Crazy Cuts: Dissecting Planar Shapes into Two Identical Parts

Alfred M. Bruckstein, Doron Shaked

HP Laboratories
HPL-2009-61

Keyword(s):
Self Docking, congruent piece decomposition

Abstract:

We analyze a well known type of puzzle in planar geometry: given a planar shape, it is required to find a cut that divides the shape into two identical parts (up to rotation and translation). Clearly not all shapes can be so dissected and for some shapes that appear in puzzles the cutting curve is quite surprising and difficult to find. In this paper we first analyze the inverse problem of assembling planar shapes from two identical parts having partially "matching" boundaries and then use the insights gained on this topic to derive an efficient algorithm to solve the dissection puzzle in quite general situations.

External Posting Date: March 21, 2009 [Fulltext]   Approved for External Publication
Internal Posting Date: March 21, 2009 [Fulltext]

Crazy Cuts: Dissecting Planar Shapes into Two Identical Parts

Alfred M. Bruckstein[*]and Doron Shaked[†]

March 1, 2009

Abstract

We analyze a well known type of puzzle in planar geometry: given a planar shape, it is required to find a cut that divides the shape into two identical parts (up to rotation and translation). Clearly not all shapes can be so dissected and for some shapes that appear in puzzles the cutting curve is quite surprising and difficult to find. In this paper we first analyze the inverse problem of assembling planar shapes from two identical parts having partially "matching" boundaries and then use the insights gained on this topic to derive an efficient algorithm to solve the dissection puzzle in quite general situations.

1 Introduction

Consider the planar shape depicted in Figure 1a. The goal is to find a cutting curve that divides this shape into to two identical shapes. The solution is seen in Figure 1b, and is not trivial to find. Several other examples of crazy-cut dissection puzzles are shown in Figure 2. The question we address in this paper is the following: given a planar shape, determine whether a simple cutting curve exists dissecting the shape into two identical parts and, if it exists, find the cutting curve efficiently. To answer this question we shall first analyze the inverse problem of assembling a planar shape from two identical shapes that

[*]Computer Science Department, Technion - IIT, Haifa, Israel. E-mail: freddy@cs.technion.ac.il. Corresponding author.

[†]HP-labs, Haifa, Israel. E-mail: doron.shaked@hp.com

have partially "matching" boundaries. This problem may be regarded as solving a simple jigsaw puzzle of two pieces (with no drawings on them).

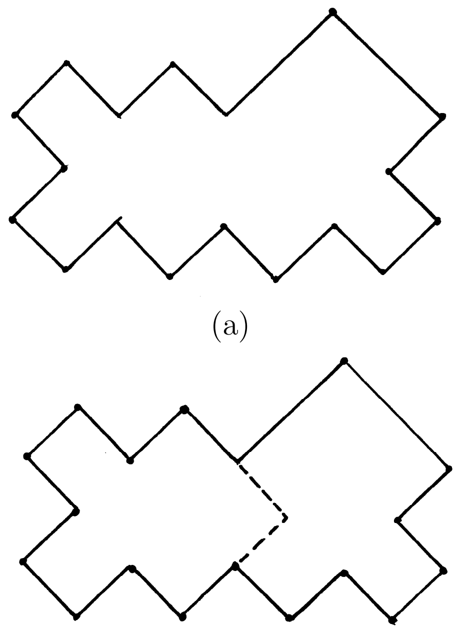


Figure 1: (a) The shape and (b) the "crazy-cut" into two identical parts (after Martin Gardner).

2 Solving Two Piece Jigsaw Puzzles

A simple planar shape (with no holes) may be represented by the closed planar curve of its boundary. Suppose we have a Euclidean invariant description of the closed boundary of the two (identical) shapes we must put together. The description can be the curvature vs arclength "invariant signature" description $k(s)$, where $s \in [0, L]$ is the arclength along the boundary (L being the total length of the planar boundary that will be assumed measurable, and $s = 0$ being an arbitrarily selected starting point on the boundary). If the boundary is

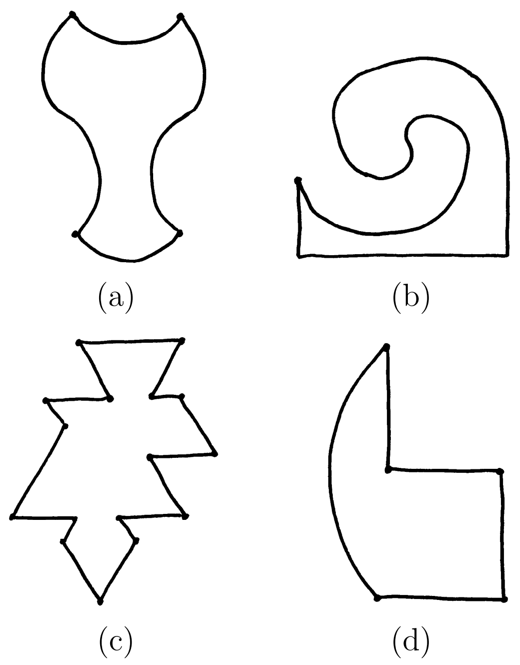


Figure 2: Some crazy-cut challenges.

non-smooth we can defined $k(s)$ as having δ- function components describing sharp angles at breakpoint or we might assume that we have an equivalent description of the boundary via $k(s)$ between breakpoints along with the turn angle information at each breakpoint. If we have to deal with polygonal planar shapes, the "boundary signature" may be a sequence of edge-length (l_i) and the turn angles (φ_i) at each vertex of the polygon (see Figure 3). For shapes digitized on $\mathbb{Z}^2$ the natural boundary signature is the so-called crack-code of the boundary, tracing the boundary of the shape built from adjacent square pixels of size (1×1) having integer-coordinate vertices.

