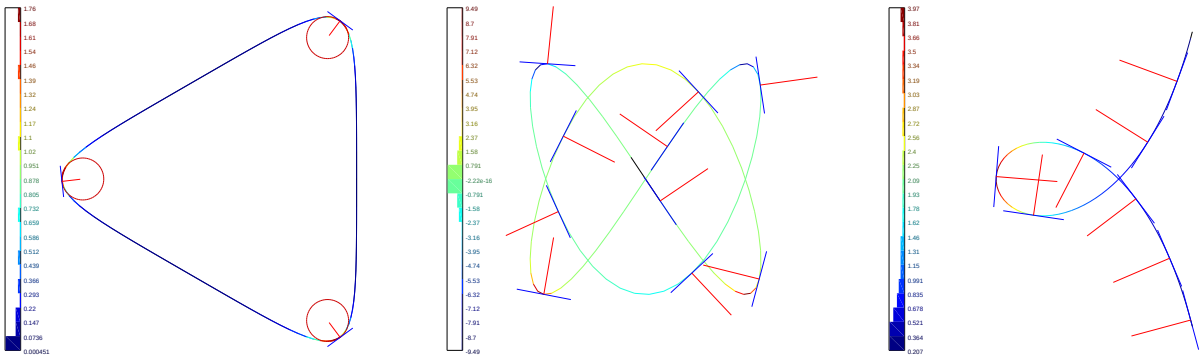
Arc–length based curvature estimator

THOMAS LEWINER[1,2], JOÃO D. GOMES JR.[1], HÉLIO LOPES[1] AND MARCOS CRAIZER[1]

[1] Department of Mathematics — Pontifícia Universidade Católica — Rio de Janeiro — Brazil
[2] Géométrica Project — INRIA – Sophia Antipolis — France
{tomlew, jgomes, lopes, craizer}@mat.puc--rio.br.

Abstract. Many applications of geometry processing and computer vision relies on geometric properties of curves, particularly their curvature. Several methods have been proposed to estimate the curvature of a planar curve, most of them for curves in digital spaces. This work proposes a new method for curvature estimation based on weighted least square fitting and local arc–length approximation. Convergence analysis of this method and noise impact on the estimator accuracy are given. Numerical robustness issues are addressed with practical solutions. The implementation of the method is compared to other curvature estimation methods.
Keywords: ***Differential Geometry. Curvature Estimation. Weighted Least–Squares. Geometry Processing.***



(a) Hypocycloid (b) Lissajous (c) Strofoid

Figure 1: *Estimated curvatures (colour), tangent lines and normal vectors: (a)* $\mathbf{r}(t) = (4\cos(t) - 2\sin(2t), 4\sin(t) + 2\cos(2t)) : t \in [-2\pi, 2\pi]$*; (b)* $\mathbf{r}(t) = (\sin(2t), \sin(3t)) : t \in [-\pi, \pi]$*; (c)* $\mathbf{r}(t) = (\frac{t^2-1}{t^2+1}, t\frac{t^2-1}{t^2+1}) : t \in [-2, 2]$.

1 Introduction

Many applications of geometry processing and computer vision relies on geometric properties of curves. In particular, the curvature measures how a curve bends, which is one of the most characteristic property of a curve considered for theoretical analysis and practical applications. For example in CAGD [16] and in Computer Vision [15], curvature motion [18], curve reconstruction [1, 3], curve compression [11], and adaptive curve approximation [14] requires accurate curvature estimators.

Several methods have already been proposed for curvature estimation, most of them in the particular case of digital spaces, i.e. curves extracted from images [6]. In this work, we shall consider piecewise–linear approximation of planar curves, which is a general framework that includes those digital curves. This approach leads to a clear theoretical analysis and serves directly to applications to geometric modeling and computer graphics.

Problem statement. Discrete curves proceed from different sources: digital curves [6], parametric or implicit curves [14], curve reconstruction [1, 3]... Piecewise–linear approximations, provides a general framework that includes the above cases. A piecewise–linear approximation $\mathbf{P}$ of a planar curve $\mathbf{r}$ is a finite sequence of m *sample points* $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\}$ of $\mathbf{r}$. We admit the presence of noise. In this paper, we will try to estimate accurately the tangent line and the curvature of the curve $\mathbf{r}$ at a point $\mathbf{p}_j$ of $\mathbf{P}$.

Contributions. In this paper, we introduce a new method for curvature estimation based on weighted least square fitting and local arc–length approximation. More precisely, we fit a second–order polynomial for each coordinate, considered as a function of the arc–length. We prove the convergence of our estimations under reasonable conditions over the sampling of the curve and the amplitude of the noise. We

Preprint MAT. 12/04, communicated on March 30^{th}, 2004 to the Department of Mathematics, Pontifícia Universidade Católica — Rio de Janeiro, Brazil. The corresponding work was published in the proceedings of the Sibgrapi 2004, pp. 250–257. IEEE Press, 2004.

provide a practical implementation of this method, addressing numerical issues with simple solutions. The preliminary results show that our methods compare nicely to the state–of–the–art, and that it has a strong stability over different conditions of noise and sampling.

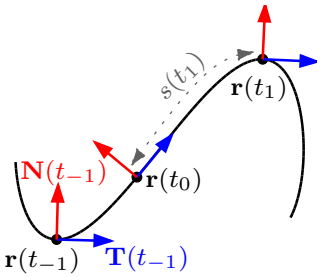


Figure 2: *The arc–length $s(t)$ helps defining the tangent and the normal vectors $\mathbf{T}(t)$ and $\mathbf{N}(t)$.*

Paper outline. Section 2 *Curvature of a planar parametric curve* introduces the concepts and notations from differential geometry of curves that will be used in this work. Section 3 *Previous and related works* discusses the previous and related works. The theoretical analysis of our method is presented in Section 4 *Theoretical framework*. Section 5 *Computational framework* details the implementation of our schemes, and introduces improvements on the method numerical robustness. Our algorithm is finally compared to the state–of–the–art in the last section.

