

# Straight Skeletons for Binary Shapes

Markus Demuth

Institute for Theoretical Computer Science  
Graz University of Technology  
markus.demuth@tugraz.at

Franz Aurenhammer

Institute for Theoretical Computer Science  
Graz University of Technology  
auren@igi.tugraz.at

Axel Pinz

Institute of Electrical Measurement and Measurement Signal Processing  
Graz University of Technology  
axel.pinz@tugraz.at

## Abstract

*This paper reviews the concept of straight skeletons, which is well known in computational geometry, and applies it to binary shapes that are used in vision-based shape and object recognition. We devise a novel algorithm for computing discrete straight skeletons from binary input images, which is based on a polygonal approximation of the input shape and a hybrid method that combines continuous and discrete geometry. In our experiments, we analyze the potential of straight skeletons in shape recognition, by comparing their performance with medial-axis based shock graphs on the Kimia shape databases. Our discrete straight skeleton algorithm is not only outperforming typical skeleton algorithms in terms of computational complexity, it also delivers surprisingly good results in its straightforward application to shape recognition.*

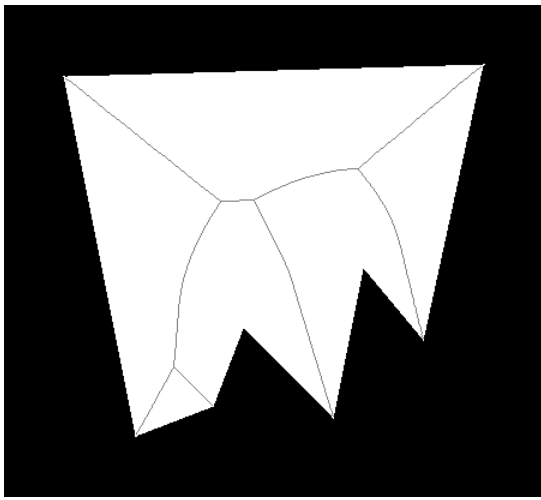
## 1. Introduction

The medial axis is a versatile descriptor of shape, in its continuous form as well as in its discrete form. In particular, the medial axis of binary images is of importance in computer graphics and vision. The skeletonization of an object is a main preprocessing task for object recognition and classification. Geometrically, the medial axis is defined to consist of all points inside the object that are equidistant from at least two points on the object boundary. A variety of exact construction algorithms exists; we refrain from a detailed discussion and refer to, e.g., [1] and references therein. In the discrete case, the medial axis is often derived from distance transforms [22] of the binary image, which are computationally expensive. Different distance transforms may lead to different medial axes. See, e.g., [9, 19] for a short

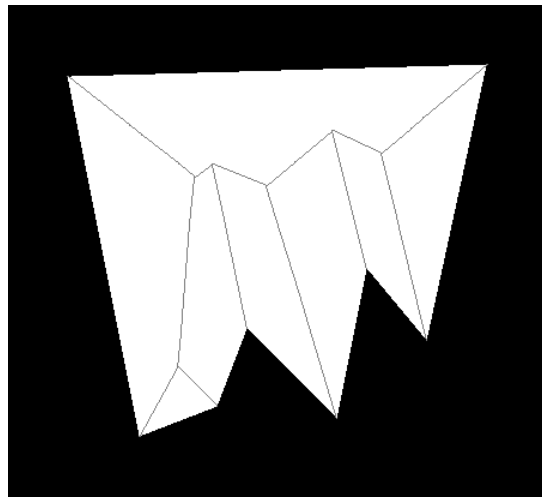
history on distance transform algorithms. As there exists no geometrically exact medial axis in the domain of discrete images [14], a variety of similar medial axes have been considered. Two algorithms which are commonly used in practice are the Hamilton-Jacobi skeleton algorithm by Siddiqi *et al.* [23] and the augmented fast marching method by Telea and van Wijk [25].

