

PARSING SILHOUETTES WITHOUT BOUNDARY CURVATURE

Ralf Juengling

Portland State University
Department of Computer Science
P.O. Box 751
Portland, Oregon 97201, USA

Lakshman Prasad

Space and Remote Sensing Sciences Group (ISR-2)
International, Space, and Response Division
Los Alamos National Laboratory
Los Alamos, NM 87545, USA

ABSTRACT

We describe a new decomposition algorithm for two-dimensional, polygonal shapes. The algorithm first finds a set of overlapping ribbon-like subshapes (“ribbons”) by grouping skeleton fragments into long smooth spines. The parts are then obtained by cutting the ribbons along lines of mutual intersection. With this approach we find part cuts that obey heuristic rules of early vision [17] without using boundary curvature.

Index Terms— Shape Decomposition, Delaunay triangulation, Saliency Network, Perceptual Grouping

1. INTRODUCTION

Part-based object recognition depends on algorithms for decomposing shapes into parts [18]. From an applications point of view, the prime requirement for such algorithms is stability of the decomposition when an object’s silhouette changes with varying viewing angle. This leaves much room for devising a part decomposition method.

There is evidence that the human visual system (HVS) recognizes parts before it recognizes objects [16]. Recent research in psychology and cognitive science has revealed some of the early vision heuristics the HVS employs to hypothesize part boundaries in the visual input [7, 17]. Since we know of no theoretically motivated performance criteria for shape decomposition our goal here is to mimic the apparent decomposition heuristics of the HVS.

Singh and co-workers presented simple shapes to subjects and asked them to partition each shape by drawing or selecting a certain number of “cuts” (straight line segments connecting two boundary points) [17]. The shapes did not resemble any particular objects and so the findings can be attributed to early vision. The authors concluded that the cuts are placed according to the following three rules:

1. Cuts link two points of minimum negative boundary curvature
2. Cuts are as short as possible
3. Cuts cross an axis of local symmetry

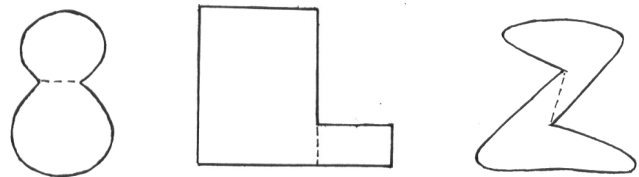


Fig. 1. Left: Shape with cut. **Middle:** Shape with short cut. **Right:** Shape with implausible cut.

Fig. 1 Left shows an example that fulfills all three rules, Fig. 1 Middle shows how the second rule selects a preferred cut when there are no two points of minimal negative curvature. Finally, Fig. 1 Right demonstrates that the first and second rule can be in conflict (Singh et al. did not say how a cut is chosen in such a case).

The first rule above (which is also known as the *minima rule* [7]) is expressed in terms of curvature, that is, a second order derivative of the boundary curve. We would like to avoid computing curvature for two reasons. First, reliably computing higher order derivatives in noisy signals is not easy [5, 1], and, second, for the purpose of estimating curvature we would have to map our polygonal representation to a more complicated representation.

Fortunately, we do not really need to know curvature values because we can characterize those boundary points selected by the first rule in other ways. The trick is illustrated in Fig. 2: In a first step our algorithm finds convex (or close to convex) “subshapes”. The boundaries of subshapes cross at the points we are looking for (Fig. 2 Right).

Our algorithm takes a polygon as input and yields a set of cuts (pairs of different boundary points) as output. In Section 2 we explain in detail how the subshapes and the cuts are found. In the remainder of this section we briefly review other work that we build upon, namely, an algorithm for computing the skeleton of a polygon in Section 1.1, and an algorithm for grouping skeleton pieces into longer curves in Section 1.2. Section 3 gives results and discusses the relation to similar algorithms in the literature.

