https://doi.org/10.26552/com.C.2009.3.43-48

Peter Tarabek *

A CONTOUR APPROACH TO THINNING ALGORITHMS

Thinning algorithms are widely used in many image processing tasks. Many thinning algorithms were proposed but they usually tend to process all image pixels in every iteration. Two approaches to contour thinning are described and a short discussion about their features is given. These approaches can be implemented as sequential or parallel algorithms with different deletion rules. Results of comparison and analysis are presented in this paper.

Keywords: thinning, skeleton, skeletonization, vectorization

1. Introduction

Many image processing and pattern recognition problems use thinning as one of the processing step. These problems include vectorization of raster maps and engineering drawings [1], character recognition and analysis [2, 3] and more [4–7]. Thinning belongs to skeletonization techniques [12] which are used to create a skeleton sometimes called medial axis. Like other segmentation techniques, thinning has its advantages and disadvantages. One of the often mentioned disadvantages is a low computational speed. Although thinning algorithms are relatively fast compared to other skeletonization techniques, they are still slow for some tasks. Many thinning algorithms were proposed [8, 9, 10, 13] but they usually tend to process all image pixels in every iteration.

In this paper the thinning technique is briefly described in section 2. In sections 3 and 4, two approaches to contour thinning with different rules are presented. Section 5 describes experiments, section 6 shows results and section 7 presents conclusions.

2. Thinning

Thinning algorithms remove outer pixels layer by layer in iterative process to produce one pixel thick skeleton. The thinning shows good results for objects, the length of which is much larger than their thickness. The skeleton is ideal for this type of objects because it is represented by a set of lines which are a natural representation of objects such as roads and characters. The result of thinning algorithms is a modified binary bitmap which must be further processed to yield a vector representation. Thinning should fulfill these requirements:

- Skeleton should be one pixel thick
- Connectivity should be preserved
- Shape and position of the junction points should be preserved
- Skeleton should lie in the middle of a shape (medial axis)
- Skeleton should be immune to noise (especially to boundary noise)
- Excessive erosion should be prevented (length of lines and curves should be preserved)

The nature of thinning algorithms can be parallel or sequential. Parallel thinning algorithms make their decisions about deleting pixels based on a bitmap resulting from the previous iteration, while sequential algorithms use an actual bitmap.

3. Contour approach

Pixels in binary images can be divided into 3 categories:

- background pixels
- contour foreground pixels
- other foreground pixels

During each iteration only contour foreground pixels can be deleted and so no other pixels need to be tested. Contour pixels usually represent only a small portion of pixels in engineering drawings and raster maps (especially drawing maps) so when all pixels are processed a large amount of processing time is wasted. For example, the image shown in Fig.1 consists of 338 322 pixels, but contour pixels represent only 12.5% of them.

The base idea of a contour approach is to process only contour pixels when conditions for deletion are tested. There are several contour tracing algorithms [14] which can be used in this process. To be able to describe the contour approach to thinning used in this paper following definitions are given:

Definition 1: The foreground pixel in a binary image is a black pixel which is a part of the important object.

* **Peter Tarabek**
Department of Transportation Networks, Faculty of Management Science and Informatics, University of Zilina, Slovakia,
E-mail: Peter.Tarabek@fri.uniza.sk

