Grouping of Isolated Non-directional Cues with Straight Offset Polygons

Toshiro Kubota
Mathematical Sciences, Susquehanna University, Selinsgrove PA, U.S.A.

Keywords: Dot Patterns, Shape Extraction, Clustering.

Abstract: When the boundary of a familiar object is shown by a series of isolated dots, humans can often recognize the object with ease. This ability can be sustained with addition of distracting dots around the object. However, such capability has not been reproduced algorithmically on computers. In this paper, we will introduce a new algorithm that groups a set of dots into multiple overlapping subsets. It first connects the dots into a spanning tree using the proximity cue. It then applies the straight polygon transformation to an initial polygon derived from the spanning tree. The straight polygon divides the space into polygons recursively and each polygon can be viewed as grouping of a subset of the dots. The number of polygons generated is O(n). We used both natural and synthetic images to test the performance of the algorithm. The results are encouraging.

1 INTRODUCTION

Consider a picture shown in Figure 1(a). We can easily recognize a dolphin in the picture, delineated by a series of isolated dots. Reproducing this capability on a computer is not difficult for this simple case. The task quickly becomes difficult when distracting noisy dots are added to the picture as shown in Figure 1(b) and (c). We can still recognize the dolphin in (b) and possibly (c). However, most clustering algorithms are not capable of dealing with them as the background noise significantly overlaps with the shape. Most perceptual organization algorithms also have difficulties in dealing with such data, as dots lack any orientation and directional information needed for many algorithms(Elder and Zucker, 1996; Mahamud et al., 2003; Wang et al., 2005).

Patterns comprised of isolated dots (often called *dot patterns*) have played important roles in various psycho-visual studies. The studies suggest bottom-up grouping and shape extraction capability in our visual system (Kubovy and Wagemans, 1995; Greene, 2008). Although they are not typical patterns we encounter in every day experience, they carry raw information that is essential to our visual processing. Thus, it is important to understand how such patterns can be processed algorithmically and how an underlying salient shape can be extracted. Various attempts have been made in the past to divide a dot pattern into coherent sets (clustering problem) (Zahn, 1971; Rosenberg and Langridge, 1973; Ahuja, 1982; Toussaint, 1980) or delineate the boundary enclosing the pattern (external shape problem)(Edelsbrunner et al., 1983; O'Rourke et al., 1987; Chaudhuri et al., 1997; Melkemi and Djebali, 2000). However, the past approaches were unable to deal with overlapping distracters as seen in Figure 1(b), highly dependent on critical parameters(Ahuja and Tuceryan, 1989; Edelsbrunner et al., 1983), and limited to a single interpretation of data(Rosenberg and Langridge, 1973).

In this paper, we propose an algorithm that derives a collection of polygonal regions from a dot pattern in a parameter free manner. It applies minimum spanning tree on the pattern followed by straight polygon transformation of (Aichholzer et al., 1995). See Figure 2 for an illustration of the idea. Figure 2(a) is a simple example of a point pattern. Figure 2(b) shows a minimum spanning tree derived from (a).As in (Zahn, 1971), the Euclidean distance between a pair of dots is used as the weight for the spanning tree. Figure 2(c) shows a straight polygon representation derived from (b). As the polygon deforms outward, a vertex of the polygon touches another part of the polygon. At the time, a new polygon is created by an enclosure of the outward growing polygon. This newly created polygon grows inward as the deformation is contained within the enclosure. Three inward growing polygons are created in the example, which are shown in thick solid lines. An inward growing polygon can further divided into multiple polygons if a concave vertex touches another side of the polygon before it vanishes. As we outline in more detail later in this paper, we can trace back vertices of a new polygon back to vertices of the spanning tree. The result-

Kubota T..
Grouping of Isolated Non-directional Cues with Straight Offset Polygons.
DOI: 10.5220/0005257003410348
In *Proceedings of the 10th International Conference on Computer Vision Theory and Applications* (VISAPP-2015), pages 341-348
ISBN: 978-989-758-089-5

ing set of vertices in the spanning tree forms another polygon using the original point set and provides a grouping instance. Therefore, for each inward growing polygon, we can associate a grouping instance of the point set.

