Sensors **2008**, *8*, 2673-2694

sensors

ISSN 1424-8220
© 2008 by MDPI
www.mdpi.org/sensors

Full Research Paper

Multidirectional Scanning Model, MUSCLE, to Vectorize Raster Images with Straight Lines

Ismail Rakip Karas [1], Bulent Bayram [2], Fatmagul Batuk [2], Abdullah Emin Akay [3] and Ibrahim Baz [1]

[1] Gebze Institute of Technology, Dept. of Geodesy and Photogrammetry Engineering, 41400, Gebze, Kocaeli, Turkey; E-mails: ragib@gyte.edu.tr; ibaz@gyte.edu.tr
[2] Yıldız Technical Univ., Dept. of Geodesy and Photogrammetry Engineering, 34349, Besiktas, Istanbul, Turkey; E-mails: bayram@yildiz.edu.tr; batuk@yildiz.edu.tr
[3] Kahramanmaras Sutcu Imam Univ., Dept. of Forest Engineering, 46100, Kahramanmaras, Turkey; E-mail: akay@ksu.edu.tr

* Author to whom correspondence should be addressed; E-mail: ragib@gyte.edu.tr; www.ismailkaras.com; Tel. +90 262 6053163; Fax +90 262 6053163

Received: 19 February 2008 / Accepted: 1 April 2008 / Published: 15 April 2008

Abstract: This paper presents a new model, MUSCLE (Multidirectional Scanning for Line Extraction), for automatic vectorization of raster images with straight lines. The algorithm of the model implements the line thinning and the simple neighborhood methods to perform vectorization. The model allows users to define specified criteria which are crucial for acquiring the vectorization process. In this model, various raster images can be vectorized such as township plans, maps, architectural drawings, and machine plans. The algorithm of the model was developed by implementing an appropriate computer programming and tested on a basic application. Results, verified by using two well known vectorization programs (WinTopo and Scan2CAD), indicated that the model can successfully vectorize the specified raster data quickly and accurately.

Keywords: Raster to vector conversion, geographic information systems, vectorization, thinning, topology, threshold.

1. Introduction

Data collection used to be the major task which consumed over 60% of the available resources since geographic data were very scarce in the early days of GIS (Geographic Information Systems) technology. In most recent GIS projects, data collection is still a time consuming and expensive task; however, it currently consumes about 15-50% of the available resources [1]. In order to reduce total project cost, data generation method of extracting data from existing archives has been widely applied. By using scanning method, analogue format data from the archives can be transformed into digital format data, which are called raster.

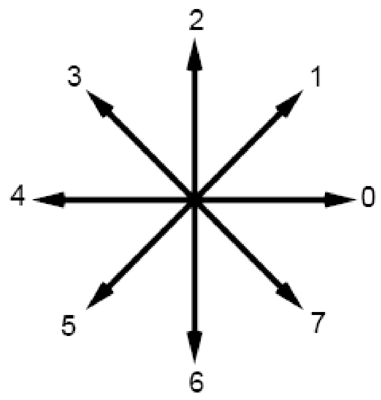
In the majority of graphical information systems, input data consist of raster images such as scanned maps in a GIS and engineering drawings in a CAD system. In order to manipulate, for example transform or select the lines and the other features from such raster images, these features must be extracted through a vectorization process [2]. Vectorization is quite important in document recognition, line detection, mapping, and drawing applications [3]. For advanced vectorization applications, the raster images must have high accuracy to preserve the original shapes of the graphical objects with the highest extent possible [4].

Line is one of the most fundamental elements in graphical information systems. Line detection is a common and essential task in many applications such as automatic navigation, military surveillance, and electronic circuits industry [5 - 6]. In previous studies, there are a large number of algorithms developed for detecting lines from raster images [7 - 8 - 9 - 10]. The vectorization methods implemented in these algorithms can be categorized into following six classes; (1) Hough Transform (HT) based methods, (2) thinning based methods, (3) contour based methods, (4) run-graph based methods, (5) mesh pattern based methods, and (6) sparse pixel based methods [11].

After the scanning, thresholding, and filtering stages, a traditional vectorization process consists of three stages (except HT based methods); (1) line thinning, (2) line following and chain coding, and (3) vector reduction (i.e. line segment approximation). In order to determine only the important points representing the medial axis, the lines on the image are to be thinned to one pixel wide by using the kernel processing [12]. Once line thinning stage is performed, the second stage is line following and chain coding the medial axis. In this stage, tracing process starts at an end pixel and continues based on the chain code directions until the last pixel in the line is reached. Fig-1 indicates the eight possible directions specified by the numbers from 0 to 7. Detail information on chain coding process can be found in [13 - 14 -15]. At the third stage, the medial axis coded in the second stage is examined and the vectors in the chain code are identified. In this process, the long vectors that closely represent the chain codes are formed while considering a user defined maximum deviation of the vectors from the chain codes [16 - 21].

In this study, a new model, MUSCLE (Multidirectional Scanning for Line Extraction), was developed to vectorize the straight lines through the raster images including township plans, maps, architectural drawings, and machine plans. Unlike traditional vectorization process, this model generates straight lines based on a line thinning algorithm, without performing line following-chain coding and vector reduction stages [22].

Figure 1. Chain Code Directions.



2. Material and Method

The logic behind the model is presented in Fig-2. The following main stages in the model are described in this section:

1. Threshold processing
2. Horizontal and vertical scanning of the binary image
3. Detecting wrongly vectorized lines
4. Correcting wrongly vectorized lines by using diagonal scanning
5. Applying topological corrections
6. Generating final vector data

2.1. Threshold Processing

In grayscale images, the objects may contain many different levels of gray tones. In this study, the objects are separated by using the threshold processing technique, with the assumption that the gray values are distributed over the image nearly homogeneous [17 - 18 - 19]. In the threshold process, a predetermined gray level (threshold value) is to be determined and every pixel darker than this level is assigned black, while every lighter pixel is assigned white. Therefore, the grayscale image was converted into a binary image [20 - 21].

2.2. Horizontal and Vertical Scanning of the Binary Image

In this stage, the horizontal and vertical lines were extracted from the binary image. The nearly vertical lines were obtained by scanning the images horizontally, while the nearly horizontal lines were obtained by scanning the images vertically. The forms of nearly vertical and nearly horizontal lines are shown in Fig-3. In Fig-3a, the lines which pass through the region 1 and 2 are defined as the nearly vertical lines and the nearly horizontal lines, respectively. Fig-3b and Fig-3c indicate the sample drawings for nearly vertical and nearly horizontal lines, respectively. In other words, if the slope (tangent) of the line is between -1 and +1, it is defined as “nearly horizontal line”. If the slope (tangent) of the line is less than -1 or greater than +1, it is defined as “nearly vertical line”.

Figure 2. Stages of MUSCLE Model.

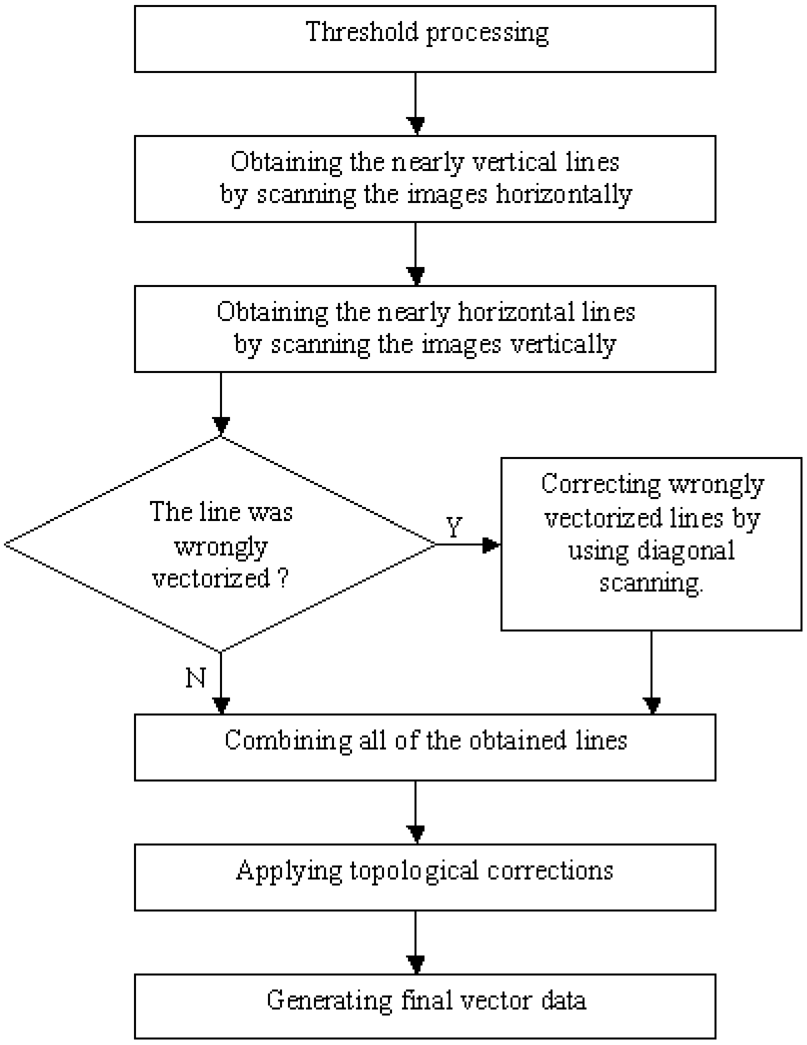
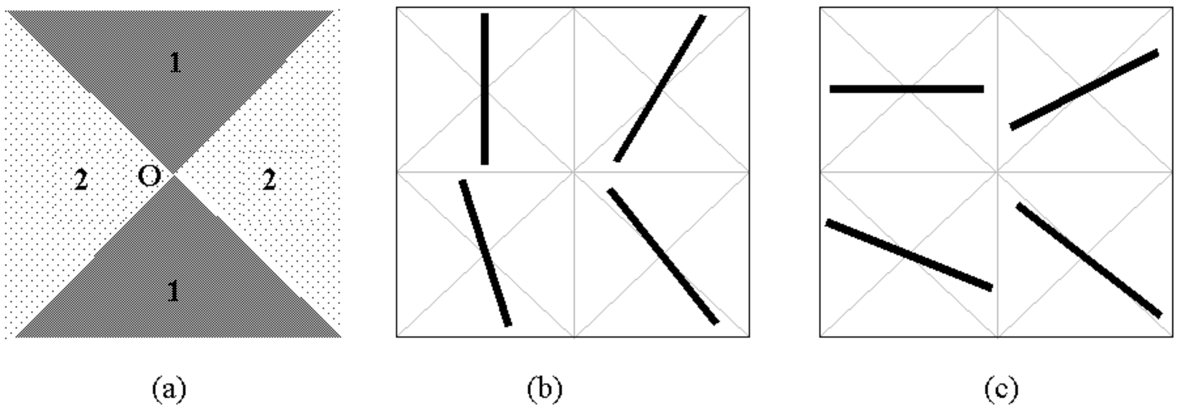

