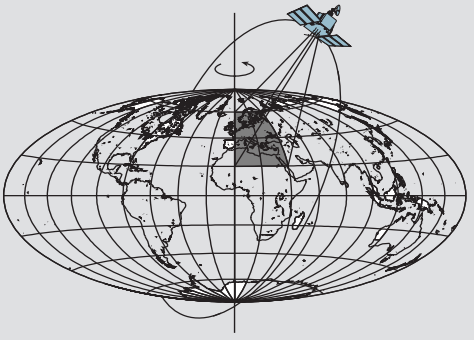
Object Recognition from AIMS Data Using Neural Networks

by

Ron Li
Fei Ma
Zhuowen Tu



Report No. 462

Geodetic and GeoInformation Science
Department of Civil and Environmental Engineering and Geodetic Science
The Ohio State University
Columbus, Ohio 43210-1275

December 1998

Object Recognition from AIMS Data Using Neural Networks

Project Report
(November 1997 – December 1998)

Submitted to
The OSU Center For Mapping

By

Principal Investigator: Dr. Ron Li
Researchers: Dr. F. Ma and Z.W. Tu

Department of Civil and Environmental Engineering
and Geodetic Science
The Ohio State University

December 1998

Object Recognition from AIMS Data Using Neural Networks

1. Introduction

AIMS (Airborne Integrated Mapping System), a fully digital and real-time airborne mapping system, has been developed at the OSU Center for Mapping (CFM). A fundamental task of AIMS is to understand geo-referenced mapping images for automatic construction of 3-D spatial databases. Based on successful research on automatic mobile mapping data processing in the previous project year (Li et al. 1997), efforts have been made this year to automate procedures for image feature extraction and for recognizing and locating objects, considering unique geometric constraints provided by GPS and INS data. Specific tasks completed include:

- Studying relationships between image features, photogrammetric models, and knowledge of objects;
- Integrating edges, region boundaries and DEM slope-aspect information in order to group and classify extracted features;
- Developing a knowledge based feature extraction system by combining several different approaches;
- Developing a Multilayer Hopfield Network; and
- Applying the neural network for truck recognition and location, as well as velocity estimation.

Two significant breakthroughs in this project year are:

- Development of algorithms that build vector lines from AIMS images by fully utilizing image feature context, geomorphologic information, camera georeferencing geometry, terrain models, and knowledge about the objects to be extracted. Experiments in extraction of roads, houses and trucks showed very promising results; and
- Development of an improved two-layer Hopfield neural network that accepts extracted vector lines from AIMS imagery and recognizes trucks on freeways. Location and velocity of the recognized trucks are established in a subsequent process.

The overall success in this project and a related project supported by Sea-Grant/NOAA for geometric modeling of one-meter resolution satellite imagery (Li 1998, Li et al. 1998a) builds a basis for further research on object recognition to support extended tasks such as automatic derivation of large-scale spatial information from one-meter resolution satellite imagery.

2. Automatic Feature Extraction from AIMS Imagery

2.1 AIMS Imagery

A set of AIMS images were taken in Madison County, Ohio with a flying height of about 1.3 km. A camera calibration was performed so that lens distortion and interior orientation parameters are available. Ground Control Points (GCPs) in the area of interest were surveyed using GPS. Figure 2.1 shows a typical AIMS image.

Figure 2.1 An example of AIMS imagery

Roads are one of the major features in the images. In addition, farmland varies in its image texture. In general, man-made objects such as roads, buildings and vehicles can be seen at the original resolution. Shadows provide additional information for most man-made objects.

We began our study by extracting edges of features based on change of pixel intensities. From the extracted edges, seeds are selected and linked together giving consideration to geometric constraints. Next, roads of specific width are constructed based on the model of anti-parallel lines. Finally, objects with height dimension such as houses and tracks are extracted and classified through shadow analysis.

2.2 Feature Extraction from Aerial Imagery

Feature extraction is an area of active research in computer vision. In the past, feature extraction has been divided into several components and stages. Examples include edge detection, contour derivation, and shape modeling. The divisions were not only based on error modeling alone, but also considered other constraints. Later models, such as the deformable contour model (Kass et al. 1987), treated these problems in a general unified manner. However, no criteria exist to test models and validate parameters.

Digital Mapping Laboratory of Carnegie Mellon University (CMU) attempted to combine all available image, scene, and domain knowledge in their feature extraction methods. For example, they used the vanishing point technique for building detection, and they incorporated automatic feature extraction and stereo matching into a semi-automatic system (Mckeown et al. 1996). In another study (Gruen and Li 1996), a semi-automatic method for linear-feature extraction from digital images was presented. The procedure combined human identification of features with high precision line extraction performed by a computer for GIS data capture.

In an early study, McKeown (1990) specified four topics of primary importance in image understanding: building detection, road extraction, stereo analysis through matching, and knowledge-based scene analysis. Although described as independent problem domains, it is clear that results from each topic must be integrated in order to automate the cartographic feature extraction process. Generally, local edge strength and direction are computed first followed by the selection of local maximums. Finally, edges are linked in order to form single-pixel-wide lines. In our method, we improve edge detection by implementing a strategy that

uses multiple resolution line extraction (from coarse to fine) and geometric constraints from GPS and INS in a multi-level approach.

2.3 Development of an Integrated Algorithm for AIMS Imagery Feature Extraction

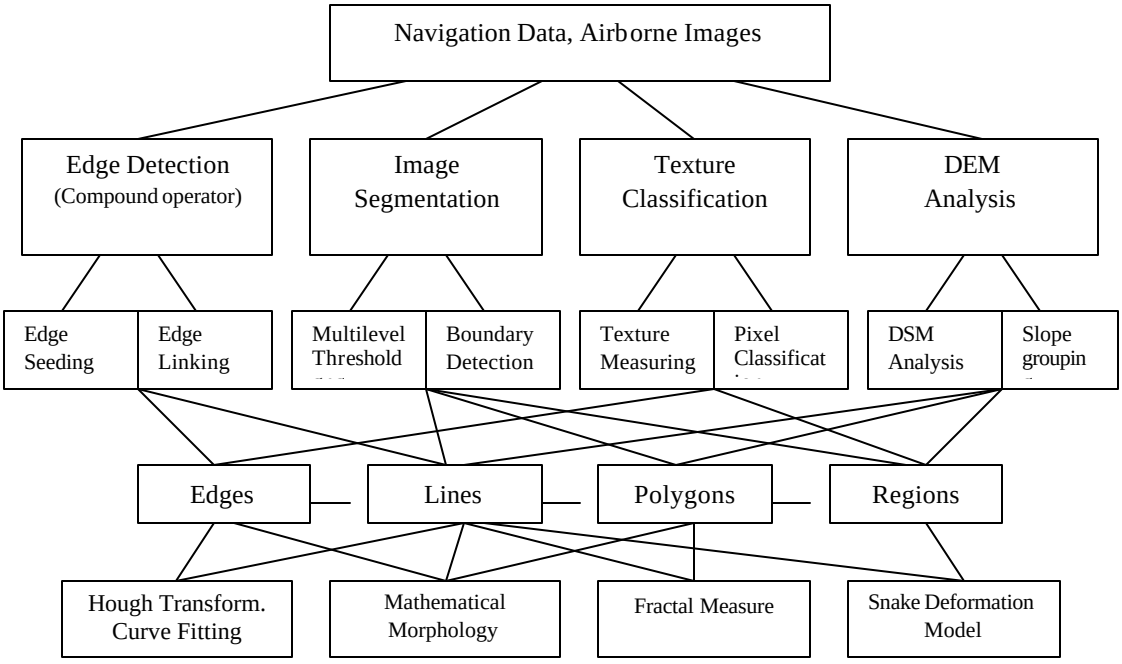


Figure 2.2 Program flowchart

Edges carry most of the information in an image and are relatively unaffected by changes in image contrast and radiometry. Therefore, our algorithm begins with multi-resolution edge detection in an AIMS image. Figure 2.2 depicts the information flow and methods used in data processing. Each major component in the figure is discussed in the following sections. The system is implemented using MS Visual C++ on a Pentium II machine.

2.3.1 A Compound Operator for Complex Edge Detection

Marr (1982) first applied different sizes of edge operators on an image to obtain a description of signal changes at different scales. This is called multi-scale or multi-resolution edge detection. Marr also suggested that zero-crossings over several scales are physically significant. Additionally, Marr and Hildrith (1980) have proposed a Laplacian of Gaussian (LoG) edge detection operator in which Gaussian-shaped smoothing is performed prior to the application of

a Laplacian operator. Although the method is well-behaved (Witkin 1983, Yuille and Poggio 1983, Lu and Jain 1989, 1992), high frequency noise and over-circular-contour spurious noise cause poor results. Another well-known gradient-operator is the Derivative of Gaussian (Drog). In this operator Gaussian-shaped smoothing is followed by differentiation. The so-called Canny's operator (Canny 1986) is based on an analytical model for step-edge and white Gaussian noise. In describing his operator, Canny placed the following criteria on edge detection: good detection, good localization and single response. His operator yields high performance in simple scene analysis, but details of implementation vary in the establishment of gradient direction, suppression of edge noise in neighborhood edges and directional gradient.

In our study, we combine LoG and Drog operators into a compound edge detector to take advantages of information from second and first order derivatives. Edges in the image are selected by both zero-crossing of LoG and those above the adaptive threshold of Drog. We use multi-resolution edge detection for complex AIMS images. With small-scale Gausian-filtered (small sigma) images, our operator may be, to some extent, sensitive to noise, but it provides us with fine changes in intensity details. Further, the operator provides coarse intensity change information with large-scale Gausian-filtered images. The adaptive threshold of the Drog operation is followed to avoid over-circular-contour spurious noise of LoG. In other words, zero-crossing of LoG is the basis for comparison with the Drog operator result when identifying edges. Thus, the interaction between these two operators acts to eliminate fine scale noise caused by separate events at different scales. Mathematically, we have

$$G_{s_1} = F(x,y) * H_{s_1}(x,y) \tag{2.1}$$

$$G_{s_2} = F(x,y) * H_{s_2}(x,y) \tag{2.2}$$

where F(x,y) represents the imagery function, $*$ denotes convolution operation, $H(x,y)$ represents Laplacian of Gaussian function, s_1 and s_2 are scale parameters ($s_1 < s_2$), and G_s is the zero-crossing of LoG operation.

Additionally, Drog operations with the constraint of G_s are

$$D_{s_1} = F(x,y) * T_{s_1}(x,y); G_{s_1} \tag{2.3}$$

$$D_{s_2} = F(x,y) * T_{s_1}(x,y); G_{s_2} \tag{2.4}$$

where $T(x,y)$ represents the derivative of Gaussian.

Thus, the edge detection works at two different scales. One for extracting the raw shape of an object, called initial resolution and the other for extracting the exact shape of the object and for distinguishing it from other similar looking objects. This is implemented by defining an initial scale s_1 based on the Canny's optimal edge detector criteria with a corresponding window size. A refined scale s_2 is chosen in order to increase edge detection details. Edges detected at the initial resolution are intensified by edges detected at the refined resolution, while the later avoids spurious noise by intersecting the results from the initial scale.

2.3.2 Region and Morphology Based Image Segmentation through a Multi-level Threshold

The method introduced above is capable of detect edges representing objects such as roads and other linear features. On the other hand, objects of reasonable uniform brightness against a background of variable brightness, for example, houses and trucks can be detected by image segmentation that is usually the division of the image into regions having similar attributes. The most basic attribute for segmentation is image amplitude (luminance for a monochrome image, and color components for a color image). Image edges and textures are also valid attributes for segmentation. In our study, we use both image amplitude and image texture to delineate region boundaries.

The basic task of luminance threshold selection is to determine a threshold value that falls at the minimum point of the histogram, i.e., between its bimodal peaks (Pratt 1991). It is proven in our study that, due to the complexity of aerial imagery, the segmentation could not be finished in one step. Instead, a multi-level threshold method was used to determine a different threshold at each step. The threshold of a previous step was adopted if and only if the luminance histogram formed bimodal peaks. A parabola is used at both of the histogram peaks to model a curve segment, and a threshold is selected at the intersection of the two parabolas. The steps of the multi-level threshold procedure is as follows:

1) Set i=0 to initialize arrays $A_0[64]$ and $A_1[64]$, and set the minimum luminance to $A_0[0] = 0$, and the maximum luminance to $A_1[0] = 255$.

2) Set a series of array $A[64/2^j]$ and compute their values as the number of pixels with the gray value between $A_0[i]+(A_1[i]-A_0[i])/64*2^j$ and $A_0[i]+(A_1[i]-A_0[i])/64*2^{j+1}$, j=0,1,… $64/2^j$.
3) Calculate the histogram of luminance distribution within the image area corresponding to i (if i equals 0, the entire image is represented; otherwise i represents a portion of the image).
4) If the histogram has no bimodal peaks, discard the mark on the area and set all pixel gray values to $64*2^j$ if it is marked as "+", or to 255- $64*2^j$ if it is marked as "-". Then go to step 6. If the histogram has bimodal peaks, place a parabola at each peak, and use the intersection of the parabolas to segment the image area into two parts and determine the threshold. Set a "+" mark on the segmented image area having a gray value larger than the threshold, and set a "-" mark on the area with a gray value lower than the threshold.
5) i++; if the area is marked "+", then $A_0[i]=A_0[i-1]+(A_1[i-1]-A_0[i-1])/2$ and $A_0[i]=A_0[i-1]$; if the area is marked "-", then $A_1[i]=A_0[i-1]+(A_1[i-1]-A_0[i-1])/2$ and $A_0[i]=A_0[i-1]$.
6) Within one of marked image areas, repeat steps 3 and 4.
7) If there are still marked areas, repeat steps 2 through 6 with corresponding level i; otherwise go to 8.
8) End.

Secondly, we delineate region boundaries through texture features. Although texture is a valuable feature for image segmentation, putting this proposition to practice, however, has been hindered by a lack of reliable and computationally efficient means to measure texture (Haralick and Shapiro 1992). In fact, texture on one hand appears as structure and at the same time has a statistical property. Effective textural classifiers should address both of these properties. In our study, we use a morphological concept, granulometric measure, to extract texture feature on the detected edges.

Since edges belong to a line type, the structure elements for texture classification consist of line segments. In a 3 x 3 window, four directional structure element are formed as:

$$E_1=\begin{matrix}0&1&0\\0&1&0,\\0&1&0\end{matrix}\qquad E_2=\begin{matrix}0&0&1\\0&1&0,\\1&0&0\end{matrix}\qquad E_3=\begin{matrix}0&0&0\\1&1&1,\\0&0&0\end{matrix}\qquad E_4=\begin{matrix}1&0&0\\0&1&0.\\0&0&1\end{matrix}$$

Suppose we have $nE_i = E_i \oplus E_i \oplus \Lambda\, E_i$ (n *times dilations*), where $\oplus$ represents the dilation operation. The spectrum of the edge relative to the structure element E_i can be acquired as:

$$SPEM = \frac{A(X \Theta nE_i)}{A(X)} \tag{2.5}$$

Where $A(\bullet)$ represents area calculation and Θ represents erosion operation. The size of the structure element is changed in order to iterate each transformation. Spectrum relative to size n describes statistic distribution of edge relative to structure element E_i. Texture features belonging to different areas are distinguished by a large difference in relative statistical distribution of structure elements. In order to classify each pixel into different texture areas, the area of each unique texture feature is estimated, and then centered at each pixel around the same area, the spectrum are calculated again. Those with high spectrum pixels are classified into a region with the same structure as selected structure elements.

In order to reduce the complexity of feature extraction, image segmentation is used to assist in edge analysis. The salient segmented regions are detected, and their boundaries are used as a condition for edge selection and linking as explained in the following steps.

2.3.3 Robust Edge Selection and Linking

Morphological edge selection and edge thinning

So far, the detected edges are still discrete edge points. They are to be aggregated and linked to form explicit edges/lines. First, the detected edges and boundaries may be of varying width (one to several pixels). A morphological thinning transformation thins detected edges to one pixel width (Serra 1982). The transformation is iterated several times, corresponding to the length of the selected segment. Then, segments with length less than a selected threshold are deleted, while segments with length larger than the threshold are preserved by conditional dilation. The procedure for morphological edge selection can be mathematically expressed as:

$$X' = X \circ \{L_i\}, i = 0,1,\Lambda\, 7 \tag{2.6}$$

$$X'' = (X' \circ \{E_i\}) \oplus \{E_i\}; X,\ i = 0,1,\Lambda\, 7 \tag{2.7}$$

where O represents the morphological thinning transformation, with structure elements

$$\begin{cases} L_i = \begin{matrix} \cdot & 1 & 0 \\ 0 & O & 1 \\ 0 & 0 & \cdot \end{matrix} & i = 1,3,5,7,\ i\ plus\ 2,\ clockwisly\ rotates\ p/2 \\ L_i = \begin{matrix} 0 & 1 & 0 \\ 0 & O & 1 \\ 0 & 0 & 0 \end{matrix} & i = 2,4,6,8,\ i\ plus\ 2,\ clockwisly\ rotates\ p/2 \end{cases},$$

$$E_i = \begin{matrix} 2 & 2 & 2 \\ 0 & O & 0 \\ 0 & 0 & 0 \end{matrix} \quad here\ 2\ represnets\ either\ 0\ or\ 1, i = 0,1,\Lambda,7,\ i\ plus\ 1, clockwisly\ rotates\ p/4$$

Conditional edge aggregation

Under the refined resolution condition, the edge segments selected at the initial resolution are expanded according to a so-called end-point conditional dilation. There are three steps in each transformation iteration. First, the end points are searched through a hit-or-miss transformation; next, the selected end points are expanded in a nearby image window (3 x 3); and finally, the expanded pixels, with intersection of edges at the refined resolution, are combined with the input edges. The procedure for conditional edge aggregation can be mathematically expressed as:

$$X''' = \bigcup_{i=0}^{7} (X_1'' \otimes E_i) \oplus \{\tilde{E}_i\}; X_2'' \qquad (2.8)$$

where X_1' and X_2' represents selected edges at the initial and the refined resolutions respectively, $\otimes$ represents the morphological hit-or-miss transformation, and the structure elements $\tilde{E}_i$ correspond to E_i as

$$\tilde{E}_i = \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{matrix} \quad i = 0,1,\Lambda,7,\ i\ plus\ 1, clockwisly\ rotates\ p/4.$$

Local edge linking

Local edges are selected within a small window (e.g. 3 x 3 or 5 x 5) in the image. If there is only a pair of end points in the window, connect them directly. If there are more than two points, the pair of end points with the same gradient direction and minimum interval is linked. The end points are acquired through morphological hit-or-miss transformation with the structure element E_i, as show above. It is desired to avoid the time consuming task of point-to-point comparison for end-point match determination. Our method performs a simultaneous dilation of all deleted end points in the direction of the contour (perpendicular to the gradient). Dilation is performed until each end point meets another end point moving in the same direction. At this time dilation ceases.

