

# Implementation and Advanced Results on the Non-Interrupted Skeletonization Algorithm

Khalid Saeed, Mariusz Rybniak, Marek Tabedzki

Computer Engineering Department  
Faculty of Computer Science  
Bialystok University of Technology  
Wiejska 45A, 15 351 Bialystok<sup>1</sup>  
POLAND  
[aidabt@ii.pb.bialystok.pl](mailto:aidabt@ii.pb.bialystok.pl)

**Abstract.** This paper is a continuation to the work in [1], in which a new algorithm for skeletonization is introduced. The algorithm given there and implemented for script and text is applied here on images like pictures, medical organs and signatures. This is very important for a lot of applications in pattern recognition, like, for example, data compression, transmission or saving. Some interesting results have been obtained and presented in this article. Comparing our results with others we can conclude that if it comes to thinning of scripts, words or sentences our method is as good as some of the latest approaches, when considering cursive script. However, when it comes to pictures, signatures or other more complicated images, our algorithm showed better and more precise results [6].

## 1 Introduction

Skeletonization is a very important stage in pattern recognition when considering almost all methods and approaches of classification. Despite this fact, in most cases, the thinning is not as precise as required, although the authors of such approaches declare that their methods are almost ideal [1,2,3,4]. In fact, there exist some methods that lead to an almost one-pixel-skeletonized image [1,3,4]. These methods proved to really be good when considering scripts or words, or some special applications [2,5] but not pictures or some complicated images like medical ones [1,6]. Since not all authors reveal the details of their algorithms or the computer programs, and they usually do not mention their drawbacks as discussed in the general survey in [7], the comparison is really problematic. In many cases it was made in such a way that the authors of this paper had implemented the algorithms of others according to the basic ideas mentioned in their published proceedings and then compared the results with ours under the same conditions. The experiments showed, in almost all considered cases, high efficiency in keeping the connectivity (non-interruption) of the image

---

<sup>1</sup> *The Rector of Bialystok University of Technology financially supports this manuscript.  
Grant no. W/II/3/01.*

contour without missing the main features of it. This is essential, particularly in compressing, transmitting or saving data of large size, or of necessary features to keep after thinning.

## 2 Theoretical Considerations

The most essential considerations in this aspect are concentrated on the following algorithm and its implementation. The general criterion for thinning scripts, words and sentences of different languages is given in [1]. Here, however, we introduce the main stages of the algorithm, which lead to a recognizable contour.

### 2.1 Algorithm of Thinning

The algorithm of thinning to one-pixel-skeleton image presented in [2] is modified [1] to have a non-interrupted image. By non-interrupted image we mean the one with a continuous contour of one-pixel-wide skeleton. This is basic for most of the applications of the algorithm whose essential steps are given below. We are considering the Arabic letter **ي** - pronounced *yaa*, but without dots. Arabic letters are of cursive character and hence very good examples of other handwritten languages or images like signatures or pictures. The same letter was used in the basic algorithm [2].

1. The image is bitmapped, first with its black pixels designated 1's:

```

                1111
                11111
1              111111
1              1
1              11
1              11111
1              1111
11             11
111            11
1111111111111
1111111111
111111

```

2. The 1's of the contour (that sticking the 0's background) are changed into 2's; those in the elbow corners into 3's. Then we obtain the following figure:

```

                2222
                23112
2              222222
2              2
2              22
2              23222
2              2222
22             22
232            22
2332222222222
231111322
                222222

```

*Remark 1.* Note that this stage may end the algorithm indicating the final contoured shape defined by the outside and inside circumferences. If it comes to pictures or some applications of image processing, this is the main aim of skeletonization. Fig.1 shows the letter  $\text{س}$  together with its contour shape after stage 2:



**Fig. 1.** The letter  $\text{س}$  without dots.

However, to show the whole method of thinning to a one-pixel-skeleton image, being the most required and practical case, consider the following additional stages [1].

3. Consider the points with 2, 3 or 4 sticking neighbors and change them into 4's:

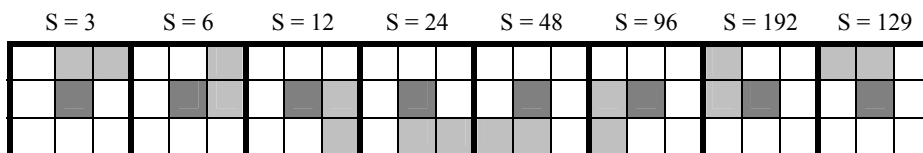
```

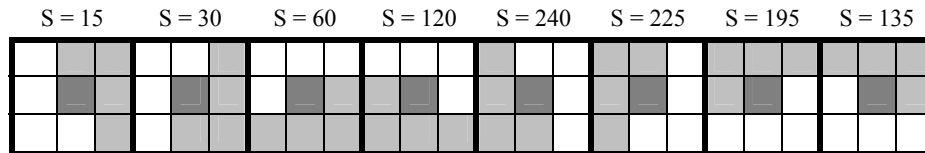
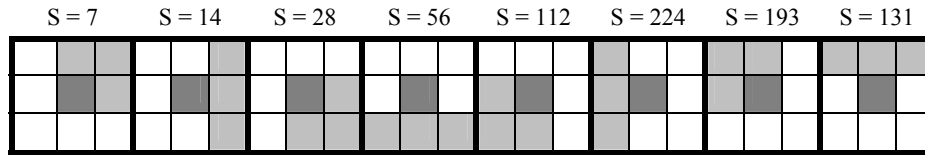
                4224
                23112
           2    222224
          2    2
          2    22
          2    43224
          2    4222
          22   42
          232  42
          43322222224
          231111324
          422224
    
```