The outward growing polygon keeps growing and encloses these inward growing polygons. We can also trace vertices of the outward growing polygon back to the spanning tree. The trace may not form a polygon as it can visit the same edge more than once (in opposite directions). For example, for the polygon pointed by an arrow in Figure 2(c), the trace provides two polygons joined by an edge that were traversed twice in both directions. One of the two polygons encloses the two rectangles in the upper part and the other encloses the polygon with five vertices in the lower part. Thus, a trace of outward growing polygon can potentially provides multiple grouping instances where each grouping can possibly combine multiple inward growing polygons.

In summary, the straight polygon transformation applied to a minimum spanning tree of a point pattern can provide multiple grouping hypotheses in a simple deterministic manner without any parameters. Our experiments presented in this paper show that the approach offers grouping performance that is robust against noise and grouping hypotheses that are agreeable to our perception. The approach complies with multiple interpretations and multiple solutions characteristics advocated in (Engbers and Smeulders, 2003) for a good grouping algorithm.

The rest of the paper is organized as follows. Section 2 describes the straight polygon deformation and straight skeleton representation. Section 3 formalize our grouping approach as outlined above. Section 4 provides empirical evaluation results. Section 5 discusses strength and limitation of the current approach and provides future directions. Section 6 concludes the paper with a brief summary.

2 STRAIGHT OFFSET POLYGONS AND STRAIGHT AXES

In this section, we describe straight offset polygon and straight skeleton representations as introduced in (Aichholzer et al., 1995) and extended in (Aichholzer and Aurenhammer, 1996). Let P be a polygon with n vertices. The process of forming the straight offset polygon representation is to shrink the polygon by moving inward each side of the polygon by self-parallel motion. Such motion can be generated by

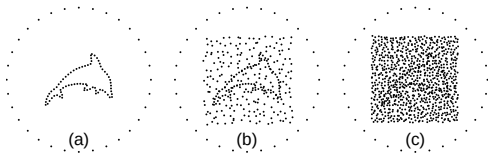


Figure 1: Dot patterns.

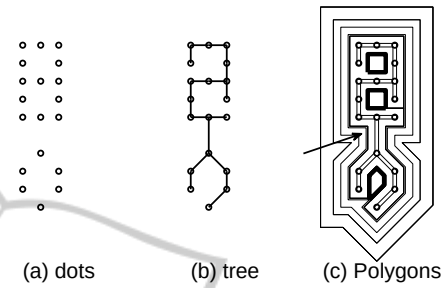


Figure 2: An example of straight polygon representation.

moving each vertex to the direction of the angle bisector with the velocity given by

$$v_i \propto 1/\sin(\theta_i/2) \quad (1)$$

where v_i is the velocity of the ith vertex and θ_i is the angle of the polygon at the ith vertex. Two events can change the shape and topology of the polygon: edge event and split event. An edge event occurs when two adjacent vertices collide and changes the shape of the polygon. A split event occurs when a concave (or reflexive) vertex collide with a side of the polygon and split the polygon into two.

See Figure 3 for an illustration of these events. It shows a polygon with 8 vertices represented by small hollow circles. The polygon undergoes shrinkage by self-parallel motion, and snapshots of the shrinkage are shown with dashed lines. First, a split event takes place when the vertex pointed by an arrow collides with a side of the polygon. The event splits the polygon into two polygons: one with 5 vertices (left polygon) and the other with 4 vertices (right polygon). These new polygons undergo the same shrinkage process independently. Each polygon experiences a number of edge events where two adjacent vertices collide. There are 3 edge events in both left and right polygons. The shrinkage stops when the polygon vanishes to a line or a point. In Figure 3, polygons at split or edge events are shown in solid lines with the exact locations of events shown with filled circles.