Curve fitting for edge linking

If a priori information about the expected shapes of objects is available (e.g. an expected rectangle for a truck or house), fitting may be carried out directly based on the desired shape. A simple piecewise linear curve-fitting procedure is the "iterative end points fit" (Duda and Hart 1973). In the first stage of the algorithm, end points are connected through a straight line. The point of greatest departure from the straight line is examined. If the separation of this point is too large, the point becomes an anchor point for two straight line-segments. The procedure then repeats till the data points are well fitted by line segments.

However, the seemingly simple algorithm is complicated by the necessity to determine anchor points. Any new anchor point may generate a need for two additional anchor points, each within two separated segments. We use a "first-in-last-out" stack algorithm to handle anchor point selection. The algorithm is described as follows.

1) Define a minimum departure which determines an anchor point, say l,
2) Set a first-in-last-out stack (STK1), then sequentially push all the points from the first end point (marked as A) to the second end point (marked as B), set two more first-in-last-out stacks (STK2 and STK3), and mark STK2 as "-",
3) Push point B into STK3, and then push point A into STK3,
4) Pop two end points A and B from STK3. If STK3 is NULL, go to step h,
5) Pop end point (marked as C) from STK1. If C==B go to step g,

6) Calculate the distance d from point C to the line AB. If d > l, push point B into STK3, and then push point C into STK3, mark STK2 as "+"; iterative pop point from STK2, and push the point into STK1 until STK2 is NULL, go to step c. Else, push point C into STK2, iterative continue with step e ,
7) If STK2 marked "-", record point A and point C, iterative pop point from STK2 until STK2 is NULL, go to step d; else push point B into STK1, let B==C, iterative pop point from STK2, and push the point into STK1 until STK2 is NULL, go to step e, and
8) Discard STK1, STK2 and STK3, then End.

Hough transformation for edge linking

Hough transformation is mainly used here for linking edge pixels of straight lines. It involves transformation of a line from Cartesian coordinate space to polar coordinate space, in which, the transformed line is simply a point at coordinates (r,q). A family of lines passing through a common point maps into the connected set of points. The main advantage of Hough transform is that it is relatively unaffected by gaps in curves and by noise. For the problem of straight-line detection, Hough technique organizes points into straight lines by considering all possible straight lines. The following is an algorithm of line detection with the Hough transformation.

1) Define the parameter space between approximate maximum and minimum values for c and m,
2) Form an accumulative array A(c,m) whose elements are initially zero,
3) For each point (x,y) in a gradient image, if the strength of the gradient exceeds a threshold, increment all points in the accumulative array along the corresponding line (i.e. A(c,m):= A(c,m) + 1, for m and c satisfying c = -mx+y),
4) The local maximum values in the accumulative array now correspond to collinear points in the image. The values themselves provide a measure of the number of points on the line.

In practice, there are two key points in this algorithm. One is in choosing the size of the array; the other is in determining a threshold to form or exclude corresponding lines. In our study, we base the size of the array on the image size. Our experiments show that in order to achieve good localized line detection, the size should be two to three times larger than the size of image. In order to exclude pseudo lines, a global maximum criterion (rather than local maximum) is selected to form a line. The longest line segment sharing two end points with detected edge pixels is recognized as a line in the image. The iterative line detection proceeds

with the deletion of recognized lines. It ends when the ratio of global maximum to medium values in the accumulative array is less than the ratio of the array size to the image size.

2.3.4 DEM Interpolation and Slope Grouping

Image features are a projection of ground objects. Inversely, it is hoped that information about the terrain such as Digital Elevation Model (DEM) should help detect object features in the image. A DEM can be acquired through photogrammetry, surveying, or contour map digitizing. In our study, a portion of a contour map was scanned and transformed into DEM data by a raster-vector conversion. The relatively small-scale map (1:24,000) generated a DEM that needs to be densified to match the 4k x 4k image. The result is a grid defined on the image array so that for each image pixel there is an elevation. The following procedures were used for DEM interpolation:

1) Contour maps are scanned and are vectorized to form DEM source data,
2) Transformations of the DEM source data form the map's geodetic coordinate system (say UTM) to photogrammetric object space coordinate system (say SPC),
3) Transformation of the object space coordinates to image space coordinates using collinear equations,
4) Transformation of the image space coordinates to digital image coordinates,
5) DEM interpolation to each pixel of image through bilinear interpolation method.

Slope magnitude and direction, which can be calculated by DEM data, are important terrain features in terrain analysis. They are calculated through DEM data as follows:

Slope magnitude:

$$s = \sqrt{\left(\partial Z / \partial X\right)^2 + \left(\partial Z / \partial Y\right)^2} \tag{2.9}$$

Direction angle:

$$a = a\tan\left(\frac{\partial Z / \partial Y}{\partial Z / \partial X}\right) \tag{2.10}$$

Grouping is performed at each grid point by its slope magnitude and direction angle. Slopes are divided into several levels depending on their magnitude. Each level corresponds to an area that shares the same terrain hypsography. This constitutes a condition for image segmentation and edge grouping. For example, freeways generally have a small slope (say 0-1 degree) for high-speed transportation. So freeways should fall into areas where slopes are less than a threshold value, and within these areas freeway edges are formed (assuming there is no error in DEM data). In addition, direction angles are grouped in to several levels from 0 to $\pi/2$.

2.3.5 Road Extraction and Road Network Construction

To extract roads, a set of seed edges are defined, which have the lowest level of slopes. Among the seed edges those pairs that have the anti-parallel character (the difference between direction angles is $\pi/2$, caused by gradients of the two opposite road edges) and a specific width (distance between the two edges) are selected as road edge candidates. Nearest end points of the edge candidates are connected with a constraint on direction angles. A preliminary road network is formed based on a principle that that the road topology should be as simple as possible (minimum number of intersections).

The preliminary network is improved by a re-segmentation procedure considering that the preliminary road network should divide the image into unconnected areas. These areas can be used as a condition for a further image segmentation. In analyzing the re-segmentation result, polygons that do not exhibit road characters are ignored. Edges near the preliminary road network which are longer than the specified threshold are selected. These edges constitute a geometric constraint used to refine the preliminary road network through a snake deformation model.

Snakes, or active contours, were originally used for semi-automatic feature extraction in computer vision (Kass et al. 1988). Starting from an initial estimate of a curve's shape and location (segments in the preliminary network in our case), the snake wriggles through the image in an iterative process (Tao et al. 1998). This evolution of the snake is driven by several elements of an energy function, which is composed of terms of various natures. These include internal terms, photogrammetric terms and external forces. Suppose $E_{snake}=E_{\text{int}}+E_{pho}+E_{ext}$. A

snake is described parametrically by v(s) = (x(s), y(s)), with s representing the arc length from the beginning of the curve. The energy function is then

$$E_{snake}=\int_0^1 E_{snake}(v(s))\,ds=\int_0^1 (E_{\text{int}}(v(s))+E_{pho}(v(s))+E_{ext}(v(s))\,ds \tag{2.11}$$

Internal energy forces are composed of a first order term and a second order term:

$$E_{\text{int}}=(a(s)\left|v'(s)\right|+b(s)\left|v''(s)\right|)/2 \tag{2.12}$$

Here, $a(s)$ and $b(s)$ are two dual weight factors. The internal forces place constraints such as smoothness of the curve. Generally, photogrammetric energy uses the gradient of the curve

$$E_{pho} = -g^{i}\left|\nabla(v(s_i)\right| \tag{2.13}$$

at point i that is determined photogrammetrically. g^{i} is a weight factor. The external attraction forces are introduced as

$$E_{ext}(v_i) = d_i((v(s_i)-l_i)^2 \tag{2.14}$$

where d_i is a weight factor and l_i the desired distance from the known photogrammetrically determined point i to the curve. This term requires that the final curve should be as close to the known points as possible.

Overall, minimizing the energy function makes the snake deform to a compromise between fitting the known points and preserving the characters such as smoothness.

2.3.6 Aboveground Objects Extraction and Classification

Shadow extraction

Shadow is an important feature in aerial imagery. In principle, using shadows, date, time, and orientation of the image, the sun's position can be determined. Inversely, artificial shadows of known objects can be generated for object recognition purposes. In photo interpretation, the

combination of an object surface (for example, a building roof) and its shadow make them distinguished from others. After image segmentation, statistic properties such as mean and variance are calculated in each segmented area. We are to find adjacent areas with very small variances. One area (shadow) has a very low intensity mean and the adjacent area (its corresponding object) has a very high intensity mean. In addition, the direction of the adjacency should be in the same as the sun.

Aboveground objects extraction

After shadow extraction, adjacent areas with high intensity means and small variances are called aboveground object areas. Within the areas, the image is once again segmented based on intensity. Objects found in the areas are considered to be aboveground objects. Edges of these objects are selected and linked to form their boundaries.

Distinguishing natural objects and man-made objects by shape classification

Aboveground objects and their shadows have specific shapes, particularly natural objects such as trees and man-made objects such as buildings and trucks. Natural objects often occur in an irregular and complex contour shapes. In contrast, man-made objects often have a regular contour shape, such as a box surface or parallel edges. There are numerous methods to distinguish between different shapes (Ballard and Brown 1982). In our study, a contour shape is measured by the complexity of direction change from pixel to pixel in a nearby 3 x 3 image window. The direction change can be represented through Freeman chain coding (Freeman 1974). Within a 3 x 3-image window, the direction is coded as:

4 3 2
5 0 1
6 7 8

Suppose a set of chain codes is $C_0C_1C_2C_3\Lambda C_n$. For any i, $0<i<n$, if $C_{i-1}==C_i==C_{i+1}$, then pixel direction at C_i has no change; otherwise there is a direction change at C_i. Moreover, if $\max\left(\sum_{k=i-m}^{i}\left|C_k-C_{k-1}\right|,\ \sum_{k=i}^{i+m}\left|C_{k+1}-C_k\right|\right)<m$, then we can say direction at C_i has no change within a step size of m; otherwise there is a direction change at C_i within a step size of m. Similar to what is found in the fractal dimension of a shape calculation, the more a contour changes

direction the greater complexity it has. Those shapes with high complexity are classified as natural objects while shapes with low complexity are classified as man-made objects.

2.3.7 Vectorization

The vectorization algorithm requires two steps. The first step is to describe the topology of the extracted objects. Intersections of extracted edges are searched by morphological transformation and are preserved as the beginning and ending points of segments in the second step. The second step uses Douglas-Peuker method to select critical points of each segment determined in the first step.

2.3.8 Line Grouping Through Perspective Geometry

In theory, a set of parallel lines in 3-D object space converges at a single point known as the vanishing point when projected into image space. Consider a unit sphere centered at the origin of the camera coordinates system, called a Gaussian sphere. A vanishing point can also be represented as a point on the Gaussian sphere, that is, a unit vector placed at the perspective center. So, given a vector Q in the object space, its vanishing point v on the Gaussian sphere can be computed as

$$v = \frac{M \times Q}{|M \times Q|} \tag{2.15}$$

where M is an image rotation matrix. In particular, suppose the solar azimuth is represented by a vector Q_1- Q_2.

$$q_1 = [0,0,1]^T \tag{2.16}$$

and

$$q_2 = [\sin q, \cos q, 1]^T . \tag{2.17}$$

The vanishing point of vertical lines in object space can be directly computed as

$$v_1 = \frac{M \times Q_1}{|M \times Q_1|} . \tag{2.18}$$

The vanishing point for the shadow lines cast on horizontal planes by vertical lines in object space is

$$v_2 = \frac{M \times Q_2}{|M \times Q_2|} . \tag{2.19}$$

Thus, extracted lines can also be grouped into either parallel lines (such as vertical lines and their shadows) which share a vanishing point, or non-parallel lines, which do not share a vanishing point.

2.3.9 Conversion of Polygons into known shapes

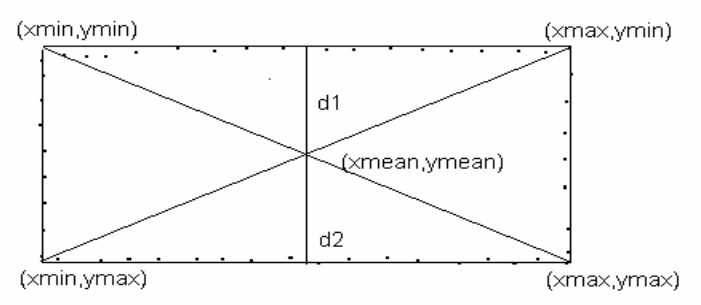


Figure 2.3 Polygon simplification

In some cases, an expected known shape can be used to simplify extracted features if we know that the features should have the known shape. For example, a quadrangle structure is typical of the top of most trucks. Thus, the extracted polygons can be simplified into quadrangle shapes. Suppose such a polygon is represented as a vector $n=(x(s),y(s))$. We need to determine four vertices $p_1(x_1,y_1)$, $p_2(x_2,y_2)$, $p_3(x_3,y_3)$, $p_4(x_4,y_4)$ by fitting the data. Let us denote maximum and minimum coordinate values as $x_{\max}=\max(x(s))$, $x_{\min}=\min(x(s))$, $y_{\max}=\max(y(s))$, $y_{\min}=\min(y(s))$. Then the center of the polygon is represented as point (x_{mean}, y_{mean}), $x_{mean}=(\max(x(s))+\min(x(s)))/2$, $y_{mean}=(\max(y(s))+\min(y(s)))/2$. We next define two diagonal lines linking four points of $(x_{\max}, y_{\max})$, $(x_{\min}, y_{\min})$, $(x_{\max}, y_{\min})$ and $(x_{\min}, y_{\max})$. The polygon points, $n=(x(s),y(s))$, are divided into four groups by the two diagonal lines, with each group between two diagonal lines. The quadrangle has two longer sides and thus, there must be two groups having more points than other two. Assume that the two lines represented by the two groups with more points are l_1 and l_2. These two lines can be determined by line fitting using the two groups of points. We can also calculate the distances between the center point, (x_{mean}, y_{mean}), to the two lines as d_1 and d_2. If the area of the original polygon is S, the length of l_1 and l_2 is approximately equal to $S/2d_1$ and $S/2d_2$. With the calculated length of l_1 and l_2 and the fitted lines, it is then trivial to determine the vertices of $p_1(x_1,y_1)$, $p_2(x_2,y_2)$, $p_3(x_3,y_3)$ and

$p_4(x_4, y_4)$. Furthermore, using the difference between the area of the simplified $\bar{S}$ and that of the polygon S, the coordinates of p_1, p_2, p_3, p_4 can be adjusted.

2.4 Experiment Results

An aerial image (Figure 2.1) with a dimension of 4k x 4k pixels has been used for a feature extraction test. The internal and external orientation parameters of this image are listed as follows:

Exposure center coordinates with 1 sigma (unit: m):

X_L= 512538.071 (s_x = 0.617), Y_L = 216637.932 (s_y = 0.620), Z_L = 1364.956 (s_z = 0.282).

Rotation angles with 1 sigma (unit: degree):

ω = -3.752 (s_w = 0.032), ϕ = 2.005 (s_f = 0.030), κ = -0.807 (s_k = 0.007).

Rotation Matrix M:

0.999288 0.014087 0.035004

-0.016355 0.997726 0.065388

-0.034003 -0.065914 0.997246

Camera model: cam50, calibration date: 30/06/1990

Principal point coordinates: x = 0.001, y = 0.001 (unit: mm)

y scale, ky = 1, Focal length f = 50.000 mm.

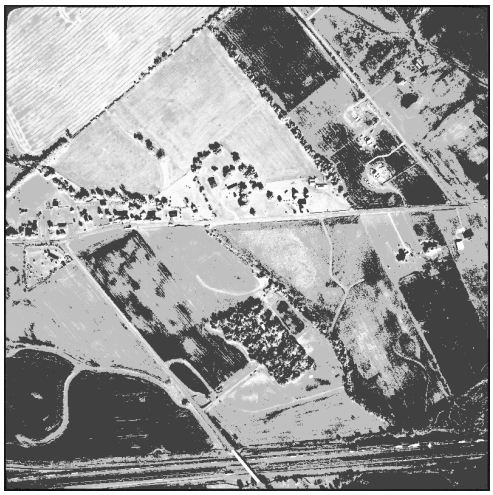


Figure 2.4 Multi-level threshold image segmentation

Figure 2.5 Edge detection at initial resolution (s = 2.0, width = 25)

Figure 2.4 shows an example of image segmentation using a multi-level threshold. The image histogram was calculated, and at each of the two peaks a parabola was used to model a curve segment. The intersection of the parabolas was selected as the threshold. In this way the image is divided into segmented areas. The histogram of each area is checked. If the histogram has two peaks, the threshold is calculated again, followed by a new image segmentation; otherwise the segmented areas are retained. When thresholding, the two values of the areas are assigned as $(256)^{i/(N+1)}$ and $256-(256)^{i/(N+1)}$ respectively, where i is the threshold and N is the highest level (2^N). The region boundaries are modified through a maximum gradient within a 3 x 3 window.

