A Comparative Study on 2D Curvature Estimators

Simon Hermann
Center for Informatics,
University of Göttingen,
Göttingen, Germany

Reinhard Klette
Department of Computer Science,
University of Auckland
Auckland, New Zealand

Abstract

Curvature is a frequently used property in two-dimensional (2D) shape analysis, directly or for derived features such as corners or convex and concave arcs. This paper presents curvature estimators which follow approaches in differential geometry. Digital-straight segment approximation (as known from digital geometry) is used in those estimators. Results of multigrid experiments are evaluated leading to a comparative performance analysis of several curvature estimators.

1 Introduction

The curvature of digital curves needs to be estimated in a wide variety of pattern recognition or image analysis applications. Because these curves consist of discrete points, no analytic (i.e., parametric) representation is at hand, and curvature can therefore not be computed as in differential geometry. The goal of methods proposed for curvature estimation is to decide whether a point on a discrete curve has a particular curvature, for example compared to those at other points in some neighborhood, or for segmenting curves into convex or concave arcs.

Proposed measures often do not correspond to the real (but unknown) curvature values of the underlying (i.e., prior to digitization) curve but based on heuristics. As far as they are designed for corner detection, such an approach may behave well in applications. See, for example, methods proposed in [1, 4, 12]. A *corner* is (informally) defined as a "high-curvature point" on a simple digital arc or curve. In early days of image analysis, low resolution images did not allow for more precise measurements, anyway. But nowadays, high resolution images support to define curvature estimation approaches which are derived from definitions of differential geometry. A review on curvature methods is given in two chapters of [7].

Section 2 presents curvature estimation methods based on three different categories of curvature definition, also adding estimators which use cubic splines, and in general with an emphasis on using digital straight segment (DSS) approximations (for related algorithms, see, e.g., [7]). We tested these estimators on specific geometric objects and derived some qualitative analysis of their performance when increasing grid resolution. Finally, a critical discussion of methods is given. Altogether, the paper contributes by proposing new curvature estimators and a testing methodology.

2 Curvature Estimation

We consider 8-curves ρ in digital pictures defined on $\mathbb{Z}^2$. In order to detect a corner at pixel p_i on a curve ρ, it is common practice to consider an angular measure based on a predecessor p_{i-k_b}, p_i itself, and a successor p_{i+k_f}, where $k_b, k_f > 0$ are constant, scaled (e.g., both equal to $k = 0.02 \cdot n$, where n is the number of pixels on ρ), or variables within a defined interval. Angular measures, resulting from k_b and k_f, are used to identify p_i as a "high curvature point".

Obviously, such non-adaptive specifications of values of k_b or k_f do not reflect the shape of the given digital curve. Adaptive (unique) specifications of k_b or k_f can be based, for example, on DSS approximation (see, e.g., [2, 6]); those values reflect the shape of the curve.

2.1 Derivative of the Tangent Angle

The curvature estimation method of [6] is an example for defining curvature based on changes in orientations of the tangent. Let p and q be two points on a plane curve, and δ the angle between positive directions of both tangents at those points. Curvature κ at p is defined to be the limit

$$\kappa(p) = \lim_{pq \to 0} \frac{\delta}{pq}$$

Algorithm **HK2003** uses backward (ending at p_i) and forward (beginning at p_i) DSSs for approximating the tangent at p_i. Function d_2 refers to the Euclidean distance.

