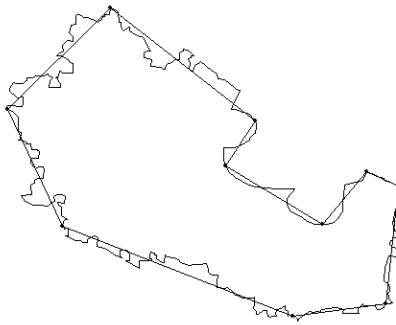


Fig. 1. A polygonal approximation with ten segments.

methods is Delaunay triangulation [4], [5]. In addition, signal compression can be achieved using a surface simplification schema. In [6] it is proposed a technique for compressing animations using triangulation and simplification.

On computer vision applications the curve segmentation problem also appears. Signal summarization and key frames detection methods using an appropriate feature set reduce the initial problem into a curve segmentation problem [7], [8]. Methods for non articulated motion tracking are based on solutions of the curve equipartition problem [9].

Another example of such segmentation approach is the 2D or 3D polygonal approximation [10], [11], [12] or convex polygons [13]. This problem asks for computing another polygonal curve that approximates the original curve (Fig. 1). There are two well-known approaches for solving the problem: graph-theoretical [14] and dynamic programming [15]. These approaches require extremely high execution time and memory when the given polygon size is getting high. Thus in a lot of applications, the main objective is not to compute the optimal solution, but to find low computation cost algorithms that will give a sub-optimal solution [16], [17]. The problem can be formulated in two ways [18], [19].

- The problem of minimum error ($Min - \varepsilon$), where the approximation error is minimized given the number of segments.
- The problem of minimum number of segments ($Min - \#$), where the approximation error is known and the goal is to find the minimum number of segments that gives error lower than the given error.

One of the most widely used high-quality curve simplification algorithms is the heuristic method commonly called the Douglas-Peucker approximate algorithm [20]. The performance of polygonal approximation algorithms can be measured under variations in scale parameters and data [21]. In addition, shape representation [22] by polygonal approximation has become a popular technique for constructing a characteristic description of a boundary in the form

Symbols	Definitions
$C(t), A, B$	The given curve, $t \in [0, 1]$. $A = C(0)$, $B = C(1)$.
N	The number of equal length segments
$d(x, y)$	Euclidean distance of $C(x), C(y)$, $d(x, y) = C(x) - C(y) _2$
$O(u, r)$	Cycle with center $C(u)$ and radius r
$U_{M \dots K}$	$\{M, M + 1, \dots, K\}$
$G(u, N)$	$G(u, N) = \{t_i, i \in U_{0 \dots N} \wedge t_0 = 0 \leq t_1 \leq \dots \leq t_N = u \wedge d(t_{i-1}, t_i) = d(t_i, t_{i+1})\}$ $C(t)$, $t \in [0, u]$ is segmented into N equal length segments

Table 1
Proof symbol table.

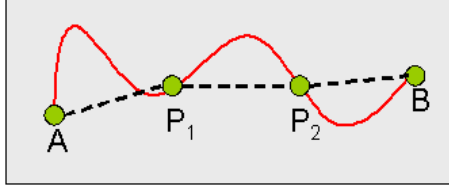


Fig. 2. A curve segmentation example for $N = 3$, $|AP_1| = |P_1P_2| = |P_2B|$.

of a sequence of straight lines. There are two main approaches to the problem: (1) fitting the curve with a sequence of line segments and (2) finding a subset of dominant points as vertices of the approximating polygon.

The rest of the paper is organized as follows: Section 2 presents the definition of EP. The proof of solution existence for each segment number is presented in Section 3. Section 4 describes the proof that a version of the EP problem is **NP**-hard. The proposed algorithms that solve the problem are presented in Sections 5 and 6. Finally, conclusions and discussion are provided in Section 7.

2 Problem Definition

The equipartition problem is defined as follows: Let $C(t), t \in [0, 1]$ be a 2D continuous, acyclic curve¹ that starts on $A = C(0)$ and ends on $B = C(1)$. We have to compute $N - 1$ sequential curve points $P_i, i \in U_{1 \dots N-1}$, $P_0 = A$, $P_N = B$ under the constraint $d(P_{i-1}, P_i) = d(P_i, P_{i+1})$, $i \in U_{1 \dots N-1}$. The problem is the curve partitioning into N equal length segments, so that the first starts from A and the last ends on B (Fig. 2). Some useful symbols are defined on Table 1.

¹ We suppose that the curve is partially algebraic. The problem can be defined with the same way in any dimension ($C(t) \in \mathbb{R}^n$).

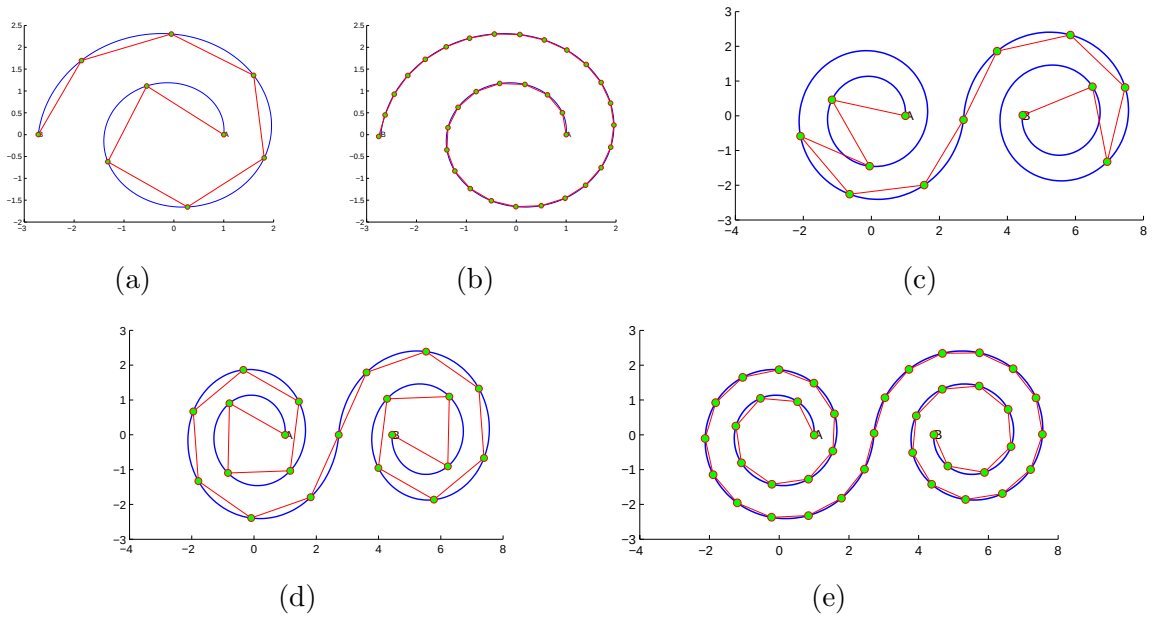


Fig. 3. Curves segmentation examples for different N . The higher is N , the better is curve approximation.

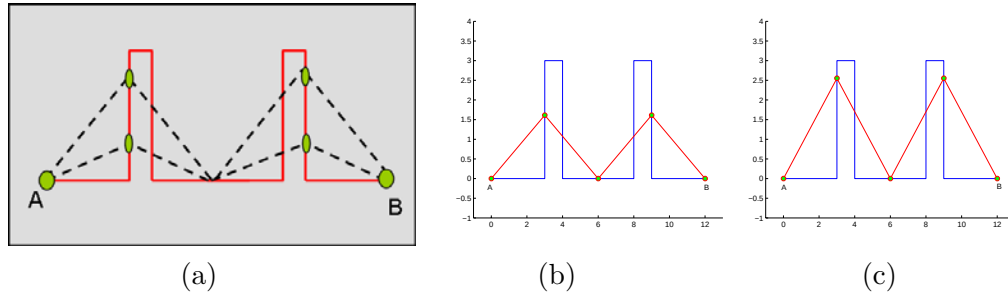


Fig. 4. A curve example where there are infinite number of solutions for $N = 4$. On figures (b), (c) are shown two of them.

The solution is obvious for $N = 1$, as we have one segment, AB . When $N = 2$, we have to locate a curve point P , so that $|AP| = |PB|$. This point can be given as the intersection of the curve with the AB segment bisector. This section always exists, because the curve is continuous and the AB segment bisector separates the plane into two semiplanes F_1, F_2 , with $A \in F_1, B \in F_2$. When N is higher than two, there is not a trivial method to compute the equal length line segments. The above problem can have more than one solutions depending on curve shape and the value of N . As N tends to infinity the problem solution (equal length segments) will be unique and it will approximate the curve (Fig. 3). However, there is not any guarantee that the approximation with $N + 1$ segments will be better than the approximation with N . In addition, we can define the $(Min-\varepsilon)$ and the $(Min-\#)$ problems like the polygonal approximation problems. The number of solutions for different values of N depends on the curve and N . A special case is the linear segment curve, where for each N , there is just one solution. There are special curves, where the number of solutions for some N is infinite (Fig. 4). More examples can be found on [23]. Concerning the problem

of existence of an efficient (*i.e.* polynomial) algorithm that given C and NN constructs an N -equipartition of C , we have proved on lemma 5 that a version of the EP problem is **NP**-hard. So, according to our current state of knowledge, the EP is possibly complete for the NP class.