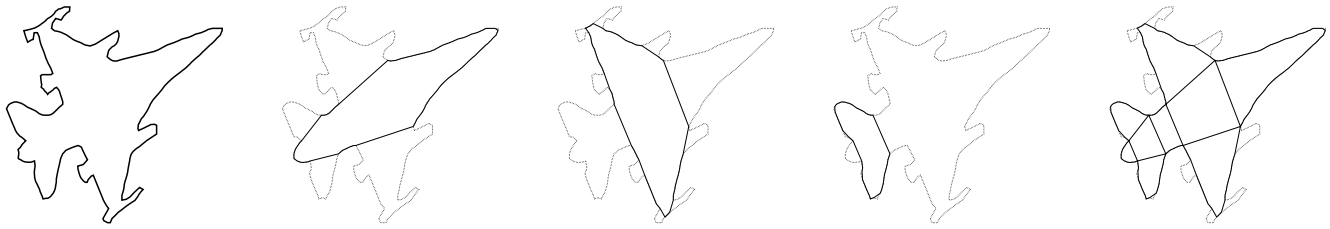


Fig. 2. Left: Fighter jet shape. **Middle:** Subshapes found by our algorithm. **Right:** Cuts obtained by overlapping subshapes.

1.1. Delaunay Triangulation for Shape Analysis

Our notion of *subshape* is equivalent to what has been called a *ribbon* in the literature [11]. A ribbon essentially is a simply connected region whose skeleton does not bifurcate, and hence consists of a single, smooth *spine*.

To find spines we need to examine a shape's skeleton. Shape skeletons derive from a notion of *local symmetry* by which two or more boundary points are paired on the grounds that they are co-tangent to a *maximum circle*. The two best known axial shape descriptors that derive from this notion are Blum's Symmetric Axis Transform (SAT) [2] and Brady's Smoothed Local Symmetries (SLS) [4]. We will look at (an approximation of) the SAT to find spines; further discussion on the two shape descriptors may be found in the literature [8].

Some algorithms for computing the SAT for polygonal shapes have been proposed [10]. Here we use Prasad's algorithm [13] to construct an approximation of the SAT. The algorithm consists of two steps. The first step is to compute the *constrained Delaunay triangulation* (CDT) of the polygon [6]. The construction of a Delaunay triangulation of a set of points derives from the concept of an *empty circle*, which is a circle that has three (or more) points *on* it and none *inside* it. Every such circle adds a triangle (or more) to the triangulation. For the *constrained* Delaunay triangulation we consider the points that are polygon vertices and modify the empty circle concept: points inside an empty circle are allowed as long as they are not *visible* from all three points on the circle [6] (a point A is visible to B when the line segment \overline{AB} does not cross any polygon edge).

The reason the CDT of a polygon may be used to construct an approximate SAT of a polygon is that empty circles converge to maximum circles when the sampling interval (the maximum distance between two adjacent polygon vertices) goes to zero. Consequently, the centers of the empty circles become points of the skeleton (and the radii of the empty circles become samples of the "distance function" [2]).

The second step of the algorithm is to turn the dual graph of the CDT (the graph with vertices corresponding to triangles and edges to pairs of adjacent triangles) into a geometric graph by assigning the empty circle centers and radii to the graph's vertices. Fig. 3 shows an example: a shape and its

CDT on the left, and the same shape with the SAT skeleton in the middle.

We need to define a few more terms. We call *chords* those edges of the CDT that are not edges of the polygon (chords are drawn as light lines in Fig. 3 Left). The triangles of the CDT can be classified as *junction*, *sleeve*, or *terminal* triangles, depending on the number of triangle edges that are chords (three, two, or one, respectively). If we remove all edges from our SAT graph that link to a "junction vertex" (a vertex that corresponds to a junction triangle), we obtain the *proto-skeleton* (Fig. 3 Middle).

1.2. Perceptual Grouping by Saliency

To find contours of objects in the output of an edge detector, edge elements need to be linked together into curves. Shashua and Ullman proposed a two-step algorithm for this problem [14], assuming an edge detector that yields *oriented* edge elements. More precisely, for a given image a graph is defined by connecting each pixel p with every pixel q in a fixed-sized neighborhood by a directed edge (p, q) . The edge detector is assumed to yield a subset A of the graph's edges E ; we call A *actual* and $V = E - A$ *virtual* edges.