Threshold processing
Obtaining the nearly vertical lines by scanning the images horizontally
Obtaining the nearly horizontal lines by scanning the images vertically
The line was wrongly vectorized ?
Y
N
Correcting wrongly vectorized lines by using diagonal scanning.
Combining all of the obtained lines
Applying topological corrections
Generating final vector data

Figure 3. Samples for nearly vertical and nearly horizontal lines.



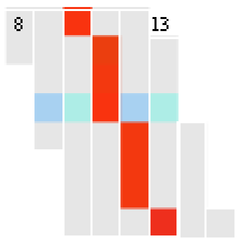
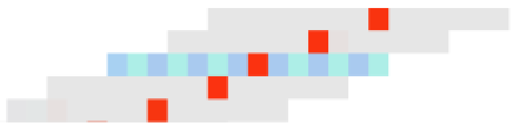
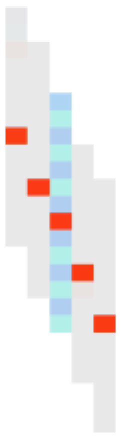
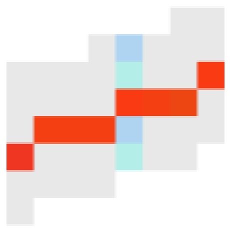
At the first step, each row on the binary image was scanned horizontally to determine the thickness of the lines and the position of the pixels, which were located in the mid-point of the lines. During this process, the value (black or white) of each pixel was checked by moving from left to right. Once the first black pixel was met, its column number was stored into the algorithm. While continuing to scan pixels, the column number of the first white pixel was also stored into the algorithm. Thus, the position of the middle pixel in the mid-point of the line could be determined by using the following equation, based on the image coordinate system:

$$\textit{The position of the middle pixel} = m + \textit{Absolute Value}\ ((n - m) / 2) \tag{1}$$

m	: column number of the first black pixel
n	: column number of the first white pixel

For example, assuming that 8th pixel is the first black pixel and 13th pixel is the first white pixel in Fig-4a. Using Equation 1, position of the middle pixel can be calculated as 10th pixel, which is then colored with red. After performing the same process for each row on the image, distribution of the red pixels for nearly vertical and nearly horizontal lines is indicated in Fig-4a and Fig-4b, respectively. In these Figures, the distribution of the red pixels indicates that the red pixels have continuity for nearly vertical lines; however they have discontinuity for nearly horizontal lines.

Figure 4. Determining red pixels by using horizontal and vertical scanning process.

	Nearly Vertical Lines	Nearly Horizontal Lines
Horizontal Scanning	 (a)	 (b)
Vertical Scanning	 (d)	 (c)

After the horizontal scanning processes were completed, only the red pixels were selected. Then, a neighborhood analysis was carried out based on the nearly vertical lines by taking the advantages of discontinuity on the nearly horizontal lines. In this method, a red pixel, which is adjacent to another red one, was searched along the lines. This process continued until no red pixels were found adjacent to each other, indicating that the end of the line has been reached. The beginning and ending points of all the nearly vertical lines were determined by using the same procedure.

At the second step, the binary image was scanned vertically, and then, the same process described above was carried out for all columns. Unlike horizontal scanning, the red pixels have continuity for nearly horizontal lines (Fig-4c); however they have discontinuity for nearly vertical lines (Fig-4d). Therefore, the neighborhood analysis was carried out based on the nearly horizontal lines and the beginning and the ending points of all the plenary horizontal lines were determined. After completing the horizontal (Fig-5a) and vertical (Fig-5b) scanning of the binary image, the final vectorized data (Fig-5c) were generated by vectorizing the nearly vertical and the horizontal lines.

2.3. Detecting Wrongly Vectorized Lines

In a case where two or more consecutive lines are nearly horizontal or nearly vertical, raster data become unmanageable and the process described in the previous stages generates wrongly vectorized lines. For example, initially, three consecutive nearly horizontal lines (AB, BC, and CD) were horizontally scanned as displayed in Fig-6a. Due to discontinuity of the red pixels between intersection points A, B, C, and D, the neighborhood analysis cannot be performed and vectorized data cannot be generated. When the raster image was vertically scanned during the second step, the neighborhood analysis yielded wrong vectorization results because of continuity of the red pixels. The algorithm recognizes point A as the beginning point of the line, skips point B and point C, and ends the line at point D. Therefore, the process generates a wrongly vectorized line between points A and D as indicated in Fig-6b.

The detection of wrongly vectorized data is performed by comparing the middle axis of the lines (red pixels) with the vectorized lines. The middle axis and the vectorized line have to be based on the same linear equation. For example, if a sample vectorized line (AB line) is a line with the beginning point of A(x_a,y_a) and the ending point of B(x_b,y_b), then, the linear equation for this vectorized line can be formed as follows:

$$(Y - y_a) / (y_a - y_b) = (X - x_a) / (x_a - x_b) \tag{2}$$

$$Y = ((y_a - y_b) / (x_a - x_b))\, X + ((y_b\, x_a - x_b\, y_a) / (x_a - x_b)) \tag{3}$$

When X coordinate of a red pixel is inserted into the linear Equation 3, and if the difference between the Y value derived from this equation and the Y coordinate of this pixel is greater than a user defined maximum deviation, the model defines this line as a wrongly vectorized line. After this process, the red pixels within the acceptable deviation range were eliminated from the image by converting them into the white pixel values. The wrongly vectorized lines with red pixels were remained unchanged within the image.

Figure 5. Horizontal and vertical scanning in vectorization process.