2.1 A Useful Property

The problem has the following important property. If we have segmented the AD into N equal length segments of length r , then we can segment the $A\dot{D}$ into $N + 1$ equal length segments of length r , if and only if $|D\dot{D}| = r$. So, we can suppose the existence of $G(u, N)$ (see Table 1) and we have to compute the $G(1, N + 1)$. The points $\dot{D}$ are defined as D is moving along from point A to point B . The question is, whether these points will cover the total curve points. Let $g_N(s) \in [0, 1]$, $s \in [0, S_N]$, $g_N(0) = 0$, $g_N(S_N) = 1$ be a continuous path function and $r_N(s) \geq 0$, $s \in [0, S_N]$ be a continuous length function. We set $g_1(s) = s$, $s \in [0, S_1]$, $S_1 = 1$ and $r_1(s) = d(0, s)$. Using the following lemma 1, the proof of existence of at least one solution for each N is equivalent to the proof that for each N there exist a continuous path function $g_N(s)$ and a continuous length function $r_N(s)$.

Lemma 1 *If there exist a continuous path function $g_N(s)$ and a continuous length function $r_N(s)$, then $\forall v \in (0, 1]$ there exist at least one $G(v, N + 1)$.*

Proof: $g_N(s)$ is continuous path function $[0, S_N]$, $g_N(0) = 0$, $g_N(S_N) = 1$. Using Bolzano theorem, $\forall v \in (0, 1]$, $\exists s_1 \in [0, S_N]$, so that $g(s_1) = v$.

Let $\Psi(s) = d(g_N(s), v)$, $s \in [0, s_1]$ be a continuous function. Then, it is derived that:

- $\Psi(0) = d(0, v) > 0$, $\Psi(s_1) = d(g_N(s_1), v) = d(v, v) = 0$, $0 < v \leq 1$.
- $r(0) = 0$ and $r_N(s_1) > 0$.

We can use the Bolzano theorem on $[0, s_1]$ to show that there exist at least one point $s_0 \in (0, s_1)$, so that $\Psi(s_0) = r_N(s_0)$. Therefore there exist at least one $u_0 = g_N(s_0)$, so that the curve $C(t)$, $t \in [0, u_0]$ is equipartitioned into N segments ($G(u_0, N)$) of length $r_N(s_0)$. So, there exist at least one equal length partition $G(v, N + 1) = G(u_0, N) \cup \{v\}$, because $r_N(s_0) = d(u_0, v)$ and $u_0 < v$.

The construction of $r_N(s)$, $g_N(s)$ can be done progressively using the $g_{N-1}(s)$ and $r_{N-1}(s)$. The construction algorithm and the proof of existence, that are based on this definition,

have many special cases and they are associated to the Euclidean distance metric. On the next section, we are going to give an equivalent problem definition, which is more general and it will result in an elegant construction algorithm and a compact existence proof.

2.2 An Equivalent Problem Definition

The smooth function $d(x, y) = |C(x) - C(y)|$, $x, y \in [0, 1]$ gives a different view and an equivalent definition of the problem. We can use as $d(x, y)$ any smooth metric² like Euclidean distance. As a smooth metric distance, $d(x, y)$ is characterized by the following properties:

- (1) $d(x, y) = 0 \Leftrightarrow x = y$ (isolation).
- (2) $d(x, y) = d(y, x)$ (symmetry).
- (3) $|d(x, w) - d(w, y)| \leq d(x, y) \leq d(x, w) + d(w, y)$ (triangular inequality).
- (4) $d(x, y)$ inherits continuity and derivative existence from the curve $C(t)$.
- (5) $d(x, y)$ can be defined in any dimension ($C(t) \in \mathbb{R}^n$).

An equivalent problem definition can be derived by the using of $d(x, y)$. A problem solution $G(1, N) = \{0, t_1, \dots, t_{N-1}, 1\}$ of curve $C(t)$, corresponds to the surface $d(x, y)$ as a point sequence, $(0, t_1), (t_1, t_2), (t_2, t_3), \dots, (t_{N-1}, 1)$. The length r of each segment is given by the following equation:

$$r = d(0, t_1) = d(t_1, t_2) = d(t_2, t_3) = \dots = d(t_{N-1}, 1) \quad (1)$$

An alternative problem definition will be the determination of $\{t_1, \dots, t_{N-1}\}$, so that Equation (1) will be satisfied. This definition is used in Section 3 in order to prove that the problem has at least one solution.

3 General Existence Proof

In this section we are going to prove that there exists at least one solution for each N . We suppose that the given curve $C(t) \in \mathbb{R}^n$ is continuous, acyclic and partially monotonous and $d(x, y)$ is a smooth distance metric like Euclidean distance metric.

² In addition, smooth semimetric distance functions can be used, which do not satisfy the triangular inequality property.

As it is derived by the proof, each function of form

$$f(x, y) = d(x, y) - d(x, Y(x)) \quad \text{or} \quad f(x, y) = d(x, y) - d(X(y), y)$$

should be partially monotonous, where $X(y)$, $Y(x)$ are partially monotonous functions of the same derivative degree with $d(x, y)$. The above statement is always true for partially algebraic curves (they are partially monotonous). Therefore all the relations will give partially algebraic representations. For this reason we can suppose that $C(t) \in \mathbb{R}^n$ is a continuous and partially algebraic curve.

Firstly, we are going to show that the problem has always solution for $N = 3$ and then we will apply inductively the proof method for higher N .

3.1 Existence Proof for $N = 3$

The function $f_2(x, y)$, which is defined by the following equation (2), is continuous and partially monotonous function,

$$f_2(x, y) = d(x, y) - d(x, 0), \quad x \in [0, 1], y \geq x. \quad (2)$$

The null plane curves³ of $f_2(x, y)$ will be continuous and partially monotonous. The total solutions of equal length segments for $N = 2$ ($G(y, 2)$) are given by the points (x, y) of these curves, because $d(x, 0) = d(x, y)$ and $y \geq x$. In the following lemma 2 we will show that $f_2(x, y) = 0$ does not contain any region, consequently we can refer to curves. By the term region, we mean that $\exists x_1, x_2, y_1, y_2, (x_1 < x_2), (y_1 < y_2)$, so that $\forall x \in [x_1, x_2]$ and $\forall y \in [y_1, y_2] \Rightarrow f_2(x, y) = 0$.

Lemma 2 *It is impossible for a part of $f_2(x, y)$ null plane to be a region.*

Proof: *If the above statement was false, then:*

- *Let (x_0, y_0) be an intrinsic point of the region, then $\exists \varepsilon > 0$ so that $\forall x \in [x_0 - \varepsilon, x_0 + \varepsilon]$ and $y \in [y_0 - \varepsilon, y_0 + \varepsilon] \Rightarrow d(x, 0) = d(x, y)$. This means that $|C(y_0) - C(x_0)| = |C(y) - C(x_0)|$, $\forall y \in [y_0 - \varepsilon, y_0 + \varepsilon]$. So, $\forall y \in [y_0 - \varepsilon, y_0 + \varepsilon]$, the points $C(y)$ are points of a circular arc of center $C(x_0)$.*
- *It is true that $|C(y_0) - C(x_0 + \frac{\varepsilon}{2})| = |C(y) - C(x_0 + \frac{\varepsilon}{2})|$, $\forall y \in [y_0 - \varepsilon, y_0 + \varepsilon]$. So,*

³ The null plane curves of $f(x, y)$ are defined by the equation $f(x, y) = 0$.

$\forall y \in [y_0 - \varepsilon, y_0 + \varepsilon]$, the points $C(y)$, are points of a circular arc of center $C(x_0 + \frac{\varepsilon}{2})$.

This would mean that $C(x_0 + \frac{\varepsilon}{2}) = C(x_0)$, which is false as $C(t)$ does not contain cycles.

In the cases (not Euclidean distances), where it is possible to exist such regions, it can be defined a curve inside the region that it is connected with exterior curve $h(s)$. In addition, the final curve will satisfy the conditions of being partially monotonous and continuous.

Let $h_2(s) = [a_2(s), b_2(s)]$, $s \in [0, 1]$ be the curve of $f_2(x, y)$ null plane, that starts from $[0, 0]$ ($h_2(0) = [0, 0]$). Then $a_2(s) \leq b_2(s)$, because the points of $h_2(s)$ are points of f_2 domain. This curve exists as $f_2(0, 0) = 0$. The following lemma proves that $h_2(s)$ ends on $y = 1$ ($b_2(1) = 1$).

Lemma 3 *The curve $h_2(s)$ ends on $y = 1$, equivalently $b_2(1) = 1$.*

Proof:

- The $C(t)$, $t \in [0, u]$ has a unique solution when u is close to 0. This means that $h_2(s)$ is not a closed curve on $[0, 0]$ (if it was closed, it would have two solutions).
- The curve $h_2(s)$ will intersect with the diagonal axis $x = y$ or the $x = 0$ axis just at the origin $(0, 0)$, because

$$f_2(x, x) = d(x, x) - d(x, 0) = -d(x, 0) < 0, \forall x > 0$$

and

$$f_2(0, y) = d(0, y) - d(0, 0) = d(0, y) > 0, \forall y > 0.$$

This also means that $f_2(x, y)$ changes its sign close to $(0, 0)$, at both sides of $h_2(s)$.