2 Curvature of a planar parametric curve

A *parametric curve* in the plane is a function $\mathbf{r} : I \subset \mathbb{R} \to \mathbb{R}^2, t \mapsto (x(t), y(t))$, where x and y are functions from I to $\mathbb{R}$. The curve $\mathbf{r}$ is said to be *regular* if x and y are C^1 and $\dot{r}(t) = \frac{d\mathbf{r}}{dt}(t)$ never vanishes on I.

From now on, let us suppose that $\mathbf{r}$ is a regular parameterized curve. The *arc–length* s from the point $\mathbf{r}(t_0), t_0 \in I$, to a given point $\mathbf{r}(t), t \in I$, is by definition $s(t) = \int_{t_0}^{t} \|\dot{\mathbf{r}}(t)\| dt$. When the curve is regular, $s(t)$ is strictly increasing, and has therefore an inverse $t(s)$. The curve can be *parameterized by the arc–length* by considering $\mathbf{r}(s) = \mathbf{r} \circ t(s)$. Along this paper, we will denote the derivation with relation to the arc–length s with a prime ($\mathbf{r}'$), and the derivation with relation to t by a dot ($\dot{\mathbf{r}}$).

The vector $\mathbf{T}(s) = \mathbf{r}'(s)$ is called the *tangent vector*. The *normal vector* $\mathbf{N}(s)$ is directly orthogonal to the tangent vector $\mathbf{T}(s)$: $\mathbf{N}(s) = (-y'(s), x'(s))$ (see Figure 2). Observe that $\mathbf{T}'(s)$ and $\mathbf{N}(s)$ are colinear, because $\|\mathbf{T}(s)\| = 1$ is constant. If $\mathbf{r}$ is a curve of class C^2 parameterized by the arc–length, then the analytical definition of the curvature $\kappa(s)$ follows the Frenet's formula: $\kappa(s) = \mathbf{T}'(s) \cdot \mathbf{N}(s)$. When the curve $\mathbf{r}(t)$ is not parameterized by the arc–length, the curvature is given by:

$$\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{(\dot{x}^2(t) + \dot{y}^2(t))^{\frac{3}{2}}} \qquad (1)$$

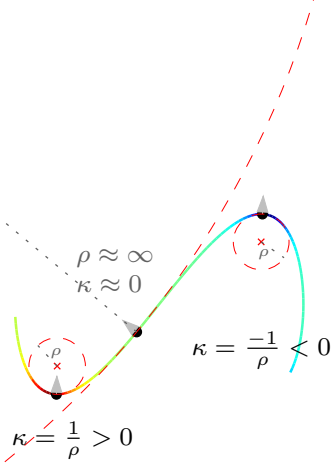


Figure 3: *The curvature is the inverse of the radius of the osculating circle. Its sign corresponds to the local convexity of the curve.*

The *radius of curvature* $\rho(s)$ is the radius of the osculating circle at $\mathbf{r}(s)$ (see Figure 3). The absolute value of the curvature can also be defined geometrically by $|\kappa(s)| = 1/\rho(s)$. The sign of $\kappa(s)$ indicates whether the curve is concave or convex at that point.

The curvature also corresponds to the variation of the tangent direction with respect to the arc–length: $\kappa(s) = \theta'(s)$, where $\theta(s) = \angle(\mathbf{T}(s), (1, 0))$.

3 Previous and related works

Several methods have already been proposed for estimating the curvature at point p_j, most of them in the particular case of digital spaces, i.e. curves extracted from images [6]. In this section, we will review the most significant to us. We implemented those methods for the comparison of Section 6 *Experimental results*. Those approaches are classified in three groups, according to which definition of curvature they are using (as done in [20]): tangent direction, osculating circle, derivation. Most methods use a sliding window of $2q + 1$ points centered around p_j.

Methods based on the tangent direction The methods of the first group estimate the derivative of the tangent direction with respect to the arc–length, i.e. $\kappa(s) = \theta'(s)$. For digital images, this requires to estimate the gradient of a polygonal approximation of an implicit curve. This is done in [20] in three ways.

The first method (referred as *line fitting*) estimates the tangent direction at the sides of a sample j by a Gaussian–weighted linear fit centered at the left point at $j - 1$ and right point at $j + 1$. The curvature is then estimated as the difference of orientation divided by the distance between the points at $j - 1$ and $j + 1$. This method is not very robust due to the numerical imprecision on angle computation.

The second method (referred as *chain code* in [20]) evaluates the local angle $\hat{\theta}(\mathbf{p}_i) = \tan^{-1}\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right)$ around j : $i = j - q \ldots j + q$. The derivation is done by convolution with a derived Gaussian Kernel G_σ: $\hat{\kappa} = \hat{\theta} * \dot{G}_\sigma$.

However, the curvature equals the derivation with respect to the arc–length. Therefore, the third method (referred as *resampling* in [20]) first performs a resampling of the

The corresponding work was published in the proceedings of the Sibgrapi 2004, pp. 250–257. IEEE Press, 2004.

curve by linear interpolations on the curve segments. This introduces a bias of 1.107 which is explicitly corrected.

In [9], the angle is estimated as the external angle around the sample points. This improves numerically the results of [20] by avoiding right angles in the computation. This last method only uses 3 points for the approximation.

Methods based on the radius of curvature. The second group of methods compute the curvature by estimating the osculating circle touching the curve ($\kappa(s) = 1/\rho(s)$).

In [5], the radius of the circle passing through p_{j-q}, p_j and p_{j+q} is estimated by: $\hat{\kappa}(\mathbf{p}_j) = \frac{\angle(\mathbf{p}_{j-q}\mathbf{p}_j, \mathbf{p}_j\mathbf{p}_{j+q})}{\|\mathbf{p}_{j-q}\mathbf{p}_j\| + \|\mathbf{p}_j\mathbf{p}_{j+q}\|}$.

