Fall 2015

Final Project Report

Kiran Nasim, Yeojin Kim *Advance Algorithm Programming*

1 Summary of Fold and Cut

The fold and cut theorem says that any shape with straight sides can be folded flat and created with a single straight cut.

1.1 Details of Straight-Skeleton Method

The algorithm by Demaine and Lubiw[4] computes a crease pattern that produces a flat origami which can be created with one cut. Fold and cut method introduced by them, involves three main steps,

1. Computing straight-skeleton
2. Computing perpendiculars
3. Deciding Mountain/Valley fold

The straight skeleton is the continuous shrinking processing of edges the shrunk edges remain parallel to the original edges. Trajectory of the vertices as we shrink the faces is to be noted and when any of these moving vertices collides with a non-adjacent edge, the polygon is split in two and the shrinking process continues for separated polygons. The shrunk edges remain parallel to the original edges until edges shrink to zero length.

1.2 Limitations of Straight-Skeleton Method

Although the method works well for any straight line drawing with or without holes, closed or open, convex or non-convex but the crease pattern can be very complex even for simple line drawings. The method works for any type of straight line drawing but at present there is only one implementation present for fold n cut.

1.3 Comparison between different fold and cut methods

There are two general methods at present to solve fold and cut problem, straight-skeleton and disk-packing method. Straight-skeleton method is related to TreeMaker but TreeMaker method works only for convex polygons where as straight skeleton method can solve non-convex and disconnected polygons also.

Disk-packing method is an inspiration from fold and cut method where for each vertex a disk is created centered at the vertex, the disks should be non-overlapping. In this way the desired cuts can be obtained by combing together all the radii of disks. There may be a need of adding more edges between center of touching disks to have triangles and quadrangles which can be folded. The method is still unimplemented completely and it is not shown that how to transform the crease pattern into flat folding so it is still unknown if shape is actually flat foldable to be created by one cut. Moreover the disk-packing method has impractical computation time.

2 Implementation Details

2.1 Straight-Skeleton

In this implementation CGAL library is used for computing straight-skeleton for polygons with no holes. CGAL package represents a straight skeleton as a specialized Halfedge Data Structure (HDS). A halfedge is actually a handle wrapping the body which implements either an edge or a bisector. The method works for both convex or non-convex polygons.

Roughly, the algorithm can be written as follow [2],

1. *Initialization*
   Compute initial angular contour bisectors.
   Compute initial Events, placing them in a priority queue ordered by their instants
      The Split Events for all reflex contour bisectors.
      The initial set of Edge Events for all consecutive intersecting contour bisectors.
2. *Propagation*
   Process each Event in turn.
      This processing generates new bisectors which in turn produce new EdgeEvents.

The straight-skeleton data structure is shown in figure 1. The vertices of the contours can only by traced from the straight skeleton data structure by circulating the border halfedges, and the resulting vertex sequence will be reversed w.r.t the input vertex sequence. The defining contour halfedges and incident halfedges around a vertex can be traced using the circulators provided by the vertex class [5].

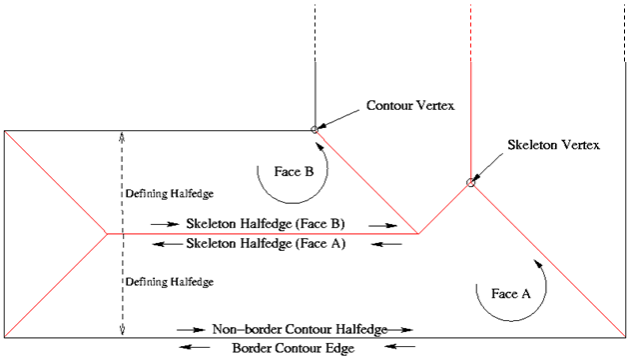
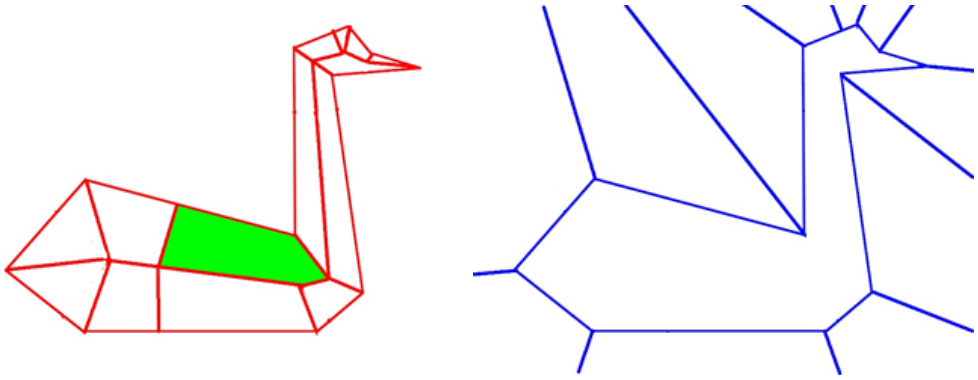


