Euclidean Eccentricity Transform by Discrete Arc Paving

Adrian Ion[1], Walter G. Kropatsch[1], and Eric Andres[2,⋆]

[1] Pattern Recognition and Image Processing Group,
Faculty of Informatics, Vienna University of Technology, Austria
{ion,krw}@prip.tuwien.ac.at
[2] University of Poitiers
SIC, FRE CNRS 2731, France
andres@sic.sp2mi.univ-poitiers.fr

Abstract. The eccentricity transform associates to each point of a shape the geodesic distance to the point farthest away from it. The transform is defined in any dimension, for simply and non simply connected sets. It is robust to Salt & Pepper noise and is quasi-invariant to articulated motion. Discrete analytical concentric circles with constant thickness and increasing radius pave the 2D plane. An ordering between pixels belonging to circles with different radius is created that enables the tracking of a wavefront moving away from the circle center. This is used to efficiently compute the single source shape bounded distance transform which in turn is used to compute the eccentricity transform. Experimental results for three algorithms are given: a novel one, an existing one, and a refined version of the existing one. They show a good speed/error compromise.

Keywords: eccentricity transform, discrete analytical circles.

1 Introduction

A major task in image analysis is to extract high abstraction level information from an image that usually contains too much information and not necessarily the correct one (e.g. because of noise or occlusion). Image transforms have been widely used to move from low abstraction level input data to a higher abstraction level that forms the output data (skeleton, connected components, etc.). The purpose is to have a reduced amount of (significant) information at the higher abstraction levels. One class of such transforms that is applied to 2D shapes, associates to each point of the shape a value that characterizes in some way it's relation to the rest of the shape, e.g. the distance to some other point of the shape. Examples of such transforms include the well known distance transform [1], which associates to each point of the shape the length of the **shortest** path to the border, the Poisson equation [2], which can be used to associate to each point the **average** time to reach the border by a random path (average

⋆ Partially supported by the Austrian Science Fund under grants S9103-N13, P18716-N13. The many fruitful discussions with Samuel Peltier are gratefully acknowledged.

D. Coeurjolly et al. (Eds.): DGCI 2008, LNCS 4992, pp. 213–224, 2008.

length of the random paths from the point to the boundary), and the eccentricity transform [3] which associates to each point the length of the **longest** of the shortest paths to any other point of the shape. Using the transformed images one tries to come up with an abstracted representation, like the skeleton [4] or shock graph [5] build on the distance transform, which could be used in e.g. shape classification or retrieval. Minimal path computation [6] as well as the distance transform [7] are commonly used in 2D and 3D image segmentation.

This paper is focusing on the eccentricity transform (ECC) which has its origins in the graph based eccentricity [8,9]. It has been defined in the context of digital images in [3,10]. It was applied in the context of shape matching in [11]. It has been shown that the transform is very robust with respect to noise. The eccentricity transform can be defined for discrete objects of any dimension, closed (e.g. typical 2D binary image) or open sets (surface of an ellipsoid), and for continuous objects of any dimension (e.g. 3D ellipsoid or the 2D surface of the 3D ellipsoid, etc.). For the case of discrete shapes, a naive algorithm and a more efficient one for 2D shapes without holes, have been presented in [3], with experimental results only for the 4 and 8 neighbourhoods. For simply connected shapes on the hexagonal and dodecagonal grid, an efficient algorithm was given in [12]. Regarding continuous shapes, a detailed study has been made for the case of an ellipse, and some preliminary properties regarding rectangles, and a class of elongated convex shapes, have been given [13]. An algorithm for finding the eccentric/furthest points for the vertices of a simple polygon was given in [14].

This paper presents an algorithm for efficiently computing the single source shape bounded distance transform using discrete circles. The idea is to use discrete circles to propagate the distance in a shape following an idea already proposed for discrete wave propagation simulation [15]. In addition, a novel algorithm and a refined one are given to compute the eccentricity transform. It has been shown that, so called, eccentric points play a major role in the transform. These points are shape border points. Different ideas are proposed in order to attempt to identify these eccentric points. Distance transforms, originating at these candidate eccentric points, are computed and accumulated to obtain the eccentricity transform. A comparison between these three methods is provided.

Section 2 gives a short recall of the eccentricity transform, the main properties relevant for this paper and gives the important facts about discrete circles. Sections 3 and 4 present the proposed algorithms, followed by experimental results. Section 5 concludes the paper and gives an outlook of the future work.

2 Recall

In this section basic definitions and properties of the eccentricity transform and discrete circles are given.

2.1 Recall ECC

The following definitions and properties follow [3,11].

