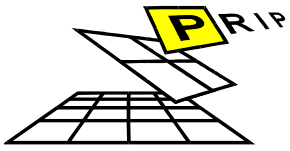
PRIP-TR-107

The eccentricity transform (computation)

Thomas Flanitzer

Pattern
Recognition &
Image
Processing
Group



Institute of
Computer Aided Automation

Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/183-2
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: e0300286@student.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-107 July 26, 2006

The eccentricity transform (computation)[1]

Thomas Flanitzer

Abstract

The eccentricity of a vertex is the longest shortest distance to any other vertex in a graph. We introduce the *eccentricity transform* which calculates the eccentricity for every point in a graph. Applied to digital images it offers some interesting properties including invariance to articulated motion and robustness whith respect to salt & pepper noise. Applied to graphs with an embedding it can be used for boundary determination. Its characteristics make it a good candidate for supporting or even replacing the distance transform as a basic tool in many feature extraction tasks (e.g. shape description). This report focuses on the computation of the eccentricity transform and explains implementation approaches.

[1]Thanks to Prof. Walter G. Kropatsch, Adrian Ion and Yll Haxhimusa for their guidance and support.

Figure 1: Instability of Skeletons derived by the distance transform with respect to small changes on the boundary.

1 Introduction

Object recognition on a computer system is mainly possible by extracting information about the objects presented on a digitised image. This can be a form of "abstraction", where essential features are extracted while unnecessary details are neglected. The extracted information can be color and geometry based. One such property determined by the objects geometry is shape which is examined by different shape analysis methods. Many of them are based on the distance transform.

This opens many problems which are caused by the instability of the distance transform e.g. with respect to small changes on the boundary of a shape. An example of this is displayed in Figure 1.

The *eccentricity transform* is intended to provide additional shape based information, which could be used to overcome these problems. Being robust against various types of noise (e.g. salt & pepper), it could be combined with or even replace the distance transform as a basic tool in several feature extraction tasks (e.g. shape description and matching).

Our work consisted of making the eccentricity transform practically usable. An efficient algorithm was developed and finally implemented in form of a MATLAB toolbox.

1.1 State of the Art

Given a grid metric ($d : \mathbb{Z}^2 \to \mathbb{Z}$), the *distance transform* [1, 5] associates to every pixel of a shape the smallest distance to a pixel on the boundary of this shape. Given a binary shape $S \subset \mathbb{Z}^2$ (ones define the shape, zeros the background) with p being a pixel of this shape, the distance transform is defined as follows:

$$dt(p) = \min\{d(p, q) : q \in \overline{S}\}$$

Algorithmically, the distance transform can be determined efficiently using a two pass algorithm [1, 5].

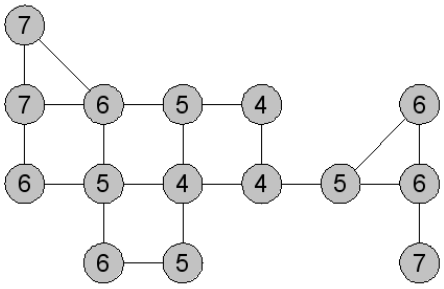


Figure 2: Eccentricity of a graph

Based on the distance transform it is possible to calculate the *medial axis transform* (MAT) [5, 1] or also called the *distance skeleton* [2]. The MAT consists of all centers of maximum disks which can be inscribed in the given shape. For each center, it contains the radius of the disk. Given a MAT it is possible to exactly reconstruct the original shape if necessary.

A similar abstraction of shapes is the *skeleton* [2, 8], which is generally determined by thinning algorithms. For the same shape its form is identical to the medial axis. But in contrast it provides no further information of the points as it is only binary. Therefore the skeleton is a very compact representation of a shape which preserves many of the topological and proportion characteristics of the original shape.

Shock graphs [7] are an example of an even further abstraction of shapes. They represent skeletons as directed, acyclic graphs using 4 types of shocks to describe the singularities.

2 Definition of the eccentricity transform

Defined for Graphs in [3], the *eccentricity* $ecc(u)$ of a given node u in a Graph $G(V, E)$ is

$$ecc(u) = \max\{d(u, v) : v \in V\}. \tag{1}$$

The eccentricity of a point is the shortest distance to the point in the graph, which is the farthest away.

The *eccentricity transform* of a graph is determined by calculating the eccentricity for every node. An example for the eccentricity transform of a graph is illustrated in Figure 2.

Applied to two-dimensional binary shapes, the nodes in the graph are associated to pixels and the distance between them is determined by the se-

(a) Shape (b) 4-nb (c) 8-nb

Figure 3: Isoheight lines for the eccentricity transform of a simple shape (a) with city-block (b) and chessboard metric (c). Where continuous, lighter means higher value.

lected metric. Important to mention is that the distance is always calculated using only pixels which are situated inside the shape. An example of the eccentricity of a shape using the city-block and the chessboard metric can be seen in Figure 3.