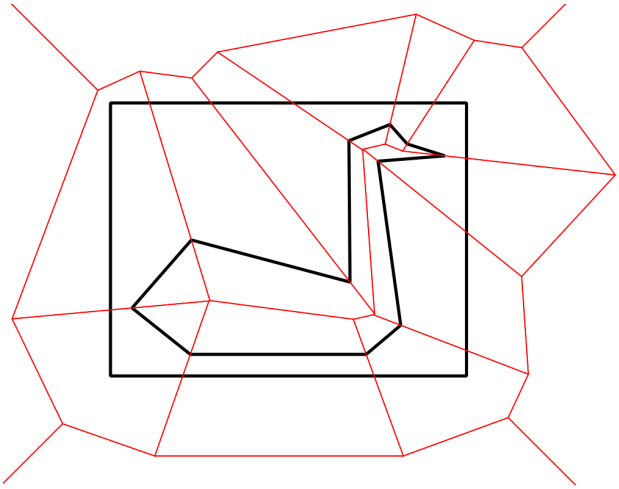
Figure 1: Straight Skeleton Data Structure

The data structure is used to create new data structures for the next step i.e. computing perpendiculars. Qt library is used to draw the results graphically.
We have used CGAL functions *create_interior_straight_skeleton_2*() and *create_exterior_straight_skeleton_2*() to have interior and exterior skeletons, shown in figure 2.



(a) Interior(showing s skeleton face)

(b) Exterior



(c) Interior+Exterior

Figure 2: Straight Skeleton

2.2 Perpendiculars

The straight line is not foldable by itself since the vertices can have degree three. Therefore, more crease lines are needed and to have these lines, from each vertex of the straight skeleton, we shoot a ray perpendicular to each cut edge in the face associated with vertex, this ray should reflects if it hits a skeleton edge but passes straight through a cut edge.

There are two types of perpendiculars.

1. *Real Perpendicular:* The perpendiculars that are incident to a skeleton vertex are real perpendiculars.
2. *Imaginary Perpendicular:* All other perpendiculars belong to this category.

The algorithm is as follows,

- o Let p be a point in a closed skeleton face f
- o Let l be a line containing p and perpendicular to the cut edge contained in f
- o Let m be $(p \cap l)$ that touches f
- o The perpendicular of p contains m and the perpendiculars of the end points of m

The data structure of generating perpendiculars is shown in Fig. 3. The green line is flow to generate real perpendiculars and the orange line is flow to generate imaginary perpendiculars. We make the structure which contains connected information between interior skeleton and exterior skeleton, called bridging graph.