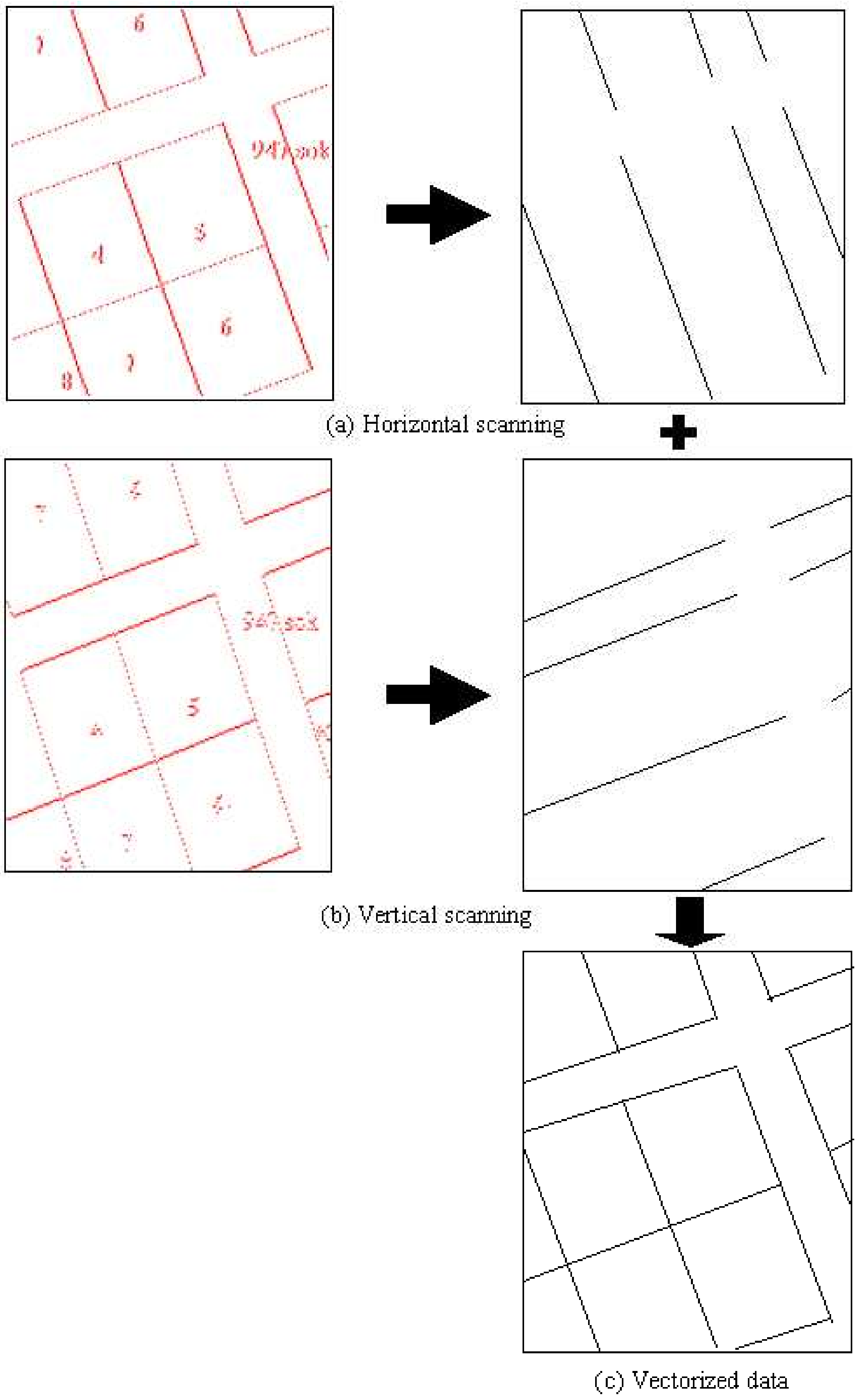
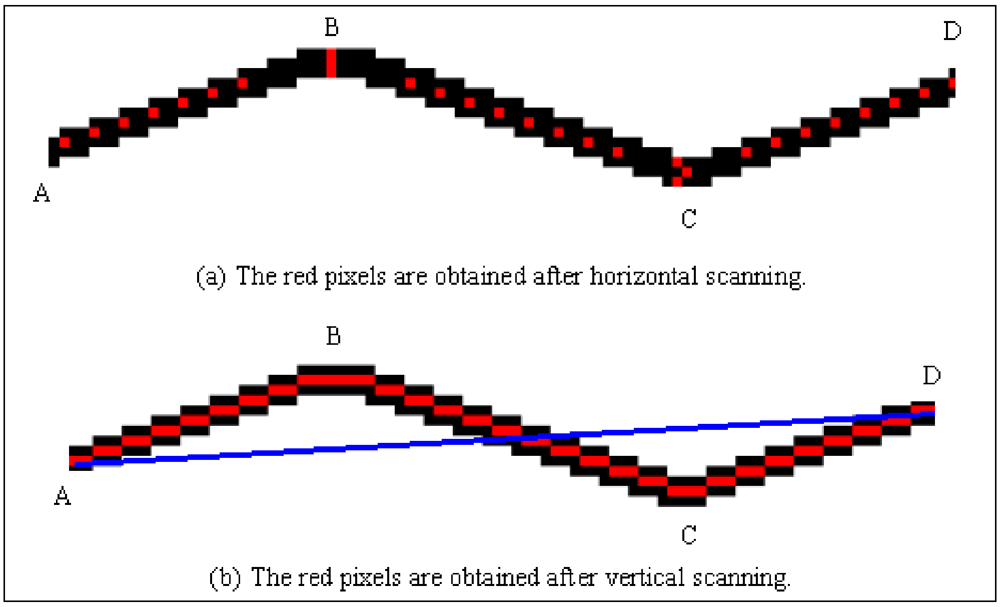


Figure 6. Detecting wrong vectorization after vertical and horizontal scanning.


B
D
A
C
(a) The red pixels are obtained after horizontal scanning.
B
D
A
C
(b) The red pixels are obtained after vertical scanning.

2.4. Correction of wrongly vectorized lines by using diagonal scanning

The image with the wrongly vectorized lines (Fig-7a) was diagonally (under 45^0 angle) scanned; first, from left to right, and then, from right to left (Fig-7b). In diagonal scanning process, if there were two consecutive red pixels along the direction of scanning, the second red pixel is eliminated. Thus, vectorized line took a discontinuous form as shown in Fig-7c. After applying the neighborhood analysis, the lines failed to have the acceptable number of pixels were not vectorized. The continuous pixels, determined by implementing diagonal scanning from both directions, were vectorized as indicated in Fig-7d. Then, corrected vector data were generated by combining both of the vectorized lines together (Fig-7e).

2.5. Applying Topological Corrections

Once horizontal, vertical, and diagonal scanning processes were completed, the topological corrections for the intersection points of the lines should have been performed. Topological corrections are very important to efficiently use the extracted vector data in GIS and other spatial applications.

2.5.1. Connecting End Points of the Lines

This circumstance occurs especially at the corner points. The correction was performed by using a special criterion as explained in the following section. This criterion was based on selecting a user-determined distance between the lines. The adjacent lines in the selected distance were connected at the algorithm. Then, the end points are joined by calculating the mean value of coordinates for two or more nodes as follows (Fig-8):

$$X_o = (X_1 + X_2 + .. + X_n) / n \qquad (4)$$

$$Y_o = (Y_1 + Y_2 + .. + Y_n) / n \qquad (5)$$

Figure 7. Correction of wrong extracted lines by using diagonal scanning procedure.

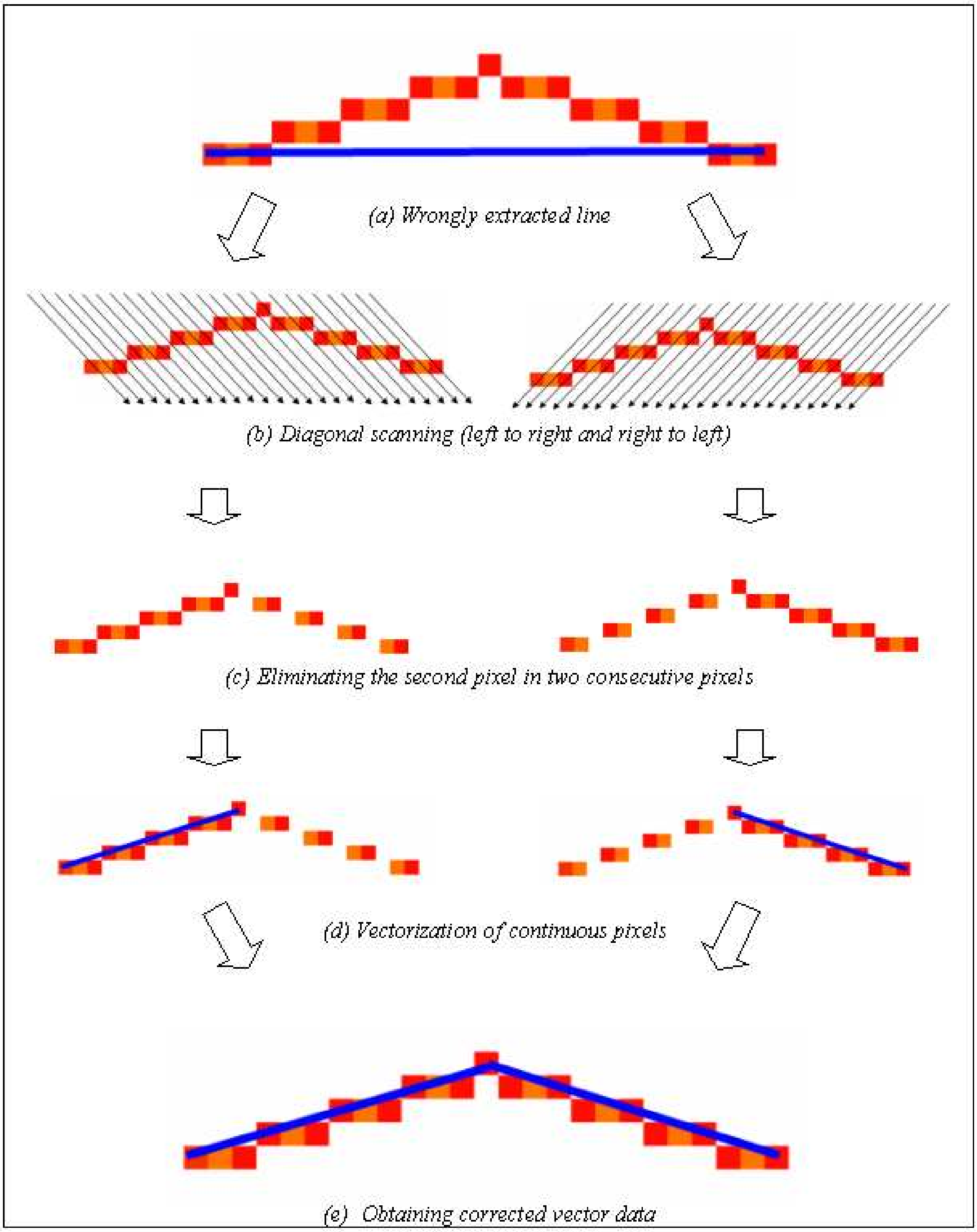
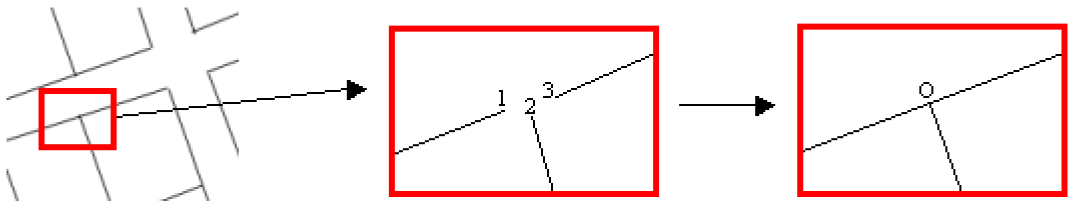