3 Properties

Diameter The maximum value of the eccentricity transform of a shape is the diameter of this shape.

Center The eccentricity transform produces a distance value for every vertex. The vertices with the minimum value therefore have the smallest distance to any eccentric vertex. These are called the center of the graph. The center of a discrete region is always a part of this region.

Robustness The eccentricity transform is robust with respect to (salt and pepper) noise. As long as the connectivity of the shape is not broken, if single vertices change, all the paths which would cross these vertices just go around them. This makes the eccentricity values only a little bit larger. Additionally in the case of discrete metrics (like 4- and 8-connectivity), there is a rather high probability that there are several paths with the same shortest length (see next property). In such cases the eccentricity is only affected if all shortest paths between a vertex and its farthest vertex are disturbed.

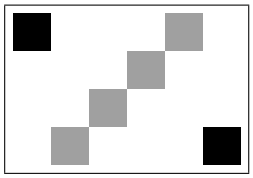


Figure 4: In the discrete space space with 4-connectivity there can be more than one shortest paths between two points (the points of interest are black, the centers of the paths between them are gray).

4-connectivity In the discrete space with 4-connectivity, the shortest path between two points is a series of steps along the two coordinate axes. Any permutation of these steps is also a shortest path. An example of this is illustrated in Figure 4. This fact increases the robustness of the eccentricity transform but has also two other consequences:

1. There is not a single midpoint between the two endpoints making the center of an elongated region a diagonal line. In fact the length of the line between the two end points is as long as the smaller coordinate differences of the two end points.

2. Since the number of midpoints depends on the angle of the discrete line the resulting centers are no more rotationally invariant (which they are in the Euclidean case).

Maxima are all on the boundary Any bounded region has a boundary which separates it from the background. The background can be considered as the complement of the region with respect to the embedding space.

Boundary determination Because the maxima are only on the boundary, the vertices that make up the boundary do not have to be known a priori like in the distance transform. Instead the eccentricity transform can start on any vertex in the region and will reach the boundary only at its maximum values. In this case the image edges are used as the stopping criterion rather than the initial condition.

Complementarity between distance transform and eccentricity transform In the distance transform the smallest values are on the boundary of a shape. The highest values lie in the center, their local maxima are the centers of maximum disks. These centers can be used to determine a skeleton or

MAT. The eccentricity transform in contrast has the highest values on the boundary and the minimum defines the center.

Invariance As stated earlier, the eccentricity transform calculates distances only inside a given region. This makes it invariant to any translation. For the Euclidean metric it is also invariant to rotation. In the discrete space this is not the case because the length of a path following an imaginary line varies with changing orientation. It is also robust with respect to articulated motion of thin regions.

4 Computation

4.1 Naive Approach

The naive way to get the eccentricity of a shape, is to determine the maximum path that can be drawn for every point inside the shape. We have implemented this algorithm to get a feeling of the values produced by the eccentricity transform and to validate later results produced by an optimized algorithm. It relies on the function *bwspdist* which calculates the distance transform of a single point, using the specified metric. It is described in Algorithm 1.

Input: Shape S, Vertex p, Connectivity C
Result: Single point distance transform D
1 set D to zero;
2 $N :=$ boundary of p with connectivity C;
3 **repeat**
4 $\quad D(N) := i$;
5 $\quad i := i + 1$;
6 $\quad N :=$ boundary of D with connectivity C;
7 **until** D *contains all vertices of* S ;

Algorithm 1: function bwspdist(S, p, C)

An implementation of this function, which is based on morphological operations, can be found in appendix B.1.

Using this function, the eccentricity can be determined rather easily, Algorithm 2 shows a pseudo code description.

The complexity of the naive algorithm is $O(\mid V \mid^3)$. The function bwspdist could be replaced by Dijkstra's algorithm [3] for solving the single source

```
Data: Shape S, Connectivity C
Result: Eccentricity transform E
for each point p of S do
    D = bwspdist(S, p, C);
    E(p) = max(D);
end
```

Algorithm 2: Naive algorithm

shortest path problem. It has a complexity of $O(\mid V \mid^2 \log(\mid V \mid))$. This would result in a total complexity of $O(\mid V \mid\mid E \mid + \mid V \mid^2 \log(\mid V \mid))$.

4.2 Optimized algorithm

The optimized algorithm also relies on the function *bwspdist* but uses it in a more clever way by calculating it only for vertices which can generate new results.

The first phase of the algorithm relies on the fact that by jumping from one vertex to its most distant counterpart, a diameter's end is found. It is used by selecting an arbitrary vertex and determining the distance to all other vertices of the shape. Then the maximum is selected for the next iteration. The distance results of every iteration are merged using the max operation. It can be proven that no more than 3 iterations are needed until the result converges. For some simple shapes the result up to here is already the eccentricity transform.

