A Connectionist Algorithm for Image Segmentation

TR89-008

February 1989

Cheng-Hong Hsieh

The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175

A CONNECTIONIST ALGORITHM FOR IMAGE SEGMENTATION

by

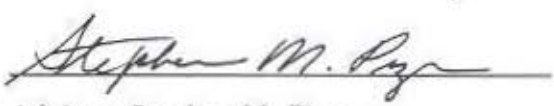
Cheng-Hong Hsieh

A dissertation submitted to the faculty of The University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

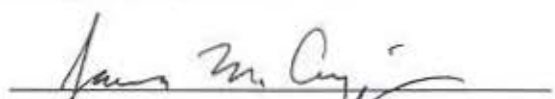
Chapel Hill

1989

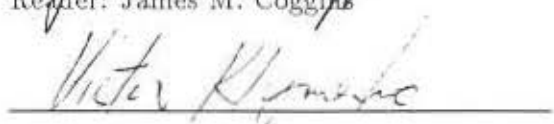
Approved by:



Advisor: Stephen M. Pizer



Reader: James M. Coggins



Reader: Victor Klymenko

CHENG-HONG HSIEH. A Connectionist Algorithm for Image Segmentation (Under the direction of STEPHEN M. PIZER.)

Abstract

An edge-based segmentation algorithm based on the knowledge in human vision was developed. The research followed Grossberg's *boundary contour system* and developed a parallel distributive algorithm which consists of multiple processing stages — mainly anisotropic edge filtering, corner detection, and spatial coherence check. The two-dimensional input information is processed in parallel within each stage and pipelined among stages. Within each stage, local operations are performed at each pixel. The application of this algorithm to many test patterns shows that the algorithm gives good segmentation and behaves reasonably well against random noise. A multiscale mechanism in the algorithm can segment an object into contours at different levels of detail.

The algorithm was compared with an approximation of Grossberg's *boundary contour system*. Both algorithms gave reasonable performance for segmentation. The differences lie in the level of image dependency of the configuration parameters of the algorithm. Also, the way random noise affects the algorithm was compared with the way it affects human object detection. Data obtained from psychophysical experiments and from application of the algorithm show a similar trend.

Acknowledgements

Studying for a graduate degree in the US was a dream too far for me to reach; with the help of the mighty God and a group of good people, the dream is coming true.

A special thank you goes to my advisor and chairman, Steve Pizer. Without his help and guidance, none of this research would be possible; without his direction as a teacher, this student would not have turned into a professional.

I owe thanks to my other committee members: Vic Klymenko for his helping me understanding human vision and advising on the design of the psychophysical experiment, James Coggins for his helpful suggestions in computer vision, Douglas Kelly for introducing me some mathematics of neural networks, and Kye Hedlund for encouraging me to pursue the connectionist architecture.

I want to express my appreciation to Prof. Jan Koenderink and Prof. Stephen Grossberg. Their kind encouragement and directions on research in vision is unforgettable.

I thank Gene Johnston and Bonnie Yankaskas for their useful opinions on performing the observer experiment, and John Gauch and Bill Oliver for their suggestions on text editing.

I am indebted to my government, the government of the Republic of China. It was the Government Scholarship for Studying Abroad which made my trip to the US possible.

Lastly, a thousand thanks to my wife, Hsilin. Her love and encouragement supported me to complete this dissertation, and thinking of my parents, who always put the future of their children before their own interests, I dedicate this dissertation to my wife, Hsilin, and my parents, Iu-Wei and Tsai-Fong.

(The partial support of NIH grants number R01 CA39059 and P01 CA47982 is gratefully acknowledged.)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Overview

The goal of this research is to develop an effective segmentation algorithm for a real-time vision machine. Since biological visual systems are generally far more powerful than conventional vision algorithms and there is convincing evidence that nature has evolved a preattentive (knowledge-free), stable, and highly-parallel early visual system, the design of the algorithm is in part based on our knowledge of the biological visual system. It is hoped that this research may not only help us to design an effective computer vision algorithm but also provide deeper understanding of the working principles of vision.

An organism (or a robot) can be viewed as performing a cycle of perception and action. Vision, associating the visual input with stored knowledge and thus enabling the organism to effectively interact with its environment, plays an important role. What are the properties of visual input and stored knowledge? What is the required processing speed? What algorithms and architectures can provide the necessary performance? This chapter provides several insights on these questions.

The chapter first describes the role of image segmentation in vision and then elaborates on my approach to the problem of segmentation. A summary of my thesis then follows. The chapter concludes with a list of the main contributions of this research.

1.1 The Problem of Image Segmentation

Image segmentation is the separation of the image into regions. The properties separating regions can be, among others, gray level, color, texture, motion, or depth. This section first argues that image segmentation should precede object recognition. Then it briefly discusses the required speed and possible objective of segmentation. Finally the section presents a more specific definition of image segmentation.

1.1.1 Properties of Segmentation

Image segmentation is in some form essential for vision. This point can be supported from two perspectives: one is from the need for information reduction and the other from the functional analysis of the visual system. In regard to the former, the information input rate to the visual system is tremendous. There are about 2×10^8 receptors in a retina. Assuming that the dynamic range of the receptor output is 5 bits and the temporal resolution is 100 ms, the information input rate is on the order of 10^{10} bits per second. The visual system can hardly respond to the information at the pixel (or receptor) level because of the resulting requirement of huge memory and processing power. Physiologically, the fact that the neurons later in the neural path tend to respond to more abstract stimuli also confirms this point.

From the point of view of visual system function, only groups of pixels, with each group indicating an object, are important because the visual system is to decide *what* is *where* in the environment. To decide *what* an object is, if segmentation does not precede recognition, the required information for recognition, like shape, shade, and spatial context, is unavailable. To decide *where* the object is, segmentation must be done first; otherwise the important information of an object's relative position and motion cannot be obtained.

Segmentation in biological visual system appears to be mainly preattentive and data-driven. Although past experience may affect segmentation through expectation, the strict time constraint — psychophysical evidence indicates that recognition requires about only 200 milliseconds — implies that a fast segmentation of image into regions indicating possible objects is necessary. Since most objects differ substantially in appearance from their environment, it is reasonable to expect knowledge-free procedures to yield reasonable segmentation in many instances. When the region is unclear or ambiguous, does expectation derived from contextual information help. Moreover, there is no evidence for how knowledge can affect segmentation. As indicated by Charles Gross's famous monkey-paw detector in the inferotemporal cortex [Gross et al, 1972], object recognition is performed by the neural layers which do not map to the retina topographically. It is difficult to imagine how a knowledge atom in the later visual path extends its axon to affect the earlier segmentation process. More probably, the visual system segments an image first and then the shape, shade, depth, motion, and texture of a selected object are delivered

to higher vision for recognition.

1.1.2 Properties of Objects in Our Visual World

The visual system has as a major function the detection of the objects in the environment, so it has been highly adapted to our physical world. The regularity of our visual environment provides important clues for understanding vision. What are the important characteristics of our visual world? Most importantly, an object usually has uniform surface properties like color, grey-scale intensity, or texture, which differs from the object's environment. Thus a filter, sensitive to the difference of that surface property across a surface boundary, can be used to detect the object. Other differences such as relative motion and depth also help segmentation but are not considered in this research.

The following observations concerning object contours in a two-dimensional image may help in detecting the real boundary of an object:

1. The bounding contour is usually closed.
2. Occlusion is ubiquitous because of the 3d to 2d projection.
3. Corners may be parts of an object and can be formed by occlusion.
4. The boundary between two corners is usually a smooth curve.
5. An object may look like a small blob or a line segment when it is small.
6. Noise and blurring are common during the imaging process.

1.1.3 A Detailed Specification of My Segmentation Algorithm

Based on the arguments above, plus a widely accepted opinion in perceptual psychology best stated by Purkinje about 150 years ago: *optical illusions contain visual truth*, the objectives of this research can be more specifically defined as the design of a segmentation algorithm which

1. detects simple closed contours successfully,
2. detects corners properly,
3. handles noise and blurring reasonably,
4. does not lose spatial accuracy,

5. functions in real-time,

6. can explain subjective contours.

1.2 My Approach

Vision research is difficult partly because the visual process is spontaneous so introspection does not help, and partly because the traditional psychophysical or physiological approaches are insufficient to explain *how* the visual tasks are fulfilled. My approach is to perform a functional analysis on visual tasks. The algorithm thus derived is then implemented by software and applied to various test patterns. The results are used to show the performance of the algorithm and are compared with human performance. In the following I will describe the reasons to adopt a connectionist approach and the constraints imposed by this approach.

1.2.1 The Appropriate Architecture for Vision

An effective algorithm design must be based on a proper architecture, but what architecture is appropriate for vision? The von Neumann machine is not the answer because a sequential computer is too slow to process the time sequence of two-dimensional images. A simple calculation further demonstrates this point: assuming that ten frames of visual input are processed per second, and each frame is of 1000×1000 pixels. Assuming further that to parse and classify all these pixels requires on average 100 instructions per pixel, then it takes a dedicated computer of 1 BIPS to achieve the required performance. This rate is not possible based on current technology.

Conventional parallel processing techniques do not seem to be the right answer either. The reason is that each node of the multiprocessor is still a von Neumann machine. The need to feed instructions and data to each processor requires the identification of simultaneously executable portions of a given algorithm. This task of parallelism detection is extremely difficult. Together with the overhead of interprocessor communication, the multiprocessing scheme is not likely to provide the required performance.

Therefore, a special-purpose hardwired circuit tuned to process two-dimensional images is needed. Just as with the biological visual system, a neural network (i.e., a connectionist) approach seems to be the answer. Contrary to the conventional computer,

a connectionist approach operates through a large number of simple processing units. When properly connected, the network performs globally useful functions.

1.2.2 Properties of a Connectionist Architecture

The most prominent property of a connectionist model is local wiring, which, as Hubel and Wiesel [1977] pointed out, is one of the common characteristics of the neural system. Local wiring distributes the sensory information so that the input image is processed in parallel. Local wiring also allows greater flexibility for changing the local structure, thus enabling the system to adapt and learn. Furthermore, the functional locality of the system assures reliability.

Adaptivity and parallelism have costs, too. Each neuron is a local processor with limited capability. Thus a task requiring global information is very difficult to implement. Moreover, the receptive field and target function of a specific neuron are decided completely by the connections along the path from the input to this neuron. Locally a neuron has no control of its function. A consequence is that if a layer of neurons is topographic to the retina, then there exist regular connections from receptors in the sensor to the neurons in this layer. However, if a neural layer is not topographic, then the spatial relationship cannot be recovered for the neural layers whose input depends solely on this layer. In other words, if vision is to answer *what* is *where*, then *where* has to be decided early in the process. The fact that only early visual areas, V1, V2, V3a are topographic to the retina [Phillips, 1984; van Essen, 1983] makes this point relevant for the human visual system.

Another problem of the connectionist approach is the costly data representation. Since a neuron can represent only the value of a specific signal, it takes numerous neurons to represent a quantity. For example, for certain edge computation algorithms a neuron is needed for every orientation at every location. Therefore, the economic use of resources is important.

To summarize, considering a parallel distributive approach, the algorithm under design must obey the following constraints:

- All operations are local.
- Each processing unit performs only simple functions.

- Spatial information can be obtained only through the interconnections to the physical layout of the processing units.
- The resources are efficiently used. A pipeline may be ideal.

1.3 The Thesis

My thesis is that an image can be reasonably segmented into regions in real-time by a parallel distributive algorithm comprising processing stages of

- multiple edge filtering with the first-order directional derivatives of Gaussian as kernels,
- corner detection based only on edge filters' outputs,
- spatial coherence processing of the edge and corner information

where these stages are applied at multiple scales.

1.4 Main Results

The results of applying my algorithm to many test patterns show that the algorithm works reasonably well for various test patterns under many conditions. The algorithm is architecturally regular, has only one parameter (edge threshold), and requires only short range communication among the simple processing units; hence it is adapted to implementation by current VLSI technology. It appears that, with proper implementation, the model will perform image segmentation in real-time.

The main contributions of this research include

1. The development of a corner detector based only on edge filter outputs.
2. The design of an effective segmentation scheme based on spatial coherence of edge and corner information.
3. The effective use of multiscale anisotropic first-order directional derivatives of Gaussian as edge filter kernels.
4. The verification and extension of Grossberg and Mingolla's finding on artifact cancellation.

Chapter 2
Background

This chapter first describes conventional approaches to the problem of image segmentation and then elaborates on a connectionist approach — Grossberg's *boundary contour system*. Lastly the chapter gives a general view of my algorithm.

