# A Near-Linear Time Guaranteed Algorithm for Digital Curve Simplification under the Fréchet Distance

Isabelle Sivignon

**gipsa**-lab, CNRS, UMR 5216, F-38420, France
isabelle.sivignon@gipsa-lab.grenoble-inp.fr

**Abstract.** Given a digital curve and a maximum error, we propose an algorithm that computes a simplification of the curve such that the Fréchet distance between the original and the simplified curve is less than the error. The algorithm uses an approximation of the Fréchet distance, but a guarantee over the quality of the simplification is proved. Moreover, even if the theoretical complexity of the algorithm is in $\mathcal{O}(n \log(n))$, experiments show a linear behaviour in practice.

## 1 Introduction

Given a polygonal curve, the curve simplification problem consists in computing another polygonal curve that (i) approximates the original curve, (ii) satisfies a given error criterion, (iii) with as few vertices as possible. This problem arises in a wide range of applications, such as geographic information systems (GIS), computer graphics or computer vision, where the management of the level of details is of crucial importance to save memory space or to speed-up analysis algorithms.

This problem has been studied for many years for various metrics: classical $L_1, L_2, L_\infty$ metrics (see [4] for a survey on simplification algorithms using these metrics), Hausdorff distance or Fréchet distance. While the $L$ norms and Hausdorff distance are relevant measures for many applications, they do not always reflect the similarity or dissimilarity of two polygonal curves (see for instance the example given in [10] for the Hausdorff distance). The main reason of this discrepancy is that these metrics consider the curves as sets of points, and do not reflect the course of the curves. However, the course of the curve may be important in some applications, like handwritting recognition for instance [11]. The Fréchet distance is often used to overcome this problem as it nicely measures the similarity between two curves. The Fréchet distance can be intuitively defined considering a man walking his dog. Each protagonist walks along a path, and controls its speed independently, but cannot go backwards. The Fréchet distance between the two pathes is the minimal length of the leash required.

### 1.1 Fréchet Distance

Given two curves $f$ and $g$ specified by functions $f : [0, 1] \to \mathbb{R}^2$ and $g : [0, 1] \to \mathbb{R}^2$, and two non-decreasing continuous functions $\alpha : [0, 1] \to [0, 1]$ and $\beta :$

I. Debled-Rennesson et al. (Eds.): DGCI 2011, LNCS 6607, pp. 333–345, 2011.

$[0,1] \rightarrow [0,1]$ with $\alpha(0) = 0, \alpha(1) = 1, \beta(0) = 0, \beta(1) = 1$, the *Fréchet distance* $\delta_F(f,g)$ between two curves f and g is defined as

$$\delta_F(f,g) = \inf_{\alpha,\beta} \max_{0 \leq t \leq 1} d(f(\alpha(t)), g(\beta(t)))$$

where $d$ denotes the Euclidean distance. The polygonal curve simplification problem was first studied for the Fréchet distance by Godau [7]. Alt and Godau proposed in [3] an $\mathcal{O}(mn)$-time algorithm to determine whether $\delta_F(P,Q) \leq \varepsilon$ for two polygonal curves $P$ and $Q$ of size $n$ and $m$, and a given error $\varepsilon > 0$. The complexity turns out to be $\mathcal{O}((m^2n + n^2m)\log(mn))$ for the actual computation of the Fréchet distance between two curves. A recent work [6] proposes a near-linear time algorithm to compute an approximation of this distance.

### 1.2 Curve Simplification Problem

In the rest of the paper, we follow the notations used in [2] or [1]. Given a polygonal curve $P = \langle p_1, \dots p_n \rangle$, a curve $P' = \langle p_{i_1}, \dots p_{i_k} \rangle$ with $1 = i_1 < \dots < i_k = n$ is said to *simplify* the curve $P$. $P(i,j)$ denotes the subpath from $p_i$ to $p_j$.

Given a pair of indices $1 \leq i \leq j \leq n$, $\delta_F(p_ip_j, P)$ denotes the error of the segment $p_ip_j$ with respect to $P(i,j)$. Then,

$$\delta_F(P', P) = \max_{1 \leq j \leq k} \delta_F(p_{i_j}p_{i_{j+1}}, P)$$

For the sake of clarity, the simplified notation $error(i,j) = \delta_F(p_ip_j, P)$ will sometimes be used. We also say that $p_ip_j$ is a *shortcut*.

$P'$ is an *$\varepsilon$-simplification* of $P$ if $\delta_F(P', P) \leq \varepsilon$. The optimization problem is the following: given a polygonal curve $P$, find a $\varepsilon$-simplification $P'$ of $P$ with the minimum number of vertices. The approach of Imai and Iri [8] leads to an optimal solution under the Fréchet distance in $\mathcal{O}(n^3)$.