We may now ask what characterizes, in terms of the invariant signature function, for instance the Euclidian-invariant curvature $k(s)$, matching portions of boundaries of two given shapes S_I and S_{II}. If $k^I(s)$ and $k^{II}(s)$ describe the boundaries of the two shapes in a clockwise traversal from arbitrary initial conditions and the portion between s^I_A and s^I_B on the boundary of S_I matches the portion between

s_A^{II} to s_B^{II} we shall have (see Figure 4) that:

$$k^I(s) = -k^{II}(\Sigma - s) \quad s \in [s_A^I, s_B^I].$$

Since clearly along the common boundary portions of the shapes S_I and S_{II} we have the same traversal rate (as arclength traversal implies unit speed clockwise travel along the boundary!) but the velocity vector at each point turns in opposite directions. Note that we also have

$$\Sigma - s_A^I = s_B^{II}$$
$$\Sigma - s_B^I = s_A^{II}$$

and therefore

$$\Sigma = s_A^I + s_B^{II} = s_A^{II} + s_B^I.$$

Indeed if we plot $k^I(s)$ as a periodic function of s with period equal to L_I, the length of the boundary of shape S_I, and similarly $k^{II}(s)$ as a periodic function of s with period $L_{II}, \ldots$ the picture looks like illustrated in Figure 5. We see that for both S_I and S_{II} we can regard the signature function $k(s)$ as being composed of alternating part $\mathbf{P_I}$ and $\mathbf{J_I}$ and $\mathbf{P_{II}}$ and $\mathbf{J_{II}}$ where $\mathbf{J_I}$ and $\mathbf{J_{II}}$, the matching portions, have same length L_J and are (up/down) and (left/right) mirror reflections of each other. The joint object, after "docking" the two jigsaw puzzle pieces together will have a signature function that can be described by $\mathbf{P_I}$ followed by $\mathbf{P_{II}}$ with a length of total $L_I + L_{II} - 2L_J$.

3 Self Docking of Shapes

Up to this point the discussion was for two general shapes S_I and S_{II}. However we are interested in matching identical shapes, i.e. $S_I \equiv S_{II} \equiv S$. In this case we shall have to have for $k(s)$, the invariant signature description of the boundary of S, that

$$k(s) = -k(\Sigma - s) \;\text{ for }\; s \in [0, \Delta] \qquad (1)$$

(where we decided w.l.o.g. to start the arclength parametrization at $s_A = 0$). Here, however, something interesting can be observed: if the intervals $[0, \Delta]$ and $[\Sigma - \Delta, \Sigma]$ are disjoint it means that there are two distinct portions on the shape's boundary that can be matched, but if the intervals overlap (i.e. $\Delta > \frac{\Sigma}{2}$ and consequently $\Sigma - \Delta < \frac{\Sigma}{2}$) we necessarily get that $k(s) = -k(\Sigma - s)$ for all the interval $[0, \Sigma]$. Indeed for $s = \frac{\Sigma}{2}$ in (1) we'll have that $k(\frac{\Sigma}{2}) = -k(\frac{\Sigma}{2}) = 0$, and more importantly that $k(\tilde{s}) = -k(\Sigma - \tilde{s})$ for $\tilde{s} \in [\Delta, \Sigma]$ too by realizing that this is simply reading equation (1) with sides interchanged, i.e. redefining the arclength $\tilde{s}$ via $\tilde{s} = \Sigma - s$. (See Figure 6). Note that in the above discussion we assume that the interval $[0, \Sigma]$ is maximal, in the sense that $k(0 - \varepsilon) \neq -k(\Sigma + \varepsilon)$. The above considerations prove the following

Self Docking Dichotomy Lemma

A given planar shape either "docks" to itself over totally disjoint matching portions of its boundary or over the exact same portion of its boundary and it cannot possibly have selfdockings that match over boundary portions that are only partially disjoint (i.e. different boundary intervals that have a common boundary portion).

Also note that if the shape "docks" to itself on the same portion of its boundary we shall always have at the midpoint of the match "an inflexion" point of zero curvature.

The consequences of these observations are far-reaching indeed, with regard to the boundary of the composite shape obtained after docking two identical parts together. If the docking was over the same portion of the boundaries of the component shapes the outer boundary will necessarily be the concatenation of two identical boundary curves, (see Figure 7). In this case the cut curve is completely "out of sight", i.e hidden inside the composite shape and in fact any symmetrical cut from M to M' (in Figure 7) will yield a possible solution.

So far we have seen that the case of self-docking along the same portion of the boundary is the trivial case of cutting a shape that has two identical curves joining together to form the composite boundary. The interesting case occurs when the self docking is along disjoint portions of the boundary. This case is illustrated in Figure 8. Let us call the matching portions $\mathbf{J}$ and $\bar{\mathbf{J}}$, and we assume that $\mathbf{J}$ and $\bar{\mathbf{J}}$ are different segments of $k(s)$, i.e. they correspond to two intervals $[s_A, s_B]$ and $[\bar{s}_A, \bar{s}_B]$ that are of the same length but disjoint in $s \in [0, L]$. Without loss of generality let us take $s_A = 0$. Then the boundary of

