A NEURAL NETWORK ALGORITHM FOR VECTORIZATION OF 2D MAPS

H. Karabork[a], B.Kocer[b], I.O. Bildirici[a], F.Yildiz[a], E.Aktas[c]

[a]Selcuk University, Faculty of Engineering, Department of Geodesy & Photogrammetry, 42075 Kampus, Konya, Turkey - hkarabork@hotmail.com, (bildirici, fyildiz)@selcuk.edu.tr
[b]Selcuk University, Faculty of Engineering, Department of Computer Engineering, 42075 Kampus, Konya, Turkey - bariskocer@selcuk.edu.tr
[c]Kahramanmaras Sutcu Imam University, Faculty of Engineering, Department of Electric & Electronic Engineering, Maras, Turkey - eaktas@ksu.edu.tr

Commission II, WG II/3

KEY WORDS: Spatial Information Sciences, Analysis, Artificial_Intelligence, Accuracy, CAD, Image, Information, Tracking

ABSTRACT:

In the last twenty-five years a great number of vectorization methods were developed. In this paper we first give an overview about the most known methods, and then propose a vectorization algorithm based on Artificial Neural Network method. This algorithm is implemented by using C# programming language. Because distortions in size and location after vectorization are important for mapping applications, we tested our algorithm and software on a cadastral map. We also compared the results of our algorithm with the results of Sparse Pixel Vectorization (SPV) algorithm. Although SPV algorithm delivers better results, our algorithm also givesacceptable results, which are suitable for mapping purposes.

1. INTRODUCTION

Vectorization or raster-vector conversion is no doubt a central part of graphics recognition. It deals with converting the scanned image to a vector form that is appropriate for further processing and analysis. Many vectorization algorithms have been developed since the computer technology was available for this purpose, and a number of software products are in the market. Because many software packages are available, the raster-vector conversion problem may be considered to be solved. Nonetheless, there are still problems of precision, robustness and stability of the vectorization processes (Tombre and Tabbone, 2000).Vectorization can be divided into low-level vectorization and high-level vectorization. The former deals with the recognition of the basic graphic objects, including straight lines, arcs, circles and curves. The latter deals with the structural analyses by domain knowledge (Song et al, 2002a). For instance, Optical Character Recognition (OCR) and graphic object recognition are such applications where high-level vectorization are needed (Wenyin and Dori, 1999).Most of the vectorization algorithms are developed to be versatile for many types of engineering drawings, such as maps, mechanical drawings, and electronic drawings (Song et al, 2002b). Most of the algorithms give good results if graphic objects are isolated, but are error prone if they intersect or touch one another (Song et al, 2002c). A typical problem is that line following will stop when a crossing on the line is countered, so that the line will be recognized as several lines separated by crossing on it. To reconstruct the original line, post-processing is needed to analyze crossings and connect the collinear lines. If the thickness of the original line is not consistent because of noise and degradation, it is complicated to reconstruct the line appropriately (Song et al, 2000). Compared to raster images, vectorized images are an economic alternative (Xu and Bai, 2000; Bai and Xu, 2001). Vector files have following benefits in mapping applications (Fernandez, 1998):

•Much less storing capacity is required.
•They can be scaled to any size without losing resolution.
•They can be modified with CAD/GIS programs. This is an important aspect, because we can add a new road or bridge to a map without redrawing it from scratch.

Because of the increasing use of the computer technology for mapping purposes there is a need to digitize existing hardcopy maps. Vectorization is an alternative to manual digitizing, which is a time consuming method. Maps have an exact scale, and the objects on it have certain sizes and exactly defined locations. Therefore the accuracy of vectorization is crucial for mapping.

In this paper we first give state of the art on the current works we reached. Among these works we did not found any application on map vectorization that is implemented with artificial neural networks (ANN) method that we use. We discuss our ANN algorithm, which is implemented with our own developed program, named VecNET. It was tested with a test image, which is a cadastral map consisting of 203 parcels with complex boundaries. Our test is primarily conspired to judge the positional accuracy of our algorithm. Therefore we plotted a vector drawing (cadastral map), and then scanned it; finally we vectorized it with VecNET. We then compared the vectorized drawing with the original drawing (ground truths). Additionally we vectorized the map with MDUS program developed by Wenyin and Dori (1997) that includes SPV algorithm. We compared the results in the same way. Although the results from MDUS seem to be more accurate, VecNET also delivers good vector drawings with acceptable distortions in size and position, which is crucial for mapping purposes.

2. VECTORIZATION METHODS

Most of the vectorization methods were developed in the last 25 years. The methods we introduce here are Hough Transformation based methods, thinning based methods, contour based methods, run graph based methods and sparse pixel based methods.