We are going to show that $h_2(s)$ is impossible to end somewhere inside the $(0, 1)^2$. Let that $h_2(s)$ ends on $(x_0, y_0) \in (0, 1)^2 \Rightarrow \exists E > 0$ (small) so that $\forall \varepsilon, E > \varepsilon > 0$ the cycles $((x_0, y_0), \varepsilon) \in [0, 1]^2$ have just one intersection point with the curve $h_2(s)$. We consider the function

$$f_o(w) = f_2(x_0 + \varepsilon \cos(w), y_0 + \varepsilon \sin(w)), \quad w \in [0, 2\pi), \text{ restriction of } f_2 \text{ on cycle } ((x_0, y_0), \varepsilon).$$

There exist points of cycle $(x_1, y_1), (x_2, y_2)$, $x_1 = x_0 + \varepsilon \cos(w_1), y_1 = y_0 + \varepsilon \sin(w_1), x_2 = x_0 + \varepsilon \cos(w_2), y_2 = y_0 + \varepsilon \sin(w_2)$, $f_o(w_1) = f_2(x_1, y_1) \geq 0$ and $f_o(w_2) = f_2(x_2, y_2) < 0$, because $f_2(x_0, y_0) = 0$ and $f_2(x, y)$ is continuous. Using the Bolzano theorem, as $f(w)$ is continuous, then on cycle there exists at least two points $(x_3, y_3), (x_4, y_4)$, so that $f_2(x_3, y_3) =$

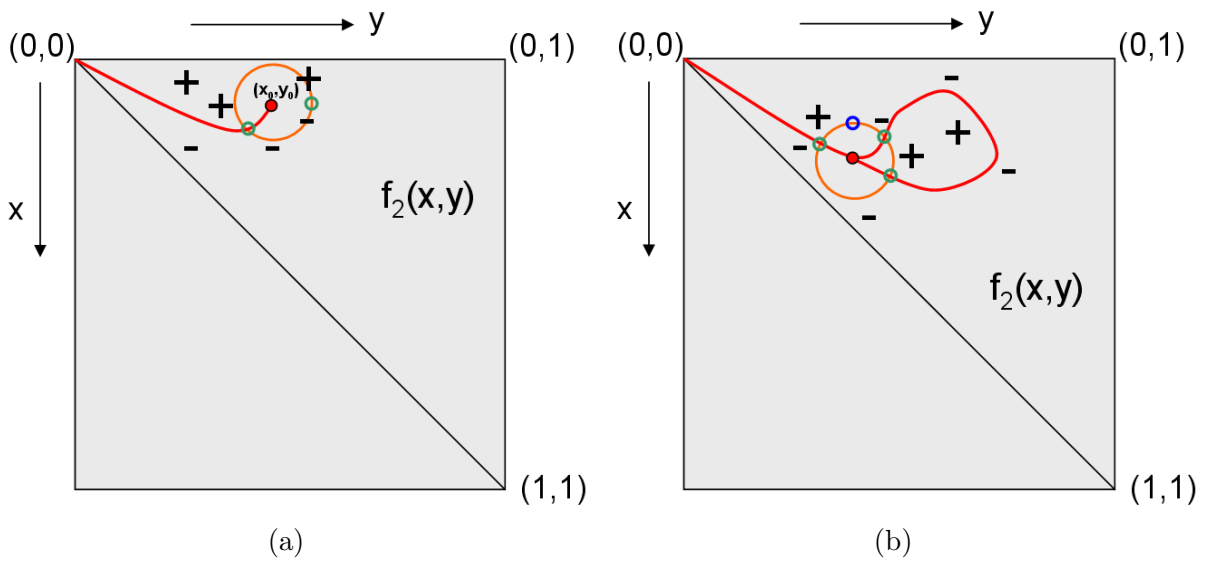


Fig. 5. **(a),(b)** The signs of $f_2(x,y)$ under the $h_2(s)$ (red line). **(a)** $g_2(s)$ can not end on $(x_0, y_0) \in (0,1)^2$, because it should have at least two intersection points (green spots) with the cycle $((x_0, y_0), \varepsilon)$. **(b)** $g_2(s)$ can not be in a loop, because it should have at least four intersection points with the cycle $((x_0, y_0), \varepsilon)$.

0, $f_2(x_4, y_4) = 0$ (Fig. 5(a)). From their definition, these points should belong in $h_2(s)$. This is impossible, because we have supposed that the cycle has just one intersection point with the curve $h_2(s)$.

We have proved in lemma 2 that $h_2(s)$ has not intersection points with the diagonal ($x = y$) and the line ($x = 0$), so $h_2(s)$ does not end on them. It is remained to show that $h_2(s)$ can not be in a loop, in order to prove that $h_2(s)$ ends on $y = 1$. This can be proved as illustrated in Figure 5(b).

We consider the continuous function $q(s)$ (Equation (3)). Using this function, we will find $\{t_1, t_2\}$ with $t_2 \geq t_1$ satisfying Equation (2) and the proposition will have been proved for $N = 3$.

$$q(s) = d(a_2(s), b_2(s)) - d(1, b_2(s)), \quad s \in [0, 1] \quad (3)$$

It holds that,

- $q(0) = d(0, 0) - d(1, 0) = -d(1, 0) < 0$
- $q(1) = d(a_2(s), 1) - d(1, 1) = d(a_2(s), 1) > 0$

At least a $s_2 \in (0, 1)$ exists (applying the Bolzano theorem) so that $q(s_2) = 0$. This means $d(a_2(s_2), b_2(s_2)) = d(1, b_2(s_2))$. Let $t_2 = b_2(s_2)$ and $t_1 = a_2(s_2)$, $\Rightarrow t_2 \geq t_1$ and $d(t_1, t_2) = d(t_2, 1)$. The (t_1, t_2) is a point of $h_2 \Rightarrow d(t_1, t_2) = d(0, t_1)$. Consequently, we found $\{t_1, t_2\}$ with $t_2 \geq t_1$ satisfying the equation (2). Finally, the problem has been proved for $N = 3$.

We are now ready to prove the existence problem for $N > 3$. These cases are faced with a generalization of the previous methodology and their proof can be done inductively. This means that we are going to show how to compute $h_{N+1}(s)$ and the problem solution (existence of a set of points $\{t_1, \dots, t_N, t_{N+1}\}$ satisfying Equation (2)), using $h_N(s)$ computed by a previous step.

If $f_N(x, y)$ is defined under the constraint $y \geq x$, then $f_{N+1}(x, y)$ will be defined under the constraint $x \geq y$. Otherwise, if $f_N(x, y)$ is defined under the constraint $x \geq y$, then $f_{N+1}(x, y)$ will be defined under the constraint $y \geq x$. In our proof, we are going to consider the first case, where $f_N(x, y)$ is defined with $y \geq x$. The proof on the other case will be the same. $h_N(s) = [a_N(s), b(s)]$ satisfies the following statement (4):

$$a_N(s) < b(s), s \in (0, 1] \quad (4)$$

Let $\{s_0 = 0, s_1, \dots, s_{M-1}, s_M = 1\}$, $s_i > s_{i-1}, \forall i \in \{0, 1, \dots, M\}$ be an appropriate partition of $h_N(s)$, so that $\forall k \in U_{1..M} \hat{h}_k^{p_k}(s) = [\hat{a}_k(s), \hat{b}_k(s)] = h_N(s)$, $s \in [s_{k-1}, s_k]$:

- $x = \hat{a}_k(s)$ to be real monotonous, then $p_k = +$
- or $y = \hat{b}_k(s)$ to be real monotonous, then $p_k = -$

It will be true that at least one from the two will be real monotonous as $h_N(s)$ is partially monotonous. This means that:

- if $p_k = +$, then $\hat{h}_k^+(x) = [x, Y_k(x)]$, $[x : \hat{a}_k(s_{k-1}) \rightarrow \hat{a}_k(s_k)]$
- else if $p_k = -$, then $\hat{h}_k^-(y) = [X_k(y), y]$, $[y : \hat{b}_k(s_{k-1}) \rightarrow \hat{b}_k(s_k)]$

The union of $\hat{h}_k^{p_k}(x)$ will result the $h(s)$. Because $x \geq y$, we can choose an appropriate partition in order $\forall k \in U_M$, $p_k = -$. The unique case, that this can not be done, is when a part of $C(t)$ is cyclic arc, then the $\hat{b}_k(s)$ will be a constant function,

$$\hat{b}_k(s) = Y_k, s \in [s_{k-1}, s_k], \quad (5)$$

and just the $\hat{a}_k(s)$ will be real monotonous. Finally, the $h_N(s)$ partition can be done into functions having common end points.

For each $k \in U_M$ we can define the function $f_{N+1}^k(x, y)$ by the following equation.

$$f_{N+1}^k(x, y) = \begin{cases} d(x, y) - d(X_k(y), y) , & p_k = -, [y : \hat{b}_k(s_{k-1}) \rightarrow \hat{b}_k(s_k)], x \geq y, X_k(y) \leq y \\ d(x, Y_k) - d(\dot{x}, Y_k) , & p_k = +, [\dot{x} : \hat{a}_k(s_{k-1}) \rightarrow \hat{a}_k(s_k)], y = Y_k, x \geq Y_k, \dot{x} \leq Y_k \end{cases} \quad (6)$$

The functions $f_{N+1}^k(x, y)$ are continuous, partially monotonous and the curves $\hat{h}_k^{p_k}(x), \hat{h}_{k+1}^{p_{k+1}}(x)$ have the same end points. Because of them, it can be defined a continuous, partially monotonous curve that will be a subset of the union of the null plane curves of $f_{N+1}^k(x, y)$. This curve starts on $[0, 0]$ and it will be $h_{N+1}(s)$. The final point of set $G(x, N+1)$ is given by the (x, y) of these curves, because:

- $d(X_k(y), y) = d(x, y)$, $x \geq y$. It is derived by inductive hypothesis that the points $(X_k(y), y) \in h_N(s) \Rightarrow G(y, N)$.
- $d(x, Y_k) = d(\dot{x}, Y_k)$ and $x \geq Y_k$ (special case of cycle arc). It is derived by inductive hypothesis that the points $(\dot{x}, Y_k) \in h_N(s) \Rightarrow G(Y_k, N)$.

It can be proven using the same methodology of lemma 2 that $h_{N+1}(s)$ does not contain any region. If $h_{+1}(s) = [a_{+1}(s), b_{+1}(s)]$, $s \in [0, 1]$ then $a_{N+1}(s) \geq b_{N+1}(s)$. The next lemma proves that $h_N(s)$ ends on line $x = 1 \Leftrightarrow a_{N+1}(1) = 1$.