S will comprise the intervals

$$\mathbf{J}_{[s_A=0,s_B]}\ \mathbf{P}_{[s_B\bar{s}_A]}\ \bar{\mathbf{J}}_{[\bar{s}_A\bar{s}_B]}\ \mathbf{Q}_{[\bar{s}_B,L]} \text{ in cyclic order.}$$

Using this motivation we'll have that the two identical shapes : S_1 : $\mathbf{JP\bar{J}Q}$ and S_2 : $\mathbf{JP\bar{J}Q}$ when docked so as to have $\mathbf{J}$ matched to $\bar{\mathbf{J}}$ will result in a combined shape with boundary described by the following syntax

$$(S_{\text{combined}}) : \mathbf{P\bar{J}QQJP} \sim \mathbf{\bar{J}QQJPP} \sim \mathbf{QJPP\bar{J}Q}$$

where $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{J}$, $\bar{\mathbf{J}}$ are the $k(s)$ portions that describe the original (component piece) S. Hence if a shape can be represented as the docking of two identical jigsaw-puzzle pieces its boundary is either of the form:

$$(S_{\text{combined}}) : \mathbf{PP} \text{ if } S : \mathbf{JP} \equiv \mathbf{\bar{J}P} \quad (\text{in this case } \mathbf{J} \equiv \bar{\mathbf{J}})$$

or of the form:

$$(S_{\text{combined}}) : \mathbf{P\bar{J}QQJP} \ \text{ if } S : \mathbf{JP\bar{J}Q} \quad (\text{in this case } \mathbf{J} \neq \bar{\mathbf{J}}).$$

4 Finding Crazy Cuts

The structure of the invariant signature representing the combined shape yields an efficient algorithm for finding the crazy cut of a planar shape if such a cut exists, or for determining that such a cut is not possible. Indeed when we are given a $k(s)$ or any Euclidean invariant signature representation of the composite boundary we have to determine whether it has, from some starting point the structure $\mathbf{PP}$ or the structure $\mathbf{P\bar{J}QQJP}$. In fact, exhaustive search for a given string of length L, which would test all starting points and all possible internal distributions of lengths of $\mathbf{P}, \bar{\mathbf{J}}, \mathbf{Q}, \mathbf{J}$ would be feasible and quite efficient if the signature was discrete. All starting point possibilities will involve L runs of checking whereas $l(\mathbf{P})$ and $l(\mathbf{J}) \equiv l(\bar{\mathbf{J}})$ are two additional parameters that are needed to be set. (Recall that $l(\mathbf{P}) + l(\mathbf{J}) + l(\bar{\mathbf{J}}) + l(\mathbf{Q}) = L$ hence these two parameters also determine $l(\mathbf{Q})$!). Hence we can test a string of length L for the structure $\mathbf{P\bar{J}QQJP}$ with an exhaustive search algorithm of $O(L^4)$ complexity, and this without any sophisticated string manipulation optimization.

Notice that the additional (fourth) $O(L)$ complexity is due to the need to verify the syntax for every choice. In the sequel we will argue that the latter can be included in the $O(L^3)$ search leading to an overall complexity of $O(L^3)$.

4.1 An Efficient Algorithm for Finding Crazy Cuts for Polygons

In this section we detail the crazy cut algorithm for a polygonal shape. A polyline description may start by specifying the coordinates of the first vertex and the direction of the first edge. If such initialization is omitted, we may, w.l.o.g. place the vertex in the origin, and orient the first edge in the positive $\hat{x}$ direction. Following the initialization we traverse the boundary of the shape in a clockwise manner, specifying L couples (l_i, φ_i) of edge lengths l_i and turn angles φ_i as see for example in Figure 3b.

Notice that since we know the polyline describes a closed shape the length of the last edge and the turn angle of the two last segments are redundant. For completeness one may assume they are given, and we may verify that the data agrees with the closed curve assumption.

We start the algorithm with a choice of two vertices and the boundary polyline segment connecting the first vertex to the second vertex in a clockwise manner. There are $O(L^2)$ possible vertex choices.

For each choice of vertices we will check the possibility that the connecting boundary segment contains the two segments **PP** in the boundary syntax sequence $\mathbf{QJPP\bar{J}Q}$. To do that, we will cut the initial boundary segment into two segments of equal length, and compare them. They are qualified to be together the **PP** segment if both halves are similar segments. Traversed clockwise on both halves edges should be of the same length, and with the same turn angles. Additionally, the internal angles on both ends of the initial segment should sum up to the internal angle corresponding the midpoint of the segment, since the midpoint should be the location where the **J** segment of one half shape meets the $\mathbf{\bar{J}}$ segment of the other half (see Figure 9a).

This is the time to review a couple of interesting special cases that may affect this part of the algorithm.

1. The boundary segment **PP** may be trivially short, that is, it may well be a single vertex. Indeed, a vertex of a shape may often be a pivot point around which one copy of a shape turns, whereby

one side of the vertex matches the other side (see e.g. Figure 10). Therefore every vertex constitutes one successful virtual selection of the two vertices above.

2. The midpoint of the segment does not have to occur on a vertex. It can well be inside an edge of the full polyline which may be the adjoining point of two vertices complementing each other to 180^0, (see e.g. Figure 9b).

3. The end point of one of the segments can occur on an edge. It should be noted that in this case the other end (corresponding to the other end of the segment **P**) has to coincide with a vertex. Note also that in this case the interface between the two copies of **P** occurs on a vertex whose internal angle is the sum of the vertex angle and 180^0, see Figure 11a . The next two paragraphs will deal with this special case.