2.1 On Image Segmentation

Traditionally, segmentation techniques based on a von Neumann architecture can be categorized into region-based and edge-based methods. Since Hubel and Wiesel [1979] discovered the organization of simple cells in the primary visual cortex, much research in computer vision has emphasized the edge-based methods. This section first describes region-based segmentation and then elaborates on edge-based algorithms. Newer algorithms based on more recent progress in computer architecture are then briefly summarized.

2.1.1 Region-based Segmentation Algorithms

Typical region-based segmentation partitions the image into connected regions made up of pixels possessing roughly uniform values of some property like grey-scale intensity. Ballard and Brown [1982] classified the algorithms into local, global, and split-and-merge techniques. *Local techniques* put a pixel into a region according to the properties of the pixel's close neighbors. An example is *blob coloring* [Ballard and Brown, 1982]. *Global techniques* group pixels into regions based on properties of pixels throughout the image. A widely used example is thresholding at the gray level which separates the peaks of a bimodal histogram. *Region splitting* splits an image into sets of regions according to some heuristics until no splitting is possible, while *region merging* merges the pixels into regions according to some heuristics until no further merging is possible. The heuristics are usually based on proximity and similarity and are implemented by local operations followed by thresholding.

The main advantages of the region-based algorithms are that they usually give closed contours and region properties can be computed during the segmentation process. There are also shortcomings. The performance of the algorithm depends on the threshold selected and the order of pixel scanning. For a real image, it is often difficult to choose a proper threshold. Moreover, a region-based algorithm often needs to calculate a global function, e.g., the histogram for bimodal thresholding, or requires a complicated intermediate data structure, e.g., a list of the pixels merged or split. These requirements make the region-based algorithms difficult to implement on a connectionist architecture which consists of only simple processing units, each working on only local information.

2.1.2 Edge-based Segmentation Algorithms

An edge-based segmentation algorithm includes an edge detection stage and an edge following stage. These two stages are separately described in the following.

Edge Detection

Generally, edge detection is performed by either convolution with a filter or by parameter setting within a model of an ideal edge or curve. An example of the the latter method is Hueckel's [1973] visual operator for edge-line recognition. Since the model-driven approach is not considered in this research, the filtering approach is described below in greater detail.

Edges are abrupt intensity changes in the image, and many methods based on this property have been proposed for edge detection. Among the most famous are the gradient methods of *Canny*, *Sobel*, *Kirsch*, and the Laplacian method of *Marr*. [Ballard and Brown, 1982; Rosenfeld and Kak, 1976; Canny, 1986; Marr, 1982]. Torre and Poggio [1986] showed that edge detection is an ill-posed problem because the numerical differentiation in the process causes the result of edge detection to depend on the input data discontinuously. To make the problem regular, the differentiation must be coupled with a smoothing filter.

For smoothing, many authors [Koenderink, 1984; Torre and Poggio, 1986; Asada and Brady, 1986] have shown that the Gaussian filter is attractive because it gives the maximal degree of causality across spatial scales. For differentiation, several Gaussian-based operators have been used, e.g., the difference of Gaussians (DOG), the Laplacian of a Gaussian (LOG), and the directional derivatives of a Gaussian.

I have chosen the first directional derivative of the Gaussian as the edge filter kernel. The decision is mainly based on Koenderink's derivative of Gaussian (GD) model [Koenderink, 1987] which provides a sound mathematical basis for detecting the intensity changes in the image. The Gaussian *n-jet* — the convolution of nth-order derivatives of the Gaussian with the image — not only describes the early visual process elegantly, but also provides an efficient computational scheme. This edge filter will be discussed in more detail in Chapter 3, on edge filtering.

Edge Following

Edge following, or *boundary completion*, combines the local edges to form a more global element such as a long line, a curve, or a simple, closed contour indicating an object or a portion of an object. According to Ashkar and Modestino [1978], edge following can be approached by exhaustive search, dynamic programming, structured tree search, and heuristic search. In the following an example of each of the above approaches except exhaustive search is briefly described.

Fischler et al. [1973] applied *dynamic programming* to boundary completion. The method requires that a description scheme of the target object and a decision metric be selected. Then the algorithm uses the decision metric to find the described object in the image by dynamic programming. In my trial implementation a nearly vertical edge which has one point in each row of the image is the target boundary. The program starts from a point in the top row and scans downward. For each position, the minimum cost and the corresponding path from the top to the position under consideration are calculated. When the bottom row is reached, the optimum nearly vertical edge is obtained.

An example of *structured tree searching* is described by Chien and Fu [1974]. Knowledge about the specific application under consideration is formulated into a criterion function which usually consists of a local term, e.g., edge strength, and a global term, e.g., the relation between the pixel under consideration and other pixels. Then the boundary points which minimize the criterion function are selected. A preprocessing stage gives several candidate starting points. Then the process of including other boundary points proceeds like a tree search. Standard techniques like backtracking and tree-pruning can be used to obtain the path of minimum cost.

algorithm	image resolution	cpu time(sec)	memory (KB)
Fischler et al.	64 × 64	73.0	79
Chen and Fu	64 × 10	61.2	40
Ashkar et al.	64 × 64	1.6	326

Table 2.1: Test results of 3 edge following algorithms.

In Ashkar and Modestino's *heuristic searching* algorithm [1978], contour extraction is first formulated into a tree search problem. A cost for traversing a branch is then defined based on the likelihood of the branch's lying on the true contour. Both local and contextual information, as well as as closeness to a prototype, are used for the cost definition. The most likely path is then extracted by a heuristic tree search similar to *algorithm A* [Nilsson, 1980].

I implemented the three algorithms in the C programming language and applied the programs to detect a nearly vertical vessel in a digital subtraction angiogram. The following table summarizes the required resources for the three approaches on a Vax 780.

Note that for Chen and Fu's algorithm a preprocessing stage first selects 20 × 10 possible edge points from the image and the tree search operates on the selected points only.

The performance of the above algorithms are far from real-time, and all three algorithms are target-dependent and require the computation of a global cost function. The parameters in the algorithms are determined empirically. When random noise increases, the edge strengths and directions, which are usually calculated by an edge operator with a small support and serve as a local term in the cost function, are prone to error. Hence the weights between the local and global costs need to be adjusted, and the algorithm is image-dependent.

In summary, edge following is an unsolved problem. New and more effective algorithms are needed.

2.1.3 Multiscale Mechanism for Image Segmentation

The concept of multiscale perception has strong physiological and psychophysical evidence. The idea originates from the well-known physiological finding that the ganglion cells in the primate retina have overlapping receptive fields of various sizes at each location of the visual field, and the average size of receptive field increases with the distance from the fovea. In psychophysics, Campbell and Robson [1968] demonstrated through a set of threshold detection studies that the visual system possesses independent, spatial-scale-tuned channels. Since the early '70's this mechanism has become a basis of several theories of visual organization.

In the past several years many computer scientists have applied this idea to design computer vision algorithms. Many previous segmentation techniques, both region growing and boundary detection types, use mainly local information, and the performance of these algorithms is limited. It has been suggested that the solution lies in using local and global information effectively, and from this point of view the multiscale scheme is certainly an attractive one. Witkin [1983] showed that intensity extrema can be localized at a finer scale after being identified at a coarser scale. Since then this concept has been used to tackle numerous problems in computer vision. Several such methods are briefly described in the following with a brief summary of each method's advantages and disadvantages.

Stack

Koenderink and Pizer's *stack* model appears most attractive. In this scheme, the multiscale version of an image, the *stack*, is generated by successively blurring the image with a two-dimensional Gaussian. An example of the use of this scheme is Lifshitz's [1987] image segmentation algorithm which follows the paths of extrema through the stack. By following the intensity extrema through multiple scales, a tree structure is generated, and the tree can then serve as a shape description of the image. Since a node in this tree corresponds to a region in the base image, the tree structure defines a segmentation. Applications of this algorithm on several medical images were rather successful.

This scheme has sound mathematical basis. It originates from the study of human visual processes and is natural for the connectionist approach. The blurring process is causal. During the tree generation process, the object containment relationship can also be computed. A problem of this approach is that the annihilation sequence of the extrema

is sensitive to small changes of the image, and hence to noise, too. Besides, the generation of the extremal paths is time-consuming.

Pyramid

The *pyramid* image description [Hong et al., 1982] is a multiscale approach based on a rectangular sampling grid. Each pixel intensity at the coarser resolution is determined by the pixel intensities within an overlapped square in the next finer level. The reduction in spatial sampling between successive levels is usually 2×2. Therefore, if the spatial sampling of the base image is $n \times n$, for the scale x level away from the base image, the spatial sampling is $n/2^x \times n/2^x$. The top level has only four pixels. This scheme is called *pyramid* because the shape of this multiscale representation resembles one.

The pyramid scheme is efficient in memory space and in processing time. Image features are detected in the coarser level, while the more accurate location of the detected feature can be obtained by tracing down the pyramid. The scheme is also conceptually easy to understand. Since the connections between elements of two successive scales are relatively fixed, a hardware implementation is feasible. The intensity of a pixel at a coarser scale is usually calculated by averaging the pixel intensities connected to this pixel in the next finer scale; the major problem of the approach is that the procedure is not causal. Thus, an image feature shown in the coarser scale may be generated as an artifact of the blurring process. An example of this effect is the aliasing of intensities in a coarser level. Besides, the relationship among features at the same scale is difficult to define. Lastly, the scheme is rigid; it is difficult to incorporate model-driven techniques into the scheme.

Difference of Low-Pass Transform (DOLP)

Crowley and Parker [1984] developed a shape representation based on the results of a set of differences of low-pass filters. The peaks and ridges of this DOLP transformation constitute a graph, which are then used as a shape description.

This shape description can be used for matching regardless of the object's size, orientation, and position. The DOLP transformation loses no information and is reversible. Matching proceeds in a top-down fashion, i.e., the most important objects in the image are compared first. Both the structure of the graph and the information at each node can be used for matching. The generation of the shape description is also computationally

efficient. The *three-dimensional ridge* of the DOLP transformation indicates the elongated form of an object and, in some sense, is like a medial axis. For a real image with multiple objects, the graph is usually complicated, and it is unclear how the graph for a particular object can be separated from others.

Zero-crossing Across Scales

Marr [1982] applied the Laplacian of a Gaussian ($\nabla^2 G$) of various standard deviations to an image. The zero-crossings of these filter outputs give the object boundaries in the image. At a coarser scale, a larger filter kernel detects the contours of larger objects. The edges, terminations, bars, and blobs in the image can be extracted from the zero-crossings of $\nabla^2 G$ in multiscale. They then serve as the *raw primal sketch* of the image.

The zero-crossings of the result of filtering by the Laplacian of Gaussian ($\nabla^2 G$) is widely used in computer vision. The scheme always gives a closed contour, is computationally efficient, and the filter kernel resembles the familiar shape of the on-center, off-surround receptive field. Nevertheless, this approach has its shortcomings, too. First, the isotropy of the filter causes spatial inaccuracy, especially for sharp corners [Berzins, 1984]. Second, the second-order differentiation in $\nabla^2 G$ amplifies noise when compared with edge detectors using only first-order derivatives. Third, no orientation is explicitly represented, which makes it very difficult, if not impossible, for this scheme to further use edge information for connectionist modeling. For example, how can subjective contours, which I conjecture to be evidence for the preattentive segmentation process, be generated? In conclusion, the zero-crossings of $\nabla^2 G$ may provide preliminary information on object boundaries, but, as Torre and Poggio [1986] commented, are insufficient to account for the segmentation process in early vision.

Snakes

Kass, Witkin, and Terzopoulos [1987] developed a contour model which integrates the model-driven and the data-driven techniques for contour detection into an energy-minimizing scheme accounting for both edge strength and the closedness and smoothness of object boundaries. With the inclusion of scale space information via successively blurred images, the scheme gives better performance on boundary detection.

The scheme is not only good at boundary completion, but can also account for other early vision phenomena such as subjective contours, motion tracking, and stereo matching. To implement this approach on a connectionist architecture, the global energy term would require a large number of neurons. Moreover, based on our laboratory work [Oliver, 1988], given an arbitrary initial condition, the relaxation algorithm based on the energy functional does not necessarily converge in a few iterations.

2.2 On the Connectionist Approach

As argued in Chapter 1, a connectionist approach is most likely to give real-time performance for early vision tasks. This section first introduces the field of neural networks and then describes Grossberg's *boundary contour system* in some detail.

2.2.1 Introduction to the Neural Network Approach

A connectionist network can be fully specified by the functions of each unit, the connections among units, and the dynamics defining the changes of the network. Usually, all units in the network have the same functional characteristics. Each of them performs a simple operation, e.g., a weighted sum of its inputs followed by a transformation according to the sigmoid function and a thresholding. The system dynamics are generally represented by the change of the connection weights (called the *synaptic weighting factors*) according to a learning rule. The most popular learning rule in the field now is *Hebbian*: the change of the weighting factor between sending neuron i and receiving neuron j, Δw_{ij}, is proportional to the response of the neurons, a_i and a_j, i.e., $\Delta w_{ij} = \eta a_i a_j$, where η is a coefficient. In my research the preattentive network is assumed to have been stabilized, so learning is not under consideration.

