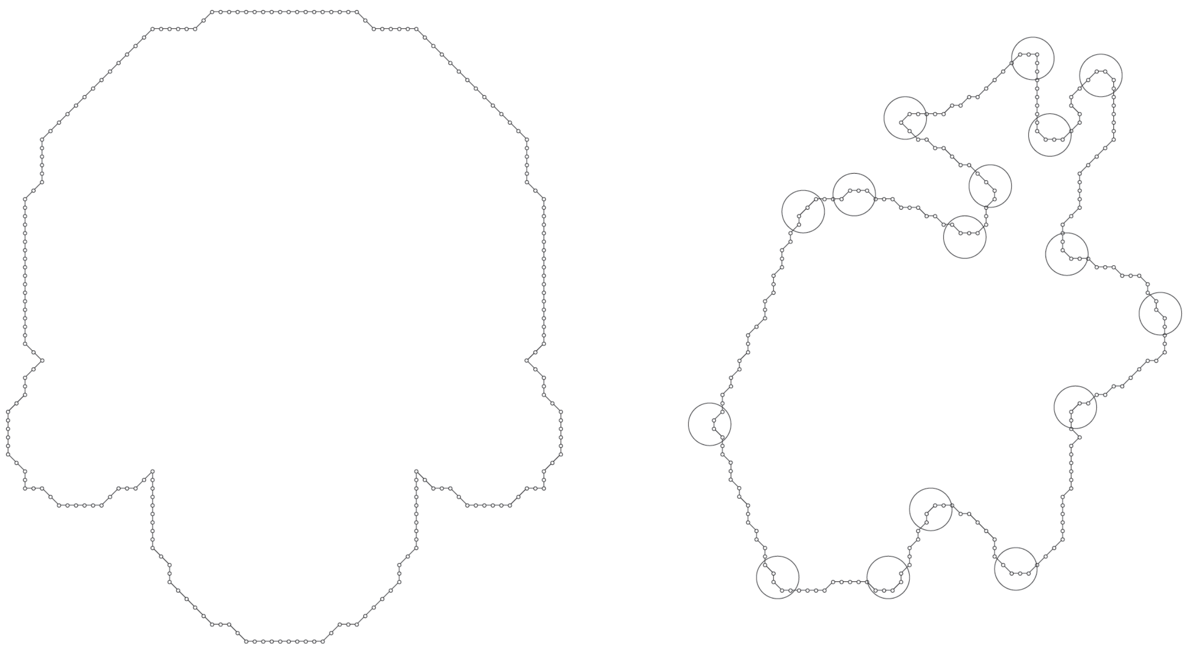
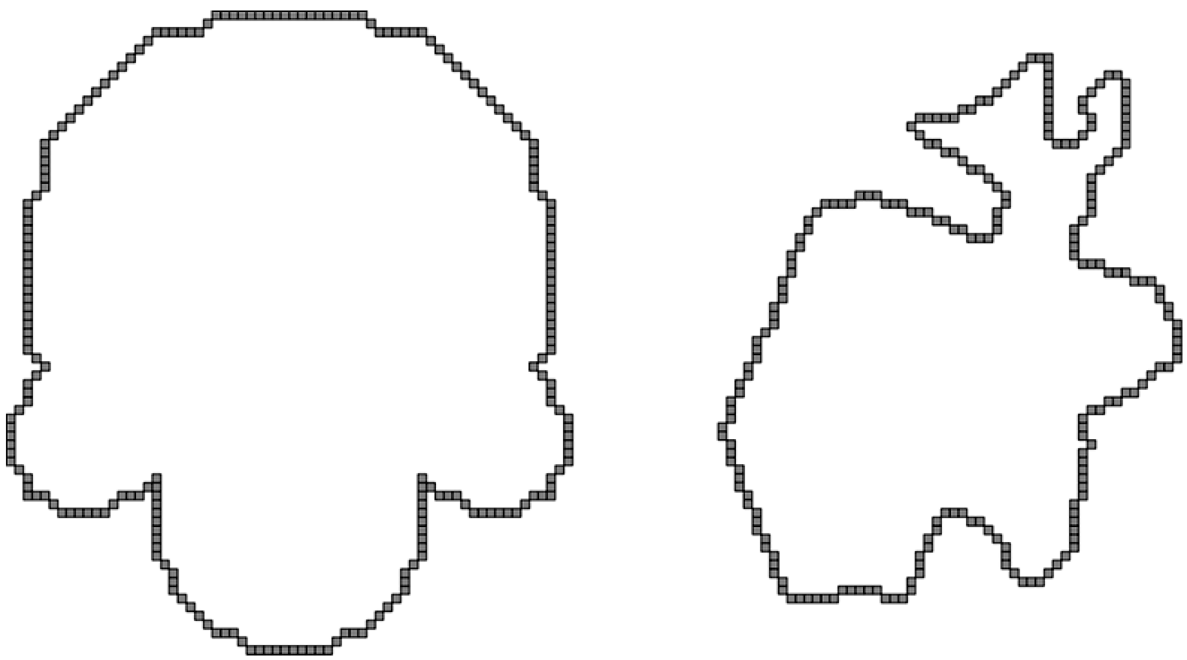
Curvature of Digital Curves