   To cover the cases described in the last item, the first part of the algorithm has to be repeated. This time again two vertices are selected, and the segment between them is assumed to be the first copy of **P**. To check this, a second copy is allocated by allocating the length of the first segment on its counterclockwise side (see Figure 11b). The algorithm described above is repeated, with one change the counterclockwise end of the second copy has an 180^0 vertex (if the second copy ends in a vertex one can skip this part).

   Naturally one has to repeat the algorithm above for every initial selection of two vertices by allocating the length of the initial boundary segment on the clockwise side of the first selection.

Having completed the first part of the algorithm where a boundary segment has qualified to be the **PP** part of the required boundary syntax $\mathbf{QJPP\bar{J}Q}$, we can continue checking the rest of the syntax. Following the boundary on both ends of the **PP** segment we start assembling the **J** and the $\mathbf{\bar{J}}$ segments. We match the length of the first edge on both ends, and make sure the next turn angles are negatives of each other. We continue until one of the conditions is broken. If the angle condition was met and one of the edge segments is shorter, we stop the $\mathbf{J\bar{J}}$ search in the vertex of the short edge, and the middle of the longer edge. If the edge length condition has been met, but not the turn angle condition, we stop at both vertices. Notice that this stage cannot fail (we start with a successful vertex condition and may well stop on the first edge).

The boundary segments we managed to traverse are the candidates for the **J** and $\bar{\mathbf{J}}$ segments. The remaining boundary segment should now correspond to the **QQ** segment. The latter is verified by cutting it in midpoint, and checking exactly like we did the **PP** segment (including the internal angle condition, see Figure 9).

It should be noted that a shape may well pass all the syntax conditions, but not have a crazy cut. To verify the validity of the crazy cut one has to make sure the boundary segment made of $\mathbf{QJP\bar{J}}$ constitutes a valid closed shape. To do that we have to concatenate all the boundary segments (including the turn angles in the ends of the boundary segments) and test whether the the resulting boundary is a simply closed contour.

To summarize the complexity of the algorithm: The first selection of points is $O(L^2)$. For each selection the traversal of all the other conditions: Checking the **PP** segment, the **J** and $\bar{\mathbf{J}}$ segments and the **QQ** segment amounts to an $O(L)$ processing time. Hence the total algorithm complexity is $O(L^3)$. A successful completion of the syntax search is so rare (in our implementations we found only one false alarm) that the additional $O(L \log L)$ complexity of the final verification does not increase the total complexity.

4.2 Algorithm for Crazy Cuts for Pixelized Shapes

Pixelized shapes are shapes made out of a collection of square pixels. Formally, they are a special case of polylines discussed above. For simplicity we can optionally modify their description such that each edge is exactly one unit length long, and turn angles are either $+90^0 - 90^0$ or 0^0 the latter being the main difference form the standard polyline description. All above depicted syntax checking algorithms remain essentially the same. Obviously, here edge length comparison becomes trivially simple, and there are no edge length mismatch cases such as in Figure 11. Additionally, if we require that both the full shape and each of the cuts are pixelized, we inevitably miss some crazy cuts (where the cuts are not pixelized). Figure 12 depicts some interesting pixelized crazy cuts that were combined and then detected via our algorithms.

5 Concluding Remarks

We have seen that analysis of the self-docking problem for planar shapes readily leads to a very nice characterization of shapes that can be split into two identical shapes and to efficient ways to solve crazy cut puzzles. We would have been elated to be the first to provide a mathematical discussion on an algorithm for crazy-cut problems, however a thorough search of the web revealed a paper of Kimmo Eriksson dealing with "crazy-cut" in 1996 [1]. His approach, subsequently criticized and elaborated upon by G. Rote and his collaborators [2, 4, 3], relies on checking whether two parallel tracings, one along the border of the shape (master) and a corresponding curve (slave) that "follows the master" tracing in a Euclidean invariant way yield a solution to the problem, when initiated at two arbitrarily selected border points. Although both Eriksson and Rote eventually obtain efficient algorithms to solve crazy cut puzzles, their approach is considerably complicated by the fact that the very simple syntax made obvious by analyzing self-docking was lacking in their work. It is very nice however to realize that both approaches eventually yield solutions to the geometric problem at hand via string analysis, the string being an Euclidean-invariant-signature-based description of the borders of the object. This way of encoding shape is very important and basic in the shape analysis field, where automatic curve matching [5], computerized jigsaw puzzle solutions [6], shape docking [7], invariant shape processing [9], and skew-symmetry detection issues [8] were dealt with via such an approach.

Note that from the work on viewpoint-invariant planar shape analysis via variations of projective, affine and similarity invariant boundary signatures, see e.g. [8, 9], we see that invoking generalized invariant signatures we could readily solve crazy cut puzzles even for shapes distorted by such, rather complicated, viewing transformations.

Acknowledgement

We are grateful to Tomer Shaked for challenging us with several hard crazy-cut puzzles, thereby initiating the work reported in this paper.

References

[1] K. Eriksson, "Splitting a polygon into two congruent pieces", The American Mathematical Monthly, Vol. 103, No. 5, pp. 393-400, May 1996.

[2] G. Rote, "Some thoughts about decomposition of a polygon into two congruent pieces", Unpublished Draft, page.mi.fu-berlin.de/∼ rote/Papers/postscript/Decomposition+of+a+polytope+into+two+congruent+pieces.ps, 1997.

[3] D. El-Khechen, T. Fevens, J. Iacono, and G. Rote, "Partitioning a polygon into two congruent pieces", Kyoto International Conference on Computational Geometry and Graph Theory, KyotoCGGT2007, Kyoto, Japan, June 11-15, 2007.