The objectives of connectionist research are two-fold: to study the structure and functions of biological systems and to investigate the design principles of this new computational device. Both software simulation and hardware implementation are used for connectionist modeling. The results are compared with human behavior or are shown to have certain computational capabilities.

The following subsections briefly describe several connectionist schemes. Among them, Grossberg's *boundary contour system* is physiologically feasible and unifies a great deal of psychological data. My research started by implementing this model. Hence the

boundary contour system, together with the problems I encountered during the implementation, are described in a separate subsection.

The Perceptron

Rosenblatt [1962] invented a class of simple pattern-learning networks called the *perceptron*, which is a single-layer network of linear threshold units without feedback. Each input pattern is presented as a teaching input. Let t_i indicate the input at neuron i. The learning rule is $\Delta w_{ij} = \eta(t_i - a_i)a_j$, i.e., the change of the weight of a connection is proportional to the response of the sending neuron and the difference between the teaching input and the response of the receiving neuron. The *perceptron convergence theorem* guarantees that if the set of patterns are learnable, the learning procedure will find a set of weights which allow the perceptron to respond correctly to all input patterns.

The perceptron stimulated a great deal of research interest in the early 60's until Minsky and Papert [1969] showed that it cannot learn some simple, common functions, for example, the exclusive-or,

Mead's Silicon Retina

Mead [1985, 1987] designed a set of *silicon retina* chips using analog VLSI. The design integrates photosensors and motion-detecting circuitry based on knowledge about the neural connections in the retina. It was shown that the chip can detect a rotating bar in real-time.

The *silicon retina* is one of the few chips implementing connectionist vision algorithms ta present. A rationale for the hardware approach to connectionist modeling is the observation that software simulation of neural networks of substantial size requires tremendous computing power and it is almost impossible to analyze problems like motion detection. The development of the chip is a milestone, but, of course, the chip implements only a prototype of a very specific function. There is still a long way to go.

Linsker's Work

Linsker [1986] investigated the self-adaptive property of the visual system by software simulation. His model includes a multilayer feed-forward network. Each layer contains hundreds to thousands of cells, and each cell has up to hundreds of inputs. The

response of each unit is normalized after activation. The connection can be excitatory or inhibitory. The learning rule is *Hebbian*. Initially, synaptic weighting factors are randomly assigned and the environmental stimuli are shown to the first layer. When learning completes, his simulation result shows that the spatial arrangement of the orientation-sensitive cells in the seventh layer looks similar to a photo taken of a monkey's visual cortex.

Linsker's work shows how computer modeling can relate to physiological data but is not very useful in designing a computer vision algorithm because the model does not aim at well-defined visual tasks.

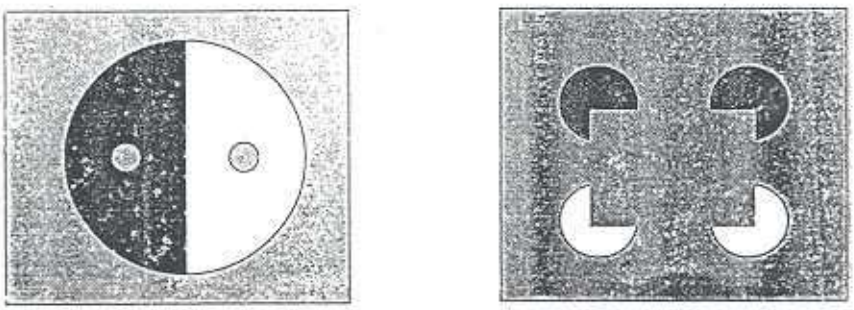
Fukushima's Neocognitron

Fukushima [1988] developed his *neocognitron* — a feed-forward, hierarchical, and multilayered network capable of learning and recognizing arbitrary patterns. The idea is mainly based on physiological evidence: there seem to exist hierarchical neural paths in the cortex for feature extraction and pattern recognition. After a two-dimensional input is presented to the network, features such as lines of various orientations and corners of various opening angles are successively extracted. These features can form numerous combinations with different spatial relationships among them. During the learning stage the input patterns are repetitively shown to the network until proper connections are established. Then, during the recognition stage, one and only one neuron in the output layer will fire when a certain input pattern is presented.

The network was applied to recognize handwritten numerals. The results show that the network can handle patterns after deformation, change of size, and shift of position. The network can be self-organizing, i.e., learn without a teacher. Nevertheless, the feed-forward network requires many neurons. For more complicated test patterns, like English letters, more neurons are required. Also it is unclear how well the network can differentiate two patterns with a small discrepancy.

2.2.2 Grossberg's Model

Grossberg, based on psychophysical and neurophysiological evidence, separated early vision into a boundary extraction process and a filling-in process. The reason for separating the early vision tasks into two stages is mainly based on the phenomena of brightness and color constancy: though lighting conditions in the environment vary con-



(a) (b)

Figure 2.1: Two examples of visual illusion. (a) Yarbus (b) Kanizsa

siderably, the perceived brightness or color of an object remains fairly constant. For a picture of multiple color patches, Land [1971] showed that the perceived color of a patch in a picture is mainly determined by the contrasts at the edges between the patch with and its surround. Furthermore, Yarbus [1967] showed that, as in Figure 2.1a, when the edges of the large circle and the vertical line are stabilized on the retina, the red color outside the large circle fills in the black and white regions except the small circles whose edges are not stabilized. The red inside the left circle looks brighter, and the red inside the right circle looks darker than the enveloping red [Grossberg and Mingolla, 1985]. This experiment shows that the object boundary can be dissociated from the object surface properties like grey-scale intensity or color. The above psychophysical evidence led Grossberg to conclude that the visual system discounts the illuminants within objects by first extracting edges of the objects and then using the brightness or color information across the edge to fill the regions within boundaries.

Grossberg then developed the *boundary contour system* to extract the object boundaries in the image. The model consists of 4 layers as illustrated in Figure 2.2. Following the first layer of edge filters, there are two competitive stages: one for edge-thinning and the other for line-end processing. The last layer is a cooperative layer. The outputs of the last layer connect to the first competitive stage and form a loop which completes the gaps on the boundary. Ideally, for a given input, the computation converges in several iterations and gives simple closed contours segmenting the image. Except for the edge filter, which is described in Chapter 3, each of these stages is briefly described in the following.

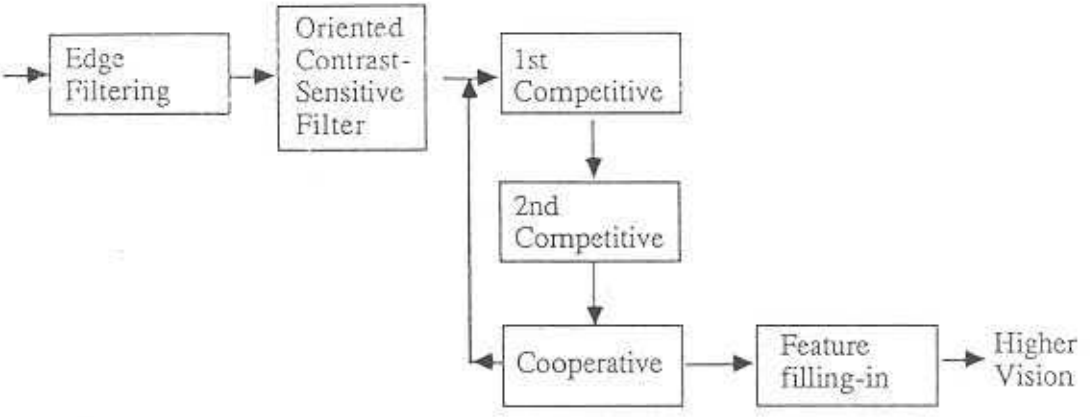


Figure 2.2: A block diagram of Grossberg's *boundary contour system*.

Oriented Contrast-sensitive (OC) Filter

The first layer after edge filtering is an oriented contrast-sensitive filter, wherein each neuron in the layer receives responses from two edge filters with the same orientation as the neuron under consideration but with opposite contrast directions. This operation is based on the fact that, as in the Kanizsa square depicted in Figure 2.1b, illusory contours can be generated by joining edges with opposite directions of contrast.

First and Second Competitive Stages

The first competitive stage defines competition between neurons of the same orientation at nearby locations, and the second competitive stage defines interorientational competition at a single position. The functions of these two processing stages can best be demonstrated by their effects on the Ehrenstein illusion depicted in Figure 2.3a.

To generate the illusory circle, Grossberg assumed that an illusory line segment perpendicular to each line end must be generated. A boundary completion process then connects the line segments to give the circle. Figure 2.3b shows the on-center, off-surround connections for the first competitive stage. For each orientation, each neuron excites the neurons near its location in the next layer and inhibits the neurons farther away. The second competitive stage is defined as a *push-pull* operation between neurons with

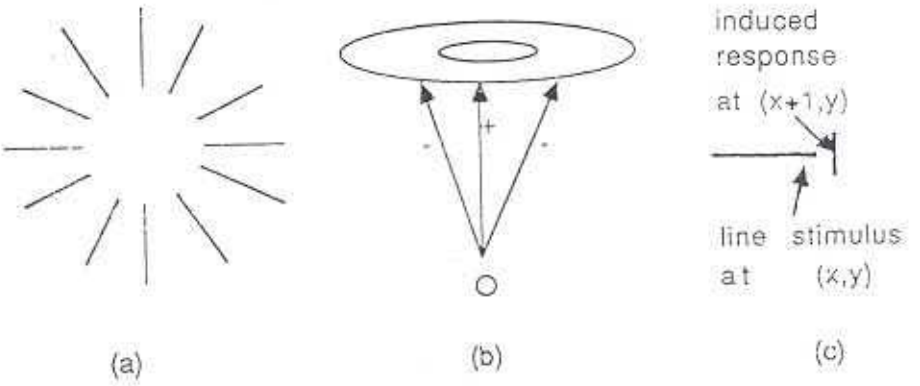


Figure 2.3: Ehrenstein illusion and Grossberg's competitive mechanisms. (a) depicts Ehrenstein illusion, (b) shows the on-center, off-surround connection, and (c) shows the effects of the two competitive stages near a line end.

perpendicular orientational sensitivity (in Grossberg's term, *a dipole competition*).

The two competitive stages generate an illusory line segment near a line end in the following way: in Figure 2.3c, there is a horizontal line segment ending at location (x, y). After the first competitive stage, the neurons with near-horizontal receptive fields at location (x+1, y) are inhibited by the neurons at (x, y). Then the second competitive stage activates the neuron with near-vertical receptive field at location (x+1, y).

Oriented Cooperation

Oriented cooperation, together with the competitive stages, forms the *cooperative-competitive (CC)* loop, which is designed for boundary completion. Each neuron in the cooperation layer has a *bipole* receptive field as illustrated in the Figure 2.4. The neuron is fired only when both halves of the receptive field are excited above threshold. The response of the neuron is then fed back to the first competitive stage through other on-center, off-surround connection. Grossberg applied the algorithm to several test patterns and showed that the CC loop converges after several iterations.

2.3 Problems Faced in Implementing the Boundary Contour System

A version of Grossberg's *boundary contour system* was implemented, and it was applied to test patterns of artificial, medical, and natural scenes. The major problems

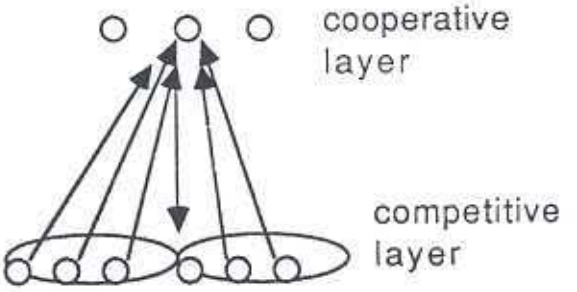


Figure 2.4: The cooperative-competitive loop of the boundary contour system

encountered during the implementation and my corresponding solutions are briefly summarized below. The details of these solutions are given in later chapters.

1) Grossberg [1985] implemented edge filtering by convolving an input image with difference-of-box (DOB) filters. Is this a reasonable choice?

My conclusion: The convolution result using the first-order directional derivatives of the Gaussian as filter kernel was compared with that of using the difference-of-boxes kernel. The conclusion is that, when the length and width of the rectangular edge filter is carefully specified, the two kernels give similar convolution results. Since the Gaussian function agrees better with our knowledge of the biological visual system, it was used in my algorithm.

