On the local properties of digital curves

Thanh Phuong Nguyen, Isabelle Debled-Rennesson

International Journal of Shape Modeling

On the local properties of digital curves *†

Thanh Phuong Nguyen‡and Isabelle Debled-Rennesson
LORIA, Campus Scientifique - BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France
{*nguyentp,debled*}*@loria.fr*

We propose a geometric approach to extract local properties of digital curves. This approach uses the notion of blurred segment [1] that extends the definition of segment of arithmetic discrete line[2] to adapte to noisy curves. A curvature estimator [3] for 2D curves in $O(n \log^2 n)$ time is proposed relying on this flexible approach. The notion of 2D blurred segment is extended to 3D space. A decomposition of the curve into 3D blurred segments is deduced and allows new curvature and torsion estimators for 3D curves. All these estimators can naturally work with disconnected curves.

Keywords: curvature, torsion, digital curve, discrete line, blurred segment

1. Introduction

Geometric properties of curves are important characteristics to be exploited in geometric processing. They directly lead to applications in machine vision, computer graphics. So curvature for 2D curves, curvature and torsion for 3D curves are interesting subjects to study digital curves.

In the 2D case, many applications are based on the curvature property in domains such as curve approximation [4], geometry compression [5], and particularly in corner detection after the pioneer paper of Attneave [6]. Curvature estimation is a key problem for many applications in image processing that require the geometric measures of represented discrete objects.

In 3D space, torsion and curvature are the most important properties that permit to study the bending of a spatial curve. Several methods have been proposed for torsion estimation. Mokhtarian [7] used Gaussian smoothing to estimate torsion directly with a torsion formula. Similarly, Kehtarnavaz et al. [8] used B-spline smoothing techniques; Lewiner et al. [5] proposed weighted least-squares fitting techniques. Raluben Medina et al. [9] proposed two methods to estimate torsion and curvature values at each point of the curve. The first used the Fourier transform, the second is based on the least squares fitting. These methods are applied in the description of arteries in medical imaging.

In the framework of the discrete geometry, estimators of geometrical parameters have been proposed, but these methods rely on the recognition of discrete line

*This work is supported by ANR in the framework of the GEODIB project, $BLAN06-2\_134999$.
†Based on "Curvature Estimation in Noisy Curves" and "Curvature and Torsion Estimators for 3D Curves" by T.P. Nguyen and I. Debled-Rennesson respectively in proceedings of CAIP'07 and ISVC'08.
‡Corresponding author

segments which is very sensitive to the noise present in the studied curves [10,11,12]. The boundary of discrete objects is often noisy due to the acquisition process. Therefore the concept of blurred segment was introduced [1], which allows the flexible segmentation of discrete curves, taking noise into account. Relying on an arithmetic definition of discrete lines [2], it generalizes such lines, admitting that some points are missing.

We propose in this paper a novel method, based on the definition of blurred segments, for the estimation of local geometric parameters of 2D and 3D curves. It uses a geometrical approach and relies on results of discrete geometry on decomposition of a curve into maximal blurred segments [10,1,3]. This paper recalls the obtained results for a 2D curvature estimator [3] and presents an extension to 3D of these results. The 3D curvature estimator given in [13] is extended with the notion of blurred segment and permits to study noisy or disconnected curves. We also propose a new approach to the discrete torsion estimation.

The paper is organized as follows. In Section 2, after recalling some definitions related to 2D blurred segments, we study the problem of adding (or removing) a point to (from) a 2D blurred segment of width ν in the case of general discrete curves. Then we propose an extension for noisy curves the notion of maximal segment of a discrete curve. An algorithm to determine all maximal blurred segments of a 2D discrete curve is given in Section 3. In Section 4, after recalling the definition of the curvature estimator adapted to 2D noisy curves, an algorithm for the determination of the curvature at each point of a discrete curve is proposed. In this section, we also present how to extend these ideas into the 3D space. The next sections propose curvature and torsion estimators for 3D curves. The last section gives experiments and comparisons with Mokhtarian's and Lewiner's methods.

2. Blurred segment of width ν

2.1. *Definitions*

2.1.1. *2D case*

The notion of blurred segments relies on the arithmetical definition of discrete lines [2]. A line, with slope $\frac{a}{b}$, lower bound μ and thickness ω (with a, b, μ and ω being integer such that $gcd(a,b) = 1$) is the set of integer points (x, y) verifying $\mu \leq ax - by < \mu + \omega$. Such a line is denoted by $\mathcal{D}(a, b, \mu, \omega)$. Let us recall definitions [1] that we use in this paper (see Fig. 1.a):

Definition 2.1. Let us consider a set of 8-connected points $\mathcal{S}_b$. The discrete line $\mathcal{D}(a, b, \mu, \omega)$ is said ***bounding*** for $\mathcal{S}_b$ if all points of $\mathcal{S}_b$ belong to $\mathcal{D}$.

Definition 2.2. Let us consider a set of 8-connected points $\mathcal{S}_b$. A bounding line of $\mathcal{S}_b$ is said ***optimal*** if its vertical distance (i.e. $\frac{\omega-1}{max(|a|,|b|)}$) is minimal, i.e. if its vertical distance is equal to the vertical distance of $conv(\mathcal{S}_b)$, the convex hull of $\mathcal{S}_b$.

Definition 2.3. A set $\mathcal{S}_b$ is a ***2D blurred segment of width*** ν if its optimal bounding line has a vertical distance less than or equal to ν i.e. if $\frac{\omega-1}{max(|a|,|b|)} \leq \nu$.

A linear recognition algorithm of the 2D blurred segment of width ν is proposed in [1].

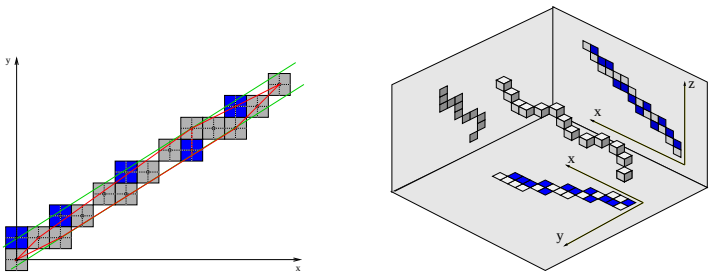


Fig. 1. From left to right: a. $\mathcal{D}(5, 8, -8, 11)$, optimal bounding line (vertical distance = $\frac{10}{8} = 1.25$) of the sequence of gray points - b. $\mathcal{D}_{3D}(45, 27, 20, -45, -81, 90, 90)$ optimal discrete line of the grey points.

2.1.2. *3D case*

The notion of 3D discrete line (see the [14,15]) is defined as follows:

Definition 2.4. A ***3D discrete line*** [43], denoted $\mathcal{D}_{3D}(a, b, c, \mu, \mu', e, e')$, with a main vector (a, b, c) such that $(a, b, c) \in \mathbb{Z}^3$, and $a \geq b \geq c$ is defined as the set of points (x, y, z) from $\mathbb{Z}^3$ verifying:

$$\mathcal{D} \begin{cases} \mu \leq cx - az < \mu + e & (1) \\ \mu' \leq bx - ay < \mu' + e' & (2) \end{cases}$$

with $\mu, \mu', e, e' \in \mathbb{Z}$. e and e' are called arithmetical width of $\mathcal{D}$.

According to the definition, it is obvious that a 3D discrete line is bijectively projected into two projection planes as two 2D arithmetical discrete lines. Thanks to that property, we naturally define the notion of 3D blurred segment by using the notion of 2D blurred segment and by considering the projections of the sequence of studied points in the coordinate planes (see Fig. 1.b)

Definition 2.5. Let $\mathcal{S}f_{3D}$ be a sequence of points of $\mathbb{Z}^3$, $\mathcal{S}f_{3D}$ is a ***3D blurred segment of width*** ν with a main vector (a, b, c) such that $(a, b, c) \in \mathbb{Z}^3$, and $a \geq b \geq c$ if it possesses a said optimal discrete line, named $\mathcal{D}_{3D}(a, b, c, \mu, \mu', e, e')$, such that

- $\mathcal{D}(a, b, \mu', e')$ is optimal for the sequence of projections of points of $\mathcal{S}f_{3D}$ in the plane (O, x, y) and $\frac{e'-1}{max(|a|,|b|)} \leq \nu$,
- $\mathcal{D}(a, c, \mu, e)$ is optimal for the sequence of projections of points of $\mathcal{S}f_{3D}$ in the plane (O, x, z) and $\frac{e-1}{max(|a|,|c|)} \leq \nu$.