Let the shape $\mathcal{S}$ be a closed set in $\mathbb{R}^2$ and $\partial\mathcal{S}$ be its border[1]. A (geodesic) path π is the continuous mapping from the interval $[0,1]$ to $\mathcal{S}$. Let $\mathit{\Pi}(\mathbf{p_1},\mathbf{p_2})$ be the set of all paths between two points $\mathbf{p_1},\mathbf{p_2} \in \mathcal{S}$ within the set $\mathcal{S}$. The geodesic distance $d(\mathbf{p_1},\mathbf{p_2})$ between two points $\mathbf{p_1},\mathbf{p_2} \in \mathcal{S}$ is defined as the length $\lambda(\pi)$ of the shortest path $\pi \in \mathit{\Pi}(\mathbf{p_1},\mathbf{p_2})$ between $\mathbf{p_1}$ and $\mathbf{p_2}$

$$d(\mathbf{p_1},\mathbf{p_2}) = \min\{\lambda(\pi(\mathbf{p_1},\mathbf{p_2}))|\pi \in \mathit{\Pi}\} \text{ where } \lambda(\pi(t)) = \int_0^1 |\dot{\pi}(t)|dt \qquad (1)$$

where $\pi(t)$ is a parametrization of the path from $\mathbf{p_1} = \pi(0)$ to $\mathbf{p_2} = \pi(1)$.

The eccentricity transform of $\mathcal{S}$ can be defined as, $\forall \mathbf{p} \in \mathcal{S}$

$$ECC(\mathcal{S},\mathbf{p}) = \max\{d(\mathbf{p},\mathbf{q})|\mathbf{q} \in \mathcal{S}\} \qquad (2)$$

i.e. to each point $\mathbf{p}$ it assigns the length of the shortest geodesics to the points farthers away from it. In [3] it is shown that this transformation is quasi-invariant to articulated motion and robust against salt and pepper noise (which creates holes in the shape).

This paper considers defined by points on a square grid $\mathbb{Z}^2$. Paths need to be contained in the area of $\mathbb{R}^2$ defined by the union of the support squares for the pixels of $\mathcal{S}$. The distance between any two pixels whose connecting segment is contained in $\mathcal{S}$ is computed using the ℓ_2-*norm*.

2.2 Properties of Eccentric Points

In general, an *extremal point* is a point where a function reaches an extremum (local or global). In the case of the geodesic distance d in a shape $\mathcal{S}$ we call a point $\mathbf{x} \in \mathcal{S}$ *maximal* iff $d(\mathbf{x},\mathbf{p})$ is a local maximum for a given point $\mathbf{p} \in \mathcal{S}$. $X(\mathcal{S})$ denotes the set of all maximal points of shape $\mathcal{S}$.

An *eccentric point* of a shape $\mathcal{S}$ is a point $\mathbf{e} \in \mathcal{S}$ that is farthest away in $\mathcal{S}$ from at least one other point $\mathbf{p} \in \mathcal{S}$ i.e. $\exists \mathbf{p} \in \mathcal{S}$ s.t. $ECC(\mathcal{S},\mathbf{p}) = d(\mathbf{p},\mathbf{e})$. For a shape $\mathcal{S}$, $E(\mathcal{S}) = \{\mathbf{e} \in \mathcal{S}|\mathbf{e}$ is eccentric for at least one point of $\mathcal{S}\}$ denotes the set of all its eccentric points. The set of eccentric points $E(\mathcal{S})$ is a subset of the set of all maximal points $X(\mathcal{S})$ i.e. $E(\mathcal{S}) \subseteq X(\mathcal{S})$ (eccentric points are global maxima for d, while maximal points are local maxima for a given point $\mathbf{p}$).

Knowing $E(\mathcal{S})$ can speedup the computation of the $ECC(\mathcal{S})$. A naive algorithm for $ECC(\mathcal{S})$ computes all geodesic distances $d(\mathbf{p},\mathbf{q})$ for all pairs of points in $\mathcal{S}$ and takes the maximum at each point $\mathbf{p} \in \mathcal{S}$. Since the geodesic distance is commutative, e.g. $d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p})$, it is sufficient to restrict $\mathbf{q}$ in (2) to the eccentric points $E(\mathcal{S})$. Instead of computing the length of the geodesics from $\mathbf{p} \in \mathcal{S}$ to all the other points $\mathbf{q} \in \mathcal{S}$ and taking the maximum, one can compute the length of the geodesics from all $\mathbf{q} \in E(\mathcal{S})$ to all the points $\mathbf{p} \in \mathcal{S}$. This reduces the number of steps from $|\mathcal{S}|$ to $|E(\mathcal{S})|$. The key question is therefore how to estimate $E(\mathcal{S})$ without knowing $ECC(\mathcal{S})$.

The following properties of eccentric points are relevant for this paper and concern bounded 2D shapes.