Left: a symmetric curve (i.e., results should also be "symmetric"). Right: "high-curvature pixels" should correspond to visual perception of "corners".



Categories of Methods (Algorithms)

Curvature can be estimated from

(C1.1) the change in the slope angle of the tangent line (e.g., relative to the x-axis);

(C1.2) derivatives along the curve; or

(C1.3) the radius of the osculating circle (also called *circle of curvature*).

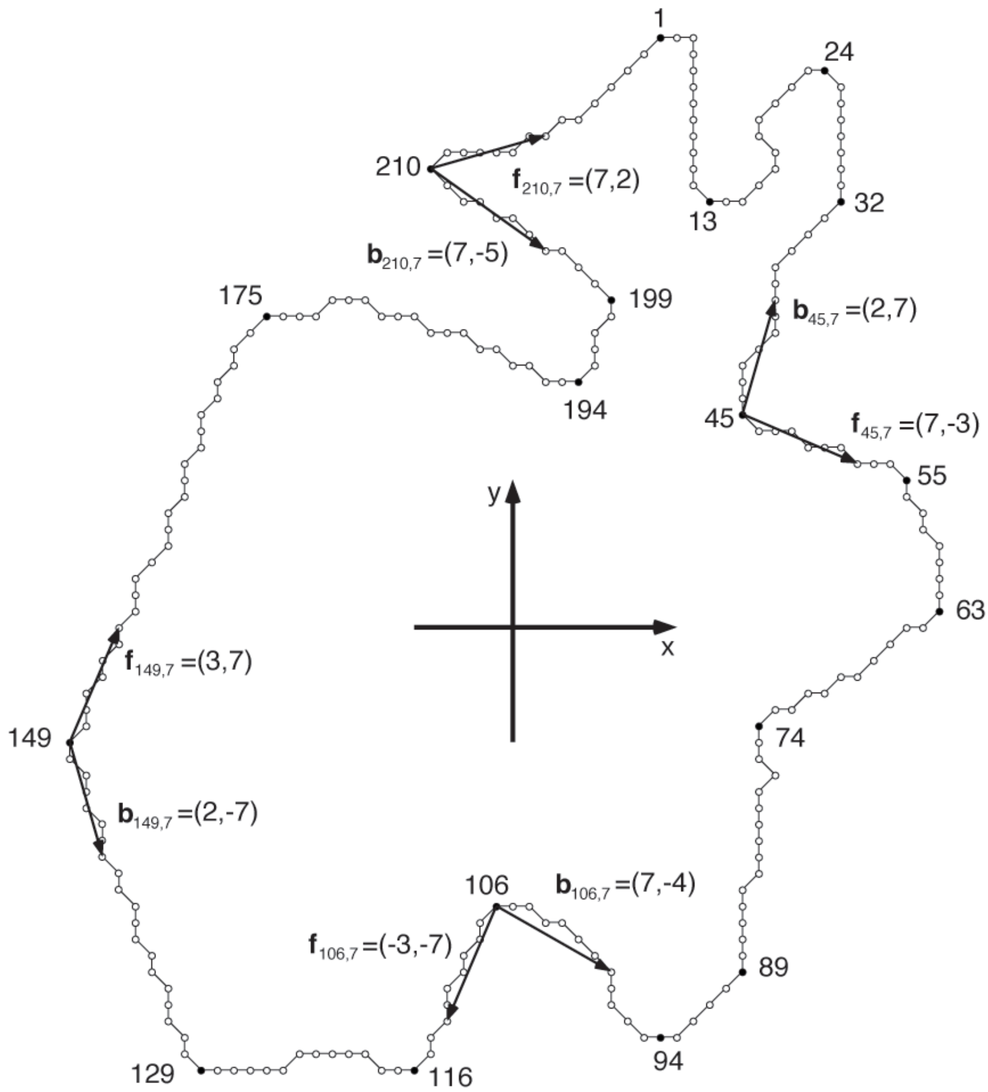
Basically, **(C1.2)** is identical to **(C1.1)**, but **(C1.1)** is towards geometric specifications of tangents, and estimation of angles of these lines, but **(C1.2)** is about estimating derivatives, without geometric constructions of tangent lines.

