FEATURE EXTRACTION: A NEURAL NETWORK ORIENTED APPROACH

Yong-Jian Zheng
Institute for Photogrammetry and Remote Sensing, University Karlsruhe
Englerstrasse 7, D-7500 Karlsruhe 1, Germany
Email: zheng@ipf.bau-verm.uni-karlsruhe.de

Abstract

Extracting features from digital images is the first goal of almost all image understanding systems. It is also difficult to solve because of the presence of noises and various photometric anomalies. Another difficulty for it is the fact that features and objects are recognized by using not only the information contained in image data but also our a priori knowledge about the semantics of the world. Thus, a feature extraction system should be robust to reduce the influence of noises and flexible to integrate different levels of knowledge for a wide range of data. In this paper, a two-stage paradigm for feature extraction is proposed, based on our conjectures about human vision ability. It includes local feature grouping and new feature describing. Based on laws of perceptual grouping and neural network modeling, we develop a novel approach for feature grouping, which finds the partion of an image into so called feature-support regions. In order to give abstract descriptions to these regions, one needs *a priori* knowledge about their semantics to construct models. So we also discuss model driven methods for feature describing. To demonstrate our approach, we present its application in the limited domain of finding and describing straight lines in a digital image. This approach can be extended to extract other more complex symbolic image events like arcs, polylines, and polygons.

1 Introduction

Human vision is the ability to extract a variety of features and cues from images and to draw inferences from them. Realizing this ability is technically difficult even if we only focus our attention on the problem of feature extraction from digital images. One difficulty is the fact that the physical transformations from objects to images are degenerated due to a variety of confounding factors, including complex uncontrolled lighting, highlights and shadows, texture, occlusion, complex 3D shapes, and digitization effects. All of these make feature extraction to vary in quite unreliable and unpredictable ways.

Besides, many features are only perceived owing to the combination of weak evidence of several other features. The evidence may be so weak that each feature, if viewed in isolation, would be uninterpretable. Now the difficulty is how to discover and group those features at a lower level that may support the extraction of a new feature with a higher level of abstraction.

To undo some degeneracies in images and make feature extraction robust, we need knowledge about image events, objects in the world, and the imaging process. We need context-information and *a priori* models to guide feature searching and describing. The difficulty, then, is when and how to integrate the relevant knowledge and the context-information during feature extraction.

Due to these difficulties, many feature extraction methods fail to find most relevant features. They may find either too many or too few image events. They may provide just a little information about extracted features, namely feature properties, which may be required to support inference drawing. And, they may tell us too few about the quality, namely the accuracy and the reliability, of extracted features which is a very important information for top-down control over the knowledge based interpretation process.

Feature extraction is a multi-level process of abstraction and representation. At the lowest level of abstraction, numerical arrays of direct sensory data are given, including digital images and the results of other processes which produce point/pixel data in register with the sensory data. Now the goal is to discover image events which may represent some symbolic-semantic information and to describe them in a relevant way. This is the first level of abstraction. At the second level of abstraction, the image events extracted earlier are used as building elments to form new image events and structures with more abstraction. This discovery-description process can be repeated at a higher level to hypothesize scene and object parts. There is no doubt that two functions are required for building

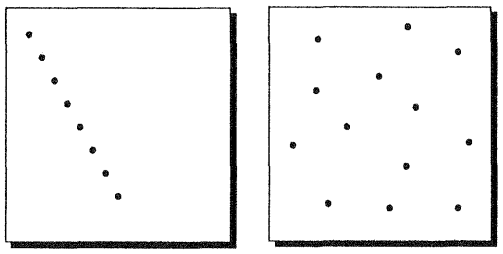


Figure 1: Two pictures containing a set of dots

a feature extraction system based on this bottom-up data-directed organization of interesting perceptual events: namely the function of grouping pixels or image events with weak evidence into new image events which would be better interpretable and the function of describing the new image events effectively to provide more information about their features. How to realize these two functions in a system is the objective which we are going to discuss.

In this paper, a two-stage paradigm for feature extraction from images is proposed. It includes two procedures of neural network based feature grouping and model driven feature describing and it takes into account the above mentioned troubles for feature extraction. In order to demonstrate this paradigm, we just look at the problem of extracting straight lines from image arrays. Based on a novel neural network model, we present a high-quality line finder which gives a complete description about geometric and photometric properties of extracted lines. The approach can be extended to find other more complex image events, including arcs, curves, polylines, and polygons.

2 A Two-Stage Paradigm

Image Understanding can be thought of as an inference process in which a description of the outside world is inferred from images of the world, having assistance of our *a priori* knowledge and experiences (ZHENG, 1992). Drawing inference from image data requires, first, the ability of discovering image events and representing their features in a relevant way, as mentioned earlier. Before we realize this ability technically, we should first know how an image event is perceived.