Algorithm 1 Curvature Estimation **HK2003**

```
Compute-curvature(Curve $\rho$)
For point $p_i$ in $\rho$ do
$k_b$ ($k_f$) is the length of the longest backward (forward)
    DSS starting at $p_i$
$l_b = d_2(p_{i-k_b}, p_i)$ and $\theta_b = \tan^{-1}\left(\frac{|x_{i-k_b} - x_i|}{|y_{i-k_b} - y_i|}\right)$
$l_f = d_2(p_{i+k_f}, p_i)$ and $\theta_f = \tan^{-1}\left(\frac{|x_{i+k_f} - x_i|}{|y_{i+k_f} - y_i|}\right)$
compute $\theta = \frac{1}{2} \cdot \theta_b + \frac{1}{2} \cdot \theta_f$
compute $\delta_b = |\theta_b - \theta|$ and $\delta_f = |\theta_f - \theta|$
    (Note that $\delta_b = \delta_f$.)
return $\frac{\delta_b}{2l_b} + \frac{\delta_f}{2l_f}$
```

Only positive curvature values are returned. Convexity or concavity of arcs cannot be decided just based on those values, but we can classify both situations when also using coordinates of p_{i-k_b}, p_i and p_{i+k_f}.

2.2 Radius of the Osculating Circle

The osculating circle at a point p on a smooth curve γ is defined in differential geometry by starting with a circle that intersects γ at p and also at two points p_b and p_f (left and right of p); see dashed circle in Figure 1. Moving both points into p results into the osculating circle at p with center c. The absolute value of curvature at point p is then defined as the reciprocal value of the radius $r = d_2(c, p)$.

The following calculation of the osculating circle makes use of the geometric property that three points uniquely define a circle. The parameters can be computed by the intersection of two bisectors, which are two different sides of the triangle defined by p_{i-k_b}, p_i and p_{i+k_f}. At point p_i we compute two DSSs as in **HK2003**. The algorithm is as follows:

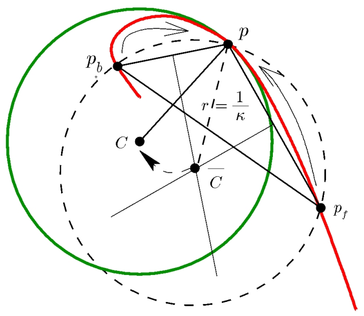

Figure 1. The dashed circle is incident with points p_b, p, and p_f; it 'moves' into the osculating circle centered at c. The triangle illustrates the discrete estimation in HK2005.

Algorithm 2 Curvature Estimation **HK2005**

```
Compute-curvature(Curve $\rho$)
For point $p_i$ in $\rho$ do
$k_b$ ($k_f$) is the length of the longest backward (forward)
    DSS starting at $p_i$
compute bisecting lines $g_b$ and $g_f$ of segments $p_{i-k_b}p_i$
    and $p_{i+k_f}p_i$
compute $c$ as intersection of $g_b$ and $g_f$
compute radius $r = d_2(c, p_i)$
return $\frac{1}{r}$
```

2.3 Derivative of the Curve

A parametrized curve $\gamma(t) = (x(t), y(t))$ allows to calculate curvature based on derivatives; the curvature is as follows

$$\kappa = \frac{\begin{vmatrix} x' & y' \\ x'' & y'' \end{vmatrix}}{(x'^2 + y'^2)^{\frac{3}{2}}} \tag{1}$$

Using second order curves. Algorithm **M2003** [10] approximates the digital curve ρ locally at p_i by second order polynomials using also pixels p_{i-k_b} and p_{i+k_f}. The approximating polynomial $\gamma(t) = (x(t), y(t))$ is defined by $x(t) = a_2t^2 + a_1t + a_0$ and $y(t) = b_2t^2 + b_1t + b_0$ with $t \in [-1, 1]$. Let $t = -1$ define p_{i-k}, $t = 0$ specifies pixel p_i, and $t = 1$ defines p_{i+k}. In this particular case, Equation (1) takes the following form: $\kappa = \frac{2(a_1b_2 - a_2b_1)}{(a_1^2 + b_1^2)^{\frac{3}{2}}}$ at point p_i.

Using spline curves. [11] uses approximating cubic B-spline curves for representing a digitized curve. Based on curvature information, corners are detected and control points are consecutively adjusted in order to fit the curve more closely. The local approach in [9] is also based on approximating splines. We can also approximate the original curve globally by an interpolating periodic spline curve. Suppose we compute a decomposition of a digital curve using DSS segmentation. We can then use the points between two consecutive DSSs as interpolating points (let us call them *DSS break points*) and get a close and smooth analytical representation of the underlying digital object. See Figure 2. By differentiating these parametric curves we compute then (positive or negative !) curvature at every sample point.