The main intention of the present paper is to introduce an alternative skeletal structure, the so-called straight skeleton [2, 3], to the computer vision community, and to present an efficient algorithm for its construction. In contrast to the medial axis, the definition of the straight skeleton is procedural, via a geometric shrinking process; see Subsection 1.1. The definition of its discrete version is nontrivial, because it depends on the way a binary image is approximated by a polygon; see Subsection 1.2. As the straight skeleton does not (and cannot) rely on any distance model, no distance transform has to be computed. This is an advantage which, in our experiments, leads to a speedup by a factor of about two, compared to medial axes with similar (and not too small) numbers of arcs. We propose a hybrid algorithm using geometric as well as pixel information to construct, in a first step, the (continuous) straight skeleton. Additional information can be obtained as well, comparable to what shock graphs [20, 24] make explicit for the medial axis. Indeed, more flexible variants of shock graphs are meaningful for the straight skeleton, by distinguishing between arcs that come from either convex or reflex vertices of the underlying polygon. Note that our algorithm provides a geometric description of the straight skeleton, whereas typical medial axis algorithms for binary images only produce a pixel approximation of the skeletal structure.

There are applications of the straight skeleton unrelated to those of the medial axis. They stem from a 3D interpre-



(a) Medial Axis. The classical medial axis is a tree structure composed of straight and parabolic arcs. No arcs emanate from reflex vertices of the defining polygon.



(b) Straight Skeleton. The straight skeleton of a polygon consists solely of straight line segments. It is a tree structure as well, whose leaves are exactly the vertices of the polygon.

Figure 1: Medial Axis vs. Straight Skeleton

tation of the straight skeleton [3] as the projection of the edge graph of a roof (or island) with fixed slope of its parts (hills). In this sense, a fast discrete straight skeleton algorithm might be of value in the automatic recognition of building structures (or geological formations) in a digital image. More potential applications of straight skeletons exist, for example, in polygon offset computations and computational origami; see, e.g., [8] for a short overview and further references.

### 1.1. Straight Skeleton

In computational geometry, the straight skeleton is a skeletal structure defined for a given planar straight-line polygon [2, 3]. It can be seen as a variant of the medial axis, because, in the case of convex polygons, both structures are identical.

The straight skeleton is defined by a shrinking process of the polygon. The edges of the polygon are moved inwards in a self-parallel manner and at the same speed. During the shrinking process, two kinds of events can occur. First, the length of an edge can shrink to zero, which is called an *edge event*. The shrinking process continues without any change after such an event. Second, a reflex polygon vertex could run into an edge. This is called a *split event*, due to the fact that the polygon is split into parts, and the shrinking process continues on each part. When all parts have shrunken to zero, the process terminates. The straight skeleton now consists of the paths drawn by the polygon vertices during the shrinking process. The union of these paths is a tree, and, following [2, 3], we denote its individual components with

*arcs* and *nodes*. Skeleton arcs are line segments, because the polygon vertices move on angle bisectors. Their endpoints, the nodes, correspond to the two events described above.

Figure 1 depicts the two skeletal structures, medial axis and straight skeleton, for the same polygon. No curved arcs do occur in the latter structure, whereas straight arcs emanate from reflex polygon vertices.

The straight skeleton is not based on any distance model and, therefore, standard distance-based construction techniques like insertion algorithms or divide and conquer methods cannot be applied. All known straight skeleton construction methods are based on the simulation of the shrinking process described above. They are computationally more expensive than algorithms which construct the medial axis. The method proposed in [3] does well for many inputs in practice but the worst-case analysis shows a running time of  $O(M^3 \log M)$ , where  $M$  denotes the number of vertices of the start polygon. The fastest deterministic algorithm to construct the straight skeleton, by Eppstein and Erickson [11], runs in  $O(M^{1+\epsilon} + M^{8/11+\epsilon} \cdot r^{9/11+\epsilon})$  time and space (where  $r$  denotes the number of reflex vertices and  $\epsilon > 0$  is fixed). It uses advanced data structures and is hard to implement. Cheng and Vigneron [8] present an algorithm to solve the motorcycle graph problem—a subproblem of computing the straight skeleton—which results in an  $O(M\sqrt{M} \log^2 M)$  randomized time construction of the latter structure. In contrast, certain medial axis algorithms run in  $O(M \log M)$  or even  $O(M)$  time; see, e.g., [1] for a short history.