Let $\rho = \langle p_1, \ldots, p_m \rangle$ be an α-curve (usually an 8-curve) in $\mathbb{Z}^2$. Let $p_i = (x_i, y_i)$, where $i = 1, \ldots, m$, and let $1 \leq k \leq m$. For each pixel p_i on ρ, we define a

forward vector $\mathbf{f}_{i,k} = p_i - p_{i+k}$ and a

backward vector $\mathbf{b}_{i,k} = p_i - p_{i-k}$,

where the indices are modulo m. Let $\mathbf{f}_{i,k} = (x^+_{i,k}, y^+_{i,k})$ and $\mathbf{b}_{i,k} = (x^-_{i,k}, y^-_{i,k})$. In the figure we use $k = 7$:



Parameter $k \geq 2$ should be selected in dependency of the shape complexity of given digital curves.

Algorithm FD1977

[H. Freeman and L.S. Davis, 1977]

This algorithm (in category **(C1.1)**) estimates changes in the slope angles ψ_i of the tangent lines at points p_i. Let

$$\theta_{i,k} = \begin{cases} \tan^{-1}\left(y^-_{i,k}/x^-_{i,k}\right), & \text{if } \left|x^-_{i,k}\right| \geq \left|y^-_{i,k}\right| \\ \cot^{-1}\left(x^-_{i,k}/y^-_{i,k}\right) & \text{otherwise} \end{cases}$$

This is not an estimate of ψ_i because this angle estimate is solely based on backward vectors of "length" k.

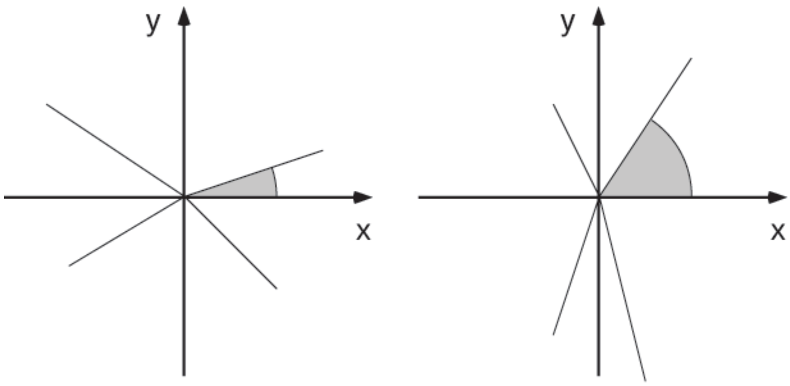


Figure 1: Left: $\left|x^-_{i,k}\right| \geq \left|y^-_{i,k}\right|$ implies that $x^-_{i,k} \neq 0$. Right: $y^-_{i,k} \neq 0$.

At pixel p_i we consider the *centered difference*

$$\delta_{i,k} = \theta_{i+1,k} - \theta_{i-1,k}$$

of slope angles at p_{i-1} and p_{i+1}, and this is a *curvature estimate* (of differences between ψ_{i-1} and ψ_{i+1}).

The method can be further extended to corner detection. With $\Delta = \tan^{-1}(1/(k-1))$ we define a small angle which defines possible deviations of a sequence of pixels from "being straight" such that it is still considered to be "reasonably straight". We calculate maximum lengths

$$t_1 = \max\{t : \forall s \, (1 \le s \le t \rightarrow -\Delta \le \delta_{i-s,k} \le \Delta)\}$$
$$t_2 = \max\{t : \forall s \, (1 \le s \le t \rightarrow -\Delta \le \delta_{i+k+s,k} \le \Delta)\}$$

of two arcs, preceding p_i or following p_{i+k}, in which the differences $\delta_{j,k}$ remain close to zero (in the interval $[-\Delta, +\Delta]$). These two arcs can be understood as "legs" of a corner at pixels $p_i, \ldots, p_{i+k}$.