In [2], the authors propose an $\mathcal{O}(n\log(n))$ algorithm. The base of their algorithm is greedy and very simple: points are added one by one while the error of the shortcut is lower than $\varepsilon$, otherwise the process starts over from the last point processed. The strength of their approach lies in the following property:

**Lemma 1.** *[2, Lemma 3.3] Let $P = \{p_1, p_2, \dots, p_n\}$ be a polygonal curve in $\mathbb{R}^2$. For $1 \leq i \leq l \leq r \leq j \leq n$, $error(l,r) \leq 2error(i,j)$*

This means that for any shortcut $p_lp_r$ such that $error(l,r) > \varepsilon$, there does not exist a shortcut $p_ip_j$ such that $error(i,j) \leq \frac{\varepsilon}{2}$. They derive the following theorem:

**Theorem 1.** *[2, Theorem 3.4] Given a polygonal curve $P$ in $\mathbb{R}^d$ and a parameter $\varepsilon > 0$, we can compute in $\mathcal{O}(n\log(n))$ an $\varepsilon$-simplification $P'$ of $P$ under the Fréchet metric error with at most the number of vertices of an optimal $\frac{\varepsilon}{2}$-simplification.*

The $\mathcal{O}(nlog(n))$ complexity is achieved with a dichotomic processs in the greedy algorithm. The question we arouse in this paper is the following: does there exist

a linear-time guaranteed algorithm to compute an $\varepsilon$-simplification ? We propose a guaranteed algorithm for digital curves whose complexity is $\Omega(n \log(n))$ in theory but behaves in $\mathcal{O}(n)$ in practice. The main features of this algorithm are the following: (i) approximation of the Fréchet distance as in [1], (ii) restriction to digital curves. In section 2, we present the approximated distance, and give the algorithm outline as a novel way of using the results of [1]. Section 3 is the core of the paper and details the approximated distance efficient and online update, with a restriction to digital curves to achieve a near-linear complexity. Section 4 begins with a theoretical study of the complexity and the guarantee of the proposed algorithm, and ends with some experimental results.

## 2 Guaranteed Algorithm Using an Approximated Distance

### 2.1 Approximating the Fréchet Distance

In [1], Ali Abam et al. study the following problem: given a possibly infinite sequence of points defining a polygonal path, maintain in a streaming setting, a simplification of this set of points with $2k$ points and a bounded error. The error is measured using the Hausdorff distance or the Fréchet distance. The Fréchet distance being computationally too costly for this framework, they show that $error(i, j)$ can be upper and lower bounded by functions of two values, namely $\omega(i, j)$ and $b(i, j)$. $\omega(i, j)$ is the width of the points of $P(i, j)$ in the direction $p_i p_j$. $b(i, j)$ is the length of the longest backpath in the direction $\overrightarrow{p_i p_j}$. Its precise definition requires the following other definitions:

**Definition 1.** *Let $l$ be a straight line of directional vector $\overrightarrow{d}$. $\alpha$ is the angle between $l$ and the abscissa axis.*

*$proj_\alpha(p)$ denotes the orthogonal projection of $p$ onto the line of angle $\alpha$.*

*If $\overrightarrow{p_l p_m}.\overrightarrow{d} < 0$, then $proj_\alpha(p_l)$ is "after" $proj_\alpha(p_m)$ in direction $\alpha$, and we denote $proj_\alpha(p_l) >> proj_\alpha(p_m)$ (see Figure 1(a)).*

**Definition 2.** *$\overrightarrow{p_l p_m}$ is a* positive shift *if and only if $\overrightarrow{p_l p_m}.\overrightarrow{d} > 0$, negative otherwise.*

**Definition 3.** *A* backpath *in direction $\alpha$ is a negative shift $\overrightarrow{p_l p_m}$ such that $l < m$ (see Figure 1(b)). $p_l$ is the* origin *of the backpath. The* length *of the backpath is equal to $d(proj_\alpha(p_l), proj_\alpha(p_m))$.*

Lemma 2 relates $\omega(i, j)$ and $b(i, j)$ to $error(i, j)$:

**Lemma 2.** *[1, Lemma 4.2] The Fréchet error of a shortcut $p_i p_j$ satisfies* $\max(\frac{w(i,j)}{2}, \frac{b(i,j)}{2}) \leq error(i, j) \leq 2\sqrt{2}\max(\frac{w(i,j)}{2}, \frac{b(i,j)}{2})$.