Lemma 4 *The curve $h_{N+1}(s)$ ends on $x = 1 \Leftrightarrow a_{N+1}(1) = 1$.*

Proof:

- *The $C(t), t \in [0, u]$ has a unique solution when u is close to 0. This means that $h_{N+1}(s)$ is not closed curve on $[0, 0]$ (if it was closed, it would have two solutions).*
- *Moreover, $f_{+1}(x, x) = d(x, x) - d(X_k(x), x) = -d(X_k(x), x) < 0, x > 0$. The only case to get zero would be when $\dot{x} = Y_k$. This is impossible by inductive hypothesis, the points of diagonal $x = y$ don't belong to the $h_N(s)$. In addition, $f_{N+1}(x, 0) = d(x, 0) - d(0, 0) = d(x, 0) > 0, x > 0$. Thus $h_{N+1}(s)$ will not have any intersection points with the diagonal $x = y$ and the line $x = 0$ apart from the $[0, 0]$. From the above, we can derive that the $f_{N+1}(x, y)$ changes its sign close to $(0, 0)$, at both sides of $h_{N+1}(s)$.*
- *In the special case of cycle arc: $f_{N+1}(x, x) = d(x, x) - d(\dot{x}, Y_k) = -d(\dot{x}, Y_k) < 0, \dot{x} > 0$. $f_{N+1}(x, 0) = d(x, 0) - d(\dot{x}, 0) = d(x, 0) - d(0, 0) = d(x, 0) > 0, x > 0$, which is derived by the equation (4), $\dot{x} = 0$ because $Y_k = 0$. Finally, it holds that $f_{+1}(x, y)$ changes its sign close to $(0, 0)$, at both sides of $h_{N+1}(s)$ and $h_{N+1}(s)$ will not have any intersection points with the diagonal $x = y$ and the line $x = 0$, apart from the $[0, 0]$.*

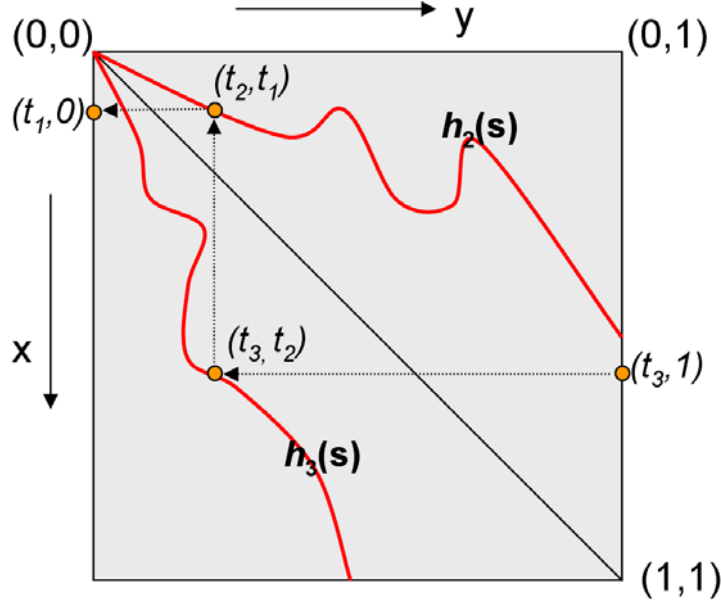


Fig. 6. An example of curve equipartition into 4 segments. It is shown the recursive computation of $\{t_1, t_2, t_3\}$ and the $h_2(s), h_3(s)$.

The proof, that $h_{N+1}(s)$ can not end somewhere inside the $(0,1)^2$, is exactly the same with the proof of Lemma 4 for $N = 3$.

We are going to prove that there exist a set of points $\{t_1, t_2, \dots, t_N, t_{N+1}\}$, $t_k > t_{k-1}$ satisfying Equation (2). Thus the problem will have solution for $N + 2$. We consider the function:

$$q(s) = d(a_{N+1}(s), b_{N+1}(s)) - d(a_{N+1}(s), 1), \quad s \in [0, 1] \quad (7)$$

It holds that

- $q(0) = d(0, 0) - d(0, 1) = -d(0, 1) < 0$ and
- $q(1) = d(1, b_{N+1}(s)) - d(1, 1) = d(1, b_{N+1}(s)) > 0$

We apply the Bolzano theorem on $q(s) \Rightarrow \exists$ at least a $s_{N+1} \in (0, 1)$ so that $q(s_{N+1}) = 0$. Hence $d(a_{N+1}(s_{N+1}), b_{N+1}(s_{N+1})) = d(a_{N+1}(s_{N+1}), 1)$. Let $t_{N+1} = a_{N+1}(s_{N+1})$ and $t_N = b_{N+1}(s_{N+1})$, $\Rightarrow t_{N+1} \geq t_N$ and $d(t_{N+1}, t_N) = d(t_{N+1}, 1)$. The point (t_{N+1}, t_N) belongs on $h_{N+1}(s)$, $\Rightarrow \exists k \in U_M$, $t_{N-1} = X_k(t_N) \leq t_N$ and $d(t_{N+1}, t_N) = d(t_{N-1}, t_N)$. The above procedure can be applied recursively to the functions $f_{N-i}(x, y)$, $i \in \{0, 1, 2, \dots, N-2\}$. It is derived that $d(t_{N-i}, t_{N-i-1}) = d(t_{N-i-2}, t_{N-i-1})$. Using this recursive approach, we can compute the $\{t_1, t_2, \dots, t_N, t_{N+1}\}$ satisfying the equation (1) (see Fig. 6). Finally, the proposition has been proven for $N > 3$.

In this section we are going to answer the question, if there exists an efficient (i.e. polynomial) algorithm that given C and N constructs an N -equipartition of C . To answer this issue we provide a partial and negative answer. Let us note that since for any polyline C with algebraically defined vertices and any N , there always exists an algebraically expressed solution, the EP problem belongs to the class of so called total function problems [24], an area of computational complexity still not rich enough in theoretical results. In fact it is not even clear whether EP belongs to the NP class, since we cannot offer (at least in this work) a guarantee that for any instance of EP there exist at least one concisely expressed solution that can be verified efficiently in P-time. Yet we are able to offer a rather negative result. More specifically in the following lemma 5 we prove that a version of EP is **NP**-hard. This version REP of EP is defined as follows:

REP:

- Input: A polygonal line C of M algebraically defined segments, a number N and a radius R .
- Query: Is there an equipartitioning of C by N division points, such that any two consecutive division points are at distance R apart?

Let us recall that the following problem (PARTITION) is **NP**-complete [25].

PARTITION:

- Input: A set of n weights, $S = \{W_k : k \in U_{1\dots n}\}$. We may suppose that $\sum_{k=1}^n W_k = 2\delta$.
- Query: Can S be partitioned into two subset S_1, S_2 of equal weight, i.e.: $\sum_{w \in S_1} W_k = \sum_{w \in S_2} W_k$?

Lemma 5 *The REP is **NP**-hard.*

Proof: *We shall reduce PARTITION to REP, thus proving that REP is **NP**-hard, therefore computationally rather intractable. To this end we shall use a geometrical ‘gadget’ consisting of the following consecutive segments (see Fig. 7):*

- *two horizontal segments of length R ,*
- *a horizontal segment **a** of length 4δ ,*
- *a vertical segment of length $\sqrt{R^2 - (4\delta)^2}$*
- *a horizontal segment **b** of length 4δ*
- *a segment of length R that leads to a ‘copy’ **c** of **b**, exactly above **b**,*

- a horizontal segment of length R ; a vertical segment of length $h = \sqrt{R^2 - p^2}$, where p is a gadget-parameter to be suitably selected later on,
- a horizontal segment of length R ,
- a copy $\mathbf{d}$ of $\mathbf{c}$ exactly above c
- and finally a horizontal segment of length $R4\delta$, followed by a horizontal segment of length R .

We shall need also an ‘initial’ version of this gadget, pictured in Fig. 7(b) (left) that starts at the middle point A of the $\mathbf{c}$ -type segment, and also a ‘final’ version, pictured in Fig. 7(b) (right), that ends at the middle point of a $\mathbf{d}$ -type segment.

Given an instance of *PARTITION* we form an instance of *REP* as follows: We set $R = 6\delta$ (division-points distance), $N = 7n - 6$ (number of division-segments), and define C by gluing n gadgets g_j together. All gadgets are as described above except gadget g_1 , (taken to be of ‘initial’ type), and gadget g_n (taken to be of ‘final’ type). The parameter p of the j^{th} gadget is set equal to weight W_j . Notice that we need only $\mathbf{P}$ -time to construct this instance of *REP*. We claim that the answer to the given *PARTITION*-instance is ‘yes’ if and only if the answer to the thus constructed *REP*-instance is also ‘yes’.

Indeed by a simple counting argument we can see that *REP* can be solved only if we can put one division point along each segment of type $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ on each of the n gadgets g_n , and no points of the segments that lead from $\mathbf{c}$ -type segments to $\mathbf{d}$ -type segments. Gadgets g_j and distance R have been defined so that the division points can be selected in a unique way, except on every segment of type $\mathbf{d}$. Let K_j be the division point on segment c_j (the $\mathbf{c}$ -type segment of gadget g_j), and L_j the division point of the d_j segment. By the selection of the parameter p of gadget g_j a circle with center K_j and radius R cuts segment d_j into two points that define a chord of length $2W_j$. Thus the division point L_j can be chosen into two ways so that its distance from the middle point of d_j (= its ‘offset’) can either be ‘increased’ by W_j or ‘decreased’ by W_j (in comparison with the distance of K_j from the corresponding middle point of c_j). Moreover the distance of the division point on a segment of type $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$, from the middle point of its corresponding segment is ‘conserved’. Thus choosing which division point to select on each segment d_j amounts to deciding whether to put weight W_j on subset S_1 or subset S_2 , since the final division L_n must coincide with the B point of the d_n segment on gadget g_n : this has length 2δ thus zeroing the total ‘offset’ of the final division point.