Since the curvature is computed on sample points and not on grid points itself, we need to map the curvature at sample points onto pixels. We propose two possible mappings:

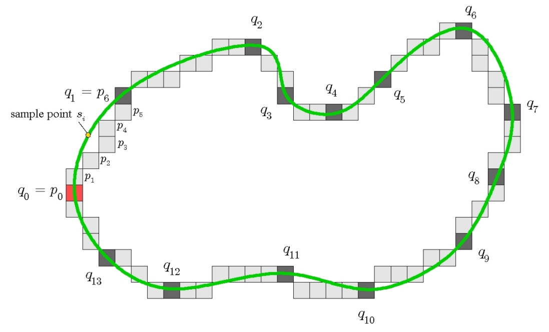

Figure 2. Interpolating periodic spline based on break points of the DSS segmentation.

(i) *Interpolation.* Suppose we have a DSS segmentation with break points $q_0 \ldots q_n$. Computing chord length parametrizations $t_1, \ldots, t_n$, we know that $s(t_i) = q_i$. This allows to compute the curvature at all interpolating points q_i. Once we know the curvature at every p_i we use linear interpolation to assign a curvature value to grid points p_i between q_i and q_{i+1}, for $i = 0, \ldots, n-1$ (see Figure 2).

(ii) *Averaging.* In this mapping, all sample points s_i of the spline are taken into account. The idea of this mapping is as follows: All grid points p_i receive a certain amount of sample points assigned to them, namely all sample points which are closer to p_i than to any other grid points. We define the mean of the sum of all curvature values of all sample points assigned to one grid point p_i as the curvature at p_i.

Experiments showed that the difference between both mappings is minor. especially for larger grid resolution. Therefore we recommend interpolation since it is unique, and not many sample points have to be computed.

3 Multigrid Analysis

We analyze the introduced measures with respect to multigrid convergence (see, for example, [7] for this evaluation strategy which corresponds to high-resolution imaging: ideal mathematical objects are digitized with increasing grid resolution, and behavior of algorithms is analyzed on those multigrid input data).

We tested curvature estimation on two geometric objects, the ellipse and the hyperbolic spiral.

3.1 Experiments on the Ellipse

We used Gauss digitization to digitize ellipses, each with increasing grid resolution. Used elliptical regions are defined by $\frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1$, where $a = 2 \cdot b$ and $10 \leq b \leq 520$. We extracted 8-curves via border tracking of resulting digital ellipses, which are the input for the curvature estimators. The curvature at point $p = (x, y)$ on the elliptical curve equals $\frac{1}{\kappa} = a^2 b^2 \left(\frac{x^2}{a^4} + \frac{y^2}{b^4} \right)$.

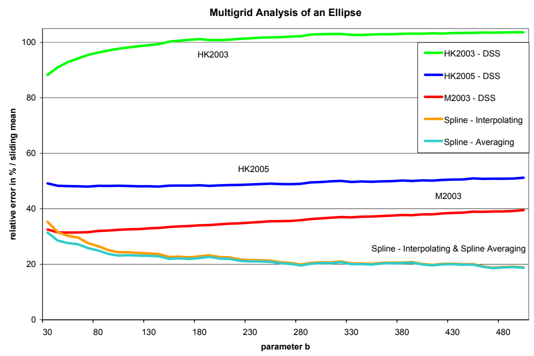

Figure 3. Relative error curves, using the DSS approach.

But how to find the corresponding point $p = (x, y)$ for a border pixel p_i on the ellipse in order to compute the absolute or relative error?