The combined result of edge detection at the initial resolution by LoG (s = 2.0 and width = 25) and Drog is illustrated in Figure 2.5. The result is thinned into a one-pixel width using a morphological thinning algorithm, which preserves topology and connection relationships. The size of the window is set to be greater than $6\sqrt{2}s$ in order to avoid deleterious truncation effects. The edge detection result at a refined resolution (with s =1.6 and width = 23) is represented by Figure 2.6. Considering that noise interference decreases as the resolution decreases, the refined resolution is set to be slightly lower than the initial resolution. The window size is set t Adobe Photoshop Clip Image is too big to be exported o increase the edge strength at the center of the window.

Figure 2.6 Edge detection at refined resolution (s = 1.6, width = 23)

Figure 2.7 Edge selection and edge linking

As represented in Figure 2.7, edges at the initial resolution are filtered by a morphological transformation, where those segments having more than 5 pixels are preserved and those having less are deleted. At the same time, segments and polygons that have endpoints are separated from those that do not. The edge segments of the refined resolution are then dilated. If two end points exist within a 3 x 3-pixel window, they are connected directly. Edge segments that have the same direction, and have their endpoints within a five-pixel radius, are also connected. Edges of length greater than 50 pixels are selected, elongated, dilated, and then intersected with other edges. Finally Hough transform finds straight lines by examining large numbers of collinear points.

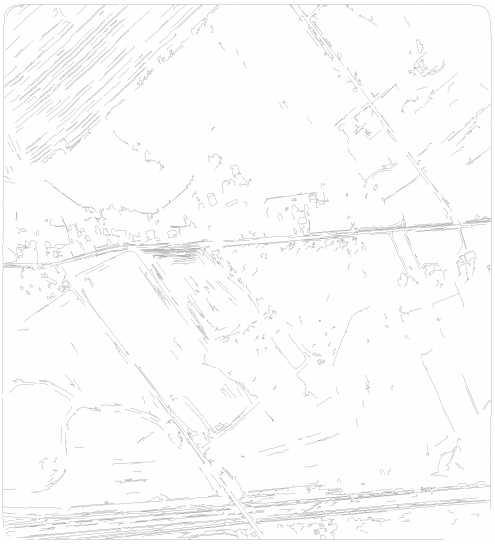


Figure 2.8 Detected lines and polygons

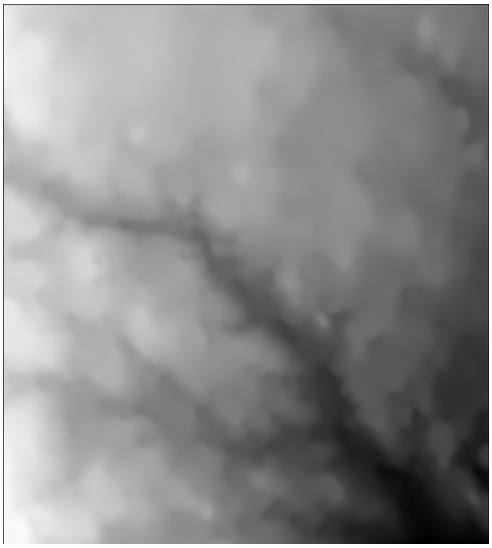


Figure 2.9 DEM interpolation

Figure 2.8 is the combined result of lines from edge detection and polygons from segmentation. All the lines and polygons are longer than 20 pixels.

A DEM was acquired by conversion of a scanned contour map. The map projection of the scanned map was UTM. The DEM was then transformed to SPC (State Plane Coordinate) system in order to correspond to the object coordinate system defined in the photogrammetric model. Geoid height was used to correct the elevation at each grid point. The 3-D object space coordinates (X, Y, Z) are then transformed to image coordinates (x, y, -f) by a collinearity equation, followed by a transformation to pixel coordinates (i, j). Finally, the elevation at each pixel is interpolated by a bilinear interpolation (Figure 2.9).

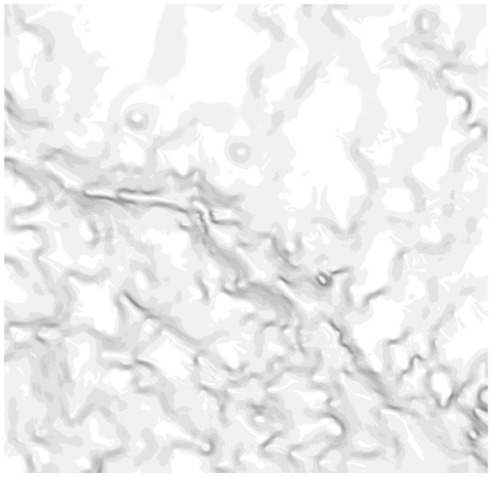


Figure 2.10 Slope grouping

Based on the elevation information at each grid point in Figure 2.9, a terrain slope was calculated by a differential operator. Slopes are grouped into 90 levels, with level 1 to10 covering slopes from 0 to 1 degree and level 10-90 covering slopes 2 degree and above.

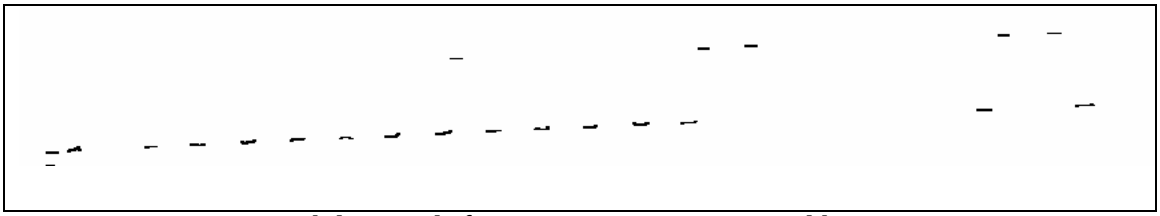


Figure 2.11 Dash line seeds from image segmentation and hit-miss operation

Let us now focus on the bottom part of the original image in Figure 2.1, where a freeway is the dominant feature. Lane-separating lines are bright-color-painted dashed lines. At the original image scale, the dashes are rectangles of the same size and equally spaced. The dashed lines were extracted through image segmentation using the multilevel threshold algorithm. Structured elements are constructed using deformed rectangles considering perspective geometry. Comparison between the segmented areas and structured elements is performed by a hit-or-miss operator to produce dash line seeds for further processing. Figure 2.11 shows the dash line seeds.

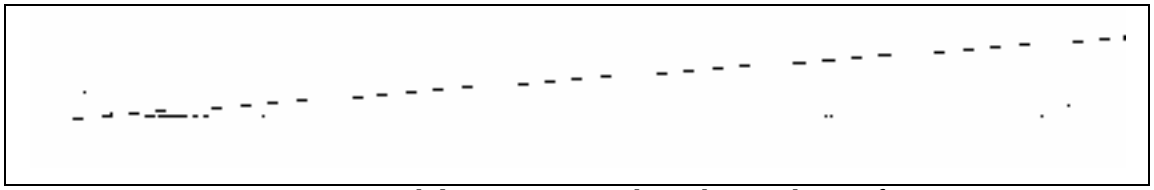


Figure 2.12 Dash line generation through Hough transform

Boundaries of the dash line seeds are next thinned to one-pixel width. A Hough transformation examines all seeds and produces equally spaced dash lines as described in Figure 2.12.

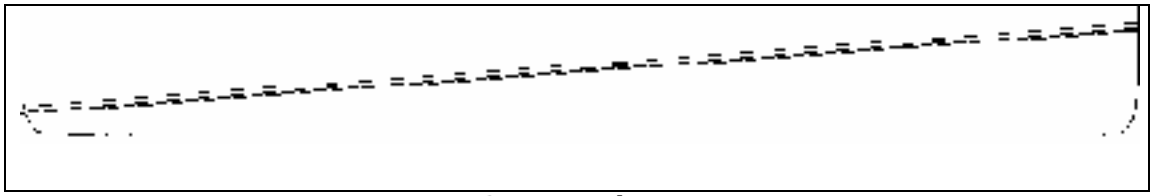


Figure 2.13 Freeway line generation

Once one dash line is produced, other dash lines can be predicted using spacing from the known dash line (Figure 2.13). On the predicted dash lines, predicted and the seeds may intersect. This can be used to confirm the existence of the predicted dash lines.

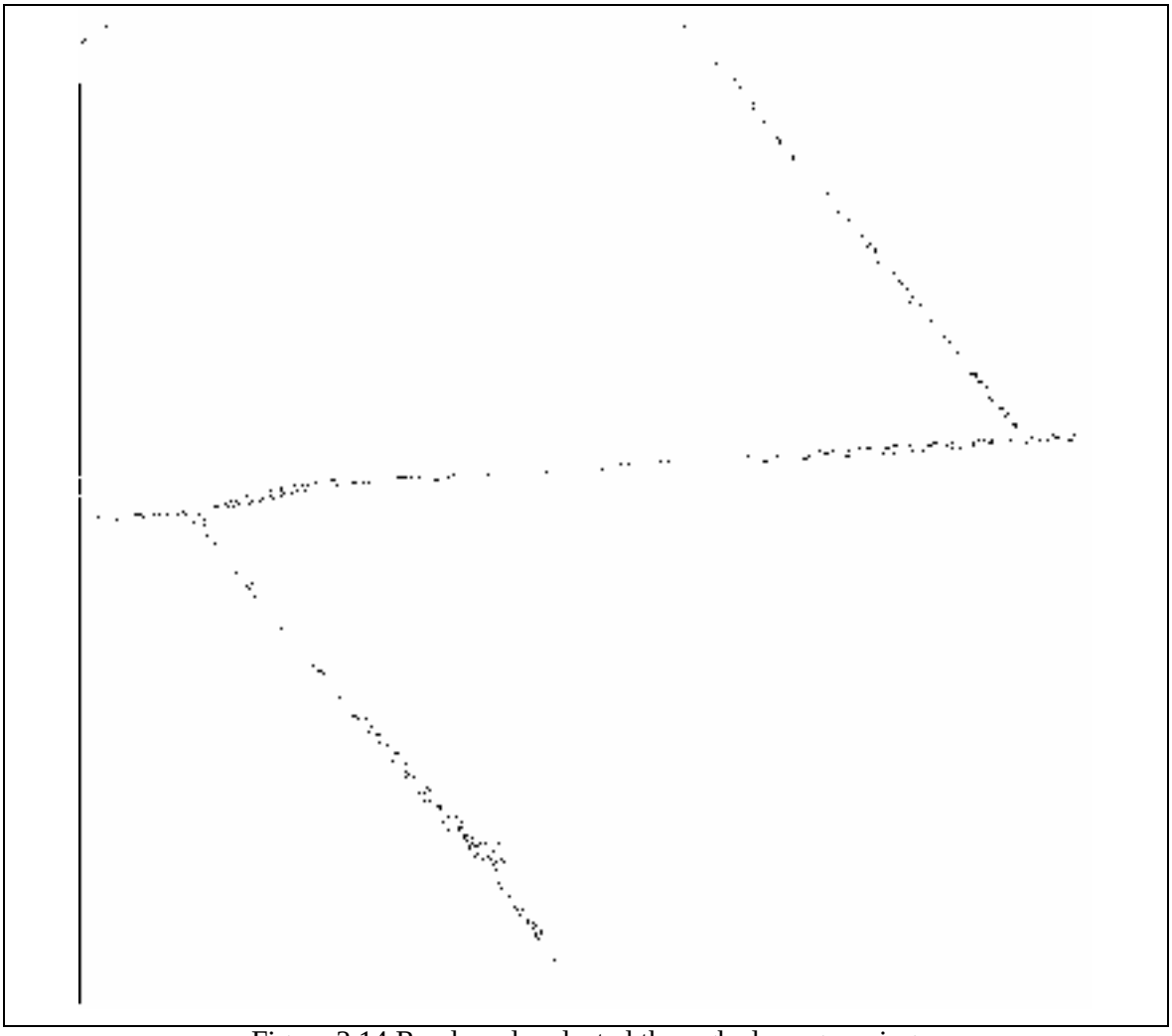


Figure 2.14 Road seeds selected through slope grouping

To capture other roads in the image, which are generally parallel double lines, edges with a length that exceeded a certain threshold (5 pixels) were selected and linked to form road seeds. On the other hand, small-segmented areas were deleted by a morphological transform, while large areas, considered to be elements of the road network, were dilated and connected. Figure 2.14 shows the road seeds that are within DEM grouping level from 0 to 10 (slope < 1 degree).

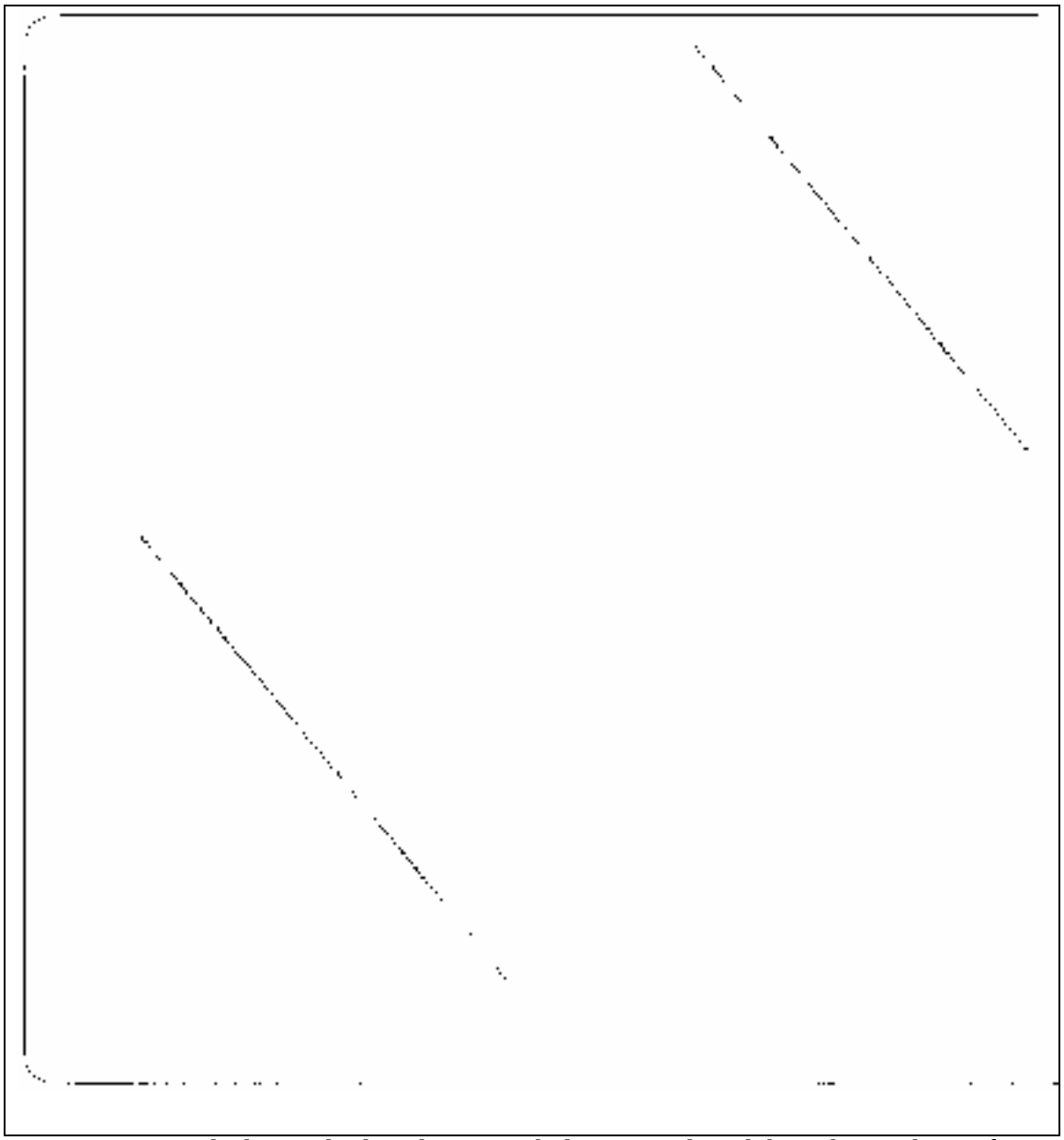


Figure 2.15 Road edge seeds along long straight lines are selected through Hough transform

Hough transform is performed on the selected road edge seeds. Long straight lines are estimated by Hough transformation. Those road edge seeds that are along the long straight lines are selected in Figure 2.15.