By tracing vertices of offset polygons, we obtain a tree-structure, which is called *straight skeletons* of the polygon. The skeletons are shown in thick solid lines in Figure 3. Without any degenerate cases where more than two points collide simultaneously or two parallel sides collide, there are $n-2$ non-leaf nodes in

the straight skeletons and $2n-3$ arcs. The skeletons also divide the original polygon into n faces.

Instead of shrinkage, we can consider expansion of the polygon by self-parallel motion of each side. This can be achieved by simply reversing the motion of each vertex. The polygon deforms and its shape and topology change as edge and split events occur as in the shrinkage case. Some vertices do not vanish and approach infinity.

As the name suggests, the straight skeletons are comprised of linear line segments. In contrast, the medial axes are comprised of linear and quadratic line segments in general(Blum, 1967). Thus, the data structure for the straight skeletons is simpler than that of the medial axes. However, the straight skeletons lack a dual interpretation of Voronoi diagram as in the medial axes. As a result, we need to simulate the shrinkage (or expansion) process to obtain the straight skeletons. The run-time of a brute force implementation of the simulation is in $O(n^3)$ where n is the number of vertices, but in practice, it runs in $O(n^2)$. The medial axes can be constructed in $O(n\log n)$ (Lee, 1982). The number of events is upper bounded by $n-2$. At each event, at most two polygons are created. Thus, the number of polygons is upper bounded by $2n-4$. Algorithm 1 shows procedures that implement the construction of the straight offset polygons. The initialization (Lines 2-4) takes $O(n^2)$ since the for-loop runs over vertices, and for each vertex, we need to examine other vertices and sides for the earliest incidence of events. The recursion (*apply*) is called for each event, which is $O(n)$. Handling an event (Lines 11-17) takes a constant time. Some events in the queue need to be updated if the current event removed the participating vertex or side for the events (Lines 18-21). The number of events that need to be updated is $O(n)$ but typically is dependent on local shape and thus is $O(1)$. For each event, it takes $O(n)$ to update. Therefore, the *apply* procedure runs in $O(n^2)$ for the worst case but $O(n)$ for typical cases. Overall, the algorithm runs in $O(n^3)$ for the worst case but $O(n^2)$ for typical cases. In this paper, we call straight offset polygons and straight skeletons combined *straight polygon representation* and the process of computing the straight polygon representation *straight polygon transformation*.

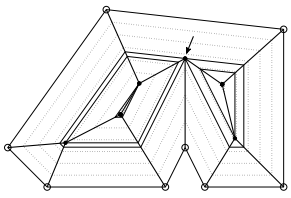


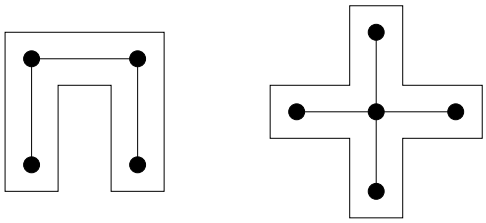
Figure 3: An example of straight offset polygons.

```
1  Procedure OffsetPolygons(P)
       Input: P: initial polygon
       Output: G: a set of offset polygons
2      foreach vertex q in P do
3          find the next event for q
4      end
5      G = apply(P)
6      return G
7  Procedure apply(P)
       Input: P: a polygon
       Output: G: a set of polygons
8      G← {P}
9      if |P| ≥ 3 then
10         extract the next event from P
11         if edge event then
12             L = polygon after collision
13             R=∅
14         end
15         else if split event then
16             L,R = polygons after split
17         end
18         Q = a set of vertices affected by the
           event
19         foreach q in Q do
20             update the next event for q
21         end
22         G ← G + apply(L) + apply(R)
23     end
24     return G
```

Algorithm 1: Procedure for straight polygon transformation.