A first option is that we identify p with p_i. A second option is that we choose p as the intersection of the ellipse $\frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1$ with the straight line $y = \frac{y_i}{x_i} x$. The resulting difference between both options proved to be of marginal impact, and errors are close to zero for $b > 200$. We decided for the second option.

For every size (or resolution) b, we computed the mean m_b of absolute and relative errors of estimated curvature, at every border pixel.

Resulting scattered points are filtered by a sliding mean using $\frac{1}{39} \sum_{i=-19}^{19} m_{b+i}$ and drawn in increments of 10 into the diagrams. Our DSS-based algorithms are also modified such that calculated values of k_b and k_f are replaced by a uniform value of $k = 0.02 \cdot n$. Looking at absolute errors, all (using the DSS-approach) estimators seem to be multigrid convergent.

But looking at relative errors in Figure 3, only the error curve of the estimator based on the spline approach shows convergent behaviour. [Note that we draw both spline approaches (interpolating and averaging) into this diagram. Since the results are similar for increasing resolutions, we decided to use only the error curve for the interpolating approach.] All other curves seem to be slightly divergent for increasing grid resolution, but do not depart more than 7% from the smallest error value. The relative error for **HK2003** exceeding 100% is relatively large.

Looking at absolute errors when using a global constant $k = 0.02 \cdot n$, all estimators also seem to be multigrid convergent, and it is noticable that a uniform,i.e. global k (see legend in figures) is of benefit compared to the original DSS-approach. This is probably due to the fact that an ellipse is a "uniformly smooth" curve; see also our discussion in the next section.

But in contrast to the DSS-approach, all relative error curves seem to be convergent when using the global con-

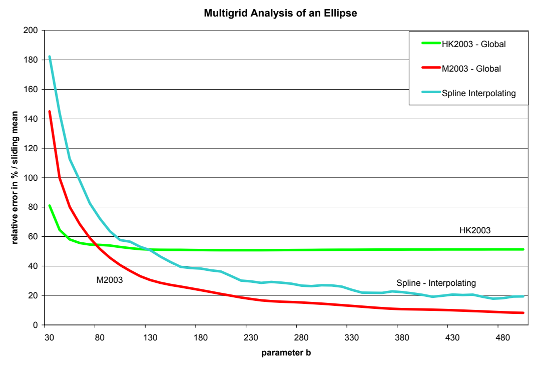

Figure 4. Relative error curves, replacing DSS by $k = 0.02n$.

stant $k = 0.02 \cdot n$. Since the error curves of **HK2005** and **M2003** are very similar to each other, we left out the one for **HK2005**. Note that **HK2005** is fitting circles to the curve, which is a special case of a second order curve. As **M2003** is fitting general second order polynomials to the underlying digital curve, we assume **M2003** to be superior over **HK2005**. Looking at the spline curve based on a global con-

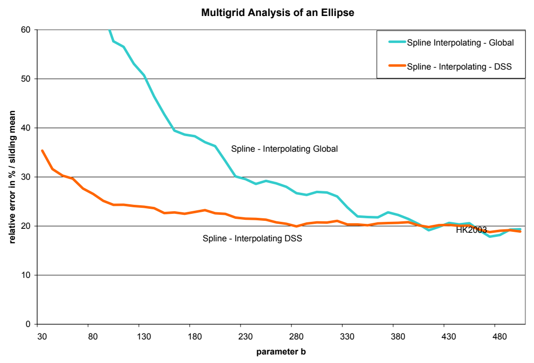

Figure 5. Comparison between spline curves, using either DSSs or global constant.

stant, we can at first see a similar behavior, but for $b < 300$ the DSS-based spline method takes the lead; see Figure 5. The DSS-based spline method performed best when testing on a circle (i.e., multigrid studies).

