Dimension-reduction technique for polygons

Yogesh H. Kulkarni* and Anil Sahasrabudhe

College of Engineering Pune,
Shivajinagar, Pune 411005, India
Email: kulkarniyh12.mech@coep.ac.in
Email: director@coep.ac.in
*Corresponding author

Mukund Kale

Siemens PLM Software,
Hinjewadi, Pune 411057, India
Email: mukund_kale@hotmail.com

Abstract: Applications such as pattern recognition, shape matching, finite element analysis need lower dimensional representation of a higher dimensional shape. Skeleton provides such one dimensional representation for a 2D or 3D shape. It can be computed for various types of inputs, such as images, solids, sketches. Although there are various methods available for computation of the skeleton, their appropriateness depends on the requirement posed by their applications. Some focus on exact computation, some are approximate, some aim at faithful backward-reconstruction; whereas some focus at proper representation for human perception. This paper focuses on a method for creation of a particular type of skeleton, called *Midcurve* (a skeleton that lies at the midway of a shape) for planar polygon. This method is based on the Divide-and-Conquer strategy and works in two phases – decomposition and midcurve creation. Towards the end of the paper some test case results are presented along with the conclusions.

Keywords: midcurves; polygon decomposition; skeleton.

Reference to this paper should be made as follows: Kulkarni, Y.H., Sahasrabudhe, A. and Kale, M. (2017) 'Dimension-reduction technique for polygons', *Int. J. Computer Aided Engineering and Technology*, Vol. 9, No. 1, pp.1–17.

Biographical notes: Yogesh H. Kulkarni is a PhD student in the Department of Mechanical Engineering, College of Engineering Pune, India.

Anil Sahasrabudhe received his PhD degree from Indian Institute of Science, Bangalore and is currently the Director at the College of Engineering Pune, India.

Mukund Kale received his PhD degree from Indian Institute of Technology, Bombay and is currently working as a Manager at Siemens PLM Software India.

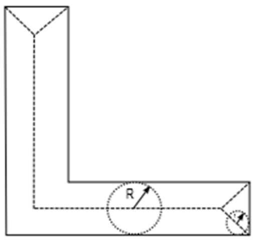
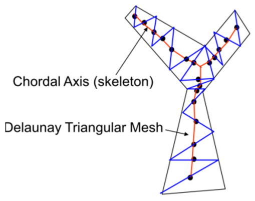
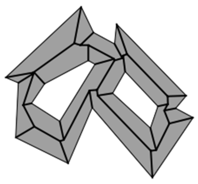
This paper is a revised and expanded version of a paper entitled 'Midcurves generation algorithm for thin polygons' presented at Emerging Trends in Engineering and Science 2014 (ETES 2014), Asansol, India, 30–31 January 2014.

1 Introduction

A variety of applications need a lower dimensional, concise and faithful representation of the original parent geometry. A skeleton is such an entity which represents the shape of its parent object. It being simpler than the parent object, applications like pattern recognition, approximation, similarity estimation, collision detection, animation, matching and deformation can be performed efficiently on it than on the parent object.

Approaches such as medial axis transform (MAT), chordal axis transform (CAT), thinning etc. are used to compute skeletons. Table 1 briefly summarises some of these along with their strengths and weaknesses. In this paper, terms like 'skeleton', 'midcurve' and 'medial' are used synonymously.

Table 1 Current medial computation methods (see online version for colours)

Method	*Figure*	*Description*	*Problems*
MAT (Ramanathan and Gurumoorthy, 2004)		The loci of centres of the maximal circle in 2D, the ball in 3D. The radius function denotes the thickness.	Unwanted branches at the corners. Sensitive to perturbations.
CAT (Quadros, 2008)		Triangulates the profile. Joins midpoints of the chords to form a medial.	Gaps at end. Expensive triangulation.
Straight skeleton (Haunert and Sester, 2004)		Starts thinning from the boundary towards inside; stops at the meeting points	Bisectors are not equidistant.

This paper focuses on 2D planar sketch profiles with an assumption that curved shapes can be converted to polygonal shape by faceting. Divide-and-Conquer has been used where decomposition partitions the given shape into sub-shapes and skeletonisation creates the midcurves in simpler sub-polygons. At the end, individual midcurves are extended and joined to form a continuous set of curves mimicking the parent shape.

2 Related work

Theoretical biologist, Blum (1967) did some pioneering work in the field of skeletonisation, with the medial axis of the shape. Survey papers, such as Attali et al. (2004), Lam et al. (1992) and Kulkarni and Deshpande (2010) have enumerated various approaches used in the skeleton creation. Decomposition of planar shapes into regular (non-intersecting) and singular (intersecting) regions and its application to skeletonisation has also been widely researched (Rocha, 1999).

Ramanathan and Gurumoorthy (2004) pointed out the difference between MAT and the midcurve. MAT does not truly reflect the local topology of the shape due to additional branches at the corners, but a midcurve does to a large extent.