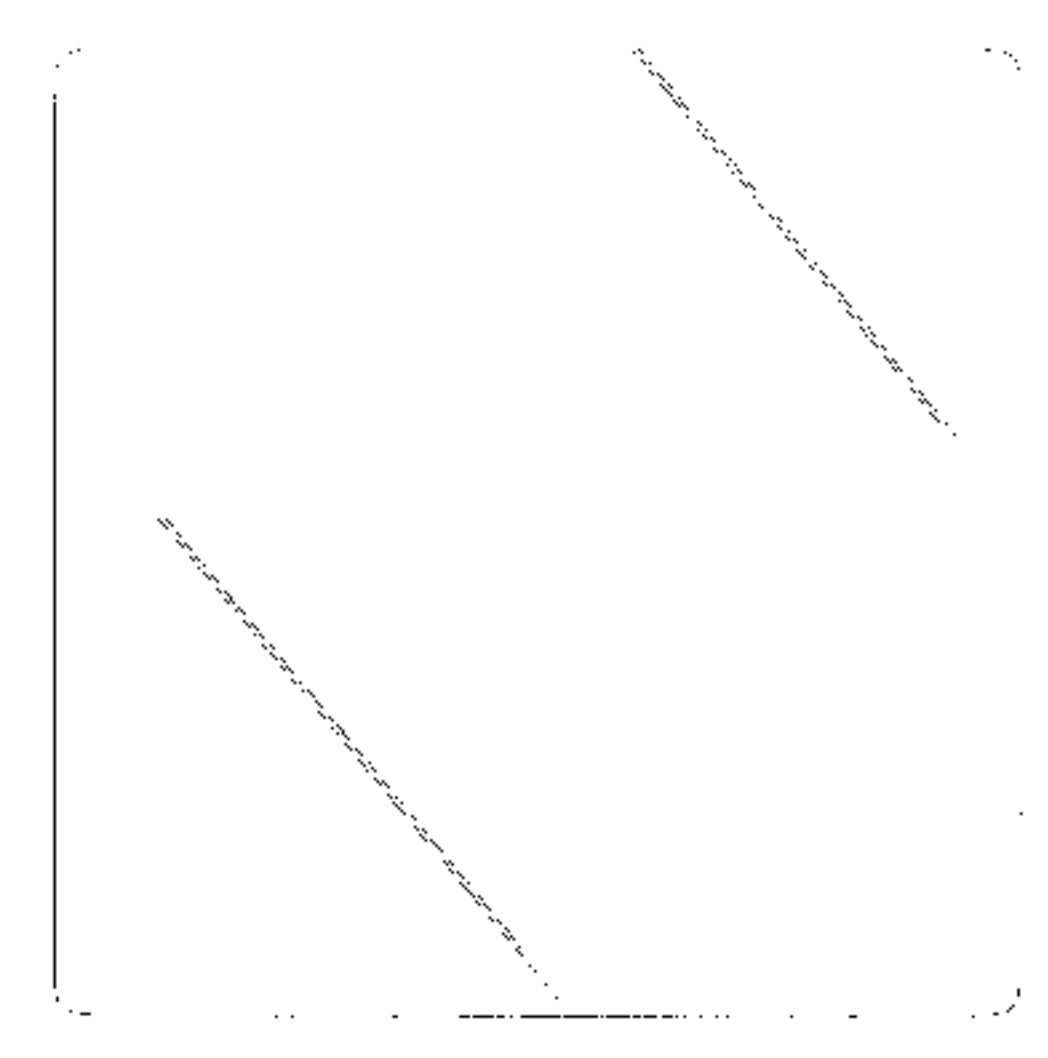


Figure 2.16 Generated parallel straight lines

Road edge seeds on the long straight lines generated by Hough transformation were connected piece-by-piece. Constraints such as road width, parallel direction and anti-parallel gradient are applied to search for two parallel boundary edge lines displayed in Figure 2.16.

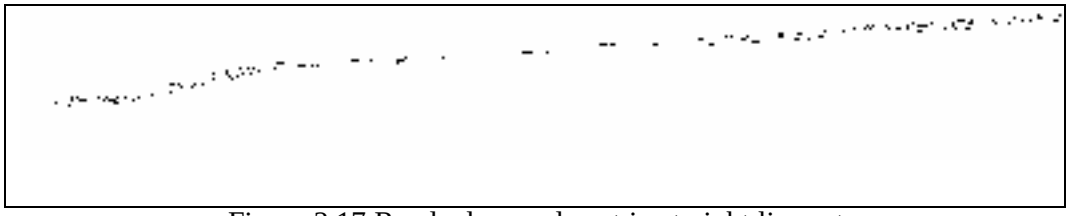


Figure 2.17 Road edge seeds not in straight line set

To capture curved roads in the image, road edge seeds that did not form straight lines were separated from the straight line set (Figure 2.17).

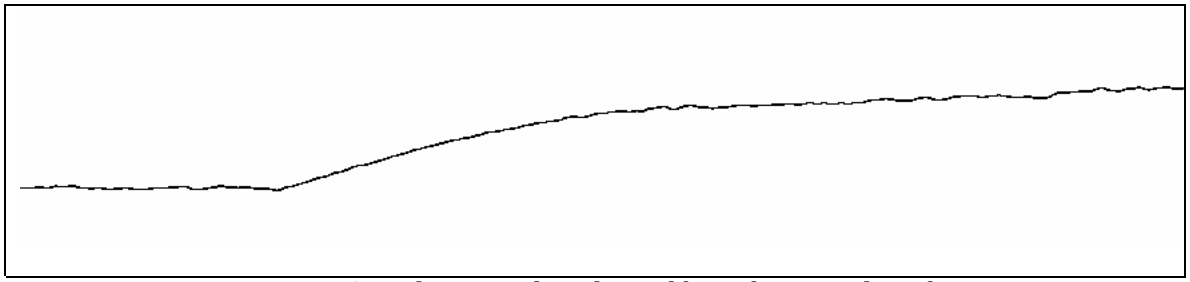


Figure 2.18 medium axis line derived by a thinning algorithm

These non-straight road edge seeds are combined by morphological dilation. A medium axis line is estimated by a thinning algorithm based on the road edge seeds (Figure 2.18).

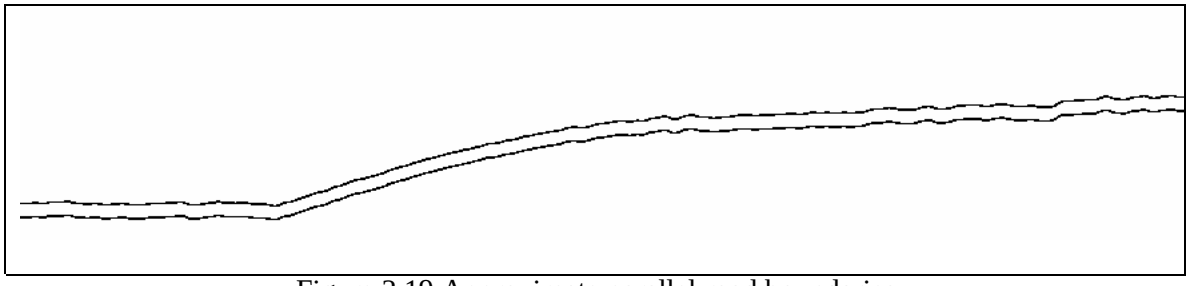


Figure 2.19 Approximate parallel road boundaries

The medium axis lines are dilated using a rectangular element structure along the road direction and with a half road-width size. The dilated contours constitute two approximate parallel road boundaries as shown in Figure 2.19.

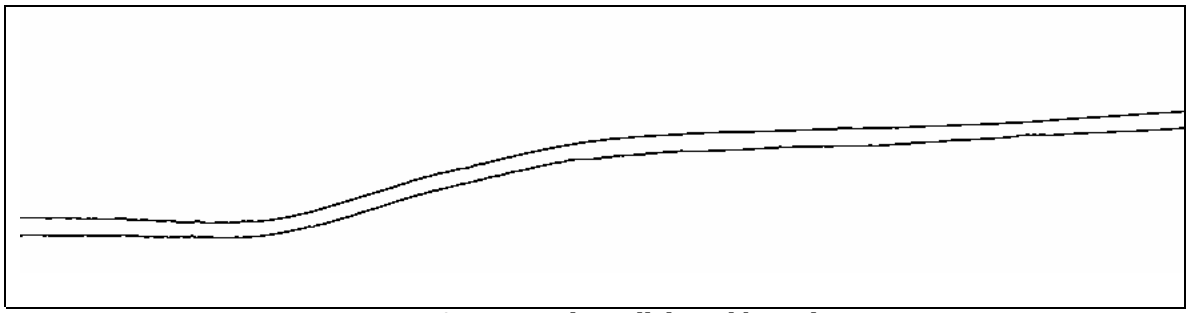


Figure 2.20 Improved parallel road boundaries

The boundary lines are further improved by applying constraints of road edge seeds and imposed smoothness Figure 2.20.

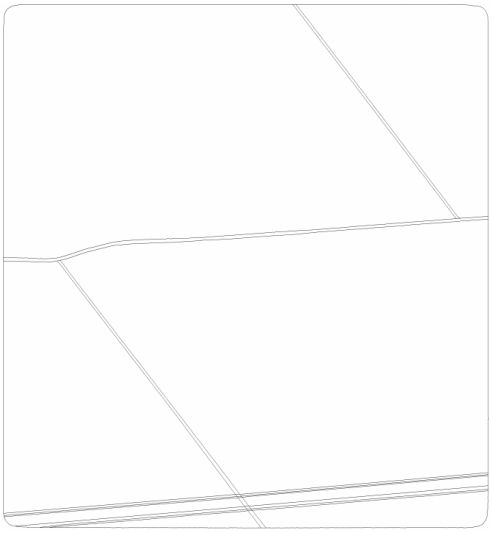


Figure 2.21 Final result of road extraction

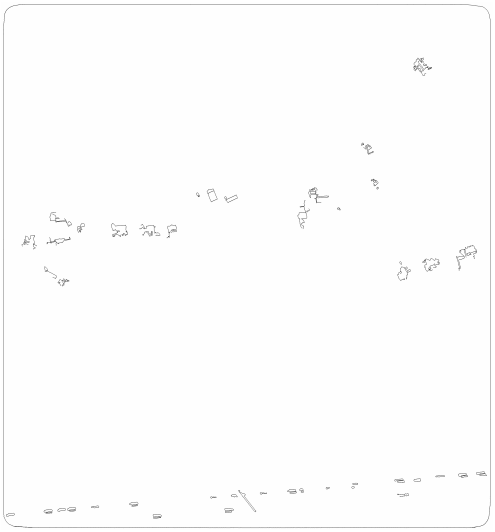


Figure 2.22 Above-ground objects (houses and trucks) detected

Overall, detected edges with slopes within 1 degree are checked against anti-parallel and road width conditions as well as other constraints. Edges passed the checks are used as road edge seeds. End points of the road seeds are detected and dilated using refined resolution. Road seeds are connected through piecewise line fitting and Hough transformation. Morphological thinning and shape smoothing, as well as short arc deleting, are also performed. This results in initial roads that are then refined by a snake deformation model (Figure 2.21).

Edges of aboveground objects, mainly house and truck boundaries, are detected and organized into polygons (Figure 2.22). Figure 2.23 is the overlay of the extracted features on the original AIMS image.

Figure 2.23 Overlay of the extracted features on the original image

3. Multilayer-based Hopfield Neural Network for 3-D Object Recognition

A single layer Hopfield neural network can be applied for recognition of point, line, or region features, as demonstrated in the last year's project report (Li et al. 1997). In the case of 3-D objects described by a hierarchy of regions, lines, and points, a recognition strategy that uses a combination of information is desirable. Geometric constraints and known object models in 3-D space are useful in such a combined recognition strategy. We introduce a two-layer Hopfield neural network that matches region and line features extracted from AIMS imagery using a known 3-D model. Specifically, recognizing trucks on freeways from the georeferenced aerial images is performed as an application of the method. In Young et al. (1997), a multi-layer Hopfield neural network was introduced to recognize pyramid shaped objects. Its optimal selection of neuron initial values provides a means to reach global minimized-energy states without annealing. Suganthan et al. (1995a) developed a selected wavelet function to determine satisfactory initial values for recognizing occluded objects with complicated shapes. Suganthan et al. (1995b) reported a self-organizing network that trains coefficients of the energy function.

In the last project year (Li et. al., 1997), street light poles were recognized using a single layer Hopfield neural network. In 3-D object space, a light pole is defined as a cylinder with a diameter of 21.2cm and a length of 6.795m. These values were obtained by manual photogrammetric measurements of light poles from the mobile mapping images. A priori knowledge that light poles should be vertical and near the mobile mapping van was applied. A light pole model is back projected onto the images and used to recognize image pole features. Light poles appearing in conjugate images are detected, recognized, and used to calculate 3-D locations.

Development of a multilayer network and application for recognition of trucks are major tasks for this project year's object recognition work. Geometric, unary, and binary constraints are imposed to match pairs of candidate features with similar pairs of the 3-D model. Unary constraints consider similarities between single candidate and model features, while binary constraints consider geometric similarities between candidate pairs and model pairs. In the original 4k x 4k AIMS images, there is little difference in the shape of trucks seen from different positions because the freeway is nearly flat.

We use a top-down strategy to achieve object recognition. The problem is treated as an optimization problem, where the correct answer is given when a global minimized energy state is reached. For basic knowledge of application of neural networks in mobile mapping object recognition, please refer to Li et al. (1997). Let C^1_{ik} and C^2_{ikjl} be unary and binary similarity measure respectively. The energy function is

$$E = -A\sum_i\sum_k\sum_j\sum_l C_{ikjl}V_{ik}V_{jl} + B\sum_i(1-\sum_k V_{ik})^2 + C\sum_i\sum_k\sum_{l\neq k}V_{ik}\times V_{il} + D\sum_k(1-\sum_i V_{ik})^2 + E\sum_k\sum_i\sum_{j\neq i}V_{ik}\times V_{jk} \cdot \tag{3.1}$$

The neuron state, V_{ik}, converges to 1.0 if the model feature i matches the candidate image feature k perfectly, otherwise, it is equal or close to 0. Thus, the first term measures similarity between the model and image features. The second term $\sum_i(1-\sum_k V_{ik})^2$ implies that the final states of neurons in the same row add up to 1, and the third term $\sum_i\sum_k\sum_{l\neq k}V_{ik}\times V_{il}$ confirms that there is at most one neuron that has a value greater than 0 in each row. This means that only one candidate image feature matches with each model feature. The forth term $\sum_k(1-\sum_i V_{ik})^2$ implies that the final states of neurons in the same column add up to 1, and the fifth term $\sum_k\sum_i\sum_{j\neq i}V_{ik}\times V_{jk}$ confirms that there is at most one neuron that has a value greater than 0 in each column. That means that each candidate image feature matches with only one model feature. Combining the second term $\sum_i(1-\sum_k V_{ik})^2$ with the third term $\sum_i\sum_k\sum_{l\neq k}V_{ik}\times V_{il}$ gives a solution that forces each model feature to match only one candidate image feature. Similarly, combining the forth term $\sum_k(1-\sum_i V_{ik})^2$ with the fifth term $\sum_k\sum_i\sum_{j\neq i}V_{ik}\times V_{jk}$ gives a solution that guarantees each candidate image feature will match only one model feature. The determination of coefficients A, B, C, D and E depends on how strictly the unique matching condition should be implemented. More discussion on relevant terms and symbols will be introduced later in this report.

3.1 Three Basic Patterns in Object Recognition

Point pattern

When points in images or the model are to be matched, unary and binary constraints can be used. They are often assigned different weights. There are two ways that unary constraints are calculated. The first method represents the patter of a point by its gradient of the neighborhood. The second method calculates the patter attribute within a circle of a certain radius. Binary constraints are more reliable than unary constraints since relative relationships between features are considered. Actually it measures the geometric similarity between a candidate line (formed by a candidate point pair) and a model line (formed by a model point pair). Thus, attributes like length and azimuth can be applied.

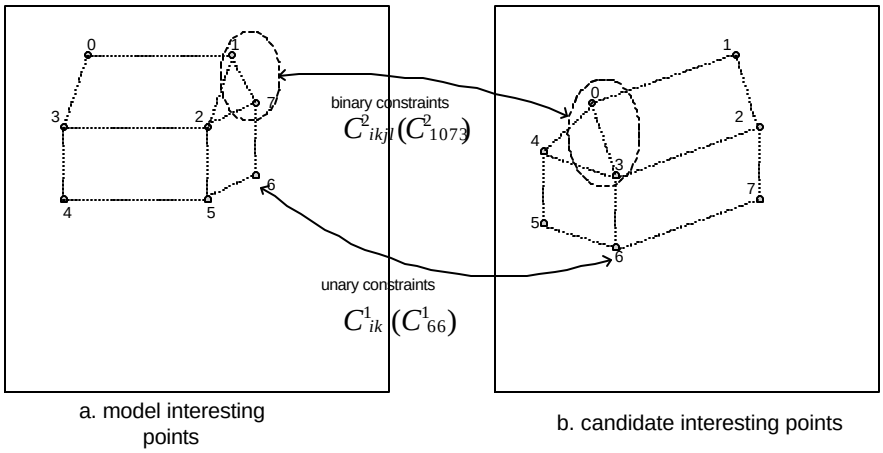


Figure 3.1 Similarity in point patterns

Point pattern matching based on a neural network may be used for finding corresponding interesting points (Nasrabadi and Chang 1992).

Line pattern

Apparently, line patterns have more unary constraints than point patters (e.g. length and azimuth). Below is a list of binary constraints:

(a) Angle formed by a line pair,
(b) Distance between center points of two lines,
(c) Distance between end points of two lines, and

(d) Length ratio between two lines.

The above binary constraints are invariant to transformation and rotation.

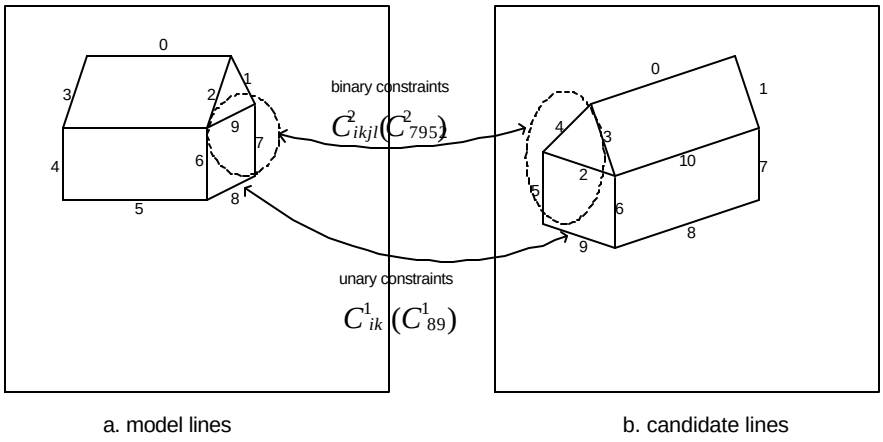


Figure 3.2 Similarity in line patterns

Region pattern

Region patterns have more binary constraints than other two types of patterns. Several geometric constraints describing region similarities are listed below.