3.2 Experiments on the Hyperbolic Spiral

We assume the Euclidean plane with the origin as pole and the x-axis as distinct oriented line. We can transform polar coordinates into Cartesian coordinates by $(r, \varphi)_{polar} = (r \cdot cos(\varphi), r \cdot sin(\varphi))_{Cartesian}$. The hyperbolic spiral is defined in polar coordinates as $\rho = \frac{a}{\varphi}$, with $a > 0$. Its curve consists of two branches which are mirror-symmetric to the y-axis. Both branches are asymptotic towards $y = a$ and the origin. The osculating circle for angle φ equals $\frac{1}{\kappa} = \frac{a}{\varphi}\left(\frac{\sqrt{1+\varphi^2}}{\varphi}\right)^3$. In our multigrid experiments we digitized the hyperbolic spiral by sampling the curve into 100,000 points, for an angle φ between $\frac{\pi}{6} \leq \varphi \leq 2\pi$. For any sample point, we took the closest grid point as an object pixel (of course, more than one sample point correspond to one object pixel in general). We choose $a = 1.0$, thus we get the polar coordinates $(\rho = \frac{1}{\varphi}, \varphi)$ for the sample points, or $(\rho \cdot cos\varphi, \rho \cdot sin\varphi)$ in Cartesian coordinates. We introduced a scaling factor s with $1 \leq s \leq 1,520$ and multiplied it with the Cartesian coordinates. Thus we have finally $(\frac{s}{\varphi} \cdot cos\varphi, \frac{s}{\varphi} \cdot sin\varphi)$. Note that increasing the scaling factor s by 1 corresponds to an increment of a by 1.

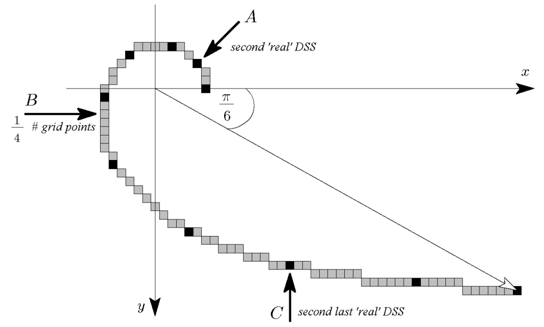

Figure 6. Digitization of the hyperbolic spiral for $\varphi \in [\frac{\pi}{6}, 2\pi]$ and scalingfactor $s = 25$.

In our experiments we used DSS-based k-values or a global constant (again, we choose $k = 0.02 \cdot n$). Results for **HK2005** and **M2003** proved to be nearly identical, and we only show **M2003**.

While tracking the border of the spiral, we computed a first maximum-length 8-DSS starting at grid point corresponding to $(s, 2\pi)$ and ending at grid point corresponding to $(s, \frac{\pi}{6})$. Then we estimated the curvature at all points between point A (see Figure 6) and point C. (Alternatively, we could also start with a maximum DSS from the end of the spiral.) For object pixel p, we find the corresponding point on the spiral by taking the line intersecting p and the origin, and measuring the distance and the angle with respect to the x-axis.

For every scaling factor s, we computed the mean m_s of the error at all border pixels between point A and point C. Again we applied the sliding mean filter, and draw results in steps of 10 into the diagram. Also again, all estimators for both approaches (DSS or global) seem to be multigrid convergent with respect to absolute errors. For higher resolutions, both approaches of **HK2003** appear to be identical. **M2003** has smaller error values for higher resolutions when applying the global constant, but errors go faster towards zero for the DSS approach: both curves intersect about at $s > 330$.

In the second experiment we used the same setup but measure the relative error in percent. Looking at relative errors in Figure 7 we notice that curves for **HK2003** are

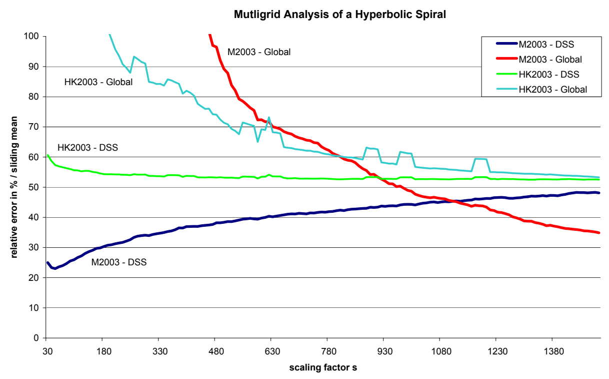

Figure 7. Multigrid experiment estimating relative errors of curvature on a hyperbolic spiral.

converging towards an relative error of 55%. Again we notice that the DSS approach is converging faster.

