Test and Measure of Circularity for Digital Curves

T. Roussillon[1], **I. Sivignon**[2], **and L. Tougne**[1]
[1]LIRIS, University of Lyon, 5 Av Pierre-Mendès France 69676 Bron, FRANCE
[2]LIRIS-CNRS, 8 Bd Niels Bohr 69622 Villeurbanne, FRANCE

Abstract - *We propose one geometric algorithm to solve both the problem of circularity test and the problem of circularity measure. The former problem consists in deciding whether a given digital curve is a digitization of a circle or not, whereas the latter problem consists in measuring how far a given digital curve is from a digitization of a circle. Our new circularity measure fulfills basic invariance properties and equals 0 for all digital circles and is positive for all shapes that are not digital circles. Experiments performed on synthetic and real images are very good.*

Keywords: Shape description, discrete geometry, computational geometry, circularity measure.

1 Introduction

In digital image analysis, a basic task consists in detecting or recognizing objects according to their shape. A lot of methods of shape representation and description are reviewed in [14]. Among all these methods, the ones based on global features, like area, eccentricity, convexity, etc. are efficient in numerous applications [11]. Discrete geometry provides effective algorithms to deal with digital curves and compute geometric estimators. For instance, a convergent perimeter estimator [2] has been derived from a linear time algorithm that recognizes digital straight segments [5].

This paper focuses on circularity test and circularity measure. Many algorithms have been proposed the past twenty years to recognize or detect digital circles or arcs (see for example [8, 6, 9, 4, 3]). However, to our knowledge, only a few papers introduce a circularity measure that equals 0 (or is extremal) for all digital circles ([8] for example).

After a study of the separating circle problem in section 2, an algorithm to perform the circularity test and compute a circularity measure is proposed in section 3. Experiments on synthetic and real images are described in section 4.

2 The separating circle problem

Let S and T be two finite sets of points in $\mathbb{Z}^2$. We want to know whether S is separable from T by a circle *C(o,r)* of center $o(x_o, y_o)$ and radius *r*. If it is, then the set of separating circles *{C}* is such that:

$$\begin{cases} \forall s \in S, \left(x_s - x_o\right)^2 + \left(y_s - y_o\right)^2 \leq r^2 \\ \forall t \in T, \left(x_t - x_o\right)^2 + \left(y_t - y_o\right)^2 > r^2 \end{cases}$$

Developing the previous inequalities, we get:

$$\begin{cases} \forall s \in S, -2\mathrm{ax}_s - 2\mathrm{ay}_s + c + f(x_s, y_s) \leq 0 \\ \forall t \in T, -2\mathrm{ax}_t - 2\mathrm{ay}_t + c + f(x_t, y_t) \leq 0 \end{cases}$$

$$\text{where} \quad \begin{array}{c} a = \mathrm{x}_o, b = \mathrm{y}_o, c = \left(x_o^2 + y_o^2 - r^2\right), \\ f\left(x, y\right) = x^2 + y^2 \end{array} \qquad (1)$$

2.1 Geometric interpretation of equation 1

Notice that equation 1 involves different interpretations of the triplet *(a, b, c)*, either as the coordinates of a 3D point in a *dual space* or as the coefficients of a 3D plane in a space that we call *extended primal space*. From now, in addition to the original plane (or (*xy*)-plane) containing the points of $\mathbb{Z}^2$ (figure 1.a), we work in a dual space (or (*abc*)-space) (figure 2.b) and in an extended primal space (or (*xyz*)-space) (figure 1.c). The extended primal space is built from the original plane such that all the 2D points are lifted, along an extra axis (the z-axis), according to a bivariate function *f*. Let S' = $\{s'(x_{s'}, y_{s'}, z_{s'}\}$ and T' = $\{t'(x_{t'}, y_{t'}, z_{t'}\}$ be the set of points that are raised in the extended primal space. In equation 1, $z = f(x,y) = x^2 + y^2$. The extended primal space and the dual space are dual, according to the classical definition of duality in computational geometry [12], that is a point in the former space maps to a plane in the latter space and conversely. Due to duality, the triplet *(x, y, z = f(x,y))*, like *(a, b, c)*, may be seen either as the coordinates of a 3D point in the extended primal space or as the coefficients of a 3D plane in the dual space.