An example why this is not the final result in all cases is illustrated in Figure 5. It is possible that some eccentric vertices are not taken into account.

To overcome this problem the algorithm performs some additional steps. Because of the fact that the minima of the eccentricity transform make up the center and the previous steps calculated the eccentricity along a diameter of the shape, the minimum of the current result should already be a good approximation of the center. The advantage is that from the center all eccentric points are easily detectable. If the distances from the center to all the other points gets calculated, the diameter endpoints are the local maxima of the result.

Due to some properties of the discrete space, there can be multiple shortest paths between two points and the center at this stage can consist of a set of points (see Fig. 4). So what is done is that an arbitrary point of the center is selected. Then the local maxima of the result are determined, which yields a set of points which might not have been taken into account in the previous iterations. They get analyzed by calculating the distances to all the other

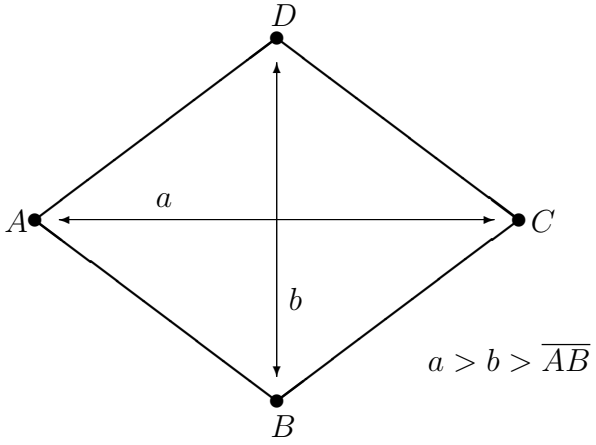


Figure 5: Some eccentric vertices may not be taken into account in the first steps of the algorithm. If the algorithm starts at point A, it will find C as the most distant point. Using C in the next iteration it will again find A and the algorithm converges although all vertices whose eccentric point is not A or C do not have the correct eccentricity value assigned.

points and the results are merged with the already determined values. Then again a point of the minima is selected and the whole procedure is repeated until the result converges. The result after that is the eccentricity transform.

A pseudo code description is shown in Algorithm 3. The optimized algorithm is much faster than the naive approach but can be proven to give correct results only on simply connected shapes (no holes).

4.3 Dead ends

Initial algorithm version The first versions of the algorithm included only the first phase of the algorithm (Algorithm 3 line 1-8). A very simple example shape where this does not work is a 3x3 square. See Figure 6 for the iterations of such a shape. We tried to fix this issue by masking the points that have already been calculated. This version produced correct results for the 3x3 square but still did not work for other shapes (e.g. every rectangle bigger than 3x3).

Points with less than 3 neighbors Another approach we have tried was to select all points of a shape which have less than 3 neighbors and then calculate the distances to all the other points. We had this idea because the

Data: Shape S, Connectivity C, Minima selection method msm
Result: Eccentricity transform E

Select an arbitrary vertex p of S;
repeat
    $D := \text{bwspdist}(S, p, C)$;
    $h := max(D)$;
    $E := \max(D, E, h - D)$;
    mark p as visited;
    $p :=$ vertex with maximum value in E;
until *E not changed* ;
`/* first convergence */`
repeat
    $M :=$ *one* arbitrary or *all* unvisited vertices with minimum of E;
    for *each unvisited m of M* **do**
        $D := \text{bwspdist}(S, m, C)$;
        $E := \max(D, E)$;
        mark p as visited;
    end
    $L :=$ all local maxima of E;
    for *each unvisited l of L* **do**
        $D := \text{bwspdist}(S, l, C)$;
        $E := \max(D, E)$;
        mark p as visited;
    end
until *E not changed* ;
`/* final convergence */`

Algorithm 3: Optimized algorithm

4	3	2
3	2	3
2	3	4

(a) First Iteration

4	3	2
3	2	3
2	3	**4**

(b) Second Iteration, convergence

4	3	4
3	2	3
4	3	4

(c) Correct result

Figure 6: The iterations until the first convergence for a 3x3 square. The used vertex in each iteration is shown bold.

diameter endpoints in a discrete metric with 4-connectivity always have less than 3 neighbors. Otherwise there is always a point that is farther away which would yield to a longer diameter. This approach did indeed work at least for 4-connectivity. But it was not very general and with some shapes it caused a lot of redundant calculations. It also did not perform any better than the optimized algorithm on not simply connected shapes.