This result does not incur by itself any fatal consequences on the equipartition problem,

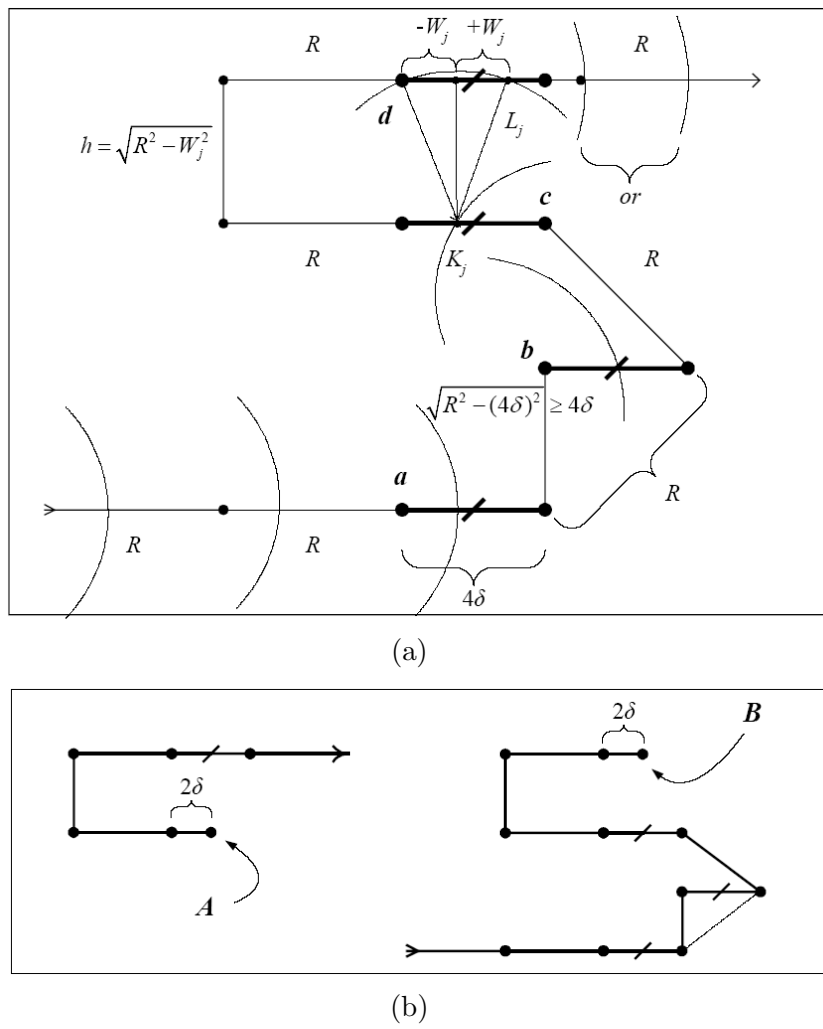


Fig. 7. A geometrical ‘gadget’ of lemma 5.

since in the EP problem we are free to choose R , and there might be a way to exploit this freedom. Yet our hands seem to be tightly roped: Fixing, for example, any two consecutive division points on C defines thereafter the distance R , and one is justified to wonder how this can be done correctly and efficiently given that deciding whether R is or is not a feasible division-distance for C is an **NP**-hard problem, as we have just shown.

5 Proof based Algorithm

This section presents the proof based algorithm and a proof that a version of the EP is **NP**-hard. The algorithm has been developed under the constraint that the input curve $C(t)$ is a polygonal curve and the distance metric is the Euclidean distance, but it can be developed for any partially algebraic curve or other distance metrics with the same way. We are going to use $d^2(x, y)$ in order to get polynomial expressions. The advantage of the method is that each computed result corresponds to an exactly problem solution. Unfortunately, the

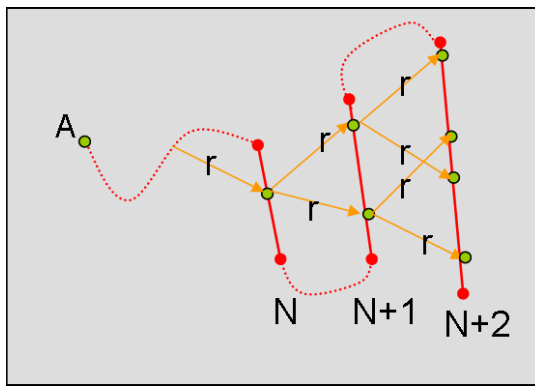


Fig. 8. The number of solutions (of length r) is possibly increased exponentially with N , in each line segment of the polygonal curve $C(t)$.

algorithm can not be implemented for $N > 4$ as it is analyzed below.

The $f_2(x, y)$ is a second order polynomial of x, y (equation (2)). The computation of $f_3(x, y)$ demands the curve $h_2(s)$, which contains expressions like $c_1s + c_2 + \sqrt{c_3s^2 + c_4s + 1}$, where c_i are constants. So, if we want to compute the $h_3(s)$ we have to square the $f_3(x, y)$ getting a fourth order polynomial. This procedure is repeated with the same way. Therefore, the number of coefficients is increased exponentially with N . This can be explained as if we have computed the solution on a linear segment, for N , then for $N+1$ the number of solutions will be possible redoubled (see Fig. 8). The computed polynomials should have the appropriate degree in order to compute the solutions, and this can be achieved only if they increase their degree exponentially. In addition, there is not a closed form solution for each polynomial of order higher than 5, so we can not solve the problem using the proof based algorithm for $N > 4$.

However, we have developed an approximate proof based algorithm which is an easy implemented algorithm utilizing the approximation properties. The greedy version of the method can be executed for each N with low computation cost $O(N \cdot M^2)$, where M is the number of sampling of $C(t)$ which determines the method accuracy. More details about this method are given in the next section 5.1.

5.1 Proof based Approximate Algorithm

The proof based approximate algorithm can be performed in any curve using any distance metric. It uses the proof methodology under the hypothesis that the function $d(x, y)$ can be approximated by a polygonal surface $\hat{d}(x, y)$. It computes at least one solution or all the solutions (greedy version). It is inductive. Thus, when they are executed for N , they solve

the problem for any number of segments less than N .

Let $i, j \in U_{1...M}$, so that segments

$$D_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}] \subset [0, 1]^2$$

, with

$$x_1 = y_1 = 0 \leq x_2 = y_2 \leq \dots \leq x_{M-1} = y_{M-1} \leq x_M = y_M = 1$$

define $\hat{d}(x, y)$ (Equation (11)). M denotes the size of sampling and it is related to the method accuracy.

$$a_{ij} = [d(x_{i+1}, y_j) - d(x_i, y_j) \quad d(x_{i+1}, y_{j+1}) - d(x_{i+1}, y_j)]^T \quad (8)$$

$$b_{ij} = [d(x_i, y_{j+1}) - d(x_i, y_j) \quad d(x_{i+1}, y_{j+1}) - d(x_i, y_{j+1})]^T \quad (9)$$

$$p = [(x - x_i) \quad (y - y_j)] \quad (10)$$

$$\hat{d}(x, y) = \begin{cases} d(x_i, y_j) + p \cdot a_{ij}, (x, y) \in D_{ij} \wedge x - x_i \geq y - y_i \\ d(x_i, y_j) + p \cdot b_{ij}, (x, y) \in D_{ij} \wedge x - x_i < y - y_i \end{cases} \quad (11)$$

The number of segments is $2M^2$, as D_{ij} is divided into two subsegments (triangulation): D_{ij}^1 where $x - x_i \geq y - y_i$ and the D_{ij}^2 where $x - x_i < y - y_i$. Then the function $f_N^k(x, y)$ would be partially linear, for each N , in the same segments D_{ij}^1, D_{ij}^2 . Therefore, for each N the null plane curves of $f_N^k(x, y)$ would be polygonal in the same segments D_{ij}^1, D_{ij}^2 and finally, $q(s)$ would be partially linear in the same segments D_{ij}^1, D_{ij}^2 . The proof based approximate algorithm computes at least one solution of the problem, if we execute the following algorithm just on null plane curves passing from $[0, 0]$ ($h_N(s)$). Otherwise, we are going to get all the solutions.

- Let $C(t_k)$, $k \in U_{1...M}$ a sampling of $C(t)$, $\{t_1 = 0 < t_2 < \dots < t_{M-1} < t_M = 1\}$. The method computes step-by-step the null plane curves of $f_N^k(x, y)$. These curves are partially linear in segments D_{ij}^1, D_{ij}^2 .
- Let L_N be the set of segments that define the null plane curves of $f_N^k(x, y)$. Each segment can be defined by its two end-points. It holds that,

$$L_1 = [(0, 0), (t_1, 0)] \cup [(t_1, 0), (t_2, 0)] \cup \dots \cup [(t_{M-1}, 0), (1, 0)].$$

L_{n+1} is computed using the precomputed L_n . Let

$$w = [(u_1, v_1), (u_2, v_2)], u_1 < v_1, u_2 < v_2$$

be a segment of L_n , $s \in D_{ij}^1$. Then we can compute segments

$$\dot{w} = [(u_1, v_1), (u_2, v_2)], u_1 < v_1, u_2 < v_2$$

of L_{n+1} that are derived by segment w .