2) Grossberg found that at each location, if the edge filter output for a certain orientation is subtracted from the edge filter output of the perpendicular orientation, then the follow-up processes behave much better. Why is it so?

My conclusion: The operation is an effective way to discount artifacts due to digital sampling and finite approximation, so this operation, called *artifact cancellation*, is included in my algorithm. However, here it is performed after edge filtering instead of being performed in the competitive-cooperative loop.

3) Edge detectors do not behave as expected near corners. At a sampling location, an edge detector is intended to detect the occurrence of a linear edge segment. When performed near a corner, does the edge filter output correctly indicate the local shape of the bounding contour? If not, how can a closed boundary be defined based on the edge

filter output? Furthermore, can the cooperative layer correctly complete the gaps on the boundary near corners?

My conclusion: The edge filter output cannot represent corners on the boundary, and a bipole field is improper to apply near corners. Hence a corner detector based on edge filter outputs was designed. The results of applying the detector to many test patterns show that the scheme is effective.

4) The first competitive stage is a shunting competitive network for edge contrast enhancement. How effective is this operation?

My conclusion: Implementation following the equations in [Grossberg and Mingolla, 1986] does not enhance the edge contrast as expected. I decided that this operation is unnecessary for segmentation because the edge filter outputs provide useful information and should not be altered in this way.

5) As illustrated in Figure 2.3, the second competitive stage is a dipole field designed to explain the process near line ends so that Ehrenstein illusion can be explained. Is the result generated by this process strong enough so that the follow-up cooperative process can use it to complete a boundary?

My conclusion: My implementation showed that the dipole field in Grossberg's model activated a small number of neurons near a line end. Since, in the Ehrenstein illusion, the gap between two nearby line ends can be several pixels wide, my implementation did not confirm that this mechanism can explain the Ehrenstein illusion satisfactorily. Instead I conjecture that a line in the image can be recognized as a long, thin object different from an edge. A specific line and line end detector and the follow-up processing may be necessary.

6) What is the range of parameters which allows the network to converge? What are the criteria for selecting the network parameters?

My conclusion: In my first implementation of Grossberg's model, there were 14 parameters that define the network connections and operations. When the parameter values were properly selected, the network converged in several iterations and gave reasonable segmentation. Programs simulating each layer of Grossberg's model were written, and

different parameters were used against various test patterns. An important consideration for selecting the parameters was the balance between the two major functions of the CC loop: boundary completion and edge thinning. When the parameters were improperly set, the network would either generate more line segments than allowed or fail to complete required boundaries. Another criterion I found useful was that parameters should cause the dynamic range of the neurons in a layer to be fully used.

7) The neural layer of *bipole* fields is intended to recover lost edge information. How effective is this operation? What if the gap to be completed, as in *Kanizsa square*, is wide? Will a multiscale scheme help to solve the boundary completion problem?

My conclusion: The competitive-cooperative loop in Grossberg's model is effective in completing the gaps between nearby edge segments. The problem is that a wider gap in the input pattern requires a longer bipole field; thus the network is image-dependent. To cope with the difficulty of image-dependence, in my current algorithm the bipole field covers only the pixel's closest neighborhood. Elongated edge filters of three elongations give edge strengths at positions along a real edge, and these propagated edge strengths then complete the gaps up to some width between two linear edge segments.

But the scheme of multiple elongated edge filters does not solve the problem that a gap with substantial width cannot be completed. My first attempt at a solution followed Rosenfeld's *pyramid* scheme [Hong et al, 1982]. Based on the layer storing the result of OC filtering (which I call the *edge map*), a pyramid as described in Section 2.1.3 was established. A CC loop was then associated with each level of the edge maps. The edge information first propagated bottom-up to the coarser levels. After the CC loop at each level performed boundary completion, the completed boundary at the coarser levels then propagated down the scales to attain better localization. This scheme was implemented, and preliminary results showed that the model could complete larger gaps between two linear edge segments. The problem with this scheme is that the algorithm requires both bottom-up and top-down information flow. The control is very complicated, and the computation takes considerable time.

Finally I conjectured that a multiscale scheme of multiple elongated edge filters, with different pixel size and corresponding sampling interval at each scale, might be able to segment the image into contours at different levels of detail and close gaps of quite

different sizes. A program was implemented based on this idea. Running the program on several test patterns showed that the scheme is effective.

In summary, for Grossberg's model there may not exist an ideal set of parameters that enables the network to converge for all input images. But if a network that has to handle vastly different input data depends on data-dependent variations of the parameters, it seems unlikely to be practical.

2.4 An Overview of My Algorithm

This section summarizes my algorithm derived from the above considerations. Figure 2.5 shows the block diagram of the algorithm. The algorithm consists of multiple subsystems, each characterizing a different scale. At each scale a segmentation of the image is provided via several processing stages. Information flow is parallel for all pixels at each stage and pipelined among stages. In a coarser scale the segmented contours are simpler and bigger, while in a finer scale details of the object contour are shown. The results of segmentation are continuously delivered to higher visual processes.

1. The *input* is taken to be a two-dimensional array of real-valued intensities ranging from 0 to 1. For artificial test patterns, the portion of a pixel covered by a figure is calculated and the pixel intensity is adjusted accordingly.

2. *Edge filtering* is performed by filters combining a smoothing operator with a differentiation operator. Each scale has multiple edge filters, each with a different elongation. The edge filter with larger support not only provides edge information with higher signal-to-noise ratio, but also propagates the edge information along a linear edge, which to some degree enables the network to complete the gaps between two edge segments. As Marr [1982] and many others pointed out, a problem with this type of feature template approach is that a significant response to the filter does not necessarily indicate the presence of the feature. The following processing stages are mainly designed to cope with this problem.

3. *Artifact cancellation* is applied after edge filtering at each sampling location. Since except near certain complicated image structures there should not be perpendicular edges at a single location, the counter-interaction between edge filters of perpendicular target orientations effectively discounts the artifact due to discrete sampling.

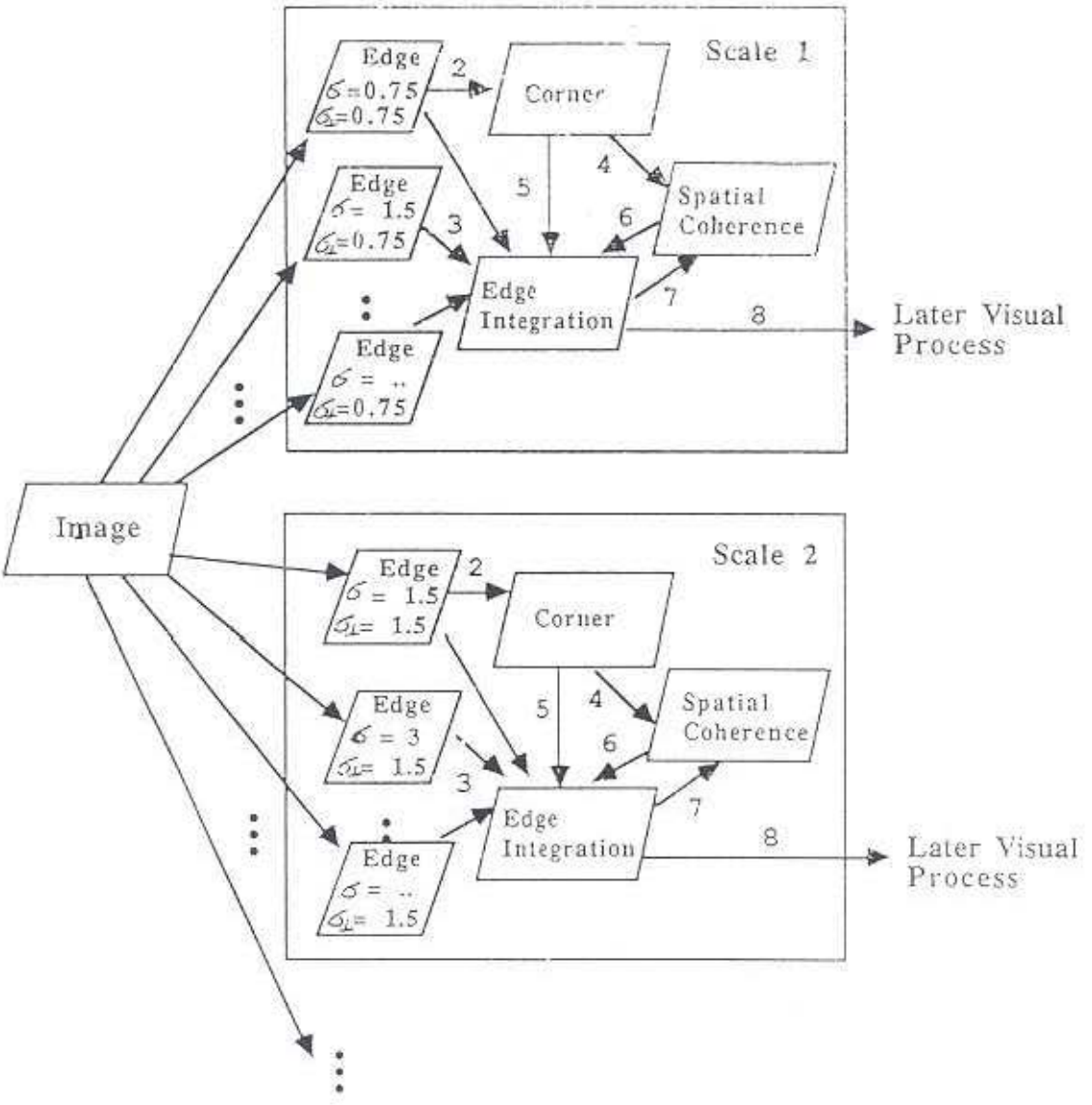


Figure 2.5: A block diagram describing the processing stages of my algorithm

4. *Corner detection* is another means to cope with the problem of false alarm in edge filtering. Corners are themselves important both for object recognition and for definition of a closed object boundary.

5. *Spatial coherence checking* verifies a detected edge or corner at a certain location by rejecting edges where the edge and corner information in adjacent locations are incoherent. The pixel size here is proportional in size to the scale.

The following chapters elaborate on these issues and describe my research. Chapter 3 describes *edge filtering* and my rationale in using multiple directional derivatives of elongated Gaussian as edge filter. Chapter 4 discusses my approach to *corner detection.* Chapter 5 describes the role of *spatial coherence checking* in my algorithm and shows the results of segmentation on several test patterns. Chapter 6 gives an evaluation of the overall algorithm by comparing its performance with Grossberg's *boundary contour system* and with human performance. The conclusions and suggested future research directions are presented in Chapter 7.

Chapter 3

Edges

Given an image, what information best indicates object boundaries? This chapter describes the measurement of intensity changes, called *edge strengths*, in a two-dimensional, static, grey-scale image.

The chapter first compares the performance of different edge filter kernels and describes my decisions on edge filter design. Then it explains why an elongated directional derivative of Gaussian (DDG) function is used in my algorithm. The operation of *artifact cancellation* is then presented. Lastly the chapter explains why multiple edge filters are applied.

3.1 Derivative-of-Gaussian vs Difference-of-Box Filters

Many two-dimensional functions have been used for edge detection. Grossberg [1985] used a difference-of-box (DOB) function and obtained interesting results, but, as discussed in Chapter 2, the first-order directional derivatives of two-dimensional Gaussian (DDG) prevails for modeling human visual system. It is worth comparing the performance of these two filter kernels.

The DDG and DOB functions can be described by the following equations. Let $G(x, y)$ represent the two-dimensional Gaussian,

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{3.1}$$

The directional derivative along the x direction gives an edge filter with the target edge orientation along direction y.

$$\frac{dG(x,y)}{dx} = -\frac{x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} = -\frac{x}{\sigma^2} G(x,y). \tag{3.2}$$

A DOB with change in x direction and width w can be written as $DOB(x, w) = sign(x) \prod(\frac{x}{2w})$, where

$$sign(x) = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ 0 & \text{elsewhere} \end{cases} \tag{3.3}$$

and

$$\prod(\frac{x}{2w}) = \begin{cases} 1/w & |x| < w/2 \\ 1/2w & x = \pm w/2 \\ 0 & \text{elsewhere} \end{cases} \tag{3.4}$$

The DDG and DOB filter kernels are shown in Figure 3.1II, wherein small *diamonds* and *squares* indicate positive and negative weights, respectively; the areas of the squares and circles indicate the magnitudes of the weights at various locations. Note that in Figure 3.1II both edge filter kernels are shown on a 8×8 grid. Though a Gaussian function extends to infinity, for practical considerations the weights for the locations with distance to the center larger than a critical distance, r, are set to zero. In my current implementation, r is selected as the smallest real number such that $G(r)/G(0) < 0.05$. For the pixels near the boundary of a filter kernel, the portion of the pixel area covered by the kernel is calculated and the weight is adjusted accordingly. This process accounts for why, in Figure 3.1IIb, pixels away from the center of the kernel have smaller weights. Figure 3.1I shows the three-dimensional graphs of these two filters.