Not simply connected shapes We have encountered that the optimized algorithm does not always produce the correct result for shapes which have holes and therefore are not simply connected. More specifically the problem occurs when the center points are not connected. Among other things, we tried to take all minima into account for finding diameter endpoints. This produced results which were more similar to the correct ones but still not equal. Therefore the algorithm at its current state is only applicable to simply connected shapes.

No first algorithm phase It should be noticed that a slight modification of the optimized algorithm would also work without the first phase. This has been tried during the development of the algorithm.

Although this approach is more simple, experiments showed that it is slower than the optimized algorithm.

5 Implementation

The described algorithm was implemented in MATLAB 7, the source code can be found in Appendix B. The function *bwexc* which implements the eccentricity transform was intended to be similar in use to *bwspdist* of the Image Processing Toolkit which implements the distance transform.

In the following, some implementation specific details are explained.

Minima selection method Several ways to select minima were implemented. These included the selection of one arbitrary minimum, one random minimum and all minima. The selection of all minima was especially implemented as an attempt to overcome the production of incorrect results with not simply connected regions (but which turned out not to solve the problem). However, the three selection methods are still in the implementation and can be selected by a function parameter. As default an arbitrary minimum is selected which is implemented by simply taking the first minimum of the vector returned by Matlab.

Don't visit again Due to the functionality of the algorithm it is likely that points are selected multiple times to calculate the distances to all the other points. But this would in any case just be a waste of time because all results are merged anyway (using max) and therefore recalculating cannot change the results. For this reason a mask matrix is managed to disable points when they get analyzed. Multiple calculation of the same data is avoided.

5.1 Room for improvements

A major issue with the current implementation is its low performance. Compared to the distance transform of the MATLAB Image Processing Toolbox it is very slow. To get an impression, on an AMD Athlon64 3000+ with 512 MB RAM it takes about 9 milliseconds to calculate the distance transform of the hand picture (129x129) which was used in the experiments. The eccentricity transform takes almost 12 seconds.

The main performance bottleneck is the function bwspdist which gets called very often and takes up most of the processing time. When calculating the hand picture, bwexc gets called 64 times and is active for 98% of the processing time. The overall performance of the algorithm could be greatly improved if this function was optimized.

6 Experiments

As a basis for the experiments we used the Kimia database [7]. As the library is quite big (137 images) and the processing time of the eccentricity transform is rather high, we used only a subset containing 10 images for most of the experiments. Each image of this subset is chosen to represent a typical type of shapes.

Because the properties of the invariance, with respect to articulated motion, were also to be tested, we selected another subset containing 10 images of a human hand in various positions.

Finally to test the correctness of the algorithm we also created a set of "problem" shapes. These are shapes which are not simply connected (see Appendix A.4). The optimised algorithm still fails to correctly determine the eccentricity transform of most of them.

In the following, some details of the experiments are explained.

6.1 Noise robustness comparison to the distance transform

We compared the robustness of the distance and eccentricity transform with respect to noise. We used two kinds of noise: Random salt & pepper noise and circumcision to simulate common segmentation errors. A table containing the results can be found in Appendix A.2.

As stated earlier, the eccentricity transform is very robust with respect to salt & pepper noise and in this case clearly outperforms the distance transform.

Looking at the results using the circumcised sets, it can be stated generally that the eccentricity is robust to circumcision as long as the diameter of the shape is not affected. In this case there is no difference to the results produced using the original shapes. If the diameter is affected the eccentricity produces rather high error rates proportional to the diameter length change.

6.2 Invariance with respect to articulated motion tested using the hands subset

In this experiment the eccentricity transform of the images of the hands set was calculated. In the results, the minima are marked with red dots, the local maxima with green dots. The results are illustrated in Appendix A.1. It can be noticed that the position of these dots on the shape roughly stays the same even when the fingers are bent or the hand is rotated.

6.3 Minima selection compared

We made an experiment which calculates the eccentricity transform for every image with all given minima selection methods. The outputs are then compared to reference results determined by the naive algorithm.

As expected, for simply connected shapes there was no difference to the results determined by the naive algorithm for none of the minima selection methods. On the basis of this fact it is sustainable to use the minima selection method with the lowest computational complexity which is the "any"-method.

To test how well the results for not simply connected shapes approximate the correct ones we compared these with respect to the root mean square error (RMSE), maximum deviation and correlation of the local maxima. The results of this experiment can be found in Appendix A.4. It is interesting that for not simply connected shapes the "all" minima selection method does

not always produce the best results. Surprisingly, the "any" and "random" method often produce results which are closer to the reference.

6.4 Correlation experiments with circumcised shapes

The values of the eccentricity transform in a discrete metric change when the shape is rotated, circumcised, scaled or affected by noise. But the position of the local maxima roughly stays the same. So the position of the local maxima and the relation between them is an interesting descriptor for a shape.