[1] This definition can be generalized to any dimension, continuous and discrete objects.

Property 1. All eccentric points $E(\mathcal{S})$ of a shape $\mathcal{S}$ lie on the border of $\mathcal{S}$ i.e. $E(\mathcal{S}) \subseteq \partial\mathcal{S}$. (*Proof* due to [3]).

Property 2. Being an eccentric point is not a local property i.e. $\forall B \subset \partial\mathcal{S}$ a boundary part (a 2D open and simple curve), and a point $\mathbf{b} \in B$, we can construct the rest of the boundary $\partial\mathcal{S} \setminus B$ s.t. $\mathbf{b}$ is not an eccentric point of $\mathcal{S}$. (*Proof* due to [13]).

Property 3. Not all eccentric points $\mathbf{e} \in E(\mathcal{S})$ are local maxima in the eccentricity transform (the ellipse in [13] is an example).

E.g. some eccentric points $\mathbf{e} \in E(\mathcal{S})$ may find larger eccentricity values $ECC(\mathcal{S}, \mathbf{e}) < ECC(\mathcal{S}, \mathbf{q})$ in their neighborhood $|\mathbf{e} - \mathbf{q}| < \epsilon$ for $\epsilon > 0$. They typically form connected clusters containing also a local maximum in $ECC(\mathcal{S})$.

Property 4. Not all eccentric points are local maxima in the distance transform from any point of the shape i.e. there is no guarantee that all eccentric points of a shape $\mathcal{S}$ will be local maxima in the distance transform $DT(\mathcal{S}, \mathbf{p})$ for any random point $\mathbf{p} \in \mathcal{S}$ (see [13] for an example).

2.3 Recall Discrete Circles

The shape bounded distance transform algorithm presented in this paper propagates the distance computation in the shape. This idea has already been used to perform discrete wave propagation simulation [15]. The propagation is performed with discrete concentric circles where each circle corresponds to a given distance range. The propagation provides a cheap distance computation. The discrete circle definition has to verify specific properties as the center coordinates and the radius aren't necessarily integers. The most important property is that circles centered on a point must fill, preferably pave, the discrete space. We can't miss points during our propagation phase. Of course, the arc/circle generation algorithm has to be linear in the number of pixels generated or we would loose the whole point of proposing a new method with a reasonable complexity. There exists several different discrete circle definitions. The best known circle is an algorithmic definition proposed by Bresenham but this doesn't correspond to the requirements of our problem. The circle doesn't pave the space, the center coordinates and radius are only integer. The definition that fits our problem is the discrete analytical circle proposed by Andres [16]:

Definition 1. *(Discrete Analytical Circle) A discrete analytical circle* $C(R, \mathbf{o})$ *is defined by the double inequality:*

$$\mathbf{p} \in C(R, \mathbf{o}) \;\; \textit{iff} \;\; \left(R - \frac{1}{2}\right) \leq \|\mathbf{p} - \mathbf{o}\| < \left(R + \frac{1}{2}\right)$$

with $\mathbf{p} \in \mathbb{Z}^2, \mathbf{o} \in \mathbb{R}^2$, *and* $R \in \mathbb{R}$ *the radius using euclidean distance.*

This circle is a circle of arithmetical thickness 1. The circle definition is similar to the discrete analytical line definition proposed by Reveillès [17].

The fact that this circle definition is analytical, and not just algorithmic like Bresenham's circle, has many advantages. Point localisation, i.e. knowing if a point is inside, outside or on the circle, is trivial. The center coordinates and the radius don't have to be integers. The circle definition can easily be extended to any dimension (see [18]). Circles with the same center pave the space:

$$\biguplus_{R=0}^{\infty} C\left(R, \mathbf{o}\right) = \mathbb{Z}^2$$

Each integer coordinate point in space belongs to one and only one circle. When you draw Bresenham circles with the same center and increasing radii there are discrete points that don't belong to any of the concentric circles. What is less obvious is that this circle has good topological properties. In fact, as shown in [18] for the general case in dimension n for discrete analytical hyperspheres, a discrete circle of arithmetical thickness equal to 1 (this is the case with the given definition) is at least 8-connected. The discrete points of the circle can be ordered and there exists a linear complexity generation algorithm [16,18].

3 Shape Bounded Single Source Distance Transform

Given a discrete connected shape $\mathcal{S}$ and an initial point $\mathbf{o} \in \mathcal{S}$. The shape bounded single source distance transform DT assigns to every point $\mathbf{q} \in \mathcal{S}$ its geodesic distance to $\mathbf{o}$:

$$DT(\mathcal{S}, \mathbf{o}) = \{d(\mathbf{o}, \mathbf{q})\}, \quad \mathbf{q} \in \mathcal{S} \tag{3}$$

Another formulation would consider the time needed for a wavefront initiated at $\mathbf{o}$ traveling in the homogeneous medium $\mathcal{S}$ to reach each point $\mathbf{q}$ of $\mathcal{S}$. Following the previous formulation we propose to model the wavefront using discrete arcs and record the time when each point $\mathbf{q}$ is reached for the first time. The wavefront travels with speed 1 i.e. 1 distance unit corresponds to 1 time unit.