A ***linear algorithm of 3D blurred segment recognition*** may be deduced from that definition. Indeed, we only need to use an algorithm of 2D blurred segment recognition in each projection plane (for example, the one describe in [1]) .

2.2. *Add (or remove) a point to (from) a 2D blurred segment of width ν*

In this section we study the problem of adding (or removing) a point to (from) a blurred segment of width ν. The recognition algorithm of 2D width ν blurred segments presented in [1] is executed in linear time. However, it only considers the incremental addition of a point to a blurred segment in the first octant. We present here the general case which requires the incremental calculation of both height and width of the convex hull after adding or removing a point. To do that, we use the results given in [16] and [17] that we briefly recall below.

Dynamic estimation of the convex hull:
The problem of dynamic estimation of the convex hull of a set of points when adding (or removing) a point to (from) this set was proposed by M.H. Overmars and J. van Leeuwen [16]. The convex hull is represented by Concatenable Queue data structure that support search, insert, removal, split and concatenate operations in $O(\log n)$ time[18]. A segment tree data structure was proposed to allow to work with convex hull based on the divide-and-conquer strategy. This strategy is based on the fact: it costs $O(\log n)$ time to determine the bridge between 2 convex hulls. In their work, a convex hull is considered as the union of two parts: the upper convex hull (U_{hull}) and the lower convex hull (L_{hull}) which correspond to 2 segment trees. They are updated after each operation of addition or removal of a point. The cost of these operations are estimated by the following theorem [16].

Theorem 1. *The convex hulls U_{hull} and L_{hull} of the set S of n points may be dynamically kept, in the worst case, in $O(\log^2 n)$ by an operation of addition or removal.*

Determination of height and width of the convex hull:
We use the double technique of binary search [17] to determine the height and width of the convex hull. In [17], the convex hull is also considered as the union of two parts U_{hull} and L_{hull}. The double technique of binary search permits to find the vertical width of the convex hull by using the concavity property of the function $height(x) = U_{hull}(x) - L_{hull}(x)$ in $O(\log^2 n)$. To do that, firstly, for each point in the upper convex hull, he applied binary search technique to detemine its opposite edge on the lower convex hull. So the height from this point to the lower hull is determined. By applying one time the binary search technique on the upper convex hull, the maximal height between the upper and lower convex hulls will be determined. So, the complexity of this double technique is $O(\log^2 n)$.

3. Maximal blurred segment of width ν

3.1. *Definitions and first proposition*

The notion of the maximal segment of a discrete curve was proposed in [10,12] and relies on the discrete line segments. This structure enables a global understanding of the discrete curve to be analyzed. We propose here an extension of that notion to blurred segments, adapted to noisy curves, by using the same notation as in [12].

Let us consider a 2D or 3D discrete curve called C, the points of C are indexed from 0 to $n-1$. C ***is a general curve and the points of*** C ***can be disconnected***. We note $C_{i,j}$ a set of successive points of C ordered increasingly from index i to j.

Definition 3.1. The predicate "$C_{i,j}$ is a blurred segment of width ν" is denoted

by $BS(i, j, \nu)$. The first index j, $i \leq j$, such that $BS(i, j, \nu)$ and $\neg BS(i, j+1, \nu)$ is called the front of i and noted $F(i)$. Symmetrically, the first index i such that $BS(i, j, \nu)$ and $\neg BS(i-1, j, \nu)$ is called the back of j and noted $B(j)$.

Definition 3.2. $C_{i,j}$ is called a **maximal blurred segment of width** ν and noted $MBS(i, j, \nu)$ iff $BS(i, j, \nu)$ and $\neg BS(i, j+1, \nu)$ and $\neg BS(i-1, j, \nu)$.

It is obvious that an equivalent characterization for a maximal blurred segment of width ν, $MBS(i, j, \nu)$, is to show that $F(i) = j$ and $B(j) = i$. In this work, we use the notion of blurred segment of width ν which is maximal on the right or on the left sides:

Definition 3.3. $C_{i,j}$ is called a **maximal blurred segment of width ν on the right side (resp. on the left side)** and noted $MBS_R(i, j, \nu)$ (resp. $MBS_L(i, j, \nu)$) if $F(i) = j$ (resp. $B(j) = i$).

Proposition 1. *Let C be a discrete curve, $MBS_\nu(C)$ the sequence of maximal blurred segments of width ν of the curve C. Then, $MBS_\nu(C) = \{MBS(B_1, E_1, \nu), MBS(B_2, E_2, \nu), ..., MBS(B_m, E_m, \nu)\}$ and satisfies $B_1 < B_2 < ... < B_m$. So we have: $E_1 < E_2 < ... < E_m$.*

Proof: We consider 2 consecutive maximal blurred segments $MBS(B_i, E_i, \nu)$ and $MBS(B_{i+1}, E_{i+1}, \nu)$. By hypothesis, $B_i < B_{i+1}$, let us suppose that $E_i > E_{i+1}$, then $MBS(B_{i+1}, E_{i+1}, \nu)$ becomes a part of $MBS(B_i, E_i, \nu)$. Therefore $MBS(B_{i+1}, E_{i+1}, \nu)$ is not a maximal blurred segment, which is contradictory.

3.2. *Algorithm for the segmentation of a curve C into maximal blurred segments*

3.2.1. *2D case*

We propose the algorithm 1 which determines all maximal blurred segments of width ν of a 2D discrete curve C according to the conditions given in section 3.1 by using proposition 1.

Complexity

Each point of the curve is scanned at most twice in this algorithm. The cost of determining a new optimal bounding discrete line when we add (or remove) a point to (from) a blurred segment is in $O(\log^2 n)$. Hence the complexity of this algorithm is in $O(n \log^2 n)$.

3.2.2. *3D case*

The algorithm 2 permits to obtain the sequence of 3D maximal blurred segments of width ν in time $O(n \log^2 n)$ for any noisy 3D discrete curve $\mathcal{C}$. It uses the algorithm 1 to determine the 2D maximal blurred segments of the projections in the coordinate planes of the points of the studied curve.

To determine the optimal discrete line of the current 3D blurred segment S_b (step marqued with (*) in the algoritm 2), we consider the characteristic of the two 2D blurred segments obtained in the planes of projection and combine them to obtain the characteristics of the optimal 3D discrete line of S_b. As the whole process is done in dimension 2, this algorithm 2 has the same complexity as the one in dimension 2. So, we have the theorem below.

Algorithm 1: Algorithm for the segmentation of a curve C into 2D maximal blurred segments of width ν

Data: C - discrete curve with n points, ν - width of the segmentation
Result: MBS_ν - the sequence of maximal blurred segments of width ν
begin
  k=0; $S_b = \{C_0\}$; $MBS_\nu = \emptyset$; $a = 0$; $b = 1$; $\omega = b$, $\mu = 0$;
  while $\frac{\omega-1}{max(|a|,|b|)} \leq \nu$ **do**
    k++; $S_b = S_b \cup C_k$; Determine $D(a, b, \mu, \omega)$ of S_b;
  bSegment=0; eSegment=k-1 ;
  $MBS_\nu = MBS_\nu \cup MBS(bSegment, eSegment, \nu)$;
  while $k < n - 1$ **do**
    while $\frac{\omega-1}{max(|a|,|b|)} > \nu$ **do**
      $S_b = S_b \setminus C_{bSegment}$; bSegment++ ;
      Determine $D(a, b, \mu, \omega)$, optimal bounding line of S_b;
    while $\frac{\omega-1}{max(|a|,|b|)} \leq \nu$ **do**
      k++ ; $S_b = S_b \cup C_k$;
      Determine $D(a, b, \mu, \omega)$, optimal bounding line of S_b;
    eSegment=k-1; $MBS_\nu = MBS_\nu \cup MBS(bSegment, eSegment, \nu)$;
end