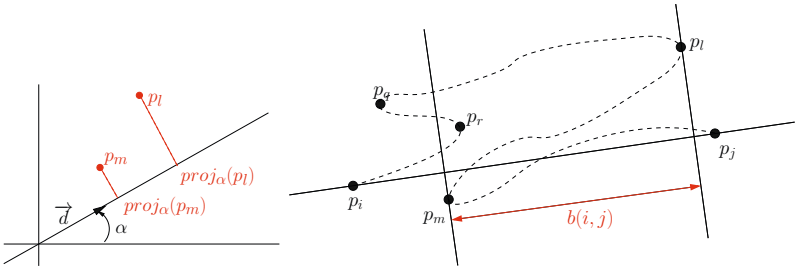



**Fig. 1.** (a) Illustration of Definitions 1 and 2. (b) Illustration of backpathes in direction $\overrightarrow{p_i p_j}$: $\overrightarrow{p_r p_q}$ is a backpath since it is a negative shift and $p_r$ is before $p_q$ on the curve $P(i,j)$. However, it is not the longest backpath: $\overrightarrow{p_l p_m}$ is also a backpath, of maximal length $b(i,j)$.

**Algorithm 1.** Greedy Fréchet simplification algorithm

$i = 1$, $j = 2$
**while** $i < n$ **do**
  **while** $j < n$ *and* $\max(w(i,j), b(i,j)) \leq \frac{\varepsilon}{\sqrt{2}}$ **do**
    j=j+1
  create a new shortcut $p_i p_{j-1}$
  $i = j - 1$,$j = i + 1$

## 2.2 Algorithm Outline

Algorithm 1 presents the general outline of the $\varepsilon$-simplification.

**Lemma 3.** *Algorithm 1 computes an $\varepsilon$-simplification $P'$ of a polygonal curve $P$ such that $|P'|$ is lower than the number of vertices of an optimal $\frac{\varepsilon}{4\sqrt{2}}$-simplification of $P$.*

*Proof.* When a shortcut $p_i p_j$ is created in $P'$, the following two properties are true: (i) $\max(w(i,j), b(i,j)) \leq \frac{\varepsilon}{\sqrt{2}}$ and (ii) $\max(w(i,j+1), b(i,j+1)) > \frac{\varepsilon}{\sqrt{2}}$. From Lemma 2, (i) implies that $error(i,j) \leq \varepsilon$ which proves that $P'$ is a $\varepsilon$-simplification of $P$. From lemma 2 again, (ii) implies that $error(i,j+1) > \frac{\varepsilon}{2\sqrt{2}}$. The hypothesis are then similar to the ones of the proof of [2, Theorem 3.4], and a similar reasoning proves the guarantee.

Note that the complexity of Algorithm 1 has not been adressed yet. Indeed, the difficulty lies in the updates of $\omega(i,j)$ and $b(i,j)$ when a new point $p_j$ is considered. These two variables depend directly on the direction $\overrightarrow{p_i p_j}$. However, when a new point is added, this direction may change drastically: Figure 2 shows an example where the maximal width and maximal backpath are achieved for different vertices of the polyline for $\overrightarrow{p_i p_j}$ and $\overrightarrow{p_i p_{j+1}}$.

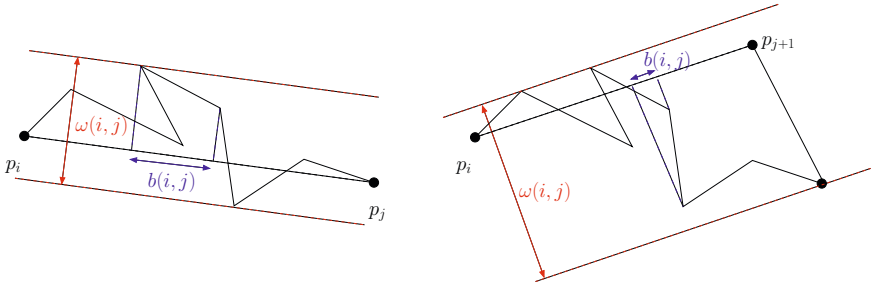



**Fig. 2.** Updating $\omega(i,j)$ and $b(i,j)$ can be costly in the general case: for instance, the longest backpath is much smaller in (b) for the direction $\overrightarrow{p_ip_{j+1}}$ than in (a) for the direction $\overrightarrow{p_ip_j}$

In the next section, we show how to update efficiently the decisions on $\omega(i,j)$ and $b(i,j)$, with a specification for digital curve to reach a near-linear time complexity.

## 3 Updating the Approximated Distance Efficiently

In [1], where the approximated distance is defined, an actual computation of the variables $\omega(i,j)$ and $b(i,j)$ is necessary since the problem is to minimize the error. However, as computing the exact values is too expensive, guaranteed approximations are updated when a new point is added. Contrary to [1], in our framework an update of these variables is not necessary, but the decisions "is $\omega(i,j) \leq \frac{\varepsilon}{\sqrt{2}}$ ?" and "is $b(i,j) \leq \frac{\varepsilon}{\sqrt{2}}$ ?" must be exact to ensure the computation of an $\varepsilon$-simplification. This section is devoted to the design of new algorithmic approaches to solve these two problems.