## 1.2. Discrete Straight Skeleton

Given a binary input image, talking of its straight skeleton is not meaningful unless an approximating polygon is defined as input. An exception are isothetic shapes, for example, those being representable by the union of a finite set of axis-parallel rectangles. Even for such a representation, the straight skeleton will contain lots of detailed and unwanted features, unless an exact representation with a small number of rectangles is possible. Efficient algorithms for constructing the discrete straight skeletons for such images exist [4], but their application is naturally limited by the special shape of the allowed input images.

In our algorithm, the first step therefore is to find a polygonal approximation of the contour of the input shape. There exists a variety of algorithms for computing polygonal approximations of a digitized curve which are based on split-and-merge-techniques [18, 31], split-techniques [10] or finding maximum curvature points [30]. We chose the algorithm proposed by Perez and Vidal [17] which computes the optimal polygonal approximation with respect to the overall distance. Optimality is quite important in this case; a good adaption of the polygon to the digitized object (i.e. preserving its salient shape information) is crucial for obtaining descriptive polygons of small size. An improved analysis of this algorithm, by Horng and Li [13], shows a running time of  $O(Mn^2)$ , where  $n$  is the number of contour pixels, and  $M$  is the number of vertices of the approximated polygon. We assume that the shapes are taken from a shape database which represents natural objects (e.g. the well-known Kimia-25 and Kimia-99 databases [21]). As such objects tend to have big convex parts and also are not too jagged, we can assume that  $n$  is rather small in the number of pixels of the given shape. Furthermore, we choose  $M$  to be a small constant, and therefore the approximation algorithm is faster than distance transforms which run in time linear in the number of shape pixels. Figure 2 illustrates the results of this polygonal approximation for two shapes of the Kimia-25 database for several values of  $M$ .

The output of our algorithm (to be described in Section 2) is a correct continuous description of the straight skeleton. From this description, details of the shock graph, like distances to the boundary, or time stamps for various types of events, can be derived in an easy manner.

There are several interesting properties of our ‘discrete straight skeleton’ (DSS) implementation, including its low computational complexity that leads to superior runtime as compared to existing medial axis implementations. If required, a pixel representation of the straight skeleton can also be computed from its continuous description. We are not aware of any work applying straight skeletons to computer vision and graphics. An exception are Tănase

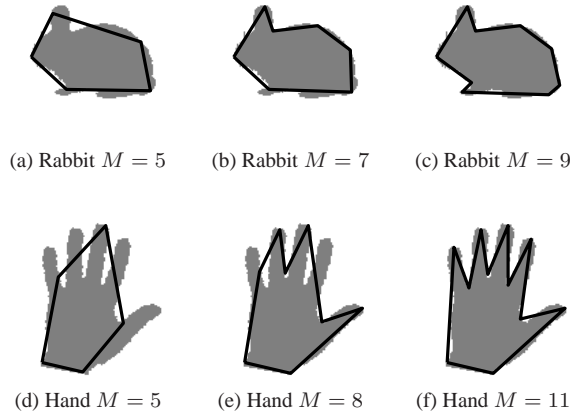


Figure 2: Polygonal approximation using the algorithm by Perez and Vidal. A few more vertices of the approximating polygon give a significantly better boundary description of these simple shapes.

and Veltkamp [28, 29], where straight skeletons (and certain variants) are used with advantage to decompose shapes from known databases into characteristic parts. To apply the DSS in computer vision, the most interesting questions are: What are the vision-specific differences between medial axis and DSS? How could the DSS be used in shape/object recognition? How does shock-graph based recognition differ for medial axis and for DSS? What are the benefits of DSS vs. these other methods? We will provide first answers to these questions in Section 3, where we validate DSS shock-graph based shape recognition on the Kimia-25 and Kimia-99 databases.

## 2. Skeleton Construction Algorithm

After the preprocessing step of approximating the contour of the given binary shape by a polygon, we can now describe the algorithm for generating the discrete straight skeleton. The algorithm is a hybrid algorithm which makes use of both continuous geometry and discrete geometry. Continuous geometry is used to find the next edge event, whereas discrete geometry is used to find the next split event. Finding the time stamp of the next split event is the most complex part in the continuous case. In the discrete case, we overcome this problem by only using discrete numbers for the time stamp. We show an outline of the discrete straight skeleton (DSS) generation for a polygon  $P$  in Algorithm 1.