The domain, defined as the complete set of solutions fulfilling the two sets of inequalities of equation 1 in the dual space, is a convex polyhedron. As a matter of fact, it is the intersection of |S| and |T| half-spaces (where the cardinality of a set is denoted by |.|). The coordinates of each point belonging to this polyhedron correspond to a separating circle of center $o(a,b)$ and radius $r = \sqrt{(a^2 + b^2 - c)}$. Hence, S and T are separable by a circle if and only if the domain is not empty. So, the separating circle problem may be solved by linear programming in the dual space. This reformulation of the problem has already been done in 4, but the Megiddo's algorithm [10], which was proposed to find one solution, is difficult to implement. We go further by working in the extended primal space, where the separating circle problem may be seen as a linear separability problem in 3D. This interpretation of equation 1 provides a simple algorithm for circularity test: S and T are circularly separable in the original plane if and only if S' and T' are separable by a plane in the extended primal space where $f(x, y) = x^2 + y^2$. Obviously, S' and T' may be reduced to their convex hull denoted by CH(S') and CH(T'). In addition, checking if CH(S') and CH(T') intersect each other or not is performed efficiently thanks to convexity (see [12] or section 3).

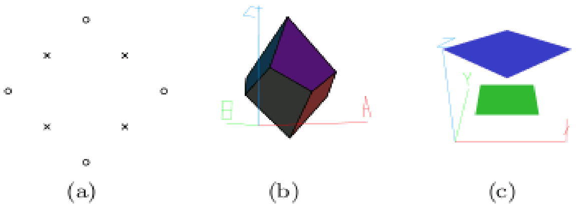


Figure 1: (a) S (crosses) and T (circles) in the original plane, (b) the complete set of solutions fulfilling the two sets of inequalities in the dual space, (c) CH(S') and CH(T') in the extended primal space are separable by a plane, because S and T are circularly separable.

2.2 On the domain structure

The domain is defined by the intersection of |S|+|T| half-spaces whose coefficients are computed from the points of S and T (equation 1). The dual of the |S| half-spaces is the upper convex hull of S', whereas the dual of the |T| half-spaces is the lower convex hull of T' (equation 1 again).

Thanks to classical results in computational geometry [12], we know that the projection on the (*xy*)-plane of the lower convex hull of T' is the closest-points Delaunay triangulation of T, that is the circumcircle of each triangle is empty, whereas the projection on the (*xy*)-plane of the upper convex hull of S' is the furthest-points Delaunay triangulation of S where the circumcircle of each triangle contains all points of S. In a dual way, the projection on the (*ab*)-plane of the intersection of the |S| half-spaces is the closest-points Voronoï diagram of S, whereas the projection on the (*ab*)-plane of the intersection of the |T| half-spaces is the furthest-points Voronoï diagram of T.

Moreover, we prove that the projection on the (*xy*)-plane of the intersection of the two groups of half-spaces is the *arc center domain* defined in [3]. We illustrate these properties in figure 2 from the example of figure 1.

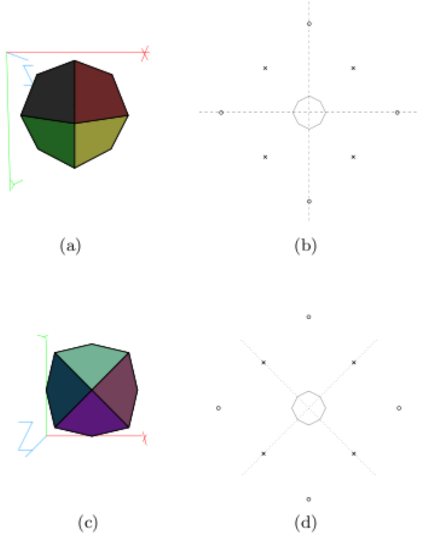


Figure 2: On the left, see the lower half (a) and the upper half (c) of the domain. Compare these halves to the Voronoï diagrams of S in dashed lines (b) and of T in dotted lines (d), with *acd*(S,T) in solid lines drawn in (b) and (d).