The wavefront propagating from a point $\mathbf{o}$ will have the form of a circle centered at $\mathbf{o}$. If the wavefront is blocked by obstacles, the circle is "interrupted" and disjoint arcs of the same circle continue propagating in the unblocked directions. For a start point $\mathbf{o}$, the wavefront at time t is the set of points $\mathbf{q} \in \mathcal{S}$ at distance $t - 0.5 \leqslant d(\mathbf{o}, \mathbf{q}) < t + 0.5$. The wavefront at any time t consists of a set of arcs $W(t) \subset \mathcal{S}$. Each arc $A \in W(t)$ lies on a circle centered at the point where the path from $\mathbf{q} \in A$ to $\mathbf{p}$ first touches $\partial\mathcal{S}$.

The computation starts with a circle of radius 1 centered at the source point $\mathbf{o}$, $W(1) = \{C(1, \mathbf{o})\}$. It propagates and clusters as presented above, with the addition that pixels with distance smaller than the current wavefront also block the propagation. An arc $A \in W(t)$ of the wavefront $W(t)$, not touching $\partial\mathcal{S}$ at time t, but touching at $t-1$, diffracts new circles centered at the endpoints $\mathbf{e} \in A$. The added arcs start with radius one and handicap $d(\mathbf{p}, \mathbf{e})$. The handicap of an

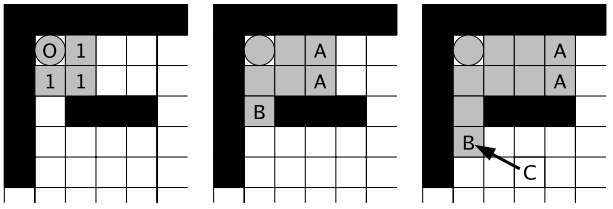


Fig. 1. The three steps during wavefront propagation (shape white, background black). Left, radius 1: circle with radius 1 and center O is bounded to an arc. Middle, radius 2: the front is splitted in two arcs A, B. Right, radius 3: arc B, touching the hole at radius 2 but not at radius 3, creates arc C with the center at the current point of B.

arc accumulates the length of the shortest path to the center of the arc such that the distance of the wavefront from the initial point **o** is always the sum of the handicap and the radius of the arc. No special computation is required for the initial angles of arcs with centers on the boundary pixels of $\mathcal{S}$. They will be corrected by the clustering and bounding command when drawing with radius 1. See Fig. 1 and Alg. 1.

The complexity of Alg. 1 is determined by the number of pixels in $\mathcal{S}$, denoted $|\mathcal{S}|$, and the number of arcs of the wave. Arcs are drawn in $O(n)$ where n is the number of pixels of the arc. Pixels in the shadow of a hole, where different parts of the wavefront meet ('shocks'), are drawn by each part. All other pixels are drawn only once. Adding and extracting arcs to/from the wavefront (W in Alg. 1) can be done in $\log(\text{size}(W))$.

For convex shapes, the size of W is 1 all the time, so the algorithm executes in $O(|\mathcal{S}|)$. For simply connected shapes, each pixel is drawn only once. Assuming an exaggerated upper bound $\text{size}(W) = |\mathcal{S}|$ and each arc only draws one pixel, the complexity for simply connected shapes is below $O(|\mathcal{S}| \log |\mathcal{S}|)$. Each hole creates an additional direction to reach a point, e.g. no hole: 1 direction; 1 hole: 2 directions - one on each side of the hole; 2 holes: maximum 3 directions - one side, between holes, other side, etc. Note that we don't count the number of possible paths, but the number of directions from which connected wavefronts can reach a point at the shortest geodesic distance. For a non-simply connected shape with k holes, a pixel is set a maximum of k times (worst case). Thus, the complexity for non-simply connected shapes with k holes is below $O(k|\mathcal{S}| \log |\mathcal{S}|)$.

4 Progressive Refinement Eccentricity Transform

In [3] an algorithm for approximating the ECC of discrete shapes was presented (will be denoted by ECC06) (see Alg. 2). The algorithm is faster than the naive one (see [3]), and computes the exact values for a class of simply connected shapes. For all other shapes it gives an approximation. Based on Properties 3 and 4 we refine algorithm ECC06 by adding a third phase to ECC06. This step finds the limits of clusters of eccentric points for which at least one eccentric point

Algorithm 1. $DT(\mathcal{S}, \mathbf{p})$ - Compute distance transform using discrete circles.