The Hough transformation (HT) is a well-known method for recognizing geometric primitives from raster images. The basic version of the algorithm just detects lines but it can simply be generalized to extract more complex objects. The major advantage of the algorithm is that it can extract desired graphic objects from a noisy environment, but it handles every pixel at least once, and needs a considerable amount of computation time (Song et al, 2002a; Xu and Bai, 2000). Wenyin and Dori (1999) discusses the application of HT for vectorization of straight-line images by transforming spatially extended patterns in binary image data into spatially compact features in a parameter space. Since this algorithm visits every pixel once, its time complexity is linear with the total number of pixels (Wenyin and Dori, 1999).

HT detects parameterized curves in images by mapping the image edge pixels into manifolds in the parameter space. The parameters that are consistent with many types of the curves in image and methods finding peaks in the parameter space can be used to detect the image curves (Olson, 1999).

The term “skeleton” has been used in general to denote a representation of a pattern by a collection of thin (or nearly thin) arcs and curves. Some authors refer to a thinned image as a line drawing representation of a pattern that is usually elongated. The terms “thinning” and “skeletonization” have become almost synonymous in the literature, and “skeleton” is used to refer to the result, apart from the shape of the original pattern or the method used (Lam et al, 1992).

Thinning or skeleton based methods usually make use of an iterative boundary erosion process to remove outer pixels, until only one-pixel-wide skeleton stays. Then, the pixels on the skeleton are linked by using a line following procedure. Finally, every graphic object is recognized by extending and fitting from the point chains. Drawbacks of thinning include lost of line thickness information and difficulties in handling distortions at intersections. The frequent pixel access and numerous merging operations of short lines also slowdown the speed of vectorization (Song et al, 2002c).

Thinning is also a crucial preprocessing step of feature extraction in many pattern recognition systems. In the OCR applications, for instance, the stroke extraction process usually follows thinning process. Thinning also plays an essential role in reducing the complexity of data acquired for vectorization of a binary line drawing, such as maps and mechanical drawings (Xu and Bai, 2000).

The contour based methods first follow the contours and then detect corresponding contours to identify line-like areas. Medial axes, mostly represented as point chains, are created between these contour pairs. These methods generally consist of four main steps: (i) extraction of the contour vectors, (ii) matching of them, (iii) creation of medial axis, and (iv) junction processing (filling the gaps remained after matching process). (Tombre and Tabbone, 2000 ; Song et al, 2002c.)

A remarkable disadvantage of contour based methods is missed pairs of contour lines at junctions, resulting in gaps that break the vectors. Keeping line widths compensates this disadvantage, which is important for post-processing. Moreover, in non-line-like areas (intersections and degraded parts), the matching of contour is generally not one-to-one, but rather one-to-many or even many-to-many, which makes the analysis more complex (Song et al, 2002c; Dori and Wenyin, 1999).

Run graph based methods examine the raster image in either row or column direction to calculate the run length encoding. The runs then analyzed to create a graph structure. The midpoint of runs in a line-like area is polygonalized to form a point chain, which becomes an edge in the graph structure, and a non-line-like area becomes a node connecting the adjacent edges. Finally, graphic object recognition is performed on the graph structure. These methods are not robust when the image quality is degraded. Furthermore, the dependence on the scanning direction leads to unsatisfactory performance for diagonal lines. They work well for sparse line images containing mainly horizontal and vertical lines, however (Song et al, 2002a; Song et al, 2002c).

The most common technique among run graph based methods is Run Length Encoding (RLE) (Song et al, 2002a). Run graph based methods are considered to be able to solve the problem of having both the connectivity and the line width information, because they record the node areas (junctions). A run graph representation can be viewed as a semi vector representation, because it employs nodes corresponding to the end points of vectors, along with a set of adjacent runs to express the digital segment between these two nodes. A polygonalization procedure is applied to the middle points of the set of runs representing the vector to find out its attributes. Like most vectorization methods, run graph based methods are vulnerable by noise and may cause unsatisfactory results at junctions, since the intersection points are not precisely located during the construction of the run graph representation (Dori and Wenyin, 1999).

The basic idea of Orthogonal Zig-Zag (OZZ) method is to follow the course of a one-pixel wide ‘beam of light’, which turns orthogonally each time it hits the edge of the area covered by the black pixels, such as a bar area. The midpoint of each run, which is the intersection of the light beam and the area within the area, is saved. If a run is longer than a predefined threshold the run stops there, an orthogonal run is made and its midpoint is recorded. This may happen when tracking along a nearly horizontal or vertical area (Wenyin and Dori, 1999). Based on the OZZ idea, Wenyin and Dori developed the Sparse Pixel Vectorization (SPV) algorithm. SPV improves the OZZ method in these ways: **a)** the general following procedure begins with a reliable starting medial axis point found by a special procedure for each black area; **b)** a general following procedure is used to handle all three cases of OZZ, i.e. horizontal, vertical and diagonal. Hence, only one pass of screening is needed, and the combination of the two passes is avoided, which makes SPV faster than OZZ; and **c)** a junction recovery procedure is applied wherever a junction is encountered during line following (Wenyin and Dori, 1999). The aim of the Sparse Pixel Vectorization (SPV) is to improve the efficiency and accuracy via the sparseness of pixel examination. The result is a sparse skeleton, and polygonalization is needed to fit the line. With this method break of lines can be partially avoided. (Song et al, 2002b).