We now describe each step in detail. We denote the vertices of  $P$  with  $v_i$  and its edges with  $e_i$ . The two edges incident to a vertex  $v_i$  are  $e_{i-1}$  and  $e_i$ . The resulting skeleton is

---

**Algorithm 1** DSS( $P$ )

---

Compute next (potential) edge event.  
**if**  $P$  would not intersect itself after the edge event **then**  
  Process edge event.  
   $P_{new} \leftarrow P$   
  Call DSS( $P_{new}$ ).  
**else**  
  Discard the edge event.  
  Compute next split event.  
  Split  $P$  into polygons  $P_1$  and  $P_2$ .  
  Call DSS( $P_1$ ).  
  Call DSS( $P_2$ ).  
**end if**

---

stored in a graph data structure  $T$  which is initialized with the leaves  $v_i$  and their corresponding bisectors  $[e_{i-1}, e_i]$ . The bisector at  $v_i$  is a (directed) halfline that results from bisecting the interior angle at  $v_i$ . For the description of the algorithm, we will also consider the *outer* bisector at  $v_i$ , that is, the halfline pointing to the exterior of  $P$  (and not contributing to the straight skeleton).

Computing the next edge event is a rather simple task. Let  $e$  be an edge of the polygon  $P$  and let the adjacent vertices of edge  $e$  be  $v_i$  and  $v_{i+1}$ . If the bisectors at vertices  $v_i$  and  $v_{i+1}$  intersect each other then they form a triangle with edge  $e$ . The height of this triangle at baseline  $e$  is the time stamp when the length of  $e$  shrinks to zero in the shrinking process. The next (potential) edge event for  $P$  is the minimum of these heights for all edges.

The next step is to check whether the polygon is intersecting itself after processing the edge event. The trivial way would be to check intersections between the edges themselves, but the number of tests is quadratic in the number of edges. With the following simple observation, we can define a criterion whether the polygon intersects itself by looking solely at the pixels on the contour of the pixel shape after shrinking.

**Observation 1.** *The shrunk polygon  $P$  is intersecting itself if and only if there exist pixels which are intersected more than once by the boundary of  $P$ .*

If the polygon is intersecting itself, then a split event actually occurs before the potential edge event. That is, the potential edge event is void. In this case, we use binary search in the interval  $[0, \text{time stamp of edge event}]$  to find the discrete time stamp for the split event, using Observation 1 repeatedly.

The following actions are carried out on the graph data structure  $T$  for the respective type of event:

1. In the case of an edge event, a new node, say  $u$ , is introduced in  $T$ , being adjacent to the two arcs which

correspond to the intersecting bisectors. A new arc is connected to  $u$ , living on the new bisector. More precisely, if  $e_i$  is the edge that vanishes, then the bisectors  $[e_{i-1}, e_i]$  and  $[e_i, e_{i+1}]$  intersect in  $u$ , and the new bisector is  $[e_{i-1}, e_{i+1}]$ .

2. For the split event, let us assume that the polygon with edges  $e_1, \dots, e_{k-1}, e_k, \dots, e_s$  is split by vertex  $v$  (with adjacent edges  $e_{k-1}$  and  $e_k$ ) at edge  $e_t$ . A new node,  $w$ , is introduced in  $T$ , and  $w$  is connected to the arc corresponding to the bisector  $[e_{k-1}, e_k]$ . The edges of the two resulting polygons are  $e_1, \dots, e_k$  and  $e_{k-1}, \dots, e_s, e_1$ , respectively. The new node  $w$  is connected to two arcs, corresponding to the bisectors  $[e_1, e_{k-1}]$  and  $[e_k, e_1]$ .

Let us now discuss the complexity of the algorithm. Concerning the edge events, we store each potential edge event (i.e., the time stamp for the corresponding two intersecting bisectors) in a priority queue of size  $M$  (or less), where  $M$  denotes the number of vertices of the polygon. The edge event with the earliest time stamp then can be identified repeatedly, and removed from the queue, in time  $O(\log M)$  each. Whether this very edge event is valid or void can be decided in time  $O(n)$ , by Observation 1, where  $n$  is the number of contour pixels of the input shape. Likewise, the next split event can be singled out in  $O(n \log d)$  time, by the binary search technique applied. Here  $d$  denotes the pixel diameter of the input shape, and we have  $d = O(n)$ . As there are  $O(M)$  events in total (the number of vertices of the straight skeleton is linear in  $M$ ), a worst-case runtime of  $O(M \cdot n \log n)$  results. Note that the  $\log M$  term from above is negligible because  $M < n$ .