Rocha (1999) and Rocha and Bernardino (1997) used both decomposition and skeletonisation for character recognition in images. From the results presented, it appears that junctions such as L and T were far from being well-connected.

Keil's algorithm (Keil and Snoeyink, 1994) finds all possible ways to remove reflexity of these vertices, and then takes the one that requires fewest diagonals.

Lien and Amato (2006) decomposed polygons 'approximately' based on iterative removal of the most significant non-convex feature.

Some of the other methods of polygon decomposition are based on use of:

- curvature to identify limbs-necks
- axial shape graph
- events occurring during computation of straight skeleton
- Delaunay triangulation.

Many of these methods need quite a bit of pre and post processing to take care of boundary noise (Lien and Amato, 2006).

Zou and Yan (2001) partitioned a shape by Delaunay triangulation, identified regular and singular regions and then created skeletons.

A formal definition of midcurve is not available (Ramanathan and Gurumoorthy, 2004). For restricted scenarios, such as, given two curves, a midcurve can be defined as a curve equidistant from the given curves (Fischer et al., 1999).

Following paper improves upon polygon decomposition algorithm by Bayazit (2013), and then uses its output for midcurve creation.

3 Proposed method

2D polygonal profile is decomposed into sub-polygons. Being simpler, sub-polygons make midcurve creation more deterministic than computing it for the whole profile at once.

3.1 Decomposition

Polygons come with different types of variations. They can be simple/self-intersecting, with/without holes, concave/convex, etc. Decomposition of a polygon into convex subpolygons can be done by dividing at all reflex (concave) vertices. Generally the criterion for decomposition is to produce a minimum number of convex components or to minimise the total boundary of these components. Within the minimum component criterion methods further classification could be done based on whether or not Steiner points (brand new, non-polygonal vertices) are allowed.

This work focuses on a special type of shapes, elongated polygons. Typical examples are alphabets, thin-wall profiles with constant thickness.

Important points to note are (Bayazit, 2013):

1 a polygon can be broken into convex regions by eliminating all reflex vertices

2 a reflex vertex can only be removed if the diagonal connecting to it is within the range given by extending its neighbouring edges; otherwise, its angle is only reduced.

3.1.1 Preliminaries

1 *Polygon:* A polygon P of n vertices is defined as:

$$P = \{P_0, P_1, \ldots, P_{n-1}\}$$

where the vertices are in counter-clockwise (ccw) order. Polygon P can also be defined in terms of a set of connected edges as:

$$P = \{\overline{P_0P_1}, \overline{P_1P_2}, \ldots, \overline{P_{n-1}P_0}\}$$

In case of shapes which have nonlinear elements, they are faceted and brought in terms of connected lines.

2 *Simple:* A polygon is simple if none of the edges intersect other edges anywhere else other than the shared endpoints of adjacent edges.

$$\forall \overline{P_iP_j}, \overline{P_kP_l} \in P, \begin{cases} \overline{P_iP_j} \cap \overline{P_kP_l} = \phi, \ j \neq k \\ \overline{P_iP_j} \cap \overline{P_kP_l} = P_k, \ j = k \end{cases}$$

3 *Diagonal:* P_iP_k is a *diagonal* of P. In other words, a *diagonal* is just a line segment between two vertices that only touches the interior of the polygon.

4 *Area: Area* formed by three vertices in order (P_{j-1}, P_j, P_{j+1}) or two consecutive *edges* $(\overline{P_{j-1}P_j} \cap \overline{P_jP_{j+1}})$ is a signed quantity, which is given by:

$$Area = P_{j-1}.X\left(P_j.Y - P_{j+1}.Y\right) + P_j.X\left(P_{j+1}.Y - P_{j-1}.Y\right) + P_{j+1}.X\left(P_{j-1}.Y - P_j.Y\right).$$

5 *Left:* P_{j+1} is *left* of $\overline{P_{j-1}P_j}$ if *area* (P_{j-1}, P_j, P_{j+1}) > 0.

6 *Right:* P_{j+1} is *right* of $\overline{P_{j-1}P_j}$ if *area* (P_{j-1}, P_j, P_{j+1}) < 0.

7 *Collinear:* P_{j+1} is *collinear* with $\overline{P_{j-1}P_j}$ if *area* (P_{j-1}, P_j, P_{j+1}) = 0.

8 *Reflex:* Let P_{j-1}, P_j, $P_{j+1} \in P$, if the interior $\angle P_{j-1}, P_j, P_{j+1}$ is greater than π then P_j is a concave or *reflex* vertex. *Area* < 0.

9 *Intersect:* For $\overline{P_iP_j}$ to intersect $\overline{P_kP_l}$, either of P_k, P_l should be on *left* of $\overline{P_iP_j}$ and the other vertex should be on *right*. Intersection could be of *line* type where extended intersection can be calculated or of *segment* type where only internal (within the range of either of the segments, $\overline{P_iP_j}$ or $\overline{P_kP_l}$) intersections are returned.