### 3.1 Decision on $\omega(i,j)$

Instead of deciding whether $\omega(i,j)$ is under a given threshold or not, we show that it is enough to check the distance between any point of $P(i,j)$ and the vector $\overrightarrow{p_ip_j}$.

*Property 1.* Let $d_{max}(i,j) = \max_{p \in P(i,j)} d(p, \overrightarrow{p_ip_j})$. We have $\frac{\omega(i,j)}{2} \leq d_{max}(i,j) \leq \omega(i,j)$.

This property is actually implicitely used in the proof of Lemma 2 in [1, Lemma 4.2]. The authors use the following inequalities to define the approximated distance (the parameters $(i,j)$ are ommited for the sake of lisibility):

$$\max(\frac{\omega}{2}, \frac{b}{2}) \leq \max(d_{max}, \frac{b}{2}) \leq error \leq \sqrt{2}\max(d_{max}, b) \leq \sqrt{2}\max(\omega, b)$$

In our framework, it is much easier to use $d_{max}$ instead of $\omega$ thanks to the algorithm of Chan and Chin [5, Lemmas 1 and 2]. Given an origin point $p_i$ and a set of points $P(i,j)$ we construct the set $S_{ij}$ of straight lines $l$ going through $p_i$ such that $\max_{p \in P(i,j)}(p,l) \leq r$.

To do so, we use the following simple fact: $d(p,l) \leq r \Leftrightarrow$ the straight line $l$ crosses the disk of center $p$ and radius $r$ (see Figure 3(a)). $S_{ij}$ is a cone computed incrementally in $\mathcal{O}(1)$ time considering for a point $p_k$ two new rays defined by $p_i$ and the disk of center $p_k$ and radius $r$ (see Figure 3(b)). As a result, deciding whether $d_{max}(i,j)$ is lower than $r$ or not is equivalent to checking whether the straight line $(p_i, p_j)$ belongs to $S_{ij}$ or not.

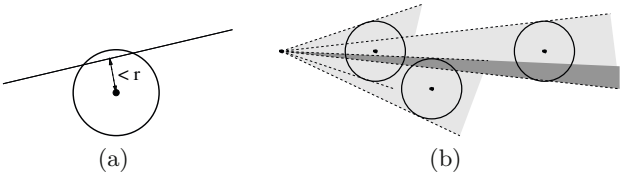



**Fig. 3.** (a) A line which is at a distance lower than $r$ from a point $p$ crosses the disk of center $p$ and radius $r$. (b) The cone $S_{ij}$ (dark gray) is computed incrementally as the intersection of the light gray cones.

### 3.2 Decision on $b(i,j)$

We first show that some particular points, named occulters, play a special role in the decision on $b(i,j)$. Then, we restrict the framework to digital curves to get a better complexity thanks to the following two facts: the computation of the occulters can be mutualized, and the number of occulters is bounded by the error.

*General considerations*

**Definition 4.** *A* occulter *for the direction* $\overrightarrow{d}$ *is a point* $p_k$ *such that for all* $l < k$, $proj_\alpha(p_k) >> proj_\alpha(p_l)$. *Moreover, an occulter is said to be* active *if there is no occulter* $p'_k$ *with* $k' > k$.

*Property 2.* There is one and only one active occulter for a given direction $\overrightarrow{d}$.

In other words, the active occulter of a curve $P(i,j)$ in the direction $\overrightarrow{d}$ is the point equal to $\arg\max(proj_\alpha(p))$. Figure 4 (a) illustrates this definition.

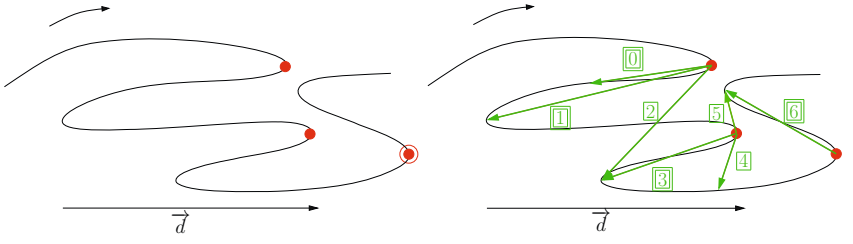
*Property 3.* The origin of the longest backpath of $P(i,j)$ is an occulter for the direction $\overrightarrow{p_i p_j}$.

*Proof.* Let $\overrightarrow{p_k p_m}$ be the longest backpath of $P(i,j)$ in the direction $\overrightarrow{p_i p_j}$. Suppose that $p_k$ is not an occulter. Then there exist a point $p_l$ of $P(i,j)$ such that $l < k$ and $proj_\alpha(p_l) >> proj_\alpha(p_k)$. Thus $\overrightarrow{p_l p_m}$ is a backpath too, and $||proj_\alpha(\overrightarrow{p_l p_m})|| > ||proj_\alpha(\overrightarrow{p_k p_m})||$, which is a contradiction with the fact that $\overrightarrow{p_k p_m}$ is the longest backpath.