Algorithm 2: Algorithm for the segmentation of a curve C into maximal 3D blurred segments of width ν

Data: $\mathcal{C}$ - discrete curve with n points, ν - width of the segmentation
Result: MBS_ν - the sequence of maximal blurred segments of width ν of $\mathcal{C}$
begin
  k=0; $S_b = \{C_0\}$; $MBS_\nu = \emptyset$; $a = 0$; $b = 1$; $\omega = b$, $\mu = 0$;
  while *the widths of 2 blurred segments obtained by projecting the points of* S_b *in the coordinate planes are* $\leq \nu$ **do**
    k++; $S_b = S_b \cup C_k$;
    Determine $\mathcal{D}_{3D}(a, b, c, \mu, \mu', e, e')$, optimal discrete line of S_b; (*)
  bSegment=0; eSegment=k-1 ;
  $MBS_\nu = MBS_\nu \cup MBS(bSegment, eSegment, \nu)$;
  while $k < n - 1$ **do**
    while *the widths of 2 blurred segments obtained by projecting the points of* S_b *in the coordinate planes are* $> \nu$ **do**
      $S_b = S_b \setminus C_{bSegment}$; bSegment++ ;
      Determine $\mathcal{D}_{3D}(a, b, c, \mu, \mu', e, e')$, optimal discrete line of S_b; (*)
    while *the widths of 2 blurred segments obtained by projecting the points of* S_b *in the coordinate planes are* $\leq \nu$ **do**
      k++ ; $S_b = S_b \cup C_k$;
      Determine $\mathcal{D}_{3D}(a, b, c, \mu, \mu', e, e')$, optimal discrete line of S_b; (*)
    eSegment=k-1; $MBS_\nu = MBS_\nu \cup MBS(bSegment, eSegment, \nu)$;
end

Theorem 2. *The decomposition of a 3D curve into maximal blurred segments of width ν can be done in time $O(nlog^2n)$.*

4. Discrete curvature of width ν

4.1. *Definition*

We recall here the curvature estimator which is adapted to noisy curves [19]. It is directly deduced from the estimator proposed by D. Coeurjolly [11] for 2D curves without noise. This technique can be seen as a generalization of the classical *order m normalized curvature* [20]. Let C be a 2D or 3D discrete curve, C_k is a point of the

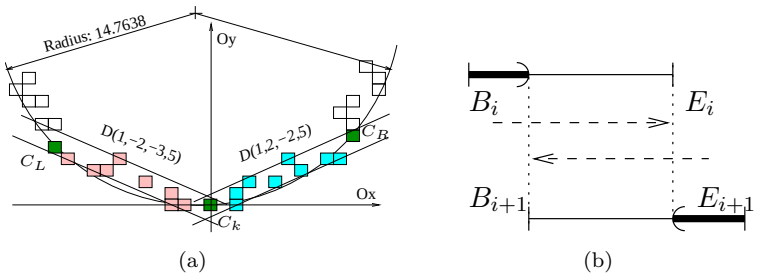


Fig. 2. a. Estimation of the 2D curvature at the point C_k with width 2; b. E_i (B_{i+1}) is front (back) of points in first (second) bold edge.

curve. Let us consider the points C_l and C_r of C such that : $l < k < r$, $BS(l, k, \nu)$ and $\neg BS(l-1, k, \nu)$, $BS(k, r, \nu)$ and $\neg BS(k, r+1, \nu)$.

The estimation of the **curvature of width ν at the point** C_k shall be determined as the inverse of the radius of the circle passing through the points C_l, C_k and C_r. To determine the radius $R_\nu(C_k)$ of the circumcircle of the triangle $[C_l, C_k, C_r]$, we use the formula given in [21] as follows (see Fig. 2.a and Fig. 3).

Let $s_1 = ||\overrightarrow{C_kC_r}||$, $s_2 = ||\overrightarrow{C_kC_l}||$ and $s_3 = ||\overrightarrow{C_lC_r}||$, then

$$R_\nu(C_k) = \frac{s_1 s_2 s_3}{\sqrt{(s_1+s_2+s_3)(s_1-s_2+s_3)(s_1+s_2-s_3)(s_2+s_3-s_1)}}$$

Then, the curvature of width ν at the point C_k is $\mathcal{C}_\nu(C_k) = \frac{s}{R_\nu(C_k)}$ with $s = sign(\det(\overrightarrow{C_kC_r}, \overrightarrow{C_kC_l}))$ (it indicates concavities and convexities of curve).

As indicated in [11], the degenerated cases, which correspond for example to colinear half-tangents, may be independently tested and, thus, a null curvature is affected to the considered point.

4.2. *Estimation of the curvature of width ν at each point of C*

In this section, we propose a new algorithm for the determination of ***the curvature of width*** ν at each of the n points of a 2D or 3D curve C. The complexity of this algorithm is better than the one of the naive algorithm, in $O(n^2)$. It consists of calculating at each point C_k, the maximal blurred segment on the right side, MBS_R, the maximal blurred segment on the left side, MBS_L, and then the circle

passing through 3 points: (left extremity of MBS_L, C_k, right extremity of MBS_R).

Description of the algorithm (see Fig. 3)
Let $MBS_R(k, r, \nu)$ and $MBS_L(l, k, \nu)$ be the maximal blurred segments on the right and left sides of the point C_k. Then, there exist $r' \leq k$ and $l' \geq k$ such that $MBS_R(k, r, \nu) \subset MBS(r', r, \nu)$ and $MBS_L(l, k, \nu) \subset MBS(l, l', \nu)$.

Let us then consider the decomposition of C into maximal blurred segments: $MBS_\nu(C) = \{MBS(B_1, E_1, \nu), MBS(B_2, E_2, \nu), ..., MBS(B_m, E_m, \nu)\}$ with $B_1 < B_2 < ... < B_m$ and $E_1 < E_2 < ... < E_m$. We look for the indices i and j such that i is the first index such that $E_i \geq k$ and j is the last index such that $B_j \leq k$. So it is obvious that $l = B_i$, $r = E_j$ and that the curvature of width ν at the point C_k is the inverse of the radius of the circumcircle of the triangle $[C_l, C_k, C_r]$. More generally, we have the following result.

Definition 4.1. Let $L(k)$, $R(k)$ be the functions which respectively represent the indices of the left and right extremities of the maximal blurred segments on the left and right sides of the point C_k.

- $\forall k$ such that $E_{i-1} < k \leq E_i$, then $L(k) = B_i$
- $\forall k$ such that $B_i \leq k < B_{i+1}$, then $R(k) = E_i$

This definition is used in the algorithm 3 (see also Fig. 2.b).

Algorithm 3: Width ν curvature estimation at each point of a 2D or 3D curve

Data: $\mathcal{C}$ Discrete curve of n points, ν width of the segmentation
Result: $\{\mathcal{C}_\nu(C_k)\}_{k=0..n-1}$ - Curvature of width ν at each point of $\mathcal{C}$
begin
  Build $MBS_\nu = \{MBS_i(B_i, E_i, \nu)\}_{i=0\ to\ m-1}$ (see Algorithm 1 for 2D case or Algorithm 2 for 3D case); $m = |MBS_\nu|$; $E_{-1} = -1$; $B_m = n$;
  for $i = 0$ *to* $m - 1$ **do**
    for $k = E_{i-1} + 1$ *to* E_i **do** $L(k) = B_i$;
    for $k = B_i$ *to* $B_{i+1} - 1$ **do** $R(k) = E_i$;
  for $i = 0$ **to** $n - 1$ **do**
    $R_\nu(C_i) =$ Radius of the circumcircle to $[C_{L(i)}, C_i, C_{R(i)}]$;
    $\mathcal{C}_\nu(C_i) = \frac{sign(\det(\overrightarrow{C_iC_{R(i)}}, \overrightarrow{C_iC_{L(i)}}))}{R_\nu(C_i)}$;
end

Remark: The bounds mentioned in the algorithm 3 are correct for a closed curve. In the case of an open curve, the instruction becomes: *for* $i = l$ *to* n - *1* - l with l fixed to a constant value. Indeed it is not possible to calculate a maximal blurred segment on the left side (resp. on the right side) at the first point (resp. at the last point) of the curve. Thus the calculation of the curvature begins (resp. stops) at the l^{th} (resp. $(n - 1 - l)^{th}$) point of the curve.