[4] D. El-Khechen, T. Fevens, J. Iacono, and G. Rote, "Partitioning a polygon into two mirror congruent pieces", 20th Canadian Conference on Computational Geometry, CCCG 2008, Montréal, Québec, August 13-15, 2008.

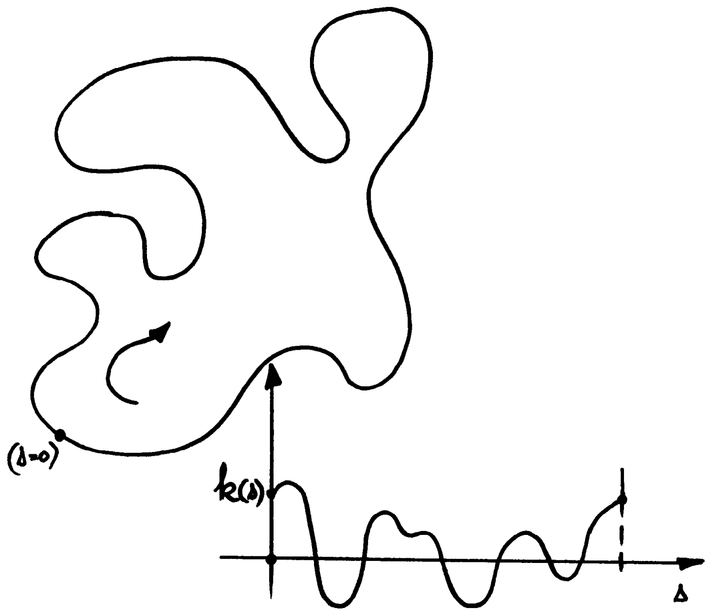
[5] H. J. Wolfson, "On curve matching", IEEE Transactions on PAMI, Vol. 21, No. 5, pp. 483-489, May 1990.

[6] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdan, "Solving jigsaw puzzles using computer vision", Ann. Oper. Res., Vol. 12, pp. 51-64, 1988.

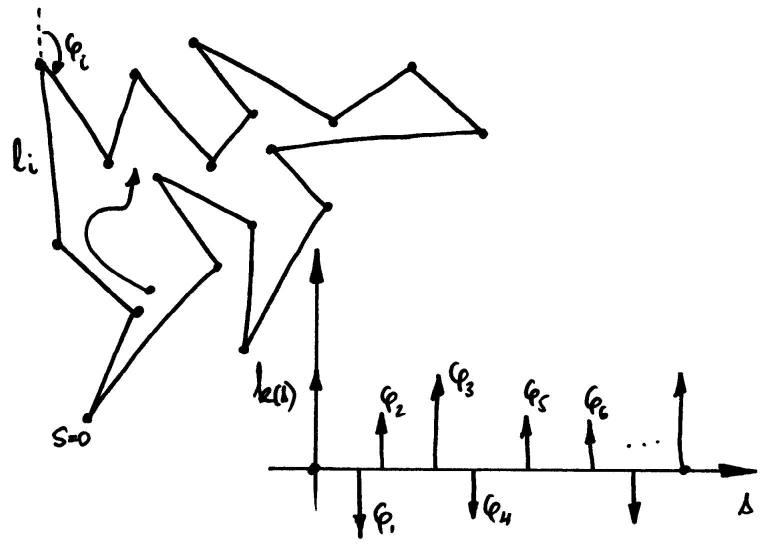
[7] A. Imiya and S. Kudo, "Docking of polygons using boundary descriptor", CAIP 2003, LNCS 2756, pp. 25-32, Springer-Verlag Berlin Heidelberg 2003.

[8] A.M. Bruckstein and D. Shaked, "On projective invariant smoothing and curve evolutions", Journal of Mathematical Imaging and Vision, Vol. 7, pp. 225-240, 1997.

[9] A.M. Bruckstein and D. Shaked, "Skew symmetry detection via invariant signatures", Pattern Recognition, Vol. 31, No. 2, pp. 181-192, 1998.

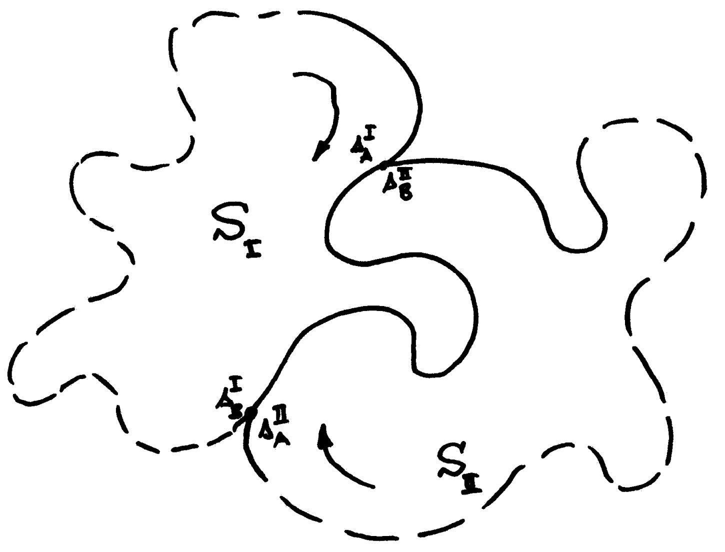


(a)