(a) Wrongly extracted line
(b) Diagonal scanning (left to right and right to left)
(c) Eliminating the second pixel in two consecutive pixels
(d) Vectorization of continuous pixels
(e) Obtaining corrected vector data

Figure 8. Connecting end points of the lines.

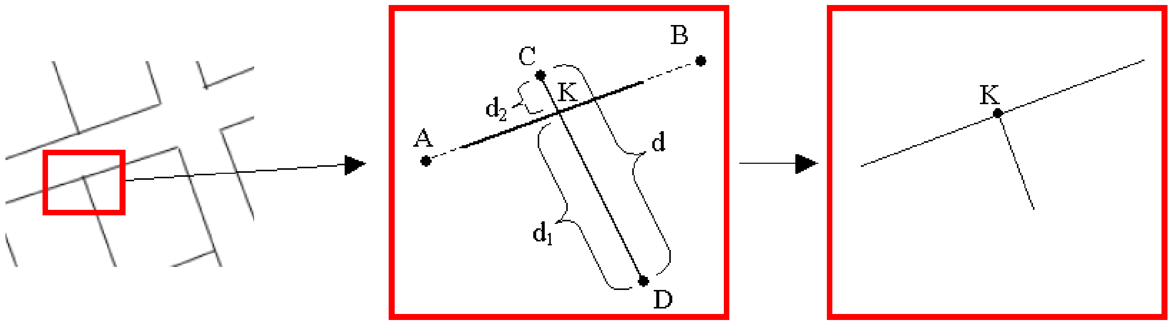


2.5.2. Correction of Overshoot and Undershoot Errors

In this process, firstly; the coordinates of the intersection points between the lines were calculated as follows (Fig. 9):

$$Y = ((y_a - y_b) / (x_a - x_b))\, X + ((y_b\, x_a - x_b\, y_a) / (x_a - x_b)) \tag{6}$$

Figure 9. The case of overshoot error. (d1 + d2 = d and d2 < p).



Secondly, point $A(x_a,y_a)$ and point $B(x_b,y_b)$ were used to determine the beginning and ending points of the line. The other line can be defined by point $C(x_c,y_c)$ and point $D(x_d,y_d)$ as follows:

$$Y = ((y_c - y_d) / (x_c - x_d))\, X + ((y_d\, x_c - x_d\, y_c) / (x_c - x_d)) \tag{7}$$

Then, the coordinates of the intersection point (K) for these two lines can be calculated by the formulations in Equations 8 and 9, respectively:

$$Y_k = (((y_d x_c - x_d y_c)/(x_c - x_d) - (y_b x_a - x_b y_a)/(x_a - x_b)) / ((y_a - y_b)/(x_a - x_b) - (y_c - y_d)/(x_c - x_d))) * ((y_a - y_b)/(x_a - x_b)) + ((y_b x_a - x_b y_a)/(x_a - x_b)) \tag{8}$$

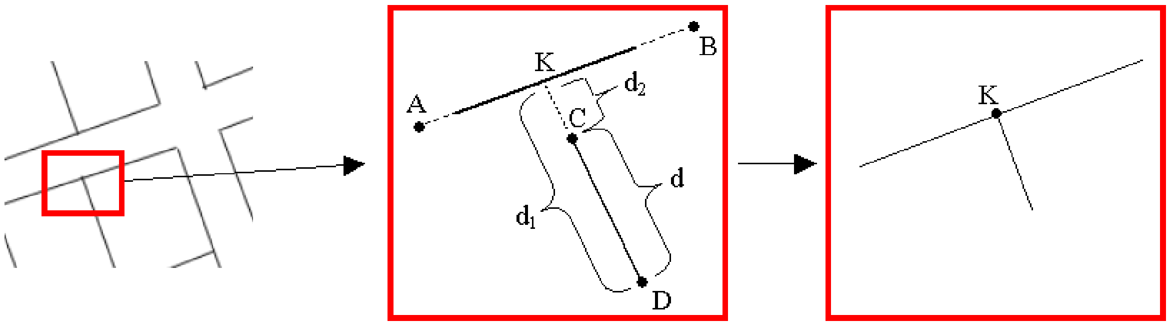
$$X_k = ((y_d x_c - x_d y_c)/(x_c - x_d) - (y_b x_a - x_b y_a)/(x_a - x_b)) / ((y_a - y_b)/(x_a - x_b) - (y_c - y_d)/(x_c - x_d)) \tag{9}$$

The distances (*dn*) from an intersection point, *K*(x_k,y_k), to the ending points of the intersecting lines can be calculated by using the following equation:

$$d_n = \sqrt{(x_n - x_k)^2 - (y_n - y_k)^2} \qquad (10)$$

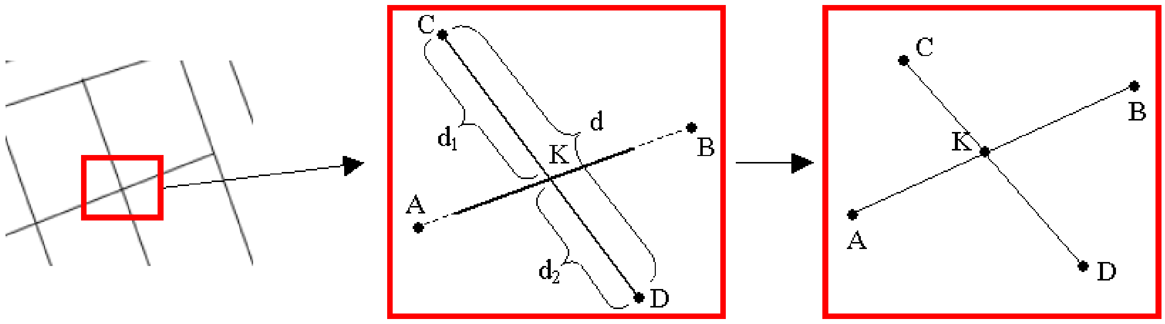
After determining the coordinates of the intersection point and the distance between intersection point and the ending points, ending points were examined to determine either there were overshoot or undershoot errors. If the length of a line (*d*) was equal to sum of the distances from two ending points (d_1 and d_2) to intersection point (*K*) and one of these distances was shorter than a user defined distance value (*p*: explained in Section 2.7.6), ending point was defined as overshoot (Fig. 9). If the length of a line (*d*) was shorter than the sum of the distances from two ending points to intersection point and one of these distances was shorter than a user defined distance value (*p*), ending point was defined as undershoot (Fig. 10). Then, the algorithm corrects the overshoot and undershoot errors by moving the ending point to the intersection point.

Figure 10. The case of undershoot error. (d1 + d2 > d and d2 < p).



If there is a case where the length of a line (*d*) was equal to sum of the distances from two ending points to intersection point and both of these distances were longer than a user defined distance value (*p*), ending point was not defined as neither overshoot or undershoot. In this case, intersection point is assigned to be a new point and the lines were divided into four new lines as indicated in Fig. 11.

Figure 11. The case where d1 + d2 = d, d1 > p, and d2 > p.