When comparing to Subsection 1.2, we see that the total runtime of the discrete straight skeleton construction is dominated by the determination of the approximating polygon,  $P$ . In typical applications, the number of contour pixels,  $n$ , will be much smaller than the total number of pixels,  $N$ . Also, the number  $M$  of vertices of  $P$  will actually be a constant depending on the particular type of shape rather than on its number of pixels. So, when the approximating polygon is already available, the computation of the straight skeleton can be expected to run in time much less than the number of pixels of the given shape. In particular, the runtime will be linear in  $N$ , or less, unless the number of contour pixels is comparable in magnitude to the total number of pixels in the shape.

The algorithm is able to handle special cases, such as multiple edge events, an edge and a split event occurring at the same time, or so-called *vertex events* where two reflex vertices collide during the shrinking process. Whereas these cases are extremely unlikely in the continuous case, they do occur in the discrete case and have to be treated correctly.

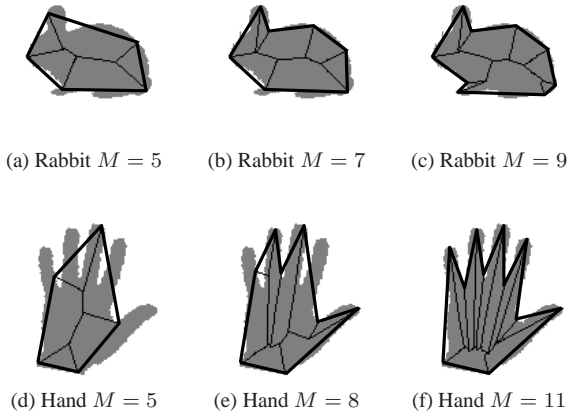


Figure 3: Discrete straight skeletons computed by our algorithm. Already in this crude form, the discrete straight skeleton turns out to be a suitable descriptor of shape.

We illustrate a few typical results of our DSS algorithm on the polygonal approximations from Figure 2. Figure 3 shows the polygons and the resulting straight skeletons for these two shapes.

### 3. Experiments

As a practical example, we applied the DSS to the well-known shock graph framework in shape matching [20, 24]. Note that during the generation of the skeleton we also get auxiliary information such as event type and time stamp and location of the event, which can be stored in the skeleton arcs and nodes. This attributed graph resembles the shock graph. The edge event nodes where an entire (sub)polygon shrinks to a point correspond to local maxima during the shrinking process, and therefore are similar to the shock type four, which is a point of maximum distance to the boundary of all neighboring medial axis points. The split event node corresponds to the shock type two, which is the center of a locally minimum circle. Straight skeleton arcs are bounded by two nodes each, and therefore store two (different) time stamps and the length of the branch. Arcs resemble the shock type one (if the two time stamps do not differ ‘much’). Finally, a dummy root node is introduced, which is connected to all edge event nodes.

There are a number of approaches for shock graph matching. One of them is to find a maximum clique in the association graph of the two graphs (see [16] for details). Another approach is to find an edit sequence for changing one shock graph into the other. The number of the edit operations serves as a similarity measure. The

edit operations delete or insert a node or an edge and re-label a node in a consistent manner. See [20] for details. Torsello [26] presented another graph matching technique based on editing shock graphs, and also proposed a graph matching algorithm based on graph clustering and embedding. The approach of the initial paper [24] on shock graph matching is based on the largest subgraph problem. An implementation of the latter by Macrini *et al.* can be found at <http://www.cs.toronto.edu/~dmac/ShapeMatcher/>. For our own better understanding of this implementation, and for more flexibility in parameter tuning, we also re-implemented this approach. In Tables 1 and 2, line ‘Macrini’s implementation’ refers to Macrini *et al.*, using their default parameters, and line ‘our re-implementation’ refers to our own variant of the algorithm including parameter tuning.