An image is a distribution of the luminance intercepted by the camera lens. Many factors, including the surface material, the atmospheric conditions, the light source, the ambientlight, the camera angle and characteristics etc., are confounded in the image and contribute to a single measurement, say the intensity of a pixel. The various factors cannot be separated, as long as they are not measured. So a single pixel can support one hypothesis only with very weak evidence and it can present some visual impressions only in combination with many other pixels. This can be made explicit by examining the two pictures illustrated in Figure 1. Here both pictures contain a set of dots as stimuli. However, only the left picture present a visual impression of a straight line. This suggests that image features are represented by a group of pixels and the first step in feature extracting is grouping such pixels into so called feature-support regions, based on our knowledge of what we want to extract.

For digital images, the situation is more complex, as they are corrupted by both discrete spatial sampling and intensity quantization, and there is stochastic component in image data. In this case, the evidence which can be given by a pixel is not only weak but also erroneous and unreliable. So, the second step in feature extraction is describing a group of noisy pixels obtained through grouping. This is an ill-posed problem as one can hypothesize an infinite number of different underlying descriptions (ZHENG, 1990). Of course, many of these descriptions are senseless. But the main question is how to find the description which we think of as the best one according to our *a priori* knowledge about what the group of noisy pixels should present. This means that a feature extraction system should be able to verify and describe feature-support regions using models and to give more comprehensive information about their features for subsequent inference and reasoning.

Obviously, features in images may have different degrees of abstraction and complexity. So, feature extraction requires a bottom-up hierarchy of the grouping-describing process, from low to higher abstraction.

3 Perceptual Grouping

The human vision is the only example for developing an artificial system to solve visual problems like feature extraction. However, our knowledge about the human visual system is very limited, no matter from physiological or psychological point of view.

One of the most obvious and interesting facts of human visual perception is the ability of the so called perceptual grouping which is based on the researches of the Gestalt psychologists. They argue that humans must be able to partion a scene into coherent, organized and independently recognizable entities or groups by using a set of generic criteria and Gestalt laws and the human visual system is very good at detecting

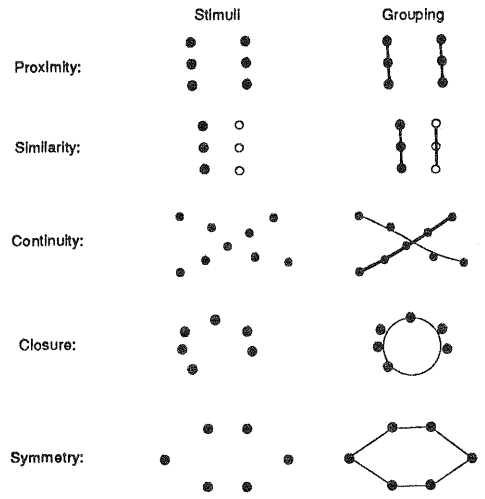


Figure 2: The Laws of perceptual grouping

geometric relationships such as collinearity, parallelism, connectivity, and repetitive patterns in an otherwise randomly distributed set of image events.

WERTHEIMER (1923) proposed one of the earliest and perhaps most acceptable sets of such laws, some of which can be roughly stated as follows (cf. Fig. 2):

- **The Law of Proximity**: the stimulus elements which are geometrically closer tend to be perceived as one entity.
- **The Law of Similarity**: the stimulus elements which have similar properties tend to be perceived as one entity.
- **The Law of Good Continuity**: the stimulus elements tend to form a group which minimizes a change or discontinuity.
- **The Law of Closure**: the stimulus elements tend to form complete figures which are *a priori* known.
- **The Law of Symmetry**: the stimulus elements tend to form complete figures which are symmetrical.
- **The Law of Simplicity**: the stimulus elements tend to form figures which require the least length for their description.

The laws of perceptual grouping provide a very important source of *a priori* knowledge to deal with noisy, incomplete, and fragmentary image information and have been therefore widely used for a variety of vision tasks (MEDIONI et al., 1984: MOHAN et al., 1989; BOLDT et al., 1989; KHAN et al., 1992).

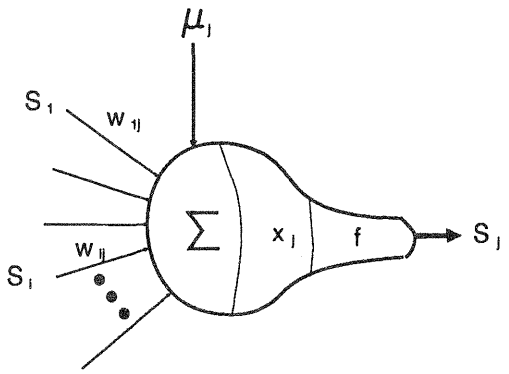


Figure 3: The McCulloch-Pitts Neuron

4 Neural Network Grouping

Humans seem to integrate the laws of perceptual grouping for aggregating image data in order to discover significant image events and cues. The main question is how to implement this ability effectively and to combine the results when different laws give different results. So, in this section, we look at this issue based on neural network modeling.