- Let $f_{N+1}^k(x, y)$, $x \in [\min(v_1, v_2), 1]$, $y \in [\min(v_1, v_2), \max(v_1, v_2)]$ correspond to segment w . The end-points of $\dot{w}$ are points of borders of D_{jz}^1 or D_{jz}^2 $z \geq j$. The number of segments $\dot{w}$ that belong on D_{jz}^1 or D_{jz}^2 is no more than one, as $f_k^{N+1}(x, y)$ is linear on D_{jz}^1 or D_{jz}^2 . So, we can compute these end points by applying the Bolzano theorem to $f_{N+1}^k(x, y)$ on borders of D_{jz}^1 , D_{jz}^2 .
- Finally, we can define the linear function $q(s) = d(a(s), b(s)) - d(1, b(s))$ in any segment of L_N , where $[(a(s), b(s))]$ denotes the curve equation of a segment of L_N . The root⁴ of $q(s)$ can be computed by applying the Bolzano theorem on end points of the segment.

We can estimate an normalized error (NE) of an estimated equipartition of length segments r by getting the standard deviation of the estimated length segments of this equipartition divided by the mean length segment of this equipartition ($E(r)$).

$$NE = \frac{\sqrt{E(r^2) - E^2(r)}}{E(r)} \quad (12)$$

NE is decreased as M increases. It holds that the mean error of the approximation of $d(x, y)$ satisfies $E(|\hat{d}(x, y) - d(x, y)|) = O(\frac{1}{M^2})$. It can be proved that NE is decreased by the same factor $O(\frac{1}{M^2})$ (see Fig. 9(a)). Moreover, the NE is not affected as N is increased (see Fig. 9(b)). However, M should be greater than N , while our experiments show that it should be $M > 2N$.

Concerning the greedy method computation cost, the computation cost for each segment s is $O(M)$. The number of segments of L_n in normal cases is $O(M)$ ⁵. Therefore the computation cost of the greedy version of the algorithm is $O(N \cdot M^2)$ while the memory requirements are $O(N \cdot M)$. If we want to compute at least one problem solution, we can compute for each N the polygonal curves $h_N(s)$ (passing from $[0, 0]$) by tracking them. The search space of

⁴ $q(s)$ is a linear function, so it has no more than one root.

⁵ In special cases where the null plane curves density is very high the cost will be $O(M^2)$.

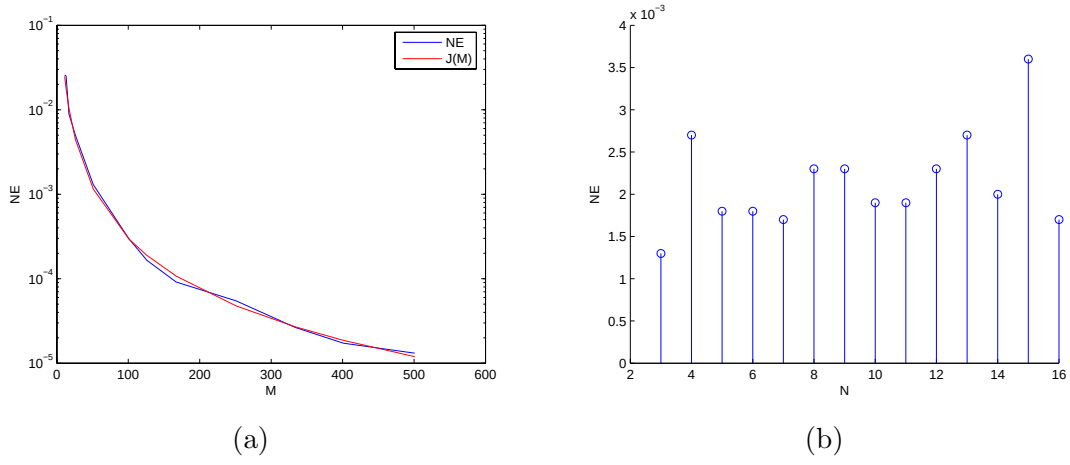


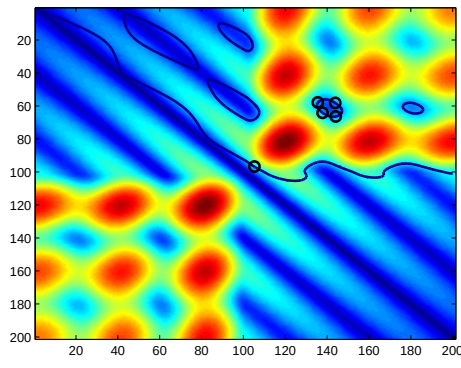
Fig. 9. The normalized error (NE) computed on curve of figure 10(g). When we get more than one solutions, the NE was computed by the mean of normalized errors on these solutions. **(a)** The NE and its approximation function $J(M) = \frac{3}{M^2}$ computed for different values of M and for $N = 3$. **(b)** The NE computed for different values of N and for $M = 50$.

a vertex of the polygonal curve $h_N(s)$ is reduced to the neighbor triangles D_{ij}^1, D_{ij}^2 of the previous precomputed vertex of $h_N(s)$. In this case, the computation cost and the memory requirements are $O(N \cdot M)$.

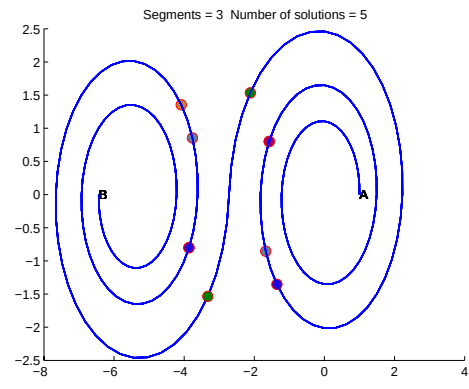
Figure 10 illustrates the results of this proposed algorithm for different curves and values of N . The null plane curves converge to the diagonal ($y = x$), as N increases (see Fig. 10(g)), and there exist exactly one solution. At least one solution belongs on the $h_N(s)$ null plane curve. However, in some cases, more solutions appear on other null plane curves (see Fig. 10(a), 10(c)).

6 Other Algorithms

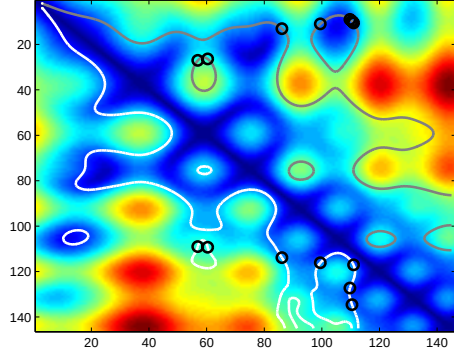
In this section, two algorithms are described, an approximate algorithm and a steepest descent based algorithm. These methods have consummative properties like: The approximate algorithm works better for low values of N , computing all the problem solutions, while steepest descent based algorithm works better for high values of N , computing just a problem solution. In addition, the approximate algorithm results can be used to initialize the steepest descent based algorithm.



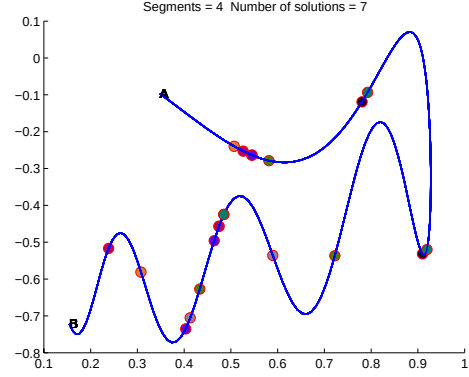
(a)



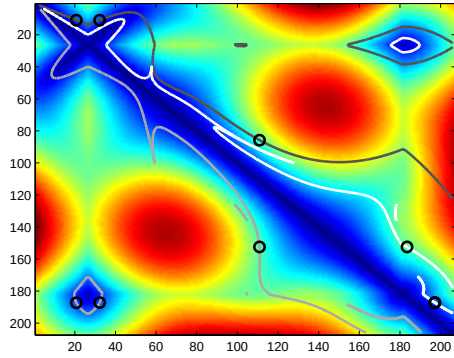
(b)



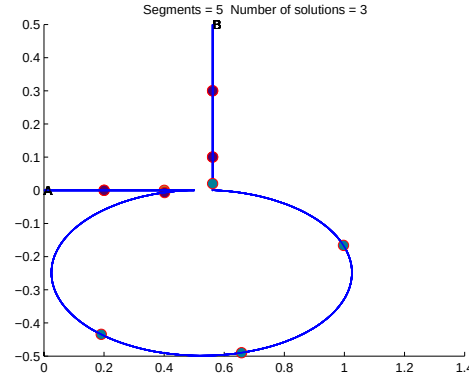
(c)



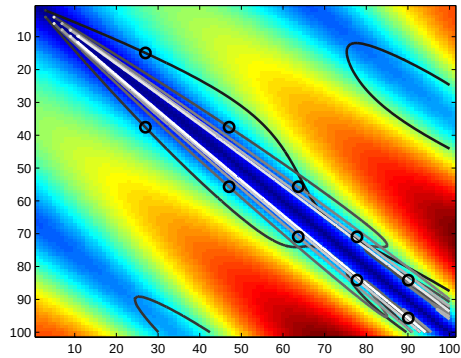
(d)



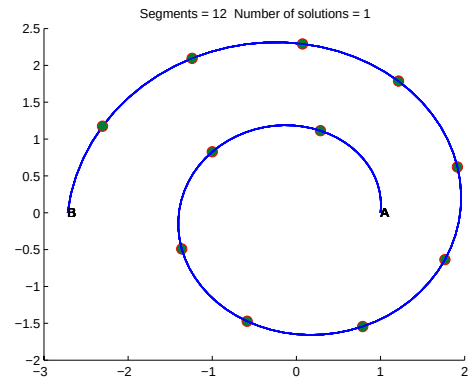
(e)



(f)



(g)



(h)

Fig. 10. Results of greedy version of proof based approximate algorithm. The estimated solutions are projected on $d(x, y)$ (left) with black cycles and on input curve $C(t)$ (right) with the same color points belonging to the same equipartition. The null plane curves are projected on $d(x, y)$, with gray colors, at both sides of diagonal $x = y$.