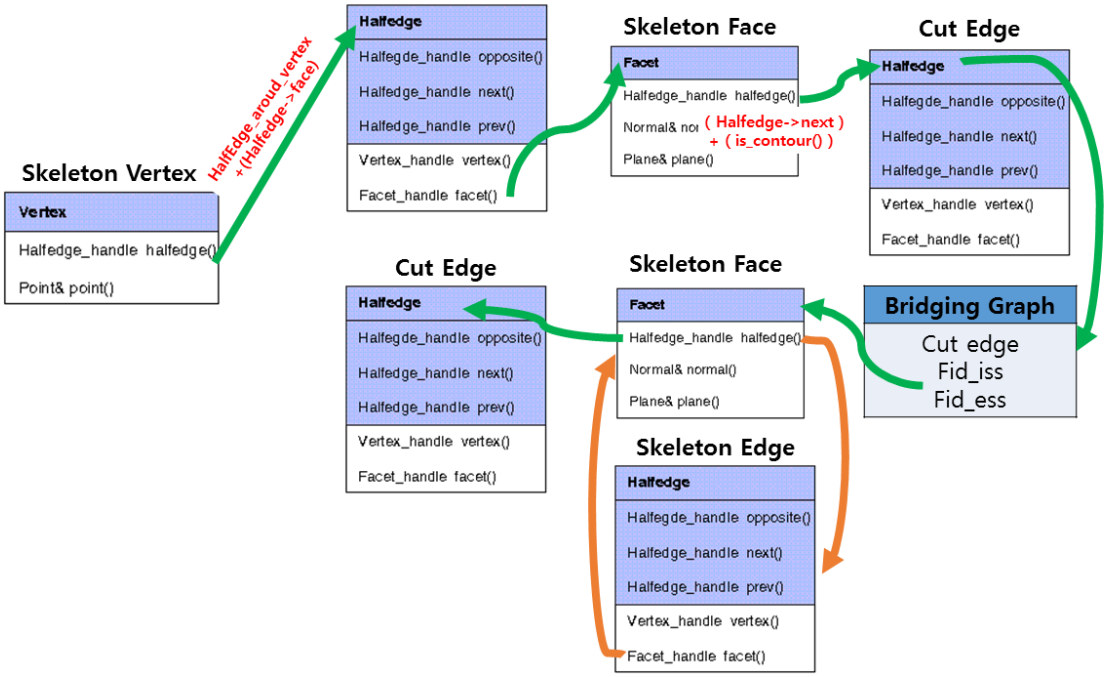


Figure 3: Computing perpendiculars.

As an example, straight skeleton drawing and perpendiculars are shown for a simple square shape in fig 4.

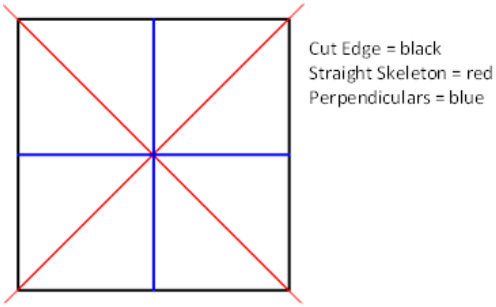


Figure 4: Straight Skeleton and perpendiculars for a square shape

It can be seen that perpendiculars originate from straight skeleton vertex and passes straight through cut edge, in case if perpendiculars hit another skeleton edge they reflect perpendicularly again.

2.3 Mountain and Valley

A folding angle of π $(-180°)$ is called a Mountain and a folding angle of $-\pi$ $(-180°)$ is called a Valley.
These are specific assignments to edges in the crease pattern to assign mountain or valley fold [3]:

- All cut edges are Valley.
- Skeleton edge is mountain if it bisects convex angle, valley if it bisects reflex angle.
- Perpendiculars start out as valley if incident to a convex vertex, else mountain but they alternate between mountain and valley every time they bounce.

We have used CGAL *right_turn* function to decide mountain or valley crease. As shown in figure below, [1]

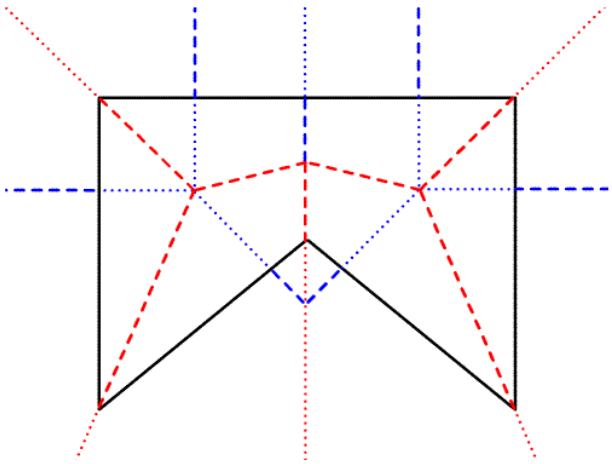


Figure 5: Assigning Mountain(dashed)/Valley(dotted)

3 Example Outputs

These are example outputs of our program. From Fig .6 to Fig .9, (a) represents the original input polygon. The straight skeleton of given polygon is marked as red line in (b). (c) is the result of computing perpendiculars, which is marked as blue line. For perpendiculars, we should remove some duplicated edges due to numerical error and the result is shown as (d). More details about deduplication is in next section. (e) shows the crease pattern of straight skeleton line. The green lines are mountain and the yellow lines are valley.