3 ALGORITHM

This section details an algorithm of computing a collection of grouping of dots from a dot pattern. Algorithm 2 shows an outline of the algorithm. First a minimum spanning tree is constructed from the input dot pattern (Line 1). The tree is traced to construct a circularly linked list of polygon vertices with the velocity given by (1) (Line 2). Figure 4 shows examples of the trace. When there is no branch in the spanning tree (a), the trace gives a polygon of $2n$ vertices where n is the number of nodes in the spanning tree. When one tree node is adjacent to every other node in the tree (b), the trace gives a polygon of $3n-3$ vertices. The number of polygon vertices are bounded by these two extreme cases. The polygon does not need any particular width as long as all vertices are prepared with the velocity defined by (1). From the initial polygon, the straight polygon transformation is applied with Algorithm 1 (Line



(a) (b)

Figure 4: Two examples of a spanning tree and an initial polygon.

3). For each moving vertex, additional attributes are added as described next.

From each offset polygon, we want to trace back to a subset of vertices in the initial polygon and in turn to the nodes in the spanning tree. For the purpose, each vertex maintains two links (π_a and π_b). For vertices in the initial polygon, $\pi_a = \pi_b = nil$. Upon an edge event, two vertices collide and a new vertex is formed. π_a and π_b of the new vertex are set to the colliding vertices in such a way that π_b is the successor of π_a in the circular linked list. By preserving this ordering consistently, the set of vertices after tracing back forms a polygon. Upon a split event, a vertex collides with a side of the polygon and two new vertices are formed. For both of the new vertices, π_a is set to the colliding vertex and π_b is set to nil. Using π_a and π_b, we can associate the sequence of vertices in an offset polygon back to a sequence of vertices in the initial polygon (Line 5).

As illustrated in Section 1 with Figure 2, a sequence of vertices obtained in Line 5 may not form a polygon. Thus, we trace the sequence and decompose it into a set of subsequences where each form a separate polygon (Line 6).

Let $m \in [2n, 3n-3]$ be the number of vertices in the initial polygon. Note that offset polygons by successive edge events trace back to the same set of vertices in the initial polygon. Thus, they do not provide any new grouping hypotheses. There can be at most $m/2$ split events. This case occurs if polygons of 3 vertices are split successively without any edge events. Each provides a distinct grouping hypothesis. These split polygons can be aggregated by the outgrowing polygon. The aggregation is done hierarchically. Thus, there can be at most $m/2$ aggregation instances. Therefore, there can be at most m grouping hypotheses in total.

Input: **Z**: a set of isolated dots
Output: **H**: a set of polygons
1 **T** ← minimum spanning tree of **Z**
2 **P** ← a thin polygon tracing around **T**
3 **G** ← OffsetPolygons(**P**)
4 **foreach** ***g*** *in* ***G*** **do**
5 | **h** = trace **g** to nodes in **T**
6 | {**J**} ← polygonal regions in **h**
7 | add each element of {**J**} to **H**
8 **end**
9 **return** ***H***

Algorithm 2: Procedure for dot grouping.

4 EXPERIMENTS

4.1 Experiment 1

Our first experiment is to apply Algorithm 2 to some point patterns derived from real images. To generate a point pattern of an image, we first apply Canny's edge detector to the image. We then sub-sample edge pixels in each 4×4 block by placing a point at the centroid (rounded to the pixel) of edge pixels in the block. The subsampling reduces the proximity cue and makes the grouping process more difficult. Figure 5 shows an example of these pre-processing steps where (a) is a gray scale image, (b) is a Canny edge image, (c) is a dot pattern, and (d) is a minimum spanning tree derived from (c).

The grouping algorithm produces a number of polygons comparable to the number of points in the dot pattern. For example, there are 403 points in Figure 5(c) and there are 486 polygons as the result of the grouping algorithm. Just to present a few polygons that are deemed representative, we use the following procedure to consolidate similar polygons and rank them.