Indeed, from equation 1, we derive the projection on the (*xy*)-plane of the intersection of the two groups of half-spaces by replacing *c*:

$$\bigcap_{\forall s\in S, \forall t\in T} -2\mathrm{ax}_s - 2\mathrm{by}_s + (2\mathrm{ax}_t + 2\mathrm{by}_t - f(x_t, y_t)) + f(x_s, y_s) < 0 \qquad (2)$$

Developing 2, one may recognizes the expression of the bisector of segment *[st]*:

$$\bigcap_{\forall s\in S, \forall t\in T} 2(x_t - x_s)x_o + 2(y_t - y_s)y_o - x_t^2 - y_t^2 + x_s^2 + y_s^2 < 0$$

Then:

$$\bigcap_{\forall s\in S, \forall t\in T} H(s,t) \qquad (3)$$

where *H(s,t)* is the half-plane bounded by bisector of *s* and *t* and containing *s*.

Expression 3 is exactly the definition of *acd*(S,T) in [3], which concludes the proof.

3 Algorithm

3.1 How to compute S and T from a digital curve ?

Let Λ be a 4-connected subset of $\mathbb{Z}^2$. We say that Λ is a digital object. Λ is a digital circle too, if and only if it is the OBQ (Object Boundary Quantization) digitization of a circle. Let us recall that an interior point is a point of the object such that its four 4-neighbors belong to the object too. Let Int(Λ) be the set of interior points. Λ\Int(Λ) is a 8-connected digital curve corresponding to the object boundary. Obviously, S may be reduced to Λ\Int(Λ) and T may be reduced in a similar way.

Since a circle is convex, S may be reduced to the convex hull of Λ\Int(Λ). Moreover, T may be reduced to the points that are the closest points to the edges of the convex hull of Λ\Int(Λ) and their bisectors [3]. As a consequence, we use the process running in linear time described in [3] in order to compute S and T from a digital curve (figure 3).

It is stated in [3], using a result of [1], that *m* = |S| + |T| is bounded by O($n^{2/3}$), where *n* is the number of points of the digital curve, if the digital curve is convex. Otherwise, *m* is bounded by O(*n*).

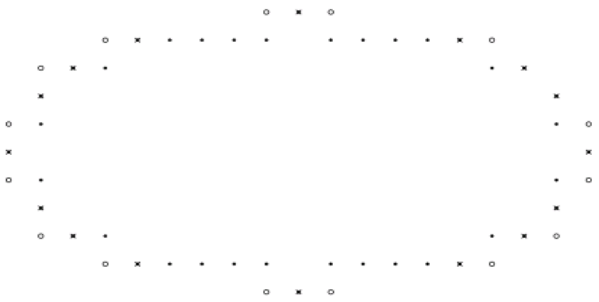


Figure 3: Given a digital curve (dots), S (crosses) and T (circles) are computed in linear time [3]

3.2 Circularity test

Checking whether S and T are separable by a circle is the core of the algorithm and is defined as follows:

Step1 : Compute CH(S') and CH(T'). We do not detail the algorithm which may run in O(m.log(m)). The computation of the convex hulls serves to facilitate the next step, because it reduces the number of points to consider and provides the property of convexity. Let us recall that it is equivalent to compute and lift along the z-axis the closest-points Delaunay triangulation of T and the furthest-points Delaunay triangulation of S (section 2.2).

Step2: Check whether CH(S') and CH(T') intersect each other or not. An elementary way to do this is to compute the minimum height between the two polyhedra CH(S') and CH(T') denoted by *h* = min Height(CH(S'),CH(T')). Height(CH(S'),CH(T')) is a function that returns the distance between the two polyhedra computed along the z-axis for each point of the domain of the function. Note that Height(CH(S'),CH(T')) is not defined anywhere. Indeed, the domain of this function is the intersection of the projection on the (*xy*)-plane of CH(S') and CH(T'), that is CH(S) ∩ CH(T). If (*h > 0*), CH(S') and CH(T') are disjoints, S and T are circularly separable, otherwise CH(S') and CH(T') intersect each other, they are not linearly separable, S and T are not circularly separable.