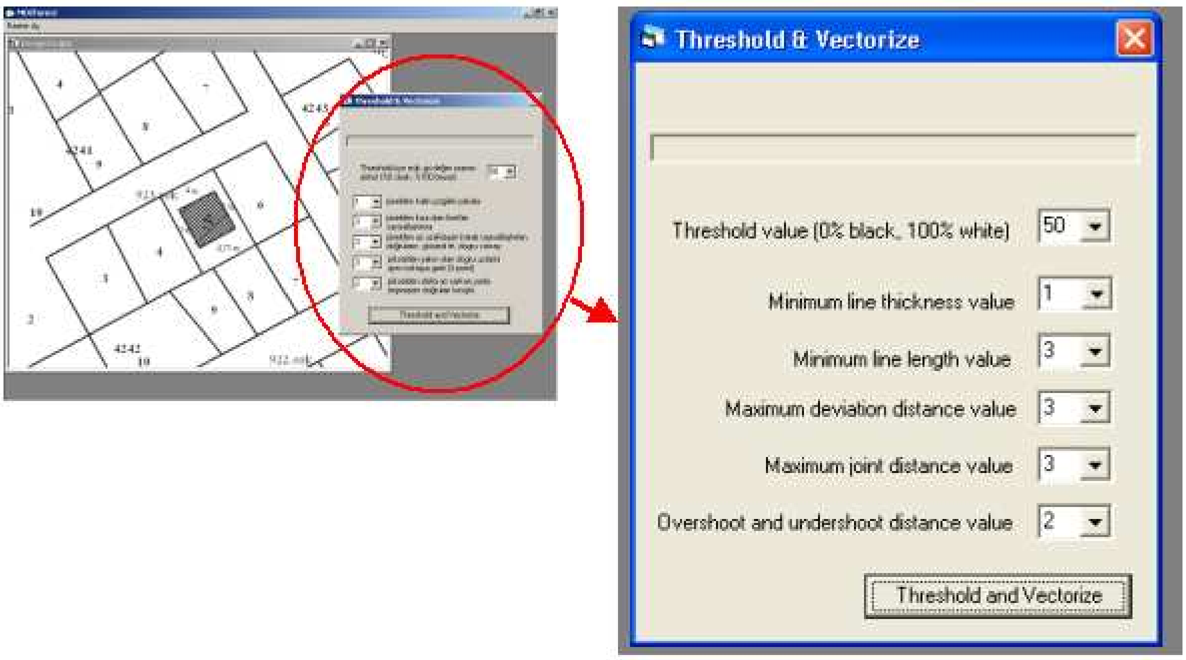
2.6. Generating the Final Vector Data

The product obtained after the vectorization process is saved to be ready to use in the “DXF” format, which is the “de facto” standard and well known exchange CAD format by all CAD software. Each output images generated after scanning and correction stages can be recorded as separate layers; therefore, the user can monitor the vectorization process and use these layers for various purposes.

2.7. Graphical User Interface and the Criteria

The algorithm was programmed in Visual Basic 6.0 platform and the graphical user interface of the model is displayed in Fig-12. The user is expected to define the specified criteria for the current raster and the future vector data before the vectorization process. The performance of the vectorization depends remarkably on these six criteria.

Figure 12. Interface of the algorithm including input window for threshold and vectorization criteria.


Threshold & Vectorize

Threshold value (0% black, 100% white) 50
Minimum line thickness value 1
Minimum line length value 3
Maximum deviation distance value 3
Maximum joint distance value 3
Overshoot and undershoot distance value 2

2.7.1. Threshold Value

Threshold value was used to define two main classes, black and white, based on the gray value distribution of the raster image. Gray values under the threshold value become black, while values above the threshold become white. Depending on the threshold value, some lines can be thicker and clarified, while some lines can be thinner and fader. Therefore, selecting an optimum value for the threshold according to the raster properties is very important for the success of the vectorization process. This optimum value can be found after having some experiences based on the trial. For example, by looking at the darkness of the output image, user can come up with optimum threshold value.

2.7.2. Minimum Line Thickness

The second criterion was the thickness of the lines in the raster image to be vectorized. For example, if the user selects three as a minimum number of pixels for line thickness, lines thinner than three pixels are ignored and not vectorized. This is useful when the user wants to extract certain objects like parcel boundaries with certain line thicknesses.

2.7.3. Minimum Line Length

Another criterion for eliminating the unnecessary details and for vectorizing the lines with sufficient length was the assignment of the minimum number of the pixels for the line length. For example, if a user selects six for this criterion, the algorithm eliminates lines that are shorter than six pixels during the vectorization process. Thus, unwanted objects such as text and noise can be eliminated from the raster image much more easily.

2.7.4. Maximum Deviation Distance

The determination of the wrongly vectorized lines was performed by considering the deviation distance between the red pixels and the vectorized line as described in section 2.3. The maximum deviation distance is defined by the user based on the sensitivity of the job. For instance, if a user selects three for this value, the vectorized lines that are more than three pixels away from the red pixels would be considered to be in the incorrect form, whereas the lines that are closer than this value would be considered to be in the correct form.

2.7.5. Maximum Joint Distance

During the topological correction of the vectorized data, the ending points of the lines that were closer to each other must have been merged for joining the broken lines. For achieving this, the user is allowed to define and input a maximum joint distance value. Consequently, if the distance between ending points of the lines is less than the input value, the model connects these points at a shared intersection point.

2.7.6. Overshoot and Undershoot Distance

The user is allowed to define and input a distance value for correcting the undershoot and overshoot errors during the vectorization process. For example, if a user assigns the value of three for this criterion, dangles and gaps which are smaller than three pixels would be eliminated and geometrically corrected as explained in section 2.5.

3. Results and Discussion

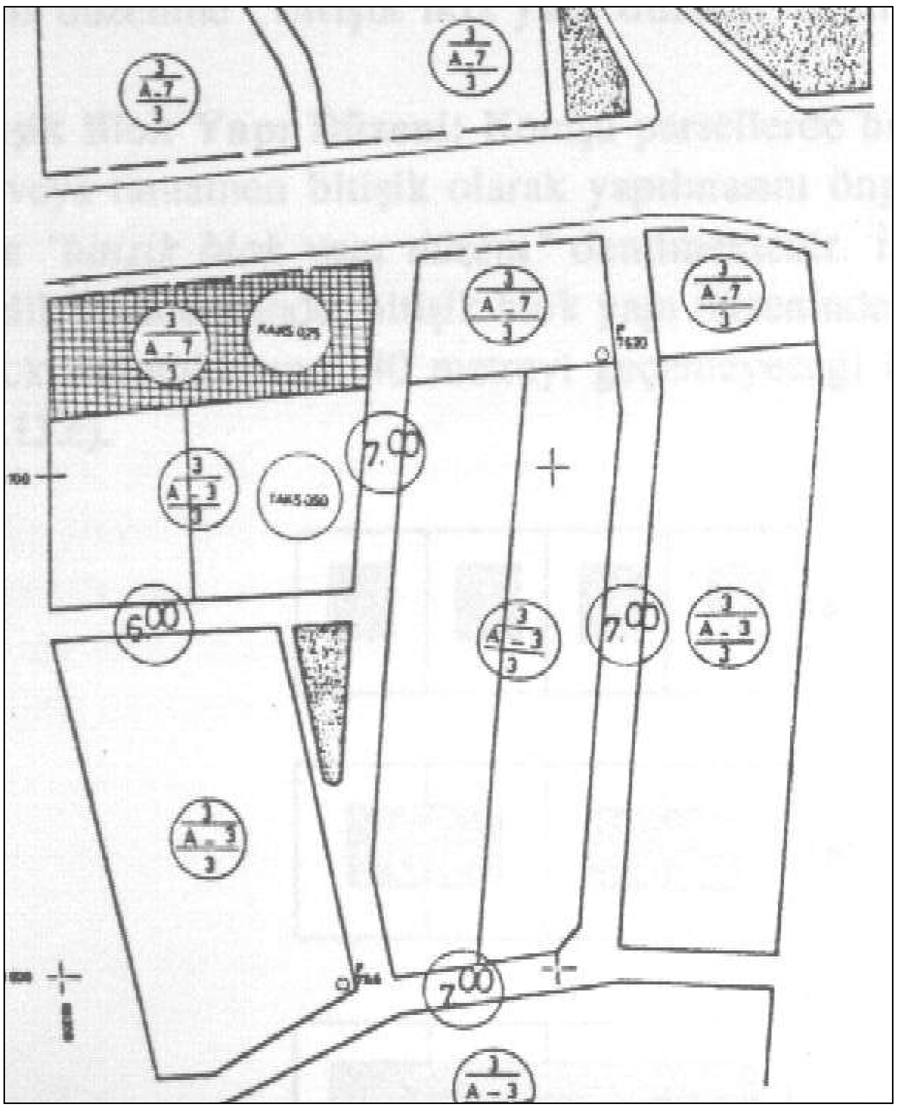
The algorithm was tested on a sample raster data of a township plan. In Turkey, township plans are highly desired maps which are generally in analogue format and subject to intensive digitizing tasks. In the model application, vectorization algorithm was applied on the map exposed in Fig-13.

The user-defined values were assigned to the criteria described in the previous section. Setting a low threshold value (50%) caused discontinuity along the lines during vectorization processes (Fig-14). However, setting a high threshold value (90%) resulted in very thick lines with noises; therefore, it caused the algorithm to induce errors and yielded bad results (Fig-15).