For each polygon, the following convexity measure is computed.

$$c(P) = \frac{\text{area}(P)}{\sum_i \|p_i - p_{i+1}\|^2} \tag{2}$$

where $P = \{p_1, p_2, \cdots, p_K\}$ is a polygon with K vertices, area(P) is the area of the polygon, and the summation in the denominator runs in modulo-K so that it computes the sum of polygon sides squared. The measure is unit-less and prefers a polygon whose shape is near circular and whose vertices are evenly distributed around the shape. We then perform connected component analysis on the set of polygons by using the following overlap measure as a similarity

measure.

$$\xi(P,Q) = \frac{\text{area}(P \cap Q)}{\text{area}(P \cup Q)} \quad (3)$$

where P and Q are two polygons being compared. $\xi(P,Q) = 0$ if there is no overlap and 1 if they are identical. P and Q are merged into the same component if $\xi(P,Q) \geq 0.5$. For each connected component, a polygon with the highest convexity measure is retained as a representative of the component.

Figure 6 shows results of the above consolidation procedure. In each row, from left to right, the first two images are the input natural image and the Canny edge image, respectively. The next six images show six representative polygons with the highest convexity measures. Polygons are drawn in thick black lines on top of the dot patterns in gray.

Note that this experiment is not to provide a full scale object extraction algorithm, which will be left for future research. But it is to summarize polygons derived from the algorithm in a concise and informative way. The selection procedure described above is crude and requires further investigation.

4.2 Experiment 2

Our second experiment is to apply the grouping algorithm to point patterns comprised of a shape and superimposed noise, and examine if the representation recovers a polygon that closely matches with the underlying shape. Thus, the experiment investigate the robustness of the representation against random dots. Figure 1 shows instances of such point patterns. There are 20 shapes of animals and common objects. Each shape was generated by tracing the boundary of its binary image, keeping every 10th point while discarding the others, and scaling them so that the shape stretched around $200 \leq x \leq 800$ and $200 \leq y \leq 800$ in the pixel coordinate. For each shape, we imposed three levels of noise, which were generated in the following way. The average spacing between adjacent points in the shape was computed. Denote the average μ. Then the area between $200 \leq x \leq 800$ and $200 \leq y \leq 800$ was divided into grids of $s\mu$ by $s\mu$ where s controlled the noise level and had a value of 1, 1.5 or 2. Within each grid, a point was placed randomly while keeping clear of 10% margin around the four border (so that no pair of noise points get too close with each other). Thus, $s = 1$ gives the highest amount of noise, and $s = 2$ gives the least amount of noise. Finally, 32 evenly spaced dots were placed around a large circle centered at $(500, 500)$ with the radius of 490. This circular pattern was intended to provide a frame of reference to human subjects in our third experiment as described below. Algorithm 2 was applied to each dot pattern. Then, a polygon with the highest overlap measure with the underlying shape as defined in (3) was selected.

Figures 7 and 8 show dot patterns used in the experiment and the most closely matched polygon in terms of (3) in the representation. In the figures, each row corresponds to one of 20 shapes. The first four columns are dot patterns used for the experiment. From left to right, they are the shape without any noise, one with minimum amount of noise ($s = 2$), one with medium amount of noise ($s = 1.5$), and one with maximum amount of noise ($s = 1$). The next four columns show the extracted polygon for each dot pattern in the first four column. The number shown in each figure is the overlap measure of (3). The average overlap measures among 20 shapes for different noise levels are 0.93, 0.87, 0.80, and 0.52 for zero, minimum, medium, and maximum noise levels, respectively. Thus, the accuracy of the representation degrade slowly as the noise level increase. At the maximum noise level, some shapes are not perceivable even for humans.

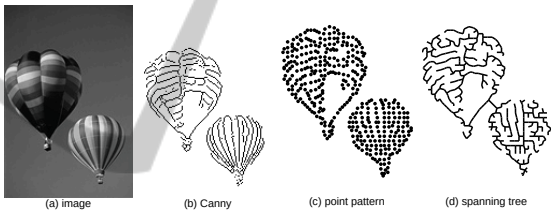


Figure 5.