Now, finally, for better stability of estimated differences in slope angles, the centered differences are "accumulated" for $k+1$ pixels, and weighted by the logarithms, for example to basis e, of the lengths of these two "legs":

$$E_{i,k} = \ln t_1 \cdot \ln t_2 \cdot \sum_{j=i}^{i+k} \delta_{j,k}$$

> A corner is detected at p_i iff $E_{i,k} > T$ and the previous corner is at distance of at least k from p_i on ρ.

Note that this procedure depends on a parameter k and on a threshold T. Values $E_{i,k}$ are curvature estimates.

(Note on ln: $\ln 2 = 0.693$ and $\ln 3 = 1.099$ [difference is 0.406], but $\ln 20 = 2.996$ and $\ln 21 = 3.045$ [difference is 0.049].)

Algorithm BT1987

[H.L. Beus and S.S.H. Tiu, 1987]

This algorithm modifies algorithm **FD1977** as follows:

(i) t_1 and t_2 are now upper-bounded by $\lfloor bm \rfloor$ (with a parameter $0 < b < 1$, and m is the number of pixels in the curve) and

(ii) the curvature estimates $E_{i,k}$ are calculated and averaged over a range of values of k (with $k_L \leq k \leq k_U$):

$$E_i = \frac{1}{k_U - k_L + 1} \sum_{k=k_L}^{k_U} E_{i,k}$$

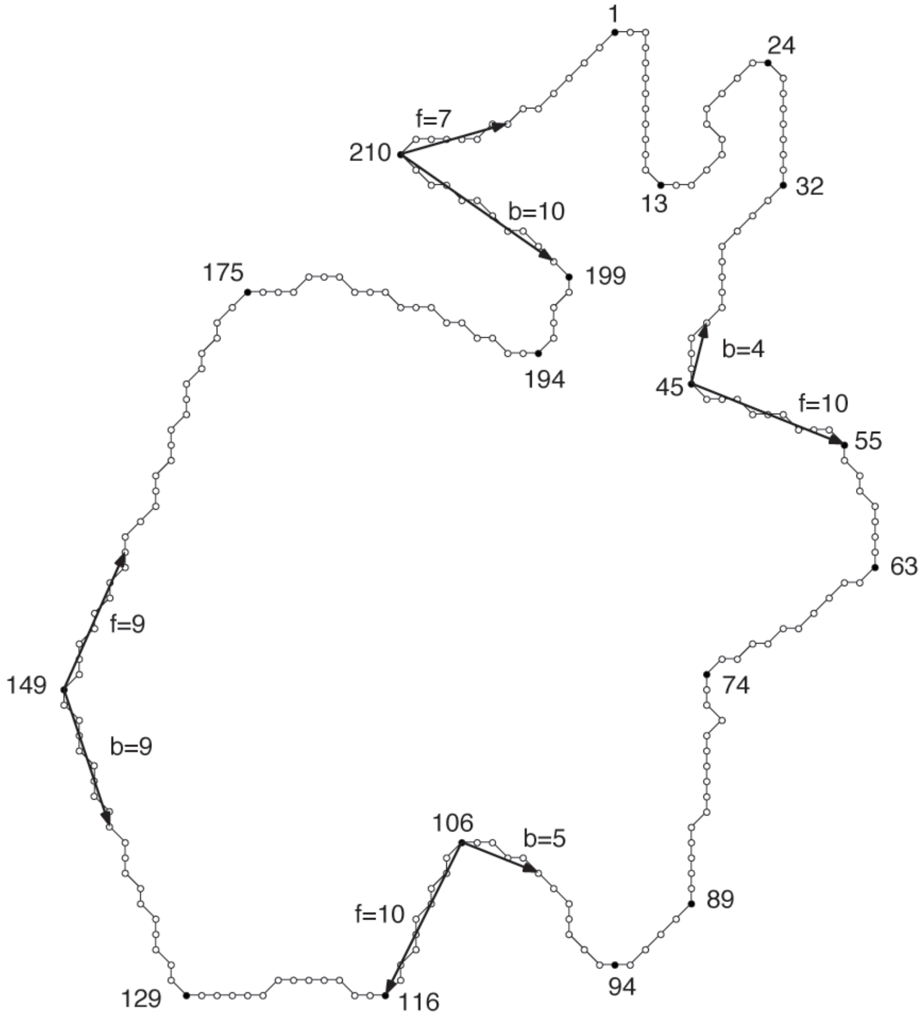
This method involves now four parameters: k_L, k_U, b, and the threshold T, instead of just two parameters k and T.