Although it is unclear which of these approaches is most suitable for ‘shock graphs’ that stem from the DSS, we combined the DSS with the shock graph matching based on the largest subgraph problem. As mentioned before, a crucial part of the DSS computation is the choice of vertices of the approximating polygon. As a first experiment, we chose this number to be fixed for all shapes and tested it on the Kimia databases. The Kimia-25 database consists of 25 images showing six objects, five images of one class and four images each for the other five classes. The Kimia-99 database contains 99 images, eleven for each of the nine objects. These databases have been tested by other shape categorization approaches [5, 6, 7, 12, 15, 20, 21, 27]. The standard test by all these approaches was to compute a similarity matrix between all images and take a look whether or not the  $k$ -nearest neighbor (with  $k$ -highest similarity value) of a given image belongs to the same class as the given image.

Table 1 and Table 2 present our results for the DSS with a fixed number of five and nine polygon vertices, respectively, for all shapes in the Kimia-25 database and the Kimia-99 database. The score in each cell of the table means that, out of 25 (or 99) images, this many  $k$ -nearest neighbors are put in the same class as the class of the given image. Additionally, the results of other approaches, and of Macrini’s shock graph implementation and our re-implementation, are included in the tables. The DSS with 5 vertices shows surprisingly good results on the Kimia-25 database. The most misses occur for the two classes ‘airplanes’ and ‘hands’ in the database, which is reasonable because the shapes of these classes cannot be convincingly approximated by a polygon with only five vertices. Table 3 shows the confusion matrix for this case. Class 3 (C 3) represents airplanes and Class 6 (C 6) represents hands. Increasing the (fixed) number of polygon vertices to nine leads to a slight improvement for the class of hands, but the results for all the

Table 1: Results for the  $k$ -nearest neighbors of the Kimia-25 database

Algorithm	k=1	k=2	k=3
Sharvit <i>et al.</i> [21]	23	21	20
Gdalyahu and Weinshall [12]	25	21	19
Belongie <i>et al.</i> [6]	25	24	22
Ling and Jacobs [15]	25	24	25
Macrini's Implementation	22	20	13
Our Re-Implementation	24	20	18
DSS (5 vertices)	23	20	14
DSS (9 vertices)	16	13	11
DSS (optimal $M$ )	24	22	18

other classes drop (see the confusion matrix in Table 4). A further increment on the number of vertices did not change the results significantly.

These initial results demonstrate clearly, that the DSS itself, by straightforward application of Macrini's shock graph, can achieve comparable recognition results as shock-graphs based on medial axes. However, choosing the best value of  $M$  for a particular query image is far from trivial. This can be seen from the rather poor results on Kimia-99, where the variability in shapes is even higher than for Kimia-25; see the third-last and second-last line (for fixed  $M = 5$  and  $M = 9$ ) in Table 2.

Our results on Kimia-25 show that we can approximate these shapes quite well by polygons with  $5 \leq M \leq 10$  points, and that the optimal value of  $M$  is class-specific. To find out, how much could be gained by automatic adaptation of  $M$  in the future, we performed the following experiment. We ran our test on Kimia-25 and Kimia-99 through all values of  $M = 5, 6, \dots, 10$  and selected for each class the value  $M$  that yielded the best result for this particular class. The results of these tests are shown in Table 5 and Table 6, and are also given in the last lines of Table 1 and Table 2 ('DSS (optimal  $M$ )').

Even more striking is the gain from class-specific adaptation in  $M$  for Kimia-99 where results, being well comparable to existing approaches again, are obtained this way for small approximating polygons. This demonstrates that  $M$  might be kept small but should be adapted to the complexity of the query shape at hand, which is a topic of our ongoing research.

## 4. Conclusion

We see two major contributions of this paper. First, we have presented the concept of a digital straight skeleton (DSS), previously mostly known in Computational Geometry, to the Computer Vision Community. We also have

Table 3: Confusion matrix for five vertices ( $M = 5$ ) on Kimia-25

	C 1	C 2	C 3	C 4	C 5	C 6
Class 1	11	0	0	0	0	1
Class 2	0	11	0	0	1	0
Class 3	1	0	7	0	0	4
Class 4	0	0	0	12	0	0
Class 5	0	1	0	0	11	0
Class 6	3	0	3	4	0	5