As we have already mentioned in section 1, we did not reach any work, which implements the ANN method for map vectorization. But we can cite some interesting papers that use methods based on artificial intelligence. Song & Civco (2004) use support vector machines for road extraction from aerial imagery. Zheng et al (2005) use Hidden Markov Model for detecting parallel lines in images consisting of text and rule lines. Their approach works, even if the documents are degraded. Dori & Velkovitch (1998) applied ANN method for segmentation and recognizing texts located in engineering drawings. In (Dias et al., 1995), a 8x8 ANN is used. The preferred ANN is trained by objects, which has corners in center of 4x4 pixel of the 8x8 window and which has no corners by randomly selected objects. This algorithm, only finds corners, and does not makes a line tracking. In (Sanchiz et al, 1996), Freeman chain code is used by normalizing it in 0-1 range. The presented algorithm evaluates every boundary pixel by its 12 or specified size of neighbour, boundary pixels. It uses neighbour pixels chain code as the input of ANN, and decides if pixel is a corner by the output of ANN. Fort he post processing, it merges detected corner points by specified methods. In (Tsai, 1997), a novel boundary-based corner detection approach using artificial neural networks (ANNs) is presented. Two neural networks are proposed: one for detecting corner points with high curvature, and the other for detecting tangent points and inflection points that generally have low curvature.

3. ARTIFICAL NEURAL NETWORKS

Artificial neural networks (ANNs) supply a general practical method for learning real-valued, vector-valued and discrete-valued functions from examples. ANNs has been inspired from biological information processing systems (Mitchell, 1997). They are mostly developed to do a nonlinear mapping from a set of inputs to a set of outputs. ANNs are designed to attain a biological system type performance that is based on a dense interconnection of simple processing elements similar to biological neurons. ANNs, which are information driven rather than data driven, are non-programmed adaptive information processing systems that can autonomously develop operational capabilities in response to an information environment, and are ideal in cases where the necessary mapping algorithm is not known and tolerance to faulty input information is required.
ANNs contain electronic processing elements (PEs) connected in a particular fashion. The behavior of the trained ANN depends on the weights, which are also referred to as strengths of the connections between the PEs. ANNs provide certain benefits over conventional electronic processing techniques that are the generalization capability, parallelism, distributed memory, redundancy and learning. ANN learning is suitable to problems, in which the training data corresponds to noisy, complex sensor data, such as inputs from cameras and microphones. It can also be applied to problems for which more symbolic representations are often used (Mitchell, 1997).

4. ALGORITHM

Our algorithm works in three steps. First, the image file is opened, converted to binary image and prepared for thinning. Thereafter the image is converted to an intermediate image, in which all lines are represented in one-pixel width. In the second step, the lines, whose width is one pixel, are tracked by ANN, and then the critical points for each object are determined. These points are saved in a point table. In the third step the objects that can be recognized as noisy are eliminated, the lines are simplified and the output file in vector format is created.

4.1 Thinning

Our thinning step is based on the conventional way. The most significant advantage of our algorithm is that it delivers the skeleton in one step, instead of working repeatedly. Doing so, the speed is increased.

In our algorithm, at first, 3 by 3 median filter is applied on the image, because the noisy pixels can cause that the pseudo lines are recognized. After applying the median filter, the image is scanned in left-right direction, row by row, until the first black pixel is found. Thereafter, beginning from that pixel, the white pixels in four main directions are sought. The number of black pixels in each direction is then saved. Thinning is performed in the direction, in which the number of black pixels is minimum (Fig.1). In this direction all black pixels are marked for deletion except the one being in the middle (Fig.2). In the next steps, the marked pixels are not processed, so the working speed is increased.