10 *Visibility/can-see:* P_k is visible from P_i if $\overline{P_iP_k}$ is a *diagonal* of P.

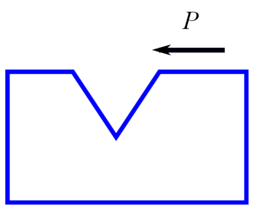
Algorithm 1 Polygon decomposition

```
Require: 2D planar polygon represented by a list of vertices
Ensure: Vertices in counter-clockwise direction
  while End of vertices list has not reached do
    Get the current vertex.
    if the current vertex is a Reflex vertex $R$ then
      Extend the edges incident at $R$ until they hit an edge
      if extension line and Polygon side are collinear then
          Find the closest point which is not internal to the extension line
        end if
        if there are no vertices to connect to then
          choose a point in the middle
        else
          Find the best vertex $Q_i$ within the Range, to form the partitioning chord
          Make sure $Q_i$ is visible from $R$
        end if
      end if
  end while
  Split the polygon at the cutting chord (line $RQ_i$)
  Send individual sub-polygons to the same process recursively till there are no reflex vertices left.
  Identify polygons with more than 4 distinct (ignoring sub-segment overlapping chords) sides. Triangulate them with Constrained Delaunay Triangulation (CDT)
```

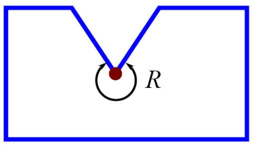
3.1.2 Steps

Let P be a simple polygon. The partitioning of P is defined by the decomposition of P into partitions of non-overlapping sub-polygons by adding internal *diagonals* between vertices P_i or by adding new (Steiner) vertices on *edges* $\overline{P_iP_j}$. Partitioning is continued till all possible cuts are made.

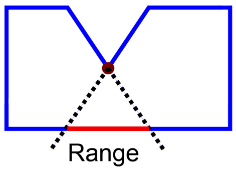
1 Go through all the vertices of the polygon one by one in counter-clockwise manner. Current vertex is called P_i.



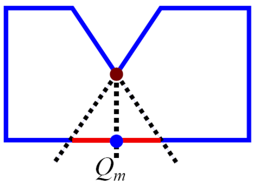
2 Check if P_i is a reflex vertex R



3 Extend lines incident at P_i (the line coming into P_i and going out of P_i) till they intersect remaining of the polygon, say at Q_1 and Q_2. Contour within Q_1 and Q_2 is called the *range*.



4 If there are no P_is within the *range* and if any of the *range* vertices are close to intersections, separate the triangle out. Else, create a new one at the middle of the contour. This newly created point Q_m is called Steiner point. RQ_m is the partition-chord to divide the polygon.



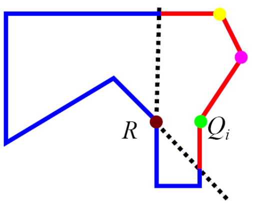
5 If there are a few vertices within the *range*, choose the best one based on following priorities:

a highest: closest reflex

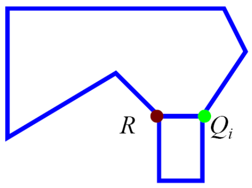
b medium: reflex

c low: closest.

Make sure that point Q_i is visible from the reflex point R. If it is not so, then this cut cannot be made. Choose the next best choice.

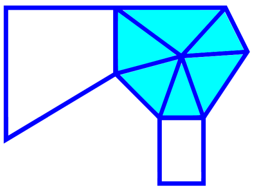


6 Once the vertex is chosen, say, Q_i, create a partition chord RQ_i and divide the polygon.



7 Send individual sub-polygons to the same process recursively till there are no reflex vertices left.

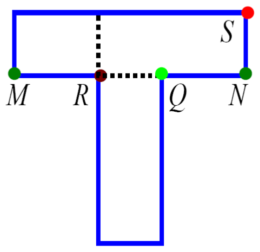
8 Identify polygons with more than four distinct (ignoring sub-segment overlapping chords) sides. Triangulate them with constrained Delaunay triangulation (CDT).



CDT takes care of remaining areas of the polygon.