If the thickness value of the line was selected to be very big, some of the necessary lines were ignored in the process (Fig-16). For example, when the minimum number of pixel for the line thickness was decreased, the one pixel- thickness was ignored in the vectorization process.

For the third criterion, when the number of pixels for the minimum line length was selected as a very big value (e.g. 9 pixels), the noise problem was mostly solved, however, some of the lines along the parcel boundaries were failed to be extracted (Fig-17a). On the other hand, when the value of this criterion was set to be a low value (e.g. 3 pixels), almost all of the lines along the parcel boundaries were extracted. However, it was observed that some of the noises over the lines were also vectorized (Fig-17b).

Figure 13. The input raster dataset used in the application.



If the value of the fourth criterion, the deviation distance between red pixels and the vectorized line, was selected to be a high value, the chance of ignoring the lines that were wrongly vectorized tends to increase (Fig-18). When the fifth criterion was selected to be a high value, the lines that were not intended to be processed were merged and vectorized as seen in Fig-19. The model application also revealed that setting a low value for the sixth criterion had yielded better results regarding the correction of the overshoot and undershoot errors.

Figure 14. The results of the vectorization process using a low threshold value.

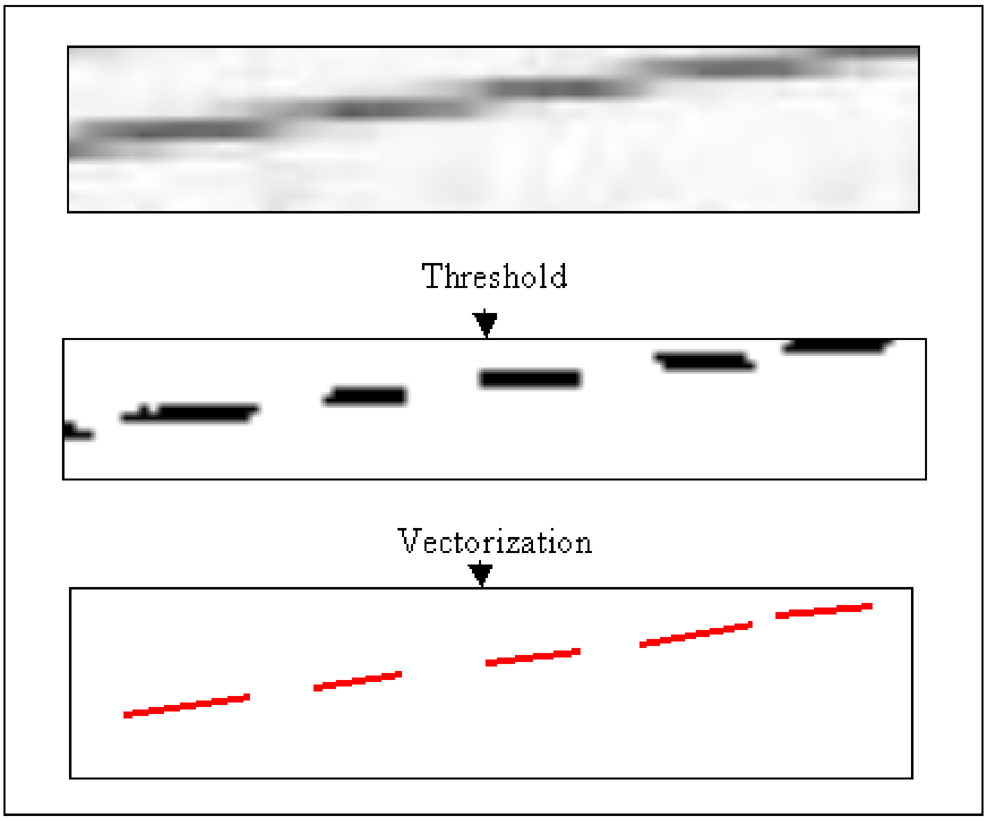


Figure 15. The results of the vectorization process using an extra high threshold value.

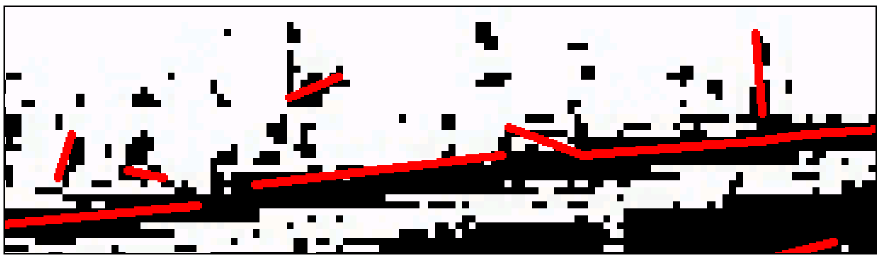


Figure16. The results of the vectorization process using a high line thickness value.

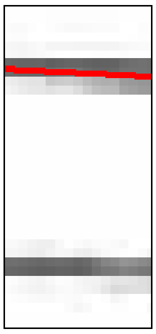


Figure 17. The results of the vectorization process using as high (a) and low (b) value for the third criterion.

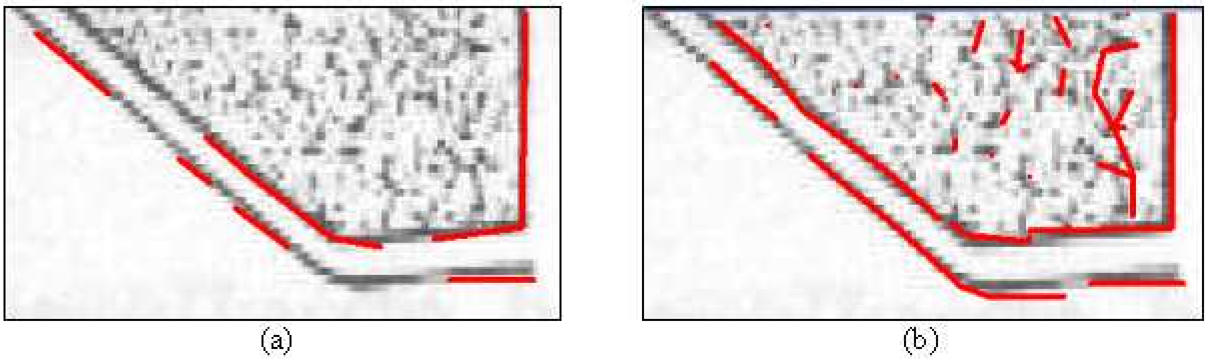


Figure 18. The results of the vectorization process using a high value for the fourth criterion.

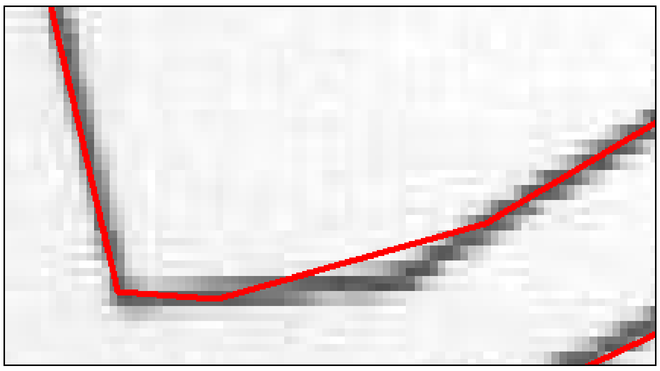
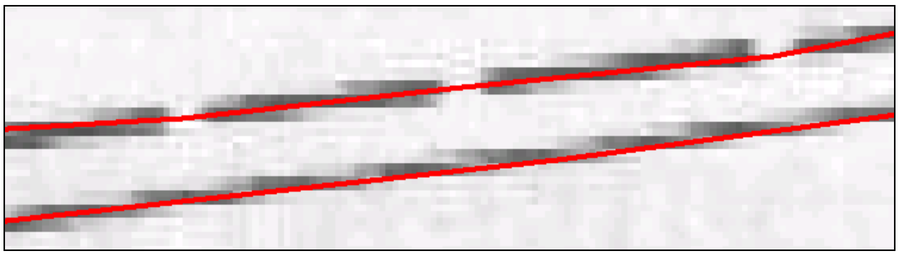


Figure 19. The results of the vectorization process using a high value for the fifth criterion.