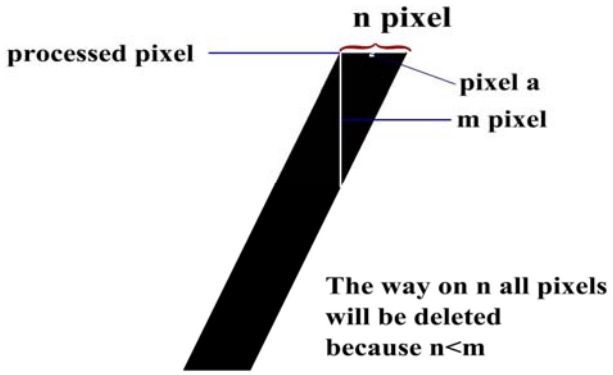


Figure1. Thinning step

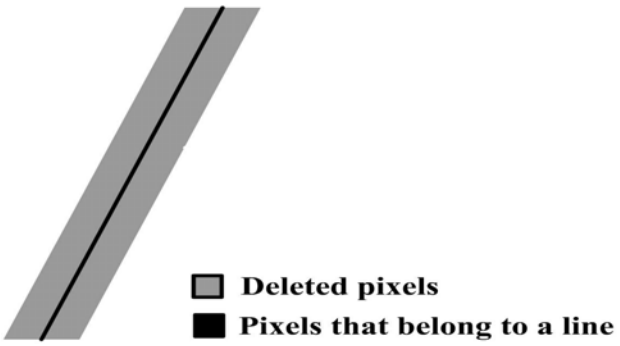


Figure2. Achieved line after the thinning

4.2 Line tracking with ANN and vectorization

Vectorization and line tracking is performed after thinning. In this step, processed data that contains one pixel wide skeleton is used. The goal is to find the critical points of the objects, and to represent these objects in vector format.

In this study, we use a supervised back-propagation to construct the ANN model. The proposed back-propagation neural network comprises an input layer, a hidden layer and one output

layer. Each layer is fully connected to the succeeding layer. The outputs of nodes in one layer are transmitted to nodes in another layer through links (Fig.3). The link between nodes indicates flow of information during recall. The ANN model uses "supervised learning" algorithm in learning phase. During learning, information is also propagated back through the network and used to update the connection weights between nodes. It has 25 inputs and 12 outputs (Fig.4). The neuron number of the hidden layer is 25. Sigmoid function is used as threshold function.
The ANN's learning table is given in table 1.

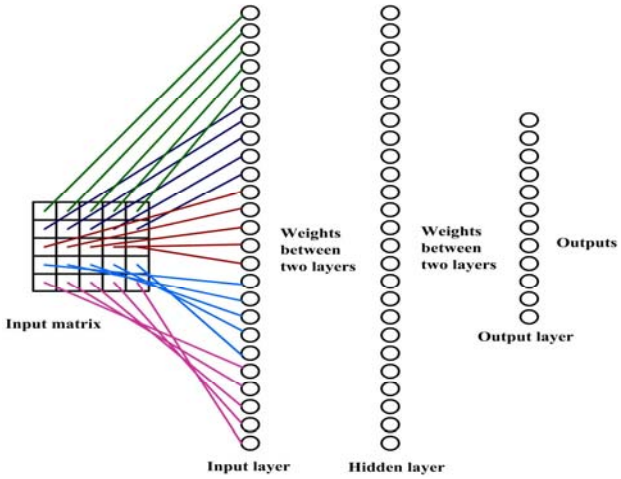


Figure 3. Used ANN model

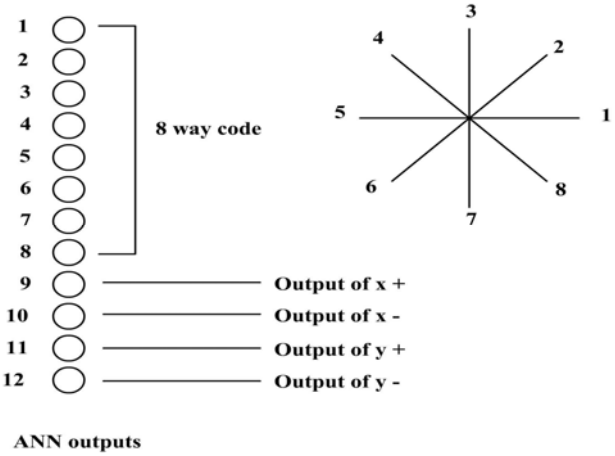


Figure 4. ANN model's output layer

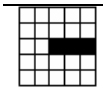
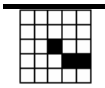
Input Values	Output 1	Output 2	Output 3	Output 4	Output 5	Output 6	Output 7	Output 8	Output 9	Output 10	Output 11	Output 12
	1	0	0	0	0	0	0	0	1	0	0	0
	1	0	0	0	0	0	0	0	1	0	1	0
	1	0	0	0	0	0	0	0	1	0	0	1
	0	1	0	0	0	0	0	0	1	0	1	0
	0	0	1	0	0	0	0	0	0	0	1	0
	0	0	1	0	0	0	0	0	1	0	1	0
	0	0	1	0	0	0	0	0	0	1	1	0
	0	0	0	1	0	0	0	0	0	1	1	0
	0	0	0	0	1	0	0	0	0	1	0	0
	0	0	0	0	1	0	0	0	0	1	1	0
	0	0	0	0	1	0	0	0	0	1	0	1
	0	0	0	0	0	1	0	0	0	1	0	1
	0	0	0	0	0	0	1	0	0	0	0	1
	0	0	0	0	0	0	1	0	0	1	0	1
	0	0	0	0	0	0	1	0	1	0	0	1
	0	0	0	0	0	0	0	1	1	0	0	1