> Note that an increase in numbers of parameters is in general a drawback in picture analysis due to uncertainties how to optimize these parameters (i.e., how to choose "'good ones").

From the original paper: "The output of the algorithm is a list of the M-most corner-like nodes ordered from highest to lowest cornerity."

Algorithm HK2003

[S. Hermann and R. Klette, 2003]



This algorithm of category **(C1.1)** calculates a maximum-length 8-DSS $p_{i-b}p_i$ of the following Euclidean length

$$l_b = \sqrt{(x_{i-b} - x_i)^2 + (y_{i-b} - y_i)^2}$$

that goes "backward" from p_i and a maximum-length 8-DSS p_ip_{i+f} of the following length

$$l_f = \sqrt{(x_{i+f} - x_i)^2 + (y_{i+f} - y_i)^2}$$

that goes "forward" from p_i on the given 8-curve.

It then calculates the following angles,

$$\theta_b = \tan^{-1}\left(\frac{|x_{i-b} - x_i|}{|y_{i-b} - y_i|}\right) \quad \text{and} \quad \theta_f = \tan^{-1}\left(\frac{|x_{i+f} - x_i|}{|y_{i+f} - y_i|}\right)$$

(if the y-difference is zero, then apply $\cot^{-1}$), the mean

$$\theta_i = \theta_b/2 + \theta_f/2$$

and the angular differences

$$\delta_f = |\theta_f - \theta_i|$$

and

$$\delta_b = |\theta_b - \theta_i|$$

Finally, the curvature estimate at p_i is as follows:

$$E_i = \frac{\delta_f}{2l_f} + \frac{\delta_b}{2l_b}$$

Note that $\delta_f = \delta_b$; we also have that

$$E_i = \delta_f(l_b + l_f)/2l_f l_b$$

Algorithm M2003

[F. Mokhtarian and A. Mackworth, 1986] is an early example of an algorithm in category **(C1.2)**. More recently, [M. Marji, 2003] assumes that the given 8-curve $\langle p_1, \ldots, p_m \rangle$, where $p_j = (x_j, y_j)$ for $1 \leq j \leq m$, is sampled along parameterized curves

$$\gamma(t) = (x(t), y(t))$$

where $t \in [0, m]$.

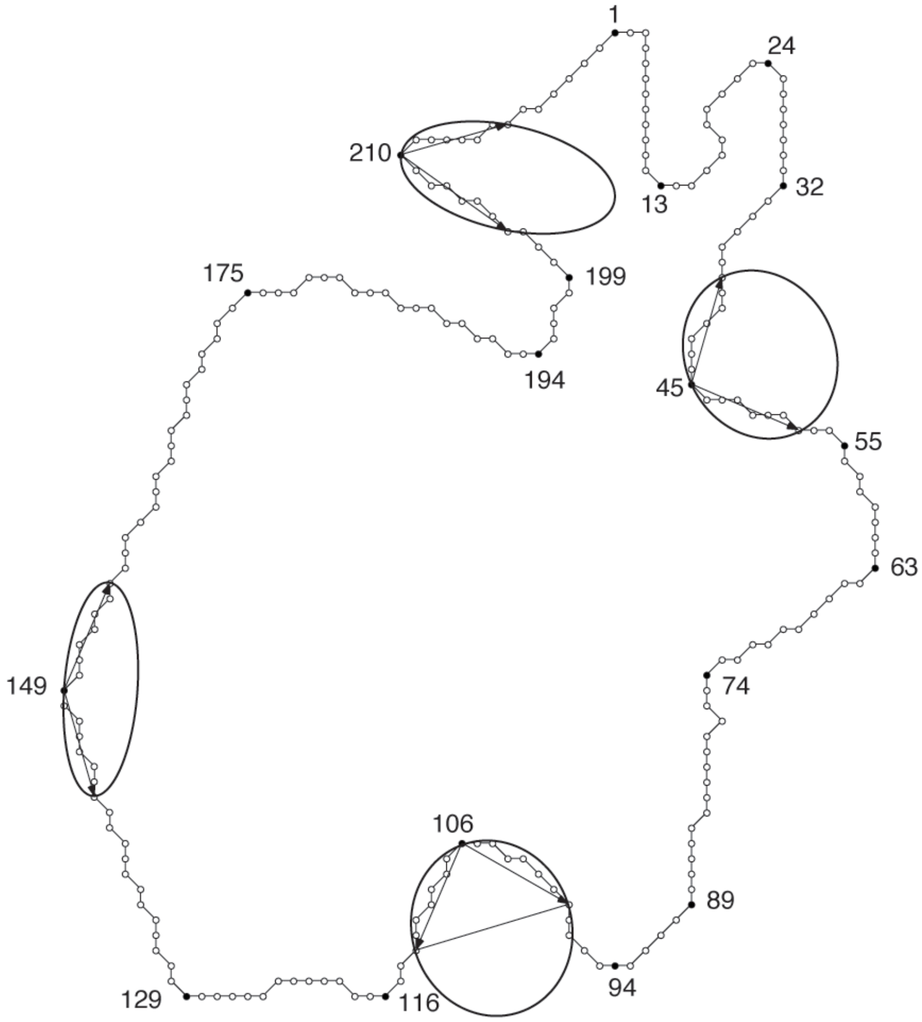
At point p_i, we assume that $\gamma(0) = \gamma(m) = p_i$ and $\gamma(j) = p_{i+j}$ for $1 \leq j \leq m - 1$. However, instead of calculating a parameterized curve passing through all m pixels we only select a few pixels "around p_i".