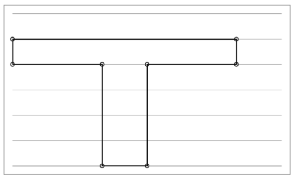
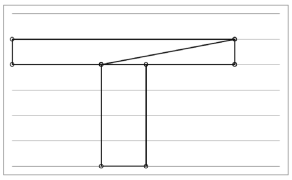
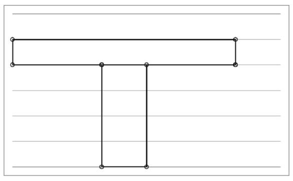
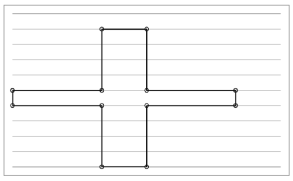
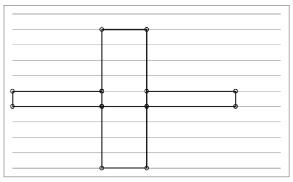
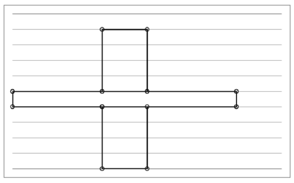
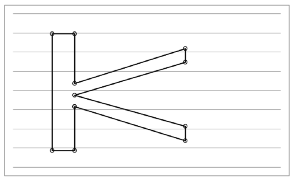
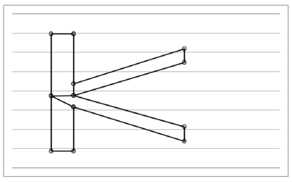
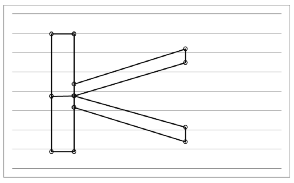
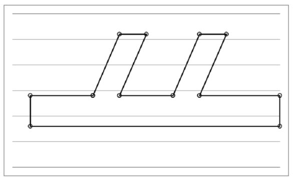
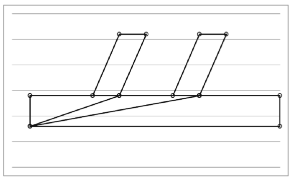
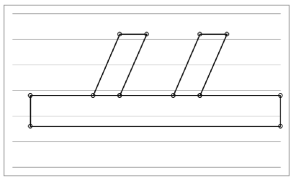
3.2 *Improvements over Bayazit's algorithm*

Algorithm 1 improves upon the Bayazit's (2013) algorithm in terms of expanding search to even include extreme vertices in the range, thereby giving minimal and elongated partitions. Midcurve is typically computed for a thin-elongated shape. This improvement results in the sub-polygons of necessary shape characteristics. If any incoming edge (*MR*) was hitting the end points of the test-line (*QN*) or was collinear, it (*Q*) was getting ignored in the existing algorithm (Bayazit, 2013). In that case the next closet vertex (*S*) was getting chosen. This was corrected in the proposed algorithm and the shorter cut (*RQ*) is done.



As the last step in the Algorithm 1 triangulates the remaining polygons, this algorithm guarantees presence of sub-polygons with 3 and 4 sides only. Further algorithms, like the one mentioned below Algorithm 2, need to enumerate a case possible only with triangles and quadrilaterals, making it deterministic. Table 2 demonstrates improvements over Bayazit's (2013) algorithm with examples. The resulting sub-polygons are sent for creating a connected midcurve.

Table 2 Improvement over Bayazit's (2013) partitioning algorithm

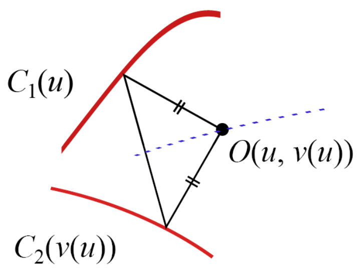
Shape	*Bayazit*	*Proposed*
		
		
		
		

3.3 *Midcurve creation*

Decomposition helped getting the sub-polygons, which are of primitive shapes and which are easier for midcurve creation compared to the original-whole shape.

3.3.1 *Preliminaries*

For the two curves $C_1(u)$ and $C_2(v(u))$, let there be a point $O(u_0, v(u_0))$ which is neither on the given curves, but is such that normals from both $C_1(u_0)$ and $C_2(v(u_0))$ meet at O. Length of the normal denoted by $\| C_1(u_0) - O(u_0, v(u_0)) \|$ in the neighbourhood of u_0 is the shortest distance.



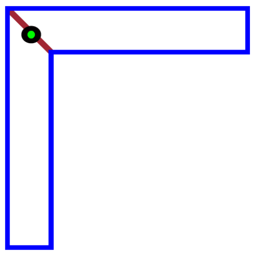
If, for all u there exists a point $O(\mathrm{u}, v(u))$ that is at equal (normal) distance from $C_1(u)$ and $C_2(v(u))$, we say that curves $C_1(u)$ and $C_2(v(u))$ are ideal-matched under the midcurve norm (Fischer et al., 1999).