Complexity
Both steps of labelling and estimation of the curvature at each point are executed in linear time. However, the determination of the maximal blurred segments are executed in $O(n \log^2 n)$. Thus the complexity of our method is $O(n \log^2 n)$. Because the complexity of a blurred segment is $O(n)$ for simple curves, and $O(n \log n)$ for general curve (see [17]), the existing method[4] for curvature estimation has $O(n^2)$

complexity with simple curve and $O(n^2 \log n)$ for general curve. Let us recall that a simple curve is a polygonal chain of line segments that do not cross each other. It is not correct for general curve. So, this algorithm is more efficient than existing method.

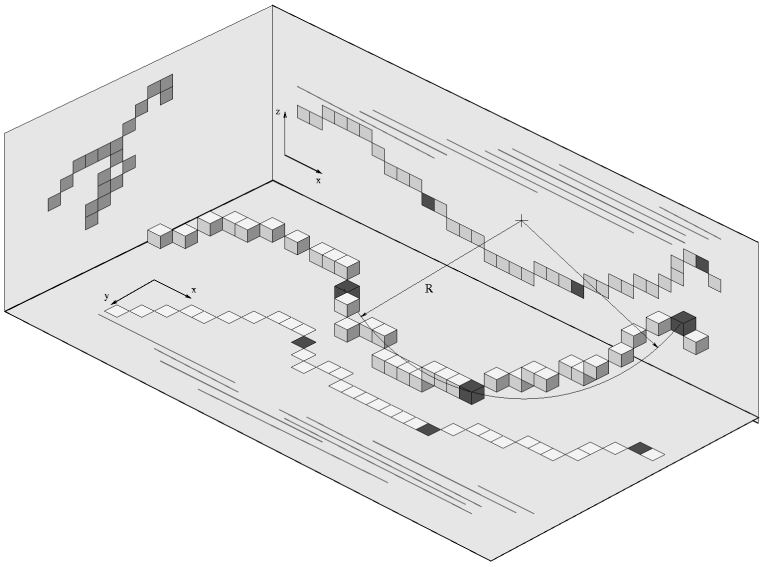


Fig. 3. Given a spatial curve, firstly the set of maximal blurred segments is computed. To determine the curvature value at the second black point, its left (resp. right) extremity is located as the left (resp. right) extremity of the corresponding maximal blurred segment, and then the curvature value is estimated as the inverse of circumcircle radius. Working width is 2.

5. Discrete torsion of width ν

5.1. *Definitions*

The curvature is not sufficient to characterize the local properties of a 3D curve. This parameter only measures how rapidly the direction of the curve changes. In case of a planar curve, the osculating plane does not change. For 3D curves, torsion is a parameter that measures how rapidly the osculating plane changes. To clarify this notion, we recall below some definitions and results in differential geometry (see the [22] for more details).

Definition 5.1. Let $r : I \to \mathbb{R}^3$ be a regular unit speed curve parameterized by t.

[i] *$T(t)$ (resp. $N(t)$) is a unit vector in direction $r^{'}(t)$ (resp. $r^{''}(t)$). So, $N(t)$ is a normal vector to $T(t)$. $T(t)$ (resp. $N(t)$) is called the unit* ***tangent vector*** *(resp.* ***normal vector****) at t.*

[ii] *$k(t) = |T^{'}(t)|$ is called the* ***curvature*** *of r at t.*

[iii] *The plane determined by the unit tangent and normal vectors ($T(t)$ and $N(t)$), is called the* ***osculating plane*** *at t. The unit vector $B(t) = T(t) \wedge N(t)$ is normal to the osculating plane and is called the* ***binormal vector*** *at t.*

[iv] $\tau(t) = |B^{'}(t)|$ *is called the* ***torsion*** *of curve at t.*

Definition 5.2. Let $r : I \rightarrow \mathbb{R}^3$ be a spatial curve parameterized by t.

i The curvature of r at $t \in I$: $k(t) = \frac{|r^{'} \wedge r^{''}|}{|r^{'}|^3}$

ii The torsion of r at $t \in I$: $\tau(t) = \frac{(r^{'} \wedge r^{''}).r^{'''}}{|r^{'} \wedge r^{''}|^2}$

According to the definition 5.2, the torsion value at a point is 0 if the curvature value at this point is 0.

5.2. *Discrete torsion*

Discrete torsion was studied in [5,7,9,8]. In this section, we propose a new geometric approach for the problem of torsion estimation that uses the definitions and results presented in the previous sections.

5.2.1. Definitions

Let ζ be a 3D discrete curve, C_k is k^{th} point of the curve. Let us consider the points C_l and C_r of ζ such that : $l < k < r$, $BS(l,k,\nu)\&\neg BS(l-1,k,\nu)$ and $BS(k,r,\nu)\&\neg BS(k,r+1,\nu)$. Recall that the curvature of width ν is estimated by circumcircle of triangle $\triangle C_l C_k C_r$. If $\overrightarrow{C_k C_l}$ and $\overrightarrow{C_k C_r}$ are colinear, the curvature value at C_k is 0, therefore the torsion value at C_k is 0. So, without loss of generality, we suppose that $\overrightarrow{C_l C_k}$ and $\overrightarrow{C_k C_r}$ are not collinear. In addition, the plane defined by $\overrightarrow{C_l C_k}$ and $\overrightarrow{C_k C_r}$ is noted (C_l, C_k, C_r), and we propose the definition below.

Definition 5.3. The ***osculating plane of width*** ν at C_k is the plane (C_l, C_k, C_r).

The osculating plane (C_l, C_k, C_r) has two unit tangent vectors : $\overrightarrow{t1} = \frac{\overrightarrow{C_l C_k}}{|\overrightarrow{C_k C_r}|}$ and $\overrightarrow{t2} = \frac{\overrightarrow{C_k C_r}}{|\overrightarrow{C_k C_r}|}$. Therefore, we have the binormal vector at the k^{th} point: $\overrightarrow{b_k} = \overrightarrow{t1} \wedge \overrightarrow{t2} = (b_x, b_y, b_z)$ So, we propose the following definition of discrete torsion of width ν.

Definition 5.4. The ***discrete torsion of width*** ν at C_k is the derivative of $\overrightarrow{b_k}$.

5.2.2. Torsion estimator

Our proposed method for torsion estimation is based on the definition 5.4. Let us remark that the set $\{\overrightarrow{b_k}\}_{k=0}^{n-1}$ can be constructed from the set of maximal blurred segments in $O(n \log^2 n)$ time. So, we can obtain the torsion value by calculating the derivative at each position of $\{\overrightarrow{b_k}\}_{k=0}^{n-1}$. The traditional method for derivative estimation of discrete sequences is Gaussian kernel [23]. We propose by the use of a geometric approach method to solve this problem.

Let us consider the curve $\zeta_1 = \{P\}_{i=0}^{n}$ that is constructed by this rule: $\overrightarrow{P_i P_{i+1}} = \overrightarrow{b_i}$, $i = 0, .., n-1$ (see Fig. 4).

Proposition 2. *The estimation of tangent vector at each point* P_i *of the curve* ζ_1 *is* $\overrightarrow{b_i}$ *(*$i = 0, .., n-1$*).*

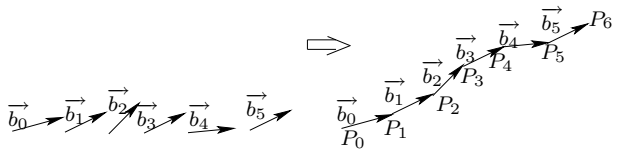


Fig. 4. The curve ζ_1 is constructed from the sequence of binormal vectors.