This result was improved in [5] by the area formula for the radius of the circle circumscribed to a triangle: $\hat{\kappa}(\mathbf{p}_i) = \frac{\sqrt{(b+c)^2-a^2}\cdot\sqrt{a^2-(b-c)^2}}{abc}$, where a, b and c are, respectively, the norm of the vectors $\mathbf{p}_j\mathbf{p}_{j-q}$, $\mathbf{p}_j\mathbf{p}_{j+q}$, and $\mathbf{p}_{j-q}\mathbf{p}_{j+q}$.

In [19], the osculating circle is approximated by a direct least–square fitting of a circle, using an intermediate Cholewsky decomposition for the optimisation [17].

Methods based on coordinate functions derivation. Finally, methods of the last group are based on the first and second derivative estimation of the curve (see Equation (1)).

In [20], the *path* method obtains the derivatives by a convolution with a derived Gaussian kernel.

In [2], the derivatives are estimated as weighted local differences among three points centered at p_j.

In [7], the derivatives are estimated by an imaginary multiplication in the frequency domain of a closed curve. This approach combines efficiently with a multi–scale analysis by convoluting the curve with different Gaussian kernels.

In [4], one coordinate of the curve is approximated by a polynomial in the second coordinate through a least–square fitting, and the derivatives are estimated by the coefficient of that polynomial. Our method also use least–square fitting, but we approximate the curve with a rotated parabola, whereas [4] restricts the parabola to be parallel to the x or y axis. More generally, our method fits each coordinate as a quadratic function of the arc–length, and estimates the curvature by derivation of that function.

4 Theoretical framework

In this section, we describe our model and approach to solve the problem of tangent line and curvature estimation.

(a) Model and notations

Consider a piecewise–linear approximation $\mathbf{P}$ of a C^3 curve $\mathbf{r}$ in $\mathbb{R}^2$: $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\}$. We admit some noise in the samples. In this theoretical analysis, we will assume that the curve is parameterized by arc–length, although the samples p_i need not to be equally spaced. We will try to estimate the first and second derivatives of the coordinate functions $x(s)$ and $y(s)$.

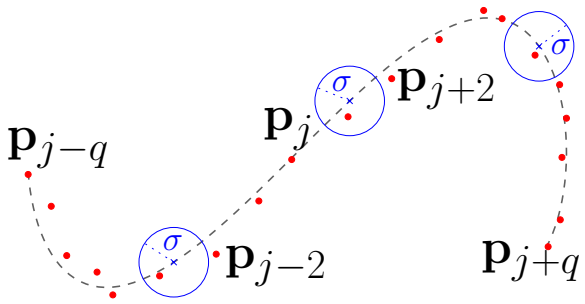


Figure 4: *Sampled curve with noise.*

Assuming that $\mathbf{p}_j$ is the origin of the curve, i.e., $\mathbf{r}(0) = \mathbf{p}_j$, we can write:

$$\begin{cases} x(s) = x(0) + x'(0)\, s + \frac{1}{2}\, x''(0)\, s^2 + g_1(s)s^3 \\ y(s) = y(0) + y'(0)\, s + \frac{1}{2}\, y''(0)\, s^2 + g_2(s)s^3 \end{cases}$$

with $g_i(s) \to 0$ when $s \to 0$. Since $\mathbf{p}_i = (x_i, y_i)$ are samples of the curve associated to the value of arc–length s_i, we can write

$$\begin{cases} x_i = x_j + x'_j\, s_i + \frac{1}{2}\, x''_j\, s_i^2 + g_1(s_i)\, s_i^3 + \eta_{x,i} \\ y_i = y_j + y'_j\, s_i + \frac{1}{2}\, y''_j\, s_i^2 + g_2(s_i)\, s_i^3 + \eta_{y,i} \end{cases}$$

where $\eta_i = (\eta_{x,i}, \eta_{y,i})$ is the noise corresponding to the point $\mathbf{p}_i$. We shall assume that the random variables η_i are independent and identically distributed (i.i.d.) with zero mean and variance σ^2 (see Figure 4). We aim to estimate x'_j, y'_j, x''_j and y''_j from the samples, i.e., the first and second order derivatives of $\mathbf{r}$ at $\mathbf{p}_j$. To obtain those values we shall use a weighted least squares (WLS) approach.

(b) The weighted least squares approach

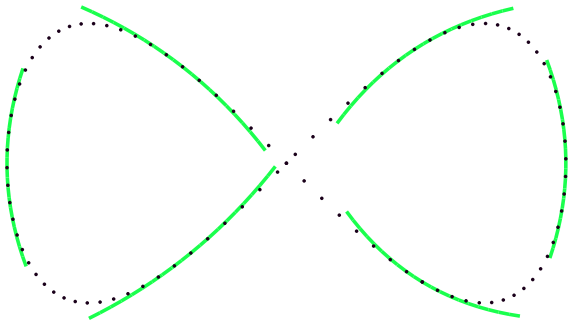
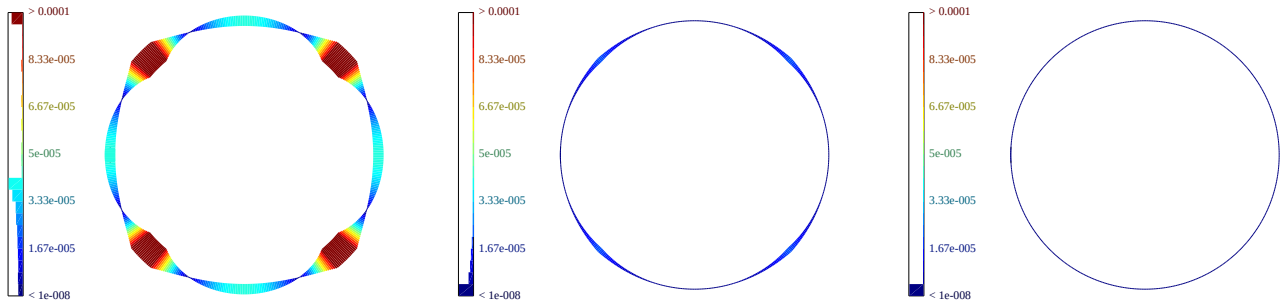