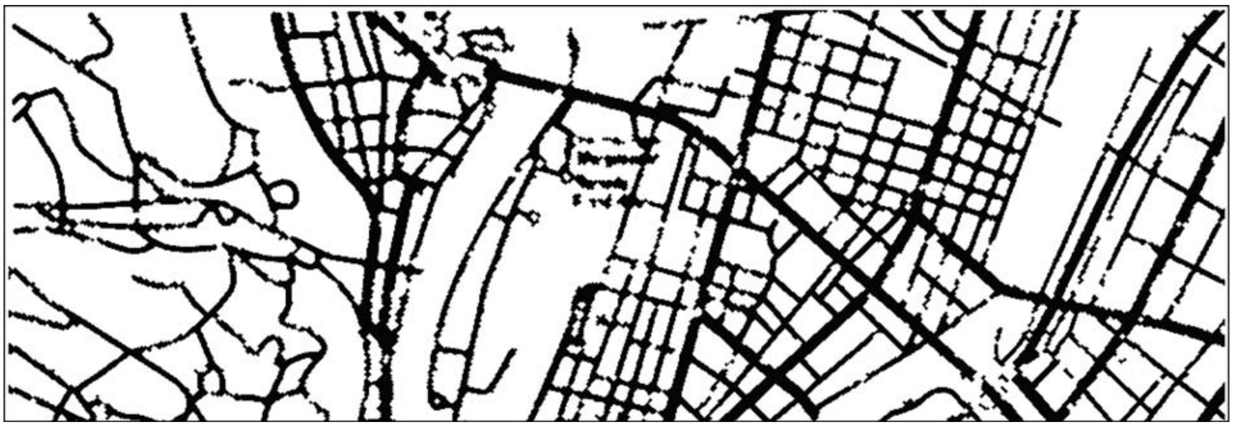

Fig. 1 Example of drawing map

Definition 2: The background pixel in a binary image is a white pixel which is a part of the unimportant background.

Definition 3: The 3×3 neighborhood of the pixel P is represented by the pixels P_1-P_8 as shown in Fig. 2.

P_8 [i−1, j−1]	P_1 [i−1, j]	P_2 [i−1, j+1]
P_7 [i, j−1]	P [i, j]	P_3 [i, j+1]
P_6 [i+1, j−1]	P_5 [i+1, j]	P_4 [i+1, j+1]

Fig. 2 Neighborhood of pixel P

Definition 4: The Contour Pixel is the foreground pixel whose 3×3 neighborhood contains at most 7 foreground pixels.

Definition 5: When contour is processed, the contour pixels are processed in clockwise order. Let P[i, j] be the processing pixel and P_7[i, j−1] be the previously processed pixel. A successor of pixel P is the first foreground pixel in its 3×3 neighborhood found in clockwise order starting from P_7 position. This means that the neighbors of P are processed in the following order: P_8, P_1, P_2, P_3, P_4, P_5, P_6, and P_7 to find the first foreground pixel. A predecessor is used to find a successor to ensure that the contour pixels will be processed in right order.

A general contour thinning algorithm can be described in 2 steps:
- recognition of contours
- iterative thinning of contours

In the first step the image is scanned to find contour pixels. When the contour pixel is recognized the whole contour is traced using the successor function. This function returns the successor of current pixel based on definition 5. In order to trace the whole contour the current pixel is marked as a predecessor and the successor is marked as a current pixel after the successor is found. This process continues until the contour is traced and all the accessed contour pixels are marked (colored) as used. This prevents from finding the same contour multiple times during the scanning process. To be able to use the successor function two pixels must be known. When the first pixel of the contour is recognized, the positions P_1, P_5, P_3 and P_7 are used to find the first background pixel. This background pixel is marked as a predecessor of the current pixel so the successor function can be used. Every contour in the image is stored for further processing by the first recognized contour pixel and its successor.

After the whole image is scanned, all contours are recognized. Next, the thinning process can begin. In each iteration, all the contours which were not marked as thinned are processed. The contour pixels are tested for deletion based on the rules and templates used by a specific thinning algorithm. A contour is marked as thinned if none of its contour pixels were marked for deletion in the previous iteration. When all the contours are marked as thinned the thinning process is finished.

In Figs. 3 and 4, the results of contour thinning using the basic deletion rules are shown. These rules mark the pixel for deletion if its connectivity number (number of black to white translations) equals to 1 and if the number of the foreground neighbor pixels is higher than 1 and less than 7.

The contour thinning algorithm based on the presented deletion rules produces distortions in junction points shown in Fig. 4. This is caused by the successor function which does not recognize all the contour pixels and produces the contour shown in Fig. 5. This contour approach to thinning (CT1) has its advantages and disadvantages. To process the contour, a smaller number of contour pixels need to be processed and what is more important the successor function and the whole algorithm are relatively fast and easy to implement as will be shown later. The disadvantage of this approach is a possible distortion in junction points. To deal with this problem, correct rules for deletion should be used. One possibility is to use two sub-iterations for pixel deletion allowing to access contour pixels which were not accessible in first or second sub-iteration.