4.3 Experiment 3

Our final experiment is to investigate how the performance of the straight polygon transformation based shape extraction described above correlates with the human perception. We recruited 20 volunteers on the campus of University. Each subject was given a set of four dot patterns and asked to delineate a salient shape in the pattern and name the object. We manually selected 16 dot patterns that have diverse range of 'easy' to 'difficult' ones. Each dot pattern was viewed by exactly five subjects. We calculated the proportion of subjects who were able to recognize the underlying object or delineated the shape accurately even though failed to name the right object. The results are given in Table 1. The first column gives the shape in the pattern, the second column gives the noise level, the third level gives the overlap measure of (3), and the last column gives the recognition rate by the humans. The overlap measure and the recognition rate are positively correlated with the Pearson coefficient of 0.54 (p=0.031).

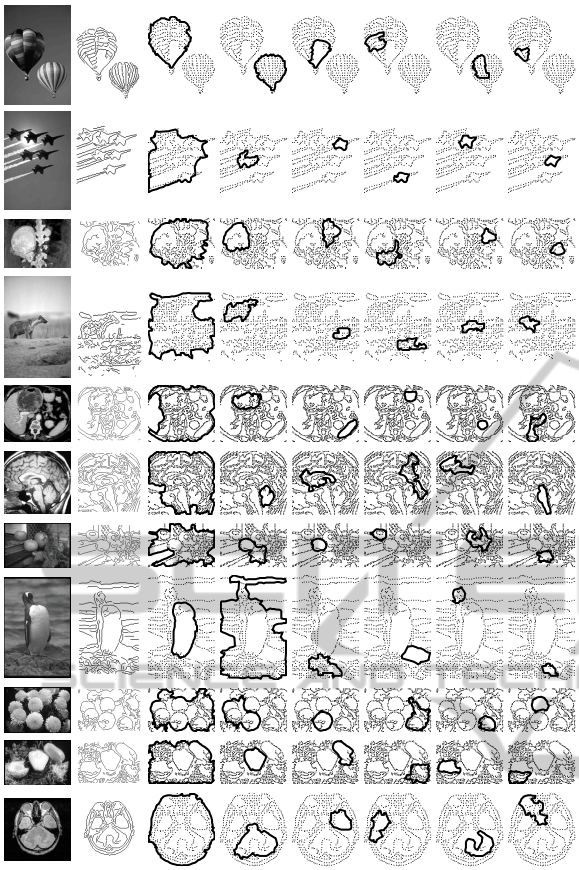


Figure 6: Results of applying the grouping algorithm to natural images. From left to right: input image, Canny edges, six grouping instances after consolidation with the highest scores as defined in 2.

Figure 7: Test patterns (first four columns) and grouping results (last four columns) I.

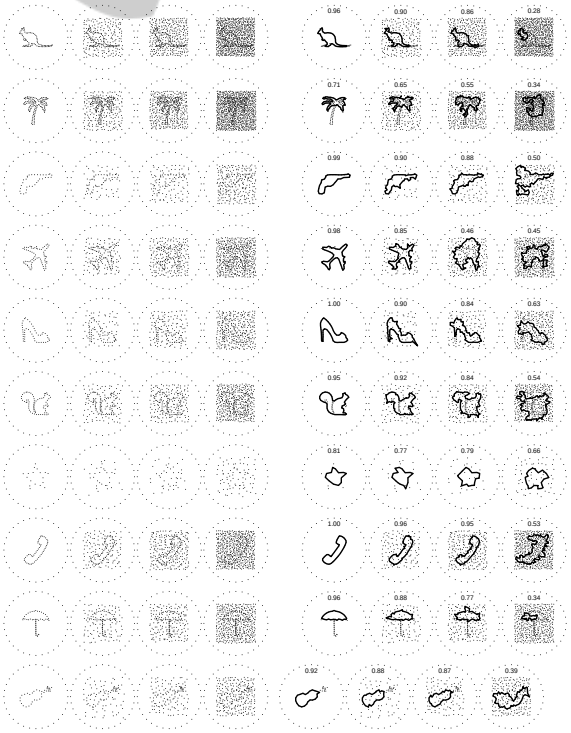


Figure 8: Test patterns (first four columns) and grouping results (last four columns) II.