*Property 4.* Consider the set of occulters $\{occ_j, j = 1 \dots n\}$ of $P(i,k)$. Let $occ_{max}$ be the active occulter. Consider all the backpathes $\overrightarrow{occ_j p_k}$ ending at $p_k$, if any. Then $||proj_\alpha(\overrightarrow{occ_{max} p_k})|| > ||proj_\alpha(\overrightarrow{occ_j p_k})||$ for all $j$.

*Proof.* By definition, $proj_\alpha(occ_{max}) >> proj_\alpha(occ_j)$ for all j, and the result follows straighforwardly.

Putting together the previous properties, we see that updating the active occulter is the key point of the computation of the longest backpath.




(a) (b)

**Fig. 4.** (a) Occulters for the direction $\vec{d}$. The only active occulter is circled. (b) Backpathes in the direction $\vec{d}$: the origin of all the backpathes is an occulter, but only the double-squared backpathes need to be considered in the algorithm.

Suppose that we are computing the backpathes in direction $\vec{d}$ for the curve $P(i,j)$. Suppose that all the points of $P(i,k)$ have been processed, and that the point $p_{k+1}$ is added. If $p_k p_{k+1}$ is a positive shift then nothing needs to be done:

- $p_{k+1}$ cannot be the origin of a backpath since it's the last point processed.
- $p_{k+1}$ is not the end of the longest backpath since $proj_\alpha(p_{k+1}) >> proj_\alpha(p_k)$.

The case when $p_k p_{k+1}$ is a negative shift is detailed in Algorithm 2. Figure 4(b) illustrates the difference cases of Algorithm 2. Only the double-squared backpathes are considered in the algorithm. Indeed, the backpathes number 2 and 5 are not taken into account because of Property 4. The backpath number 4 is not considered either because the end of the backpath follows a positive shift: we know for sure that the backpath number 3 is longer. However, note that the backpath number 0 is considered: the length of this backpath may be greater than the threshold.

According to Algorithm 1 we see that Algorithm 2 must be applied for any direction $\overrightarrow{p_l p_m}$ where $p_l$ and $p_m$ are any two points of the curve, with $l < m$. In the general case, for a polygonal curve of $n$ points, there are $\mathcal{O}(n^2)$ such directions, that can be computed as a preprocessing of the curve. In the case of a digital curve embedded in an image of size $n \times n$, the possible directions are known *a priori* but $\mathcal{O}(n^2)$ directions must still be considered. However, we see in the following that the computation of the backpathes can be mutualized in the case of digital curves.

**Algorithm 2.** Active occulter update and backpath computation for a direction $\overrightarrow{d}$

```
1 Let occ_max be the active occulter for P(i,k) in direction d.
2 if p_k p_{k+1} is negative then
3     if p_{k-1} p_k is positive then
4         if proj_α(p_k) >> proj_α(occ_max) then
5             occ_max = p_k
6     The vector occ_max p_{k+1} may be a backpath.
```

*Digital curves specificities.* In the following, we consider 8-connected digital curves, but the algorithm also works for 4-connected curves.

An elementary shift is a vector $\overrightarrow{p_k p_{k+1}}$. For a digital curve, there are only 8 elementary shifts, given by the chain codes, and denoted $\overrightarrow{e_i}, i = 0 \dots 7$ in the following. We also classify the directions $\overrightarrow{d} = (d_x, d_y)$ into 8 octants as illustrated in Figure 5.

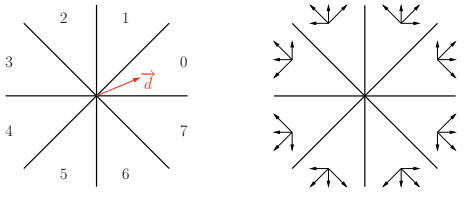



**Fig. 5.** (a) Clustering of the directions into octants: for instance, the direction $\overrightarrow{d}$ belongs to the octant 0. (b) Illustration of the elementary positive shifts for each octant.

**Lemma 4.** *Consider any elementary shift $\overrightarrow{e_i}$. Then, the sign of $\overrightarrow{d}.\overrightarrow{e_i}$ is the same for all the directions $\overrightarrow{d}$ of a given octant.*

*Proof.* An octant clusters all the directions with a fixed sign for $d_x$ and $d_y$, and such that the order on $d_x$ and $d_y$ is the same. For instance, in the octant 0, we find all the directions such that $0 \leq d_y < d_x$. Since the components $e_{i,x}$ and $e_{i,y}$ of $\overrightarrow{e_i}$ lie in $\{-1, 0, 1\}$, the sign of $\overrightarrow{d}.\overrightarrow{e_i}$ only depends on the signs and order of $d_x$ and $d_y$, which ends the proof.