Figure 5: *Second–order weighted least square fitting.*

First, we need an estimate for the arc–length s_i. Define Δl_k as the length of the vector $\mathbf{p}_k\mathbf{p}_{k+1}$, where k ranges from 1 to $(m-1)$. Since we have assumed that $\mathbf{p}_j$ is the origin of the curve, the arc–length estimator from $\mathbf{p}_j$ to $\mathbf{p}_i$ is defined as $\Delta l_i^j = \sum_{k=j}^{i-1} \Delta l_k$, when $i > j$, and $\Delta l_i^j = -\sum_{k=i}^{j-1} \Delta l_k$, when $i < j$.

In our approach we will restrict our calculus to a sliding window of $2q+1$ points centered around p_j: we will only use



(a) Parabola fitting of [4]. (b) Dependent coordinate method. (c) Dependent coordinate method with rotation.

Figure 6: *Least–square fitting defect can be compensated by rotation: we drew big errors by wide line, the scale being the same on the three figures.*

the sample points $\mathbf{p}_{j-q}, ..., \mathbf{p}_{j+q}$ of $\mathbf{P}$. We will now describe how to obtain the estimations of the derivatives x'_j and x''_j, and the same solution is used to obtain y'_j and y''_j.

Considering the distinct abscissas $\Delta l^j_{j-q}, ..., \Delta l^j_{j+q}$ at which ordinates $x_{j-q} - x_j, ..., x_{j+q} - x_j$, $y_{j-q} - y_j, ..., y_{j+q} - y_j$ are assigned. We will look for the quadratic functions.

$$\begin{cases} x(s) = x_j + x'_j\, s + \frac{1}{2}\, x''_j\, s^2 \\ y(s) = y_j + y'_j\, s + \frac{1}{2}\, y''_j\, s^2 \end{cases}$$

that better fits these data in the weighted least squares sense (see Figure 5). In other words, we shall look for x'_j and x''_j that minimize

$$E_x(x'_j, x''_j) = \sum_{i=j-q}^{j+q} w_i \left(x_i - x_j - x'_j\, \Delta l^j_i - \tfrac{1}{2}\, x''_j\, (\Delta l^j_i)^2 \right)^2 .$$

and similarly for y'_j and y''_j. The real numbers w_i are the weight of the point $\mathbf{p}_i$. Such numbers are to be chosen positive, relatively large for small $|\Delta l^j_i|$ and relatively small for larges $|\Delta l^j_i|$. For example, we can consider weights of the form $w(\Delta l) = \alpha\, \exp(-\beta \Delta l^2)/\Delta l^k$.

When $q > 1$, the above WLS problems have a well–known solution [10]. So we can write the following formulas for the derivatives of $\mathbf{r}$:

$$\begin{cases} x'_j = \frac{ce-bf}{ac-b^2}, & x''_j = \frac{af-be}{ac-b^2}, \\ y'_j = \frac{cg-bh}{ac-b^2}, & y''_j = \frac{ah-bg}{ac-b^2}. \end{cases}$$

$$\text{where : } \begin{cases} a = & \sum_{i=j-q}^{j+q} w_i^2\, (\Delta l^j_i)^2 \\ b = \frac{1}{2} & \sum_{i=j-q}^{j+q} w_i^2\, (\Delta l^j_i)^3 \\ c = \frac{1}{4} & \sum_{i=j-q}^{j+q} w_i^2\, (\Delta l^j_i)^4 \\ e = & \sum_{i=j-q}^{j+q} w_i^2\, \Delta l^j_i\, (x_i - x_j) \\ f = \frac{1}{2} & \sum_{i=j-q}^{j+q} w_i^2\, (\Delta l^j_i)^2\, (x_i - x_j) \\ g = & \sum_{i=j-q}^{j+q} w_i^2\, \Delta l^j_i\, (y_i - y_j) \\ h = \frac{1}{2} & \sum_{i=j-q}^{j+q} w_i^2\, (\Delta l^j_i)^2\, (y_i - y_j). \end{cases}$$

Observe that when the weights are symmetrical and the samples are equally spaced around $\mathbf{p}_j$ the term b vanishes. The 4–connected digital curves are examples of such situation. With those results, our curvature estimator is given by:

$$\hat{\kappa}(\mathbf{p}_j) = \frac{eh - fg}{ac - b^2}.$$

(c) Convergence analysis: sampled curve without noise

In the following, we shall denote by δ the maximal distance between samples: $\delta = \max\left\{ \left|\Delta l^j_{j-q}\right|, \left|\Delta l^j_{j+q}\right| \right\}$, and by K_0 and K_1 the maximum of the curvature and its derivative around $\mathbf{p}_j$: $K_0 = \max\{|\kappa(s)|, |s| \le \delta\}$ and $K_1 = \max\{|\kappa'(s)|, |s| \le \delta\}$, where $\kappa(s)$ is the curvature of $\mathbf{r}$ at s. Let $\phi = \left(ac + \frac{1}{2}\,|b| \sum w_i^2\, |\Delta l^j_i|^3 \right) / \left(ac - b^2 \right)$ and $\theta(\varepsilon) = \frac{\sin(\varepsilon/2)}{\varepsilon/2}$.In the technical report [12], we give precise proofs of the following results.