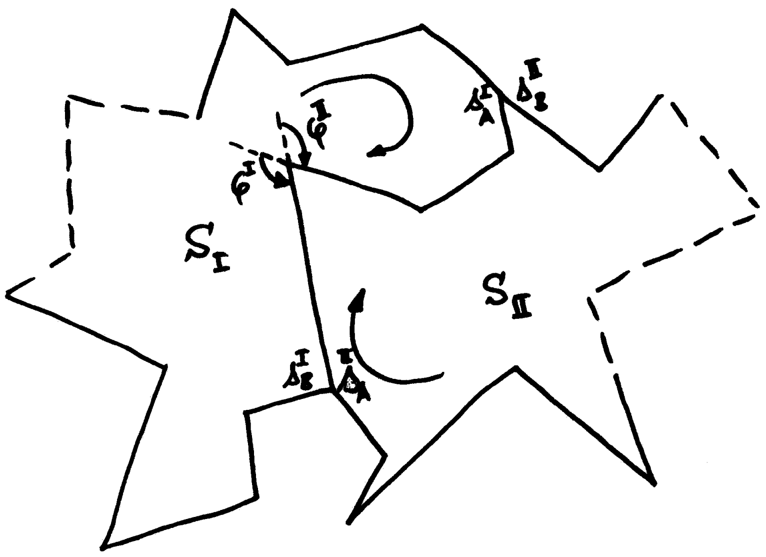


(b)

Figure 3: (a) Euclidean invariant boundary signatures for shape description (smooth case). (b) Polygonal case.



(a)



(b)

Figure 4: (a) Smooth shape dockings (b) Polygonal shapes.

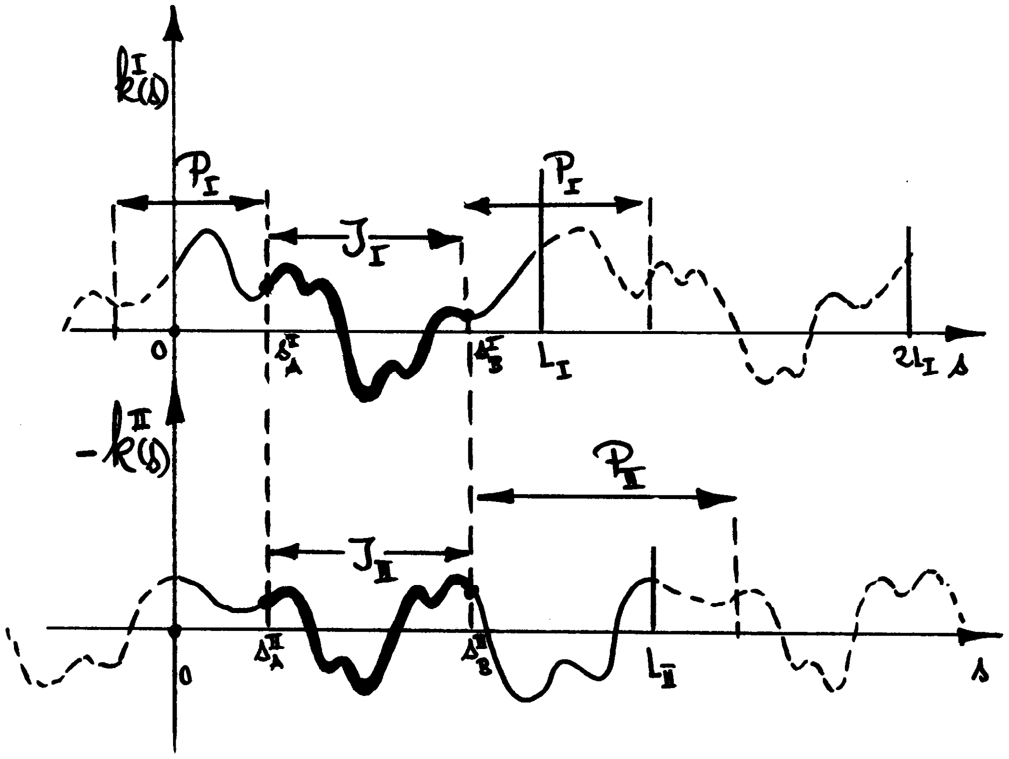
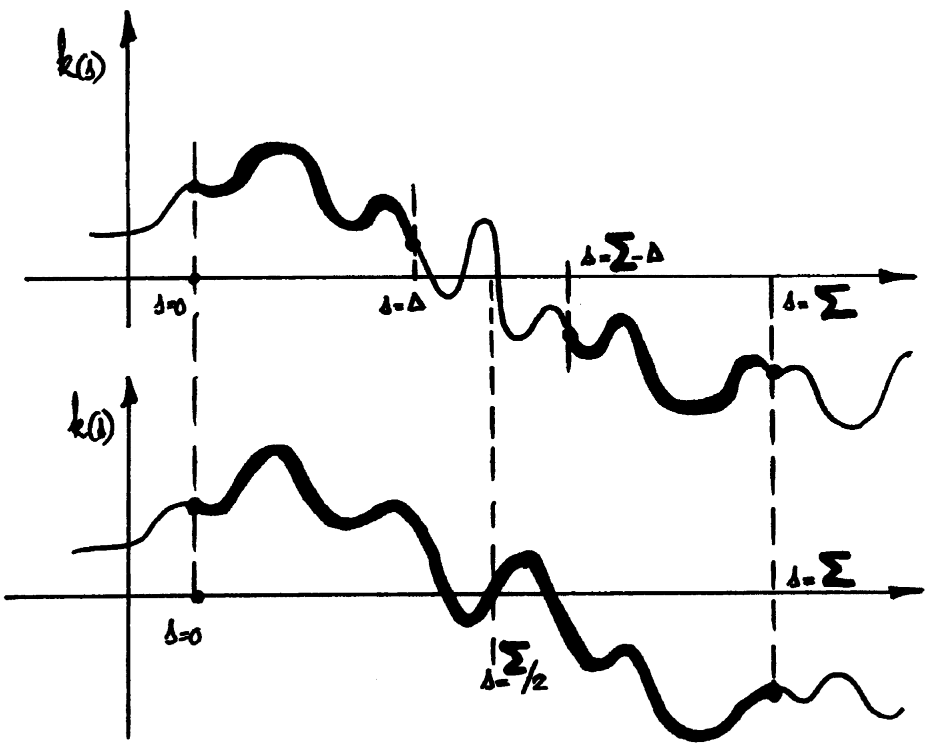