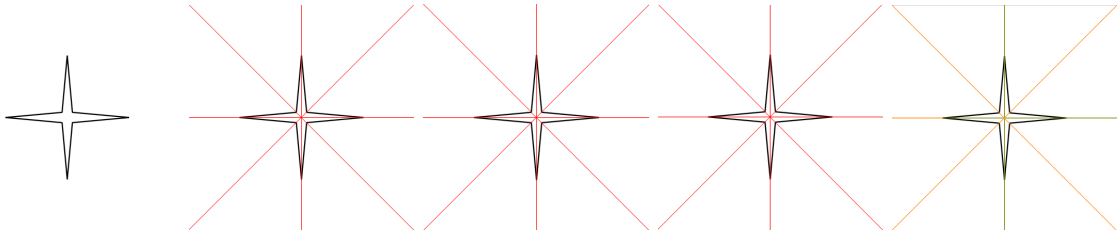


Figure 6: Simple polygon example 1 : star

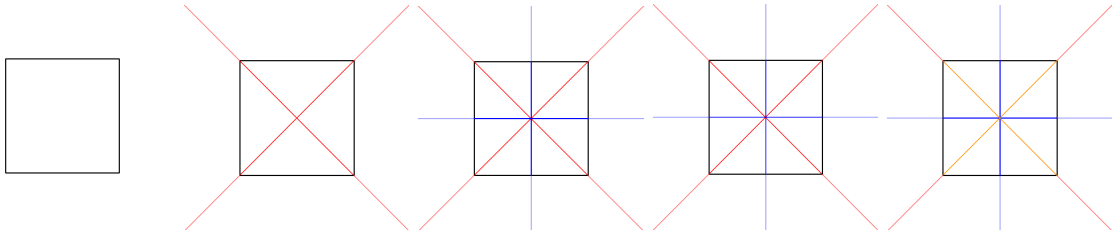


Figure 7: Simple polygon example 2 : square

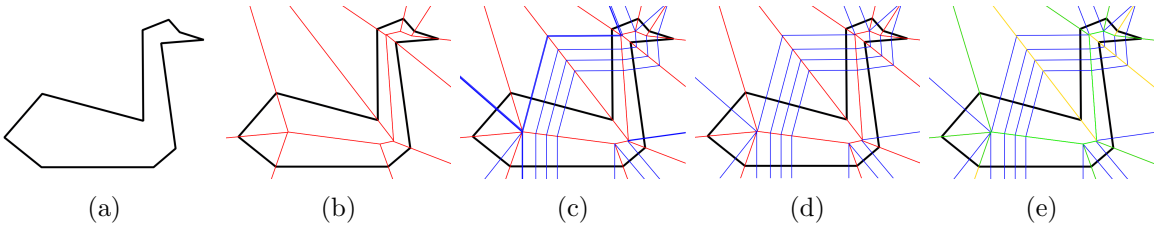


Figure 8: complex polygon example 1 : swan

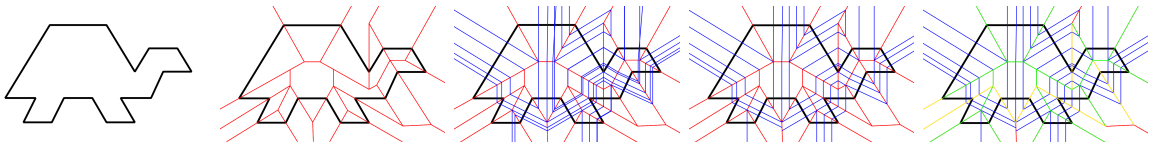
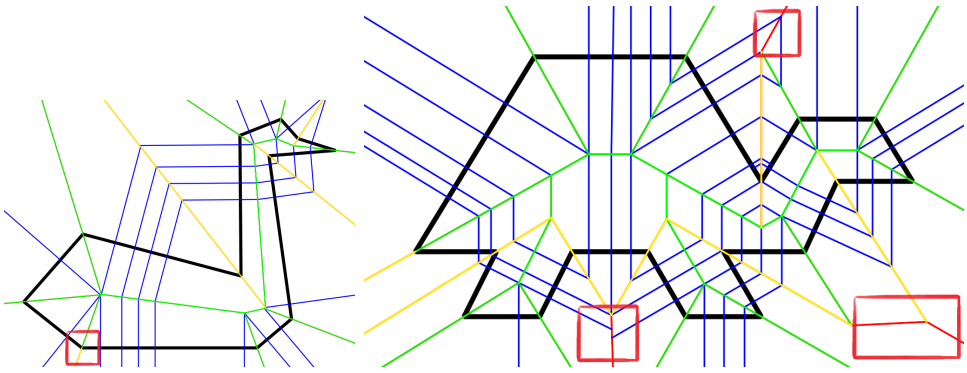