The brute force algorithm to find the height is to compute the graph G^* that is the intersection between the planar graph G_S and the planar graph G_T, where G_S (resp. G_T) is the projection on the (*xy*)-plane of the upper (resp. lower) convex hull of

S' (resp. T') (figure 4). If $|G^*| = 0$, then CH(S) ∩ CH(T) = ∅. In this degenerate case, it is noticeable that the digital curve from which S and T have been computed is a digital straight segment (figure 4.a and 4.c). If $|G^*| > 0$, it remains to compute the height function for each vertex of G^* and take the minimum. We may decide whether the digital curve from which S and T have been computed is a digital circle or not according to the sign of *h* as explained before.

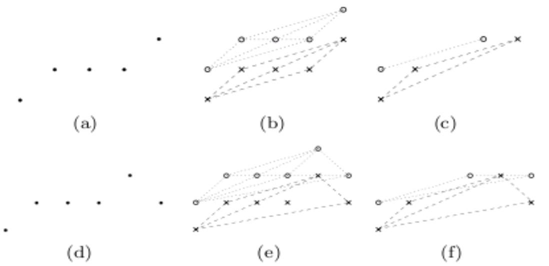


Figure 4: Given a digital curve drawn in (a) and (d), S (crosses) and T (circles) are computed in a naive way (b) and (e) and in a clever way (c) and (f) (subsection 3.1). G_S is the graph in dashed lines whereas G_T is the one in dotted lines. G^* is the intersection of the two graphs. It has 0 vertex in (b) and (c) because (a) is a digital straight segment. However, it has 4 vertices in (e) and (f) because (d) is not a digital straight segment.

The brute force algorithm runs in O(m^2), since G^* has at most m^2 vertices. However, it is possible to take advantage of the convexity of the height function to get an algorithm in O(*m*log*m*). By lack of space, we will not detail this algorithm, but the reader can refer to [12, pages 310-315] for example.

With this optimization, the whole algorithm is linear in time. Indeed, if the digital curve is not convex, it is not a digital circle. This decision may be taken during the preprocessing step described in [3] running in linear time. Thus, the core of the algorithm described before may only be applied on convex digital curves. Since m is bounded by O($n^{2/3}$) in that case, as stated in subsection 3.1, the core of the algorithm runs in O(*m*log*m*), that is O($n^{2/3}$log*n*), which is sublinear. Hence, the overall complexity of the algorithm is O(*n*).

4 Circularity measure

By working in the extended primal space, we are able to measure how far a given digital curve is from a digital circle. This is not possible by working in the dual space, since the domain is empty if the digital curve is not a digital circle. The algorithm proposed in section 3.2 does not need to be modified. The minimum height between the two polyhedra CH(S') and CH(T') is viewed as a circularity measure. When $h>0$, CH(S') and CH(T') are disjoints, the measure is set to 0: the digital curve is a digital circle. Otherwise, when $h<0$, CH(S') and CH(T') intersect each other, the measure is set to -*h*. Our assumption is that this measure fulfills basic invariance properties (subsection 4.1) and increases as the digital object gets away from a digital circle (subsection 4.2). Notice that the algorithm runs in O(*n*log*n*), since the algorithm is applied on convex or concave digital curves, contrary to the case of circularity test (section 3.2).

4.1 Invariance properties

We study the impact of rigid transformations onto the z-coordinate of points of S' and T'.

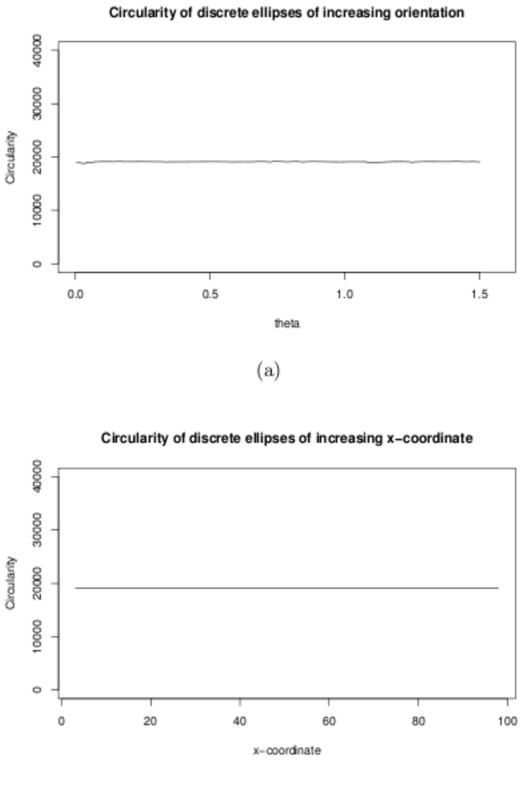
Rotation: Let us denote *û* the image of point *u* after a rotation of center the origin and angle θ, such that $\hat{u}(x_{\hat{u}}=x_u cos\theta - y_u sin\theta,\ y_{\hat{u}}=x_u sin\theta + y_u cos\theta)$. Then $f(x_{\hat{u}},y_{\hat{u}}) = f(x_u,y_u)$. As a consequence, the measure is rotation invariant, if errors due to digitization are neglected.