Table 1. Learning set of ANN's

The "line tracking with ANN" is performed in following steps (Fig.5): (Px: current x position on the image; Py: current y position on the image)

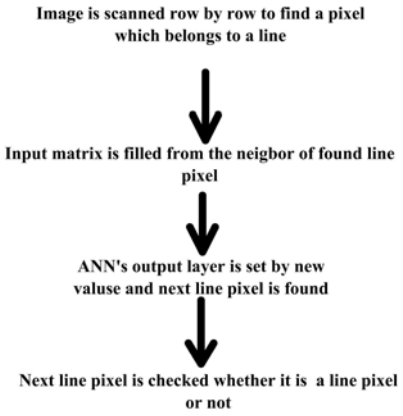


Figure.5 Flowchart for line tracking with ANN

1-Beginning with the first row, image is scanned row by row to find a pixel, which belongs to a line (simply a black pixel). When this pixel is found, Px and Py values are saved to show this pixel.

2-Input matrix is filled from the neighbor of the found line pixel on position (Px, Py). And current line pixel is deleted ("set to white") for being not to reprocess at step 1.
3-After input matrix is filled, ANN is worked and ANN's output layer is set by new values. In this step the algorithm is to answer two questions:

a.Is this pixel a critical point?

b.Where is the next pixel that belongs to a line?

Algorithm answers first question by evaluating the first 8 of 12 outputs of ANN. Seven of 8 outputs values are approximately "0", and one of 8 values is approximately "1". The output whose value is approximately "1" shows the tracking line's way in 8 way chain code. If this chain code is different from the chain code, which is previously found, this point is then marked as critical. It means that the direction of the line has been changed at this pixel.

Algorithm answers second question by evaluating the last 4 of the 12 outputs of ANN. If 9th output value is approximately "1", x position of the next line-pixel equals Px+1. If 10th output value is approximately "1", x position of the next line-pixel is Px-1. If 11th output value is approximately "1", y position of next line-pixel is then Py+1. If 12th output value is approximately "1", y position of next line-pixel then is Py-1. If all these values are "0", nothing is done.

For example: (O denotes outputs);

$$\begin{matrix} O_9 \approx 1 \\ O_{10} \approx 0 \\ O_{11} \approx 0 \\ O_{12} \approx 0 \end{matrix} \Rightarrow \begin{matrix} P_x = P_x + 1 \\ P_y = P_y \end{matrix} ; \qquad \begin{matrix} O_9 \approx 0 \\ O_{10} \approx 1 \\ O_{11} \approx 1 \\ O_{12} \approx 0 \end{matrix} \Rightarrow \begin{matrix} P_x = P_x - 1 \\ P_y = P_y + 1 \end{matrix}$$

4- After next line-pixel is found, it is checked whether it is a line-pixel or not. If this pixel is a line-pixel, the algorithm goes to step 2. If not, the algorithm searches for a line-pixel in "n x n" neighborhood, where n can be set parametrically (typically 5). If a line-pixel is found in this neighborhood, Px and Py are set and algorithm goes to step 2. Otherwise, algorithm goes to step 1.

4.3 Slope Based Critical Point Elimination:

The critical points obtained during the line tracking with ANN are saved in a table. In this table it is marked that which point belongs to which object. This table is transferred to next stage, where non-critical points are eliminated using a slope-based approach (Fig.6).

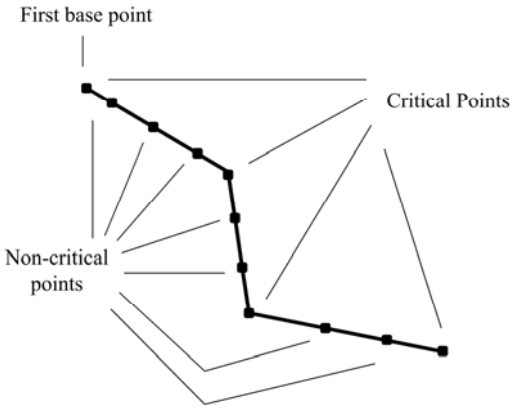


Figure6. Critical and Non-critical points

At this process, a point is selected being the base for slope. Initially, this point is the first point of the object. Then, the first base slope is determined with the second point in the table. In the next step, using the base point, the local slope for the third point is determined. Similarly, the slopes for the other points are calculated from the base point. If the difference between the base slope and the local slope of any point is bigger than a specified threshold value, this point is marked as critical, so the next base point will be this point. If not, this point is assumed as non-critical and is deleted. Then the steps above are repeated (Fig.7).