To test this kind of invariance we used an experiment where the eccentricity transform is calculated for a shape with and without noise. Then the local maxima of both results and the correlation between them are determined. This was done using random salt & pepper noise and circumcision as noise.

The correlation results, which can be found in Appendix A.3, look very promising. Even when the original shapes are circumcised radically, the correlation roughly drops below 0.8.

6.5 Development of eccentricity transform results with respect to increasing salt & pepper noise

Earlier experiments showed that the eccentricity transform is stable with respect to 5% salt & pepper noise. In this experiment we measured how our optimized algorithm develops when the noise is increasing. In Appendix A.5 the RMSE, the maximum deviation and the local maxima correlation are drawn over the noise level for three different shapes. The results were also determined for each minima selection method.

The results show that the RMSE develops rather linear for compact shapes like the car. It is more subject to fluctuations if the shape is less compact.

The maximum deviation always rises very early to a high level where it remains rather constant. This happens because single vertices get cut out of the shape, which very likely happens on the boundary. The eccentricity value then drops to zero which results in a high deviation.

The correlation of the local maxima seems to be not so stable against increasing salt & pepper noise.

7 Conclusion

The eccentricity transform applies to every pixel of a shape the length of the longest shortest path to any other pixel of the shape. Its robustness with respect to salt & pepper noise and minor segmentation errors has been tested and verified. It was especially compared to the distance transform and has shown to be more stable in many cases. Besides its general robustness, the local maxima of the resulting values also offer some interesting properties. Their positions relative to the shape are highly invariant to articulated motion and rotation which might be of use for shape description.

An efficient algorithm was developed and implemented in MATLAB. It works much faster than the naive algorithm and gives correct results on simply connected shapes. It does not always work on not simply connected shapes, but it has been shown that it gives at least a good approximation of the correct results.

References

[1] Reinhard Klette, Azriel Rosenfeld: *Digital Geometry*, Morgan Kaufmann Publishers Inc, US, 2004

[2] Gisela Klette: *Skeletons in Image Processing*, CITR-TR-112, CITR Tamaki, University of Auckland, 2002

[3] Fred Buckley, Frank Harary: *Distance in Graphs*, Perseus Books (Sd) (January, 1990)

[4] Remco C. Veltkamp, Michiel Hagedoorn: *State-of-the-Art in Shape Matching*, Utrecht University, Department of Computing Science

[5] Robert M. Haralick, Linda G. Shapiro: *Computer and Robot Vision*, Addison-Wesley Publishing Company, 1992

[6] Axel Pinz: *Bildverstehen*, Springer Verlag Wien New York, 1994

[7] Kaleem Siddiqi, Ali Shokoufandeh, Sven J. Dickinson, Steven W. Zucker: *Shock Graphs and Shape Matching*, International Journal of Computer Vision, 30, 1999

[8] Milan Sonka, Vaclav Hlavac, Roger Boyle: *Image Processing, Analysis and Machine Vision*, Chapman & Hall Computing, 1993

A Experiments

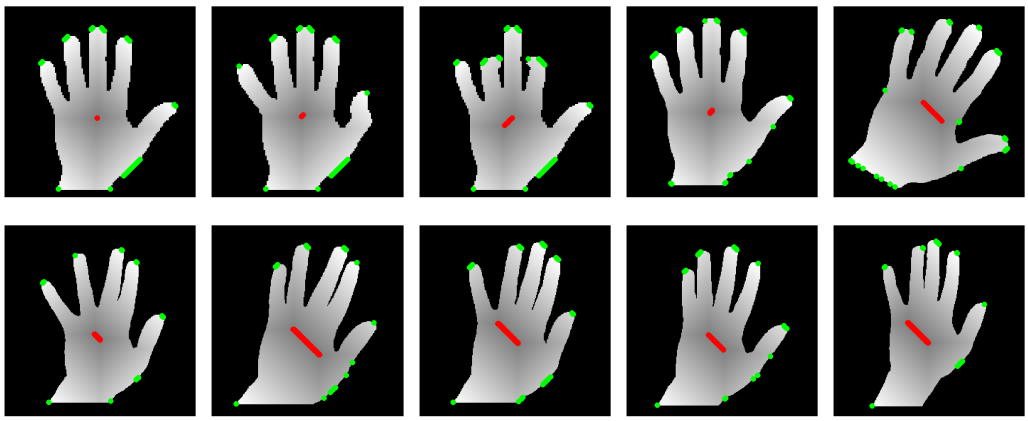
Abbreviations used in the description of the experiments

- RMSE . . . Root mean square error
- max.dev . . . Maximum deviation
- cor . . . Correlation