Input: Discrete shape $\mathcal{S}$ and pixel $\mathbf{p} \in \mathcal{S}$.

1: **for all** $\mathbf{q} \in \mathcal{S}$ **do** $D(\mathbf{q}) \leftarrow \infty$ /**initialize distance matrix**/
2: $D(\mathbf{p}) \leftarrow 0$
3: $W \leftarrow \mathrm{Arc}(\mathbf{p}, 1, [0; 2\pi], 0, \emptyset)$ /**Arc(**c**enter, **r**adius, **a**ngles, **h**andicap, **p**arent)**/
4:
5: **while** $W \neq \emptyset$ **do**
6: $A \nleftarrow \arg\min\{A.r + A.h | A \in W\}$ /**select and remove arc with smallest radius+handicap**/
7:
8: /**draw arc points with lower distance than known before, use real distances**/
9: $D(\mathbf{m}) \leftarrow \min\{D(\mathbf{m}), A.h + d(A.\mathbf{c}, \mathbf{m}) | \mathbf{m} \in A \cap \mathcal{S}\}$
10:
11: $P_1, \ldots, P_k \leftarrow$actually drawn (sub)arcs/parts of A /**split and bound**/
12: $W \leftarrow W + \mathrm{Arc}(A.\mathbf{c}, A.r + 1, P_i.\mathbf{a}, A.h, A), \forall i = 1..k$ /**propagate**/
13:
14: /**diffract if necessary**/
15: **if** $A.\mathbf{p}$ touches $\partial\mathcal{S}$ on either side **then**
16: $\mathbf{e} \leftarrow$ last point of A, on side where $A.\mathbf{p}$ was touching $\partial\mathcal{S}$
17: $W \leftarrow W + \mathrm{Arc}(\mathbf{e}, 1, [0; 2\pi], D(\mathbf{e}), A)$
18: **end if**
19: **end while**

Output: Distances D.

has been found (this is phase 2, lines 19-28 of Alg. 3). The refined algorithm is denoted by ECC06'.

Algorithms ECC06 and ECC06' try to identify the ECC centers (smallest ECC value). Computing the $DT(\mathcal{S}, \mathbf{c})$ for a center point $\mathbf{c}$ is expected to create local maxima where eccentric points lie. For non-simply connected shapes the center can become very complex, it can contain many points and it can be disconnected. This makes identifying all center points harder, as not all eccentric points are farthest away from all center points. Missing center points can lead to missing eccentric points which leads to an approximation errors.

4.1 New Algorithm (ECC08)

Algorithm ECC08 first attempts to identify at least one point from each cluster of eccentric points. Like in ECC06', scanning along $\partial\mathcal{S}$ is then used to find the limits of each cluster.

Inspecting a shape $\mathcal{S}$ is done by repeatedly computing $DT(\mathcal{S}, \mathbf{q})$ for the highest local maximum in the current approximation of $ECC(\mathcal{S})$ and accumulating the results (using max) to obtain the next approximation. After some steps, this process can enter an infinite loop if all reachable points have been visited already. Such a configuration is called an oscillating configuration and the visited points are called oscillating points [19]. If $DT(\mathcal{S}, \mathbf{q})$ with $\mathbf{q} \in \mathcal{S}$ is considered as an approximation for $ECC(\mathcal{S})$, the error is expected to be higher around $\mathbf{q}$

Algorithm 2. $ECC06(\mathcal{S})$ - Eccentricity transform by progressive refinement.

Input: Discrete shape $\mathcal{S}$.

for all $\mathbf{q} \in \mathcal{S}, ECC(\mathbf{q}) \leftarrow 0$ /*initialize distance matrix*/
$\mathbf{p} \leftarrow$ random point of $\mathcal{S}$ /*find a starting point*/

/*Phase 1: find a diameter*/
while $\mathbf{p}$ not computed **do**
  $ECC \leftarrow \max\{ECC, DT(\mathcal{S}, \mathbf{p})\}$ /*accumulate & mark $\mathbf{p}$ as computed*/
  $\mathbf{p} \leftarrow \arg\max\{ECC(\mathbf{p}) | \mathbf{p} \in \mathcal{S}\}$ /*highest current ECC (farthest away)*/
end while

/*Phase 2: find center points and local maxima*/
$pECC \leftarrow 0$ /*make sure we enter the loop*/
while $pECC \neq ECC$ **do**
  $pECC \leftarrow ECC$
  $C \leftarrow \arg\min\{ECC(\mathbf{p}) | \mathbf{p} \in \mathcal{S}\}$ /*find all points with minimum ECC*/
  for all $\mathbf{c} \in C$, $\mathbf{c}$ not computed **do**
    $D \leftarrow DT(\mathcal{S}, \mathbf{c})$ /*do a distance transform from the center*/
    $ECC \leftarrow \max\{ECC, D\}$ /*accumulate & mark $\mathbf{c}$ as computed*/