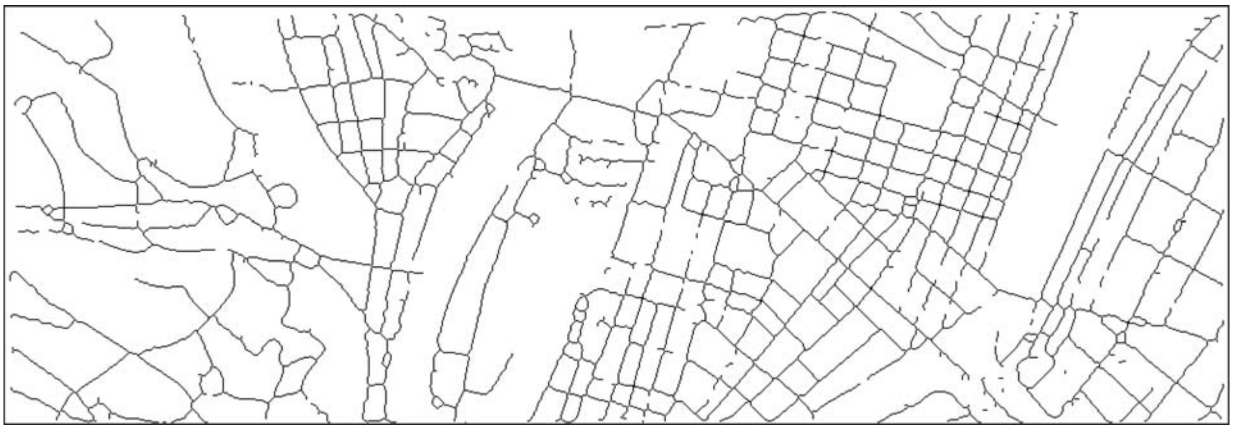

Fig. 3 Skeleton of drawing map

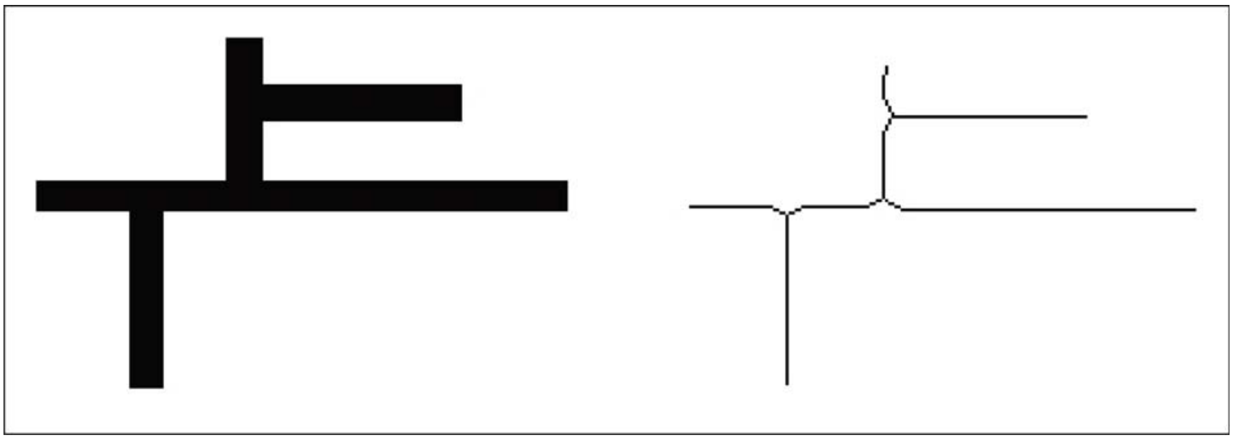

Fig. 4 Distortions in junction points

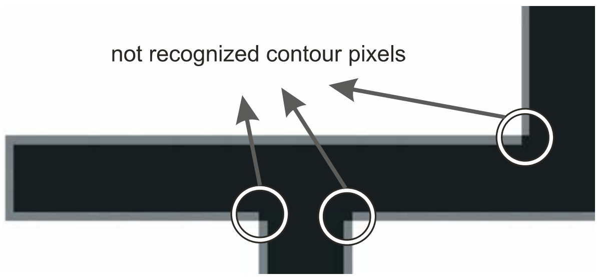

Fig. 5 Problem of contour pixels recognition

Another possibility is to improve the successor function to recognize all contour pixels. This function will use the same definition of a successor pixel (def. 5), but for all the successors which represent diagonal neighbors (P_2, P_4, P_6, P_8) of a current pixel, the next pixel in clockwise order is examined. If this pixel is a foreground pixel it is marked as a successor instead of the original one. Although this process seems to be easy, the contour approach to thinning based on such a successor function brings a lot of complications. The first problem is shown in Fig. 6. If the successor function is used, there is a high probability to create an infinite

