A SPECIAL SKELETONIZATION ALGORITHM FOR CURSIVE WORDS

Steinherz, T.; Intrator, N.; Rivlin, E.

A SPECIAL SKELETONIZATION ALGORITHM FOR CURSIVE WORDS

TAL STEINHERZ, NATHAN INTRATOR
Tel-Aviv University
Ramat Aviv 69978, Israel
(talstz,nin)@math.tau.ac.il

EHUD RIVLIN
Department of Computer Science
Technion
Haifa 32000, Israel
ehudr@cs.technion.ac.il

We present a novel approach for finding a pseudo-skeleton of a cursive word's image. This pseudo-skeleton preserves all the necessary components of a cursive word such as: loops, curves, junctions, end-points etc. It is expected to be useful for cursive word recognition.

Keywords: Skeleton, Thinning, Cursive Words.

1 Introduction

It is no secret that cursive word recognition could have been much simpler if one was given the skeleton of a word's image rather than the binary version. Nevertheless, most of the off-line recognition systems that handle cursive words choose not to deal with skeleton computation, and prefer to engage with various morphological operations in order to extract features directly from the binary image [1]. Exceptions can be found in [2,3,4] One reason for this is the poor performance of current algorithms for skeleton computation on cursive word data. In fact most of the current skeletonization algorithms are not specifically adapted for cursive hand writing but are useful for thining of any binary object. Recently, several specific application oriented skeletonization algorithms were introduced. For example, one can find unique skeletonization algorithms for *Character Recognition* [5,6,7]. It is our belief that one can improve the results of cursive word skeletonization by designing a unique algorithm for cursive words, based on the characteristics of handwriting. This is the motivation for the algorithm we will discuss next.

In: L.R.B. Schomaker and L.G. Vuurpijl (Eds.), Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, September 11-13 2000, Amsterdam, ISBN 90-76942-01-3, Nijmegen: International Unipen Foundation, pp 529-534

2 Skeletons

Finding the skeleton of an object given its binary image was one of the first problems dealt with in computer vision [8]. There is no formal definition of a skeleton, let alone a mathematical proof for the validity of a proposed skeleton given its original binary image. However, In general one may say that *a skeleton is a set of thin lines (one-pixel thick), attached one to another in a few connection points, that preserve the topological and geometric properties of its originating object.* There are mainly two classes of skeletonization methods. In the first category also known as "medial axis transformation", center lines are extracted by connecting the centers of blocks that cover the object. The center of a block is usually found by maximizing some kind of a distance function. The second category consists of "thinning" algorithms [9] which are constructed by *successive removal of outer layers of pixels from an object while retaining any pixel whose removal would alter connectivity or shorten the legs of the skeleton.*

2.1 Pseudo-skeletons

In a series of articles in the late 80's and early 90's (for example [4]), J. C. Simon defined cursive handwriting as: "displacing a pen from left to right in an oscillating movement, with loops, descendants (legs), and ascendants (poles)". In other words a cursive word consists of a single backbone (*axis*), that is a chain of strokes with possible loops that goes from left to right, and numerous sub-strokes (*tarsi*) that hang above and below the main body. In fact, a-posteriori the axis may be considered as a single continuous stroke. The pseudo-skeleton attempts to preserve these features.

It seems that for cursive word recognition purposes one does not necessarily need a classic skeleton of the binary word's image. Instead he would rather have a thin image that preserves the important features, approximately in their original location. We call a thin image of this kind a *Pseudo-skeleton.*
Definition: a Pseudo-skeleton of a cursive word is a thin image that satisfies the following requirements:

- Each binary loop is associated with a thin loop of the same shape proportional size, and approximately the same location.
- Each terminal stroke is associated with an end-point.
- Each stroke intersection is associated with a junction.
- The size and curvature of every stroke is preserved.