(a) Area of a region,
(b) Perimeter of a region,
(c) Fourier descriptors describing the shape of a region, and
(d) Region deformation.

Other binary constraints that are invariant to transformation and rotation are a distance between two region centers and the area ratio between two regions.

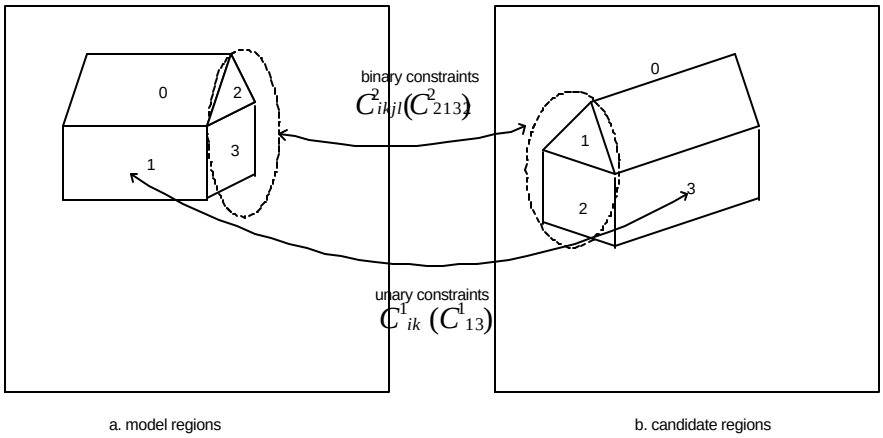


Figure 3.3 Similarity in region patterns

3.2 Single Layer Hopfield Neural Network

3.2.1 Fundamental equations

Object recognition by graph matching, also referred to as morphism, is a mapping from a scene graph to a model graph. The morphism can be categorized on the basis of the constraints that are enforced during the mapping as follows: when the mapping is one-to-one and onto, it is an isomorphism; when it is one-to-one, it is a monomorphism; and when it is many-to-one, it is a homomorphism.

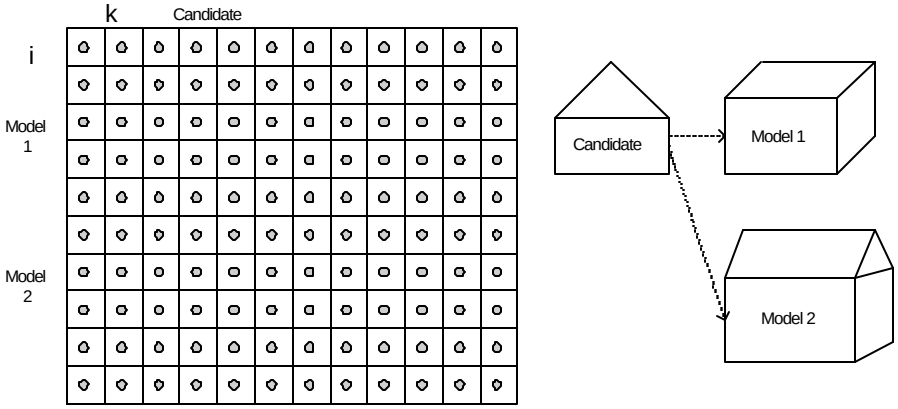


Figure 3.4 Neuron states and model-candidate correspondence

Different values of coefficients A, B, C, D and E in Equation (3.1) apply to various cases of our tasks. For monomorphism, coefficients B, C, D and E are assigned high values based on the assumption that one model feature will uniquely match one candidate feature (Figure 3.4). The final solution yields a one-to-one mapping. In the case of homomorphism, coefficients B and C are assigned low values (even zero) based on the assumption that one model feature will match several image candidate features. In this scenario the final solution yields a many-to-one mapping.

The following is a detailed discussion of equations for single-layer Hopfield neural network. Let C_{ikjl} denote both similarity and disparity between model feature pair (i,j) and candidate image feature pair (k,l). It is then represented as:

$$C_{ikjl} = C^1_{ik} + C^1_{jl} + C^2_{ikjl}. \tag{3.2}$$

where

$$C^1_{ik} = \sum_{n=1}^{N_1} w^1{}_n f_n{}^1(x_{in}, y_{kn}) \tag{3.3}$$

and

$$C^2_{ikjl} = \sum_{n=1}^{N_2} w^2{}_n f_n{}^2(x_{ijn}, y_{k\ln}). \tag{3.4}$$

In the above equations C^1_{ik} and C^2_{ikjl} represent unary and binary similarity respectively. C^1_{ik} encodes compatibility between model feature i and candidate feature k, and C^2_{ikjl} encodes compatibility between the correspondence of the model feature pair (i,j) and that of the candidate feature pair (k,l) (Figure 3.4). f is similarity-measuring function and weighted by w that meets the condition

$$2\sum_{n=1}^{N_1} w^1{}_n + \sum_{n=1}^{N_2} w^2{}_n = 1. \tag{3.5}$$

There are three types of measuring functions

a) Sign function

Sign function is a simple function with one parameter q (Figure 3.5).

$$f(x,y)=\begin{cases}1, & if\ |x-y|<q\\ -1, & otherwise\end{cases} \tag{5.6}$$

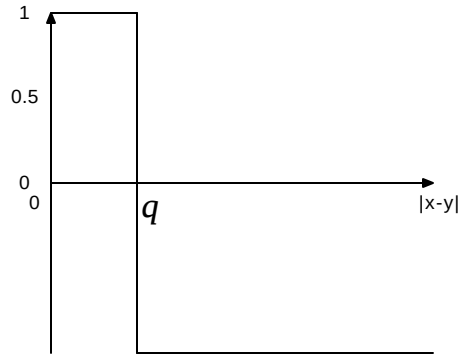


Figure 3.5 Sign function

x and y are similarity measures (such as length of a line) of a candidate feature and model feature, respectively. The parameter q is sometimes difficult to determine in case the measure selected is sensitive. A small change of q may alter the recognition result.

b) Linear function

A linear function

$$f=\begin{cases}1, & if\ |x-y|<a\\ 2|x-y|-(a+b)/(a-b), & if\ a<|x-y|<b\\ -1, & if\ |x-y|>b\end{cases} \tag{3.7}$$

has more practical use than sign function in some cases. In addition to its simplicity, it provides a smoother function in comparison to sign function.

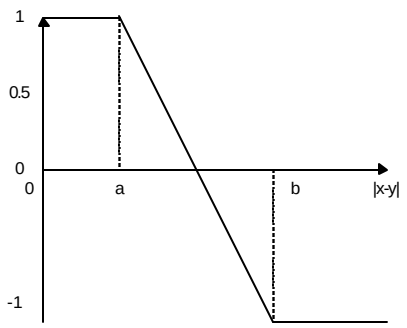


Figure 3.6 Linear function

The linear function has more practical use than the sign function, and it is very easy to handle. The disadvantage of the sign function is that its first derivative is not smooth.

c) Sigmoid function

A sigmoid function

$$f(x,y,k,t) = -\tanh(\, k(|x-y|-t) \tag{3.8}$$

satisfies our decision function requirements appropriately. The shape of the curve near the threshold is approximately that of the linear function. The curve is generally smooth. Depending on the parameter k, the first derivative at t could be also smooth.

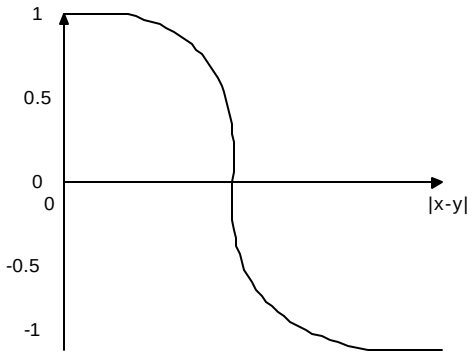


Figure 3.7 Sigmoid function

Neuron State Output function

For neuron i, if its charge is u_i that is defined in section 3.3.1, its neuron state output function is represented as

$$v_i = g(u_i) = \frac{1}{1+e^{-u_i/T}}. \tag{3.9}$$

T is the "temperature" (an annealing term) that determines the speed and quality of the final solution. A very large value of T will cause neuron values to be 0.5, while a very small a value will drive the network to a local minimum state, or converges very slowly. An annealing process keeps the value of T large at the beginning and reduces the T value as iteration progresses. This is important for achieving a global minimum state and a fast convergence.

Initialization

The initial values of neuron states can be chosen randomly as described in Lin et al (1991). It may set the neural network to a local minimum state. As stated above, an annealing process may overcome this problem. However, a careful selection of initial values may not require annealing. If C_{ik} are calculated and examined, initial neuron states are to be computed as

$$V_{ik}^0 = \begin{cases} C_{ik}^1 / \sum_{j \in S_i} C_{ik}^1, & if \sum_{j \in S_i} C_{jk}^1 > w \; and \; C_{ik}^1 > w \\ C_{ik}^1, & if \sum_{j \in S_i} C_{jk}^1 > w \; and \; C_{ik}^1 > 0 \\ 0, & if \; C_{ik}^1 < 0 \end{cases} \tag{3.10}$$

where S_i $is \{k \mid C_{ik}^1 > 0\}$ and $w = \sum_{n=1}^{N_1} w_n$.

Combining matched features

After iterations using homomorphism, each neuron reached its final state V_{ik}. Those final states close to 1 yield matches between corresponding candidate image features with model features. However, there is still a need to put the matched features to form object(s). The following

procedure combines the features under the assumption that there are N features forming an object.

a) Establish N sets of $S_i = \{k | V_{ik} \approx 1\}, i = 1,...,N$. Each set contains all the candidate features that matched the corresponding model features.

b) Establish an empty set Q.

c) Set i to be 1.

d) For each $k \in S_i$ we get $m_{ik} = V_{ik}, k \in S_i$ if Q is empty, otherwise $m_{ik} = \sum_{(j,l) \in Q} (C_{ikjl} + C_{jlik})$. Find the feature k_n that satisfies $\forall k_p \in S_i,\ m_{ik_n} \geq m_{ik_p}$, add (i, k_n) to Q.

e) If $i = N$, one object is recognized and detected, go back to step a); otherwise, $i = i + 1$ and go back to step b).

3.2.2 Truck Recognition

The task is to recognize trucks moving on the highway from stereo aerial image sequences (Figure 3.8) using Hopfield neural network described above. Onboard GPS and INS instruments are used to capture exterior orientation parameters at the time of exposure.

Figure 3.8 AIMS image used for truck recognition

The image has a dimension of 4096 x 4096 pixels. A strip in lower part of the image is cut, enlarged and displayed in Figure 3.9. Trucks are obviously visible. To learn properties of the neural network applied to the truck recognition, we first used the line and region features digitized from the image when displayed on the screen. In Section 3.4 features automatically extracted from the image are used.

Figure 3.9 Zoomed portion of Figure 3.8 where trucks are visible

To simplify the recognition problem, we disregard complicated shapes of the front part of trucks. We focus ourselves on the part of a car that can be simplified as a box. From the camera, the top of a truck and its shadow are always visible (Figure 3.10). Most sides of the truck are not visible in the image.

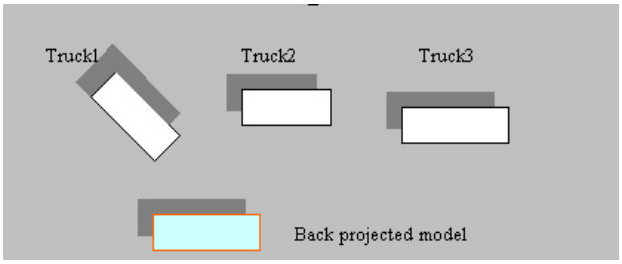


Figure 3.10 Simplified truck image features and back projected model

Point pattern

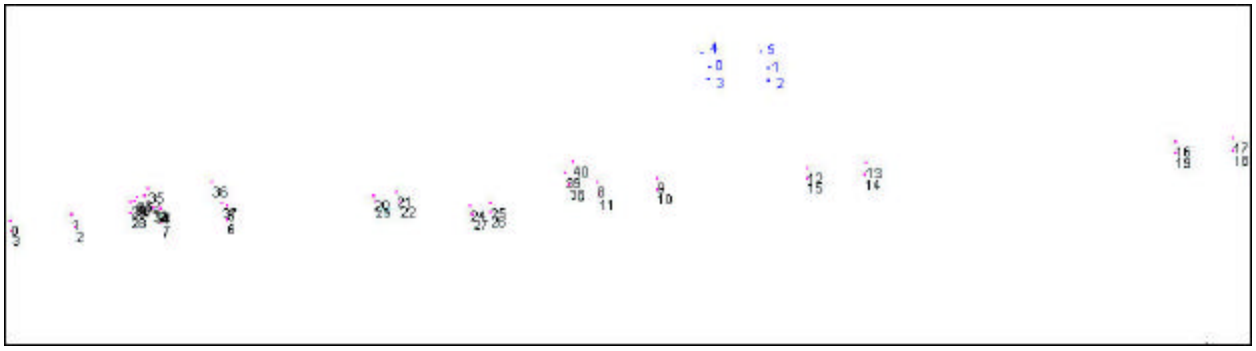


Figure 3.11 Extracted corner points of truck and model points

Figure 3.11 depicts digitized corner points of the trucks (number 0 – 40) and the six back projected model points (number 0 – 5) describing the top and the shadow. We set parameters in the energy function of Equation 3.1 as in Table 3.1.

	A	B	C	D	E
Value	0.6	0.0	0.0	0.8	0.8

Table 3.1 Parameters in the energy function

We use the following two binary constraints:

- Distance between two points, and
- Azimuth angle of a straight line formed by two points.

Using sigmoid function in Equation 3.8, we have $C_{ikjl} = 0.5 f_1^2(x,y) + 0.5 f_2^2(x,y)$, where x and y in f_1 and f_2 are the above binary constraint measures.

Table 3.2 gives final states of neurons. Six model points of the back projected truck top and the shadow are listed in the rows and digitized image points are listed in the columns.

Obviously the point patter match with these constraints do not provide good result.

ID of digitized image point	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Neuron state (Matching with model point 0)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Neuron state (Matching with model point 1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Neuron state (Matching with model point 2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Neuron state (Matching with model point 3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Neuron state (Matching with model point 4)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Neuron state (Matching with model point 5)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID of digitized image point	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Neuron state (Matching with model point 0)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Neuron state (Matching with model point 1)	0	0	0	0.1	0.1	0	0	0	0	0	0	0	0	0	0
Neuron state (Matching with model point 2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Neuron state (Matching with model point 3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Neuron state (Matching with model point 4)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Neuron state (Matching with model point 5)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID of digitized image point	30	31	32	33	34	35	36	37	38	39	40				
Neuron state (Matching with model point 0)	0	0	0	0	0	0	0	0	0	0	0				
Neuron state (Matching with model point 1)	0	0	0	0	0	0	0	0	0	0	0				
Neuron state (Matching with model point 2)	0	0	0	0	0	0	0	0	0	0	0				
Neuron state (Matching with model point 3)	0	0	0	0	0	0	0	0	0	0	0				
Neuron state (Matching with model point 4)	0	0	0	0	0	0	0	0	0	0	0				
Neuron state (Matching with model point 5)	0	0	0	0	0	0	0	0	0	0	0				

Table 3.2 Final neuron states using point pattern matching

Line Pattern

Shadows of trucks within an image vary from truck to truck because of several factors (e.g. pose and shape of head) that complicated their shape. However, the tops of trucks generally approximate a rectangle. Considering that line pattern matching only requires line-wise matching, we use the four edges of a truck as a model, while leaving other features being taken care of in region pattern matching. A typical truck model is 15 meters in length, 3.2 meters wide, and 4 meters in height. We hold this configuration fixed during our line pattern recognition task. With these known values, we back project the model to the current image, which has known interior and exterior orientation parameters from GPS and INS (Figure 3.12).

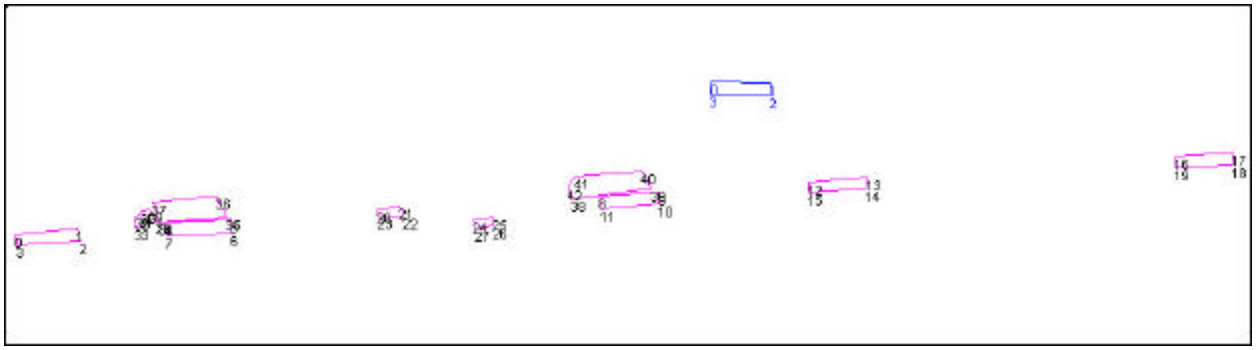


Figure 3.12 Extracted truck line segments and back projected model lines

Here we have one unary constraint:

1) Similarity between length of model edge i and that of candidate line segment k.

The applied binary constraints are:

1) Difference of length ratio between model edges (i and j) and candidate line segments (k and l),
2) Angular difference between model edges (i and j) and candidate line segments (k and l), and
3) Difference of end points distance between model edge i and edge j and that between candidate line segment k and line segment l.