Scaling: Let us denote *û* the image of point *u* after a scaling by a factor α such that $\hat{u}(x_{\hat{u}}=\alpha x_u, y_{\hat{u}}=\alpha y_u)$. Then $f(x_{\hat{u}},y_{\hat{u}}) = \alpha^2 f(x_u,y_u)$. As a consequence, the measure needs to be normalized by the squared size of the object.

Translation: Let us denote *û* the image of point *u* under a translation by a translation vector $v(x_v,\ y_v)$ such that $\hat{u}(x_{\hat{u}}=x_u+x_v, y_{\hat{u}}=y_u+y_v)$. Then $f(x_{\hat{u}},y_{\hat{u}}) = f(x_u,y_u)-2x_u x_v-2y_u y_v-x_v^2-y_v^2$. The problem is rather complex, because the effect of the translation on the return value of *f* is not the same for all points, contrary to scaling. It is difficult to know how to normalize the measure. The simplest solution is to set the origin of the coordinates system to center of inertia of S.

We generated hundreds of digital ellipses (OBQ digitization) with various parameters : *a* (resp. *b*), small (resp. great) semi-axis, θ, the angle between the main axis of the ellipse and the x-coordinate axis, x_ω and y_ω the coordinates of the ellipse center. As stated above, the origin of the coordinates system is set to the center of inertia of S.

Figure 5.a shows that the circularity measure is quite translation invariant in that case. Figure 5.b shows that the circularity measure is also rotation invariant. As expected, errors due to digitization are very small.



(a)

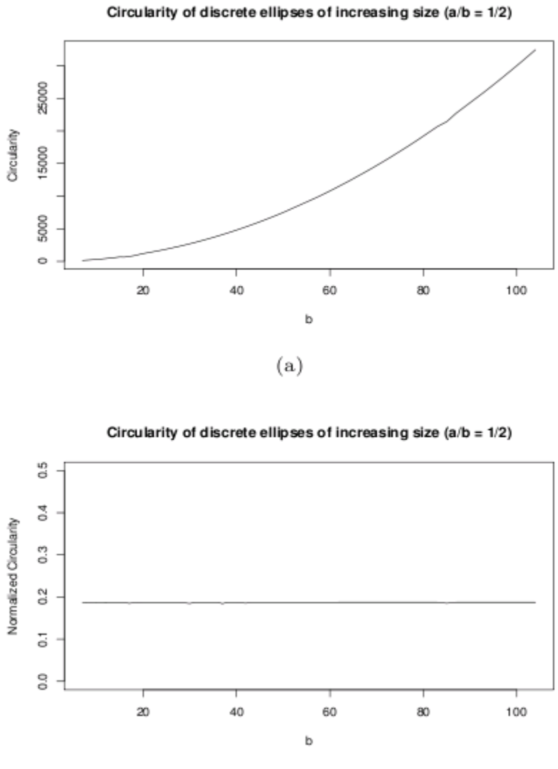
(b)

Figure 5: One hundred of digital ellipses were generated according to the following rules: $\omega(0,0)$, $a=80$, $b=160$, θ is ranging from 0 to $\pi/2$ in (a) ; $a=80$, $b=160$, θ and y_ω are set to 0 and w_ω is ranging from 0 to 100 in (b). Circularity is plotted against θ (a) and against x_ω (b).