Proof. In differential geometry, the tangent vector of a curve $r(t)$ at the point $P_{t_0} = r(t_0)$ is defined as: $t(t_0) = r^{'}(t_0) = \lim_{h\to 0} \frac{r(t_0+h)-r(t_0)}{h} = \lim_{h\to 0} \frac{\overrightarrow{P_{t_0}P}}{h}$. Therefore, in discrete space, the tangent vector at the point $P_i = \alpha(i)$ can be estimated as $t(i) = \frac{r(i+1)-r(i)}{1} = \frac{\overrightarrow{P_iP_{i+1}}}{1} = \overrightarrow{P_iP_{i+1}} = \overrightarrow{b_i}$.

Proposition 3. *The torsion value at each point of the 3D discrete curve ζ corresponds to the curvature value of the curve ζ_1.*

Proof. Thanks to definition 5.4, the discrete torsion at C_k of ζ curve is the derivative of $\overrightarrow{b_k}$. In addition, $\overrightarrow{b_k}$ is the tangent vector at the k^{th} point of ζ_1 curve. So,this value is also curvature value at the k^{th} point of ζ_1 curve.

Algorithm 4: Width ν torsion estimation at each point of ζ

Data: ζ 3D discrete curve of n points, ν width of the segmentation
Result: $\{\mathcal{T}_\nu(C_k)\}_{k=0..n-1}$ - Torsion of width ν at each point of ζ
begin
  Build $MBS_\nu = \{MBS_i(B_i, E_i, \nu)\}_{i=0\ to\ m-1}$;
  $m = |MBS_\nu|$; $E_{-1} = -1$; $B_m = n$;
  for $i = 0$ *to* $m-1$ **do**
    for $k = E_{i-1}+1$ *to* E_i **do** $L(k) = B_i$;
    for $k = B_i$ *to* $B_{i+1}-1$ **do** $R(k) = E_i$;
  for $i = 0$ **to** $n-1$ **do**
    $\overrightarrow{t1} = \frac{\overrightarrow{C_iC_{L(I)}}}{|\overrightarrow{C_iC_{L(i)}}|}$; $\overrightarrow{t2} = \frac{\overrightarrow{C_iC_{R(i)}}}{|\overrightarrow{C_iC_{R(i)}}|}$; $\overrightarrow{b_i} = \overrightarrow{t1} \wedge \overrightarrow{t2}$;
  Construct $\zeta_1 = \{P_k\}_{k=0}^n$, with $\overrightarrow{P_kP_{k+1}} = \overrightarrow{b_k}$;
  Estimate the curvature value of width ν at each point of the curve ζ_1 as torsion value of corresponding point of the curve ζ (see Algorithm 2);
end

Remark: The bounds mentioned in the algorithm 4 are similar to the ones in the algorithm 3.

Therefore, by using these two propositions, we can estimate the torsion value at each point of ζ curve by determining the curvature value at the corresponding points of curve ζ_1. Our proposed method is presented in the algorithm 4, which uses the curvature estimator presented in section 4.

6. Experiments and comparisons

6.1. *Experiments*

We evaluated our methods on this computer configuration: CPU Pentium 4 with 3.2GHz, 1G of RAM, linux kernel 2.6.22-14 operating system. Because the estimated result is not correct for the beginning and the end of an open curve (see the bounds mentioned in the algorithms 2 and 3), during the phase of error estimation, we use a border parameter to eliminate this influence.

6.1.1. *Error measure*

We introduce three criteria for measuring error: mean relative error (meanRE), max relative error (maxRE) and quadratic relative error (QRE). Let us consider 2 sequences: the actual results $\{IR_i\}_{i=1}^n$ and the estimated results $\{RR_i\}_{i=1}^n$ at each position. Then, we can define:

$$meanRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|RR_i - IR_i|}{IR_i} \tag{1}$$

$$maxRE = max\left\{\frac{|RR_i - IR_i|}{IR_i}\right\}, i = 1, .., n \tag{2}$$

$$QRE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{|RR_i - IR_i|}{IR_i}\right)^2} \tag{3}$$

These criteria are modified from clasic error criteria. They allow us to measure how an estimated result respects the profile of an actual result.

6.1.2. *Curvature experiments*

We present in Fig. 5 some experiments with our curvature estimator for planar curves. Two discrete curves (Fig. 5.a and 5.c) are presented with the plot of their curvature values calculated at each point of the curves with width 2 (Fig. 5.b and 5.d). The points of the curve 5.c that correspond to the peaks (black squares) of the associated curvature graph 5.d, are indicated by black pixels. We recognize that these black pixels are well located on the corners of the curve 5.c. As a result, an application for corner detection can be deduced based on this curvature estimator.

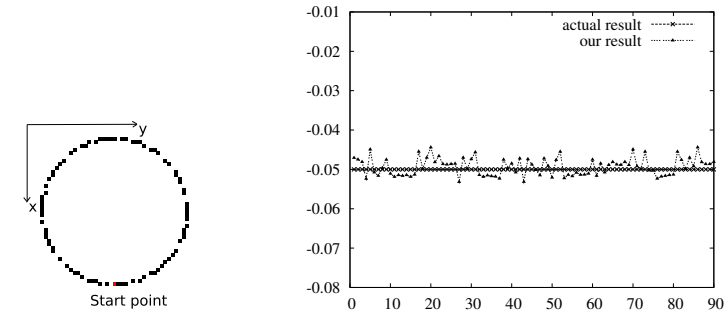
For 3D curves, Fig. 6 presents some experiments with our 3D curvature estimator on some actual spatial discrete curves: helix, Viviani's curve,... [a] with their actual curvature profiles (Fig. 6.b and 6.e) and their estimated curvature profiles using by our estimator (Fig. 6.c and 6.f).

Concerning the error approximation, the Tab. 1 shows the error of our method in relation with actual results when we test with these above curves.

6.1.3. *Torsion experiments*

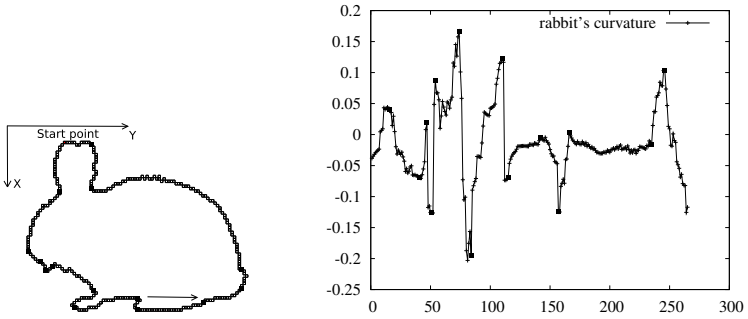
We present some experiments of our method on some ideal 3D curves : helix, Viviani's, spheric, horopter and hyper helix curves. The tests are done after a process of discretisation of these 3D curves (see Fig. 7).

[a]Math curves, http://www.mathcurve.com/courbes3d/courbes3d.shtml



(a) Noisy circle, radius = 20

(b) max=-0.0444, min=-0.0531, mean=-0.0474



(c)

(d) max=0.1761, min=-0.201

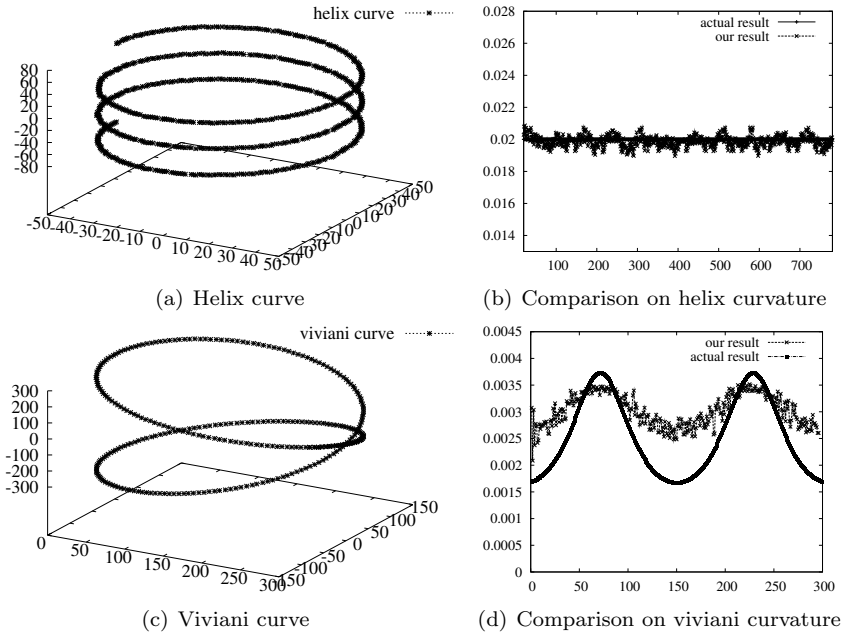
Fig. 5. Examples of 2D curvature extraction with $\nu = 2$: (a) A discrete circle (radius=20) - (b) associated curvature profile. (c) A rabbit discrete curve - (d) associated curvature profile.