To compare the performance of these two filter kernels, I applied the two edge filters with various parameters to several test patterns. The results show that when the length and width of the DOB filter is equal to the equivalent length and width of the DDG filter, the two kernels give similar convolution results, where the terms *equivalent length* and *equivalent width* follow Bracewell's definition [1986]: for a positive function, the equivalent width is the area covered by the function divided by the function's maximal value.

The equivalent length and width of a DDG filter can be calculated as follows. A cross section along the target edge orientation of the DDG filter is a one-dimensional Gaussian which has the maximum at the center. The equivalent length of the kernel, L, is then

$$L = \frac{\int_{-\infty}^{\infty} G(y)dy}{G(y)|_{y=0}} = \sqrt{2\pi}\,\sigma. \tag{3.5}$$

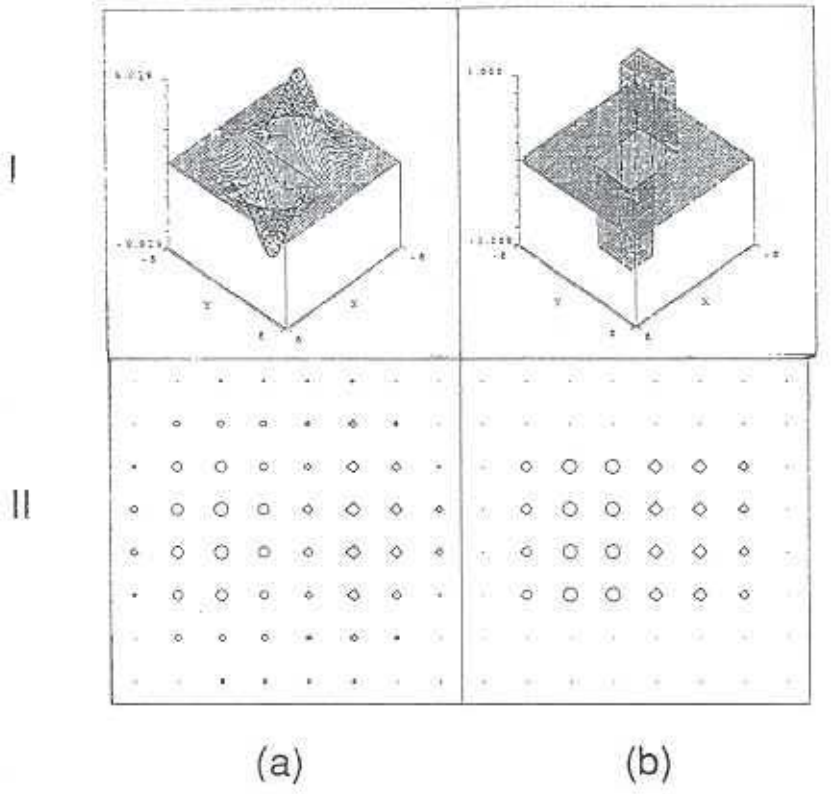


Figure 3.1: A comparison between a derivative-of-Gaussian and a difference-of-box filter. Row (I) are the three-dimensional line diagrams for (a) DDG function with $\sigma = 1.5$ and (b) DOB function with $L = 3.76$ and $W = 2.473$; Row (II) are the corresponding edge filter kernels. (The sizes of diamonds and circles in row (II) are proportional to negative and positive weights at each location within the kernel. The circle size represents the intensity.)

image	resolution	#edge strn > 0	max diff	ave diff	SD
c30	16 × 16	648	0.1725	0.0549	0.0418
tri	32 × 32	1107	0.2039	0.0674	0.0491
owl	64 × 64	16071	0.1608	0.0214	0.0229
ct3	128 × 128	35011	0.1098	0.0160	0.0145

Table 3.1: Difference between the convolution results of a directional derivative-of-Gaussian filter and a difference-of-box filter. The test patterns, e.g., *c30* and *tri*, are described in Table 5.1 and depicted in Chapter 5.

Similarly, in the perpendicular direction the profile is the first-order derivative of a Gaussian, $G'(x) = dG(x)/dx$. The maximum occurs at $x = \sigma$ when

$$\frac{d^2G(x)}{dx^2} = -\frac{1}{\sigma^2}\left(1 - \frac{x^2}{\sigma^2}\right) G(x) = 0. \tag{3.6}$$

The equivalent width, considering only $G'(x) > 0$, is

$$W = \frac{\int_0^\infty G'(x)dx}{G'(x)|_{x=\sigma}} = \frac{G(x)|_0^\infty}{-\frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-1/2}} = e^{1/2}\,\sigma. \tag{3.7}$$

In Figure 3.1b, $\sigma = 1.5$, $L = 1.5 \times \sqrt{2\pi} = 3.76$ and $W = 1.5 \times e^{1/2} = 2.473$.

The filters shown in Figure 3.1 were applied to several test patterns. One of the test patterns, *triangle*, together with the edge filter outputs are shown in Figure 3.2. The size of the circle in Figure 3.2a indicates the intensity of the pixel, which ranges from 0 to 1; the length of each line segment in 3.2b and 3.2c indicates the edge strength with orientation as indicated by the line segment. Figure 3.2b and 3.2c also shows that at each sampling location, edge filters of four target orientations were applied.

Table 3.1 lists the difference (labeled as *max diff* and *ave diff* in the table) between the DOB and DDG filter outputs for several test patterns. In the table, *SD* stands for standard deviation of the difference and, like maximal and average difference, is calculated based only on the places and orientations where at least one of the two edge filter outputs is nonzero. The number of these places in each image is also listed in the table. The average difference in the convolution results ranges from 0.01 to 0.07. Compared with the value of an edge strength, which ranges from 0 to 1, the difference between the outputs of the two filters is small.

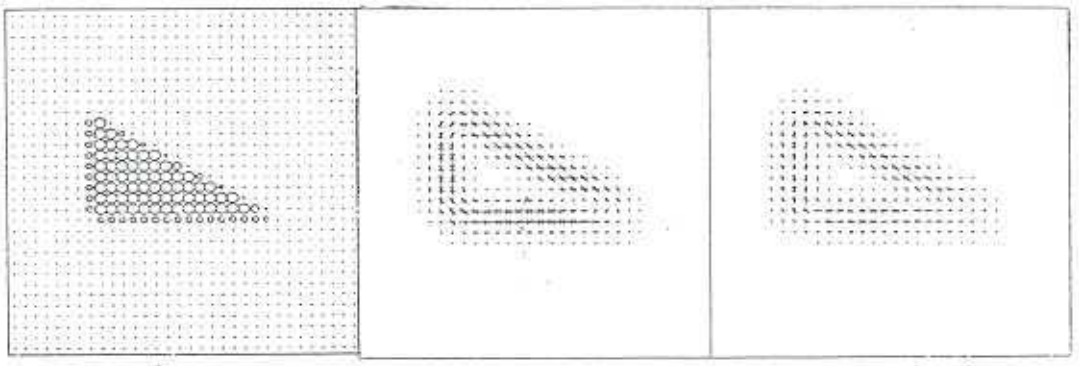


Figure 3.2: A comparison between the convolution results with a derivative-of-Gaussian and a difference-of-box filter. (a) is input pattern; (b) and (c) show the results of DDG filter and DOB filter, respectively. The size of a circle in (a) is proportional to the intensity at the location. Likewise the length of a line segment in (b) and (c) is proportional to the edge strength (i.e., strength of the edge filter response).

The research reported in the remainder of this dissertation uses the DDG as the edge filter kernel because the Gaussian function has the following desirable properties [Pizer, 1988] and accords more with our knowledge of the biological visual system.

1. The Gaussian is isotropic and strictly decreasing about mean.
2. A n-dimensional multivariate Gaussian is separable into n one-dimensional Gaussians.
3. A convolution of two Gaussian functions gives another Gaussian.
4. A Fourier transform of a Gaussian gives another Gaussian.
5. The Gaussian is the solution to the diffusion equation, $\partial u/\partial t = k\,(\partial^2 u/\partial x_1^2 + \cdots + \partial^2 u/\partial x_n^2)$ in n dimensions.
6. The Gaussian blurring operation is causal for image features associated with the linear combination of the derivatives of the image.
7. The central limit theorem says that the effect of repetitive convolution with any kernel will have the effect of convolution with a Gaussian kernel when the repetition increases without bound.

3.2 Issues in Edge Filter Design

This section discusses several questions about the edge filter design including edge threshold, sampling location, angular resolution, polarity, kernel size, and computation of the edge filter kernels.

Edge Threshold

Noise is inherent in the image, and a remedy for it is to insert an *edge threshold* in the algorithm. Below the threshold, the edge filter outputs are ignored. In the current implementation, the edge threshold is a program parameter. It is usually set to a small value, typically 0.03 on a scale of 0 – 1. The reason for selecting a small edge threshold will be discussed further in Chapter 5.

Sampling Location

Where in the image should the edge filters be applied? For an image represented in a two-dimensional array, Figure 3.3a shows two possibilities — one is to sample at the center of the pixel, and the other, at the corners. Figure 3.3b shows the edge filter kernels of target orientation of 0° based on the corner sampling location, and 3.3c on the center sampling location. Figures 3.3e and 3.3f demonstrate the convolution results of the two types of edge filters against an input pattern shown in 3.3d. Given a perfect edge, the figure shows that, for the corner sampling location, positions along a single line in the edge direction have significant output, and for the center sampling location, edge filtering at positions along two parallel lines have significant responses. Edge filtering in my algorithm is applied at the corner sampling location because it is more natural to do so for a step edge such as those in several of the test patterns to be used later.

Angular Resolution

How many edge filters with different target orientations should be applied at each sampling location? One possible answer comes from physiological data. Hubel and Wiesel [1977] showed that a hypercolumn in vertebrate primary visual cortex has adjacent neurons sensitive to orientations differing by 10° to 15°. So a choice of 12 to 18 orientations is reasonable. The effectiveness of filtering in multiple orientations has been confirmed by work in computer vision [Coggins, 1986].

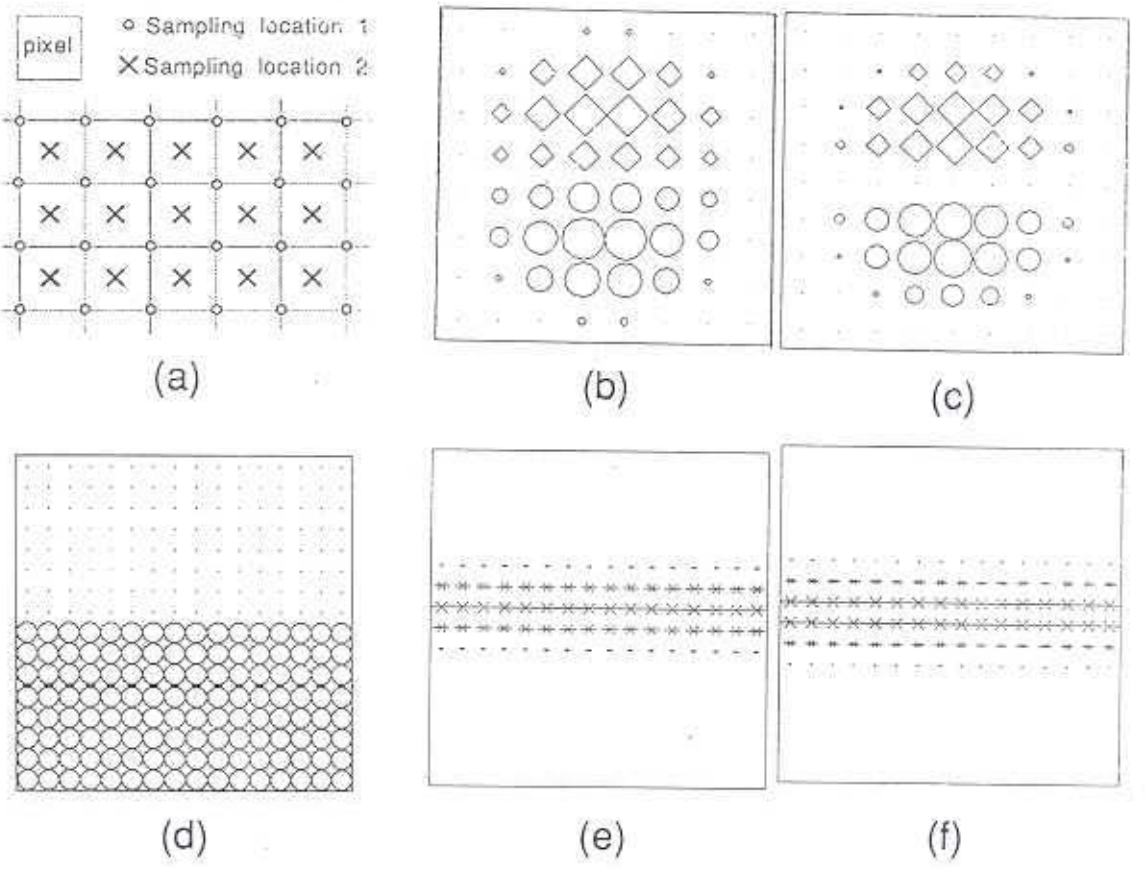


Figure 3.3: A comparison between two schemes of sampling locations. (a) depicts the two sampling locations; (b) and (c) are DDG kernels with $\sigma = 1.5$ pixels of sampling location 1 and 2, respectively; (d) is an input pattern of a horizontal edge; (e) and (f) show the convolution results of (d) with the two edge filters in (b) and (c), respectively.