Table 4: Confusion matrix for nine vertices ( $M = 9$ ) on Kimia-25

	C 1	C 2	C 3	C 4	C 5	C 6
Class 1	2	0	0	7	0	3
Class 2	0	10	0	1	1	0
Class 3	3	0	5	4	0	0
Class 4	5	0	0	4	0	3
Class 5	0	1	0	0	11	0
Class 6	5	0	0	2	0	8

Table 5: Confusion matrix for the optimal number of vertices  $M_{opt}$  on Kimia-25

	$M_{opt}$	C 1	C 2	C 3	C 4	C 5	C 6
Class 1	5	11	0	0	1	0	0
Class 2	5	0	11	0	0	1	0
Class 3	5	3	0	8	1	0	0
Class 4	5	0	0	0	12	0	0
Class 5	5	0	1	0	0	11	0
Class 6	10	2	0	2	0	0	11

devised a novel algorithm to compute the DSS. Our algorithm is computationally efficient, and is especially suited to calculate the DSS from small polygons. In various cases, DSS will present an alternative to classical medial axis algorithms in Computer Vision, where the computation of costly distance transforms is needed. Our algorithm avoids distance transform calculations, and provides a geometric description of the skeleton. Second, we have performed a first experimental validation and comparison of DSS with medial axis on the Kimia shape databases. We obtain surprisingly good results, already for very small polygon sizes  $M$ . This demonstrates that DSS is a valid representational concept for shape representation and shape-based recognition in Computer Vision.

Our ongoing research is focused on improving the pre-processing step, by automatically choosing the optimal number of polygon vertices,  $M$ . In the end, this should lead to a single, integrated algorithm that computes a DSS directly from a binary input shape, adapting  $M$  depending on

Table 2: Results for the  $k$ -nearest neighbors of the Kimia-99 database

Algorithm	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
Belongie and Malik [5]	97	91	88	85	84	77	75	66	56	37
Tu and Yuille [27]	99	97	99	98	96	96	94	83	75	48
Kimia <i>et al.</i> [20]	99	99	99	98	98	97	96	95	93	82
Ling and Jacobs [15]	99	99	99	98	98	97	97	98	94	79
Macrini's Implementation	95	86	82	84	76	70	60	54	46	36
Our Re-Implementation	93	90	82	78	69	64	59	59	49	40
DSS (5 vertices)	72	68	57	59	54	49	44	31	31	23
DSS (9 vertices)	78	64	61	52	50	45	42	46	32	26
DSS (optimal $M$ )	94	93	86	86	79	76	74	49	48	39

Table 6: Confusion matrix for the optimal number of vertices  $M_{opt}$  on Kimia-99

	$M_{opt}$	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9
Class 1	9	63	0	18	0	6	21	2	0	0
Class 2	5	0	103	0	7	0	0	0	0	0
Class 3	7	4	5	79	0	3	5	14	0	0
Class 4	5	0	35	0	74	1	0	0	0	0
Class 5	5	0	20	10	1	76	0	3	0	0
Class 6	9	24	0	18	0	0	67	1	0	0
Class 7	7	2	0	35	1	2	1	60	9	0
Class 8	6	0	0	0	7	0	0	1	102	0
Class 9	8	3	0	3	0	3	0	1	0	100

the complexity of the shape.

In our future research, we plan to systematically investigate which kind of shock graph algorithm is best suited for DSS. Finally, we would like to apply DSS-based shock graphs for applications in shape-based object category recognition.

## Acknowledgements

This work was supported by the Austrian Science Fund (FWF) under the doctoral program Confluence of Vision and Graphics W1209. We wish to thank Wolfgang Aigner for discussions on the algorithm in Section 2.