Figure 5: The signatures of two shapes S_I and S_{II} showing the docking portions $\mathbf{J_I}$ and $\mathbf{J_{II}}$ and the portions $\mathbf{P_I}$ and $\mathbf{P_{II}}$ that will make the boundary of the joint shape.

(a)



(b)

Figure 6: Self docking dichotomy: (a) disjoint matching intervals (b) same boundary portion matching.

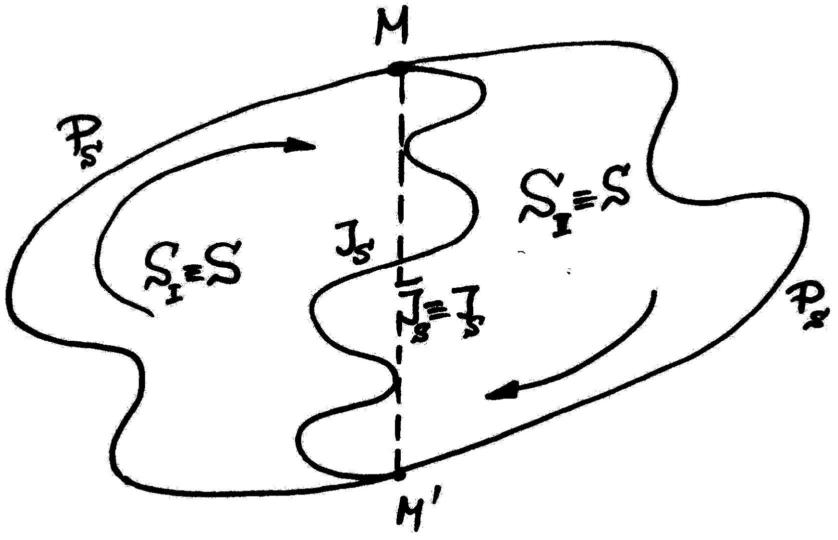


Figure 7: "Self-docking" over the same boundary portion of the two identical shapes S and S. The self docking boundary portion $\mathbf{J}$ is completely hidden inside the composite shape, whose boundary is of the form $\mathbf{P_s P_s}$, where $\mathbf{P_s}$ is the free portion of the boundary of S : $\mathbf{P_s J_s}$.

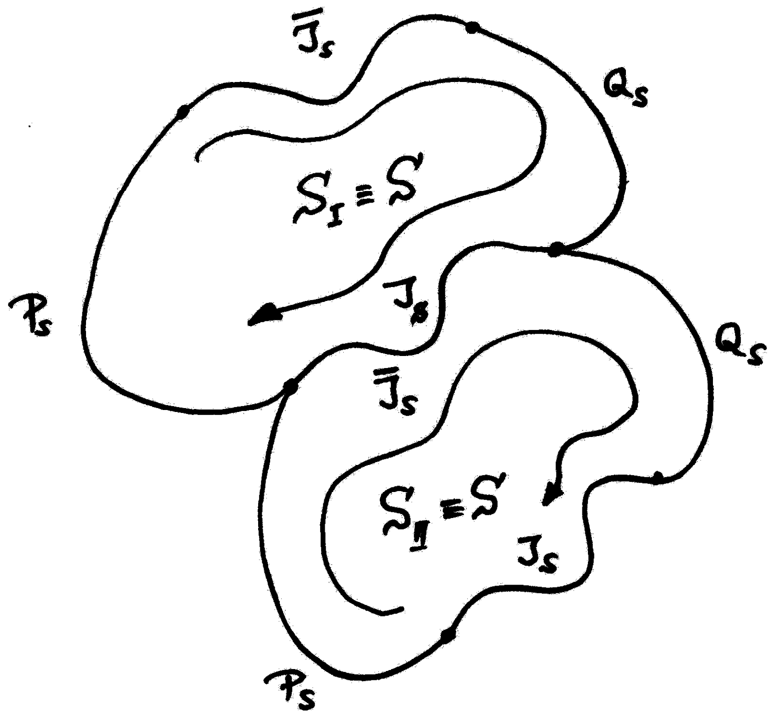
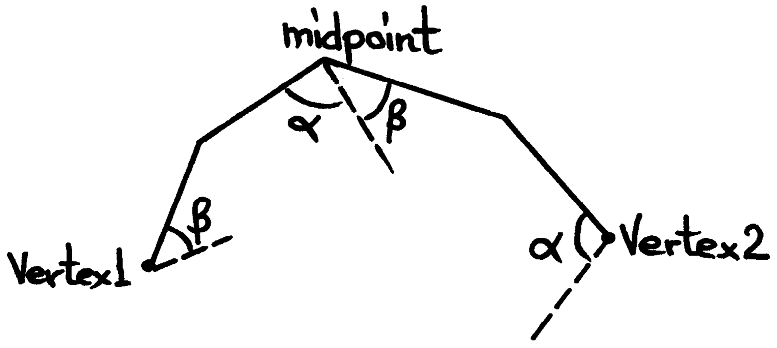
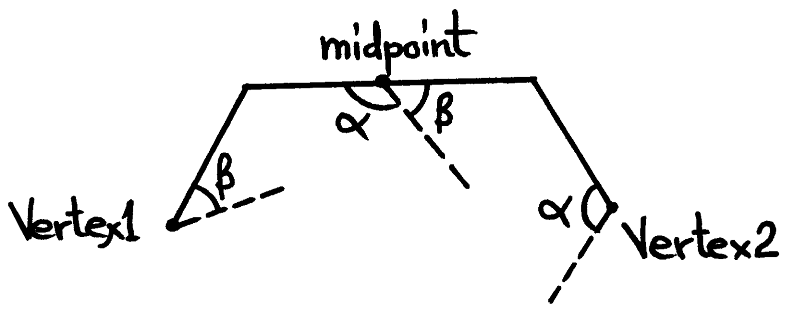