Table 3.3 gives final states of neurons that matches model lines (blue lines marked 0 - 3 in Figure 3.12) with candidate lines (red lines marked 0 – 42 in Figure 3.12). Each neuron state ranges from 0 to 1 and has a subscript that represents the identifier (ID) of an object (truck top in this case). For example, $v_{ij}=1_3$ means that model line I matches with image line j. Furthermore, the matched image line is one of the lines forming object with ID 3. Observing Table 3.3, we know that there are five objects (truck tops) recognized from the scene: object-0 including lines-10, 11, 8, 9 (red lines in Figure 3.3); object-1 including lines-16, 17, 18, 19; object-2 including lines-12, 13, 14, 15; object-3 including lines-2, 3, 0, 1; and object-4 including lines 6, 7, 4, 5. There are still some neurons with a final state of one that do not from an object, for example (36, 0). This is because the neuron (or matched line) does not have a partner to form an object with.

ID of extracted line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Neuron state (Matching with model line 0)	0	0	1_3	0	0	0	1_4	0	0	0	1_0	0	0	0	1_2
Neuron state (Matching with model line 1)	0	0	0	1_3	0	0	0	1_4	0	0	0	1_0	0	0	0
Neuron state (Matching with model line 2)	1_3	0	0	0	1_4	0	0	0	1_0	0	0	0	0	1_2	0
Neuron state (Matching with model line 3)	0	1_3	0	0	0	1_4	0	0	0	1_0	0	0	1_2	0	0

15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	0	0	1_1	0	0	0	0	0	0	0	0	0	0	0
1_2	0	0	0	1_1	0	0	0	0	0	0	0	0	0	0
0	1_1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1_1	0	0	0	0	0	0	0	0	0	0	0	0

30	31	32	33	34	35	36	37	38	39	40	41	42		
0	0	0	0	0	0	1	0	0	0	0.4	0	0		
0	0	0	0	0	0	0	0	0	0	0	0	0		
0	0	0	0	0.4	0	0	0	0	0	0	0	0		
0	0	1	0	0	0	0	0	0	0	0	0	0		

Table 3.3 Final neuron states by line pattern matching

Region Pattern

Figure 3.13 Extracted regions and back projected model region

Figure 3.13 shows the extracted regions (in red) from a partial image and two regions (0 and 1) of the back projected model. To match the model regions with the extracted image regions,

both unary and binary constraints for region pattern matching are used. The unary constraints are:

1) Similarity between area of model region i and that of candidate region k,
2) Similarity between perimeter of model region i and that of candidate k,
3) Similarity between average gray value of model region i and that of candidate k, and
4) Similarity between width-to-length ratio of model region i and that of candidate k as $with(i)/length(i)$ and $with(k)/length(k)$.

The binary constraints include

1) Difference between the distance from the center point of model region i to that of model region j and that from candidate region k to candidate region l, and
2) Gray value difference between model region i and model region j and that between candidate region k and candidate region l.

	A	B	C	D	E
Value	0.6	0.0	0.0	0.8	0.8

Table 3.4 Parameters of the energy function used for region pattern matching

ID of extracted region	0	1	2	3	4	5	6	7	8	9	10	11	13
Neuron state (Matching with model region 0)	1_1	1_0	1_2	1_3	0.99	0	0	0	0	0	0	0	0
Neuron state (Matching with model region 1)	0	0	0	0	0	0	0	0	1_0	1_2	1_3	1_1	1

Table 3.5 Final neuron states by region pattern matching

Given energy function parameters in Table 3.4, Table 3.5 shows the final neuron states of the region pattern matching. Similar to the result of line pattern matching, there are four objects (truck - top and shadow) recognized from the scene: Object-0 including 1 and 8; object-1 including 0 and 11; object-2 including 2 and 9; and object-3 including 3and 10. Referring back to Figure 3.3, we see that the recognized region pairs are correct truck tops and their shadows. Region 4 may be a truck top that has no corresponding shadow. Likewise, region 13 may be the shadow of a truck. In region pattern matching, there is no strict requirement for the number of line segments forming a candidate and corresponding model to be the same. However, we may

lose a shape in region pattern recognition that would not otherwise be lost in line pattern recognition. For example, candidate region 4 actually is the top of a truck but we can not ascertain this by region pattern methods. Given this reality, we have adopted a multi-layer Hopfield neural network that takes advantage of both line pattern and region recognition algorithms in a single network.

3.3 Multilayer Hopfield Neural Network

The above object recognition is based on several separate single layer neural networks. However, the interrelationship between line pattern recognition and region pattern recognition adds more constraints and thus achieves better results. Our objective is to utilize a two-layer network for truck recognition. The interrelationship is based on the following realities:

1) The top of a truck shown in the image is nearly rectangular in shape, while its shadow has a more complex shape due to the sun light direction and truck head shape;
2) The line pattern recognition could yield a match for the top, but fail in shadow verification, which is a very important clue; and
3) The region pattern recognition considers the top and shadow at the same time, but it does not take full advantages of line patterns.

Connections among neurons in each single layer are fully dependent on geometric and photogrammetric constraints and are fixed before the initial iteration. During iterations the interconnections between the two layers vary. Let L_1 denote layer 1, which is a line pattern layer, and L_2 denote layer 2, which is a region pattern layer. We thus have an energy function

$$E = E_1(L_1) + E_2(L_2). \tag{3.11}$$

where $E_1(L_1) = E_{11}(L_1, L_1) + E_{12}(L_1, L_2)$ and $E_2(L_2) = E_{22}(L_2, L_2) + E_{21}(L_2, L_1)$. $E_{11}(L_1, L_1)$ and $E_{22}(L_2, L_2)$ are same as the terms in Equation (3.1). The energy relevant to interlayers are

$$E_{12} = a_1 \times \left(-\frac{1}{2}\right) \sum_{i_1} \sum_{k_1} \sum_{i_2} \sum_{k_2} B^{12}{}_{i_1 k_1 i_2 k_2} V_{i_1 k_1} V_{i_2 k_2} \tag{3.12}$$

$$E_{21} = a_2 \times \left(-\frac{1}{2}\right) \sum_{i_2} \sum_{k_2} \sum_{i_1} \sum_{k_1} B^{21}{}_{i_2 k_2 i_1 k_1} V_{i_2 k_2} V_{i_1 k_1}. \tag{3.13}$$

$B^{12}_{i_1k_1i_2k_2}$ is a connectivity variable from neuron (i_1,k_1) in layer L_1 to neuron (i_2,k_2) in layer L_2. $B^{21}_{i_2k_2i_1k_1}$ is a similar term. They change dynamically during iterations. We also have $B^{12}_{i_1k_1i_2k_2} \neq B^{21}_{i_2k_2i_1k_1}$ because contributions from one layer to another layer are non-symmetric (Figure 3.14). Using energy equation (3.11), we can recognize the trucks when a global minimized energy value is achieved.

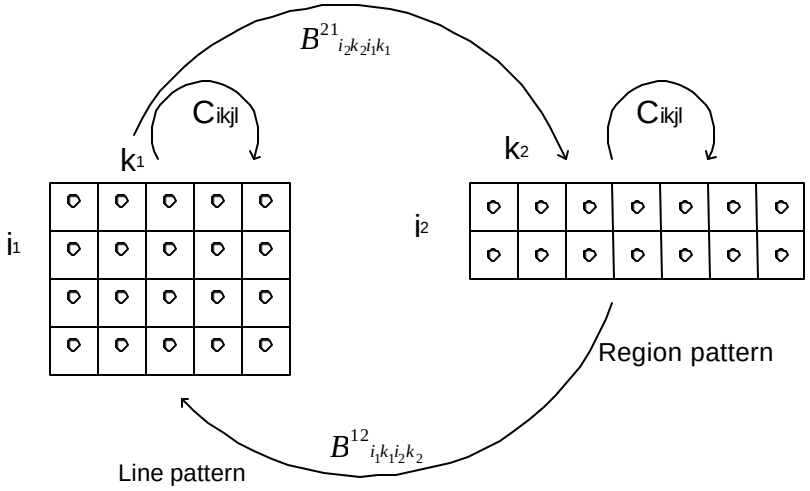


Figure 3.14 Two layer (line and region) Hopfield neural network

The connectivity term

$$B^{12}_{i_1k_1i_2k_2} = \begin{cases} 2\times\left(V2_{i_2k_2} - \frac{1}{2}\right) & if\ Line\ k_1 \in Area\ k_2\ and\ i_2 = 0 \\ -2\times\left(V2_{i_2k_2} - \frac{1}{2}\right) & if\ Line\ k_1 \in Area\ k_2\ and\ i_2 = 1 \\ 0 & otherwise \end{cases} \tag{3.14}$$

contributes when a model region is a truck top and a candidate line belongs to a candidate region or when the model region is a truck shadow and the candidate line belongs to candidate region. Similarly, $B^{21}_{i_2k_2i_1k_1}$ is defined as

$$B^{21}_{i_2k_2i_1k_1} = \begin{cases} 2\times\left(V1_{i_1k_1} - \frac{1}{2}\right) & if\ Line\ k_1 \in Area\ k_2\ and\ i_2 = 0 \\ -2\times\left(V1_{i_1k_1} - \frac{1}{2}\right) & if\ Line\ k_1 \in Area\ k_2\ and\ i_2 = 1 \\ 0 & otherwise \end{cases} \quad (3.15)$$

3.3.1 Process of the two layer Hopfield neural network

The following describes the process of the two layer Hopfield neural network:

Step 1. Calculate connectivity parameters $C1_{ikjl}$ of layer 1 and $C2_{ikjl}$ of layer 2.

Step 2. Set the initial states $V1_{i_1k_1}$ of layer 1 and $V2_{i_2k_2}$ of layer 2 using equation (3.11) respectively.

Step 3. Obtain $B^{12}_{i_1k_1i_2k_2}$ using equation (3.14).

Step 4. Update the values of $u1_{i_1k_1}$ and $V1_{i_1k_1}$

for ($i_1 = 0;\ i_1 < m_1;\ i_1$++)

for ($k_1 = 0;\ k_1 < n_1;\ k_1$++)

$$u1^{t+1}_{i_1k_1} = u1^{t}_{i_1k_1} + \tfrac{1}{6}(K_1 + 2*K_2 + 3*K_3 + K_4)$$

where $K_1 = h\times f(u1^{t}_{i_1k_1}) = h\times(A\sum_j\sum_l C1_{i_1k_1jl}V1^{t}_{jl} - B(\sum_l V1_{i_1l} - 1) - C\sum_{l\neq k_1} V1_{i_1l} - D(\sum_j V1_{jk_1} - 1) - E\sum_{j\neq k_1} V1_{jk_1} - \partial_1\sum_{i_2}\sum_{k_2} B_{i_1k_1i_2k_2}V2_{i_2k_2})$

$$K_2 = h\times f(u1^{t}_{i_1k_1} + \tfrac{1}{2}K_1)$$

$$K_3 = h\times f(u1^{t}_{i_1k_1} + \tfrac{1}{2}K_2)$$

$$K_4 = h\times f(u1^{t}_{i_1k_1} + K_3)$$

Step 6. Compute $V1^{t+1}_{i_1k_1} = g(u1^{t+1}_{i_1k_1})$.

Step 7. Obtain $B^{21}_{i_2k_2i_1k_1}$ using equation (3.15).

Step 8. Update the values of $u2_{i_2k_2}$ and $V2_{i_2k_2}$

for ($i_2 = 0;\ i_2 < m_2;\ i_2$++)

for ($k_2 = 0;\ k_2 < n_2;\ k_2$++)

$$u2_{i_2k_2}^{t+1} = u2_{i_2k_2}^{t} + \tfrac{1}{6}(K_1 + 2*K_2 + 3*K_3 + K_4)$$

where $K_1 = h \times f(u2_{i_2k_2}^{t}) = h \times (A\sum_j \sum_l C2_{i_2k_2jl} V2_{jl}^{t} - B(\sum_l V2_{i_2l} - 1) - C\sum_{l \neq k_2} V2_{i_2l} -$

$$D(\sum_j V2_{jk_2} - 1) - E\sum_{j \neq k_2} V2_{jk_2} - \partial_2 \sum_{i_1} \sum_{k_1} B_{i_2k_2i_1k_1} V1_{i_1k_1})$$

$$K_2 = h \times f(u2_{i_2k_2}^{t} + \tfrac{1}{2} K_1)$$

$$K_3 = h \times f(u2_{i_2k_2}^{t} + \tfrac{1}{2} K_2)$$

$$K_4 = h \times f(u2_{i_2k_2}^{t} + K_3)$$

Step 9. Update: $V2_{i_2k_2}^{t+1} = g(u2_{i_2k_2}^{t+1})$.

Step 10. If convergence, then stop, otherwise go back to step 3.

3.3.2 Result using two layer Hopfield neural network

ID of extracted line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Neuron state (Matching with model line 0)	0	0	1_3	0	0	0	1_4	0	0	0	1_0	0	0	0	1_2
Neuron state (Matching with model line 1)	0	0	0	1_3	0	0	0	1_4	0	0	0	1_0	0	0	0
Neuron state (Matching with model line 2)	1_3	0	0	0	1_4	0	0	0	1_0	0	0	0	0	1_2	0
Neuron state (Matching with model line 3)	0	1_3	0	0	0	1_4	0	0	0	1_0	0	0	1_2	0	0

15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	0	0	1_1	0	0	0	0	0	0	0	0	0	0	0
1_2	0	0	0	1_1	0	0	0	0	0	0	0	0	0	0
0	1_1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1_1	0	0	0	0	0	0	0	0	0	0	0	0

30	31	32	33	34	35	36	37	38	39	40	41	42		
0	0	0	0	0	0	0	0	0	0	0	0	0		
0	0	0	0	0	0	0	0	0	0	0	0	0		
0	0	0	0	0	0	0	0	0	0	0	0	0		
0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 3.6 Final neuron states of the line layer by a two-layer network

Table 3.6 and 3.7 list the results of combined line pattern (layer 1) and region pattern (layer 2) matching by the two layer Hopfield neural network. Comparing to Table 3.3 and Table 3.5 where line and region were matched separately, no change has occurred. However, the speed of convergence is accelerated because line patterns strengthen the certainty of region patterns to which these lines belong, and vice versa. In addition, the multilayer approach should provide more robust results because of constraints between layers. This advantage will be demonstrated in the following case where the extracted lines are from an automatic procedure.

ID of extracted region	0	1	2	3	4	5	6	7	8	9	10	11	13
Neuron state (Matching with model region 0)	1_1	1_0	1_2	1_3	1	0	0	0	0	0	0	0	0
Neuron state (Matching with model region 1)	0	0	0	0	0	0	0	1	1_0	1_2	1_3	1_1	1

Table 3.7 Final neuron states of the region layer by a two-layer network

3.4 Experiment Using Detected Polygons and Lines

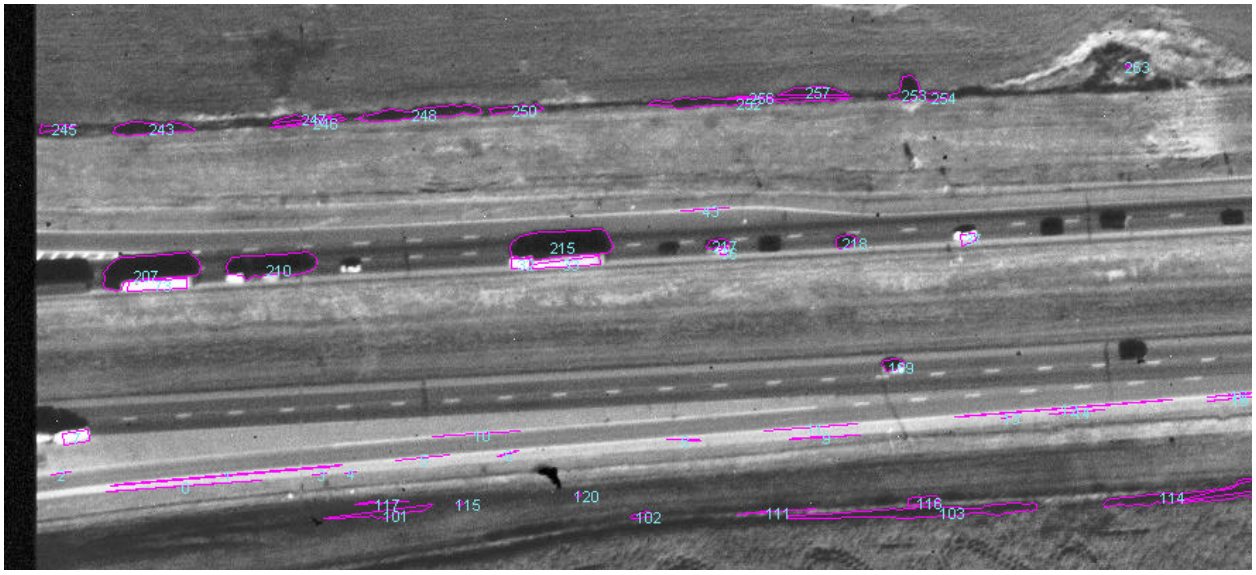
So far the input lines and regions were all manually digitized from the screen. To automate the feature extraction procedure and check the robustness of the recognition algorithms, we will use the lines and polygons extracted from a strip of the image (Figure 3.15). To match the line pattern, we have four model line segments and 5,000 candidate line segment which require $4\times5000\times4\times5000=$ 400M memory units to store matrix C_{ikjl}, 4×5000 =20000 memory units to store a matrix of U_{ik}, and $4\times5000=20000$ memory units to store matrix V_{ik}. In addition, for region pattern recognition, we have two model regions (top and shadow). There are 500 candidate polygons that require $2\times500\times2\times500=1MB$ memory units to store matrix C_{ikjl}, $2\times500=1000$ memory units to store matrix U_{ik}, and $2\times500=1000$ memory units to store matrix V_{ik}. To apply the two-layer Hopfield neural network we need at least 40.10442M memory units. Also, because the computation complexity in each iteration is $O(n^2m^2)$, the time needed for all iterations is very significant.