    $M \leftarrow \{\mathbf{q} \in \mathcal{S} | D(\mathbf{q})$ local maximum in $\mathcal{S}$ & $\mathbf{q}$ not computed$\}$
    for all $\mathbf{m} \in M$, $\mathbf{m}$ not computed **do**
      $ECC \leftarrow \max\{ECC, DT(\mathcal{S}, \mathbf{m})\}$ /*accumulate & mark $\mathbf{m}$ as computed*/
    end for
  end for
end while

Output: Distances ECC.

and smaller around the points farther away from $\mathbf{q}$ i.e. the points with highest values in $DT(\mathcal{S}, \mathbf{q})$. Whenever an oscillating configuration is reached all points of $\partial\mathcal{S}$ which are local minima in the current ECC approximation are selected for distance computation. If the last operation does not produce any unvisited points as ECC local maxima, the search is terminated (Alg. 3).

All three algorithms try to find $E(\mathcal{S})$ and compute $DT(\mathcal{S}, \mathbf{e})$ for all $\mathbf{e}$ in the current approximation of $E(\mathcal{S})$. As $E(\mathcal{S})$ is actually known only after computing $ECC(\mathcal{S})$, all algorithms incrementally refine an initial approximation of $ECC(\mathcal{S})$ by computing $DT(\mathcal{S}, \mathbf{q})$ for candidate eccentric points $\mathbf{q}$ that are identified during the progress of the approximation. For ECC06' and ECC08, one eccentric point per cluster is sufficient to produce the correct result.

4.2 Experimental Results

We have compared the three algorithms, ECC06, ECC06', and ECC08, on 70 shapes from the MPEG7 CE-Shape1 database [20], 6 from [21], and one additional new shape (see Table 5).

Algorithm 3. $ECC08(\mathcal{S})$ - Eccentricity transform by progressive refinement.

Input: Discrete shape $\mathcal{S}$.

for all $\mathbf{q} \in \mathcal{S}, ECC(\mathbf{q}) \leftarrow 0$ /*initialize distance matrix*/
$ToDo \leftarrow$ random point of $\mathcal{S}$ /*find a starting point*/

/*Phase 1: inspect shape*/
while $ToDo \neq \emptyset$ **do**
  $\mathbf{p} \nleftarrow \arg\max\{ECC(\mathbf{p})|\mathbf{p} \in ToDo\}$ /*remove point with highest current ECC*/
  $ECC \leftarrow \max\{ECC, DT(\mathcal{S}, \mathbf{p})\}$ /*accumulate & mark $\mathbf{p}$ as computed*/

  /*add not computed local maxima to $ToDo$*/
  $ToDo \leftarrow ToDo \cup \{\mathbf{q} \in \mathcal{S}|ECC(\mathbf{q})$ local maximum in $\mathcal{S}$ & $\mathbf{q}$ not computed$\}$

  /*test if an oscillating configuration was found*/
  if $ToDo = \emptyset$ **then**
    $ToDo \leftarrow ToDo \cup \{\mathbf{q} \in \partial\mathcal{S}|ECC(\mathbf{q})$ local minimum in $\partial\mathcal{S}$ & $\mathbf{q}$ not computed$\}$
  end if
end while

/*Phase 2: find limits of clusters of eccentric points */
$ToDo \leftarrow$ all neighbours in $\partial\mathcal{S}$ of all eccentric points in ECC
while $ToDo \neq \emptyset$ **do**
  $\mathbf{p} \nleftarrow \arg\max\{ECC(\mathbf{p})|\mathbf{p} \in ToDo\}$ /*remove point with highest current ECC*/

  $ECC \leftarrow \max\{ECC, DT(\mathcal{S}, \mathbf{p})\}$ /*accumulate & mark $\mathbf{p}$ as computed*/

  /*do we need to continue in this direction?*/
  if ECC changed previously i.e. $\mathbf{p}$ is an eccentric point **then**
    $ToDo \leftarrow ToDo \cup \{\mathbf{q} \in \partial\mathcal{S}|\mathbf{q}$ is a neighbour of $\mathbf{p}$ in $\partial\mathcal{S}$ & $\mathbf{q}$ not computed$\}$
  end if
end while

Output: Distances ECC.

The MPEG7 database contains 1400 shapes from 70 object classes. One shape from each class was taken (the first one) and reduced to about 36,000 pixels (aspect ratio and connectivity preserved). Table 1 summarizes the main characteristics of the 70 shapes, their sizes and the range of smallest and largest eccentricity values. The smallest eccentricity appears at the center of the shape and the largest eccentricity corresponds to its diameter.

