A Novel Algorithm for Skeleton Extraction From Images Using Topological Graph Analysis

Liping Yang Diane Oyen Brendt Wohlberg
Los Alamos National Laboratory
Los Alamos, NM, USA
{liping.yang,doyen,brendt}@lanl.gov

Abstract

Skeletonization, also called thinning, is an important pre-processing step in computer vision and image processing tasks such as shape analysis and vectorization. It is a morphological process that generates a skeleton from an input image. Many thinning algorithms have been proposed, but accurate and fast algorithms are still in demand. In this paper, we propose a novel algorithm using embedded topological graphs and computational geometry that can extract skeletons from input binary images. We compare three well-known thinning algorithms with our method, with the experimental results showing effectiveness of the proposed method and algorithms.

1. Introduction and Related Work

Thinning of binary images is an important pre-processing technique in computer vision and image processing. Thinning generates a compact representation of images called a skeleton. A skeleton is a central line extraction of an object via thinning [11]. The skeleton can be useful for feature extraction, and representation of objects' topology, because a skeleton captures essential topology and shape information of an object in a simple form [10]. The thinning process reduces redundant information in images and thus reduces image complexity for tasks such as shape analysis and scene understanding.

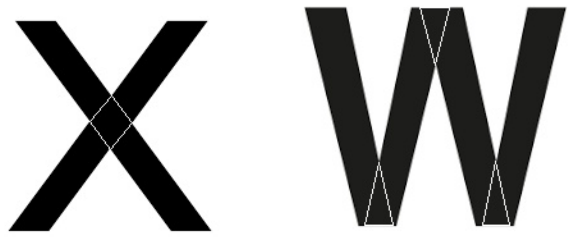
Thinning algorithms are categorized into two classes [7, 9, 5, 1]: iterative (pixel-based), and non-iterative (non-pixel based). The proposed algorithm in this paper is non-iterative. We compare our algorithm against the three most commonly cited algorithms in the literature [17, 12, 8].

The Zhang-Suen Thinning algorithm [17], proposed in 1984, is the most widely used and well-proved thinning algorithm, because of its robustness. A number of variants of the Zhang-Suen algorithm have also been proposed, e.g. [12, 3, 2]. Zhang-Suen and its following algorithms are raster-based, making it very computationally expensive. Our method takes advantage of vector-based methods (i.e., topological graphs and computational geometry) to extract skeletons of input binary images. Our results show the effectiveness of our approach and algorithms based on graphs and computational geometry (i.e., visibility polygon, kernel, and local kernel [16]).

2. Approach and Algorithms

In this paper, we propose an algorithm based on topological graphs and computational geometry algorithms (i.e., kernel and local kernel) for skeleton extraction from binary images. We use the idea of the kernel of a polygon, which is the set of all points of polygon P from which the entire interior of P is *visible*; that is, a straight line segment contained entirely within P can connect any point in the kernel with any point in P. If P is concave, then the kernel may or may not be empty. If the kernel is empty, then we use the idea of a local kernel. See Figure 1 for a brief explanation and examples of kernels and local kernels. The formal definitions of the kernel and local kernel is given in [16]. The algorithm for finding the kernel and local kernels is provided in Algorithm 1.

Once the kernel or local kernels are found, we divide the binary image into a set of neighboring convex polygons including the kernel, local kernels and the remaining polygons that make up the complete image. The image is now represented as a graph in which nodes represent the region polygons and links connect nodes rep-



(a) kernel (b) local kernel

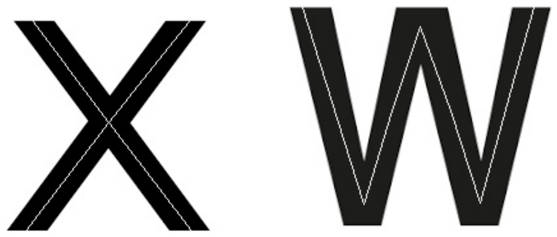
Figure 1: Examples of a kernel and local kernels. A kernel is a sub-polygon that can see all the parts of an input polygon. Not every polygon has a kernel. The region outlined in white in (a) provides an example of a kernel of the polygon “X”. The local kernel, as its name implies, is a localized kernel; it is a set of regions that can see most of the input region, when the kernel does not exist. For example, the three white outlined regions in (b) are the local kernels of the polygon “W”. Formal definitions of kernel and local kernel can be found in [16].

resenting polygons that share an edge (neighboring regions in the image are also neighbors in the graph). The skeleton is then constructed by connecting a straight line segment between the centroids of each neighboring pair of nodes. Leaf nodes have only one neighbor, and thus the skeleton would end at the center of the node; therefore we extend the line segment beyond the centroid of the leaf node to the boundary of the polygon represented by the node. See Figure 2 for the skeletons extracted using our algorithm corresponding to the kernel and local kernels shown in Figure 1.