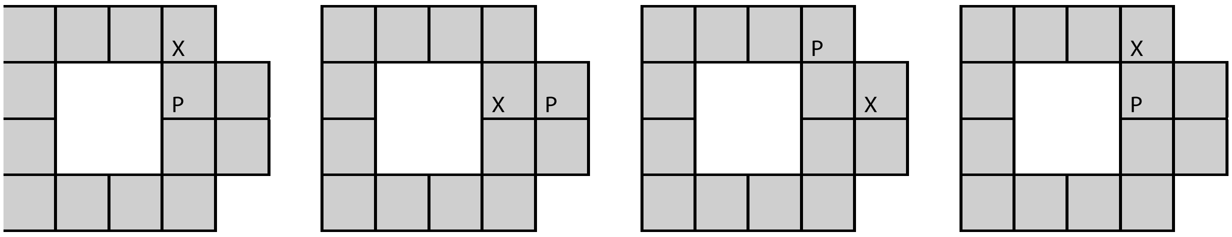

Fig. 6 Infinite loop

loop. In Fig. 6, four successive steps are shown. The current pixel is represented by ‘P’ and predecessor by ‘X’. When the contour is processed using this successor function, an infinite loop is created.

So definition 5 of the successor has to be modified in order to prevent from creating the infinite loops.

Definition 6: When contour is processed, contour pixels are processed in clockwise order. Let P be the processing pixel, X be the previously processed pixel and XX be the predecessor from previous step (where X was actual pixel). The successor of pixel P is the first foreground pixel from its 3×3 neighborhood found in the clockwise order starting from X position which differs from XX pixel.

A new contour approach to thinning (CT2) can be defined using the new successor function and the successor definition 6. This approach uses information about the actual pixel (P), its predecessor (X) and a predecessor from the previous step (XX) to process contours. Using the information about XX pixel the situation in Fig. 6 can be solved (see Fig. 7).

4. Implementation problems

Although some problems of CT1 and CT2 approaches were described in the previous section, there are still other issues to be solved. Critical points have to be defined before discussing these problems.

Definition 7: Critical points CP, CX and CXX are pixels which are stored for each contour and are used to start and stop contour processing. CP stands for the current pixel, CX stands for the predecessor and CXX represents the predecessor from previous step.

The first problem is to set “stopping rules”. When the contour is processed, a position of the current pixel P is compared to CP and a position of the current predecessor X is compared to CX. When these positions match, processing of the contour is stopped. In CT2, also the position of XX is compared to CX.

When some of the critical points are deleted, their positions must be updated for further accurate contour processing. This process can influence quality of results in place of these critical points.

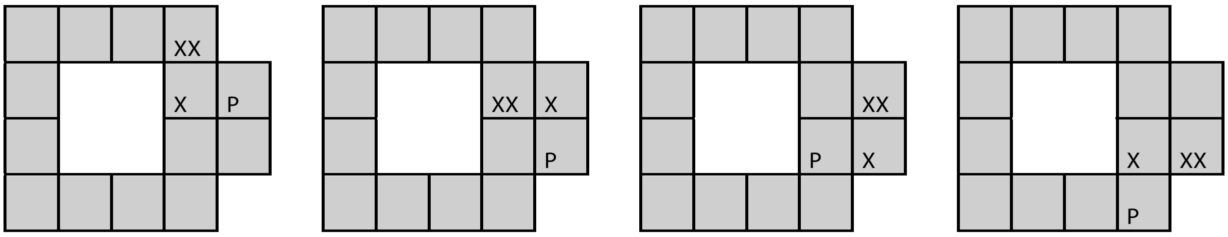

Fig. 7 Correct processing of contour

Information about all these pixels must be stored in order to correctly start and stop processing of contours. Sometimes X and XX pixels are at the same location. In this case XX pixel needs to be ignored when the successor function is used. Some other problems with implementation of CT2 are described in the next section. The result of CT2 using the same deletion rules as in CT1 is shown in Fig. 8.