Algorithm 2 Midcurve creation

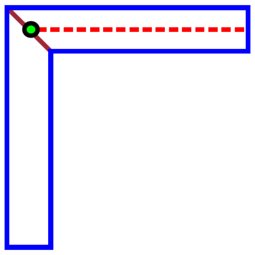
```
Require: A list of partitioned 2D Planar polygons, each represented by a list of vertices in
         counter-clockwise direction
  Find internal-common edges called chords
  Iterate over all polygons and create chords at Full or Partial overlaps
  while End of Polygons list has not reached do
      Get the current polygon P
      Get chords that are part of P
      Look at the various combinations based on number of sides and number of chords
      Generate midcurve
      Assign midcurve on relevant side of the chord
  end while
  Extend chords that are not connected with other neighboring chords
```

3.3.2 Steps

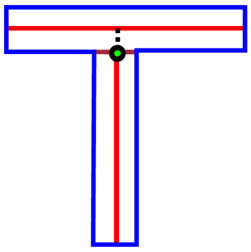
Partitioning: Decompose polygons into sub-polygons of primitive shapes using Algorithm 1. Each additional edge inserted during the decomposition is called as the 'chord'.



Generate midcurve for individual polygons taking chords into consideration. 'Thinness' is an important criterion in choosing midcurve for an individual shape. Midcurve is generated along longer-length and not across shorter-width.



In shapes like ‘L’, midcurves from both sub-polygons across the chord join together at a point, naturally. But in case of shapes like ‘T’, the horizontal midcurve does not connect with the common chord. In this case, one of the midcurves needs to be extended to join the other one.



Each such shape can create its own midcurve based on number of sides and also where the cutting-chords lie on this individual shape. Tables 3 and 4 have more details. A chord is a common interface-boundary shared between two sub-polygons. Each chord will have two sides owned by two different sub-polygons. Each sub-polygon needs to look at its own shape, slenderness and decide its own midcurve.

Table 3 Triangle cases of midcurve configuration (see online version for colours)

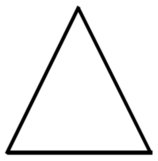
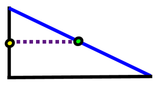
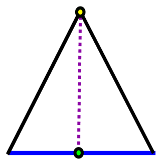
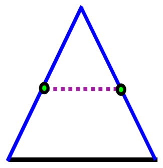
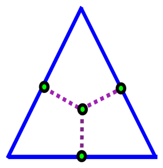
Shape	*Chords*	*Rule*	*Diagram*
Triangle	None	No midcurve	
	One	Join midpoint of the shorter side	
	One	Join opposite vertex if both the sides are of same length	
	Two	Join bisectors	
	Three	Join to centroid	

Table 4 Polygon cases of midcurve configuration (see online version for colours)