Notice that there are 8 such possibilities in each case depending on the way the pixels surrounding the under test point  $x$  are filled. The weight of each of these pixels is taken from the following array:

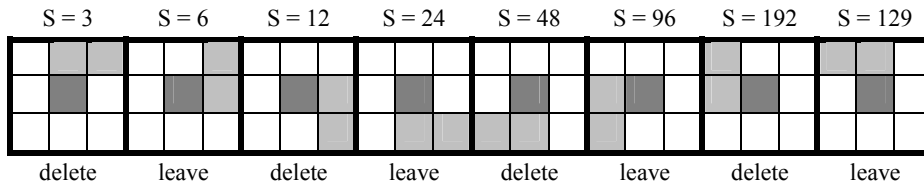
128	1	2
64	$x$	4
32	16	8

Therefore, we have 24 different possibilities given in three different arrays with two, three or four neighbors, respectively:





The dark points (x's) show the pixels to either be removed or left, depending on their position in the image. As an example, the following array shows the successive points of removing or leaving for the case of two-neighbour points. The sum  $s$  of the surrounding pixels, ranging from 0 to 255, decides whether to delete the point or not:



The following Deletion Array gives the sums of points to be removed:

3	5	7	12	13	14	15	20
21	22	23	28	29	30	31	48
52	53	54	55	56	60	61	62
63	65	67	69	71	77	79	80
81	83	84	85	86	87	88	89
91	92	93	94	95	97	99	101
103	109	111	112	113	115	116	117
118	119	120	121	123	124	125	126
127	131	133	135	141	143	149	151
157	159	181	183	189	191	192	193
195	197	199	205	207	208	209	211
212	213	214	215	216	217	219	220
221	222	223	224	225	227	229	231
237	239	240	241	243	244	245	246
247	248	249	251	252	253	254	255

4. Delete the 4's until getting the following shape:

```

                22
               23112
              22222
             2
            2
           2
          2
         2
        22
       232
      3322222222
     23111132
    2222

```

5. Check for deleting the unnecessary 2's and 3's in the figure above without interrupting the connectivity of the image, designating the essential-to-be-left points by 1's, to get:

```

                1311
               1
              1
             1
            311
           1
          1
         3
        33
       311113

```

This stage is sometimes repeated until reaching the following final image:

```

                1111
               1
              1
             1
            11
           1
          1
         1
        1
       111
      11111

```

## 2.2 Computer Flow Chart

The algorithm has been programmed using *MFC C++* language and given the name *KMM*. The flow chart of the computer program is given in the Appendix. A number of examples are tested by this program and compared with other approaches for speed, complexity and cost.

## 3 Examples

The examples we consider here differ from those considered in [1], as we are interested in picture images than texts. We are considering Fourier picture, the colon as an example of medical organs, and the signature of one of the authors.

### 3.1 Pictures

A number of pictures have been tested according to the algorithm of this paper. Pictures and photos - color, black and white - were considered, of which Fourier's photo was one. Fig.2 shows Fourier's original photo and its contour thinned form. Notice that the important features of the image are still saved.



Fig. 2. Fourier Image thinned for the purpose of compression and data saving

### 3.2 Signatures

Another example used to demonstrate feature vector extraction is a typical signature recognition [8]. Although the algorithm used in signature recognition does not need to thin the image of the signature before classifying it, the thinning proved to simplify the process of description and classification. Fig.4 shows a signature sample with its thinned shape, with all features kept unchanged.