Due to both space and time limitations, we define image subspaces. In each such image subspace there is possibly only truck inside and a neural network is established for truck

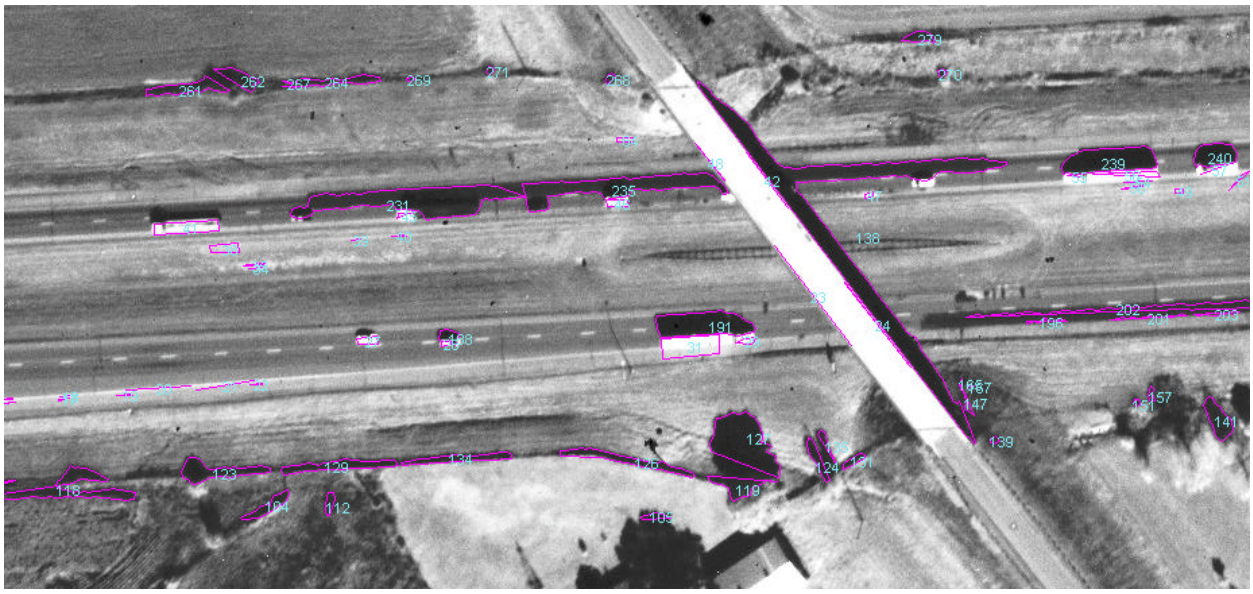
recognition. It is to note that each truck should have at least a top which is nearly a rectangle and has a bright average gray value (say greater than 125). Suppose that $G_{avg}(i)$ is the average gray value of i-th polygon and $G_{avg}(Img)$ is the average gray value of all polygons of the image. The system will check each polygon. If for polygon i we have $G_{avg}(i) > G_{avg}(Img)$, the polygon is a bright polygon and is a candidate of the top of a truck. A window is defined with the center at the polygon centroid. The window size is chosen in such a way that it is large enough to include a truck and its shadow, but small enough to exclude a second truck. This window size is held fixed through out the procedure. Furthermore, the back-projected model does not change appreciably from place to place. Under the above circumstances, it is clear that only one truck would be in the window. Therefore, we apply monomorphism (one-to-one mapping) in the matching process. This makes the coefficients B and C in the energy function significant values (0.6), while other coefficients are maintained the same.

A two layer Hopfield neural network was used to recognize the trucks among the extracted lines and polygons in Figure 3.15. According to final neuron states, there are two categories of features that may match the trucks.

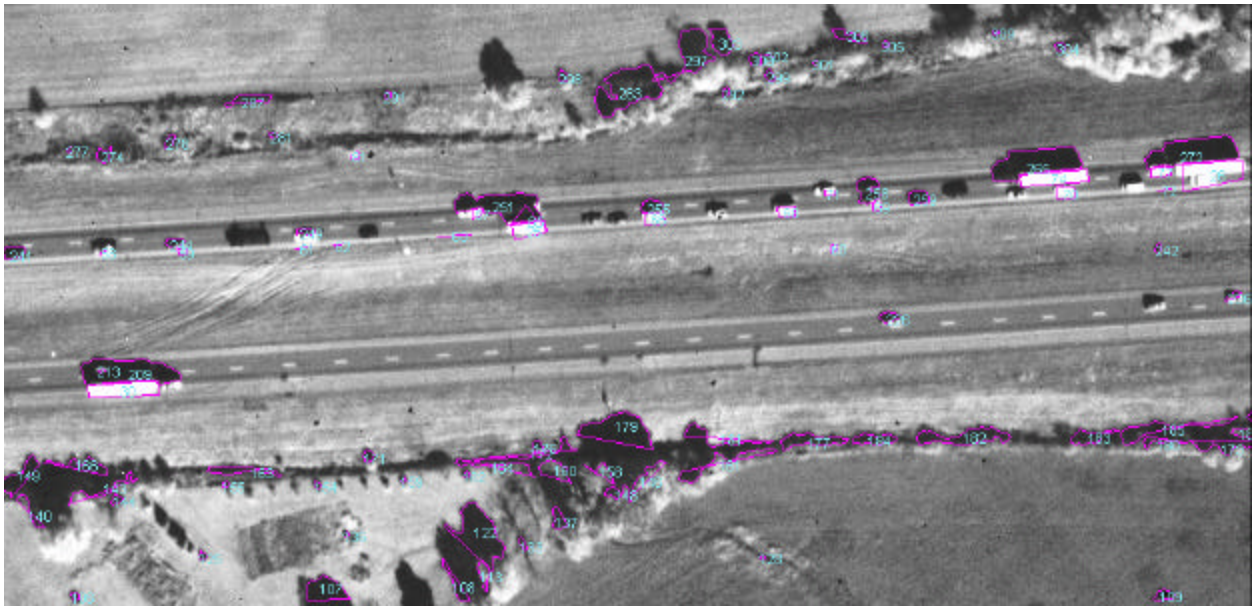
I) Two polygons match with the model, one with the truck top and the other with the shadow of the model. The edges of the polygon that matches with the model top match with those of the model top.



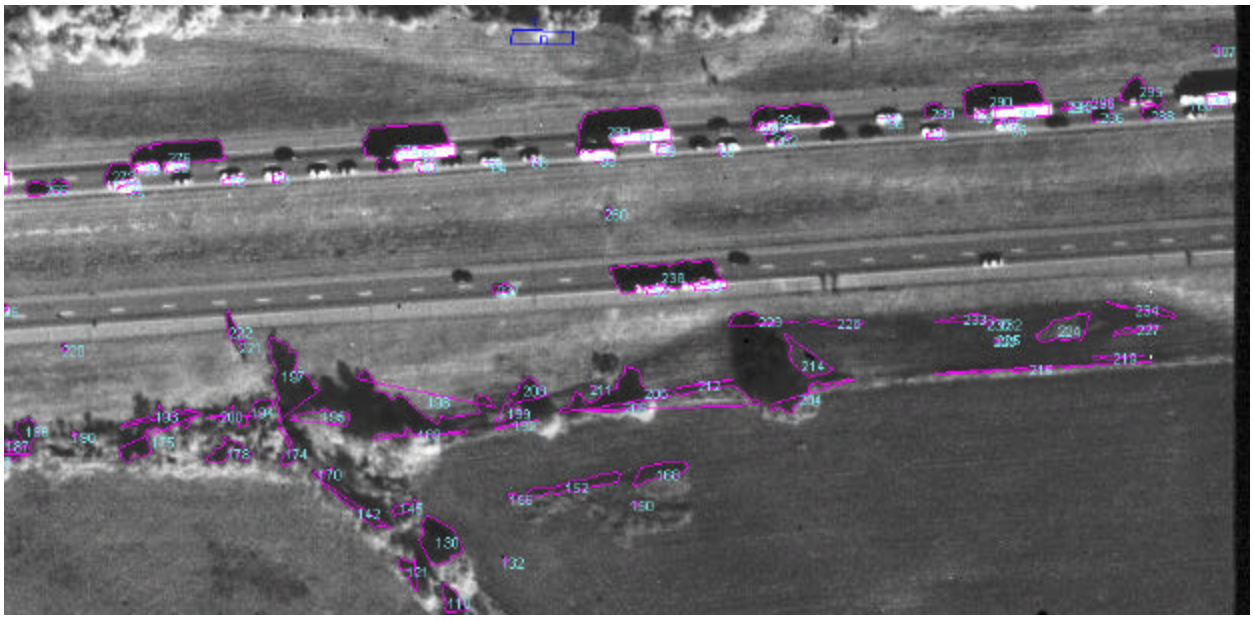
(a) Part I of automatically extracted lines and polygons



(b) Part II of automatically extracted lines and polygons



(c) Part III of automatically extracted lines and polygons



(d) Part IV of automatically extracted lines and polygons

Figure 3.15 (a) Part I, (b) Part II, (c) Part III and (d) Part IV of automatically extracted lines and polygons overlaid on the image at the original scale

II) In the region pattern layer there is no polygon pairs matching with the top and shadow of the model. However, there are polygon edges matching with those of the model top in the line pattern layer. This may be due to a loss of shadow information caused, for example, by a small gray value difference or occlusion.

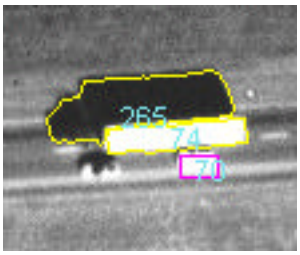


Figure 3.16 A candidate truck that matches with both region pattern model and line pattern model

Figure 3.17 shows a truck in the upper right of Figure 3.18 (c). The extracted lines and polygons are of very high quality. Polygon 74 describes the top of the truck and polygon 265 describes its shadow. Table 3.7 is the computed relevant neuron states of the truck extracted from single layer networks and a two layer network. Obviously it is a category I match.

	Single layer Hopfield neural network				Two layer Hopfield neural network			
	Line pattern				Line pattern layer			
	Model line ID				Model line ID			
Region Id (line ID)	0	1	2	3	0	1	2	3
74 (296)	0.000	0.000	1.000	0.000	0.028	0.000	1.000	0.000
74 (297)	0.000	0.000	0.000	1.000	0.000	0.031	0.000	1.000
74 (298)	1.000	0.000	0.000	0.000	1.000	0.000	0.028	0.000
74 (299)	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.031
	Region pattern				Region pattern layer			
	Model region ID				Model region ID			
Region ID	0		1		0		1	
74	1.000		0.002		1.000		0.002	
265	0.000		1.000		0.000		1.000	

Table 3.7 Category I match, an example from single layer networks and a two-layer network

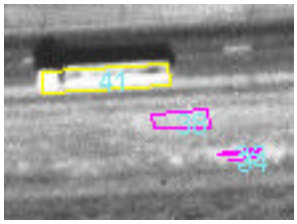


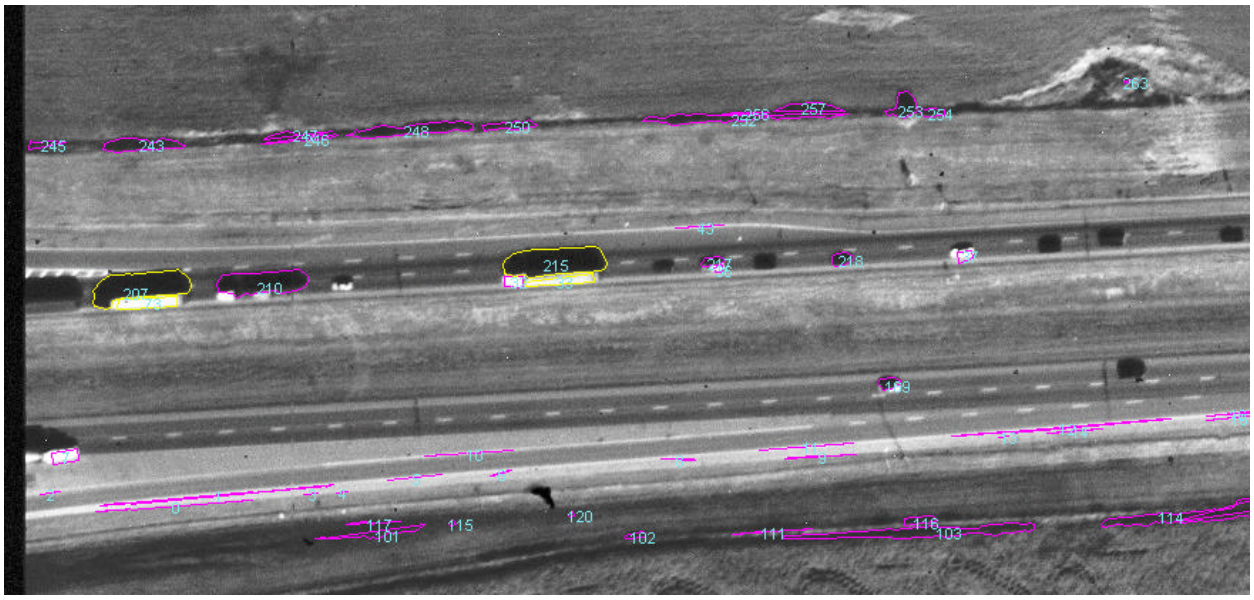
Figure 3.17 A candidate truck that matches with line pattern model only

The truck in Figure 3.17 lies in the upper left of Figure 3.18 (b). Table 3.9 is the computed relevant neuron states of the same truck. The edges of polygon 41 match with the model in the line pattern layer. But the polygon does not match with the shadow of the model in the region pattern layer. The two-layer neural network is enhanced by the line layer, which makes neural states of the region pattern layer higher than those of a single layer because the four line segments of the polygon match the model lines in the line pattern layer very well. This is an example of a category II match.

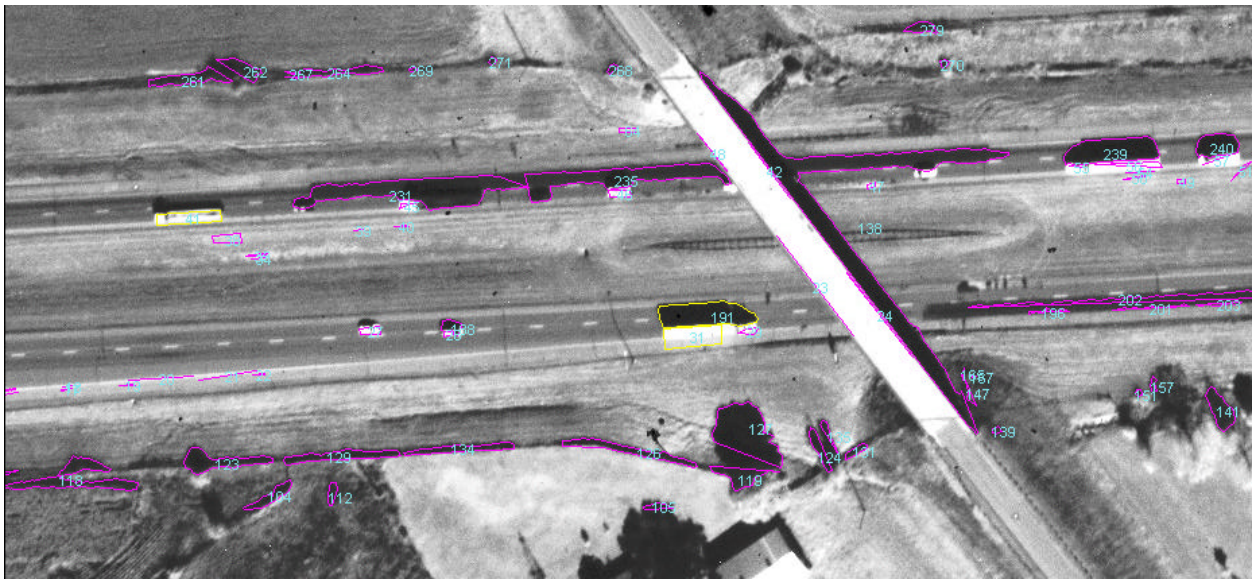
	Single layer Hopfield neural network				Two layer Hopfield neural network			
	Line pattern				Line pattern layer			
	Model line ID				Model line ID			
Region Id (line ID)	0	1	2	3	0	1	2	3
41 (164)	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
41 (165)	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000
41 (166)	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000
41 (167)	0.000	0.000	0.000	1.000	0.000	1.000	0.000	1.000
	Region pattern				Region pattern layer			
	Model region ID				Model region ID			
Region Id	0		1		0		1	
218	0.968		0.344		1.000		0.245	

Table 3.9 Category II match, an example from single layer networks and a two-layer network

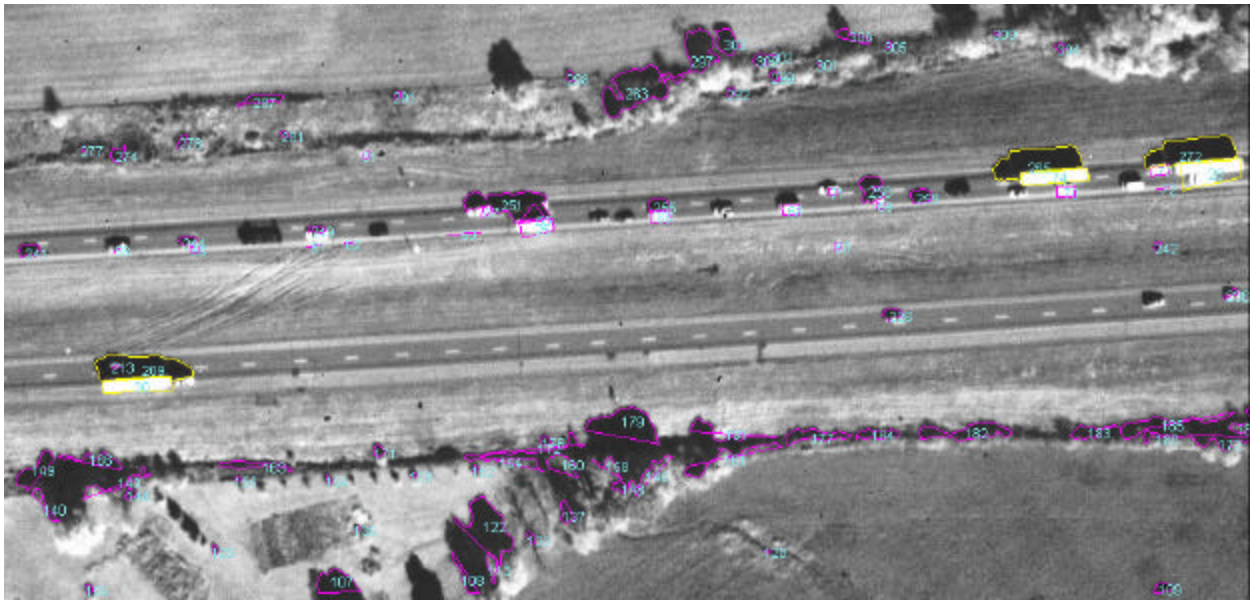
Figure 3.18 illustrates all recognized trucks including category I and II in yellow lines.