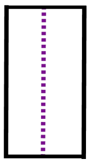
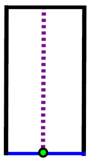
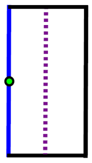
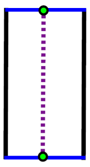
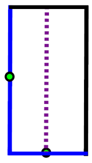
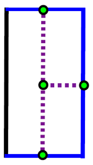
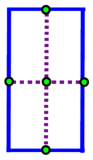
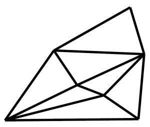
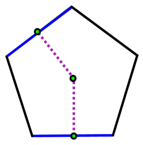
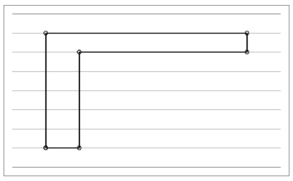
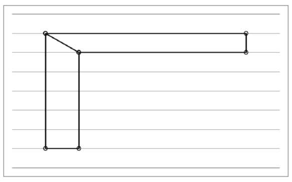
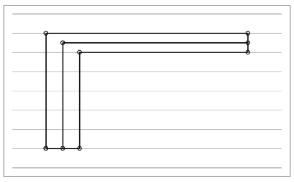
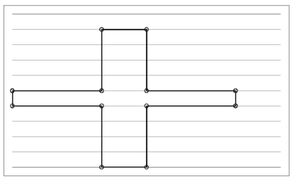
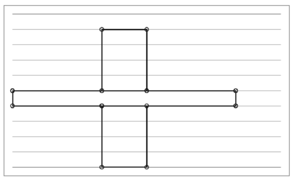
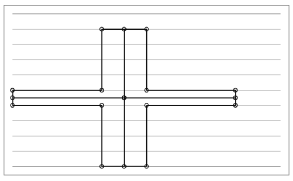
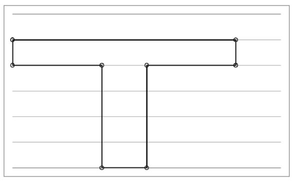
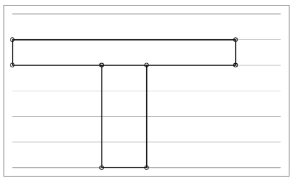
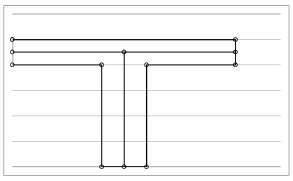
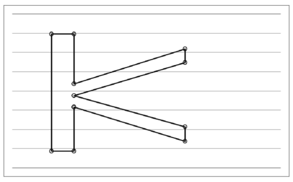
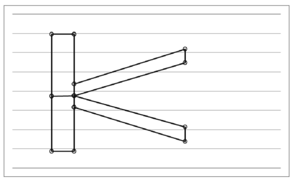
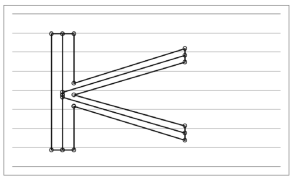
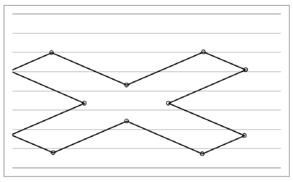
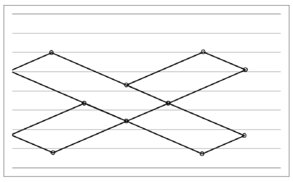
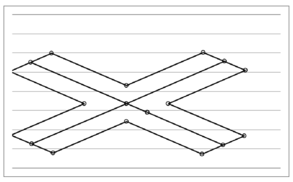
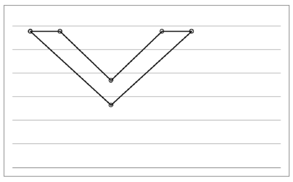
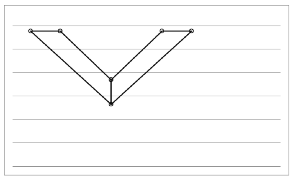
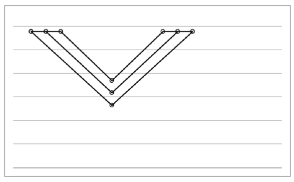
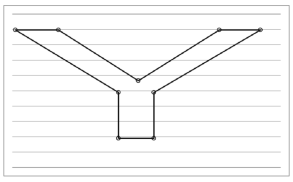
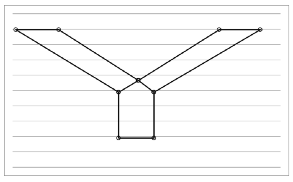
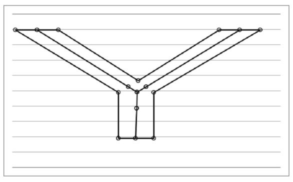
Shape	*Chords*	*Rule*	*Diagram*
Quad	None	Find the shortest side and create a midcurve in the direction average of both the adjacent sides	
	One (shorter)	Create a midcurve in the direction average of both the adjacent sides	
	One (longer)	None	
	Two (opposite)	Join midpoints	
	Two (adjacent)	Ignore the chord on the longer side and use the one-chord rule	
	Three	Join to the centroid	
	Four	Join to the centroid	
Polygon	None	If thin, then CDT or CAT	
	Any	Join to the centroid	

Table 5 Partitions and midcurve computation

Shape	*Partitions*	*Midcurve*
		
		
		
		
		
		
		

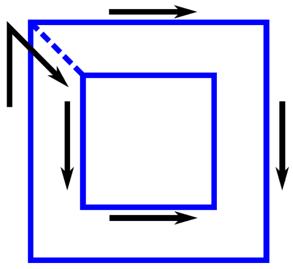
After creating individual midcurves, they all may or may not join at the chords. In case it does not join at any side of the chord, some extension has to be provided from the other side of that chord. Chords are processed to ignore the ones which have partial overlap with the sub-polygon sides or are collinear. This method gives cleaner (without branches) and a connected midcurve compared to the previously cited methods. A pseudo code for midcurve generation process is presented in Algorithm 2.

Many polygonal shapes can be broken down into *regular* and *singular* sub-shapes. *Singular* regions correspond to the intersections, whereas *regular* regions are the remaining parts of the shapes (Zou and Yan, 2001). Typical connection types (You, 2002) and their midcurves are presented in Table 5.

3.3.3 Limitations

In practical scenarios, input geometry may not be clean. It may have degeneracies, which need to be addressed or corrected for a desired output. The current algorithm has following limitations even for a clean input:

- *Faceting:* Even if the input is made up of curves, it is faceted to convert to a polygon. This approximation results in an approximation of the output as well. Thus, it does not give a true midcurve. It is possible to fit a curve through points of the midcurve segments, but it may not match the expected result.
- *Holes:* Input to the algorithm is in the form of a single ordered list of vertices, making a non-self-intersecting closed polygon.

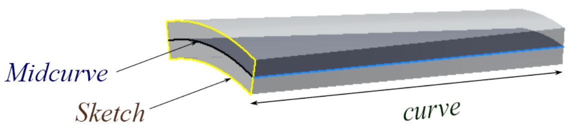


In case of holes, a connecting line is added to reach the internal holes to form a path, which does not traverse the given segments twice. If the number of holes are more, the situation becomes challenging to find the right connecting chords.