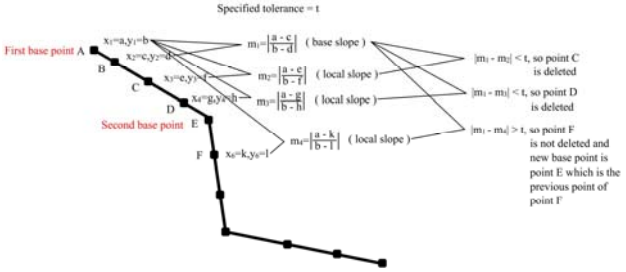


Figure7. Finding critical points with gradient difference

At this stage the remaining points are the exact corners, i.e. the real critical points. Here it is possible that the intersecting lines in the image do not intersect after vectorization. To minimize this effect, the first and the last pixels of every object are tested if there are other critical points in a specified range. If there are such points, the coordinates are averaged, so the nodes are clustered. Doing so, incomplete parts at some objects can be avoided. Finally, the critical points of the objects that remain in the point table are transferred to a file in DXF-format. Doing so a 2D drawing is created.

5. APPLICATION

5.1 Software Development

In order to implement and to test the vectorization algorithm explained above, a program was developed with the programming language C#. The software developed in this study is called VecNET.

5.2 Test of the Proposed Algorithm

In order to test of the algorithm proposed here, a digital vector map - a 1:1000 scale cadastral map showing parcels only – is chosen (Fig.8). The content of the vector file is plotted, and then scanned. So we have the ground truths (original coordinates in the vector file). The image size is 14396x11798 pixels. It contains 203 parcels. The sizes of the parcels vary from 23.77 to 14054.17 m^2 (on ground). Average area is about 1317 m^2. This image is then vectorized with our program VecNET and MDUS by Wenyin & Dori (1997). MDUS (Machine Drawing Understanding System) includes three algorithms: sparse pixel vectorization algorithm, stepwise recovery arc segmentation algorithm, and dashed line detection algorithm. In this study we only used the sparse pixel vectorization algorithm of MDUS. Additionally, the vectorized data is transformed to initial coordinate system by using affine transformation method. The affine transformation is defined by using the grid marks on the image. So we define a transformation between the pixel coordinate system and the initial coordinate system of the original vector file. The RMS error of the affine transformation is ±5.3 cm for VecNET and ±4.7 cm for MDUS on the ground.

Using the vectorized data, areas of the parcels were calculated after vectorization (VecNET and SPV), and were compared with the ground truths. The area changes for VecNET and SPV were calculated as percentage rates (Fig.9, Fig.10 and table 2).

Figure 8: Test image

To determine how the absolute positions of the parcels are changed after vectorization,minimum bounding rectangles (MBR) of the parcels are used. The center points of MBRs after vectorization (VecNET and MDUS) were compared with the ground truths. In table 3, Fig.11 and Fig.12 the displacements, in x and y direction, of the center points of MBRs are given. Fig.11 shows the histogram of changes (ds) after vectorization (VecNET). Fig.12 shows the histogram of changes (ds) after vectorization (MDUS).

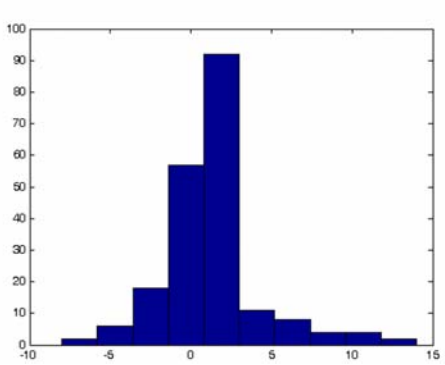


Figure 9: Changes in area after vectorization with VecNET percent

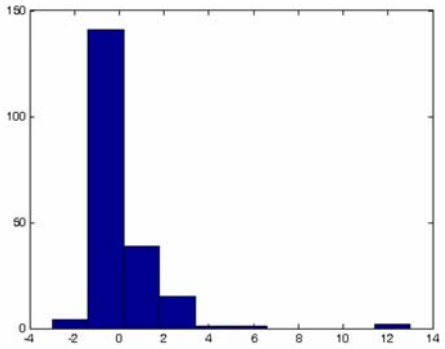


Figure 10: Changes in area after vectorization with MDUS in percent

	Object No	Max. Value	Min. Value	Mean Value	Standard Deviation
VecNET	203	14	-8	1.31	3.12
SPV	203	13	-3	-0.27	1.64