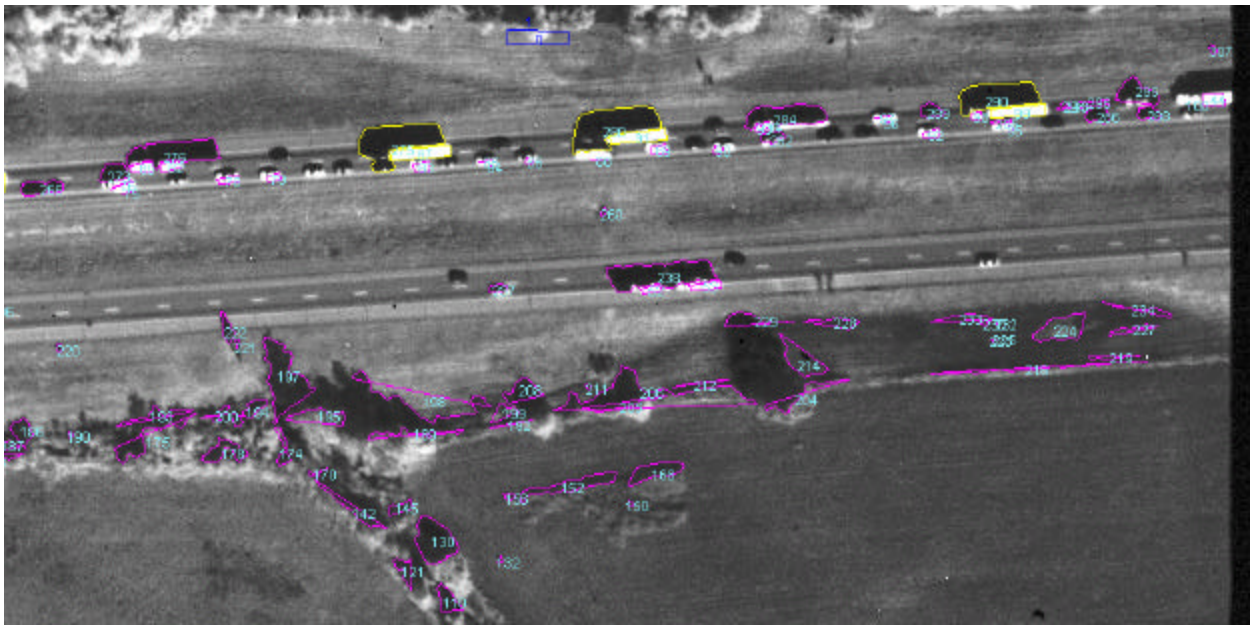
(a) Recognized polygons in Part I of the image strip



(b) Recognized polygons in Part II of the image strip



(c) Recognized polygons in Part III of the image strip



(d) Recognized polygons in Part IV of the image strip

Figure 3.18 Recognized trucks (yellow lines) in four parts of Figure 3.15

3.5 Test on Position and Velocity Calculation

For a feature that is fixed on the ground, two images are needed to calculate its 3D coordinates, though more photographs could improve coordinate accuracy. In the case of truck recognition and location some additional problems occur because the trucks move with varying velocity and direction. Neither the truck's velocity nor position are available to us. The two collinear equations from one image are

$$x = -f\frac{a_{11}(X-X_c)+a_{12}(Y-Y_c)+a_{13}(Z-Z_c)}{a_{31}(X-X_c)+a_{32}(Y-Y_c)+a_{33}(Z-Z_c)}$$

$$y = -f\frac{a_{21}(X-X_c)+a_{22}(Y-Y_c)+a_{23}(Z-Z_c)}{a_{31}(X-X_c)+a_{32}(Y-Y_c)+a_{33}(Z-Z_c)}. \qquad (3.16)$$

The exterior orientation parameters and interior orientation parameters are known from GPS and INS aboard the aircraft and system calibration (Li 1998). We have three unknowns (X,Y,Z) in two equations. In addition, a DTM is introduced to assistant in estimating 3D coordinates of the moving trucks. To estimate the approximate position of a truck, the calculation is performed as following:

1) Obtaining the minimum and maximum elevation values in the area of consideration,
2) Adding the known average truck height to these two elevations to get (Z_{min}, Z_{max}),
3) Using the image coordinates (x,y), (Z_{min},Z_{max}) and Equation 3.16 to compute two sets of 3-D coordinates (X_{min},Y_{min},Z_{min}) and (X_{max},Y_{max},Z_{max}),
4) Using (X_{min},Y_{min}) and (X_{max},Y_{max}) to define a bounding region in the DTM which should contain the trucks, and
5) Back-projecting the grid points of the DTM within the above region to the image space to get $(x,y)_{back}$ to obtain the 3-D coordinates (X,Y,Z) of the truck image image (x,y) as the grid point whose $(x,y)_{back}$ is closest to (x,y).

Assume that a truck moves with a fixed velocity within a short exposure interval (for example 7 seconds). Since the truck's elevation does not change appreciably during this interval, we use Z values from the DTM. At this time we manually recognized corresponding trucks in the images in light of the difficulty caused by sometimes varying velocity of the trucks. Efforts will be made in the future to automate this process. Equations from three images are:

$$x_0 = -f\frac{a_{11}(X_0-X_c)+a_{12}(Y_0-Y_c)+a_{13}(Z_0-Z_c)}{a_{31}(X_0-X_c)+a_{32}(Y_0-Y_c)+a_{33}(Z_0-Z_c)}$$

$$y_0 = -f\frac{a_{21}(X_0-X_c)+a_{22}(Y_0-Y_c)+a_{23}(Z_0-Z_c)}{a_{31}(X_0-X_c)+a_{32}(Y_0-Y_c)+a_{33}(Z_0-Z_c)}$$

$$x_1 = -f\frac{a'_{11}(X_0+(t_1-t_0)\times V_x-X'_c)+a'_{12}(Y_0+(t_1-t_0)\times V_y-Y'_c)+a'_{13}(Z_1-Z'_c)}{a'_{31}(X_0+(t_1-t_0)\times V_x-X'_c)+a'_{32}(Y_0+(t_1-t_0)\times V_y-Y'_c)+a'_{33}(Z_1-Z'_c)}$$

$$y_1 = -f\frac{a'_{21}(X_0+(t_1-t_0)\times V_x-X'_c)+a'_{22}(Y_0+(t_1-t_0)\times V_y-Y'_c)+a'_{23}(Z_1-Z'_c)}{a'_{31}(X_0+(t_1-t_0)\times V_x-X'_c)+a'_{32}(Y_0+(t_1-t_0)\times V_y-Y'_c)+a'_{33}(Z_1-Z'_c)}$$

$$x_2 = -f\frac{a''_{11}(X_0+(t_2-t_0)\times V_x-X''_c)+a''_{12}(Y_0+(t_2-t_0)\times V_y-Y''_c)+a''_{13}(Z_2-Z''_c)}{a''_{31}(X_0+(t_2-t_0)\times V_x-X''_c)+a''_{32}(Y_0+(t_2-t_0)\times V_y-Y''_c)+a''_{33}(Z_2-Z''_c)}$$

$$y_2 = -f\frac{a''_{21}(X_0+(t_2-t_0)\times V_x-X''_c)+a''_{22}(Y_0+(t_2-t_0)\times V_y-Y''_c)+a''_{23}(Z_2-Z''_c)}{a''_{31}(X_0+(t_2-t_0)\times V_x-X''_c)+a''_{32}(Y_0+(t_2-t_0)\times V_y-Y''_c)+a''_{33}(Z_2-Z''_c)} \quad (3.17)$$

In the above equations there are four unknowns (X_0,Y_0,V_x,V_y) in six equations, which could be solved by the Least Square Method (LSM). Actually a bundle adjustment is performed to estimate all the unknowns together. Table 3.10 gives exterior orientation parameters of three images used in our example.

		Photo (1185)	Photo (1187)	Photo (1189)
Exposure Center	X_c (m)	512538.071	512628.474	512739.333
	Y_c (m)	216637.932	216311.665	215916.059
	Z_c (m)	1364.956	1363.102	1362.703
Rotation Matrix	a_{11}	0.999288	0.999720	0.999847
	a_{12}	-0.016355	0.015189	0.003515
	a_{13}	-0.034003	-0.018134	0.017119
	a_{21}	0.014087	-0.015777	-0.002904
	a_{22}	0.997726	0.999340	0.999362
	a_{23}	-0.065914	-0.032730	-0.035598
	a_{31}	0.035004	0.017625	-0.017233
	a_{32}	0.065388	0.033006	0.035543
	a_{33}	0.997246	0.999300	0.999220

Table 3.10 Exterior orientation parameters of three images used

Table 3.11 illustrates the results of location and velocity estimation. The image coordinates given are those of the center of the top rectangle in each image. The 3-D coordinates are the truck location (center of the top) at the times of imaging.

It should be noted from Table 3.11 that the average speed of all four tracks is about 2 meters per second or 7.2 kilometers per hour. This is also verified by interpreting the three images in Figure 3.19 where vehicles in the direction of extracted trucks have small distances between them, indicating a traffic jam. In the opposite direction, distances between vehicles are significantly larger, signaling normal traffic.

			Photo (1185)	Photo (1187)	Photo (1189)	Velocity (m/s)
Truck 0	Image coordinates		(3208, 3572)	(2803, 2712)	(2295, 1417)	V_x = -2.005 V_y = -0.327
	3D coordinates	X (m)	512861.521	512848.114	512831.832	
		Y (m)	216079.517	216077.424	216074.883	
		Z (m)	327.500	327.556	327.836	
Truck 1	Image coordinates		(3259, 3568)	(2855, 2710)	(2346, 1414)	V_x = -2.005 V_y = -0.327
	3D coordinates	X (m)	512877.599	512864.102	512847.711	
		Y (m)	216080.780	216078.539	216075.818	
		Z (m)	327.629	327.219	327.667	
Truck 2	Image coordinates		(3261, 3580)	(2856, 2721)	(2348, 1425)	V_x = -2.001 V_y = -0.275
	3D coordinates	X (m)	512878.333	512864.767	512848.292	
		Y (m)	216076.670	216074.804	216072.538	
		Z (m)	327.110	326.728	327.556	
Truck 3	Image coordinates		(3209, 3582)	(2805, 2723)	(2295, 1429)	V_x = -1.990 V_y = -0.330
	3D coordinates	X (m)	512862.110	512848.513	512831.999	
		Y (m)	216076.133	216073.916	216071.224	
		Z (m)	327.260	327.468	327.815	
Average velocity						V_x = -2.003 V_y = -0.315

Table 3.11 Estimated truck location and velocity

To verify the estimated truck location and velocity, we choose one truck and take its positions at three imaging epochs and back project the truck onto three images as polygon 0, 1, and 2 respectively. In the first image (photo 11885) polygon 0 matches the truck. Polygon 1 and 2 match the truck in the second and third image respectively. The back projected polygons show the truck's trajectory also.

(a) Photo 1185 (b) Photo 1187 (c) Photo 1189

Figure 3.19 Back projection of one truck with velocity calculated at different times

4. Conclusions

Overall, we have made a significant progress in this project year, in both automatic feature extraction from AIMS imagery and in automatic object recognition using neural networks. The following conclusions can be drawn from this project year's results:

1) An integrated edge detector combining first and second order derivatives has been developed and validated using AIMS data,
2) Unique GPS/INS and DEM information has been used to assist in feature detection,
3) A set of tools such as shadow analysis, anti-parallel segments, morphological transformation, and snake deformation models were used to successfully extract road networks,
4) Object models, DEM, and photogrammetric constraints were applied in neural networks to produce robust matching results,
5) A two layer Hopfield neural network was applied in truck recognition to improve interlayer relationship (lines and polygons) and to enhance computational efficiency, and
6) A test on estimation of location and velocity of moving objects (trucks) from AIMS imagery is carried out. It is to note that this potential will open a very unique application in dynamic monitoring if the model can be improved.

With the above results, we are very confident that georeferenced imagery such as AIMS can be used to generate traditional mapping products such as line maps, orthophotos, DEM, and other digital databases much more *automatically and efficiently*. There are automatic operations that can only be implemented using such data sets. Furthermore, this research forms a basis for conducting similar important research using upcoming one-meter resolution satellite imagery.

5. Acknowledgments

The research team appreciates the Research Grant, AIMS data and assistance in GPS survey from the Center for Mapping. Discussions with Drs. John Bossler, Terry Caelli, Charles Toth, Dorota Grejner-Brzezinska and Mr. Panayiotis Zafiropoulos are very helpful and appreciated.

References

Ballard, D.H. and C.M. Brown 1982, Computer Vision, *Prentice-Hall, Inc*., New Jersey.

Canny, J. 1996, A Computational Approach to Edge Detection, *IEEE Trans. Pattern Anal. Machine Intell.* (PAMI), vol.8, No.6.

Duda, R.O. and P.E. Hart 1973, Pattern Classification and Scene Analysis, *Wiley*, New York.

Freeman, H. 1974, Computer Processing of Line Drawing Images, *Computer Surveys*, Vol.6, No.1, pp.57-98.

Gruen, A. and H. Li 1996, Linear Feature Extraction with LSB-Snakes from Multiple Images, *International Archives of Photogrammetry and Remote Sensing*, Vol.31, Part B3, pp.266-272.

Haralick, R.M. and L.G. Shapiro 1985, Image Segmentation Techniques, *Computer Vision, Graphics, Image Processing*, Vol.29, No.1, pp.100-132.

Kass, M., A. Witkin and D. Terzopoulos 1987, Snakes: Active Contour Models, *International Journal of Computer Vision*, Vol.1, No.4, pp.321-331.

Li, R. 1998, Potential of High-resolution Satellite Imagery for National Mapping Products, Journal of *Photogrammetric Engineering and Remote Sensing*, Vol.64, No.12, pp.1165-1169.

Li, R., W. Wang, and H.-Z. Tseng 1997, Geometric Constraints In Image Sequences and Neural Networks for object Recognition, Project Report, Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, 78p.

Li, R., G. Zhuo, J.-K. Liu, A. Gonzalez and Y. Felus 1998a. Coastal Mapping and Change Detection Using One-meter Resolution Satellite Imagery, Project Report, Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, 80p.

Li, R., W. Wang, and H.-Z. Tseng, 1998b, Object Recognition and Measurement from Mobile Mapping Image Sequences Using Hopfield Networks: Part I, *Proceedings of ASPRS/RT Annual Conference* 1998, CDROM.

Lin, W.-C., F.-Y. Liao, C.-K. Tsao and T. Lingutla 1991, A Hierarchical Multiple-View Approach to Three-Dimensional Object Recognition, *IEEE Transactions on Neural Networks*, Vol.2, No.1, pp.84-92.

Lu, Y., Jain, R.C. 1989, Behavior of Edges in Scale Space, *IEEE Trans. Pattern Anal. Machine Intell. (PAMI)*, Vol.11, pp.337-356.

Lu, Y., Jain, R.C. 1992, Reasoning About Edges in Scale Space, *IEEE Trans. Pattern Anal. Machine Intell.* (PAMI), Vol.14, pp.450-468.

Marr, D.C. and E. Hildreth 1980, Theory of Edge Detection, *Proc.Roy.Soc.Lond.B*, pp.187-217.

Marr, D. 1982, Vision, *Freeman*, New York.

Mckeown, D. M. 1990, Toward Automatic Cartographic Feature Extraction. In L.F. Pau editor, *Mapping and sptial Modeling for Navigation*, volume F 65 of NATO ASI Series, Springer-Verlag, Berlin, pp.149-180.

McKeown, D.M. et al. 1996, Research in the Automated Analysis of Remotely Sensed Imagery: 1994-1995, CMU-CS-96-101, Digital Mapping Laboratory, CMU, Pittsburgh.

Nasrabadi, N.M. and Y.C. Chang 1992, Hopfield Network for Stereo Vision Correspondence, *IEEE Transactions on Neural Networks*, Vol.3, No.1.

Pratt, W.K. 1991, Digital Image Processing, Second Edition, *Wiley-Interscience*.

Serra, J. 1982 & 1988, Image Analysis and Mathematical Morphology, *Academic Press*, London, Vol.1 and Vol.2.

Witkin, A.P. 1983, Scale-Space Filtering, Proceedings of *IJCAI*, pp.1019-1022.

Yuille, A.L. and T.Poggio 1983, Scaling theorems for zero-crossings, Massachusetts Inst. Technol., Cambridge, A.I. Memo 722.

Young, S.S. and P.D. Scott and M.N. Nasser 1997, Object Recognition Using Multilayer Hopfield Neural Network, *IEEE Trans. on Image Processing*, Vol.6, No.3, pp.357-371.