## References

- [1] O. Aichholzer, W. Aigner, F. Aurenhammer, T. Hackl, B. Juettler, and M. Rabl. Medial axis computation for planar free-form shapes. *Computer Aided Design*, 41:339–349, 2009.
- [2] O. Aichholzer and F. Aurenhammer. Straight skeletons for general polygonal figures in the plane. In *COCOON '96: Proceedings of the Second Annual International Conference on Computing and Combinatorics*, pages 117–126, London, UK, 1996. Springer-Verlag.
- [3] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gaertner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.
- [4] G. Baraquet, D. Eppstein, M. Goodrich, and A. Vaxman. Straight skeletons of three-dimensional polyhedra. In *Proc. 16th European Symp. Algorithms*, pages 148–160. Springer LNCS 5193, 2008.
- [5] S. Belongie and J. Malik. Matching with shape contexts. In *CBAIVL '00: Proceedings of the IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'00)*, page 20, Washington, DC, USA, 2000. IEEE Computer Society.
- [6] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.
- [7] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, and R. Kimmel. Partial similarity of objects, or how to compare a centaur to a horse. *International Journal of Computer Vision*, 84:163–183, 2009.
- [8] S.-W. Cheng and A. Vigneron. Motorcycle graphs and straight skeletons. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 156–165, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.
- [9] D. Coeurjolly and A. Montanvert. Optimal separable algorithms to compute the reverse Euclidean distance transformation and discrete medial axis in arbitrary dimensions.

- IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(3):437–448, 2007.
- [10] R. Dinesh, S. S. Damle, and D. S. Guru. A split-based method for polygonal approximation of shape curves. In *PReMI*, pages 382–387, 2005.
- [11] D. Eppstein and J. Erickson. Raising roofs, crashing cycles, and playing pool: applications of a data structure for finding pairwise interactions. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry*, pages 58–67, New York, NY, USA, 1998. ACM.
- [12] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1312–1328, 1999.
- [13] J.-H. Horng and J. T. Li. An automatic and efficient dynamic programming algorithm for polygonal approximation of digital curves. *Pattern Recogn. Lett.*, 23(1-3):171–182, 2002.
- [14] R. Kimmel, D. Shaked, N. Kiryati, and A. M. Bruckstein. Skeletonization via distance maps and level sets. *Computer Vision and Image Understanding*, 63:382–391, 1995.
- [15] H. Ling and D. Jacobs. Using the inner-distance for classification of articulated shapes. In *CVPR05*, pages II: 719–726, 2005.
- [16] M. Pelillo, K. Siddiqi, and S. W. Zucker. Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1105–1120, 1999.
- [17] J.-C. Perez and E. Vidal. Optimum polygonal approximation of digitized curves. *Pattern Recogn. Lett.*, 15(8):743–750, 1994.
- [18] B. K. Ray and K. S. Ray. A new split-and-merge technique for polygonal approximation of chain coded curves. *Pattern Recognition Letters*, 16(2):161 – 169, 1995.
- [19] E. Remy and E. Thiel. Exact medial axis with Euclidean distance. *Image and Vision Computing*, 23:167–175, 2005.
- [20] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):550–571, 2004.
- [21] D. Sharvit, J. Chan, H. Tek, and B. B. Kimia. Symmetry-based indexing of image databases. *Journal of Visual Communication and Image Representation*, 9(4):366–380, December 1998.
- [22] F. Shih and C. Pu. Medial axis transformation with single-pixel and connectivity preservation using Euclidean distance computation. In *Proc. 10th Int. Conf. Pattern recognition*, pages 723–725, 1990.
- [23] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker. Hamilton-Jacobi skeletons. *International Journal of Computer Vision*, 48(3):215–231, 2002.
- [24] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. In *ICCV*, pages 222–229, 1998.
- [25] A. Telea and J. J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*, pages 251–ff, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.
- [26] A. Torsello. *Matching Hierarchical Structures for Shape Recognition*. PhD thesis, University of York, 2004.
- [27] Z. Tu and A. L. Yuille. Shape matching and recognition - using generative models and informative features. In *ECCV (III)*, pages 195–209, 2004.
- [28] M. Tănase and R. Veltkamp. Polygon decomposition based on the straight skeleton. In *Proc. 19th ACM Symp. Computational Geometry*, pages 58–67, 2003.
- [29] M. Tănase and R. Veltkamp. A straight skeleton approximating the medial axis. In *Proc. 12th European Symp. Algorithms*, pages 809–821. Springer LNCS 3221, 2004.
- [30] W.-Y. Wu. An adaptive method for detecting dominant points. *Pattern Recognition*, 36(10):2231 – 2237, 2003.
- [31] Y. Xiao, J. J. Zou, and H. Yan. An adaptive split-and-merge method for binary image contour data compression. *Pattern Recognition Letters*, 22(3-4):299 – 307, 2001.