Proposition 1 (Convergence without noise) *(a) If $\delta K_0 \le \varepsilon$. Then the estimation error is bounded:*

$$\left| x'_j - x'(s_j) \right| \le \phi\, \tfrac{1-\theta(\varepsilon)}{\theta(\varepsilon)}\, x'(s_j) + \phi\, \tfrac{\varepsilon}{\theta(\varepsilon)}.$$

(b) Suppose that $\delta K_0 \le \varepsilon$ and $\delta K_1 \le \varepsilon$. Then:

$$\left| x''_j - x''(s_j) \right| \le \phi\, \frac{1 - \theta^2(\varepsilon)}{\theta^2(\varepsilon)}\, x''(s_j) + \phi\, \frac{\varepsilon}{\theta^2(\varepsilon)}\, \left(1 + x'(s_j)\, K_0 \right).$$

In other terms, the products δK_0 and δK_1 should be small, which corresponds to the intuition that the sampling must be denser in regions of high curvature. If not, some samples are too far from $\mathbf{p}_j$ to be correctly used in the estimate of the first derivatives of $\mathbf{r}$ at $\mathbf{p}_j$.

(d) Convergence analysis: noisy curve

Let :

$$\Gamma_0 = \left(c^2 \sum w_i^4\, (\Delta l^j_i)^2 + \tfrac{b^2}{4} \sum w_i^4\, (\Delta l^j_i)^4 \right) / (ac - b^2)^2$$
$$\Gamma_1 = \left(\tfrac{b^2}{4} \sum w_i^4\, (\Delta l^j_i)^2 + a^2 \sum w_i^4\, (\Delta l^j_i)^4 \right) / (ac - b^2)^2$$

In the particular case where the samples are symmetrically distributed around $\mathbf{p}_j$ and the weights w_i are equal, we have $b = 0$ and $\Gamma_0^{-1} = a \approx \delta^2 q$, and $\Gamma_1^{-1} = c \approx \delta^4 q$, where we assumed that q is big in the approximation.

Proposition 2 (Convergence with noise) *(a) Assume that $\sigma^2\, \Gamma_0 \le \gamma$. Then the error of estimation $\left| x'_j - x'(s_j) \right|$ is bounded by the sum of the errors of proposition 1(a) and a random variable of zero mean and variance less than γ.*

The corresponding work was published in the proceedings of the Sibgrapi 2004, pp. 250–257. IEEE Press, 2004.

(b) Assume that: $\sigma^2\,\Gamma_1 \leq \gamma$. Then the error of estimation $\left|x''_j - x''(s_j)\right|$ is bounded by the sum of the errors of proposition 1(b) and a random variable of zero mean and variance less than γ.

In other words, the products $\sigma^2\Gamma_0$ and $\sigma^2\Gamma_1$ should be small, which again corresponds to the intuition that the number of points $2q+1$ considered for the approximation must increase with the noise. If not, the noise is too strong for us to guarantee the estimation for $\left(x'_j, y'_j\right)$ and $\left(x''_j, y''_j\right)$.

Algorithm 1 Set Weighted Least Squares Variables (j).

1: $\Delta l[]$ = a = b = c = e = f = g = h = 0 ;
2: **for** i = 1 ... 2q **do**
3: $\quad\Delta l$[i] $\leftarrow \Delta l$[i-1] + $\|p_{j-q+i-1}p_{j-q+i}\|$;
4: **end for**
5: $m = \Delta l$[j] ;
6: **for** i = 0 ... 2q **do**
7: $\quad\Delta l$[i] $\leftarrow \Delta l$[i] - m ; *// Centering dl on j*
8: $\quad w$ = weight(Δl[i])^2 ;
9: $\quad a \leftarrow a$ + w (Δl[i])^2 ;
10: $\quad b \leftarrow b$ + w (Δl[i])^3 ;
11: $\quad c \leftarrow c$ + w (Δl[i])^4 ;
12: $\quad e \leftarrow e$ + w (Δl[i]) (x_{j+i}-x_j) ;
13: $\quad f \leftarrow f$ + w (Δl[i]) (y_{j+i}-y_j) ;
14: $\quad g \leftarrow g$ + w (Δl[i])^2 (x_{j+i}-x_j) ;
15: $\quad h \leftarrow h$ + w (Δl[i])^2 (y_{j+i}-y_j) ;
16: **end for**
17: $d = ac - b^2$; *// determinant*

5 Computational framework

The method we introduced is extremely simple to implement. It has two variants we called *Independent coordinates* and *Dependent coordinates*. Moreover, the numerical results can be improved by a simple *rotation* on the data, in order to have the tangent direction close to the horizontal (see Figure 6).

Our algorithm follows directly from the analysis of Section 4 *Theoretical framework*: we compute the coefficient $a, b, c, e, f, g,$ and h and solves the WLS method (see Algorithm 1).

(a) Independent coordinates method

This method computes the estimations for x'_j, y'_j, x''_j and y''_j. To do so we find the solution for the two WLS problem independently for x and y. The resulting tangent vector $\mathbf{T} = (x'_j, y'_j)$ is not constrained to be unitary, and the vector (x''_j, y''_j) is not constrained to be orthogonal to $\mathbf{T}$. We normalize the normal vector estimate in the direction of $sign(\kappa)(x''_j, y''_j)$. The solution is then carried out by Algorithm 2.