Algorithms 1 and 2 provide details of our method. When implementing the algorithms, we use NetworkX [6] to store the polygon as a graph, because it is easier to update and merge the subdivided contour polygon according to its local kernel(s) when represented and stored as a graph, and the topological relations between polygons are represented and reserved in graphs. The skeleton graph is the dual graph of the subdivided contour structure graph (see an illustrated example in Figure 3).

Algorithm 1: Computing local kernels

```
Input: An embedded graph $G_p$ representing a
       concave polygon $P$
/* embedded graph means each node of the
   graph contains coordinates, so it is
   drawable on a plane.                      */
Output: a list of local kernel polygons of $P$
1  $C \leftarrow$ list(concave nodes of $G_p$) in CCW order
   // only concave node nodes to calculate
   local kernel, there is one local kernel
   for each concave node
             // CCW means counter-clockwise
2  $L_k \leftarrow$ null /* initialize a list to store
     local kernels                             */
   /* if concave node list $C$ is not empty    */
3  while $C$ do
4      $c \leftarrow$ current first node in $C$
5      calculate visibility polygon of $P$ at $c$
6      $P_vc \leftarrow$ visibility polygon of $P$ at $c$
7      $k_c \leftarrow$ calculate local kernel of $P_vc$
8      append $k_c$ to $L_k$
9      if $k_c$ contains nodes in $C$ then
10         remove the concave node from $C$ /* if a
               concave node can see another
               concave node, it is not necessary
               to calculate the local kernel for
               that concave node, because it is
               redundant.                      */
11 return $L_k$
```



(a) (b)

Figure 2: Skeletons extracted using our algorithm corresponding to the kernel and local kernels shown in Figure 1.

3. Experiments and Results

The results of our method can be found in Figure 4. The implementation used OpenCV, SymPy [13], and NetworkX [6]. The local kernel is implemented by the authors. We can see that the skeletons extracted using our method keep the topology of the image objects very well. Since the algorithm is not raster-based, it is much

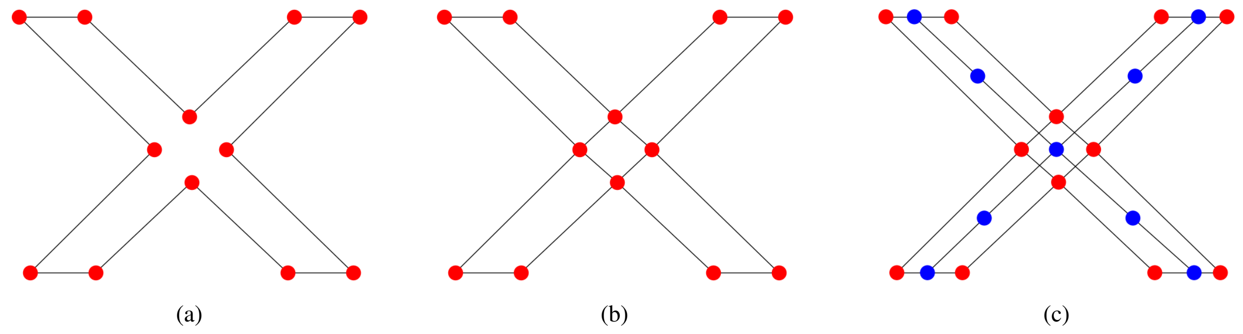


Figure 3: Graphs corresponding to the skeleton shown in Figure 2 (a). (a) structure graph generated from contours of input image, (b) updated structure graph (subdivided by the kernel), and (c) skeleton graph (i.e., the graph with blue nodes) overlaid by updated structure graph.