5 DISCUSSION

As our experiments show, the straight polygon representation captures salient shape contained in a dot pattern. We found strong correlation between presence of a shape in the representation and detectability of the shape by human observers. Thus, the representation possesses a level of robustness against random noise comparable to that of the human vision.

We attribute the effectiveness of the straight polygon representation to the following three characteristics. First, events are at discrete points in time and space. In comparison, the medial axis transform undergoes events at continuous points in time and space when a circular front collides with another front. This characteristic makes a simple recursive implementation and straightforward organization of the events. Second, simple tree type data structure can be constructed to fully capture the evolution of polygons. It can then be used to reverse the deformation and trace

Table 1: Performance measures on the algorithm and humans. The overlap column shows the maximum overlap measure as defined on (3) among polygons derived by the algorithm. The recognition column shows the proportion of five subjects who could recognize the shape. The two measures are positively correlated with the Pearson correlation coefficient of $0.54 (p = 0.031)$.

shape	noise	overlap	recognition
Arrow	Min	0.917	4/5
Camel	Mid	0.776	4/5
Palm Tree	Max	0.343	1/5
Pistol	Min	0.899	2/5
Anchor	Max	0.385	0/5
Balloon	Min	0.959	4/5
Cherries	Mid	0.738	5/5
Star	Mid	0.786	1/5
Car	Min	0.974	4/5
Dolphin	Max	0.762	1/5
Fish	Mid	0.845	2/5
Umbrella	Mid	0.774	5/5
Duck	Min	0.951	5/5
Horse	Mid	0.641	5/5
Shoe	Min	0.904	5/5
Violin	Max	0.392	0/5

a vertex of a polygon to a point in the dot pattern. Third, the curvature dependent smoothing characterized by (1) finds grouping quickly with less amount of dilation than the medial axis transform. See Figure 9 for an illustration of comparing the straight skeleton transform and medial axis transform. The thick solid line is an initial minimum spanning tree where the transformation starts. The thick dashed line shows the front of the straight polygon transform when the first split event took place. The thin solid line shows the front of the medial axis transform. Only locations where the fronts are vastly different between the two are shown.

We are still at an early stage of developing the straight polygon representation into one that is more applicable to other computer vision tasks. Here are some of tasks that are left for our future endeavor. The representation captures salient patterns, but at the same time, keeps many non-salient ones. Since the the number of groupings is $O(n)$, it is feasible to examine each grouping more carefully than the simple evaluation of Experiment 1 using the convexity measure of (2). We need to investigate how to quantify the saliency of grouping and use it to filter non-salient ones. Supervised learning may be able to assist this task.

The initial polygon that is derived from a minimum spanning tree is rather artificial and, in many cases, does not reflect our perception. A better approach is to use more perceptually amiable initial grouping. To build an initial grouping, we can employ the dot sequencing algorithm of (Rosenberg and Langridge, 1973), for example. Most likely, there will be multiple disjoint polygons at the beginning of the straight polygon transform. Then, the algorithm needs to consider split events with which two disjoint polygons are merged into one and handle them slightly differently from events with which a single polygon is split into two.

By allowing multiple polygons at the beginning, the algorithm can easily extended to edge grouping. Each edge fragment generates a distinct polygon. Eventually, edges are merged to form a polygonal area. We can trace back the polygon vertices in time as we outlined in this paper and find grouping of edge pixels that originate the polygon.