Algorithm 2 Independent coordinates WLS Solution (j)

1: **call** Set Weighted Least Squares Variables (j);
2: $\mathbf{T}_x = (ce - bf)/d$;
3: $\mathbf{T}_y = (cg - bh)/d$;
4: $\mathbf{N}_x = (af - be)/d$;
5: $\mathbf{N}_y = (ah - bg)/d$;
6: $\kappa = (eh - fg)/d$;
7: $\mathbf{N}$ = sign(κ) ($\mathbf{N}/\|\mathbf{N}\|$) ;

(b) Dependent coordinates method

We observe that when the curve is parameterized by the arc–length, we must have:

$$\begin{cases} x'^2 + y'^2 = 1 \\ x'x'' + y'y'' = 0 \end{cases}$$

Thus, with those two equations we can use estimates for x'_j and x''_j to obtain estimates for y'_j and y''_j, or vice–versa. This selection depends whether $|\mathbf{T}_x| < |\mathbf{T}_y|$. The algorithm 3 solves the WLS problem for one coordinate, and deduces the estimations of the other coordinate. Algorithm 3 guarantees

Algorithm 3 Dependent coordinates WLS Solution (j)

1: **call** Set Weighted Least Squares Variables (j);
2: $\mathbf{T}_x = (ce - bf)/d$;
3: $\mathbf{T}_y = (cg - bh)/d$;
4: **if** $|\mathbf{T}_x| < |\mathbf{T}_y|$ **then** *// Considering x(y)*
5: $\quad\mathbf{T}_y$ = sign($\mathbf{T}_y$)$\sqrt{(1 - \mathbf{T}_x^2)}$;
6: $\quad\mathbf{N}_x = (af - be)/d$;
7: $\quad\mathbf{N}_y$ = -($\mathbf{T}_x\mathbf{N}_x$)/$\mathbf{T}_y$;
8: **else** *// Considering y(x)*
9: $\quad\mathbf{T}_x$ = sign($\mathbf{T}_x$)$\sqrt{(1 - \mathbf{T}_y^2)}$;
10: $\quad\mathbf{N}_y = (ah - bg)/d$;
11: $\quad\mathbf{N}_x$ = -($\mathbf{T}_y\mathbf{N}_y$)/$\mathbf{T}_x$;
12: **end if**
13: $\kappa = \mathbf{T}_x\mathbf{N}_y$ - $\mathbf{T}_y\mathbf{N}_x$;
14: $\mathbf{N}_x$ = -$\mathbf{T}_y$; $\mathbf{N}_y = \mathbf{T}_x$;

the geometrical properties of the tangent and the normal vector, that is, $\mathbf{T}$ is unitary and that $\mathbf{N}$ is orthogonal to $\mathbf{T}$. The use of the above algorithm is well suited when the curve is almost the graphic of a function in the axis: $y = f(x)$. The best axis is chosen at line 4 of Algorithm 3. However, a simple rotation helps getting closer to that case.

(c) Rotation

Least–square fitting works very well when the input points are well distributed. However even on basic cases such as a simple circle, the input points can be almost aligned vertically. To avoid this situation, following [4], we choose one of the x or y axis as reference for the parameterisation (see Section 5(b) *Dependent coordinates method*). Even though, in the case of [4], the parabola degenerates to a line when the tangent direction is at 45^o (see Figure 6). In the *dependent*

coordinates method (not in the *independent* one), the numerical precision is decreasing with the angle. To compensate this numerical error, we can compute first an estimation of the tangent with one of our methods, and then use this tangent to better distribute the samples. This operation is simply performed by a rotation on the input points before the summation (before line 12 of Algorithm 1):

$$\begin{bmatrix} x_i - x_j \\ y_i - y_j \end{bmatrix} \mapsto \begin{bmatrix} \mathbf{T}_x & \mathbf{T}_y \\ -\mathbf{T}_y & \mathbf{T}_x \end{bmatrix} \begin{bmatrix} x_i - x_j \\ y_i - y_j \end{bmatrix}.$$

(d) Boundary conditions.

We introduced our algorithm for computing the curvature at a point $\mathbf{p}_j$ with $2q+1$ samples centered at that point. However, for points close to the boundary of a curve, this condition cannot be not verified. In that case we can either reduce the width q of our sliding window, or simply compute the curvature using a non–centered window. In the last case, we do not have the theoretical guarantees of Section 4 *Theoretical framework*: the coefficient ϕ of Proposition 1. Nevertheless, our experimental results remain coherent, although less precise.

6 Experimental results

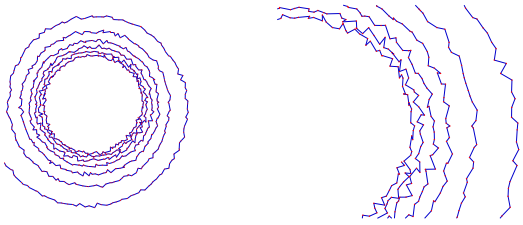
We have implemented our method with four variants: the independent and dependent coordinate method with or without a rotation to make the tangent line close to horizontal. In this section, we will discuss our results and we will compare our performance to some important methods in the literature.

(a) Experimental setting

We will discuss our tests using three curves:

- Circle: $\mathbf{r}(t) = (\cos(t), \sin(t)) : t \in [0, 2\pi]$;
- Ellipse: $\mathbf{r}(t) = (2\cos(t), \sin(t)) : t \in [0, 2\pi]$;
- Spiral: $\mathbf{r}(t) = (\frac{2}{\sqrt{t}}\cos(t), \frac{2}{\sqrt{t}}\sin(t)) : t \in [10, 50]$.

All of them were uniformly sampled in time, and therefore, the samples were equally spaced for the circle, but not for the ellipse and the spiral. The noise was simulated as a uniform



(a) Curve trace (b) Curve detail

Figure 7: *Noisy spiral curve (1000 samples, $\sigma = 1$).*

random variable in the disk of radius σ, where σ is a proportion to the average distance between consecutive samples (see Figure 7). We considered the width q of our sliding window between 1 and 30. We have also experimented the algorithm with weights given by the formula presented in Section 4 *Theoretical framework* (see Figure 9).

Since we have the parametric formula for those examples, we computed the real curvature using automatic differentiation [8]. We have measured the relative error between the unbiased estimated curvature $\hat{k}$ and the real value k by the formula:

$$RE(p_j) = \left| \frac{(\hat{\kappa}(p_j) - \hat{\kappa}_{mean}) - (\kappa(p_j) - \kappa_{mean})}{\kappa(p_j)} \right|$$

and considered the arithmetic mean of this relative error along the curve.