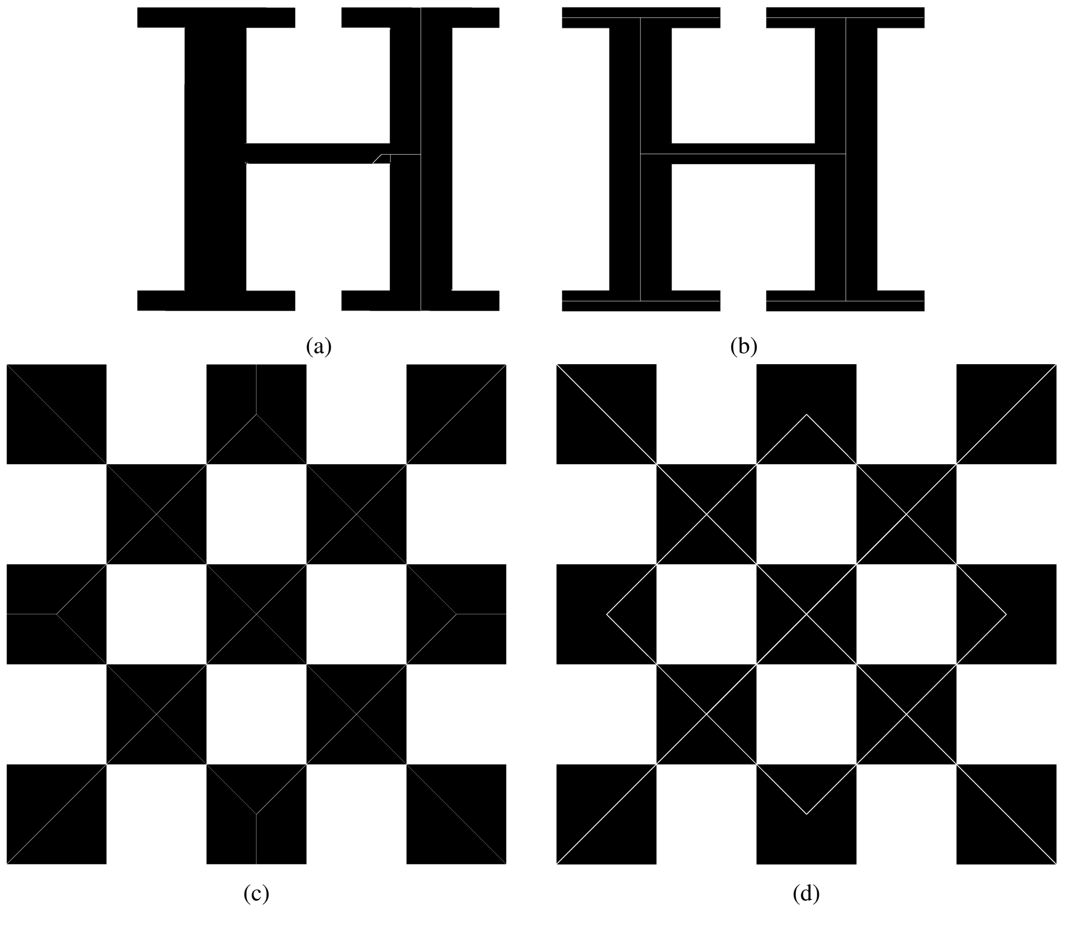


Figure 4: Results and comparison. (a) and (c) are generated by Zhang-Suen [17] algorithm (the other two algorithms we compared with produce very similar results as those using Zhang-Suen algorithm, thus we only provide the results using Zhang-Suen algorithm here). (b) and (d) show the results from our algorithm.

Algorithm 2: Skeleton extraction using graphs

```
Input: Binary image $M$
Output: Skeleton of $M$
1  $C \leftarrow$ contours of $M$  // $M$ is a binary image
     with White as the target, if not, invert
     it
   /* for each contour                                */
2  foreach $c \in C$ do
3      if $c$ is concave then
4          calculate local kernel of $c$
5          $K \leftarrow$ local kernels of $c$
6          subdivide the contour polygon $c$ according
            to $K$
7          $P$ = a set of subdivided convex polygons
             /* the set of convex polygons are
             stored in graph format               */
8
9          find centroid of each convex polygon
10         connect all centroids /* it is a graph,
             we call it skeleton graph            */
11         $L \leftarrow$ leaf nodes of the skeleton graph
           foreach leaf node $l \in L$ do
12             extend the lines and intersect with the
                contour poly $c$
13             add the intersection node and connect
                it to the corresponding leaf node
14         return skeleton graph
           /* The embedded graph is the
             skeleton of the input binary
             image $M$                            */
```

faster than the commonly-used thinning algorithms such as Zhuang-Suen [17].