In order to make the circularity measure scale invariant, it has been divided by d^2 where d is the diameter of the convex hull of the digital curve. The diameter is easily computed thanks to the rotating calipers algorithm [13]. Figure 6 shows the circularity measure before (a) and after (b) the normalization.

4.2 Descriptive Behavior

To verify that our circularity measure increases as the digital object gets away from a digital circle, it is assessed with digital ellipses of increasing elongation, regular polygons with increasing number of sides, noisy circles and digital images of fruits. In figure 7, the circularity logarithmically increases with the elongation of the ellipses.



(a)

(b)

Figure 6: One hundred of digital ellipses were generated according to the following rules: $\omega(0,0)$, $\theta = 0$, $a/b = 1/2$ and b is ranging from 5 to 105 in (a) and (b). Circularity is plotted against b, the size of the ellipses in both figures, but the measure in normalized (b).

We generated fifty regular polygons of fixed perimeter but increasing number of sides. In Figure 8, the circularity decreases with the number of sides and converges towards 0. The bigger the number of sides, the more the polygons look like a circle and the closer to zero the circularity is.

In order to study the impact of the amount of noise onto the circularity, we generated hundreds of noisy circles. We implemented a degradation model very close to the one in [7]. In Figure 9, the circularity increases with the amount of noise. The noisier the circle, the more it looks different from a circle and the higher the circularity is. Notice that the measure remains small, so it is rather robust.

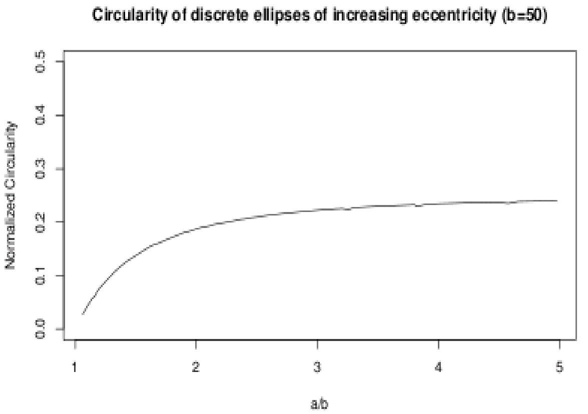


Figure 7: One hundred of digital ellipses were generated according to the following rules: $\omega(0,0)$, $\Theta= 0$, $b = 50$ and a is ranging from 10 to 50. Normalized circularity is plotted against a/b, the elongation of the ellipses.

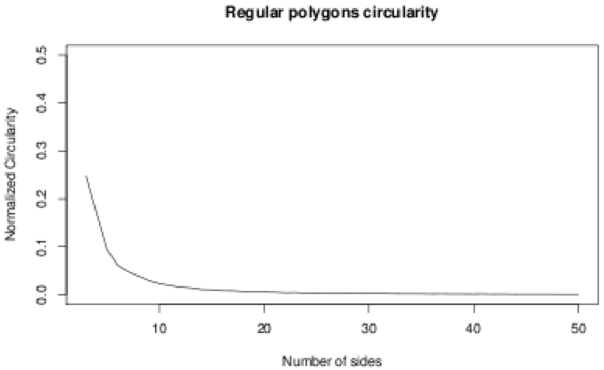


Figure 8: 50 regular polygons, the perimeter of which is approximatively equals to 1325 (the unit is pixel). Circularity is plotted against the number of sides.

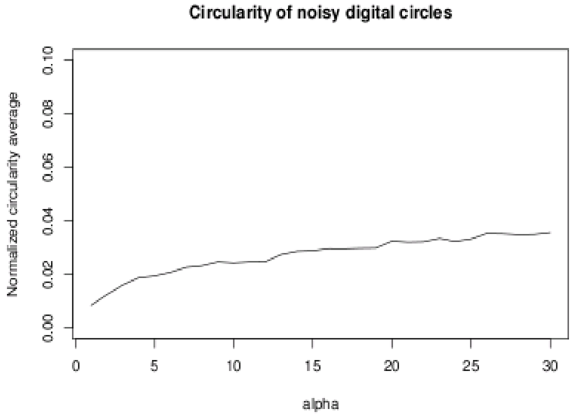


Figure 9: Nine sets of circles that are more and more noisy have been generated. Parameter alpha ranging from 1 to 30 controls the amount of noise. The average circularity measure is plotted against parameter alpha.