Functions $x(t)$ and $y(t)$ are locally interpolated at p_i by the following second-order polynomials,

$$x(t) = a_0 + a_1 t + a_2 t^2$$
$$y(t) = b_0 + b_1 t + b_2 t^2$$

and curvature is calculated by calculating derivatives of these two functions (see page 3 of Lecture 21).

For a second order polynomial it is sufficient to use three pixels for interpolation. We use $x(0) = x_i$, $x(1) = x_{i-k}$, and $x(2) = x_{i+k}$ with an integer parameter $k \geq 1$; this is analogous for $y(t)$.



The curvature at p_i is then defined by the following:

$$E_i = \frac{2(a_1 b_2 - b_1 a_2)}{(a_1^2 + b_1^2)^{1 \cdot 5}}$$

Note: The general formula is

$$\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) \ - \ \dot{y}(t)\ddot{x}(t)}{\left(\dot{x}(t)^2 + \dot{y}(t)^2\right)^{-1 \cdot 5}}$$

with

$$\ddot{x}(t) = \frac{\mathrm{d}^2 x(t)}{\mathrm{d}t^2} \quad \text{and} \quad \ddot{y}(t) = \frac{\mathrm{d}^2 y(t)}{\mathrm{d}t^2}$$

Algorithm CMT2001

[D. Coeurjolly and O. Teytaud, 2001]

This algorithm, which is in category **(C1.3)**, involves approximation of the radius of the osculating circle. At each point p_i, we calculate a maximum-length DSS centered at p_i. This DSS is used as an approximate segment of the tangent at p_i, and its length l_i corresponds to the radius r_i of the osculating circle by $r_i \approx l_i^2$.

The algorithm as published in 2001 calculates an *inner radius*

$$I_i = \lceil (l_i - 1/2)^2 - 1/4 \rceil$$

(note: ceiling function) and an *outer radius*

$$O_i = \lfloor (l_i + 1/2)^2 - 1/4 \rfloor$$

(note: floor function) and returns

$$E_i = 2/(I_i + O_i)$$

as an estimate of the curvature at p_i.

Coursework

Related material in textbook: first paragraph in Section 10.4 and Subsection 10.4.2.

A.22. [5 marks] Implement algorithm **M2003** and run it on the two 8-curves (or curves similar to these) shown on page 1, as well as on digital circles of diameter $30, \ldots, 100$. Discuss the influence of different values of k, for $4 \leq k \leq 10$ by

(i) comparing calculates curvature estimates with the true value for the case of the digital circles,

(ii) calculating local maxima of curvature estimates for detecting corners for the two curves on page 1, and

(iii) comparing curvature estimates at corresponding positions for the symmetric curve on page 1.

Optionally, (which may contribute **[1 mark]**) you may also use further curves for these discussions of curvature estimates.