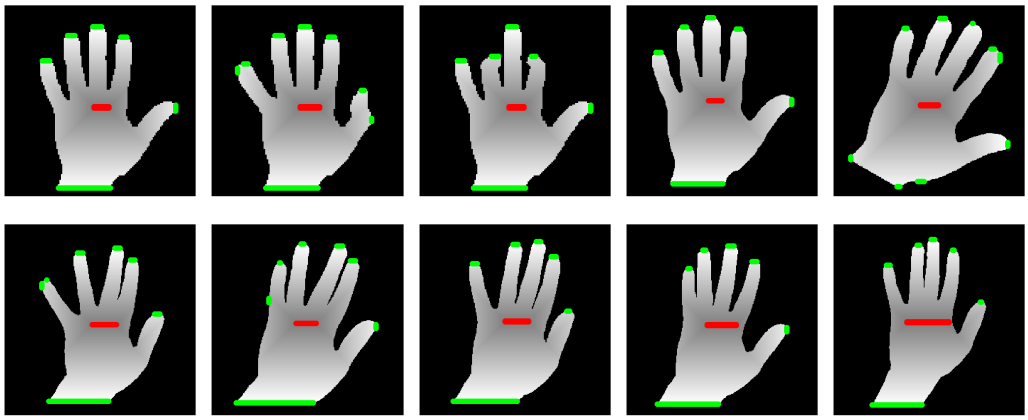
A.1 Eccentricity transform of human hand images

The green dots mark the local maxima, the red dots the minima.

4-connectivity



8-connectivity



A.2 Noise robustness comparison between DT and Ecc

For this experiment only 4-connectivity was used. RMSE and max.dev give a comparison to the results without noise.

Original shapes											
											
Salt & Pepper Noise (5%)											
dt	RMSE	15.83	6.27	3.58	8.48	6.84	3.36	5.51	2.79	4.33	0.42
	max.dev	43	17	10	30	24	14	15	11	15	4
ecc	RMSE	0.03	0.24	1.48	0.05	0.19	1.55	0.09	1.77	0.10	0.56
	max.dev	2	2	4	2	2	4	2	6	2	6
Circumcised shapes I											
											
dt	RMSE	2.51	1.33	0.46	1.01	0.36	1.77	0.98	0.35	2.71	0.04
	max.dev	19	8	5	7	6	11	8	5	8	1
ecc	RMSE	10.96	0.00	0.00	0.00	0.00	0.00	5.38	0.00	0.00	0.00
	max.dev	16	0	0	0	0	0	13	0	0	0
Circumcised shapes II											
											
dt	RMSE	8.01	0.35	0.90	0.66	2.35	0.80	2.15	0.62	0.30	0.04
	max.dev	35	4	10	6	14	8	12	9	5	1
ecc	RMSE	27.41	49.17	71.86	0.06	30.07	36.82	8.85	21.90	53.73	26.70
	max.dev	38	64	83	2	42	64	24	46	63	36

A.3 Correlation of the local maxima between the eccentricity transform of a shape and its circumcised equivalent

The correlation values shown have been calculated between the results using the circumcised and the original shapes.

Original shapes


Circumcised shapes I										
										
4-nb.	0.73	1.00	1.00	0.96	0.96	1.00	0.77	1.00	1.00	0.95
8-nb.	0.93	1.00	0.72	0.97	1.00	0.82	1.00	1.00	1.00	0.98
Circumcised shapes II										
										
4-nb.	0.71	0.79	0.97	0.96	0.89	0.97	0.71	0.98	0.87	0.92
8-nb	0.48	0.45	0.90	0.96	0.72	0.65	0.98	0.97	0.73	0.98

A.4 Similarity of the results of the optimized algorithm to the correct results using all different minima selection methods

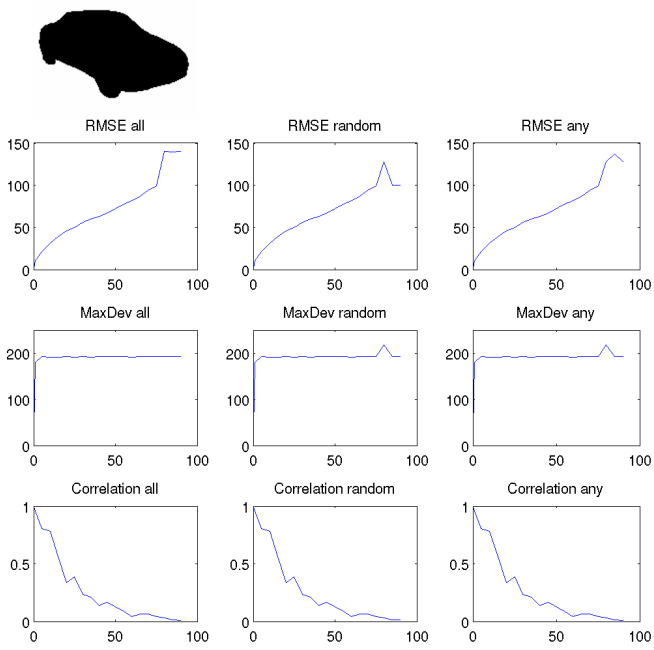
The values show the difference to the results determined by the naive algorithm.