4. Conclusion, Limitation and Future Work

We have proposed a novel skeleton extraction algorithm that can keep the topological relations among objects in images well. The skeleton extracted using our algorithm is not raster-based, thus it retains more information for further advanced shape and scene analysis in images compared with raster-based skeleton extraction methods. From Figure 4, it is obvious that our current skeleton extraction algorithm works very well for images with line boundaries, and its main limitation is that it does not deal with curve skeleton extraction. In the future, we will extend our algorithm to solve the cases with curves. Potential solutions to improve our current algorithm to be able to cope with curve skeletons are: (1) use Douglas-Peucker algorithm [14, 4] or Visvalingam's algorithm [15] to simplify complex shapes (e.g., those with curves) to poly-lines. While Douglas-Peucker is the most well-known for simplifying geometry, Visvalingam's algorithm [15] is more effective and has a remarkably intuitive explanation, becuase it progressively removes points with the least-perceptible change. We will use both algorithms to simplify complex shapes and evaluate which one works best for skeleton extraction; or (2) use Bézier curve to model the curve skeleton via storing the curve parameters (e.g., control points) into graphs.

Topological graph-based skeleton extraction has clear and intuitive advantages for further shape analysis, and also for automation of effective feature extraction for machine learning algorithms for image analysis. Thus, many potential applications could benefit from the algorithm and approach proposed in this paper, such as intelligent image interpretation for visually impaired people (it is much easier to describe the spatial arrangements and spatial relationships among objects appeared in an image, compared with raster-based methods).

References

[1] Lynda Ben Boudaoud, Basel Solaiman, and Abdelkamel Tari. Implementation and comparison of binary thinning algorithms on GPU. *Computing*, pages 1–27, 2018.

[2] Lynda Ben Boudaoud, Basel Solaiman, and Abdelkamel Tari. A modified ZS thinning algorithm by a hybrid approach. *The Visual Computer*, pages 1–18, 2018.

[3] Wei Chen, Lichun Sui, Zhengchao Xu, and Yu Lang. Improved Zhang-Suen thinning algorithm in binary line drawing applications. In *2012 International Conference on Systems and Informatics (ICSAI2012)*, pages 1947–1950. IEEE, 2012.

[4] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973.

[5] Matús Gramblicka and Jozef Vasky. Comparison of thinning algorithms for vectorization of engineering drawings. *Journal of Theoretical and Applied Information Technology*, 94(2):265, 2016.

[6] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *In Proceedings of the 7th Python in Science Conference (SciPy)*, pages 11–15, 2008.

[7] Erin Hastings. A survey of thinning methodologies. *Pattern analysis and Machine Intelligence, IEEE Transactions*, 4(9):869–885, 1992.

[8] Anil Kumar and Vinaya Teja. Edge detection based on Otsu method and Stentiford algorithm. *Interna-*

tional Journal of Engineering Research Technology, 3(11):1527–1530, 2014.

[9] Harish Kumar and Paramjeet Kaur. A comparative study of iterative thinning algorithms for BMP images. *International journal of computer Science and information Technologies (IJCSIT)*, 2011.

[10] J. Komala Lakshmi and M. Punithavalli. A survey on skeletons in digital image processing. In *2009 international conference on digital image processing*, pages 260–269. IEEE, 2009.

[11] Louisa Lam, Seong-Whan Lee, and Ching Y Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 14(9):869–885, 1992.

[12] H. E. Lü and Patrick Shen-Pei Wang. A comment on a fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 29(3):239–242, 1986.

[13] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, 2017.

[14] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256, 1972.

[15] Maheswari Visvalingam and James D Whyatt. Line generalisation by repeated elimination of points. *The cartographic journal*, 30(1):46–51, 1993.

[16] Liping Yang and Michael Worboys. Generation of navigation graphs for indoor space. *International Journal of Geographical Information Science*, 29(10):1737–1756, 2015.

[17] T. Y. Zhang and Ching Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.