On the other hand, as mentioned in Chapter 1, the data representation in a connectionist approach is costly. The smallest number of processing units, if functionally adequate, should be used. Since the objective of edge filtering in this research is to find the object contours in the image, the essence of an edge strength lies in its indication of the local orientation of the object boundary. Since, on a rectangular sampling grid, a pixel has only 8 adjacent pixels, the boundary at the pixel can assume one of only four orientations locally. In other words, each of the four orientations points to two of the pixel's 8 adjacent pixels so that the connectivity can be checked point by point on a contour. Edge filtering on orienatations besides the 4 mentioned above are possible but make checking for local connectivity more difficult. Thus the algorithm applies edge filtering in 4 orientations ($0°, 45°, 90°$, and $135°$) at each sampling point. Figure 3.4b show the edge filter outputs of the scheme of 4 target orientations, and 3.4c of 12 target orientations.

A consequence of performing edge filtering in a finite number of orientations is that, according to Canny [1986], if an edge does not align with one of the target edge orientations, the convolution result of the edge will drop to a portion of the case of perfect alignment. For the scheme of 4 target orientations, the worst case occurs when the angle of a real edge and the target orientation of the closest filter is $22.5°$ apart; in that case the filter output will fall to about 75% of its possible maximum. Imperfect alignment of filter to edge causes the effective threshold to be slightly higher than the specified value. For a small edge threshold of, say, 0.03, the effect of imperfect alignment causes the edge strengths below 0.04 to be ignored. The problem is not serious from a practical point of view.

Polarity

The edge filter has two opposite directions of contrast (called *polarity* later) as illustrated in Figure 3.5, where the positive and negative weights are arbitrarily assigned to each half of the filter. The polarity information is represented by the sign of the edge strength and is kept for later processing.

Filter Kernel Size

What standard deviation, σ, should be used for the DDG function? It is well-known that a bigger kernel gives a better signal-to-noise ratio, and a smaller one gives better localization of the detected feature. A clue comes from the psychophysical data.

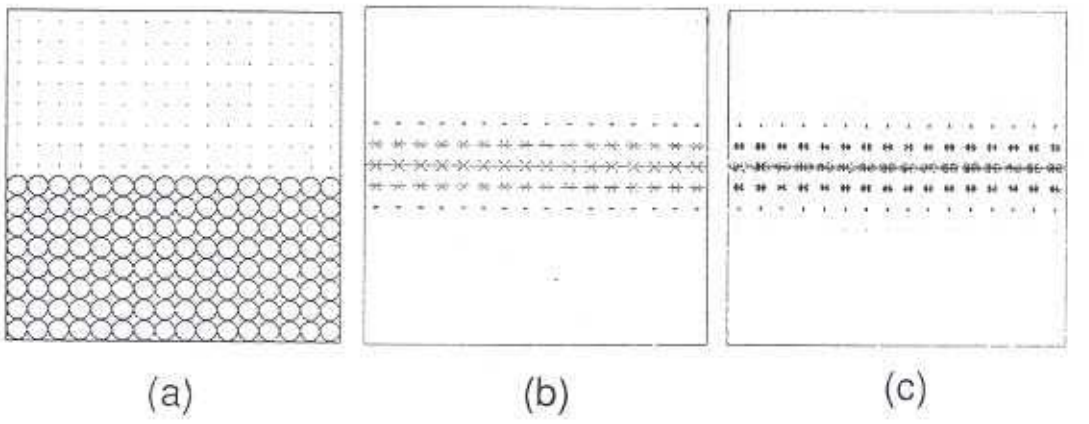


Figure 3.4: A comparison of applying edge filters of 4 and 12 orientations. (a) is the input pattern; (b) and (c) show the convolution results of 4 orientations and 12 orientations, respectively.

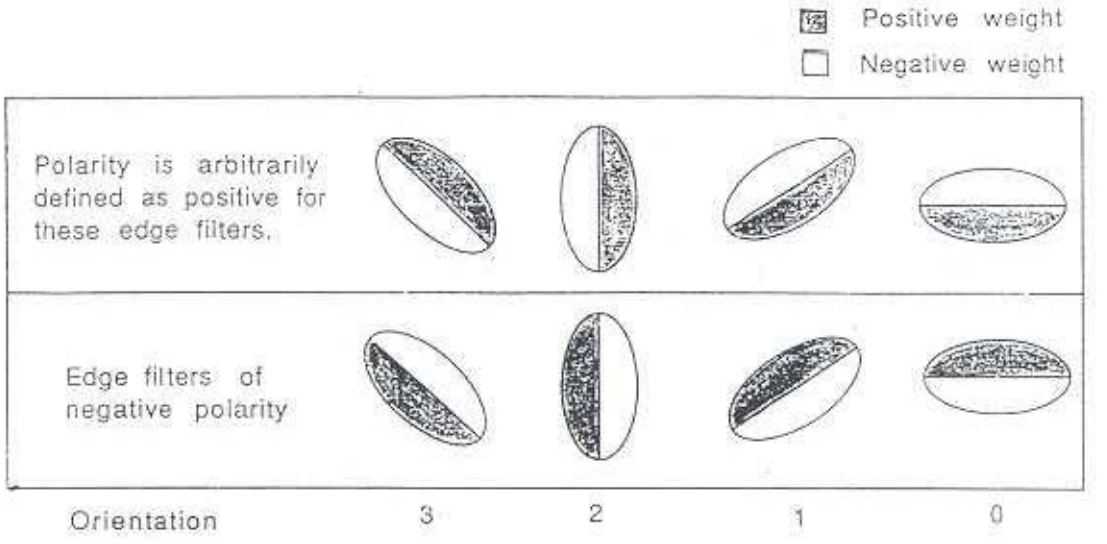


Figure 3.5: The definition of the polarity of edge filters. Edge filters, labeled by integers 0 – 3, are arranged from right to left to reflect the usual representation of counterclockwise rotating angles.

Marr [1982] concluded based on Wilson and Bergen's [1979] data that for cells with on-center, off-surround receptive fields, 4 scales with σ's approximately equal to 3, 6, 12, and 23 pixels suffice to explain many psychophysical phenomena. Marr, Poggio, and Hildreth [1980] further pointed out that an additional smaller filter, say with $\sigma = 1.5$ pixels, is likely. Note that, according to $G(r)/G(0) < 0.05$ to decide the filter support, the DDG filter with $\sigma = 1.5$ has radius equal to 3.67 pixels and covers 42.35 pixels. It is rather large; should there be a smaller one? As shown in Figure 3.6, the filter support of $\sigma = 0.75$ pixels has radius of 1.83 pixels and covers 10.58 pixels. Only one or two pixels at each side of the target orientation have substantial weight. Therefore it gives better possible spatial accuracy than the one with $\sigma = 1.5$ pixels. Nevertheless, when σ decreases, the filter is more sensitive to typical levels of noise. Noise sensitivity is a substantial problem for the filter with $\sigma = 0.75$. It is debatable whether this filter should exist in a real vision system, but it is worth investigating its properties. Thus in this research the filter with $\sigma = 0.75$ pixels is also considered.

A problem related to the kernel size is the sampling interval. Shall we apply edge filtering at every pixel? There are two major concerns: one is the aliasing error, and the other is how the sampling intervals should vary for edge filters of different sizes. As for the aliasing error, according to Pizer [1987] the relationship between the sampling interval, h, and the acceptable relative aliasing error, ε, can be approximated by the equation,

$$h = \frac{\pi\sigma}{\sqrt{2\ln(\frac{2}{\pi\varepsilon})}}. \tag{3.8}$$

Since in the connectionist approach the pixel intensity is represented by the response of a processing unit, subpixel sampling is difficult to implement. The sampling interval is usually in integral pixels. In the current implementation the smallest edge filter in use has $\sigma = 0.75$ pixels and is sampled with interval $h = 1$ pixel. Hence $\varepsilon \simeq 0.04$. Is $\varepsilon \simeq 0.04$ good enough? Considering the biological visual system and the limit is on the dynamic range of the neural circuit, which Barlow [1986] estimated to be 2 orders of magnitude, $\varepsilon \simeq 0.04$ is reasonably small.

Regarding the sampling intervals for edge filters of different sizes, h is proportional to σ according to the above equation. So a filter with a larger σ can have a proportionally larger sampling interval without causing greater aliasing error. Based on the above description and the fact that the test patterns under consideration have highest resolution of 128×128, the σ's of the edge filters are selected to be 0.75, 1.5, 3, and 6 pixels, and

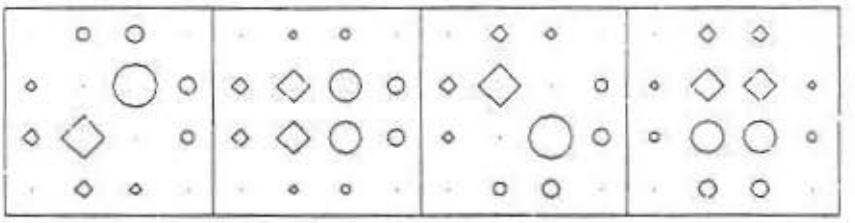


Figure 3.6: The smallest edge filter kernel used in this research.

the corresponding h's are 1, 2, 4, and 8 pixels. Moreover, considering the connectionist architecture, in a coarser scale there are fewer processing units, but each processing unit has proportionally more input connections. That the total number of connections are the same for different scales may contribute to an efficient implementation. The sampling interval is proportional to the size of edge filters in a scale, which is similar to Crowley's DOLP approach [Crowley and Parker, 1984] though the sampling interval in DOLP varies by a factor of $\sqrt{2}$ whereas that factor of my algorithm is 2.

Computation of the Edge Filter Kernel

The edge filter kernel can be calculated as follows. Let $E(x, y; \theta)$ be the edge filter output of orientation θ at location (x, y), and $I(x, y)$ be the input image intensity at (x, y).

$$E(x, y; \theta) = K(x, y; \theta) * I(x, y), \tag{3.9}$$

where $*$ is convolution and $K(x, y; \theta)$ is the edge filter kernel. Evidently,

$$K(x, y; \theta) = \{w(x, y) | w(x, y) = k\, \partial G(x, y)/\partial x_\theta,\ \ x^2 + y^2 \leq r^2\}, \tag{3.10}$$

where, $\partial/\partial x_\theta$ stands for the directional derivative in orientation θ, r is the smallest real number such that $G(r)/G(0) < 0.05$, and k is a normalization constant such that

$$\sum_{(x,y)\ni w(x,y)<0} k\,|w(x, y)| \;=\; \sum_{(x,y)\ni w(x,y)>0} k\, w(x, y) = 1. \tag{3.11}$$

The normalization is necessary because the test pattern is represented by a two-dimensional array of real numbers ranging from 0 to 1 and the normalization causes the convolution with a perfect maximal edge in the image to be 1. With the edge strengths ranging from 0 to 1, we can compare the edge strengths from different edge filters and specify the edge threshold based on a common standard.