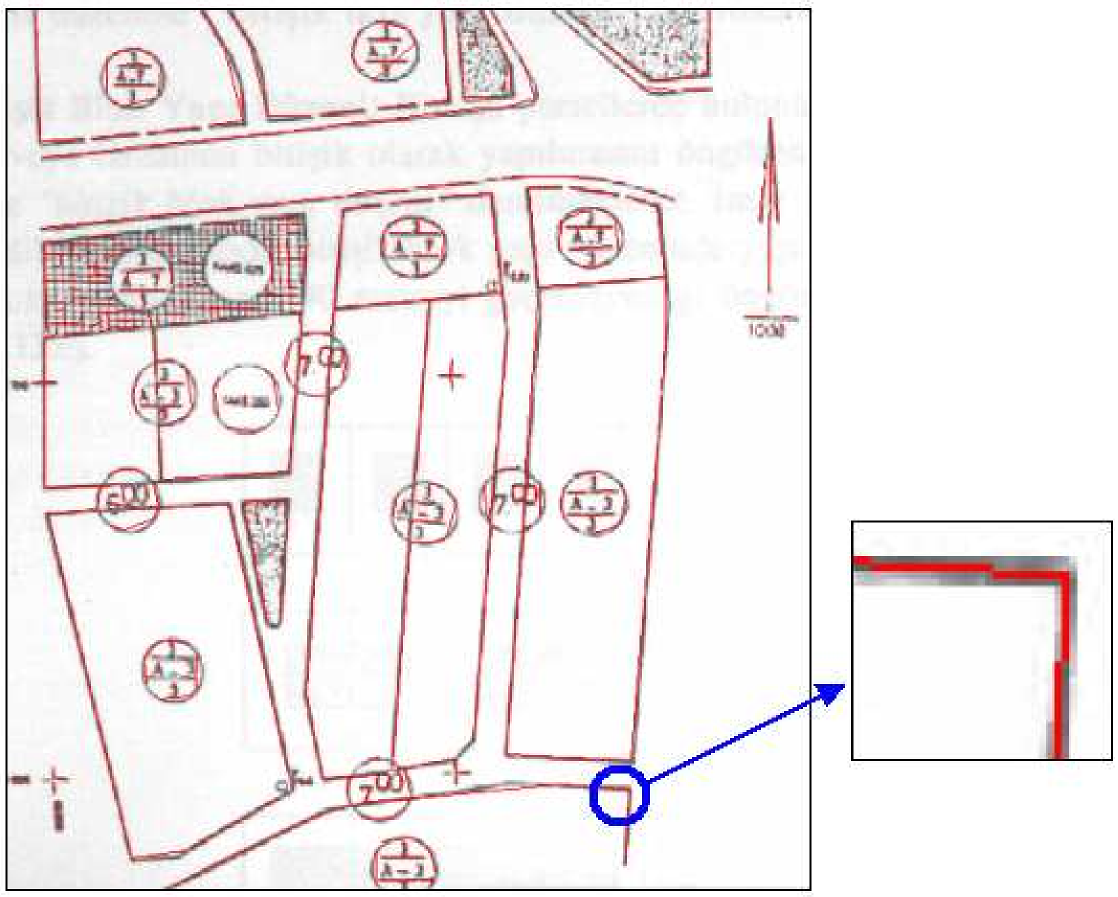


In order to generate the best vectorization out of the sample data, a number of alternative values were tested over the criteria. The goal was to vectorize all the lines at the parcel boundaries and to eliminate the remnants of the noises and texts as mush as possible. Our results indicated that the best vectorization was performed by using the following combinations over the criteria:

- The threshold value as 65%,
- The minimum line thickness value as 1 pixels,
- The minimum line length value as 6 pixels,
- The maximum deviation distance value as 4 pixels,
- The maximum joint distance value as 4 pixels,
- The overshoot and undershoot distance value as 2 pixels.

The final vector data generated by applying the vectorization process with the criteria above are displayed in Fig-20. Consequently, a great success was achieved for fixing the lines at the parcel boundaries; however, the noises could not be removed completely.

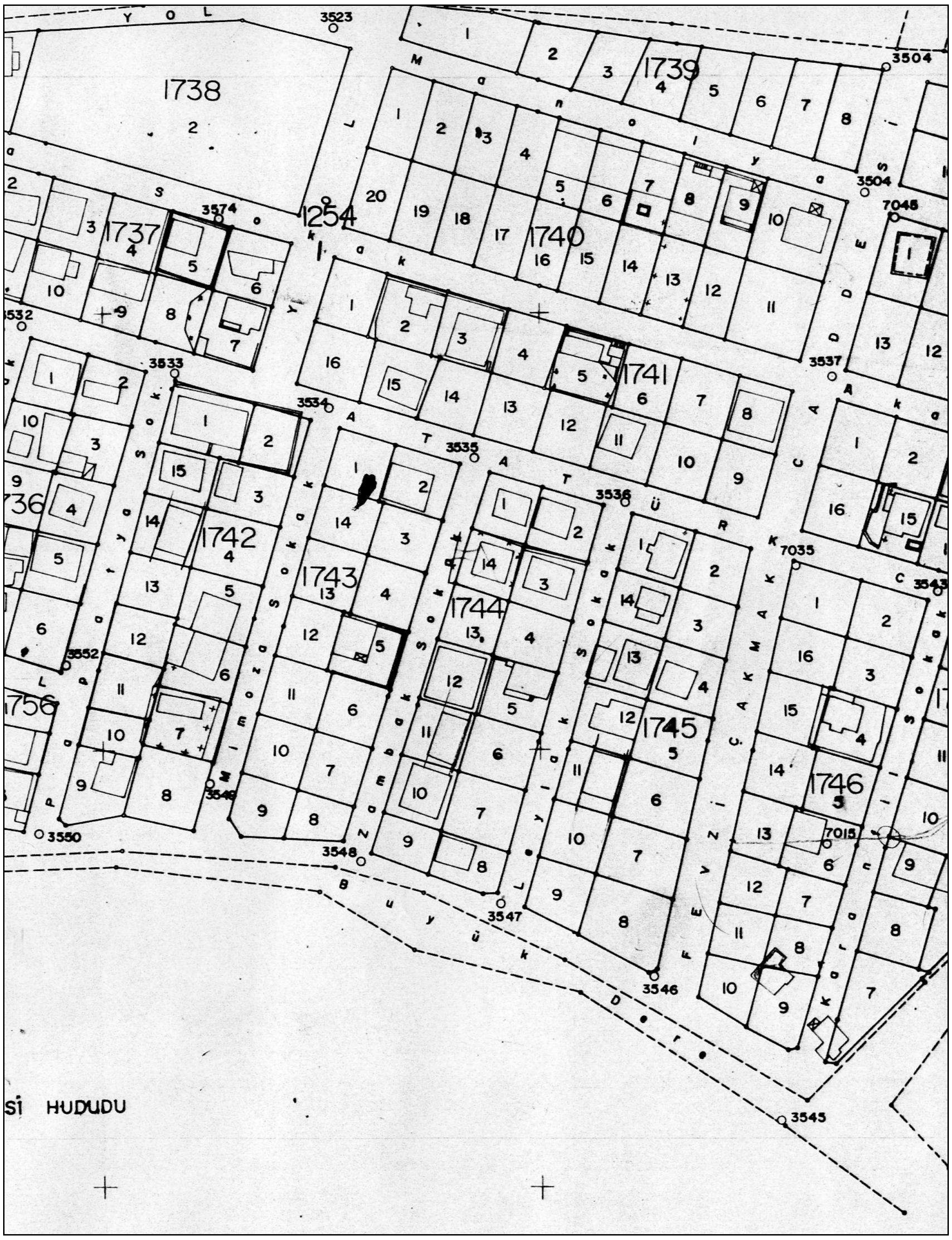
Figure 20. Overlay of the first test image and final output processed by the vectorization algorithm.



The performance of the proposed model was evaluated by comparing the results with two commercial raster-to-vector programs, WinTopo and Scan2CAD. For comparison, the images in Fig. 13, Fig. 21, and Fig. 22 were processed by all the three models and the results were indicated in Table

1. The images were acquired in 200 dpi and 8 bit radiometric resolution by using scanner. So, before the threshold processing, images have 256 grey values.

Figure 21. The second test image used in the comparison.



Our results indicated that WinTopo completed the vectorization process in the shortest computation time; however, it divided the lines into many pieces, which resulted in too many objects. This also required intense and time consuming post-processing process after the vectorization. Scan2CAD performed a quality vectorization process with acceptable number of objects. However, there were still some errors on vector images.

Figure 22. The third test image used in the comparison.