In the first step the algorithm computes a *measure of saliency* $s_n(p, q)$ for every edge (p, q) in E . Concerned with finding long and smooth curves composed mostly of actual edges, Shashua and Ullman defined a measure of saliency for *sequences of edges* of length n . The measure assigns higher values to sequences that include more actual edges and that feature smaller changes of direction between subsequent edges. The measure is of a form that allows finding the globally most salient curve of length n by dynamic programming (DP). The quantity $s_n(p, q)$ computed for each edge (p, q) by the DP algorithm is the saliency of the most salient edge sequence of length n that starts with edge (p, q) .

The second step of the algorithm is to *trace a curve* (sequence of edges) in the graph. In the process of computing $s_n(p, q)$ in the first step, the DP algorithm considers all possible successor edges (q, r) , $r \neq p$, selects a "best" successor edge, and stores it as $succ(p, q)$ for the second step. With a successor edge assigned to every edge in the graph, tracing a curve is almost trivial. The non-trivial task is to specify a stopping criterion (this is again trivial when the algorithm is

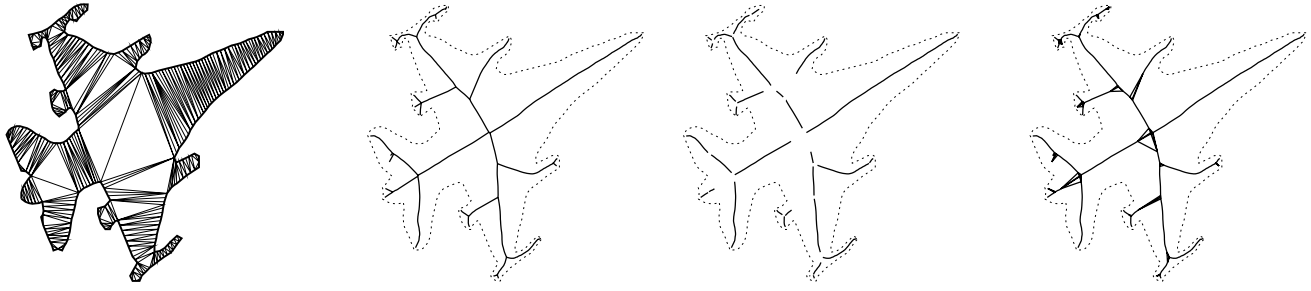


Fig. 3. Left: Fighter jet shape and its CDT. **Middle:** SAT skeleton and proto-skeleton of jet shape. **Right:** Saliency graph.

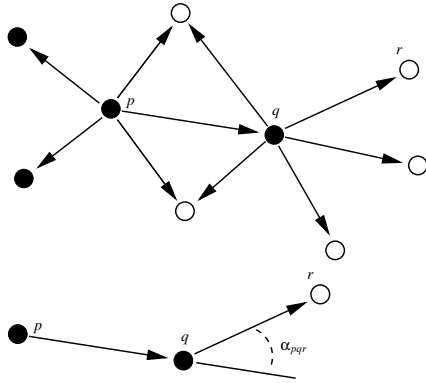


Fig. 4. Top: Saliency update of edge (p,q) ; r ranges over non-filled points. **Bottom:** Definition of α_{pqr} .

to find closed curves [14]). In the the next section we describe what stopping criterion we use.

The Sashua and Ullman algorithm may be applied to general geometric graphs (i.e., without constraints on point location and number of neighbors). Two things need to be specified, an *initial saliency* σ_{pq} for every edge $(p,q) \in E$, and a *coupling term* c_{pqr} for every pair $(p,q), (q,r)$ of adjacent edges.

The initial saliency, a non-negative value, gives the saliency of an edge “as is”. A reasonable choice is $\sigma_{pq} = 0$ when (p,q) is a virtual edge and a value proportional to the Euclidean length when (p,q) is an actual edge.

The coupling term c_{pqr} is a value between zero and one. It should quantify how well edge (q,r) continues a curve that ends in edge (p,q) . Shashua and Ullman suggested a term similar to $c_{pqr} = \exp(-\alpha_{pqr} \tan \frac{\alpha_{pqr}}{2})$, with the definition of α_{pqr} given in Fig. 4 Bottom. Note that c_{pqr} is equal to one when (p,q) and (q,r) point in the same direction and zero when the two edges point in opposite directions.