Table 2: The maximum, minimum, mean and standard deviation values of the parcel areas after Vectorization (VecNET and MDUS)

$$ds = \sqrt{dx^2 + dy^2}$$

$$dx = x_v - x_g ; dy = y_v - y_g$$

dx: displacement in x direction
dy: displacements in y direction
ds: displacement of the center points of MBRs
xv, yv: Coordinates after vectorization
xg, yg: Initial coordinates (ground truth)

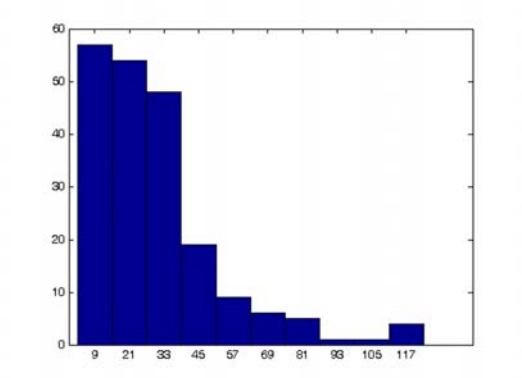


Figure 11: Histogram of displacement for the center points of MBRs of the vectorized (VecNET) parcels with the ground truths

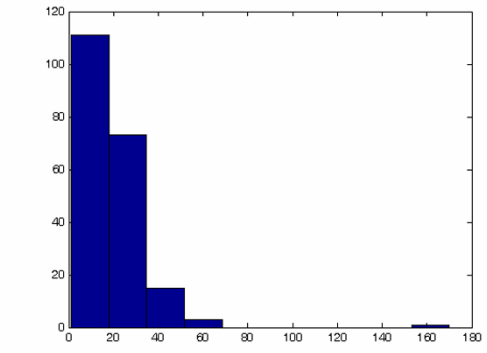


Figure 12: Histogram of displacement for the center points of MBRs of the vectorized parcels with respect to the ground truths (MDUS)

	Object No	Max. Value (cm)	Min. Value (cm)	Mean Value (cm)	Standard Deviation (cm)
VecNET	203	123.0	3.0	30.1	22.5
SPV	203	170.0	1.00	19.2	15.6

Table 3: The maximum, minimum, mean and standard deviation values of the displacements of MBRs of parcels after Vectorization (VecNET and MDUS)

The mean of the displacements of MBRs is 19.25 cm with a standard deviation of 15.60 cm for MDUS. The mean of the displacements of MBRs is 30.1 cm with a standard deviation of 22.49 cm for VecNET. It is slightly greater than the common drawing error (0.2mm on the map, 20 cm on the ground), but no significant displacements of the parcels are observed as seen in Fig.13.

The area changes of the parcels show both increasing and decreasing trends. The mean of the changes is 1.31% with a standard deviation of 3.12%. If we think of a parcel with a size of 1000 m^2, these values are 14 m^2 and 30.6 m^2, respectively.

The area changes of the 185 parcels vary –5% and 5%. For other 18 parcels the area changes and sizes are shown in table 4. It can be seen from the table that these errors (percentages

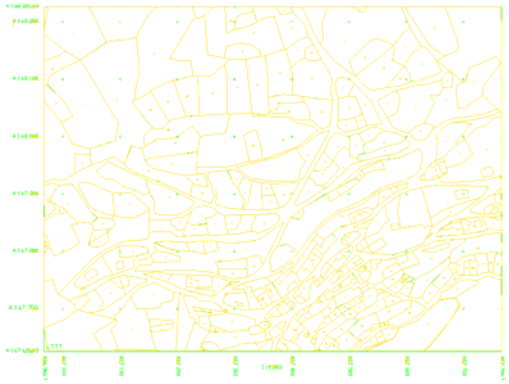


Figure 13: Initial parcels (green line) and vectorized parcels (brown lines)

greater than %5) occur in parcels with different sizes, i.e. these errors do not depend on the parcel size. Additionally, the positional distribution of them is diverse (Fig.14).

Parcel Area (m^2 on ground)	**The area changes (as percentages)**
95.764	12
105.173	6
111.027	9
120.983	14
127.012	9
131.911	8
155.592	6
295.454	8
348.915	6
463.86	7
476.803	6
678.163	10
743.587	11
806.758	7
808.474	-8
1205.33	10
1896.999	10

Table 4: The area changes and sizes for other 18 parcels

Figure 14: The distribution of the parcels, which have area changes greater than %5.