Where thin means that

$$\forall \text{ pixel }, |\text{neighbors(pixel)}| = \begin{cases} 1 & \text{iff the pixel is an end} - \text{point.} \\ 2 & \text{iff the pixel is not an end} - \text{point or a junction.} \\ 3+ & \text{iff the pixel is a junction (of type X or T)} \end{cases}$$

Note that in this case a junction of 4th order ($X - junction$) or above, may turn into some (2) junctions of 3rd order ($T - junction$).

3 Construction of a pseudo-skeleton

Based on the structure of a cursive word we came to the conclusion that one can compute a pseudo-skeleton in two phases regarding the axis and tarsi respectively. Following is a short description of our algorithm.

3.1 Loop extraction

We find the loops to be the anchors of a cursive word. Hence, given a binary image(I), the first step is to extract all the available loops. In this case extraction means that for each loop in the binary image, one should create a thin loop of similar shape and size in the skeleton. For example one can use the internal body edge of the binary word as the thin version of the loops.

3.2 Graph transformation

At this stage, the image is transformed into a graph (G). Each node is associated with a black, i.e. body, pixel of the word's binary image. Moreover, there is an isomorphism between the nodes of the graph ($V = \{v_1 \dots v_N\}$) and the black pixels in the binary image. In particular $N =$ the number of black pixels in the binary image.

The neighboring matrix (M) is formed as follows:

$$M_{i,j} = \begin{cases} 1 \text{ iff the pixels represented by } i \text{ and } j \text{ are } 8 - \text{neighbors} \\ 0 \text{ otherwise} \end{cases}$$

In this case the distance between a pixel and each one of its 8−neighbors is considered to be identical. Next, all nodes that are associated with pixels that are part of a thin loop are grouped. Grouping means that the set of nodes are replaced by a single special node. The neighbors of the new node are accepted by merging the neighbors of each one of the replaced nodes. A new line in the neighboring matrix for a special node k that replaced the set of nodes L is formed by:

$$M'_{k,j} = \begin{cases} 1 \text{ iff } \exists i \in L | M_{i,j} = 1 \\ 0 \text{ otherwise} \end{cases}$$

3.3 Axis computation

We define the axis of a word to be the set of pixels associated with the nodes that are part of the shortest path between the nodes that represent the left most and right most pixels of the word's image. Note that if the shortest path crosses a loop - the whole loop is considered part of the axis. According to the isomorphism between graph G and the word's image I, one can see that the thinning requirement is satisfied. The shortest path is found using a standard BFS algorithm.

3.4 Tarsi computation

Given the axis, one can separate the remaining nodes (pixels) into two groups according to their distance from the axis. In this case the distance refers to the *Geodesic* distance in the binary image which is equivalent to the shortest path in the preliminary version of graph G, i.e., before nodes of loops were grouped. Moreover a distance from a single node to the axis is the length of the shortest path from the node to closest node of the axis. The first group of nodes includes those that are close to the axis up to a threshold. Given that the threshold was calculated according to an estimation of the stroke width, one can assume that these nodes are associated with pixels that belong to the strokes of the axis. On the other hand the other nodes belong to the various tarsi. When loops are considered a single entity, the image of each tarsus is like a thick tree. In this case the root of the tree is a group of nodes that have at least one node of the axis as a neighbor, and the leaves of the tree are the groups of nodes located at the peak of the edge of terminal branches. Therefore one can recursively select an end point and connect it to the backbone. The connection to the backbone will skeletonize the branches on the way. In this case one can map all the tarsi, one terminal branch after the other, with the following procedure:

```
procedure compute_tarsi(axis,threshold)
{
  current_skeleton = axis;
  while (TRUE)
  {
    current_end_point = find_most_distant_edge_pixel(I,current_skeleton);
    shortest_path = find_shortest_path(G,current_skeleton,end_point);
    if length(shortest_path) < threshold
      return current_skeleton; no more tarsus branches
    else
      current_skeleton = current_skeleton + shortest_path;
```

```
  }
}
```

The final process of the skeletonization is to smooth the paths between every two junctions. The output of the algorithm may produce paths with small fluctuations, and a smoothing process ensures that consecutive pixels are in the general direction of the curve.