The saliency is computed iteratively via the following up-

date equations:

$$s_0(p,q) = \sigma_{pq} \tag{1}$$

$$s_{n+1}(p,q) = \sigma_{pq} + \max_{(q,r) \in E, r \neq p} c_{pqr} s_n(q,r) \tag{2}$$

We use the these equations with somewhat different initial saliencies and a different coupling term, which are discussed in the next section.

2. OUR ALGORITHM

Our goal is to find shape cuts by intersection of ribbon-like subshapes. Ribbons are characterized as having the simplest possible skeleton, a single smooth spine. The first step therefore is to discover substructures in the shape’s skeleton that look like smooth spines.

A problem occurs when boundary perturbations or small-scale shape features distort an otherwise smooth skeleton by inflicting bifurcations. This is a well-known phenomenon. To avoid this problem we ignore SAT skeleton bifurcations altogether and work with the proto-skeleton. Our idea is that Shashua and Ullman’s algorithm will highlight long smooth spines spanning several well-aligned skeleton fragments when the proto-skeleton is embedded in an appropriate “saliency graph”. The saliency graph provides the virtual edges needed to complete gaps between skeleton fragments. With a saliency graph G and a counter $i = 0$ we extract subshapes as follows:

1. Assign initial saliency values to the edges in G
2. Run the saliency algorithm and extract the most salient curve C_i
3. Find the subshape S_i corresponding to C_i
4. Set initial saliency values of all edges in C_i to zero
5. Increment i and go back to Step 1.

We terminate the iteration when the subshapes S_i together cover 97 or more percent of the shape’s area. Note that each iteration yields a different most salient curve because the initial saliency values are different in each iteration. The subshape S_i is the union of the triangles that correspond to ver-

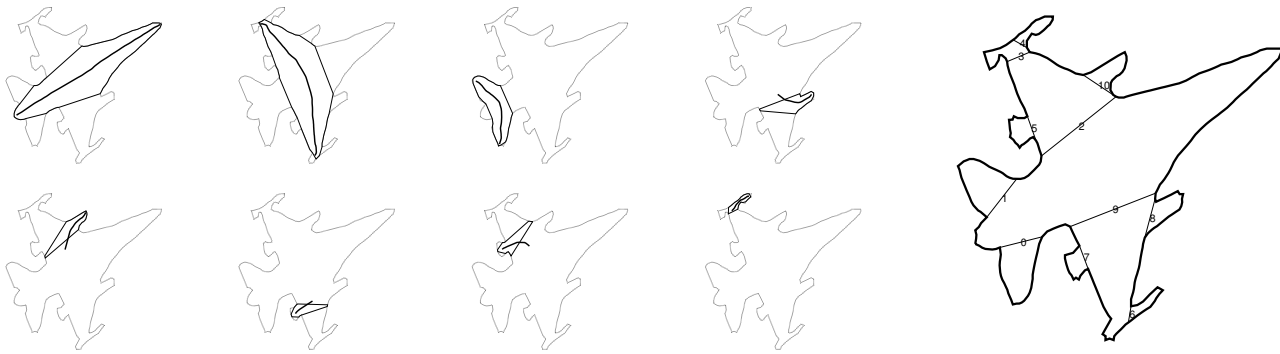


Fig. 5. Left: Salient spines and corresponding subshapes found in first 8 iterations. **Right:** Cuts obtained from the subshapes.

tices in C_i . Fig. 5 Left shows what spines and subshapes we get for the fighter jet shape during the first eight iterations.

Each new subshape S_i may contribute zero, one, or two cuts. For each subshape we first check which ends of the spine reach the boundary and which ends are inside the shape. For example, the spines of the first three subshapes in Fig. 5 Left reach the shape boundary on both ends, all other spines shown reach the boundary on one end only. We trace a spine from a boundary end and move into the shape until a triangle shared by some subshape S_j , $j < i$ is reached. The chord through which the shared triangle was entered becomes a cut. If no such triangle exists and if the other end of the spine is not also a boundary end, then the last chord on the trace becomes a cut.

The intuition behind this arbitrary looking procedure is that the first subshape will in general account for the bulk of the shape, while subsequent subshapes will correspond to protrusions in order of decreasing “mass” (we explain below how we realize this ordering). Thus, one may think of the first subshape as the “torso” and of following subshapes as “limbs” and “ears”, and the idea behind our cut selection is to find the chords that separate a limb (or ear) from the torso.