Correct ECC values are computed by the naive algorithm RECC as the maximum of the distance transforms of all boundary points $\partial\mathcal{S}$ (according to Property 1). Table 2 compares the performance of the 3 algorithms:

max.pixel error: maximum difference between RECC and ECC per pixel / max.eccentricity for this shape;

max.error size: maximum number of pixels that differ between RECC and ECC / size of this shape;

Table 1. Characteristics of shapes from the MPEG7 database

measure	ranges from	to
sizes in pixel	683	28821
smallest eccentricity in pixel (ECC_{min})	28	235
maximum eccentricity in pixel (ECC_{max})	55	469

Table 2. Results of 70 images from the MPEG7 database

measure	**ECC06**	**ECC06'**	**ECC08**
max.pixel error	4.45 / 221.4	4.27 / 221.4	6.57 / 266.6
max.error size	4923 / 19701	2790 / 19701	2359 / 19701
#DT(ECC) / #DT(RECC)	8%	10%	15%
100% correct	44 / 70	60 / 70	56 / 70

Table 3. 'Worst' results from the MPEG7 database

shape characteristics					max.ECC.diff.			size of ECC.diff.		
nb.	name	ECC_{min}	ECC_{max}	size	ECC06	ECC06'	ECC08	ECC06	ECC06'	ECC08
58	pocket	170.7	266.6	13815	3.750	0.000	6.568	2241	0	1318
48	hat	126.5	221.4	19701	4.454	4.274	4.274	4923	2790	2359
5	Heart	108.1	213.4	24123	2.784	0.731	0.731	2378	482	482
4	HCircle	127.0	250.2	28821	0.000	0.000	1.454	0	0	404
18	cattle	99.6	198.2	9764	1.223	1.223	1.223	2154	258	258
11	bird	116.0	230.1	14396	1.209	0.000	0.000	3963	0	0

#DT(ECC) / #DT(RECC): average number of times the distance transform(DT) is called wrt. RECC (in percent);

100% correct: the number of shapes for which the error was 0 for all pixels / the total number of shapes.

All three algorithms produce a good ECC approximation in about 8% to 15% of the time of RECC. There are only a few shapes for which the approximation was not 100% correct and the highest difference was about 7 pixels less than correct eccentricity in an image where the eccentricities varied from 170 to 266.6 pixels. Table 3 lists the 6 worst results with the three algorithms. Each shape is characterized by its number, its name, the range of eccentricities of RECC and the number of pixels (size). The next columns list the largest difference in eccentricity value and the number of pixels that were different. To judge the quality of the results we selected the example *hat* which had errors in all three algorithms (Fig. 2 shows the results by a contour line plot with the same levels). Algorithms ECC06' and ECC08 compute the correct eccentricity transform for all of the "problem" shapes showing the improvement of the discrete arc paving with respect to 4- and 8-connectivity used in [21] (see Table 4).

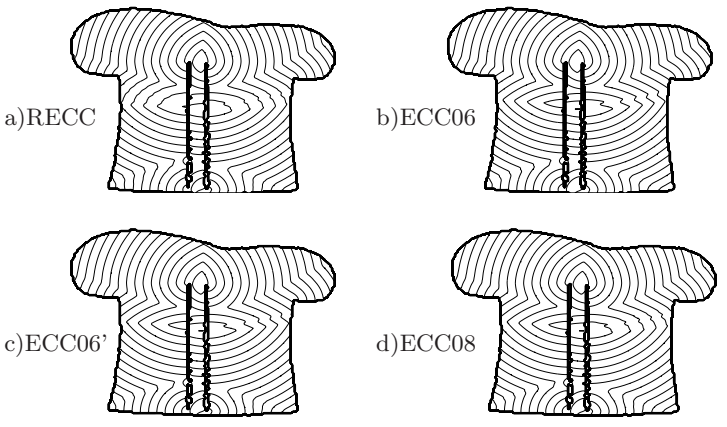
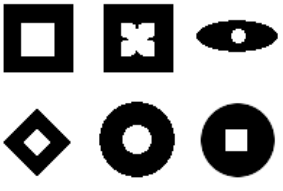


Fig. 2. Results of example shape *hat*

Table 4. Results on the 6 "problem" shapes from [21]



measure in %	ECC06	ECC06'	ECC08
max.pixel error	1.521 / 74.7	0.00 / 74.7	0.00 / 74.7
max.error size	675 / 1784	0 / 1784	0 / 1784
$\frac{\#DT(ECC)}{\#DT(RECC)}$	27%	50%	48%
100% correct	1 /6	6/6	6/6

Table 5. Results for image '3holes'

measure	ECC06	ECC06'	ECC08
max.pixel error	1.913 /409.2	1.100 / 409.2	12.773 / 409.2
max.error size	360 / 19919	119 / 19919	698 / 19919
$\frac{\#DT(ECC)}{\#DT(RECC)}$	7%	8%	9%

Table 5 shows the results of the three algorithms on an example 2D shape. On this example ECC06 and ECC06' produce better results than ECC08.