To end, the circularity measure has been applied to real digital images of fruits. Figure 10 shows the images and gives the corresponding circularity measures in the last row. It turns out that the measure reflects fairly well the visual dissimilarity between the shape of the fruits and the circular shape. The apple has a shape that is the closest to the one of a circle. The pear comes second. The banana and the pineapple have two similar measures rather high but for two different reasons: the banana is very elongated whereas the pineapple has many concavities.

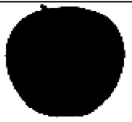
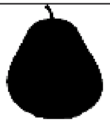
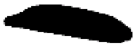
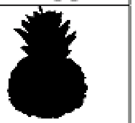
apple	pear	banana	pineapple
			
0.055	0.166	0.236	0.222

Figure 10: Four fruits with their circularity measure.

5 Conclusions

One geometric algorithm has been presented to solve both the problem of circularity test and the problem of circularity measure. It runs in O(n) in the former case (which is better than [8, 6, 9, 3]) and in O($n$log$n$) in the latter case (which is better than [8]). It is based on classical tools of computational geometry and as a consequence, it is rather easy to implement. The circularity measure is normalized in order to fulfill basic invariance properties: translation, rotation, scale invariance. Moreover, the measure may be computed on pieces of digital circle and equals 0 for all pieces of digital circle and is positive for all digital curves that are not pieces of digital circle, contrary to naive implementations with area and perimeter estimators. Experiments done on synthetic and real images show that the measure reflects fairly well the visual dissimilarity between a given digital curve and a digital circle.

For the future, we think about carrying out an experimental study about the circularity of pieces of digital curves (*i.e.* digital arcs), in order to see how the proposed measure deals with occlusions. Moreover, we would like make the algorithm on-line, without decreasing its complexity, for the robust recognition of digital arcs. Finally, we are interested in the shapes that can be studied in a similar way: it seems straightforward for parabolas, but more difficult for ellipses.

6 References

[1] D. Acketa and J. Zunić. On the maximal number of edges of convex digital polygons included into a m x m-grid. *Journal of Combinatorial Theory*, *Series* A, 69:358–368, 1995.

[2] D. Coeurjolly and R. Klette. A comparative evaluation of length estimators of digital curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:252–257, 2004.

[3] D. Coeurjolly, Y. Gérard, J.-P. Reveillès, and L. Tougne. An elementary algorithm for digital arc segmentation. *Discrete Applied Mathematics*, 139(1- 3):31–50, 2004.

[4] P. Damaschke. The linear time recognition of digital arcs. *Pattern Recognition Letters*, 16:543–548, 1995.

[5] I. Debled-Renesson and J.-P. Reveillès. A linear algorithm for segmentation of digital curves. *International Journal of Pattern Recognition and Artificial Intelligence*, 9:635–662, 1995.

[6] S. Fisk. Separating points sets by circles, and the recognition of digital disks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:554– 556, 1986.

[7] T. Kanungo, R. M. Haralick, H. S. Baird, W. Stuezle, and D. Madigan. A statistical, nonparametric methodology for document degradation model validation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1209–1223, 2000.

[8] C. E. Kim and T. A. Anderson. Digital disks and a digital compactness measure. *In Annual ACM Symposium on Theory of Computing*, pages 117–124, 1984.

[9] V. A. Kovalevsky. New definition and fast recognition of digital straight segments and arcs. *In International Conference on Pattern Analysis and Machine Intelligence*, pages 31–34, 1990.

[10] N. Megiddo. Linear programming in linear time when the dimension is fixed. *SIAM Journal on Computing*, 31:114–127, 1984.

[11] M. Peura and J. Iivarinen. Efficiency of simple shape descriptors. In *3rd International Workshop on Visual Form*, 1997.

[12] F. P. Preparata and M. I. Shamos. *Computational geometry* : an introduction. Springer, 1985.

[13] G. Toussaint. Solving geometric problems with the rotating calipers. In Proceedings MELECON 83, *Mediterranean Electrotechnical Conference*, 1983.

[14] D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19, 2004.