Fig. 5 Right shows what cuts result from this procedure for the jet shape. In this case, the first subshape (cf. Fig. 5 Left) contributes no cut because it does not overlap with any earlier found subshapes and the spine does not end terminate inside the shape. The second subshape overlaps with the first, and the two resulting cuts separate the wings from the fuselage. Similarly, the fourth subshape overlaps with the second and contributes a single cut which separates a protrusion from the wing. We now discuss details of our algorithm.

2.1. Constructing the Saliency Graph

Bifurcations in the SAT occur at junction triangles. Often two or more junction triangles cluster together and form a polygon as, for instance, the two big central junction triangles in the CDT of the fighter jet shape (Fig. 3 Left). We will more generally speak of “junction polygons”. A junction polygon

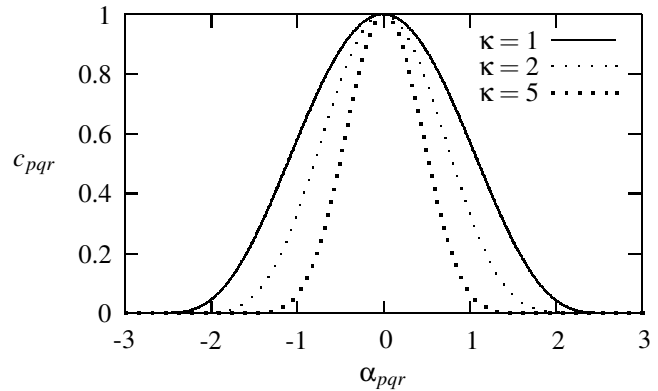


Fig. 6. Coupling over change of direction for different κ .

with three vertices is a junction triangle and gives rise to a degree-three bifurcation. Junction polygons with more than three vertices give rise to several bifurcations.

In the proto-skeleton, a skeleton fragment ends at a junction polygon. Any other fragment ending at the same junction polygon may be a good continuation of the first fragment. Hence, the saliency graph should contain virtual edges connecting all end-points of fragments ending at the same junction polygon. Fig. 3 Right shows the saliency graph that results when this rule is applied to the proto-skeleton of the fighter jet.

In summary, to construct the saliency graph first find all junction polygons and then, for each junction polygon, add edges between all endpoints of skeleton fragments. To determine junction polygons traverse the dual graph of the CDT and join any two adjacent triangles when they are both junction triangles.

2.2. Saliency Computation

For the computation of saliency we use the update equations (1) and (2) but with somewhat different initial saliency

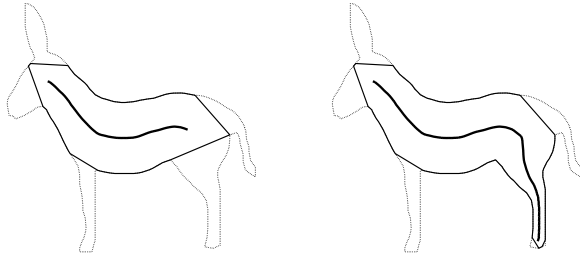


Fig. 7. Left: First subshape of the donkey shape. **Right:** First subshape without the second trace termination criterion.

values and coupling terms than those proposed by Shashua and Ullman. The two vertices p and q in each edge (p, q) of the proto-skeleton correspond to triangles T_p and T_q , respectively. We choose the sum of the areas of these triangles as initial saliencies. Let $A(T)$ denote the area of triangle T .

$$\sigma_{pq} = \begin{cases} 0 & \text{if } (p, q) \text{ is a virtual edge} \\ A(T_p) + A(T_q) & \text{if } (p, q) \text{ is an actual edge} \end{cases} \quad (3)$$

Note that with these initial saliencies the extracted subshapes tend to be ordered by decreasing area (cf. Fig. 5 Left).

We added a parameter κ to the coupling term which allows us to control the degree of bending in admissible spines. A greater value of κ corresponds to a narrower range of angles with effective coupling (cf. Fig. 6).

$$c_{pqr} = \exp\left(-\kappa \alpha_{pqr} \tan \frac{\alpha_{pqr}}{2}\right) \quad (4)$$

For all results presented here we used $\kappa = 8$ and did 60 iterations of the saliency computation.