(b) Results

Our experimental results confirm the convergence analysis: our methods improve when increasing the number of samples (inside the same time domain), i.e. when reducing the average distance Δl between consecutive samples(see Figure 10). In the noiseless case ($\sigma = 0$), we have observed that, for all curves (not restricted to those 3) and methods considered, the relative error RE increase with q (see Figure 8). This is not surprising, since the curvature estimation should be better if we use points closer to the base point. We observe that the behaviour of our method is similar to the other ones, and that considering non–constant weights can improve those results (see Figure 9). In the noisy case we observed that the use of more sample points can improve the estimates. The ideal number of points q depends on the curve, on the sampling and on σ (see Figure 11). But we observed that even if we take more points than the ideal value, the relative error does not grow too much.

7 Conclusion

The curvature estimators that we have proposed performed experimentally close to the best in the literature. They are also robust with respect to noise and work in a great variety of sampling conditions. Notice that our method does not only estimate the curvature, but also the tangent line, the normal vector, and the osculating circle (see Figure 1).

A very important advantage is that it can be immediately generalized for the estimation of curvature and torsion of curves in $\mathbb{R}^3$. Other advantage of our method is that it can be easily implemented. The program we used for comparison is available at [13].

We plan to generalize our method to the case of point clouds in plane and also in space.

References

[1] N. Amenta, S. Choi and R. Kolluri. The Power Crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19(2–3):127–153, 2001.

[2] A. Belyaev. Plane and space curves. curvature. curvature–based features. Max–Planck–Institut für Informatik, 2004.

The corresponding work was published in the proceedings of the Sibgrapi 2004, pp. 250–257. IEEE Press, 2004.

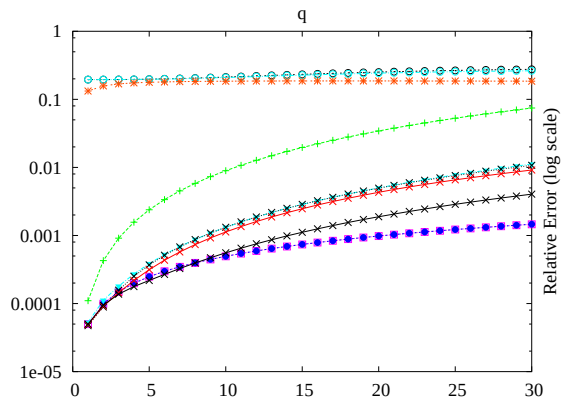


Figure 8: *Noiseless spiral with 2000 sample points ($w(\Delta l) = 1$) (legend on Figure 10(a)).*

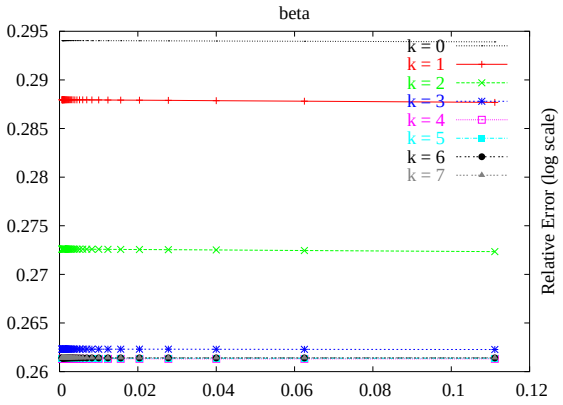
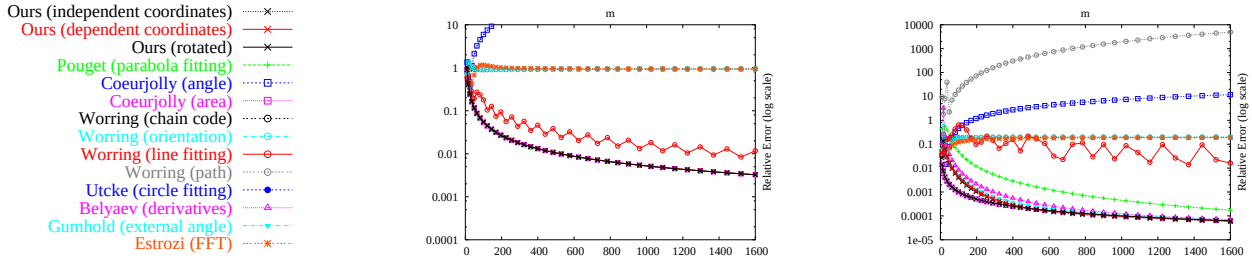


Figure 9: *Noiseless ellipse with 100 sample points, $q = 5$, with various weights $w(\Delta l) = \exp(-\beta \Delta l^2)/\Delta l^k$.*



(a) Legend. (b) Ellipse. (c) Spiral.

Figure 10: *Convergence when the sampling rate increases ($\sigma = 0, w(\Delta l) = 1$): Not all the methods we introduced in the prior work converges when $\Delta l \to 0$, especially for irregularly sampled curves as (c).*

[3] H. H. Biscaro, A. Castelo and G. Nonato. A topological approach to curve reconstruction from scattered points. *Prépublicações do ICMC–USP*, 2004.

[4] F. Cazals, F. Chazal and T. Lewiner. Molecular shape analysis based upon Morse–Smale complex and the Connolly function. In *Symposium on Computational Geometry*, pages 351–360. ACM, 2003.

[5] D. Coeurjolly, M. Serge and T. Laure. Discrete curvature based on osculating circles estimation. In *Lecture Notes in Computer Science*, volume 2059, pages 303–312. Springer, 2001.

[6] L. da Fontoura Costa and R. M. Cesar Jr. *Shape analysis and classification*. CRC Press, 2000.