Figure 8: "Self-docking" over disjoint boundary portions of the two identical shapes $S_I \equiv S_{II} \equiv S$. The "self-docking" portions $\mathbf{J}$ and $\bar{\mathbf{J}}$ are visible in the composite shape whose boundary has the form $\mathbf{P_s \bar{J}_s Q_s Q_s J_s P_s}$ where the boundary of S is $\mathbf{P_s \bar{J}_s Q_s J_s}$.



(a)



(b)

Figure 9: **PP** segment candidates and angle conditions.

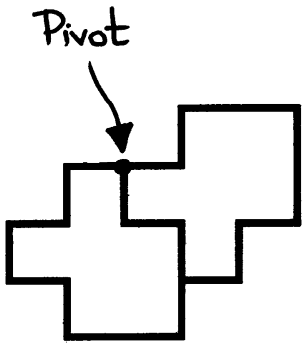
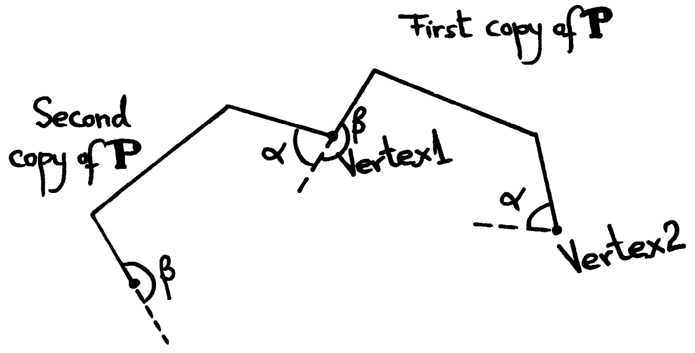
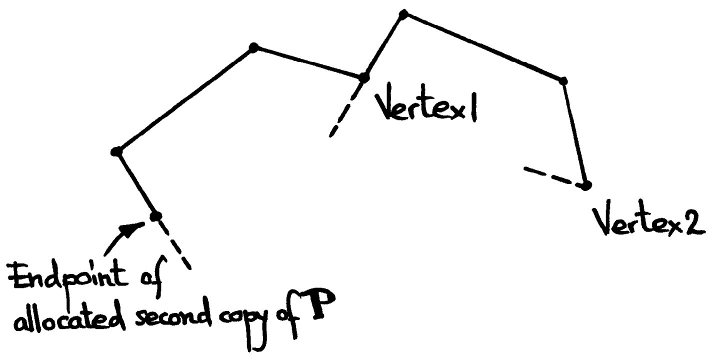


Figure 10: A pivot vertex constituting a trivial **PP** segment.



(a)



(b)

Figure 11: The case where one end of **P** is inside an edge.

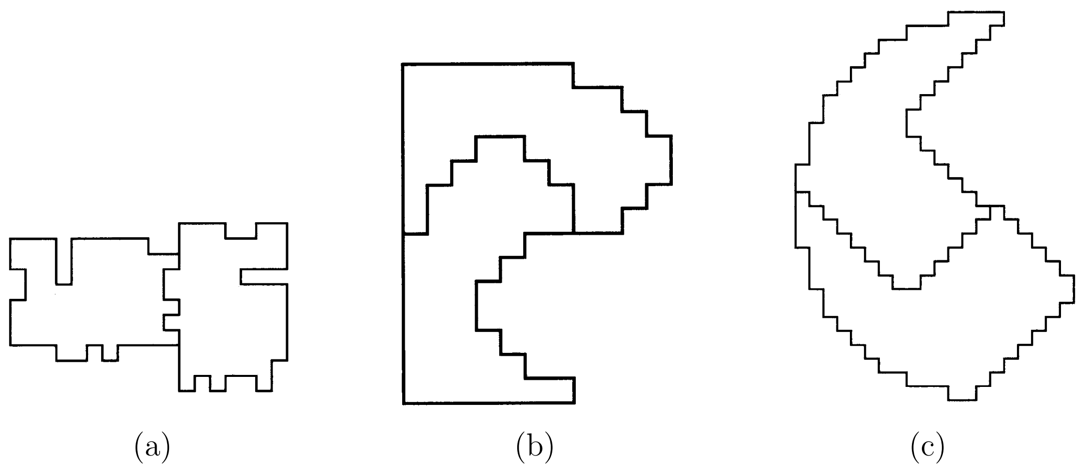


Figure 12: Some interesting pixelized crazy cuts.