For **M2003** we again observe a divergent behaviour of the DSS-approach and a convergent behaviour for the global approach. But we have to point out that for resolutions between 30 and 1,000 the error of the DSS approach remains clearly below 45% while the error for the global constant does not drop below 100% for resolutions smaller than 450. It is interesting to ask for the reason for this behaviour, since we expected a faster convergence for the global constant based on the results for the ellipse.

4 Discussion

A recent work [8] indicates that estimation of curvature in digitized images is barely possible with an error better than 40% without using subpixel accuracy and numerical optimization, even in high resolution images. Keeping this in mind we can conclude as follows.

Applying the DSS approach for curvature estimators does not seem to guarantee multigrid convergence (in difference to DSS-based multigrid convergent length estimation). However, from our experiments we could also conclude that DSS based curvature estimators have a good overall performance, even for low resolutions. (This is probably due to the fact that no parameters have to be adjusted, since the shape of the curve is reflected in the DSS.)

The global constant allowed for some superior results over the DSS approach for the high-curvature section of the hyperbolic curve. The global constant seems to guarantee in general multigrid convergence, at least in those experiments. But it is easy to construct examples in which the global constant fails, due to non consideration of the underlying curve. (Imagine, for example, a sine-like curve with p_i on a peak and a k such that the p_{k_b} and p_{k_f} lie on neighboring peaks, thus resulting in nearly no angle.)

Estimated values at segments of high curvature seem to be more accurate than those at segments of low curvature.

The spline-based curvature estimation performs best and is highly recommended for future curvature estimators when using very-high (from today's point of view) resolution images.

More details about curvature estimations and related experiments can be found in [5].

References

[1] H. L. Beus and S. S. H. Tiu. An improved corner detection algorithm based on chain-coded plane curves. *Pattern Recognition*, **20**:291–296, 1987.

[2] D. Coeurjolly, S. Miguet, and L. Tougne. Discrete curvature based on osculation circle estimation. In Proc. *Int. Workshop Visual Form*, LNCS 2059, pages 300–312, Springer, Berlin, 2001.

[3] M. Epstein. On the Influence of parametrization in parametric interpolation. *SIAM J Numer. Analysis*, **13**:261–268,1976.

[4] H. Freeman and L. S. Davis. A corner finding algorithm for chain-coded curves. *IEEE Trans. Computers*, **26**:297–303, 1977.

[5] S. Hermann. Feature analysis of digital curves. MSc thesis, Computer Science Department, The University of Auckland, 2005.

[6] S. Hermann and R. Klette. Multigrid analysis of curvature estimators. In Proc. *Image Vision Computing New Zealand*, pages 108–112, Massey University, 2003.

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

[8] V. Kovalevsky. Curvature in digital 2D images. *Int. J. Pattern Recognition Artificial Intelligence*, **15**:1183–1200, 2001.

[9] F. Lu and E. E. Milios. Optimal local spline approximation of planar shape. In Proc. *Int. Conf. Acoustics Speech Signal Processing*, Vol. 4, pages 2469–2472, 1991.

[10] M. Marji. On the detection of dominant points on digital planar curves. PhD thesis, Wayne State University, Detroit, Michigan, 2003.

[11] G. Medioni and Y. Yasumoto. Corner detection and curve representation using cubic B-splines. In Proc. *Robotics and Automation*, Vol. 3, pages 764-769,1986.

[12] A. Rosenfeld and J.S. Weszka. An improved method of angle detection on digital curves. *IEEE Trans. Computers*, **24**:940–941, 1975.