As a consequence, for all the directions of a given octant the elementary positive and negative shifts are the same. In Figure 5(b), all the elementary positive shifts are depicted for each octant. Thus, the tests of Algorithm 2 lines 2-3 are the same for all the directions of a given octant. Nevertheless, on line 4, the projections of two points on a direction $\overrightarrow{d}$ are compared, and this test is not so easy when an octant of directions is considered. In the following the octant 0 is

considered, but a similar reasoning can be done for the other cases. Consider the Figure 6(a), where the plane is divided into four areas according to the position of a point $p$:

- for any point $q$ in the gray area, we have $proj_\alpha(q) >> proj_\alpha(p)$ for any direction of the octant 0;
- for any point $q$ in the dashed area, we have $proj_\alpha(p) >> proj_\alpha(q)$ for any direction of the octant 0;
- in the white area, the order of the projections changes, as illustrated in Figure 6(b-d).

Thus, we do not have only one active occulter to update anymore, but a set of occulters for each octant. Any active occulter is associated to an interval of angles for which it is actually the active occulter. From Property 2, we derive that these intervals are disjoint and that there union is the interval $[0, \frac{\pi}{4}]$. Algorithm 3 describes how to update the active occulters for the directions of octant 0.

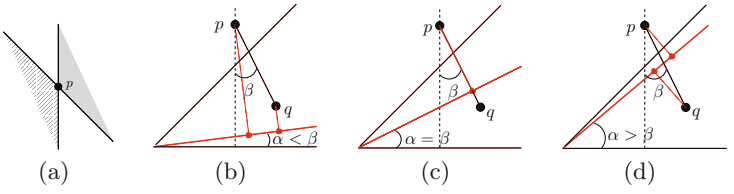



**Fig. 6.** When the point $q$ in the the white area (a), the order of the projections of $p$ and $q$ on the line of angle $\alpha$ changes according to the angle $\beta$ (b-d)

The complexity of Algorithm 3 depends on the number of active occulters. The following lemma is used in the complexity analysis in Section 4 to prove that the number of occulters per octant is bounded by the approximation error in the case of digital curves.

**Lemma 5.** *For a digital curve and for a given octant, there is at most one active occulter per line and per column of* $\mathbb{Z}^2$.

*Idea of the proof.* When two points $p$ and $q$ are on the same row or the same column, then for all the directions of a given octant, either $p$ is an occulter for $q$ or $q$ is an occulter for $p$.

*List of forbidden directions.* From Algorithm 1, we see that the length of the longest backpath is tested for each point, which defines a new direction. Moreover, we see from Algorithm 2 line 6 that for each negative shift, we can have as many backpathes as active occulters. All in all, testing individually all the possible backpathes when a new point is added is too costly. To solve this problem, we propose to maintain a "set" of the directions for which there exist a backpath of length greater than the error $\varepsilon$.

**Algorithm 3.** Update of the list of active occulters for the octant 0

Let $p$ be the last point added, we want to check if $p$ is an active occulter.
**forall** *the active occulters* $p_i(\alpha_{i_{min}}, \alpha_{i_{max}})$ **do**
  $v = \overrightarrow{p_i p}$
  **if** $\overrightarrow{v}.(1,0) < 0$ *and* $\overrightarrow{v}.(1,1) < 0$ **then**
    p is not an active occulter
  **if** $\overrightarrow{v}.(1,0) > 0$ *and* $\overrightarrow{v}.(1,1) > 0$ **then**
    $p_i$ is not an active occulter anymore
    $p$ is an active occulter on $[0, ?]$ with $\alpha_{i_{min}} <? \leq \frac{\pi}{4}$
  **if** $\overrightarrow{v}.(1,0) > 0$ *and* $\overrightarrow{v}.(1,1) < 0$ **then**
    compute the angle $\beta$ ; `/* see Figure 6 */`
    **if** $\alpha_{i_{min}} \leq \beta < \alpha_{i_{max}}$ **then**
      $p$ is an active occulter on $[0, \beta]$
      $p_i$ is an active occulter on $[\beta, \alpha_{i_{max}}]$
    **if** $\beta < \alpha_{i_{min}}$ **then**
      $p_i$ is still an active occulter
    **if** $\beta \geq \alpha_{i_{max}}$ **then**
      $p_i$ is not an active occulter anymore
      $p$ is an active occulter on $[0, ?]$ with $\alpha_{i_{max}} \leq ? \leq \frac{\pi}{4}$
  *+similar process for symmetrical cases (roles of* $p_i$ *and* $p$ *inverted)*

This set actually consists of a list of intervals defined as follows. Consider a backpath $\overrightarrow{p_k p_m}$ of length $l$. Then the length of the projection of this backpath on the direction $\overrightarrow{d}$ is a function of $l$ and the angle between $\overrightarrow{p_k p_m}$ and $\overrightarrow{d}$. This is illustrated in the Figure 7: for a given backpath of length $l$ and angle $\alpha$, and a given error $\varepsilon$, the interval of directions for which the projection of the backpath has a length greater than $\varepsilon$ is computed easily. Eventually, such an interval is computed for each backpath of length greater than $\varepsilon$, and the list of all these intervals is called the list of forbidden directions.