Figure 9: complex polygon example 2 : turtle

We implemented crease pattern correctly only with interior skeleton. In case of exterior skeleton, some creases are wrong or missing(Marked as red square in Fig. 10.)



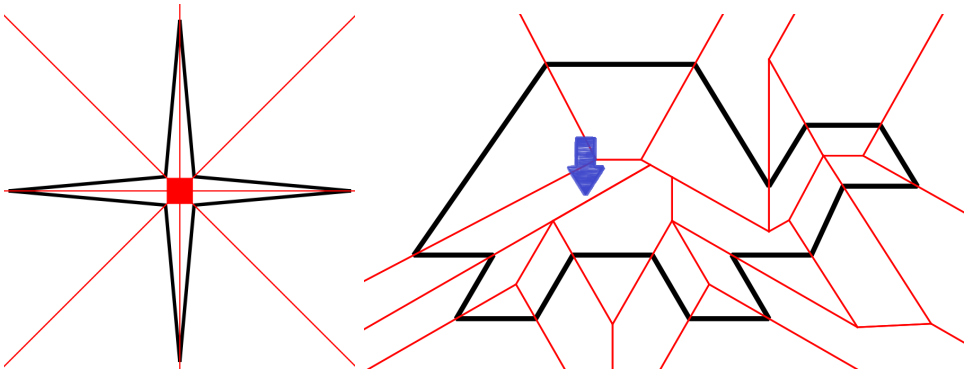
(a) (b)

Figure 10: crease pattern of polygons. Green lines mean mountain and yellow lines mean valley.

4 Known bugs/limitations

The program requires CGAL and boost library static compilation. Qt is required to view the results. The current implementation uses visual studio 2010 platform. The input polygons are drawn in clockwise order and the implementation works for convex or non-convex shapes with no holes.

Our program have bugs and limitations due to several fundamental problems. The first problem is about computing straight skeleton. According to [5], skeleton vertex will be merged if two points are very closed to each other, but it turns out that some points are not merged. The star shape in Fig. 11.(a) is symmetric with respect to vertical half line, but there are two skeleton vertex(marked as red square) and the distance between them is less than 1e-31. Furthermore, if we change some length of edges keeping symmetry, straight skeleton function returns weird results such as the edge pointed by blue arrow in Fig. 11.(b).



(a) (b)

Figure 11: Bugs of straight skeleton.

This problem also causes incorrect results in computing perpendiculars. The perpendiculars pointed by yellow arrow in Fig. 12.(a) and 12.(c) are duplicated many times. In addition to unmerged points, the numerical error is accumulated as the algorithm finds new perpendiculars recursively.

Therefore, we should erase perpendiculars which is thought to be the same perpendicular. Fig. 12.(b) and Fig. 12.(d) are results of deduplication. However, this measure causes other problems such as face which has two perpendiculars or unconnected perpendiculars (Marked as green circle in Fig. 12.(d)). This is because it is also hard to determined which perpendiculars are same or not with numerical errors.
Unmerged vertex also obstruct the algorithm to obtain accurate vertex information. Straight skeleton vertex can have the odd number of edges around the vertex after generating perpendiculars(Marked as green circle in Fig. 12.(b)), but it is hard to obtain the total degree of the skeleton vertex.