Curves	N^0	***Border***	***meanRE***	***maxRE***	***QRE***	***Time***
Circle	90	6	0.0186	0.1118	0.0316	100
Helix	760	20	0.0074	0.0519	0.0132	610
Viviani	274	20	0.1185	0.7248	0.2198	200

Table 1. Error estimation on the curvature results, N^0: number of points, time is caculated in ms.

In most cases presented in Tab. 2 and Fig. 7, the mean relative errors do not overtake 0.15, and the quadratic relative errors do not overtake 0.015. If the actual torsion of the input curve has a value which is close to 0 at some positions, the obtained result is not very good. Let's see the case of Viviani's curve in the Tab. 2 (without threshold) and Fig. 9.a. In this case, the maximal relative error is high (15.6036). In spite of that, the mean relative error is acceptable (0.628899). In particular cases, if most of input curves has a torsion value which is close to 0, the obtained result is the worst (see Fig. 8, Fig. 9.b).

Like other approximation methods, our method doesn't perform well when the actual value of the torsion approximates zero. It comes from the formula of relatif error. The divergence between actual value and estimated value reduce slowly when the actual value is close to 0. So the relatif error is so high in this situation. Let



(a) Helix curve (b) Comparison on helix curvature

(c) Viviani curve (d) Comparison on viviani curvature

Fig. 6. Experiments of curvature estimator, working width $\nu = 2$

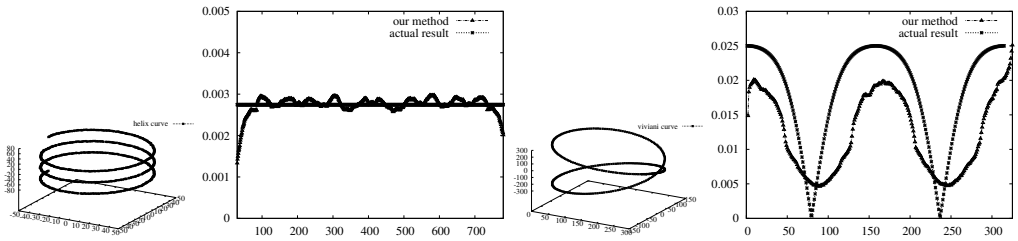
Curves	N^0	N^1	***m***eanRE		***m***axRE		***Q***RE		***Time***
			Without	With	Without	With	Without	With	
Spheric	255	30	0.1317	0.1317	0.3428	0.3428	0.1642	0.1642	280
Horopter	239	30	0.0827	0.0827	0.1862	0.1862	0.0979	0.0979	300
Helix	760	30	0.0481	0.0481	0.5145	0.5145	0.0814	0.0814	920
Viviani	274	30	0.6289	0.4570	15.6036	3.5128	1.6228	0.6156	290
Hyperhelix	740	30	8551.24	1.1527	154378	3.8019	24704.2	1.6210	720

Table 2. Error estimation on the torsion results without (with a threshold), N^0: number of points, N^1: border, time is caculated in ms. In the second line, without: without threshold; with: with a threshold = 0.0005

Estimator	N^0	N^1	***m***eanRE		***m***axRE		***Q***RE		***Time***
			Without	With	Without	With	Without	With	
Curvature	274	30	0.5191	0.5010	1.0618	1.0085	0.7355	0.7098	NA
Torsion	274	30	0.5367	0.4987	1.2160	1.0036	0.7615	0.7066	NA

Table 3. Error estimation on the curvature and torsion results of Lewiner's method on viviani's curve without (with a threshold), N^0: number of points, N^1: border, time is caculated in ms. In the second line, without: without threshold; with: with a threshold = 0.0005

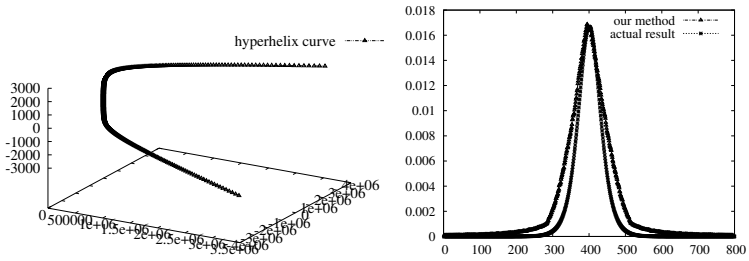
us consider the case of an hyper helix curve (see Fig. 8). The problem is that the torsion approximation is not good at nearly-0 values. In spite of that, the



(a) Helix curve (b) Comparison on helix torsion (c) Viviani's curve (d) Comparison on viviani torsion

(e) Spheric curve (f) Comparison on spheric torsion (g) Horopter curve (h) Comparison on horopter torsion

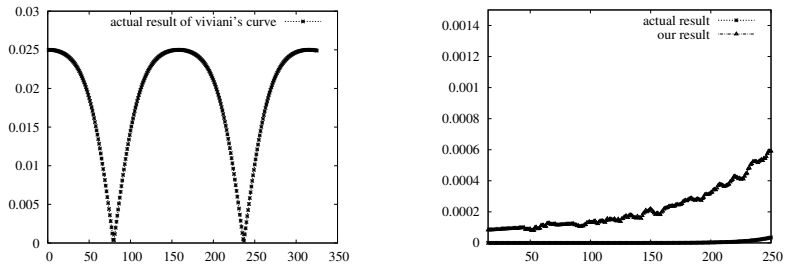
Fig. 7. Experiments on torsion estimator, working width $\nu = 2$.



(a) Hyperhelix curve (b) Comparison on hyperhelix torsion

Fig. 8. Most of the hyper helix curve has a torsion value close to 0. So in this case, the obtained result is the worst.

approximation value is also close to 0 but the relative rate between approximation value and actual value is very high. In the Fig. 9.b, we show the relation between approximation torsion and actual torsion of an hyper helix curve from the index 15 and to the index 250. In this index interval, the actual torsion is close to 0. So, the relative error between approximation torsion and actual torsion is very high, in spite of that the approximation value does not overtake 0.006. So, we propose to consider only the points whose actual torsion value is greater than the threshold. In the equations 1, 2 and 3, n is replaced by number of points whose actual torsion value is greater than a threshold. The Tab. 2 shows also the approximated error



(a) (b)

Fig. 9. a:Torsion of Viviani's curve; b:The actual torsion and our result obtained with a hyperhelix curve between the index 15 and 250.

Curves	***No***	***Border***	***Sigma***	***meanRE***	***maxRE***	***QRE***	***Time***
Hyper helix	740	30	4	0.451744	5.39524	0.796259	200
Spheric	255	30	9	0.32234	0.852819	0.47207	140
-	–	–	10	0.315752	0.843035	0.461791	220
-	–	–	12	0.302302	0.962883	0.442564	290
Viviani	274	30	7	0.466501	0.999349	0.664052	100
-	–	–	9	0.457997	0.996809	0.653339	140
-	–	–	12	0.451581	0.994291	0.643084	300

Table 4. Error estimation on the torsion results with Mokhtarian's method, No: number of points, time is caculated in ms. Implementation is developped and run on Matlab 7.5.0.

with a threshold equal to 0.0005.

6.2. *Comparisons to other approachs*

We compare our estimators to some well-known methods in the literature.
Mokhtarian's approach: Mokhtarian [24,7] proposed an approach using a scale space technique which is based on Gaussian fitting.