This method computes approximately all the problem solutions in a discrete sampling set of $C(t)$ given N . The solutions are derived by finding the local minima of computed segments lengths variance. The advantage of this method is that it computes all the problem solutions. However, it is possible to compute as solution a local minimum which gives low variance. The solution accuracy is increased as the curve sampling increases.

Let $C(t_m), m \in U_{1...M}$ be a sampling of $C(t)$. Let $S_j = \{t_i^j, i \in U_{0,...,N}, t_0^j = 0, t_N^j = 1\}$ be an partition of $C(t)$ into N segments. The lengths of S_j are given by the equation:

$$L(i)^j = |C(t_{i+1}^j) - C(t_i^j)|_2, i \in U_{0,...,N-1}.$$

An exhaustive algorithm, which examines all the possible cases of S_j , would have complexity $O(KM^{N-1})$. Thus it can not be executed even if $N = 4$. So, the main goal is to reduce the complexity of the proposed algorithm using a dynamic programming approach. First, the algorithm computes the distances from A to each curve point. These points will be the second points of the candidate ‘equipartitions’. Then it computes step-by-step the points of candidate ‘equipartitions’ from the third until the N . We select from these ‘equipartitions’, those that satisfy the conditions:

- their mean segment length should be between $\frac{|A-B|_2}{N}, \frac{\int_0^1 |\dot{C}(u)|_2 du}{N}$
- their N point distance from B should be almost equal to their mean segment length
- their lengths variance should be local minimum and lower than a threshold.

The lengths variance is related with the approximation error. If the variance is zero, then the lengths will be equal, giving us an exact solution. A pseudocode of this procedure can be found on chapter 3 of [23]. The time complexity of the algorithm is proportional to $NK \log K$ when the number of examined partitions is bounded. However, its execution time depends on the input curve shape. If many solutions are generated (possible exponentially), as N is increased, then the time and memory requirements will be increased exponentially with N . As our experiments show, it can not be executed for large N ($N > 30$) because of high execution time and memory requirements. In addition, when N is large, the segments length of a solution will be low, which means that we need many sampling points in order to get good approximation. So, K should be high increasing the execution time.

Results of this proposed algorithm are shown in Fig. 11. We have implemented the method

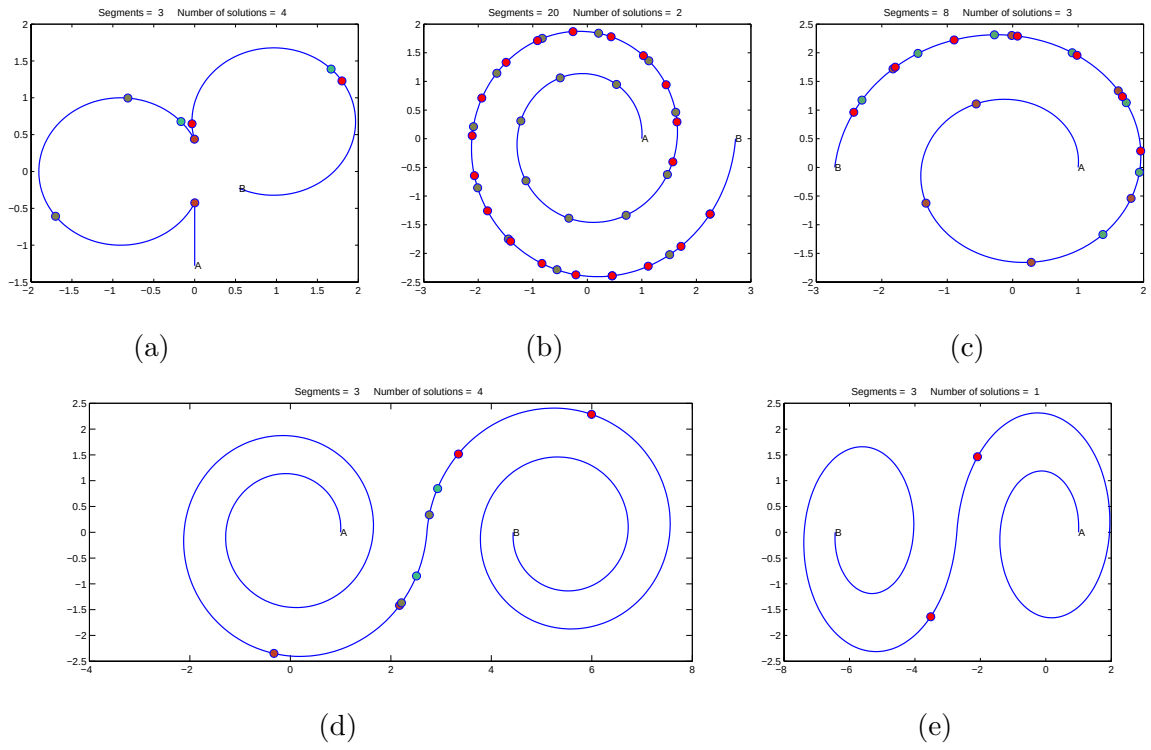


Fig. 11. Approximate algorithm results for different N and curves. The same color points belong to the same equipartition.

in Matlab and it needs less than one minute to solve the problem in each of the examined curves of Fig. 11, where $M \simeq 300$. The run times have been obtained using Pentium 4 at 2.8 GHz.

6.2 Steepest Descent based Method

The steepest descent based method (SDM) converges to the most close to an initial equipartition, solution, given this initial equipartition. The major advantage of this method is that the computed segments will have exactly the same length, as the end of the last segment is converging to B . In some cases the algorithm can not converge as there may appear local minima or jumps (loops) between different solutions. These phenomena are increased, when the initialization is far enough from an existing solution. But, when the problem has unique solution, which is usually observed for high N , then the algorithm will converge. A pseudocode of this procedure is given hereafter.

Steepest Descent based Algorithm

```

 $s = r_0$ 
 $t_0 = 0$ 
 $P_0 = A$ 
Repeat
  for  $i=1:N$ 
     $P_i = C(t_i) : (t_i > t_{i-1}) \wedge (|P_{i-1}P_i| = s)$     end
     $s = \begin{cases} s + \lambda \frac{|B-P_N|}{N}, & P_N \text{ belongs inside of curve } C(t) \\ s - \lambda \frac{|B-P_N|}{N}, & P_N \text{ belongs outside of curve } C(t) \end{cases}$ 
  Until  $|P_N - B| < T$ 

```

The learning rate λ determines the number of steps which are needed for convergence. However, when λ is set to a high value ($\lambda > 0.5$), it can cause instability and not convergence. For better convergence, we can start the method with $\lambda \approx 0.5$ and we decrease it to $\lambda \approx 0.05$. Conclusively, if we want to solve the problem for lower N , where there are possibly many solutions and local minima, it is better to execute first the approximate algorithm, getting a good initialization for the steepest descent based algorithm. For higher N , where the problem might have a unique solution, the proposed method will converge towards the exact solution, even if the initialization is not close to the solution. The time complexity of the algorithm is $O(N \cdot S)$, where S denotes the number of steps that are needed for convergence. S is dependent on curve shape and how close to the final solution the initialization is. Results of the steepest descent based algorithm are shown in Fig. 12.

6.2.1 Genetic Version of Steepest Descent based Method

In this section we are going to analyze a genetic version of steepest descent based method (GSDM) which does not demands initialization. First, two SDM initialized are executed with $r_0 = \frac{|A-B|_2}{N}$, $r_1 = \frac{\int_0^1 |\dot{C}(u)|_2 du}{N}$, the minimum and the maximum limits of the segment length of the estimated equal segments, respectively. Then, the iterative part of the method is executed. We compute all the equipartitions that have segments lengths equal to r for each candidate segment length r . For each equipartition of them we start a SDM. In addition, we start a SDM initialized with a random segment length r (genetic approach) when a constant number of iterations have passed. If a SDM has not converged after a number of iterations (possibly local minimum), it will be killed. This method is executed until