A neural network is a computational model that is a directed graph composed of nodes (sometimes referred to as units or neurons) and connections between the nodes (cf. ZEIDENBERG, 1990). With each node is associated a number, referred to as the node's activation. Similarly, with each connection in the network, a number is also associated, called its weight. The three main issues in neural network research are *network connection schemes, update rules*, and *learning rules*. For different tasks one should use different network models.

4.1 McCulloch-Pitts Neuron

We begin with the McCulloch-Pitts neuron (cf. Fig. 3) which is a basic building element of many neural networks. As shown in Figure 3, the activity x_j of a neuron is the sum of inputs that arrive via weighted pathways. The input from a particular pathway is an incoming signal S_i multiplied by the weight w_{ij} of that pathway. These weighted inputs are summed independently:

$$x_j = \sum_i S_i w_{ij} + \mu_j = \mathbf{S} \cdot \mathbf{w}_j + \mu_j, \qquad (1)$$

where μ_j is a bias term, which is formally equivalent to the negative of a threshold of the outgoing signal function. The outgoing signal $S_j = f(x_j)$ is typically a nonlinear function (binary, sigmoid, or threshold-

linear) of the activity x_j in that neuron.

The dot product of the the signal vector **S** times the stored weight vector $\mathbf{w}_j$ in (1) is a measure of the similarity of these two vectors. This means that the McCulloch-Pitts neuron receives the incoming pattern **S**, compares it with the pattern $\mathbf{w}_j$ stored in memory, and reacts to their similarity. Of course, there are also some other types of measurement of similarity between patterns. Probably the best known measure of similarity is the Euclidean distance, given by

$$\sqrt{\sum_i (S_i - w_{ij})^2}. \quad (2)$$

This generalizes to the Minkowski metric, given by

$$\left[\sum_i (S_i - w_{ij})^n\right]^{\frac{1}{n}}. \quad (3)$$

In fuzzy logic, two scalars' similarity is given by

$$max[min(S_i, w_{ij}), max(1 - S_i, 1 - w_{ij})], \quad (4)$$

where S_i and w_{ij} should be drawn from interval between 0 and 1 inclusive.

4.2 Learning

Using some McCulloch-Pitts neurons, one can build different types of neural networks for different purposes. Most work in neural networks involves learning. So the goal of most neural network models is to learn relationships between stimuli, though many other things can also be learned, such as the structure of the network, the activation functions, even the learning rules themselves. For the task of feature grouping during feature extraction process, the main goal is to design a learning system which compute a classification, where a large set of input patterns is mapped onto a relatively small set of output patterns, which represent sets into which the input patterns are classified.

When developing a neural network to perform a particular pattern-classification operation, we typically proceed by gathering a set of exemplars, or training patterns, then using these exemplars to train the system by adjusting weights on the basis of the difference between the values of output units and the desired pattern. This kind of learning is referred to *supervised learning.*

Another important kind of learning is so called *unsupervised learning* which occurs without a teacher. Such a learning algorithm learns to classify the input

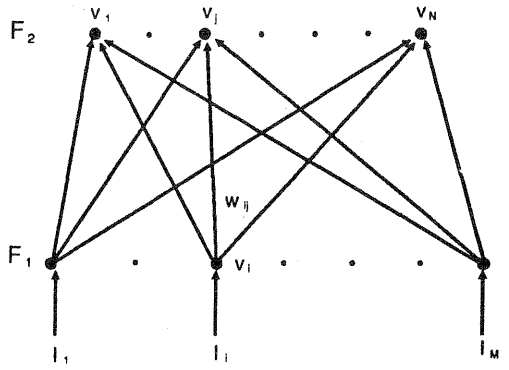


Figure 4: The competitive learning architecture

sets without being told anything. It does this clustering solely on the basis of the intrinsic statistical properties of the set of inputs. This property is just that what we want to perform a grouping operation before features can be described.

4.3 Competitive Learning

A mechanism of fundamental importance in unsupervised learning is described by the phrase *competitive learning*, which was developed in the early 1970s by contributions of MALSBURG, GROSSBERG, AMARI, and KOHONEN (cf. CARPENTER and GROSSBERG, 1988). Its main principle can be shown by using the following simpler mathematical formalism (cf. Figure 4).

There is a two-layer system of M input neurons ($\mathbf{F}_1$) and N output neurons ($\mathbf{F}_2$). These two layer of neurons are fully connected by using the weight $w_{ij}, i = 1, ..., M;\ j = 1, ..., N$. Now, let I_i denote the input to the i^{th} node v_i of $\mathbf{F}_1$, $i = 1, ..., M$, and let

$$x_i = \frac{I_i}{\sum_i I_i} \quad (5)$$

be the normalized activity of v_i in response to the input pattern $\mathbf{I} = (I_1, I_2, ..., I_M)$. The output signal S_i of v_i, as mentioned earlier, is usually a nonlinear function of x_i and it, for simplicity, can be assumed to be equal x_i.