4 Discussion

To the best of our knowledge, this is the first attempt to design a unique skeleton computation method for binary images of cursive words. A related method [10] evaluates pixels of the skeleton according to handwriting characteristics. Clearly, one can obtain a more sophisticated and accurate algorithm by limiting the variance of the objects it refers to. In this case, the identification of loops as anchors to any skeleton as well as characterizing the desirable legs of the computed skeleton simplifies the computation. We were motivated by the fact that unique skeletonization methods have already been shown useful in several OCR systems [11,12,13].

Furthermore, sup-parts of our proposed algorithm are useful for two other tasks: *slant computation* and *possible segmentation point (PSP) identification (selection)*. *Slant computation* is achieved by considering vertical strokes only. Our experiments proved that for each tarsi, the longest branch represents a vertical stroke. This is the first branch that is being skeletonized in the relevant tarsi. Hence, one can isolate the vertical strokes in order to compute the slant.

It is well known that *PSPs* are always on the axis. Moreover they are somewhere between the tarsus intersection points. Therefore, given the pseudo-skeleton found by our algorithm, the number of *PSP* candidates is greatly reduced. Usually, based on the pattern of the axis, one can select a subset of these pixels that will be of a practical size while containing the real set of segmentation points.

Acknowledgments

This work was partially supported by the Hermann Minkowski – Minerva Center for Geometry and by a grant from the Israeli Academi of Science to the Center for Geometric Computing and its Applications at Tel Aviv University.

References

1. T. Steinherz, E. Rivlin, and N. Intrator. Off-line cursive script word recognition: A survey. *International Journal of Document Analysis and*

Recognition, 2(2):90–110, 1999.

2. V. Govindaraju and R. K. Krishnamurthy. Holistic handwritten word recognition using temporal features derived from off-line images. *Pattern Recognition Letters*, 17(5):537–540, 1996.
3. H. Bunke, M. Roth, and E. G. Schukat-Talamazzini. Off-line cursive handwriting recognition using hidden Markov models. *Pattern Recognition*, 28(9):1399–1413, 1995.
4. J. C. Simon. Off-line cursive word recognition. *IEEE Proceedings*, 80(7):1150–1161, 1992.
5. H. Nishida, T. Suzuki, and S. Mori. Thin line representation from contour representation of handprinted characters. In *Proceedings Int. Workshop on Frontiers in Handwriting Recognition*, pages 57–68, 1991.
6. C. Chouinard and R. Plamondon. Thinning and segmenting handwritten characters by line following. *Machine Vision and Applications*, 5:185–197, 1992.
7. T.S. Cheng, C. Chiang, S.M. Roan, and H.C. Fu. Feature-preserving thinning algorithm for optical character recognition. In *Proceedings of the SPIE - Character Recognition Technologies*, pages 279–290, 1993.
8. L. Lam, S.W. Lee, and C.Y. Suen. Thinning methodologies: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, September 1992.
9. B.K. Jang and R.T. Chin. Analysis of thinning algorithms using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):541–551, June 1990.
10. T. Caesar, J. Gloger, A. Kaltenmeier, and E. Mandler. Recognition of handwritten word images by statistical methods. In *Proceedings Int. Workshop on Frontiers in Handwriting Recognition*, pages 409–416, 1993.
11. L. Lam and C.Y. Suen. An evaluation of parallel thinning algorithms for character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(9):914–919, 1995.
12. R. Plamondon, M. Bourdeau, C. Chouinard, and C.Y. Suen. Validation of preprocessing algorithms: A methodology and its applications to the design of a thinning algorithm for handwritten characters. In *Proceedings Int. Conf. on Document Analysis and Recognition*, pages 262–269, 1993.
13. R. Plamondon, C.Y. Suen, M. Bourdeau, and C. Barriere. Methodolgies for evaluating Thinning algorithms for character recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(5):1247 – 1270, October 1993.