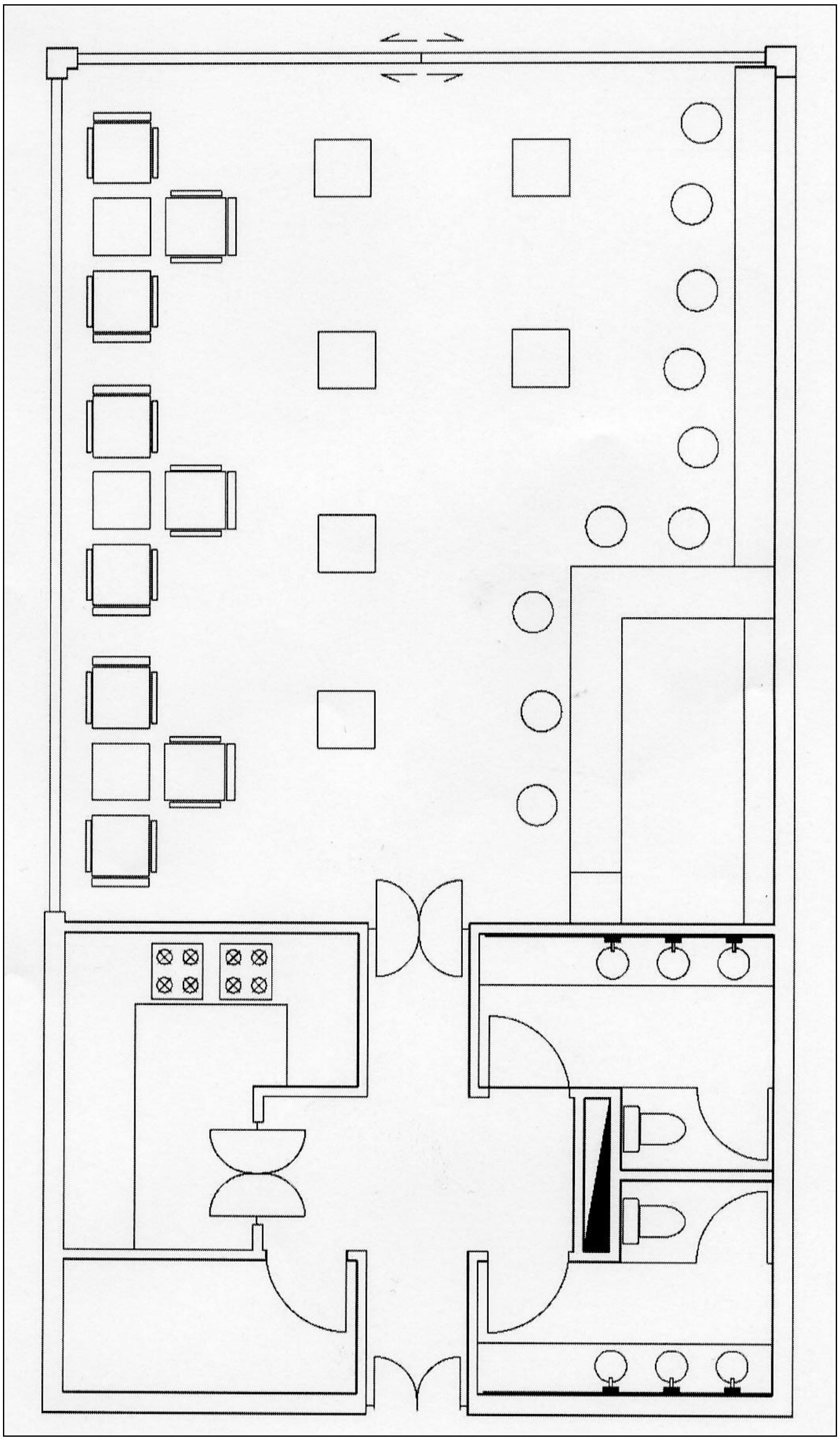


Table 1. The summary of the sample application considering number of final objects and processing time.

	The Number of Final Objects			Processing Time (sec)		
Programs	Image 1	Image 2	Image 3	Image 1	Image 2	Image 3
MUSCLE	628	4074	1391	6.0	68.0	33.0
WinTopo	2308	7040	3788	0.5	3.0	2.0
Scan2CAD	712	1855	573	3.0	36.0	22.0

MUSCLE also provided very successful results, especially for the images with straight lines. The model generated an individual vector for each piece of line, which reduced the number of final objects. This feature also reduced the computation time in the process of correcting the errors. However, total time spent on vectorization process was longer than the time spent by using the other two commercial programs since the current version of the MUSCLE was not professionally optimized.

5. Conclusions

In this study, a new model, MUSCLE, was developed by implementing an appropriate computer programming to automatically vectorize the raster data with straight lines. The model allows users to define specified criteria which are crucial for the success of vectorization process. A basic vectorization application presented in this study cannot be totally generalized, yet it showed that this model is able to successfully vectorize raster images with straight lines. More work in necessary to improve the quality of the vectorized image such as automated selection of user defined optimum combinations for the criteria.

The unique contribution of this model can be described briefly as its potential for vectorizing straight lines based on a line thinning and simple neighborhood analysis, without performing line following-chain coding and vector reduction stages. Besides, the model has the ability to vectorize not only the maps with linear lines such as cadastral map sheets, township plans, etc., but also other documents such as technical drawings, machine pieces, architectural drawings, etc., which are to be converted from a analogue format to digital format. It is highly anticipated that MUSCLE can provide a quick and simple way to efficiently vectorize raster images. There are several opportunities to improve this model such as vectorizing curve lines and refining model interface.

The sample application, where the performance of the model has been compared with the two well known commercial vectorization programs, indicated that the current version of MUSCLE can perform a successful vectorization task. It was believed that refining and optimizing the algorithm by professionals would improve the vectorization process. Further researches are also required through an extended and a diversified sample space to expose the feasible application areas of MUSCLE. Yet, our results suggest that MUSCLE may offer opportunities for replacing the complicated digitizing tasks with a concise, automatic and computer-aided process.

References

1. Longley, P.A.; Goodchild, M.F.; Maguire, D.J.; Rhind, D.W. In *GIS Data Collection, Geographic Information Systems and Science* **2001**, Hoboken, NJ: John Wiley & Sons, 203-224.
2. Nieuwenhuizen, P.R.; Kiewiet, O.; Bronsvoort, W.F. An Integrated Line Tracking and Vectorization Algorithm. *Proceedings of Eurographics '94,* Oslo, Norway, Sept., **1994**, *3*(3), 349-359
3. Zhong, D. X. Extraction of embedded and/or line-touching character-like objects. *Pattern. Recogn.* **2002**, *35*(11), 2453 – 2466.
4. Dori, D.; Wenyin, L. Automated CAD Conversion with the Machine Drawing Understanding System: Concepts, Algorithms, and Performance. *IEEE T. Syst. Man Cyb.* **1999** - Part A: Systems And Humans, 29(4), 411-416.
5. Shpilman, R.; Brailovsky, V. Fast and robust techniques for detecting straight line segments using local models. *Pattern Recogn. Lett.* **1999**, *20*(9), 865-877.
6. Climer, S.; Bhatia, S. K. Local Lines: A linear time line detector. *Pattern Recogn. Lett.* **2003**, *24*, 2291–2300.
7. Miao, L.; Liu, X.; Peng, Q.; Bao, H. BRDC: binary representation of displacement code for line. *Comput. Graph.* **2002**, *26*(3), 401–408.
8. Lagunovsky, D.; Ablameyko, S. Straight-line-based primitive extraction in grey-scale object Recognition. *Pattern Recogn. Lett.* **1999**, *20*(10), 1005-1014.
9. Madhvanath, S.; Kim, G.; Govindaraju, V. Chaincode Contour Processing for Handwritten Word Recognition. *IEEE T Pattern Anal.* **1999**, *21*(9), 928-932.
10. Hori, O.; Tanigawa, S. Raster-to-vector Conversion by Line Fitting Based on Contours and Skeletons. *Proc., Int. Conf. Document Analysis and Recognition.* **1993**, Tsukuba (Japan), 353-358.
11. Wenyin, L.; Dori, D. From Raster to Vectors: Extracting Visual Information from Line Drawings. *Pattern. Ana.l Appl.* **1999**, *2*(2), 10-21.
12. Treash, K.; Amaratunga, K. Automatic Road Detection in Grayscale Aerial Images. *J. Comput. Civil Eng*. **2000**, *14*(1), 60-69.
13. Freeman, H. On the encoding of arbitrary geometric configurations. *IEEE Trans. Electron*. **1961**, *10*, 260-268.
14. Freeman, H. Computer Processing of line-drawing images. *Comput. Surv.* **1974**, *6*(1), 57-97.
15. Freeman, H.; Davis, L.S. A corner-finding algorithm for chain-coded curves. *IEEE Trans. Electron*. **1977**, *26*(1), 297-303.
16. Sun, J. and Wu, X. Shape Retrieval Based on the Relativity of Chain. *Lect. Notes Comput. Sc.* **2007**, V. 4577/2007, P. 76-84
17. Wang, L.; Bai, J. Threshold selection by clustering gray levels of boundary. *Pattern Recogn. Lett.* **2003**, *24*(12), 1983–1999.
18. Belkasim, S.; Ghazal, A.; Basir, O.A. Phase-based optimal image thresholding. *Digit. Signal Process.* **2003**, *13*, 636–655.
19. Liao, P.S.; Chen, T.S.; Chung, P.C. A Fast Algorithm for Multilevel Thresholding. *J. Inf. Sci. Eng.* **2001**, *17*, 713-727.

20. Hasson, N.N.; Aljunid, S.A.; Badlishah, A. R. Simplification of Raster Images to Extract Visual Information. *International Journal of Computer Science and Network Security* **2006**, *6*(11), p.49.
21. Jennings, C. *Computer Vision for Line Drawings*. **1993**, MSc Thesis, University of Calgary.
22. Karas, I.R. *Evaluating Topological Relationships of The Objects in 3D GIS and Network Analysis*, **2007**, PhD Thesis, Yildiz Technical University, Istanbul, Turkey.