A planar curve, Γ is first parameterized by the arc length parameter u: $\Gamma = (x(u), y(u))$. By using Gaussian fitting, an evolved version Γ_σ is computed: $\Gamma_\sigma = (X(u,\sigma), Y(u,\sigma)) = (x(u) \otimes g(u,\sigma), y(u) \otimes g(u,\sigma))$, where $g(u,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-u^2}{2\sigma^2}}$ So the curvature at each point can be computed on the evolved curve Γ_σ as follows

$$\kappa(u,\sigma) = \frac{X_u(u,\sigma)Y_{uu}(u,\sigma) - X_{uu}(u,\sigma)Y_u(u,\sigma)}{\sqrt{(X_u(u,\sigma)^2 + Y_u(u,\sigma)^2)^3}}$$

where $X_u(u,\sigma) = x(u) \otimes g_u(u,\sigma)$, $X_{uu}(u,\sigma) = x(u) \otimes g_{uu}(u,\sigma)$, $Y_u(u,\sigma) = y(u) \otimes g_u(u,\sigma)$, $Y_{uu}(u,\sigma) = y(u) \otimes g_{uu}(u,\sigma)$, $g_u(u,\sigma) = \frac{\partial g(u,\sigma)}{\partial u}$, $g_{uu}(u,\sigma) = \frac{\partial \partial g(u,\sigma)}{\partial \partial u}$ Similarly, in the case of spatial curve, an evolved version of the curve is $\Gamma_\sigma = (X(u,\sigma), Y(u,\sigma), Z(u,\sigma))$. So, the curvature and torsion can be calculated respectively as follows.

$$\kappa(u,\sigma) = \frac{\sqrt{(X_u Y_{uu} - X_{uu} Y_u)^2 + (Y_u Z_{uu} - Y_{uu} Z_u)^2 + (Z_u X_{uu} - Z_{uu} X_u)^2}}{\sqrt{(X_u^2 + Y_u^2 + Z_u^2)^3}}$$

$$\tau(u,\sigma) = \frac{X_u(Y_{uu}Z_{uuu} - Z_{uu}Y_{uuu}) + Y_u(Z_{uu}X_{uuu} - X_{uu}Z_{uuu}) + Z_u(X_{uu}Y_{uuu} - Y_{uu}X_{uuu})}{(X_uY_{uu} - X_{uu}Y_u)^2 + (Y_uZ_{uu} - Y_{uu}Z_u)^2 + (Z_uX_{uu} - Z_{uu}X_u)^2}$$

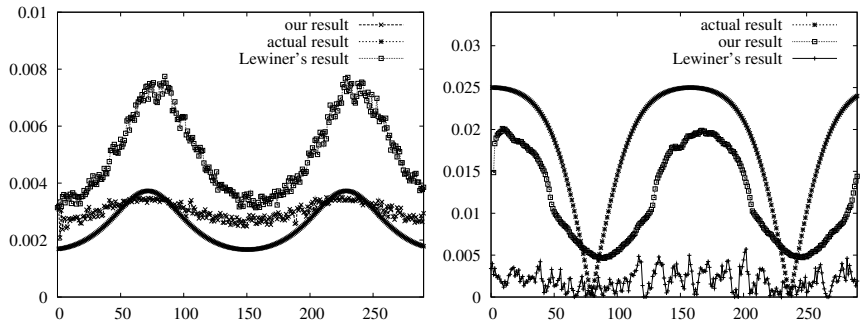
Lewiner et al's approach: Lewiner et al. [5,25] proposed curvature and torsion estimators based on a parametric curve fitting technique that uses weighted least-squares fitting. Consider a sequence of points $\{p_i\}$ on a smooth curve r. For each point p_j, he considered a window of $2q+1$ points around $p_j = r(j)$ to estimate the derivatives of r(s). Let s_i be the arc-length corresponding to sample p_i. It can be estimated as $l_i = \sum_{k=0}^{i-1} p_k p_{k+1}$. To determine curvature, considering that $p_0 = r(0)$ is the origin, he used the second order approximation: $r(s) = r^{'}(0)s + \frac{1}{2}r^{''}(0)s^2 + g1(s)s^3$ with $\lim_{s\to 0} g1(s) = 0$. The estimation of derivatives are obtained by a weighted least square minimization. Consider the case of a planar curve. His idea is to locally fit a parametric curve $(\hat{x}(s), \hat{y}(s))$ to the curve, with one of the coordinate functions, say $\hat{x}$, being quadratic in the arc-length: $\hat{x}(s) = x_0 + x_0^{'}.s + \frac{1}{2}x_0^{''}.s^2$. The derivatives $x_0^{'}$ and $x_0^{''}$ can be estimated by minimizing $E_x(x_0^{'}, x_0^{''}) = \sum_{i=-q}^{q} \omega_i(x_i - x_0^{'}l_i - \frac{1}{2}x_0^{''}(l_i)^2)^2$. The weight ω_i of the point p_i can be considered simply $\omega_i = 1$. The estimates $y_0^{'}$ and $y_0^{''}$ are obtained by both the unit norm of the tangent and the orthogonality of the tangent and the normal: $(x_0^{'})^2 + (y_0^{'})^2 = 1$ and $x_0^{'}x_0^{''} + y_0^{'}y_0^{''} = 0$. A similar approach with spatila curve. So the curvature can be calculated by this formula:

$$\kappa(t,q,\sigma) = \frac{r^{'} \times r^{''}}{||r^{'''}||}$$

For torsion estimation, the third order approximation is used: $r(s) = r^{'}(0)s + \frac{1}{2}r^{''}(0)s^2 + \frac{1}{6}r^{'''}(0)s^3 + g2(s)s_i^4 +$ with $\lim_{s\to 0} g2(s) = 0$. For spatial curve, the torsion estimator fits a cubic parametric curve to the sample points. So, $x_0^{'}$, $x_0^{''}$ and $x_0^{'''}$ should minimize: $E_x(x_0^{'}, x_0^{''}), x_0^{'''} = \sum_{i=-q}^{q} \omega_i(x_i - (x_0^{'}s_i + \frac{1}{2}x_0^{''}s_i^2 + \frac{1}{6}x_0^{'''}s_i^3))^2$. A similar approach is used to computed $y_0^{'}$, $y_0^{''}$, $y_0^{'''}$, $z_0^{'}$, $z_0^{'}$, $z_0^{'''}$. The torsion is given by

$$\tau(t,q,\sigma) = \frac{(r^{'} \times r^{''}).r^{'''}}{||r^{'} \times r^{''}||^2}$$

Comparison: We present in Fig. 10, 11 and Tab. 3, 4 a comparison between our estimators and these methods. We admit that the result of Mokhtarian's method depends largely on the value of the parameter σ. When the value of σ is too small, the obtained result is very noisy, but the higher value for σ can lead to inexact results and margin effects in which the result is totally different from actual result. Our method use 1 parameter that is the width, Mokhtarian's method used 1 parameter that is σ. Lewiner's method use 2 different parameters (σ and q), so the number of parameters of this method is more than 2 others. The result of this torsion estimator seems to be more noisy than other methods. Concerning the quality of approximated results, our method is better than 2 methods, but among these 3 methods, the method of Mokhtarian is the fastest thanks to its linear complexity.



(a) Comparison on curvature estimator (b) Comparison on torsion estimator

Fig. 10. Comparisons between our curvature and torsion estimators with Lewiner's approach on Viviani's curve. The results of Lewiner are supplied by the author.

7. Conclusions

We have presented in this paper the methods to estimate curvature for planar curves, and the curvature and torsion for spatial curves. These methods benefit from the improvement of curvature estimator in 2D case [3], so they are efficient than using existent method [1,17] for blurred segment recognition to construct these estimators.

These estimators permit to extract local properties of spatial curves. We hope to identify and classify 3D objects by using these estimators. Our work can also be applied for an application to fibres in 3D images of paper that has been presented in [13]. Another applications can be geometric compression [25] or medical imaging [9]. In the futur, we will look for an application for DNA curve matching based on curvature and torsion estimation.