Overall ECC06' produces the best results with a computation speed between ECC06 and ECC08. ECC06 is the fastest in this experiment.

5 Conclusion

This paper presents a method for efficiently computing the shape bounded distance transform using discrete circles, which in turn is used by the three approximation strategies: ECC08, a novel one; ECC06, an existing one; and ECC06', a refined version of ECC06. They all approximate the eccentricity transform of a

discrete 2D shape. Experimental results show the excellent speed/error performance. Extensions to 3D and higher dimensions are planned.

References

1. Rosenfeld, A.: A note on 'geometric transforms' of digital sets. Pattern Recognition Letters 1(4), 223–225 (1983)
2. Gorelick, L., Galun, M., Sharon, E., Basri, R., Brandt, A.: Shape representation and classification using the poisson equation. CVPR (2), 61–67 (2004)
3. Kropatsch, W.G., Ion, A., Haxhimusa, Y., Flanitzer, T.: The eccentricity transform (of a digital shape). In: Kuba, A., Nyúl, L.G., Palágyi, K. (eds.) DGCI 2006. LNCS, vol. 4245, pp. 437–448. Springer, Heidelberg (2006)
4. Ogniewicz, R.L., Kübler, O.: Hierarchic Voronoi Skeletons. Pattern Recognition 28(3), 343–359 (1995)
5. Siddiqi, K., Shokoufandeh, A., Dickinson, S., Zucker, S.W.: Shock graphs and shape matching. International Journal of Computer Vision 30, 1–24 (1999)
6. Paragios, N., Chen, Y., Faurgeras, O.: Handbook of Mathematical Models in Computer Vision, vol. 06, pp. 97–111. Springer, Heidelberg (2006)
7. Soille, P.: Morphological Image Analysis. Springer, Heidelberg (1994)
8. Harary, F.: Graph Theory. Addison-Wesley, Reading (1969)
9. Diestel, R.: Graph Theory. Springer, New York (1997)
10. Klette, R., Rosenfeld, A.: Digital Geometry. Morgan Kaufmann, San Francisco (2004)
11. Ion, A., Peyré, G., Haxhimusa, Y., Peltier, S., Kropatsch, W.G., Cohen, L.: Shape matching using the geodesic eccentricity transform - a study. In: 31st OAGM/AAPR, Schloss Krumbach, Austria, May 2007, OCG (2007)
12. Maisonneuve, F., Schmitt, M.: An efficient algorithm to compute the hexagonal and dodecagonal propagation function. Acta Stereologica 8(2), 515–520 (1989)
13. Ion, A., Peltier, S., Haxhimusa, Y., Kropatsch, W.G.: Decomposition for efficient eccentricity transform of convex shapes. In: Kropatsch, W.G., Kampel, M., Hanbury, A. (eds.) CAIP 2007. LNCS, vol. 4673, pp. 653–660. Springer, Heidelberg (2007)
14. Suri, S.: The all-geodesic-furthest neighbor problem for simple polygons. In: Symposium on Computational Geometry, pp. 64–75 (1987)
15. Mora, F., Ruillet, G., Andres, E., Vauzelle, R.: Pedagogic discrete visualization of electromagnetic waves. In: Eurographics poster session, EUROGRAPHICS, Granada, Spain (January 2003)
16. Andres, E.: Discrete circles, rings and spheres. Computers & Graphics 18(5), 695–706 (1994)
17. Reveillès, J.-P.: Géométrie Discrète, calcul en nombres entiers et algorithmique (in french). University Louis Pasteur of Strasbourg (1991)
18. Andres, E., Jacob, M.A.: The discrete analytical hyperspheres. IEEE Trans. Vis. Comput. Graph. 3(1), 75–86 (1997)
19. Schmitt, M.: Propagation function: Towards constant time algorithms. In: Acta Stereologica: Proceedings of the 6th European Congress for Stereology, Prague, September 7-10, vol. 13(2) (December 1993)
20. Latecki, L.J., Lakämper, R., Eckhardt, U.: Shape descriptors for non-rigid shapes with a single closed contour. In: CVPR 2000, Hilton Head, SC, USA, June 13-15, pp. 1424–1429 (2000)
21. Flanitzer, T.: The eccentricity transform (computation). Technical Report PRIP-TR-107, PRIP, TU Wien (2006)