6. CONCLUSION

In recent two decades a great number of vectorization algorithms have been published, which we summarized in the first and second section. Among these methods, an approach with ANN is hardly to be found. Our vectorization algorithm is based on ANN, which is a powerful algorithm to solve problems like vectorization. We developed our own software, which uses our algorithm and tested the software with a test image. The accuracy of vectorization is a key issue for mapping purposes. The accuracy should be considered in two ways: size and location. It is important to know how big the distortions in size and in location after vectorization are. Therefore we tested our algorithm with an image, which is actually gained by using a vector drawing. We compared the values obtained from vectorized drawing with the ground truths. The changes in size and location after vectorization are found to be acceptable for mapping purposes, which confirms the applicability of our algorithm.

It is obvious that the MDUS program delivers better results than our program VecNET, but the results of the VecNET seem to be acceptable. Additionally we also observed that the VecNET produces acceptable results in the crossings of the parcel boundaries.

ACKNOWLEDGEMENT

The authors wish to acknowledge for the cooperation and the financial assistance given by the Scientific Research Found (BAP) of Selcuk University.

REFERENCES

Bai, Y. B., Xu, X. W., 2001. Object Boundary Encoding- A New Vectorisation Algorithm For Engineering Drawings, Computers In Industry 46 (2001), 65-74

Dias P.G.T., Kassim A.A. and Srinivasan V., 1995., Neural Network Classifier for Detecting Corners in 2D Images, Systems, Man and Cybernetics Intelligent for the 21st Century,vol.1,pp.661-666

Dori, D., Wenyin, L., 1999. Sparse Pixel Vectorization: An Algorithm and Its Performance Evaluation, IEEE Transactions on Pattern Analysis and Machine Intelligence 21(3), 202-215

Dori, D.,Velkovitch Y., 1998. Segmentation and Recognition of Dimensioning Text from Engineering Drawings, Computer Vision And Image Understanding Vol. 69, No. 2,196–201

Fernandez, X., Riveiro, F., Lopez., O.M., 1998. A Vectorizer For Color Topographic Maps, Proceeding of the IASTED International Conference Signal and Image Processing, Las Vegas, Nevada-USA

Lam, L., Lee, S.W., Suen, C.Y, 1992. Thinning Methodologies-A Comprehensive Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 14(9), 869-885

Mitchell, T. M., 1997. Machine Learning, McGraw-Hill Companies, Singapore, 414pp

Olson, C.F., 1999. Constrained Hough Transforms for Curve Detection, Computer Vision and Image Understanding 73(3), 329-345

Sanchiz J.M., Inesta J.M. and Pla F., 1996, A Neural Network-Based Algorithm to Detect Dominant Points From the Chain-Code of a Contour, Pattern Recognition Proceeding of ICPR, vol. 4, pp. 325-329

Song MJ., Civco D., 2004, Road Extraction Using SVM and Image Segmentation, Photogrammetric Engineering and Remote Sensing 70 (12): 1365-1371

Song, J., Cai, M., Lyu, M. R., Cai, S., 2002a. Graphics Recognition From Binary Images: One Step or Two Step, 16 th International Conference on Pattern Recognition (ICPR'02) 3, Quebec City, QC, Canada

Song, J., Su, F., Chen, J., Cai, S., 2000. A Knowledge-Aided Line Network Oriented Vectorisation Method For Engineering Drawings, Pattern Analysis & Applications (2000)3, 142-152

Song, J., Su, F., Li, H., Cai, S., 2002b. Raster to Vector Conversion Of Construction Engineering Drawings, Automation in Construction 11(2002), 597-605

Song, J., Su, F., Tai, C.L., Cai, S., 2002c. An Object-Oriented Progressive-Simplification-Based Vectorization System For Engineering Drawings: Model, Algorithm, and Performance, IEEE Transactions on Pattern Analysis and Machine Intelligence 24(8), 1048-1060

Tombre, K., Tabbone., S., 2000. Vectorization in Graphics Recognition: To Thin or not to Thin, International Conference on Pattern Recognition (ICPR'00)- 2, Barcelona, Spain

Tsai D.M, 1997, Boundary-Based Corner Detection Using Neural Networks, Pattern Recognition,vol.30no.1,pp.85-97

Wenyin, L., Dori, D., 1999. From Raster to Vectors: Extracting Visual Information From Line Drawings, Pattern Analysis & Applications (1999) 2, 10-21

Wenyin, L., Dori, D. (1997) A protocol for performance evaluation of line detection algorithms, Machine Vision and Applications (1997) 9, 240–250.

Xu, X. W., Bai, Y. B., 2000. Computerising Scanned Engineering Documents, Computers In Industry 42(2000), 59-71

Zheng Y., Li H., Doermann D., 2005, A Parallel-Line Detection Algorithm Based on HMM Decoding, IEEE Transactions On Pattern Analysis and Machine Intelligence, Vol. 27, No. 5, 777-792