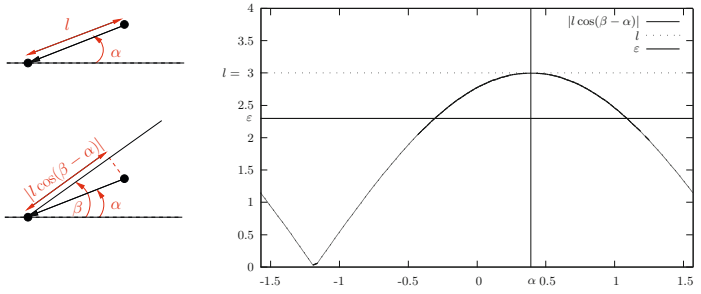
At the end, we have the following equivalence: $b(i, j)$ is greater than $\varepsilon$ if and only if the direction $\overrightarrow{p_i p_j}$ belongs to the list of forbidden directions. This equivalence enables to test efficiently $b(i, j)$ in $\mathcal{O}(\log(n_i))$ for $n_i$ intervals (see Algorithm 1).

## 4 Theoretical and Experimental Results

### 4.1 Complexity and Guarantee Analysis

**Theorem 2.** *Algorithm 1, combined with Algorithm 2 and Algorithm 3, compute in* $\mathcal{O}(n \log(n_i))$ *an* $\varepsilon$*-simplification of a digital curve* $P$ *under the Fréchet distance with at most the number of vertices of an optimal* $\frac{\varepsilon}{4\sqrt{2}}$*-simplification.*

*Proof.* The proof of the guarantee is given by Lemma 3. The complexity of the algorithm lies in: (i) the number of active occulters $n_o$ and (ii) the size of




(a) (b)

**Fig. 7.** (a) Illustration of the function defining the length of the projection of a back-path. (b) Plot of this function: when a threshold $\varepsilon$ is fixed, the interval of angles for which the length is greater than $\varepsilon$ is defined.

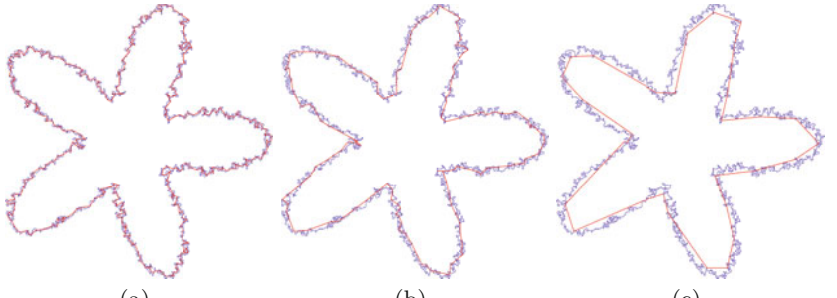
the list of intervals of forbidden directions $n_i$. Indeed, the general complexity is $\mathcal{O}(n(n_o + \log(n_i)))$. Nevertheless, putting together Lemma 5 and the fact that the width is bounded by the error, we get that $n_o$ is also upper bounded by the error, which ends the proof.

$n_i$ is more difficult to bound since one interval may be added after each negative shift, but we see in the following section that experimentally, the algorithm runs in linear time.

### 4.2 Experimental Behaviour

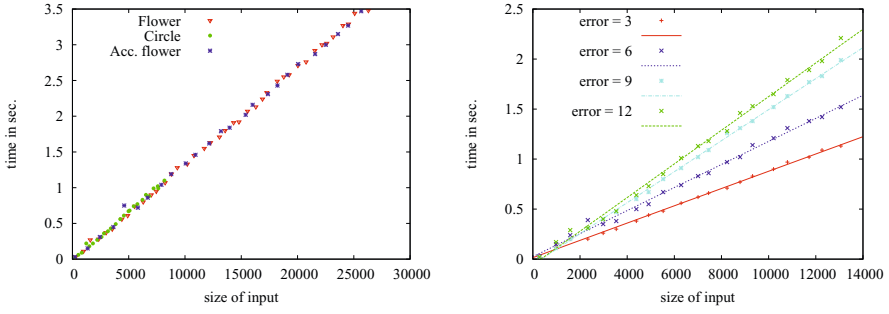
Figure 8 illustrates the results of our algorithm for a noisy flower with 5 extremities, for three different values of the $\varepsilon$ parameter. The images were generated with the Imagene toolkit [9].