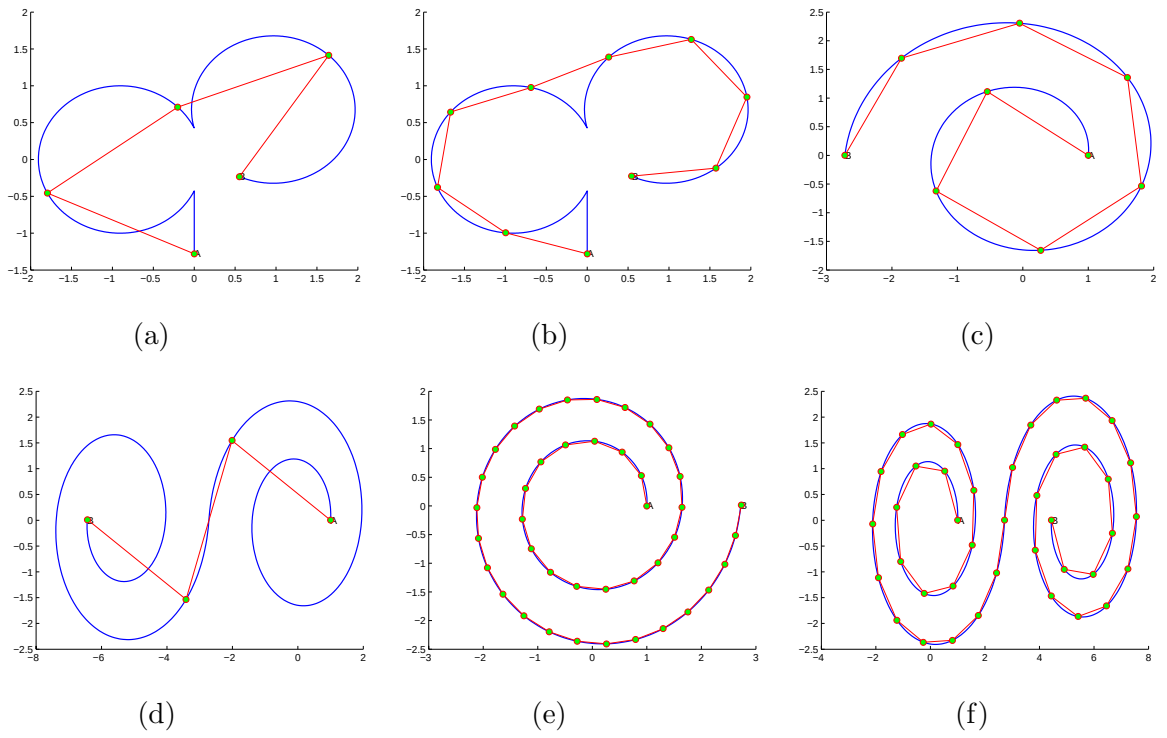


Fig. 12. Steepest descent based algorithm results for different N and curves.

a SDM approximates a problem solution, the end of the computed equal length polygon approximates the end of curve B .

The time complexity is $O(N \cdot S \cdot K)$, where S denotes the convergence steps and K denotes the maximum number of executed SDM. K is possible to take large values over the time. In this case, a possible improvement of the method would be to use a priority list (FIFO) storing the the candidate equipartitions that are going to initialize next SDMs. A new SDM is started if the current number of executed SDM is less than K_{max} , where K_{max} denotes the maximum number of executed SDM.

In summary, the execution time of this method depends on curve and the N . If there are many local minima the algorithm needs many steps to converge. The convergence is always obtained because of genetic step of the algorithm. Theoretically, the method could compute all the problem solutions, if we did not stop it on the first convergence. However, this would demand many steps and there is not any statement to recognize that all the solutions have already been computed. We have implemented the methods in Matlab without optimizations, the SDM needs a few seconds when it converges. In most of the cases, GSDM needs less than one minute to converge. But this is dependant on curve.

In this paper, we have discussed the curve equipartition problem (EP). We proved that the problem has at least one solution for any partially algebraic curve and for any number of segments. The equivalent problem definition, using function $d(x, y)$, led to a compact proof and an approximate algorithm which is flexible on changes of distance metric. In addition, it led to an algorithm that computes exactly the problem solutions. However, this accurate algorithm needs to solve polynomials that increase their degree exponentially with N , so it can not be implemented for $N > 4$. We have proved that a version of EP is **NP**-hard. This means that the EP possibly being complete for the NP class, thus we worked on development of approximate methods solving the EP. The approximate proof based algorithm can compute at least one solution or all the solutions using a greedy version of the algorithm. These algorithms are inductive. Thus, when they are executed for N , they solve the problem for any number of segments less than N , without additional computation cost. The results of this method can be used to initialize the steepest descent based algorithm converging to the real solution. The execution time of the method is independent of curve and it is related to N and to number of samples M . This property gives a guaranty of the maximum number of steps that needed are bounded. The computation complexity of the greedy version is $O(N \cdot M^2)$. However, we can compute at least one solution in $O(N \cdot M)$ steps. Moreover, the normalized error NE of an computed equipartition is decreased by $O(\frac{1}{M^2})$.

Furthermore, two other algorithms, an approximate algorithm and a steepest descent based algorithm, have been presented having consummative properties. The steepest descent based algorithm is the simplest, it has low computation cost converging towards to a solution specially for high N . The approximate algorithm computes all the problem solutions in a discrete set. It is appropriate for low N and it can be used to initialize the steepest descent based algorithm. The SDM is possible to not converge to a solution. This occurs when the initialization is far enough from a solution and there are local minima. However, we have proposed a genetic version of steepest descent based algorithm that converges always to a solution, without initialization. The execution time of these algorithms depends on curve and N .

As future work, we plan to study and propose other algorithms that can solve the equipartition problem or apply the proposed methodology solving other problems. A challenging goal would be to apply it into the polygon approximation problem. In addition, the pro-

posed method can be applied to key frames extraction in a scene. The distance between the key frames will be equal and this means that their content is equidistant. We can use any distance metric based on video content, camera motion and scene object motion as the proposed method can be applied into any smooth distance metric. Non articulated motions can be tracked using the proposed method. Until now, we have applied the methodology on snake tracking problem [9]. The time correspondence between the equal spaced computed points is done automatically. We plan to apply it into another non articulated motion tracking problems, like lips tracking, where the color information can not be sufficient to solve the problem.

References

- [1] C. Panagiotakis and G. Tziritas, “A speech/music discriminator based on RMS and zero-crossings,” *IEEE Trans. on Multimedia*, vol. 7, no. 1, pp. 155–166, Feb. 2005.
- [2] S. Liapis, E. Sifakis, and G. Tziritas, “Colour and texture segmentation using wavelet frame analysis, deterministic relaxation and fast marching algorithms,” *Journal of Visual Communication and Image Representation*, vol. 15, no. 1, pp. 1–26, March 2004.
- [3] H. Delinhette, “General object reconstruction based on simplex meshes,” *International Journal of Computer Vision*, vol. 32, pp. 111–142, 1999.
- [4] P. S. Heckbert and M. Garland, “Survey of polygonal surface simplification algorithms,” in *SIGGRAPH*, 1997.
- [5] C. Sohler, “Fast reconstruction of delaunay triangulations,” *Computational Geometry*, vol. 31, pp. 166–178, 2005.
- [6] V. Kwatra and J. Rossignac, “Space-time surface simplification and edgebreaker compression for 2D cel animations,” *Intern. Journal of Shape Modeling*, vol. 8, pp. 119–137, 2002.
- [7] D. DeMenthon, V. Kobla, and D. Doermann, “Video summarization by curve simplification,” *ACM Multimedia*, pp. 211–218, 1998.
- [8] A. D. Doulamis, N. Doulamis, and S. Kollias, “Non-sequential video content representation using temporal variation of feature vectors,” *IEEE Trans. on Consumer Electronics*, vol. 46, pp. 758–768, 2000.
- [9] C. Panagiotakis and G. Tziritas, “Construction of animal models and motion synthesis in 3D virtual environments using image sequences,” in *Proc. of the second International Symposium on 3DPVT (3DPVT 2004)*, 2004.
- [10] A. Kolesnikov, “Efficient algorithms for vectorization and polygonal approximation,” Ph.D. dissertation, University of Joensuu, Finland, 2003.
- [11] R. Weibel, “Generalization of spatial data: principles and selected algorithms,” *Algorithmic Foundations of Geographic Information System*, 1997.
- [12] K.-L. Chung, W.-M. Yan, and W.-Y. Chen, “Efficient algorithms for 3-d polygonal approximation based on lise criterion,” *Pattern Recognition*, vol. 35, pp. 2539–2548, 2002.

- [13] M. A. Lopez and S. Reisner, "Hausdorff approximation of convex polygons," *Computational Geometry*, vol. 32, pp. 139–158, 2005.
- [14] A. Katsaggelos, L. Kondi, F. Meier, J. Ostermann, and G. Schuster, "MPEG-4 and ratedistortion -based shape-coding techniques," *Proceedings of the IEEE, Special Issue on Multimedia Signal Processing*, vol. 86, no. 6, pp. 1126–1154, 1998.
- [15] M. Saloti, "An efficient algorithm for the optimal polygonal approximation of digitized curves," *Pattern Recognition Letters*, vol. 22, pp. 215–221, 2001.
- [16] P. Agarwal, S. Har-Peled, N. Mustafa, and Y. Wang, "Near-linear time approximation algorithms for curve simplification in two and three dimensions," in *Proc. of the 10th European Symposium on Algorithms (ESA '02)*, 2002.
- [17] A. Kolesnikov and P. Franti, "Reduced-search dynamic programming for approximation of polygonal curves," *Pattern Recognition Letters*, vol. 24, pp. 2243–2254, 2003.
- [18] Y. Kurozumi and W. Davis, "Polygonal approximation by the minimax method," *Computer Vision, Graphics and Image Processing*, pp. 248–264, 1982.
- [19] H. Imai and M. Iri, "Polygonal approximations of a curve (formulations and algorithms)," *Computational Morphology*, pp. 71–86, 1988.
- [20] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing*, vol. 1, pp. 244–256, 1972.
- [21] P. Rosin, "Assessing the behaviour of polygonal approximation algorithms," *Pattern Recognition*, vol. 36, pp. 505–518, 2003.
- [22] M. Marji and P. Siy, "Polygonal representation of digital planar curves through dominant point detectiona nonparametric algorithm," *Pattern Recognition*, vol. 37, pp. 2113–2130, 2004.
- [23] C. Panagiotakis, G. Georgakopoulos, and G. Tziritas, "Curve segmentation into equal segments," Computer Science Department, University of Crete, Tech. Rep. CSD-TR-05-02, 2005 (in greek).
- [24] N. Megiddo and C. Papadimitriou, "On total functions, existence theorems, and computational complexity," *Theoretical Computer Science*, vol. 81, pp. 317–324, 1991.
- [25] R. M. Karp, "Reducibility among combinatorial problems," *Proc. Sympos. IBM Thomas J. Watson Res. Center on Complexity of Computer Computations*, pp. 85–103, 1972.