3.3.4 Usage

Midcurve has a wide variety of applications as it produces dimensionally reduced representation of the 2D shapes. Some representative usages are:

- *Midsurface:* For sweep based volumes, midsurface is nothing but sweeping of the midcurve of the sketch.



- *Pattern matching:* Instead of finding similarity between 2D profiles, it is easier to do the same with midcurve.
- *Shape retrieval:* For retrieving a particular 2D shape from the database, midcurve can be used as a signature, instead of string/tag based, thus making selection more deterministic.

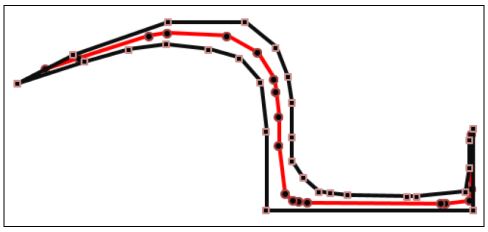
4 Results

The approach presented in this paper has been applied not just to the English alphabets but also to some typical real-life shapes.

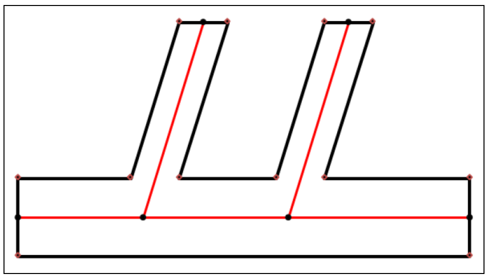
Curved profiles are faceted into polygons, making individual midcurve approximate. More accurate output can be achieved by lowering faceting-segments-size, but then the computation increases. Although MAT would give a *true centreline* output, it may have disadvantages such as extra branches, sensitivity to perturbations and so on.

Shapes mentioned below are taken from academic papers to demonstrate enhancements in the resultant midcurve.

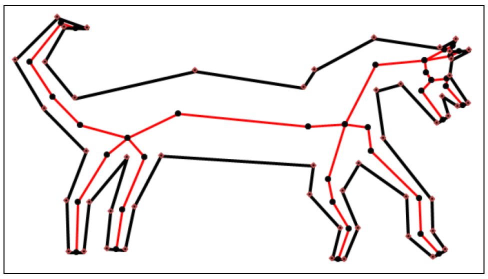
1 A glass profile was presented by Fischer et al. (1999). They had to carry out an additional loop-elimination step to remove the self-intersections that occurred after the first offset operation. This example has been particularly chosen for showcasing midcurve of a profile, having variable thickness. Midcurve is cleaner than whatMAT would generate, i.e. without branches at the corners.



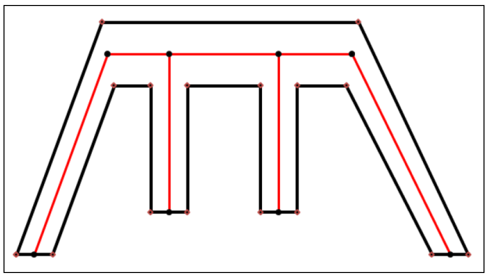
2 Profile presented by Ramanathan and Gurumoorthy (2004).



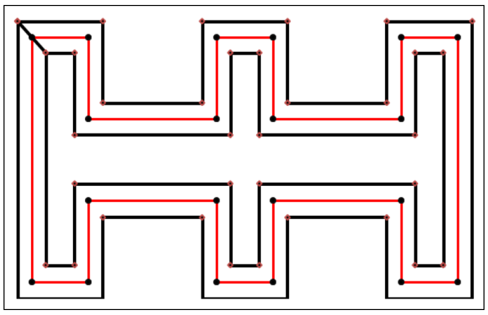
3 Pinto's (de Moura Pinto and Dal Sasso Freitas, 2009) example of planar polygonal Horse profile. All the branches at the corners have been eliminated.



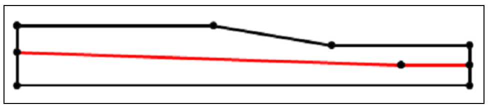
4 Sheen et al. (2010) took a typical plastic injection part profile. The approach presented here does not need any additional split operations.



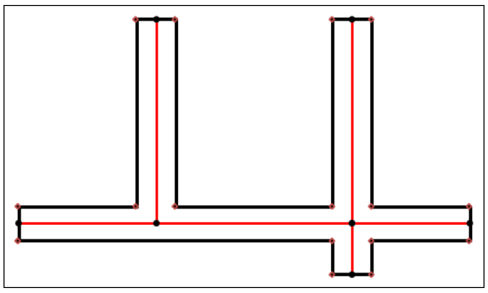
5 Sheen et al. (2010) also took the following profile to compare midsurface creation in two commercial CAD software packages. Both produced incorrect results.