In Figure 9(a), runtime results are depicted for noisy synthetic data of increasing size: a circle, a flower with 5 extremities, and a phase accelerating flower with 5 extremities. We see that the general behaviour is clearly linear in time. In



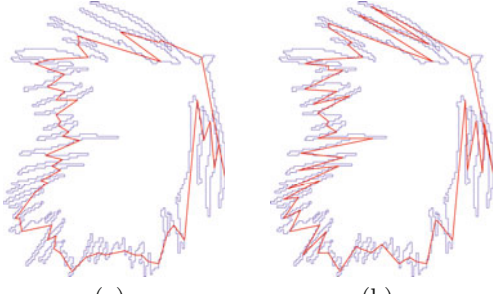
(a) (b) (c)

**Fig. 8.** Results of our algorithm on a noisy flower for a parameter $\varepsilon$ equal to 3 in (a), 8 in (b) and 15 in (c)




(a) (b)

**Fig. 9.** Runtime results for noisy synthetic shapes. In (b), a noisy flower with five extremities is used.



(a) (b)

**Fig. 10.** Comparison of the simplification results on a leaf, with an error equal to 8 with a criterion on the width only in (a), and approximated Frechet distance – width and backpath length – in (b)

Figure 9(b), we study the runtime for different values of $\varepsilon$ for noisy synthetic flowers of increasing size. Once again, we see a linear behaviour for any value of $\varepsilon$. A more detailed study of the evolution of the slope would give some hints to refine the theoretical complexity analysis.

Lastly, in Figure 10 we compare the result of the approximated Frechet simplification algorithm (where the width and the backpathes length are taken into account) with the result of a simplification algorithm with a width criterion only (points are added while $w(i, j)$ is lower than the error). We see that for the same error, even if the polygons returned by the two algorithms roughly have the same number of vertices (56 in (a), 60 in (b)), the sharp features of the shape are better preserved with the approximated Frechet distance.

## 5 Future Works

The first perspective is to refine the theoretical complexity from $\mathcal{O}(n \log(n))$ to $\mathcal{O}(n)$ since the experimental results show a linear behaviour. Another perspective

is to extend this algorithm (and its complexity) to general polygonal curves. Indeed the main idea of the algorithm is to cluster the directions used for the projection according to the directions of elementary shifts. This clustering is possible if the minimal angle between two elementary shifts directions is known, which is trivial in the case of digital curves, but could be computed as a preprocessing for polygonal curves.

## References


1. Abam, M.A., de Berg, M., Hachenberger, P., Zarei, A.: Streaming algorithms for line simplification. In: SoCG 2007, pp. 175–183. ACM, New York (2007)
2. Agarwal, P.K., Har-Peled, S., Mustafa, N.H., Wang, Y.: Near-linear time approximation algorithms for curve simplification. Algorithmica 42(3-4), 203–219 (2005)
3. Alt, H., Godau, M.: Computing the Fréchet distance between two polygonal curves. International Journal of Computational Geometry 5(1), 75–91 (1995)
4. Buzer, L.: Optimal simplification of polygonal chains for subpixel-accurate rendering. Comput. Geom. Theory Appl. 42(1), 45–59 (2009)
5. Chan, W.S., Chin, F.: Approximation of polygonal curves with minimum number of line segments. In: Ibaraki, T., Iwama, K., Yamashita, M., Inagaki, Y., Nishizeki, T. (eds.) ISAAC 1992. LNCS, vol. 650, pp. 378–387. Springer, Heidelberg (1992)
6. Driemel, A., Har-Peled, S., Wenk, C.: Approximating the fréchet distance for realistic curves in near linear time. In: SoCG 2010, pp. 365–374. ACM, New York (2010)
7. Godau, M.: A natural metric for curves—computing the distance for polygonal chains and approximation algorithms. In: Jantzen, M., Choffrut, C. (eds.) STACS 1991. LNCS, vol. 480, pp. 127–136. Springer, Heidelberg (1991)
8. Imai, H., Iri, M.: Polygonal approximations of a curve: formulations and algorithms. In: Computational Morphology, pp. 71–86. Elsevier, Amsterdam (1988)
9. Lachaud, J.O.: ImaGene, `https://gforge.liris.cnrs.fr/projects/imagene/`
10. Pelletier, S.: Computing the Fréchet distance between two polygonal curves, `http://www.cim.mcgill.ca/~stephane/cs507/Project.html`
11. Sriraghavendra, E., Karthik, K., Bhattacharyya, C.: Fréchet distance based approach for searching online handwritten documents. In: ICDAR 2007: Int. Conference on Document Analysis and Recognition, pp. 461–465. IEEE Computer Society, Los Alamitos (2007)