2.3. Extracting Salient Spines

To extract a most salient spine we first find the edge (p, q) of the saliency graph which maximizes the sum $s_n(p, q) + s_n(q, p)$. Starting from this edge we trace the curve in both directions until we hit an end or until one of two termination criteria is fulfilled.

The first criterion concerns the angle α_{pqr} to a successor edge (q, r) . We terminate when it exceeds a certain threshold (in practice we express the threshold in terms of the coupling value and terminate when $c_{pqr} \geq 0.2$). The second criterion concerns the points on the subshape's current boundary. Recall that every vertex of the saliency graph corresponds to a triangle, and hence to three points on the boundary. We terminate when transitioning from the current head of the trace, q , to the next vertex, r , would bring about too concave an angle in the subshape boundary (again we use a simple threshold and terminate when an exterior angle would be below $0.8 * \pi$). Fig. 7 Left illustrates a case where we terminate by the second criterion.

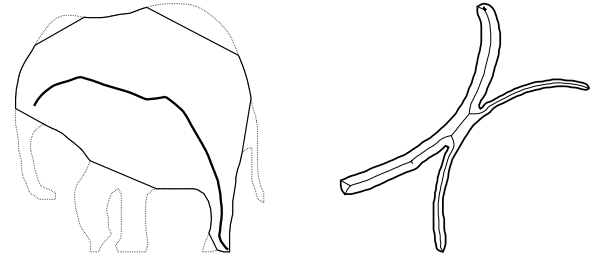


Fig. 10. Left: First subshape of the elephant shape. **Right:** The desired cut is not a chord of the CDT.

3. RESULTS AND DISCUSSION

Figs. 8 and 9 show some result of our algorithm. Most of the cuts comply with the rules cited in the first section, but some seem out of place, and some seem to be missing. For example, the cuts dividing the two wings of the first jet Fig. 8 are unwarranted, while the upper tail fin should be cut just like the lower tail fin. Or, the cut dividing the torso of the second fish and the cut through the elephant's back in Fig. 8 are not justified.

We found that unsatisfactory cuts like these can usually be attributed to poor results of the first phase of our algorithm (subshape extraction). In some cases, like the elephant's back, the error is a consequence of our too simple a concept of subshape (single spine). To avoid errors of the first kind we need to make the subshape extraction more robust, to avoid errors of the second kind we consider evaluating selected cuts a second time. For both cases we expect that it will be necessary to take into account boundary features. Thus, we acknowledge what other authors have pointed out before, namely, that comprehension of shape requires an analysis that integrates axial and boundary features [8, 17].

Finally, we want to add that the desired cuts may not always be found among the chords of the CDT, Fig. 10 Right gives an example.

3.1. Related work

The idea of searching for shape cuts in the edges of a Delaunay triangulation is due to Prasad [12]. Brady put forth the idea of grouping skeleton fragments into long, smooth spines to be used in a more expressive shape descriptor than the SAT; he also noted the utility of such a representation for shape decomposition [4]. Massad and Medioni have proposed a method for decomposing a shape into "overlapping parts" [9]. Their method is quite different from our method for finding subshapes in that it analyzes the shape boundary instead of an axial descriptor of the shape.

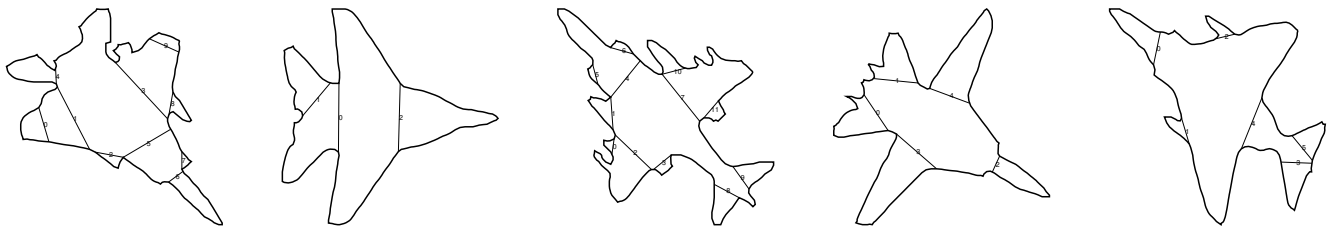


Fig. 8. More jets with cuts.