6 Sheen et al. (2010) presented a couple of cases where midcurve could be ambiguous, due to step-ramp kind of a profile. Having a gradual change in the midcurve is desirable in this situation. The following output given by the current algorithm looks appropriate:



7 Woo (2013) presented the following example to demonstrate when extensions are appropriate. Extension of the midcurve of the first vertical bar up to boundary is invalid whereas of the second vertical bar is valid. In the second case, corresponding extended part-faces are present.



5 Conclusions

Skeleton represents lower dimensional equivalent of a 2D or 3D shape. It is computed by methods like MAT, CAT, straight skeleton. Output computed by these methods typically does not represent the shape as per human perception, due to presence of gaps and unnecessary branches. Skeleton that lies midway of the 2D shape is called *midcurve*. For a non-trivial shape, instead of computing midcurve of the whole shape in one go, one can divide the shape in more manageable, simpler shapes, for which midcurve generation is a way simpler problem.

This paper presents such a *divide-and-conquer* strategy. Polygons are decomposed into primitive sub-polygons, Midcurves are created for each of them, and wherever necessary they are joined by extension. The results of improved partitioning [over Bayazit's (2013) approach] and its usage in creating properly connected midcurve are shown.

References

Attali, D., Boissonnat, J.D. and Edelsbrunner, H. (2004) 'Stability and computation of the medial axis a state-of-the-art report', *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, Springer-Verlag.

Bayazit, M. (2013) *Poly Decomp*, Polygon Decomposition Program polydecompbayazit.zip [online] http://mnbayazit.com/406/bayazit (accessed June 2013).

Blum, H. (1967) 'A transformation for extracting new descriptors of shape', in Wathen-Dunn, W. (Ed.): *Proc. Models for the Perception of Speech and Visual Form*, November, Cambridge, MA, MIT Press, pp.362–380.

de Moura Pinto, F. and Dal Sasso Freitas, C.M. (2009) 'Fast medial axis transform for planar domains with general boundaries', *SIBGRAPI Brazilian Symposium on Computer Graphics and Image Processing*, pp.96–103.

Fischer, A., Smolin, A. and Elber, G. (1999) 'Mid-surface of profile-based freeforms for mold design', *Journal of Manufacturing Science and Engineering, Transactions of the ASME*, Vol. 121, pp.202–207.

Haunert, J-H. and Sester, M. (2004) 'Using the straight skeleton for generalisation in a multiple representation environment', *ICA Workshop on Generalisation and Multiple Representation.*

Keil, M. and Snoeyink, J. (1994) 'On the time bound for convex decomposition of simple polygon', *Proceedings of the 10th Canadian Conference on Computational Geometry*, School of Computer Science, McGill University, Canada, pp.54–55.

Kulkarni, Y. and Deshpande, S. (2010) 'Medial object extraction – a state of the art', *Proc. of the 3rd International Conference on Advances in Mechanical Engineering*, S.V. National Institute of Technology, Surat, Gujarat, India.

Lam, L., Lee, S-W. and Suen, C.Y. (1992) 'Thinning methodologies – a comprehensive survey', *IEEE-PAMI*, September, Vol. 14, No. 9, pp.869–884.

Lien, J-M. and Amato, N.M. (2006) 'Approximate convex decomposition of polygons', *Comput. Geom. Theory Appl.*, August, Vol. 35, No. 1, pp.100–123.

Quadros, W.R. (2008) 'An approach for extracting non-manifold mid-surfaces of thin-wall solids using chordal axis transform', *Eng. with Comput.*, Vol. 24, No. 3, pp.305–319.

Ramanathan, M. and Gurumoorthy, B. (2004) 'Generating the mid-surface of a solid using 2d mat of its faces', *Computer Aided Design and Applications*, Vol. 1, pp.665–674.

Rocha, J. (1999) *Polygon Decomposition into Singular and Regular Regions*, Tech. Rep., University of the Balearic Islands.

Rocha, J. and Bernardino, R. (1997) 'Singularities and regularities on line pictures via symmetrical trapezoids', *Proceedings of the 4th International Conference on Document Analysis and Recognition, ICDAR '97*, IEEE Computer Society, Washington, DC, USA, pp.809–812.

Sheen, D-P., Son, T-g., Myung, D-K., Ryu, C., Lee, S.H., Lee, K. and Yeo, T.J. (2010) 'Transformation of a thin-walled solid model into a surface model via solid deflation', *Computer-Aided Design*, Vol. 42, No. 8, pp.720–730.

Woo, Y. (2013) 'Abstraction of mid-surfaces from solid models of thin-walled parts: a divide-and-conquer approach', *Computer-Aided Design*, Vol. 47, pp.1–11.

You, X. (2002) *Skeletonization Based on Wavelet Transform*, Computer Science Department, Hong Kong Baptist University.

Zou, J.J. and Yan, H. (2001) 'Skeletonization of ribbon-like shapes based on regularity and singularity analyses', *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 31, No. 3, pp.401–407.