Acknowledgements

A preliminary version of this work was presented at the International Symposium on Visual Computing, Las Vegas, NV, 2008, Special Track on Discrete and Computational Geometry and their Applications in Visual Computing [26].

We would like to thank Thomas Lewiner for supplying his program to evaluate performance.

We would like to thank the reviewers for their constructive and detailed comments that permit us to improve the manuscript.

References

1. I. Debled-Rennesson, F. Feschet, J. Rouyer-Degli, Optimal blurred segments decomposition of noisy shapes in linear time, Computers & Graphics 30 (1).
2. J.-P. Reveillès, Géométrie discrète, calculs en nombre entiers et algorithmique, thèse d'état. Université Louis Pasteur, Strasbourg (1991).
3. T. P. Nguyen, I. Debled-Rennesson, Curvature estimation in noisy curves, in: CAIP, Vol. 4673 of Lecture Notes in Computer Science, Springer, 2007, pp. 474–481.
4. J.-P. Salmon, I. Debled-Rennesson, L. Wendling, A new method to detect arcs and segments from curvature profiles, in: ICPR (3), 2006, pp. 387–390.

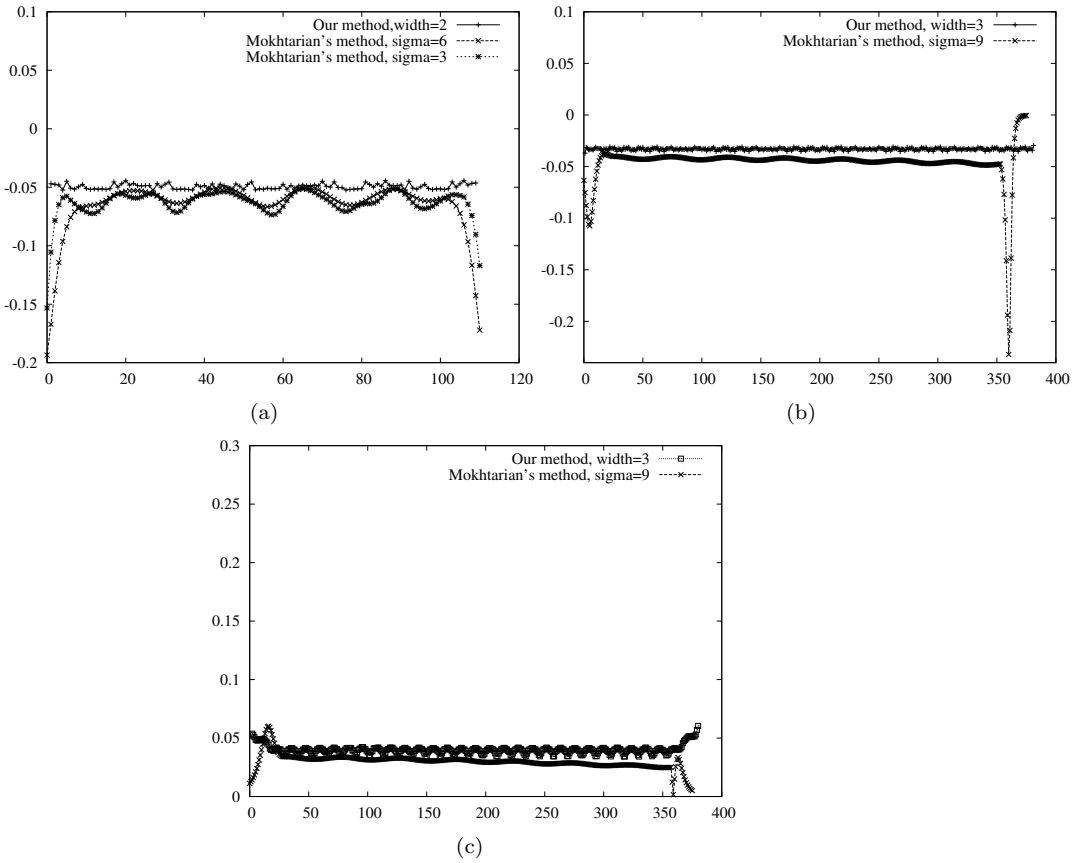


Fig. 11. a. Comparison between our 2D curvature estimator and Mokhtarian's 2D curvature estimator on a circle, radius is 20; b (resp. c): Comparison between our 3D curvature (resp. torsion) estimators and Mokhtarian's 3D curvature (resp. torsion) estimators on an helix curve.

5. T. Lewiner, J. D. G. Jr., H. Lopes, M. Craizer, Curvature and torsion estimators based on parametric curve fitting., Computers & Graphics 29 (5) (2005) 641–655.
6. E. Attneave, Some informational aspects of visual perception, Psychol. Rev. 61 (3).
7. F. Mokhtarian, A theory of multiscale, torsion-based shape representation for space curves., Computer Vision and Image Understanding 68 (1) (1997) 1–17.
8. N. D. Kehtarnavaz, R. J. P. de Figueiredo, A 3-d contour segmentation scheme based on curvature and torsion, IEEE Trans. Pattern Anal. Mach. Intell. 10 (5) (1988) 707–713.
9. R. Medina, A. Wahle, M. E. Olszewski, M. Sonka, Curvature and torsion estimation for coronary-artery motion analysis, in: SPIE Medical Imaging, Vol. 5369, 2004, pp. 504–515.
10. F. Feschet, L. Tougne, Optimal time computation of the tangent of a discrete curve: Application to the curvature., in: DGCI, Vol. 1568 of LNCS, 1999, pp. 31–40.
11. D. Coeurjolly, S. Miguet, L. Tougne, Discrete curvature based on osculating circle estimation., in: IWVF, Vol. 2059 of LNCS, 2001, pp. 303–312.

12. J.-O. Lachaud, A. Vialard, F. de Vieilleville, Analysis and comparative evaluation of discrete tangent estimators., in: DGCI, Vol. 3429 of LNCS, 2005, pp. 240–251.
13. D. Coeurjolly, S. Svensson, Estimation of curvature along curves with application to fibres in 3d images of paper, in: SCIA, 2003, pp. 247–254.
14. I. Debled-Rennesson, Reconnaissance des droites et plans discrets, Ph.D. thesis, Louis Pasteur University (1995).
15. D. Coeurjolly, I. Debled-Rennesson, O. Teytaud, Segmentation and length estimation of 3d discrete curves, in: Digital and Image Geometry, Vol. 2243 of LNCS, Springer, 2000, pp. 299–317.
16. M. Overmars, J. van Leeuwen, Maintenance of configurations in the plane, J. Comput. and Syst. Sci. 23 (1981) 166–204.
17. L. Buzer, An elementary algorithm for digital line recognition in the general case., in: DGCI, Vol. 3429 of LNCS, 2005, pp. 299–310.
18. A. V. Aho, J. E. Hopcroft, J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1974.
19. I. Debled-Rennesson, Estimation of tangents to a noisy discrete curve, in: Vision Geometry XII, SPIE, Vol. 5300, 2004, pp. 117–126.
20. A. Rosenfeld, E. Johnston., Angle detection on digital curves, IEEE Transactions on Computers (1973) 875–878.
21. J. Harris, H. Stocker, Handbook of mathematics and computational science, Springer-Verlag, 1998.
22. J. Oprea, Differential geometry and its applications, 2007.
23. M. Worring, A. W. M. Smeulders, Digital curvature estimation., Computer Vision Graphics Image Processing: CVIU 58 (3) (1993) 366–382.
24. F. Mokhtarian, A. K. Mackworth, A theory of multiscale, curvature-based shape representation for planar curves, IEEE Trans. Pattern Anal. Mach. Intell. 14 (8) (1992) 789–805.
25. T. Lewiner, J. D. G. Jr., H. Lopes, M. Craizer, Arc-length based curvature estimator, in: SIBGRAPI, 2004, pp. 250–257.
26. T. P. Nguyen, I. Debled-Rennesson, Curvature and torsion estimators for 3d curves, in: ISVC (1), Vol. 5358 of Lecture Notes in Computer Science, Springer, 2008, pp. 688–699.