Now, each neuron v_j of $\mathbf{F}_2$, $j = 1, ..., N$, compares the incoming pattern $\mathbf{S} = (S_1, S_2, ..., S_M)$ with the stored weight vector $\mathbf{w}_j = (w_{1j}, w_{2j}, ..., w_{Mj})$ by using a measure of similarity mentioned above and gives a activity x_j like (1). The neuron v_k with the maximum activity $x_k = \max(x_j)$ is selected (winner take all). This is the neuron whose weight vector is most similar to the input vector. This weight vector is then

adjusted to be even more similar to the incoming vector by the rule

$$\frac{dw_{ik}}{dt} = \gamma x_k(-w_{ik} + x_i), \tag{6}$$

where γ is a constant referred to the learning rate. In this way, the network illustrated in Figure 4 can be trained to classify input patterns **I** presented to $\mathbf{F}_1$ into mutually exclusive recognition categories separated by sharp categorical boundaries.

It has been, however, mathematically proved that such learning process stabilizes only if the input patterns form not too many clusters, relative to the number of coding nodes in $\mathbf{F}_2$ (GROSSBERG, 1976). A competitive learning model does not always learn a temporally stable code in response to an arbitrary input environment. Certain instabilities may arise in the competitive systems such that different nodes might respond to the same input pattern on different occasions. Moreover, later learning can wash away earlier learning if the environment is not statically stationary or if novel inputs arise. All of these suggest that the competitive learning can not deal with the so called stability-plasticity dilemma (CARPENTER and GROSSBERG, 1988).

4.4 The Stability-Plasticity Dilemma

To deal with the stability-plasticity dilemma, we need systems which can remain plastic, or adaptive, in response to significant events and yet remain stable in response to irrelevant events, and which can preserve its previously learned knowledge about group properties while continuing to learn new incoming patterns. The stability-plasticity dilemma, faced by all intelligent systems capable of autonomously adapting in real time to unexpected changes in their world, can be solved based on the so called *adaptive resonance theory* (ART) developed by GROSSBERG (1976). A key idea to solving this problem is to add a feedback mechanism between the competitive layer $\mathbf{F}_2$ and the input layer $\mathbf{F}_1$. This mechanism facilitates the learning of new information without destroying old information, automatic switching between stable and plastic modes, and stabilization of the encoding of the classes done by the nodes.

A simplified way to implement this idea is to make a vigilance test of similarity before the weight vector is adjusted. Suppose that an input pattern **I** activates $\mathbf{F}_1$. Let $\mathbf{F}_1$ in turn activate the node, or hypothesis, v_k at $\mathbf{F}_2$ which has the maxmum activity and whose weight vector is therefore most similar to **I**. Now a matching threshold ϱ called vigilance is given. This threshold determines how close a new input pattern must be to a stored exemplar to be considered similar. If a bad match takes place, then a reset burst is triggered. This reset burst shuts off the node v_k for the remainder of the weight adapting cycle and **I** is stored as the weight vector of a previously uncommitted node at $\mathbf{F}_2$. If a good match takes place, the weight vector of v_k is adapted by the rule

$$\frac{dw_{ik}}{dt} = \frac{1}{n_k + 1}(I_i - w_{ik}), \tag{7}$$

where n_k denotes how many times the node v_k has been adapted till now.

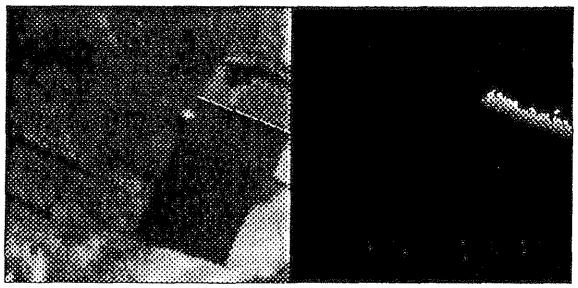
5 A Novel Line Finder

Feature extraction, as mentioned earlier, is a multilevel process of abstraction and representation, from image representation (purely numeric) to object or object-class models (highly abstracted). At the lowest level of feature extraction, potentially useful image events, such as homogeneous regions (collections of contiguous image data points with similar properties), lines, and curves, are extracted from the image data. These image events can then be used as building elements to form more complex features like polylines and polygons, in a bottom-up abstraction hierarchy. In this section we only pay attention to finding and describing lines in the image data. We want to demonstrate our two-stage paradigm for feature extraction by presenting its application in this limited domain.

5.1 Discovering Line Context

To facilitate the analysis let us first look at an image illustrated in Figure 5a and try to find lines in it. The first question we are facing is what a line means in a numerical array of intensities. Actually, a line is just an abstraction. It is a visual impression produced by a group of pixels and each pixel gives only a very weak evidence for building this impression, even if there were no stochastic component in this pixel. This is quite intuitive if we focus our attention on a region near the ridge of the roof (cf. the white line in Figure. 5a) and extract this region from image (cf. Figure 5b). It is clear that the ridge of the roof in the image is only perceived owing to the combination of weak evidence of all pixels in this so called line support region (HANSON and RISEMAN, 1987). So, a reasonable step for line finding is the perceptual organization of image pixels into a *supporting line context* prior to making any decisions about any potential underlying structures.