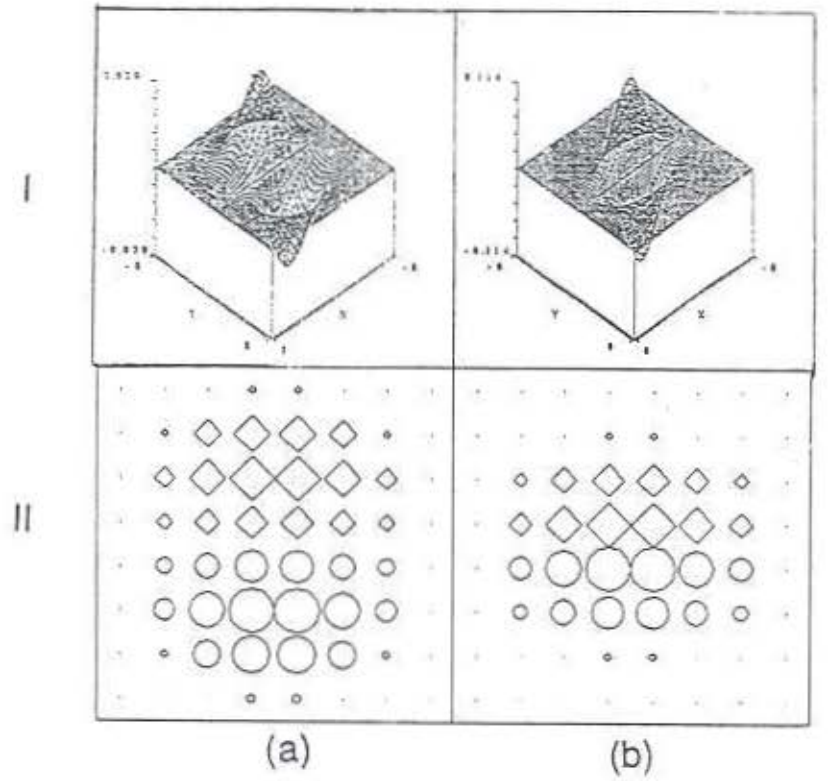


Figure 3.7: The kernels of an elongated and a nonelongated edge filter. Row (I) are three-dimensional line diagrams for edge filters with (a) $\sigma = \sigma_\perp = 1.5$ and (b) $\sigma = 1.5$, $\sigma_\perp = 0.75$. Row (II) are The corresponding edge filter kernels.

To calculate the kernel for different target edge orientation, Koenderink [1987] shows that in $\Re^2$ derivatives at all directions can be calculated by a rotation of coordinates, i.e..

$$\frac{\partial G}{\partial x_\theta}(x, y; \theta) = \frac{\partial G}{\partial x}(x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta). \tag{3.12}$$

3.3 Elongated Gaussian Kernel

The above edge filter was implemented and tested with Gaussian kernels of various σ's, but the performance of the edge filters in multiscale was not satisfactory. The reason is that edge filters are designed to detect the object boundary, and filters at a coarser scale should detect bigger features. However, the above-mentioned filter (equation 3.10) is instead tuned to detect the edges blurred to a certain degree.

To cope with this problem, I used an elongated Gaussian edge filter with widths $\sigma_\perp$. At each sampling point, multiple edge filters with constant width and different lengths are applied. The use of elongated edge filters is justified by Canny [1986], who concluded

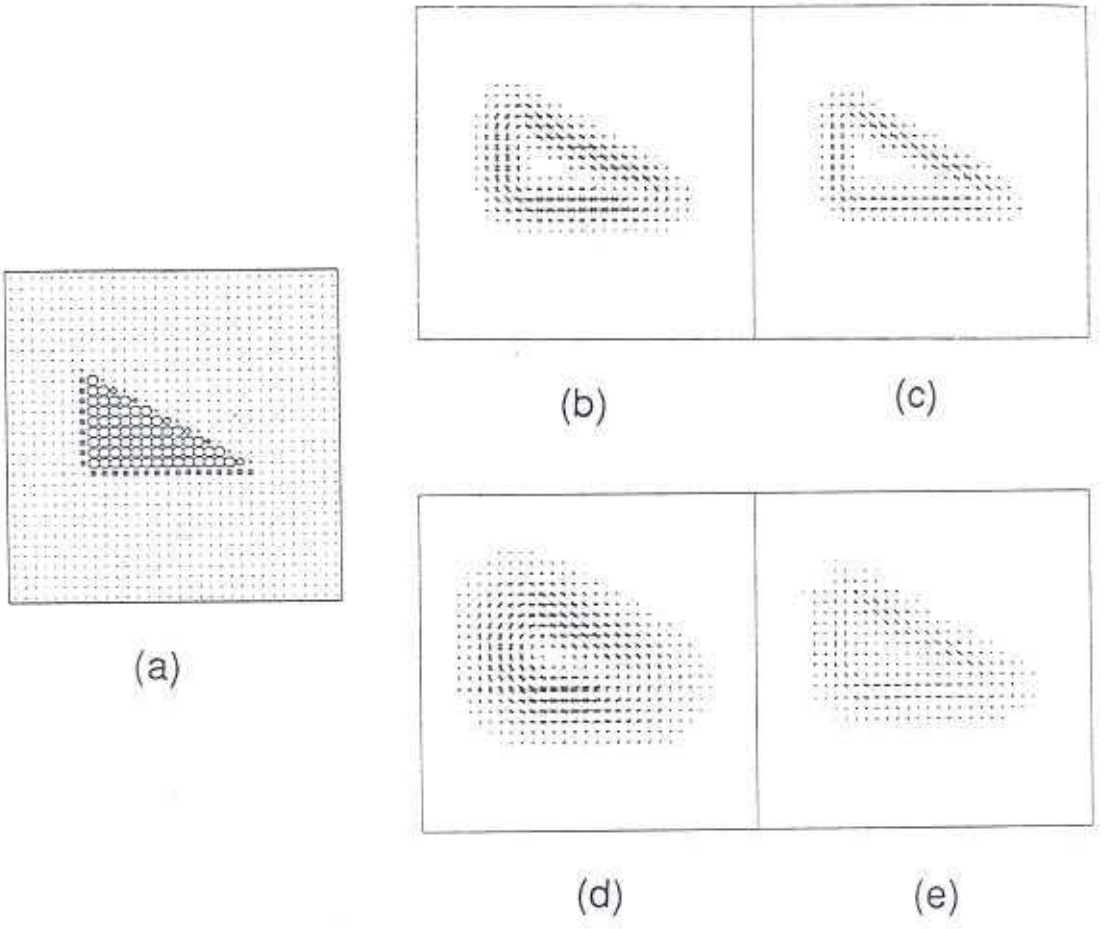


Figure 3.8: A comparison between the convolution results of elongated and nonelongated edge filters. (a) is input pattern; (b),(c),(d), and (e) show results of (a) with edge filters of (b) $\sigma = \sigma_\perp = 1.5$, (c) $\sigma = 1.5$, $\sigma_\perp = 0.75$, (d) $\sigma = \sigma_\perp = 3.0$, (e) $\sigma = 3.0$, $\sigma_\perp = 0.75$.

that an elongated edge filter gives better localization and signal-to-noise ratio of edges in the image. Coggins [1986] used pure orientation filters but no psychophysical support.

Figure 3.7 compares the shape of one of these filter kernels with a nonelongated edge filter. Figure 3.8 compares the convolution results of this filter and a nonelongated edge filter on the triangle input pattern of Figure 3.8a, in which the boundary is not a sharp edge. Note that in Figure 3.8b – 3.8d the object boundary is measured as many parallel edge strengths instead of a sharp contour. Two reasons contribute: the edge is not sharp, and the edge filter width extends over more than 1 pixel.

The kernel of an elongated edge filter is the first-order directional derivative of a multivariate normal function [Duda and Hart, 1973]. The following describes an example of the elongated edge filter kernel. Let σ be the standard deviation in the target orientation and $\sigma_\perp$ the standard deviation in the direction perpendicular to the target orientation. The elongated Gaussian function for target orientation of 0° can be represented as

$$F(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \times \frac{1}{\sqrt{2\pi}\sigma_\perp} e^{-\frac{y^2}{2\sigma_\perp^2}}, \tag{3.13}$$

and the edge filter with target orientation of 0° is

$$\frac{dF(x,y)}{dy} = -\frac{y}{\sigma_\perp^2} F(x,y). \tag{3.14}$$

The edge filter of other target orientations can be calculated based on this function and a rotation of coordinates.

As for the relation between σ and $\sigma_\perp$, the filters with $\sigma < \sigma_\perp$ are not useful because edges are by definition a sharp change in the direction perpendicular to the target orientation. On the other hand, a very long kernel is unnecessary because there are not many long linear edges in the image. The highest ratio of $\sigma/\sigma_\perp$ used in the current implementation is 4 based on psychophysical evidence — the bandwidth of the human spatial-frequency channels is 1 to 2 octaves [Ginsburg, 1978; Wilson and Bergen, 1979], so in the spatial domain a factor of 2 to 4 among the filter sizes may be reasonable.

3.4 Artifact Cancellation

Grossberg and Mingolla [1986, 1987], in designing the *boundary contour system*, found that boundary completion was more satisfactory if the magnitude of the edge

strength is decreased by the magnitude of the edge strength of perpendicular target orientation. Applying a program implementing the *boundary contour system* to several test patterns confirms that this operation is important for the success of the algorithm.

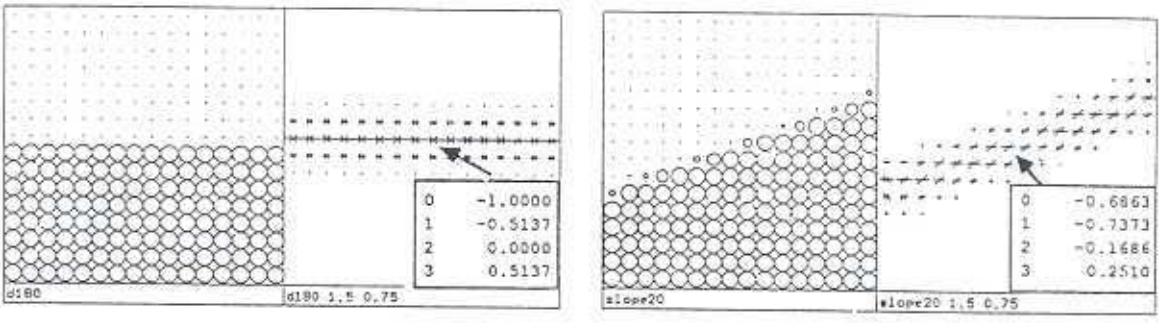
Why is the above-mentioned operation important? A clue comes from a more careful examination of the relationship between a real edge and the four edge filters applied at a sampling point. A real edge in the image can orient with an edge filter perfectly, or it can orient between two edge filters with successive target orientations.

I applied edge filtering on two edges: one oriented at 0° and the other at 20° and found that more than 2 out of the 4 edge filters always gave significant responses. Figure 3.9a shows an edge in perfect alignment with a edge filter of target orientation 0°; 3.9b shows an edge at an orientation between two edge filter orientations. The convolution results with an edge filter of $\sigma = \sigma_\perp = 0.75$ for the two differently oriented edges at indicated point are also shown in a table in each figure. At the indicated sampling point 3 out of 4 edge strengths are significant for the edge oriented at 0°, and all 4 edge filters give significant responses for the edge oriented at 20°. At each sampling point, the presence of more than two edge filters responding to a real edge contradicts the intuition that a real edge should be indicated by at most two edge filters of successive target orientations. This constraint is important because the wrongly-indicated edge strengths, if used in a later stage of segmentation, will cause errors.

Grossberg and Mingolla's subtraction approach is an effective way to remedy this problem. It involves a subtraction from the magnitude of the edge strength under consideration ($E(x,y;\theta)$) of the edge strength of the perpendicular orientation ($E(x,y;\theta_\perp)$) and accepting the result only if it is positive. Let $A(x,y;\theta)$ be the result after artifact cancellation:

$$A(x,y;\theta) = sign(E(x,y;\theta))\ \max(0, |E(x,y;\theta)| - |E(x,y;\theta_\perp)|). \qquad (3.15)$$

Figures 3.10a and 3.10b show how two perpendicular edge filters are applied to an edge. For the filter outputs with target orientation right on edge, as in Figure 3.9a and 3.10a, the edge strength is not affected. Edge strengths of all other situations are adjusted downward. For the examples in Figure 3.9, after artifact cancellation the edge strengths become (-1, 0, 0, 0) and (-0.5177, -0.4863, 0, 0) for the edge of orientation 0° and 20°,



(a) (b)

Figure 3.9: Two possible relations between edge filters and an edge in the image. (a) an edge aligned perfectly with an edge filter, and (b) an edge oriented (20°) between two edge filters. (Numerals 0, 1, 2, and 3 in the rectangle indicates the edge strengths of orientations 0, 1, 2, and 3, respectively.)

respectively. These edge strengths give more accurate information about the edges under consideration.

Coggins [1986] showed that a more accurate estimation of the boundary orientation can be obtained by a vector sum over all orientations. However, for a connectionist approach the representation of the result of the vector sum requires many processing units (neurons). Moreover, to verify the connectivity under the postulated conditions, there is no need to determine the exact orientation at each boundary point — only the neighboring points to which this pixel is connected need to be determined.

The only place artifact cancellation may destroy useful information is around a right-angled corner as illustrated in Figure 3.10c. However, since corners are separately detected in my algorithm, the loss of this information does not affect the detection of a right-angled corner.

Figure 3.11 shows the convolution results before and after artifact cancellation. Many edge strengths are diminished and a better representation of the input pattern is obtained.