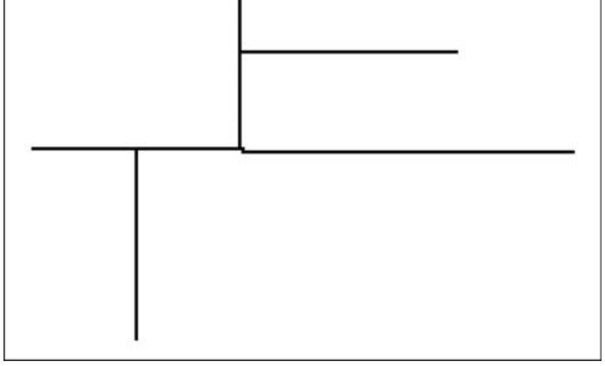

Fig. 8 Result of CT2

There are some other issues with deletion rules for both CT1 and CT2. One problem was shown in Figs. 4 and 5. In CT2, the most important problem is the problem of “staircase pixels”. These pixels can belong to two contours and when they are deleted by one contour, critical points of other contour can be deleted too, making further processing of the second contour impossible. To deal with this problem we can search through all the contours to find out if this pixel represents a critical point of other contour or not. If the result is positive we can update critical points of the given contour. This process is time consuming. A better approach is to create rules which keep staircase pixels and remove them in a post-processing step. Because of the mentioned problems and some other issues with CT2 implementation, CT2 approach is hard to implement and its behavior is hard to predict for some cases. Also it is slower than CT1 so the CT1 approach was tested in experiments.

5. Experiments

Two tests were used to evaluate the CT1 quality. In both tests Zhang-Suen thinning algorithm [10] and CT1 are used. Zhang-Suen algorithm is often used as a reference algorithm and in our case it represents thinning algorithms which process all the image pixels. Another advantage of this algorithm is that it uses two sub-iterations for pixel deletion. These deletion rules and two sub-iterations process can solve problems shown in Fig. 4 and so they were used in CT1 for these tests.

Test 1 compares quality of result based on performance measurements proposed in [11], where a thinning rate, number of components and noise sensitivity are evaluated. The second test consists of 4 cases and its purpose is to compare the processing time of both algorithms based on 4 different parameters of input images. These parameters represent percentage of background pixels (pBP), contour pixels (pCP) and non-contour foreground pixels (pFP) and a number of contours (NC). Detailed information about these images is shown in Table 1.

Parameters of input images used in test 2 Table 1.

	Dimensions	pBP	pCP	pFP	NC
Image 1	5000×8000	38.4%	7.4%	54.2%	11 984
Image 2	5000×8000	69.3%	4.0%	26.7%	8 270
Image 3	5000×8000	38.2%	0.8%	61.0%	138
Image 4	5000×8000	64.8%	0.6%	34.6%	155

6. Results

The image shown in Fig. 9 was used for test 1. Performance measurements show that the results for this input are the same for both algorithms (see Table 2.). More tests with the same criteria and manual verification were made. No fundamental differences were found in these tests.

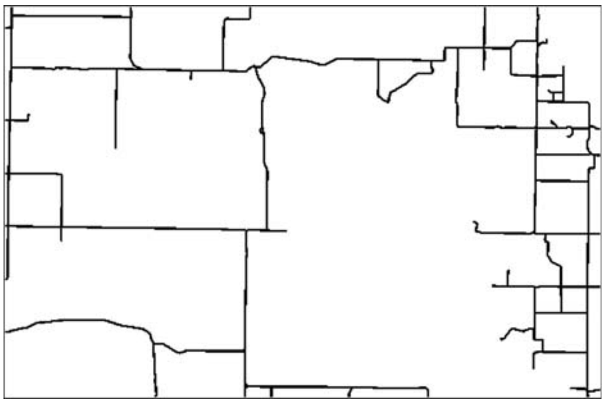

Fig. 9 Input image used in test 1

Performance measurements Table 2.

	Thinning rate	Number of components	Noise sensitivity
Zhang-Suen	693	1	62
CT1	693	1	62

The results of test 2 are shown in Table 3. For image 1 which consists of a large number of contours (11 984), the computation time is approximately the same for both algorithms. As the number of contours decreases and the number of background and non-contour foreground pixels increases, CT1 algorithm became faster than Zhang-Suen thinning algorithm. On the other hand, if images with more contours were tested, CT1 algorithm would perform much slower. This means that CT1 algorithm is much faster for shape objects the length of which is comparable to their thickness. When dealing with elongated objects, especially with objects the length of which is much larger than their thickness, the situation is more complicated. For example, engineering drawings and drawing maps usually have characteristics similar to image 2. They usually consist of a large percentage of background pixels, the number of contours is relatively high and they tend to have a lower percentage of non-contour foreground pixels. This situation can differ from image to image, but CT1 algorithm should perform faster for majority of them.