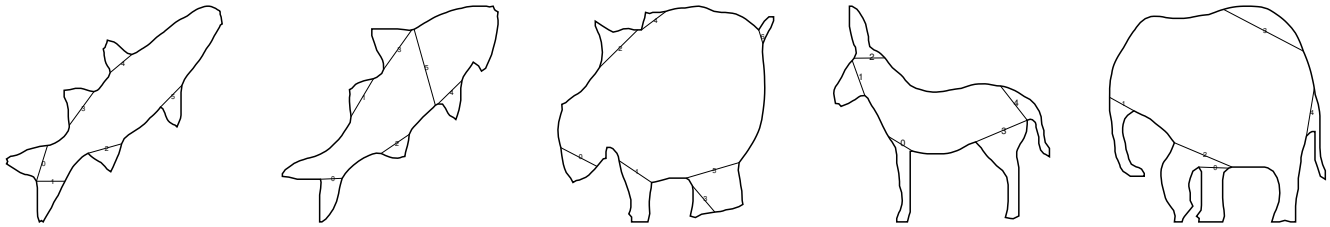


Fig. 9. Some cuts of fish and game.

4. ACKNOWLEDGEMENTS

This work was supported by Los Alamos National Laboratory and the J. S. McDonnell Foundation. We used J. R. Shewchuk's Triangle code [15], and Y. Lecun and L. Bottou's Lush programming environment [3]. Thanks to M. Mitchell for helpful comments on an earlier draft.

5. REFERENCES

- [1] H. Asada and M. Brady. The curvature primal sketch. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(1):2–14, Jan. 1986.
- [2] H. Blum and R. N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10(3):167–180, 1978.
- [3] L. Bottou and Y. LeCun. *Lush Reference Manual*. <http://lush.sourceforge.net/lush-manual.pdf>, 2003.
- [4] M. Brady and H. Asada. Smoothed local symmetries and their implementation. *International Journal of Robotics Research*, 3(3):36–61, 1984.
- [5] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov. 1986.
- [6] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. CRC Press, 1997.
- [7] D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, 18:65–96, 1985.
- [8] M. Jenkinson and M. Brady. A saliency-based hierarchy for local symmetries. *Image and Vision Computing*, 20(2):85–101, 2002.
- [9] A. Massad and G. Medioni. 2-D shape decomposition into overlapping parts. In *Visual Form 2001, 4th International Workshop on Visual Form, Proceedings*, volume 2059 of LNCS, pages 398–409. Springer, 2001.
- [10] R. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition, CVPR*, pages 63–69, Los Alamitos, California, 15–18 June 1992. IEEE Computer Society.
- [11] J. Ponce. On characterizing ribbons and finding skewed symmetries. *Computer Vision, Graphics, and Image Processing*, 52(3):328–340, Dec. 1990.
- [12] L. Prasad. Rectification of the chordal axis transform and a new criterion for shape decomposition. In G. D. E. Andres and P. Lienhardt, editors, *Discrete Geometry for Computer Imagery, 12th International Conference, Poitiers (France), April 2005, Proceedings*, volume 3429 of LNCS, pages 263–275. Springer, 2005.
- [13] L. Prasad and R. L. Rao. A geometric transform for shape feature extraction. In *Proceedings of the SPIE*, volume 4117, pages 222–233. IEEE, 2000.
- [14] A. Shashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *International Conference on Computer Vision*, pages 321–327, 1988.
- [15] J. R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In M. C. Lin and D. Manocha, editors, *WACG*, volume 1148 of LNCS, pages 203–222. Springer, 1996.
- [16] T. Shipley and P. Kellman, editors. *From fragments to objects: Grouping and segmentation in vision*, volume 130 of *Advances in Psychology*. Elsevier Science, 2001.
- [17] M. Singh, G. Seyranian, and D. Hoffman. Parsing silhouettes: The short-cut rule. *Perception and Psychophysics*, 61:636–660, 1999.
- [18] S. C. Zhu and A. L. Yuille. Forms: A flexible object recognition and modelling system. *International Journal of Computer Vision*, 20(3):187–212, 1996.