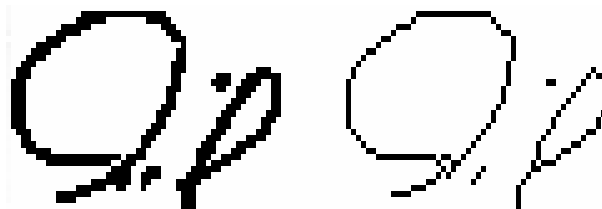
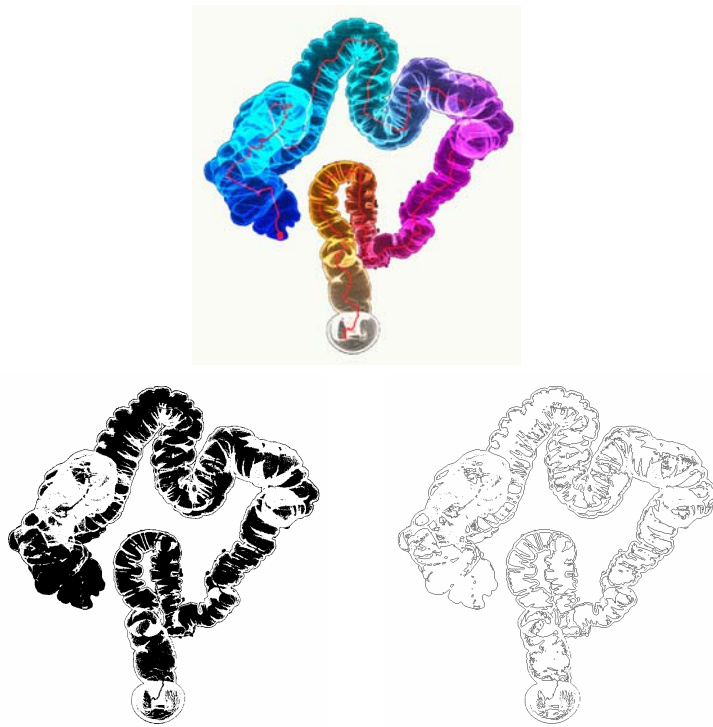


Fig. 4. Signature Thinning

### 3.3 Medical Images

Consider the colon in Fig.3 in its three different forms, original three-dimensional colour image, black and white one and the thinned to its contour final shape. This is one of several examples tested to contour the medical images for preparation to compress before transmitting or saving as the skeletonized image.



**Fig. 3.** Colon Thinning

### 4 Conclusions

For the aim of pattern recognition [9,10,11], the algorithm described in [1] is sufficient to achieve good results. The interruption in the continuity of lines forming the skeleton of an image, in many cases, showed more practical features for classification and description following the criterion in [2]. However, that method does not provide one-pixel width skeletonization. The one presented here, though, satisfies the conditions for having uninterrupted skeleton of the tested image. This condition is required before applying many algorithms in image processing. The

algorithm presented in this paper showed significant improvement and in some cases higher efficiency and more practical results.

## References

1. K. Saeed, "Text and Image Processing: Non-Interrupted Skeletonization," Accepted for publication and presentation in WSES/IEEE - CSCC'01, World Scientific and Engineering Society Multi-Conference on Circuits, Systems, Communications and Computers, July 8-15, Crete, Greece 2001.
2. K. Saeed, R. Niedzielski, "Experiments on Thinning of Cursive-Style Alphabets," ITESB'99, Inter. Conf. on Information Technologies, June 24-25, Mińsk 1999.
3. Yung-Sheng Chen, "The Use of Hidden Deletable Pixel Detection to Obtain Bias-Reduced Skeletons in Parallel Thinning," Proceedings of 9<sup>th</sup> ICPR'96 - IEEE, Vol.1, pp. 91-95, Vienna 1996.
4. Y.Y. Zhang, P.S.P. Wang, "A Parallel Thinning Algorithm with Two-Subiteration that Generates One-Pixel-Wide Skeletons," Proceedings of 9<sup>th</sup> ICPR'96 - IEEE, Vol.2, pp. 457-461, Vienna 1996.
5. G. Ososkov, A. Stadnik, "Face Recognition by a new type of Neural Networks," Proceedings of World Scientific and Engineering Society WSES- NNA'2001 Conf., WSES Press, pp. 304-308, February 12-14, Tenerife 2001, Spain.
6. K. Saeed, "New Approaches for Cursive Languages Recognition: Machine and Hand Written Scripts and Tests," Invited Paper in Proceedings of World Scientific and Engineering Society WSES- NNA'2001 Conf., WSES Press, pp. 304-308, February 12-14, Tenerife 2001, Spain.
7. M. Ghuwar and W. Skarbek, "Recognition of Arabic Characters - A Survey," Polish Academy of Science, Manuscript No.740, Warsaw 1994.
8. A. Hodun, "Signature Recognition," B.Sc. Thesis, Bialystok University of Technology, Bialystok 2001, Poland.
9. K. Saeed, "Three-Agent System for Cursive-Scripts Recognition," Proc. CVPRIP'2000 Computer Vision, Pattern Recognition and Image Processing - 5<sup>th</sup> Joint Conf. on Information Sciences JCIS'2000, Vol.2, pp. 244 -247, Feb. 27 - March 3, New Jersey 2000.
10. K. Saeed, A. Dardzińska, "Cursive Letters Language Processing: Muqla Model and Toeplitz Matrices Approach," FQAS'2000 – 4<sup>th</sup> Inter. Conference on Flexible Query Answering Systems, pp. 326-333, Oct. 25 – 27, Recent Advances, Springer-Verlag, Warsaw 2000.
11. K. Saeed, "A Projection Approach for Arabic Handwritten Characters Recognition," Proc. of ISCI - International Symposium on Computational Intelligence, pp.106-111, Aug. 31 – Sep. 1, New Trends and App. in Comp. Intelligence, Springer-Verlag, Kosice, Slovakia 2000.



## Appendix: The Computer Flow Chart of the Program *KMM*