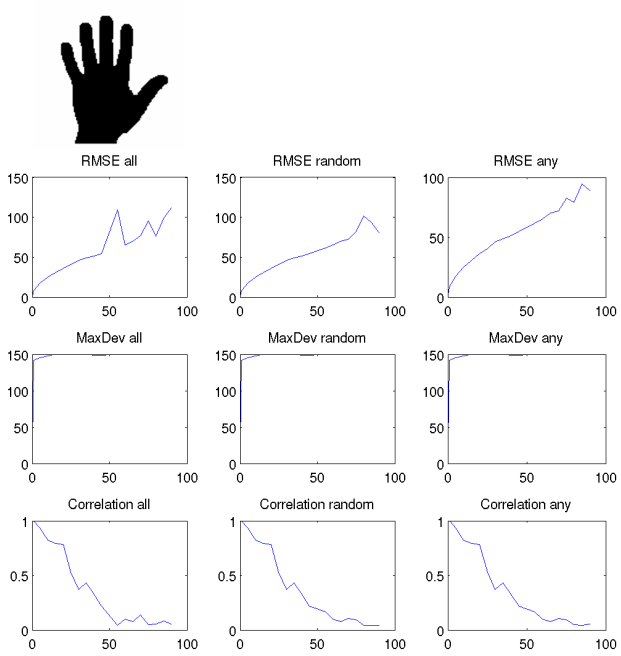
							
		4-connectivity					
all	RMSE	0.00	0.11	0.33	0.00	0.38	0.24
	max.Dev	0.00	1.00	1.00	0.00	1.00	1.00
	corr	1.00	1.00	0.71	1.00	0.84	0.96
random	RMSE	0.12	0.11	0.33	0.00	0.57	0.34
	max.Dev	1.00	1.00	1.00	0.00	2.00	1.00
	corr	1.00	1.00	0.71	1.00	0.73	0.92
any	RMSE	0.12	0.14	0.23	0.00	0.66	0.36
	max.Dev	1.00	1.00	1.00	0.00	2.00	1.00
	corr	1.00	1.00	0.87	1.00	0.72	0.89
		8-connectivity					
all	RMSE	19.00	18.84	9.47	15.41	18.83	15.53
	max.Dev	25.00	25.00	12.00	19.00	21.00	21.00
	corr	0.58	0.62	1.00	1.00	1.00	0.89

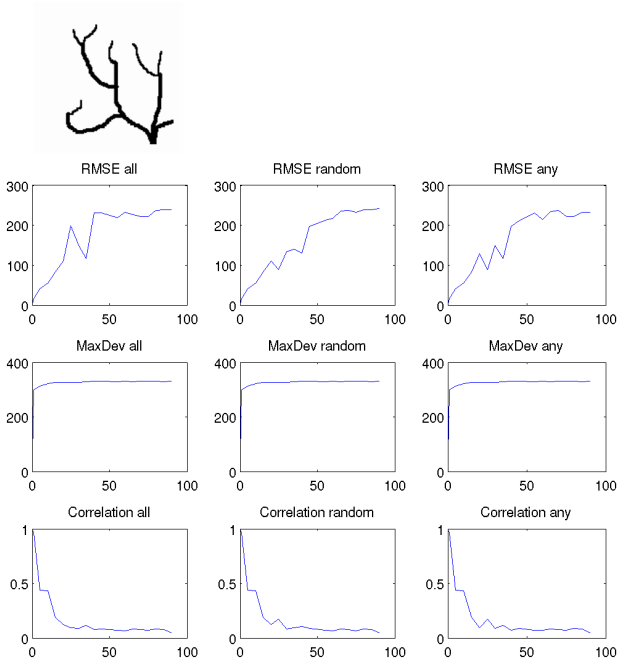
random	RMSE	19.00	18.84	9.47	15.41	18.93	15.71
	max.Dev	25.00	25.00	12.00	19.00	22.00	22.00
	corr	0.58	0.62	1.00	1.00	0.86	0.78
any	RMSE	19.00	18.84	9.47	15.41	19.10	15.58
	max.Dev	25.00	25.00	12.00	19.00	23.00	21.00
	corr	0.58	0.62	1.00	1.00	0.79	0.87

A.5 Development of eccentricity transform results with respect to increasing salt & pepper noise

The values show the difference to the results determined using the original shape.