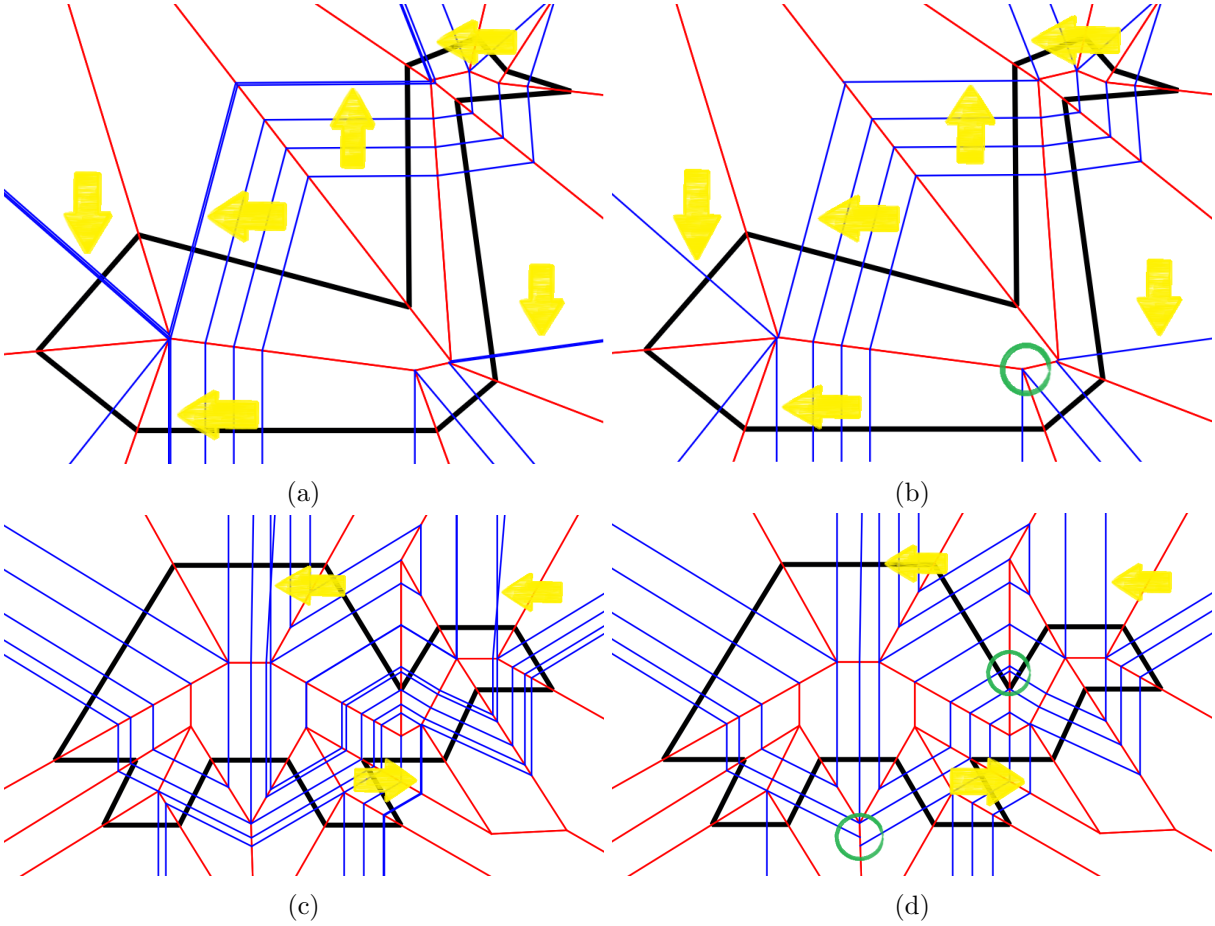


Figure 12: Bugs of perpendiculars.

The other problem is data structure which only bridges few primitives. Our approach only connects interior skeleton face and exterior skeleton face which are incident to cut edge. It increases the amount of computation to access data, because we need to take a long way around to access specific primitives. If we had bridging all the primitives in exterior skeleton and interior skeleton, we could have reduced the computation. Imprecise function in CGAL increases the computation, too. Moreover, if we had considered the order of primitives in CGAL, we could have expected less computational time.

References

[1] Eric Biunno. The origami polygon cutting theorem, mcgill university, fall 2003. 2.3

[2] Fernando Cacciola. A cgal implementation of the straight skeleton of a simple 2d polygon with holes. *SciSoft*. 2.1

[3] Erik Demaine. Lecture: Geometric folding algorithms: Linkages, origami, polyhedra (fall 2010). 2.3

[4] Martian L. Demaine Erik D. Demaine and Anna Lubiw. Folding and cutting paper. *Deaprtment of Computer Science, University of Waterloo, Canada*. 1.1

[5] CGAL Manual. Cgal 4.7 - 2d straight skeleton and polygon offsetting. 2.1, 4