Results of test 2 (sec.) Table 3.

	Image 1	Image 2	Image 3	Image 4
Zhang-Suen	22.6	16.7	162.6	119.3
CT1	21.6	13.2	45.9	42.8

7. Conclusion

Two contour thinning principles CT1 and CT2 were presented in the paper. These approaches represent general principles which can be used with different deletion and implementation rules. Also they can be implemented as sequential or parallel algorithms. CT1 seems to be more robust. It can be easier to implement and perform faster than CT2. In section 6, CT1 and Zhang-Suen algorithms were compared. The CT1 algorithm was implemented with the same principles as Zhang-Suen algorithm (parallel nature, the same deletion rules and two sub-iterations). Both algorithms produce a skeleton with similar characteristics. Although when it comes to computational speed, CT1 tends to be faster for majority of input images (for all the images in our tests), there are still situations where the classical approach, processing all the pixels in the image, is faster.

References

[1] GOMIS, J. M., COMPANY, P., GIL, M. A.: *Vectorization in Recovering Engineering Drawings.* II Seminario Italo-Espanol, “Diseno y fabricabilidad de los productos industriales”, Marina di Equa (Napoli), 24-26 de Junio de 1998.

[2] ARICA, N., YARMAN-VURAL, F. T.: *An Overview of Character Recognition Focused on Off-Line Handwriting.* IEEE Trans. Systems, Man, and Cybernetics-Part C: Applications and Rev., vol. 31, no. 2, pp. 216–233, 2001.

[3] PERVOUCHINE, V., LEEDHAM, G.: *Document Examiner Feature Extraction: Thinned vs. Skeletonised Handwriting Images.* Proceedings of The IEEE Region 10 Technical Conference, (TENCON05), November 2005.

[4] ZOU, J. J., HONG, Y.: *Vectorization of cartoon drawings.* Selected papers from the Pan-Sydney workshop on Visualisation – Volume 2, 2000.

[5] AH-SOON, CH., TOMBRE, K.: *Variations on the Analysis of Architectural Drawings.* Fourth International Conference Document Analysis and Recognition (ICDAR'97), 1997.

[6] YAO-YI CH., KNOBLOCK, C. A., CHING-CHIEN CH.: *Automatic Extraction of Road Intersections from Raster Maps.* In Proceedings of the 13th ACM International Symposium on Advances in Geographic Information Systems, 2005.

[7] HASTHORPE, J., MOUNT, N. J.: *The generation of river channel skeletons from binary images using raster thinning algorithms.* Proceedings of the GIScience Research UK 15th Annual Conference, 2007.

[8] NG, G. S., ZHOU, R. W., QUEK, C.: A *Novel Single Pass Thinning Algorithm.* IEEE Transaction on System Man and Cybernetics, 1994.

[9] BERNARD, T. M., MANZANERA, A.: *Improved Low Complexity Fully Parallel Thinning Algorithm.* Proceedings of the 10th International Conference on Image Analysis and Processing, 1999.

[10] ZHANG, T. Y., SUEN, C. Y.: *A fast parallel algorithm for thinning digital patterns.* Commun. ACM, 27(3), 1984.

[11] TARABEK, P.: *Performance measurements of thinning algorithms.* Journal of Information, Control and Management Systems, Vol. 6, 2008.

[12] TOMBRE, K., AH-SOON, C., DOSCH, P., MASINI, G., TABBONE, S.: *Stable and Robust Vectorization: How to Make the Right Choices.* Graphics Recognition – Recent Advances, 2000.

[13] LAM, L., LEE, S. W., SUEN, C. Y.: *Thinning Methodologies – A Comprehensive Survey.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 9, September 1992.

[14] GHUNEIM, A. G.: *Contour tracing.* Project for the Pattern Recognition course, http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/index.html

Acknowledgement:
This work has been supported by grant VEGA No. 1/3775/06.