The perceptual laws are general grouping criteria during feature extraction. Among them, *proximity*, *similarity* and *continuity* can be utilized to aggregate image pixels into line support regions. As showed



a | b

Figure 5: A supporting line context

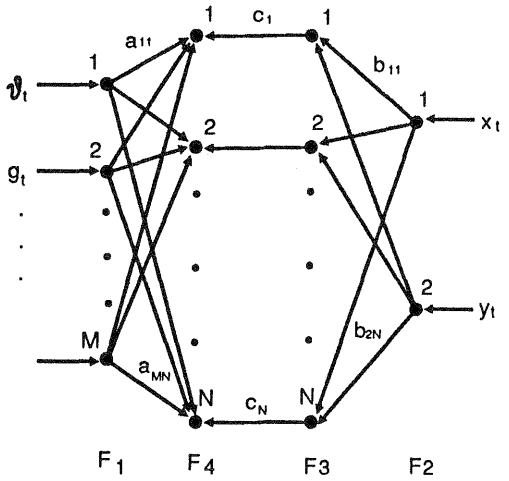


Figure 6: A novel neural network for line grouping

in Figure 2, the performance of proximity grouping depends on the spatial location of pixels. Those pixels are grouped which are closer and lie on the line. Connectivity is another important law for grouping and its performance depends also on the spatial location of pixels. Those pixels are grouped which are connected to each other on a straight line. On the contrary, the performance of similarity depends on local radiometric properties of pixels. Those pixels are grouped which are similar, for instance, in intensity, gradient magnitude and orientation.

Now, the main goal is to integrate these grouping criteria for an effective implementation and to combine the results when different criteria give different results. For this purpose, a novel neural network has been developed. As showed in Figure 6, the network has four layers denoted by $\mathbf{F}_i, i = 1, 2, 3, 4$. The layer $\mathbf{F}_1$ has M neurons and it receives the input vector $\mathbf{I}_1 = (\theta_i, g_i, ...)$ containing gradient orientation θ_i, gradient magnitude g_i, and other local radiometric properties of the i^{th} pixel. $\mathbf{I}_2 = (x_i, y_i)$ is the second input vector containing the coordinates of the same pixel and it is presented to the layer $\mathbf{F}_2$. The layer $\mathbf{F}_4$ is the output layer containing N neurons and $\mathbf{F}_3$ is a hidden layer which contains also N neurons. These layers are connected by the weight $a_{ij}, i = 1, ..., M, j = 1, ..., N$, between $\mathbf{F}_1$ and $\mathbf{F}_4$, $b_{ij}, i = 1, 2, j = 1, ..., N$, between $\mathbf{F}_2$ and $\mathbf{F}_3$, and $c_i, i = 1, ..., N$, between $\mathbf{F}_3$ and $\mathbf{F}_4$.

To train this network to aggregate pixels into line support regions, we apply all pixels in an image whose gradient magnitudes are greater than a threshold as input data. For simplicity, let $\mathbf{I}_1$ only contain the gradient orientation θ_i of the i^{th} pixel. For the first inputs $\mathbf{I}_1 = (\theta_1)$ and $\mathbf{I}_2 = (x_1, y_1)$, the first node v_1 at $\mathbf{F}_4$ is chosen and the weights which are referred to the direct and indirect connections between v_1 and other nodes at $\mathbf{F}_1$, $\mathbf{F}_2$ and $\mathbf{F}_3$ are adapted by the rules

$$a_{11} = \theta_1, \; c_1 = x_1 b_{11} + y_1 b_{21}, \\ b_{11} = \cos a_{11}, \; b_{21} = \sin a_{11}, \tag{8}$$

and the adapting number n_1 of v_1 is set to 1. For the t^{th} inputs $\mathbf{I}_1 = (\theta_t)$ and $\mathbf{I}_2 = (x_t, y_t)$, the input to the i^{th} node of $\mathbf{F}_3$ equals $s_i = x_t b_{1i} + y_t b_{2i}, i = 1, ..., N$ which is, for the sake of convenience, also its output signal. It is clear that s_i is just a matching score for the similarity between inputs and stored weights. For the i^{th} node of $\mathbf{F}_4$, the situation is more complex. It has the input $\theta_t - a_{1i}$ from $\mathbf{F}_1$ and the input $s_i - c_i$ from $\mathbf{F}_3$, $i = 1, ..., N$. Both inputs can be normalized by using

$$P_{ai} = \exp\left[-\frac{(\theta_t - a_{1i})^2}{2\sigma_a^2}\right], \; P_{ci} = \exp\left[-\frac{(s_i - c_i)^2}{2\sigma_c^2}\right], \tag{9}$$

where σ_a and σ_c are tow normalizing constants, and P_{ai} and P_{ci} can be thought of as two matching scores for the two inputs from $\mathbf{F}_1$ and $\mathbf{F}_3$. P_{ai} gives a measure to the performance of similarity grouping, while P_{ci} gives a measure to the performance of proximity grouping. Now, all nodes of $\mathbf{F}_4$ compete to recognize features in the input layers. Here the main question is how to measure the match of the i^{th} node of $\mathbf{F}_4$ using a matching score P_i. This is a problem of drawing inference based on P_{ai} and P_{ci}. When calculated using probability theory, the matching score P_i can be derived based on *Bayes' rule*:

$$P_i = P(P_{ai}, P_{ci}) = P(P_{ai} \mid P_{ci}) P(P_{ci}) \tag{10}$$

Based on *fuzzy logic*, the matching score P_i can be calculated as follows:

$$P_i = P(P_{ai}, P_{ci}) = \min(P_{ai}, P_{ci}). \tag{11}$$

Now, for the t^{th} pixel with the inputs $\mathbf{I}_1 = (\theta_t)$ and $\mathbf{I}_2 = (x_t, y_t)$, only those nodes of $\mathbf{F}_4$, which have been triggered by the neighbor pixels of the t^{th} pixel during the last learning, compete with each other. After

competition, the k^{th} node of $\mathbf{F}_4$ which has the largest matching score P_k is chosen. If P_k is greater than a vigilance threshold ϱ, the winning node v_k of $\mathbf{F}_4$ then triggers associative pattern learning within the weights which sent inputs to this node. The learning rules are written as follows:

$$\frac{da_{1k}}{dt} = \frac{1}{n_k+1}(\theta_t - a_{1k}),$$
$$\frac{dc_k}{dt} = \frac{b_{1k}}{n_k+1}(x_t - x^*) + \frac{b_{2k}}{n_k+1}(y_t - y^*),$$
$$\frac{dn_k}{dt} = 1,\ b_{1k} = \cos a_{1k},\ b_{2k} = \sin a_{1k}, \quad (12)$$

where the coordinates (x^*, y^*) can be computed using

$$\begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} b_{2k}^2 & -b_{1k}b_{2k} \\ -b_{1k}b_{2k} & b_{1k}^2 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} c_k b_{1k} \\ c_k b_{2k} \end{bmatrix}, \quad (13)$$

and their meaning will be given later. Otherwise, if P_k is less than ϱ, a previously uncommitted node v_j of $\mathbf{F}_4$, whose adapting number is zero, is selected. Its weights are adapted according to the following rules:

$$a_{1j} = \theta_t,\ c_j = x_t b_{1j} + y_t b_{2j},$$
$$b_{1j} = \cos a_{1j},\ b_{2j} = \sin a_{1j}, \quad (14)$$

and n_j is set to 1.

The learning process just stated repeats itself automatically at a very fast rate till each pixel in the image is presented to the net more than one time. After that, the net can group pixels into line support regions. Figure 5 shows, for instance, a typical line support region containing those pixels which trigger the activity of the same node v_* at $\mathbf{F}_4$. All of these pixels have a similar gradient orientation and lie close to a hypothetic straight line which has been learned by the net during the learning process and can be represented by using the equation

$$x \cos a_{1*} + y \sin a_{1*} - c_* = 0, \quad (15)$$

where a_{1*} and c_* are stored in the so called long-term memory storage (adaptive weights) of the net. Now, we come back to the meaning of (x^*, y^*) (cf. (13)). It can be proved that (x^*, y^*) are just the coordinates of the projection of the pixel (x_t, y_t) onto the hypothetic straight line which is represented by the node v_k.

5.2 Model Driven Line Description

After grouping process, some line support regions in the image are extracted and each of them may hide a potential line structure. How to make this line structure explicit is thus the main issue of this section. A line, as mentioned above, is just a visual impression produced by a line support region. To characterize this impression quantitatively, models for what we want to extract are required. These models can then be used to fit line support regions. A good fit suggests a good line description. So, the tasks of line description include *model generating, parameter estimation*, and *quality description*.

name	line i
is-a	line
type	I
length	83.0 pixel
end points	(79.2, 162.7), (109.8, 239.8)
θ	2.8 radian
ρ	−13.7 pixel
ϵ	2.0
α	77.1 intensity level
β	19.2 intensity level
σ_0	8.2 intensity level
σ_θ	0.002 radian
σ_ρ	0.4 pixel
σ_ϵ	0.1
σ_α	2.5 intensity level
σ_β	1.7 intensity level

Table 1: A line frame

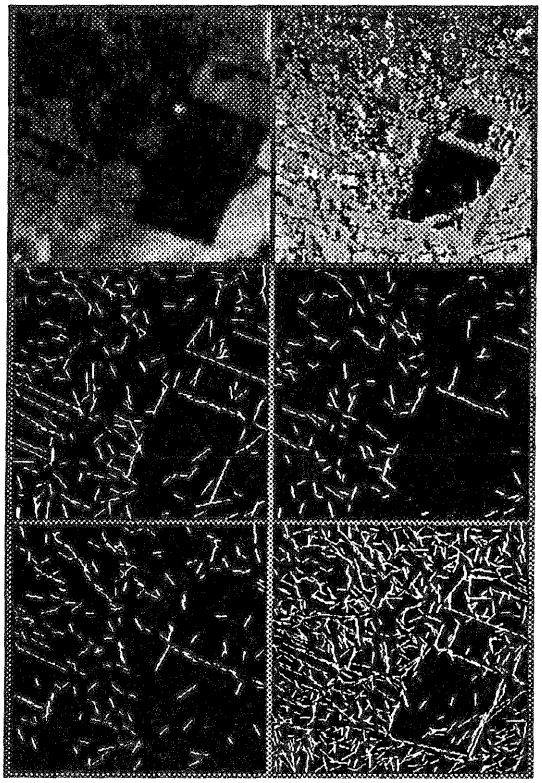
There are many ways to generate a line model which can be implicit or explicit, analytical or functional. For the sake of convenience, we define the following models to describe four line types:

Model

$$\mathbf{I}: \quad I(x,y) = \alpha\left[1 + \exp(-\tfrac{x\cos\theta + y\sin\theta - \rho}{\epsilon})\right]^{-1} + \beta$$
$$\mathbf{II}: \quad I(x,y) = \alpha \exp\left[-\tfrac{1}{2}(\tfrac{x\cos\theta + y\sin\theta - \rho - 1}{\epsilon})^2\right] + \beta$$
$$\mathbf{III}: \quad I(x,y) = -\alpha \exp\left[-\tfrac{1}{2}(\tfrac{x\cos\theta + y\sin\theta - \rho + 1}{\epsilon})^2\right] + \beta$$
$$\mathbf{IV}: \quad I(x,y) = \alpha(x\cos\theta + y\sin\theta - \rho) + \beta,\ \beta = \tilde{I}$$

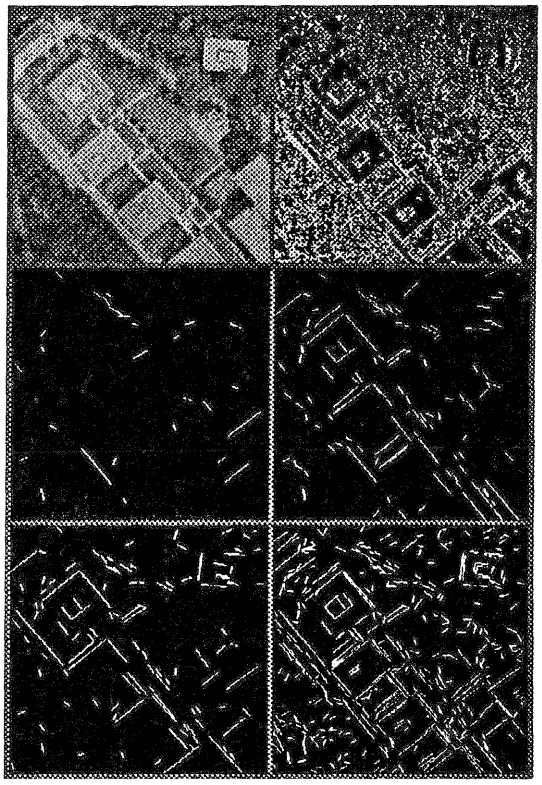
where $I(x,y)$ denotes the intensity of a pixel (x,y), $\tilde{I}$ denotes the average intensity of all pixels within a line support region, weighted by their gradient magnitude, and $\Omega = (\theta, \rho, \epsilon, \alpha, \beta)$ is a set of parameters describing geometric and radiometric aspects of the line's behavior.

Given a line support region $\mathbf{R} = I_i(x_i, y_i), i = 1, ..., n$ and a line model $I(x,y) = f(x,y,\theta,\rho,\epsilon,\alpha,\beta)$, it is not difficult to estimate the unknown parameters Ω based on, i.e., the least squares estimation technique. This technique, however, is very sensitive to the presence of outliers, i.e., to intensities with very large deviations from the underlying surface. For reducing the effect of outliers on the estimates, we need new methods known as robust estimators (HUBER, 1981). Here the parameters Ω are estimated by minimizing a penalty function of the residuals ,i.e., $\sum_i^r \varphi(r_i)$, where r_i denotes the residual. This is a minimization problem which can be solved as iteratively reweighted least squares with the definition of the weights depending on $\varphi(r_i)$



a	b
c	d
e	f

Figure 7: The line extraction from the first image



a	b
c	d
e	f

Figure 8: The line extraction from the second image

(ZHENG and HAHN, 1990). The weight function $\varphi(r_i)$ can be, for instance, the Cauchy function defined by

$$\varphi(r_i) = \frac{1}{1 + (\frac{r_i}{\sigma_r})^2},$$

where σ_r is the standard deviation of the fit and can also be estimated.

For the inference and reasoning process of a higher abstraction level during feature extraction, it is desired to know something about the quality of features extracted from low level processing. Many feature extraction algorithms, however, lack a detailed and comprehensive description of extracted features. Actually, after parameter estimation, it is also possible to estimate the *posteriori* accuracy of the estimation. As a measure for the global fittingness of data to a model, the estimate

$$\sigma_0 = \frac{\sum_i^r w_i r_i^2}{n - u} \qquad (16)$$

can be used, where w_i is the weight and u is the number of the unknown parameters. Besides, the *posteriori* accuracies of the parameters in Ω can also be estimated and they are denoted by $(\sigma_\theta, \sigma_\rho, \sigma_\epsilon, \sigma_\alpha, \sigma_\beta)$.