[7] L. F. Estrozi, L. G. R. Filho, A. G. C. Bianchi, R. M. Cesar Jr and L. da Fontoura Costa. 1D and 2D fourier–based approaches to numeric curvature estimation and their comparative performance assessment. *Digital Signal Processing*, 13:172–197, 2003.

[8] A. Griewank. *Evaluating derivatives, principles and techniques of algorithmic differentiation*, volume 19 of *Frontiers in Applied Mathematics*. SIAM, 2000.

[9] S. Gumhold. Designing optimal curves in 2d. In *CEIG*, pages 61–76, 2004.

[10] P. Lancaster and K. Salkauskas. *Curve and surface fitting: an introduction*. Academic Press, 2002.

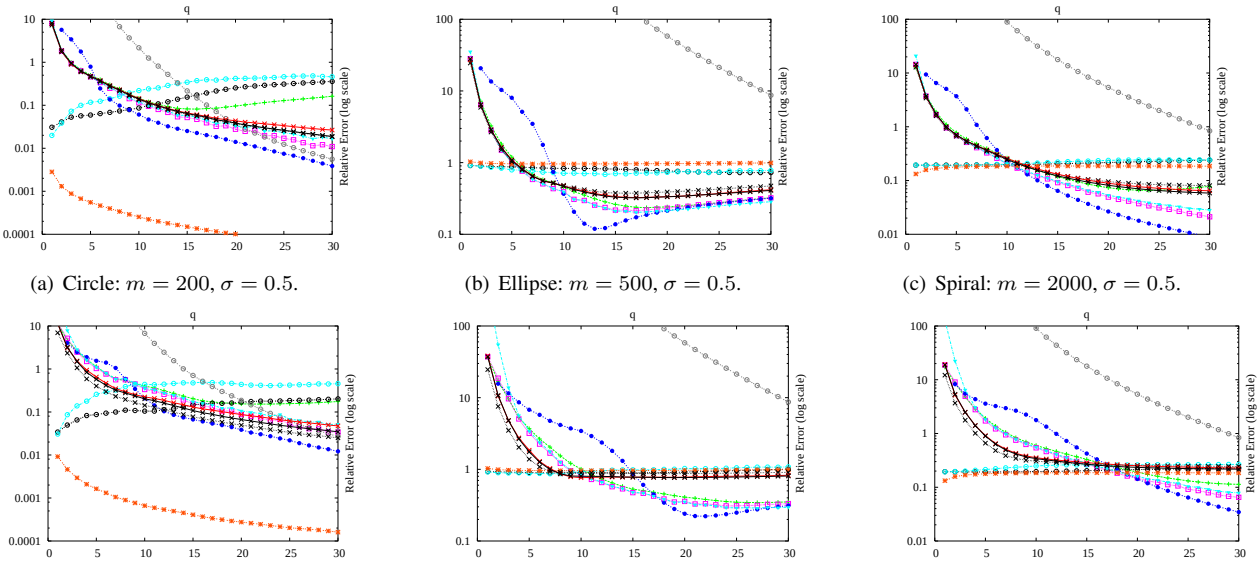
[11] T. Lewiner, J. Gomes Jr, H. Lopes and M. Craizer. Arc–length based curvature estimator. In *Sibgrapi*, pages 250–257, Curitiba, Oct. 2004. IEEE.

[12] T. Lewiner, J. Gomes Jr, H. Lopes and M. Craizer. Curvature estimation: theory and practice. Technical report, Department of Mathematics, PUC–Rio, 2004.

[13] T. Lewiner, J. Gomes Jr, H. Lopes and M. Craizer. http://www.mat.puc–rio.br/tomlew/-curvature.zip, 2004.

[14] H. Lopes, J. Rossignac, A. Safonova, A. Szymczak and G. Tavares. Edgebreaker: a simple compression for surfaces with handles. In C. Hoffman and W. Bronsvort, editors, *Solid Modeling and Applications*, pages 289–296, Saarbrücken, Germany, 2002. ACM.

[15] F. Mokhtarian and A. K. Mackworth. A theory for multiscale, curvature based shape representation for planar curves. *Transactions on Pattern Analysis and Machine Intelligence*, 14:789–805, 1992.



(a) Circle: $m = 200$, $\sigma = 0.5$. (b) Ellipse: $m = 500$, $\sigma = 0.5$. (c) Spiral: $m = 2000$, $\sigma = 0.5$.

(d) Circle: $m = 200$, $\sigma = 1.5$. (e) Ellipse: $m = 500$, $\sigma = 1.5$. (f) Spiral: $m = 2000$, $\sigma = 1.5$.

Figure 11: *Noise in the samples ($w(\Delta l) = 1$) (legend on Figure 10(a)): the circle fitting of [19] minimizes its error for a specific value of q as specified in [19], however the minimization sometimes degenerates ((a),(d)). The FFT–based method of [7] is very robust to noise, although not being always optimal for larger q. Our results are never far from the best ones, although the best methods differ from case to case.*

[16] N. M. Patrikalakis and T. Maekawa. *Shape interrogation for computer aided design and manufacturing*. Springer, New York, 2002.

[17] V. Pratt. Direct least–squares fitting of algebraic surfaces. *Computers & Graphics*, 21(4):145–152, 1987.

[18] J. A. Sethian. *Fast marching methods and level set methods*. *Cambridge Monograph on Applied and Computational Mathematics*. Cambridge University Press, 1999.

[19] S. Utcke. Error–bounds on curvature estimation. *Lecture Notes in Computer Science*, 2695:657–666, 2003.

[20] M. Worring and A. W. M. Smeulders. Digital curvature estimation. *CVGIP: Image Understanding*, 58(3):366–382, 1993.

The corresponding work was published in the proceedings of the Sibgrapi 2004, pp. 250–257. IEEE Press, 2004.