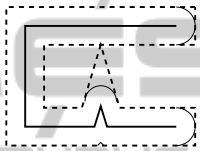


Figure 9.

6 CONCLUSION

Our approach uses a minimum spanning tree and straight polygons as intermediate representations, from which multiple grouping hypotheses are derived by tracing back each polygon vertex to the original point set and extracting disjoint polygons in the trace. Unlike traditional dot grouping algorithms, our approach provides multiple grouping instances. Each dot moves in at least two directions, thus can participate in both foreground and background and provide multiple interpretations. The algorithm is parameter free, simple, and deterministic.

Some of grouping hypotheses are perceptually more salient than others. We need to assign a score to each polygon using the geometry of the polygon in a way the score reflects the perceptual saliency. The initial grouping by a minimum spanning tree groups all dots into a single polygon, thus simplifies the subsequent algorithm slightly. However, it brings erroneous grouping especially in thin structures and between two separate clusters of points. More elaborate initial grouping needs to be tried and studied.

ACKNOWLEDGEMENT

This work is supported by the United States National Science Foundation grants CCF-1117439 and CCF-

1421734.

REFERENCES

Ahuja, N. (1982). Dot pattern processing using voronoi neighborhoods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (3):336–343.

Ahuja, N. and Tuceryan, M. (1989). Extraction of early perceptual structure in dot patterns: integrating region, boundary, and component gestalt. *Computer Vision, Graphics, and Image Processing*, 48(3):304–356.

Aichholzer, O. and Aurenhammer, F. (1996). *Computing and Combinatorics*, volume 1090 of *Lecture Notes in Computer Science*, chapter Straight skeletons for general polygonal figures in the plane, pages 117–126. Springer.

Aichholzer, O., Aurenhammer, F., Alberts, D., and Gartner, B. (1995). A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1(12):752–761.

Blum, H. (1967). A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form*, 19(5):362–380.

Chaudhuri, A., Chaudhuri, B. B., and Parui, S. K. (1997). A novel approach to computation of the shape of a dot pattern and extraction of its perceptual border. *Computer Vision and Image Understanding*, 68(3):257–275.

Edelsbrunner, H., Kirkpatrick, D., and Seidel, R. (1983). On the shape of a set of points in the plane. *Information Theory, IEEE Transactions on*, 29(4):551–559.

Elder, J. H. and Zucker, S. W. (1996). Computing contour closure. In *Proc. 4th European Conference on Computer Vision*, pages 399–412, Cambridge, UK.

Engbers, E. A. and Smeulders, A. W. M. (2003). Design considerations for generic grouping in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(4):445–457.

Greene, E. (2008). Additional evidence that contour attributes are not essential cues for object recognition. *Behavioral and Brain Functions*, 4(1):26.

Kubovy, M. and Wagemans, J. (1995). Grouping by proximity and multistability in dot lattices: A quantitative gestalt theory. *Psychological Science*, 6(4):225–234.

Lee, D.-T. (1982). Medial axis transformation of a planar shape. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (4):363–369.

Mahamud, S., Williams, L., Thornber, K., and Xu, K. (2003). Segmentation of multiple salient closed contours from real images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(4):433–444.

Melkemi, M. and Djebali, M. (2000). Computing the shape of a planar points set. *Pattern Recognition*, 33(9):1423–1436.

O'Rourke, J., Booth, H., and Washington, R. (1987). Connect-the-dots: a new heuristic. *Computer Vision, Graphics, and Image Processing*, 39(2):258–266.

Rosenberg, B. and Langridge, D. (1973). A computational view of perception. *Perception*, 2(4):415.

Toussaint, G. T. (1980). The relative neighbourhood graph of a finite planar set. *Pattern recognition*, 12(4):261–268.

Wang, S., Kubota, T., Siskind, J., and Wang, J. (2005). Salient closed boundary extraction with ratio contour. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(4):546–561.

Zahn, C. T. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *Computers, IEEE Transactions on*, 100(1):68–86.