Now, the line support region illustrated in Figure 5 is used to estimate the parameters in model I. The results can be stored in a form of knowledge representation known as *frame* (MINSKY, 1975) (cf. Tabular 1). It is to say that the accuracy of the line position is about 0.4 pixel (subpixel accuracy) and the accuracy of the line orientation is about 0.002 radian. The line estimated in this way is shown in Figure 5a using the white line.

6 Experimental Results

The algorithm described in the previous sections was applied to two aerial images. Due to the limited space of this paper, many interesting intermediate results can not be discussed in this section. Here we just illustrate the lines found by the algorithm.

The first image (cf. Figure 7a) shows an aerial scene with a house and fence on rolling terrain. Due to shadows and poor contrast, the roof borders get fragmented. The image was used to train the net illustrated in Figure 6. After training, the net can automatically group image pixels into line support regions

(cf. Figure 7b) and these regions then can be used to estimate the model parameters Ω. Figure 7c-f show the four types of the line segments detected in the image, which are longer than 5 pixels. All detected line segments, as mentioned above, are described comprehensively by a set of attributes (cf. Table 1) and they can be used as building elements in the next two-stage feature extraction process to find and detect more complex image events and structures like parallels, T and U structures, rectangles, and polygons. Figure 8a shows the second aerial image with buildings, trees and roads. Figure 8b shows the line support regions grouped by the net. Figure 8c-f show the line segments of the four different types detected in the image, which are longer than 5 pixels.

7 Conclusion

This paper has advanced the beginnings of a theory of feature extraction with the following main elements: 1) a two-stage paradigm of grouping and description for the hierarchic feature extraction process; 2) a neural network based approach for grouping local features and integrating the perceptual laws effectively; 3) methods for model driven feature description; and 4) a novel line finder as the application of the theory. It is important to understand that the feasibility of the theory consists in its applications to finding other more complex image events and structures. So, among the goals of future work will be the application of the theory to developing algorithms for extracting other features and the further analysis of the behavior of neural networks designed for feature extraction.

References

[1] Boldt, M., R. Weiss, and E. Riseman, 1989. Token-Based Extraction of Straight Lines, *IEEE Tran. on Systems, Man, and Cybernetics*, Vol. 19, No. 6, pp. 1581-1594.

[2] Carpenter, G. A. and S. Grossberg, 1988. The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network, *Computer*, Vol. 21, pp. 77-88.

[3] Grossberg, S., 1976. Adaptive Pattern Classification and Universal Recoding, I: Parallel Development and Coding of Neural Feature Detectors, *Biological Cybernetics*, Vol. 23, pp. 121-134.

[4] Hanson, A. and E. Riseman, 1987. The VISIONS Image Understanding System, *Advances in Computer Vision*, C. Brown (Ed.), Erlbaum Press, USA.

[5] Huber, P. J., 1981. *Robust Statistics*, New York: Wiley, 1981.

[6] Khan, G. N. and D. F. Gillies, 1992. Extracting contours by perceptual grouping, *Image and Vision Computing*, Vol. 10, No. 2, pp. 77-88.

[7] Medioni, G. and R. Nevatia, 1984. Matching Images Using Linear Features, *IEEE PAMI*, Vol. 6, No. 6, pp. 675-685.

[8] Minsky, M. L., 1975. A Framework for Representing Knowledge, *The Psychology of Computer Vision*, P. H. Winston (Ed.), New York: McGraw-Hill.

[9] Mohan, R. and R. Nevatia, 1989, Using Perceptual Organization to Extract 3-D Structures, *IEEE PAMI*, Vol. 11, No. 11, pp. 1121-1139.

[10] Wertheimer, M., 1923. Laws of Organization in Perceptual Forms, *Psychologische Forschung*, vol. 4, pp. 301-350.

[11] Zeidenberg, M., 1990. *Neural Network Models in Artificial Intelligence* , Ellis Horwood Limited, England.

[12] Zheng, Y.-J., 1990. Inverse und schlecht gestellte Probleme in der photogrammetrischen Objekt Rekonstruktion, *Inaugural-Dissertation*, Fakultät für Vermessungswesen, Universität Stuttgart, Stuttgart.

[13] Zheng, Y.-J. and M. Hahn, 1990. Surface Reconstruction from Images in the Presence of Discontinuities, Occlusions and Deformations, *Intern. Archives of Photogrammetry and Remote Sensing*, vol. 28, part 3/2, pp. 1121-1144, Wuhan.

[14] Zheng, Y.-J., 1992. Inductive Inference and Inverse Problems in Computer Vision, *Robust Computer Vision*, Förstner/Ruwiedel (Eds.), Karlsruhe: Wichmann, pp. 320-350.