RMSE all
RMSE random
RMSE any
MaxDev all
MaxDev random
MaxDev any
Correlation all
Correlation random
Correlation any


RMSE all
RMSE random
RMSE any
MaxDev all
MaxDev random
MaxDev any
Correlation all
Correlation random
Correlation any

B Matlab Sourcecode

B.1 bwspdist.m

```
%    Eccentricity (computation)
%    ---------------------------------------------------------
%    Thomas Flanitzer <t.flanitzer@gmx.at> e535 0300286
%    Version: 0.2   Date: 2. 6. 2006
%    ---------------------------------------------------------
%    Single Point Distance Transform
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function D = bwspdist(S, p, CONN)

    % check params
    if (nargin < 3 || CONN == 4)
        sel = strel('diamond',1);
    elseif (CONN == 8)
        sel = strel('square',3);
    end

    % D same Size as S
    D = zeros(size(S));

    % set seed
    D(p(1),p(2))=1;

    % Continously dilate the form
    v = 1;
    DL = D>0;
    % pixels to dilate
    toDil = DL;

    while(true)
        DB = imdilate(toDil, sel);
        % cut with the form
        DBS = DB & S;
        % exclude already dilated pixels
        DB = DBS & ~DL;
        % form for the next iteration
        DL = DBS;
        % dilate only new pixels
        toDil = DB;
```

```
        % find new
        B = find(DB);

        % no new: end
        if isempty(B), break, end;

        % set distance value
        D(B) = v;
        v = v+1;
    end

    % clear seed
    D(p(1),p(2))=0;
end
```

B.2 bwexc.m

```
%   Eccentricity (computation)
%   ---------------------------------------------------------
%   Thomas Flanitzer <t.flanitzer@gmx.at> e535 0300286
%   Version: 0.1   Date: 2. 6. 2006
%   ---------------------------------------------------------
%   BW Eccentricity
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function E = bwexc(S, CONN, MINSELECT)

    % DEBUG on/off
    DEBUG = 0;

    % check params
    if (nargin < 2)
        CONN = 4;
    end

    if (nargin < 3)
        MINSELECT = 'any';
    end

    S = ~S;

    % DVA-mask
    M = ones(size(S));
```

```
% eccentricity matrix
E = zeros(size(S));

% select first point
shapePixels = find(S);
[p(1) p(2)] = ind2sub(size(S),shapePixels(ceil(rand(1)*
    length(shapePixels))));
% exclude point for further iterations
M(p(1),p(2)) = 0;

% phase 1
i = 1;
while(1)
    EOLD = E;

    D = bwspdist(S, p, CONN);
    M(p(1),p(2)) = 0;

    % find next point
    m = max(D(:));
    [p(1) p(2)] = find(D==m, 1, 'first');

    % merge results with eccentricity matrix
    E = max(max(D, m-D), E);
    E = E.*double(S);

    % if convergence: end
    if (~isequal(E,EOLD))
        continue;
    else
          break;
    end
end

if (DEBUG)
    img = 1;
end

while(1)
    EOLD = E;

    % find minumum value
```

```
mi = min(E(find(S)));

switch lower(MINSELECT)
    case 'all'
        mins = find((E.*M)==mi);

    case 'any'
        mins = find((E.*M)==mi,1,'first');

    case 'random'
        mins = find((E.*M)==mi);
        if (length(mins) > 0)
            mins = mins( ceil(rand(1)*length(mins))
                );
        end

    otherwise
        error('Unknown minimum-selection method');
end

RMax = 0;
% use all selected minima
for i=1:length(mins)
    % calc bwspdist of a point
    [x(1) x(2)] = ind2sub(size(S), mins(i));
    Dmi = bwspdist(S, x, CONN);

    % merge results with eccentricity matrix
    E = max(Dmi, E);
    M(mins(i)) = 0;

    % find local maxima
    RMax = RMax | imregionalmax(Dmi, CONN);
end

if (DEBUG)
    % DEBUG: show local maxima
    subplot(3,3,img);
    imshow(S,[]);
    hold on;
    % DEBUG: show minima
    [ry rx] = ind2sub(size(S), mins);
    plot(rx, ry, 'm*');
```

```
        end


        % exclude already visited points
        RMax = RMax.*M;
        % determine remaining points
        rm = find(RMax);

        if (DEBUG)
            % DEBUG: show local maxima (already visited
                excluded)
            [ry rx] = ind2sub(size(S), find(RMax));
            plot(rx, ry, 'y*');
            img = img+1;
        end

        % process points
        for i=1:length(rm)
            [p(1) p(2)] = ind2sub(size(S), rm(i));

            % bwspdist of current point
            D = bwspdist(S, p, CONN);
            M(p(1),p(2)) = 0;
            m = max(D(:));

            % merge results with eccentricity matrix
            E = max(m-D, max(D, E));
            E = E.*double(S);
        end


        if (~isequal(E, EOLD))
            continue;
        else
            break;
        end

    end % while

end
```