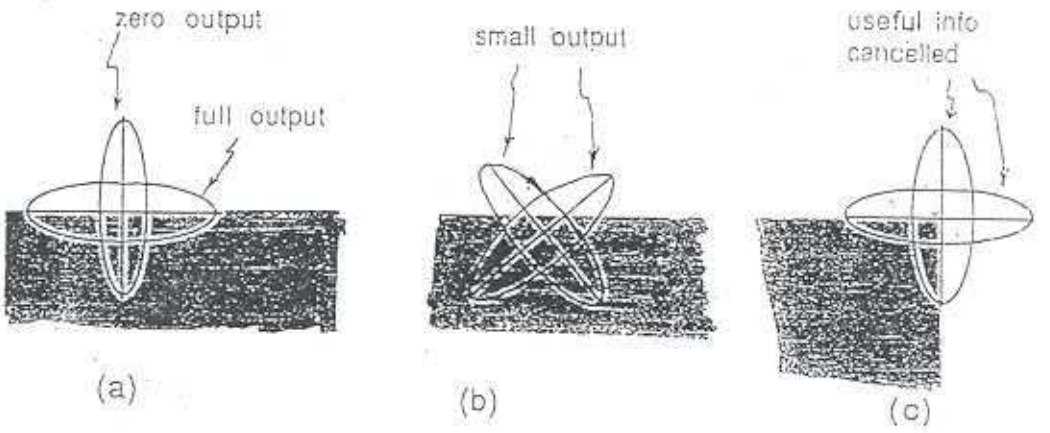


Figure 3.10: The artifact generated by edge filters. (a) the target orientation is right on edge, (b) two filters with target orientation perpendicular to each other, and (c) the situation when the artifact cancellation process may destroy useful information.

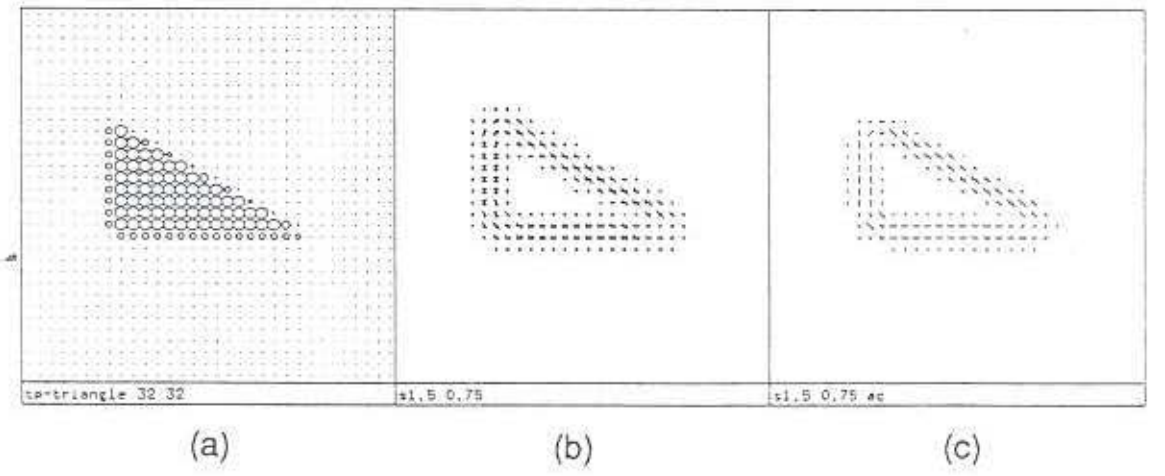
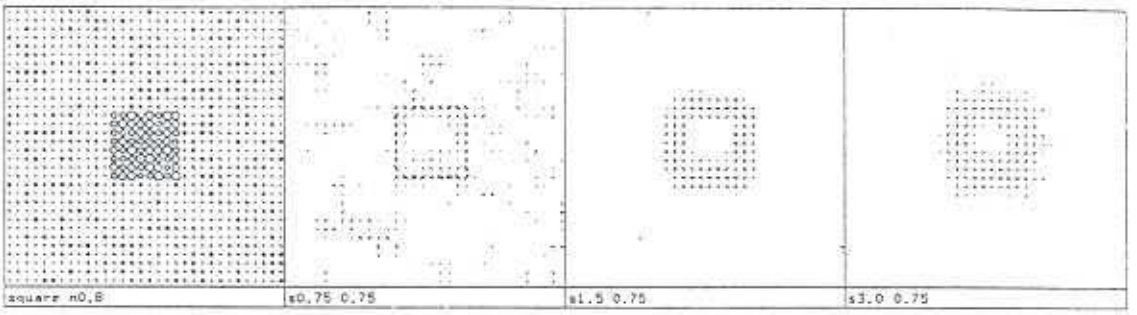


Figure 3.11: The effect of *artifact cancellation*. (a) is the input pattern; (b) and (c) show the convolution results with edge filtering of $\sigma = 1.5$ and $\sigma_\perp = 0.75$ pixels before and after artifact cancellation.



(a) (b) (c) (d)

Figure 3.12: The convolution results of a noisy image (a) with filters of (b) $\sigma = \sigma_\perp = 0.75$, (c) $\sigma = 1.5$, $\sigma_\perp = 0.75$, and (d) $\sigma = 3.0$, $\sigma_\perp = 0.75$.

3.5 Multiple Edge Filters at a Sampling Location

In order to attain an object boundary under various levels of noise, multiple edge filters need to be applied at each sampling point. A small edge filter behaves better for images without noise because fewer false edges are detected and valid edges will be better localized. A small edge filter can also detect short edges; segmentation based on their outputs shows the details of the contour. Unfortunately, when noise is present, using a small edge filter often runs into problems. Figure 3.12 demonstrates how three edge filters (one isotropic and two elongated, with fixed $\sigma_\perp$) respond to a noisy image. Evidently, it is beneficial to use edge filters with larger support for their higher noise immunity. Since an input image can have various levels of noise, a compromise is to apply multiple edge filters at a sampling point. Figure 3.13 shows the kernels of three edge filters, and Figure 3.14 illustrates the supports of multiple-sized edge filters of target orientation of 0° applied at a sampling point.

Another important consideration for using the multiple, elongated edge filters is how to complete the gaps between two linear edges as in the subjective contour of the Kanizsa square. The fact that a longer edge filter responds beyond a linear edge is necessary for the completion of the Kanizsa square. However, a given elongated edge filter can propagate the edge only to a certain extent. With a fixed number of elongated edge filters at a particular scale, the gap that a scale can complete is limited. This is one of the reasons

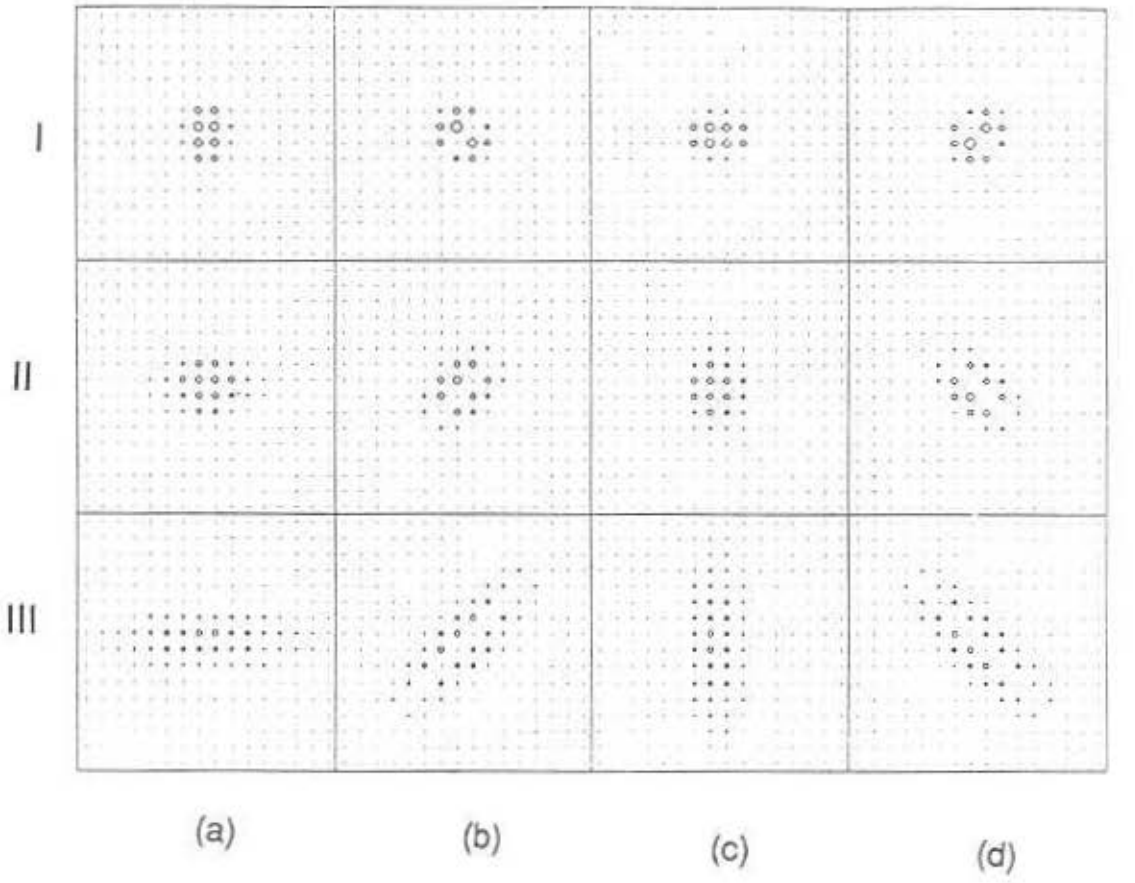


Figure 3.13: The kernels of the multiple edge filters. Row (I), (II), (III): $\sigma_{\perp} = 0.75$ and $\sigma = 0.75, 1.5, 3.0$, respectively; Column (a),(b),(c), and (d) show the target edge orientations of 0°, 45°,90°, and 135°.

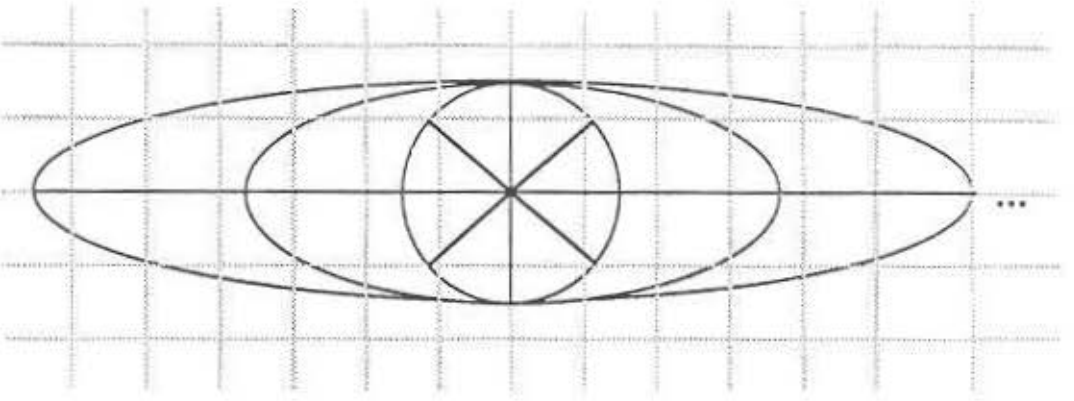


Figure 3.14: Multiple edge filters applied at a sampling point.

that the algorithm contains multiple segmentation mechanisms as shown in Figure 2.5 — a coarser scale with $\sigma/\sigma_\perp$ equal to that of a finer scale can complete a bigger gap. The effect of these mechanisms will be described in more detail in Chapter 5.

The integration of information from multiple scales has long been an open research topic. The following summarizes several such schemes. Canny [1986] used *feature synthesis* to decide whether a pixel is an edge point: if the small edge filter output at the location is above threshold, the pixel is deemed an edge; if not, the output of a bigger filter is compared with a value calculated based on the edges indicated by the small edge filters within the support of the big filter. If the two values are significantly different, the pixel is still marked as an edge point. This scheme gives reasonable performance, but extra filtering and more complicated control would be required for hardware implementation. Besides, an edge detected by the Canny edge detector sometimes shifts from its original position. Bergholm [1988] applied the Canny edge detector in multiscale and traced down the scale to attain better localization. Hildreth [1983] suggested that a linear interpolation scheme of the multiscale information might explain human hyperacuity, the ability to detect differences in spatial position smaller than the sampling distance of foveal receptors. Finally Coggins [1986] suggested an interesting method to integrate information from different scales: multiscale information is first mapped onto a feature space based on which effective image analysis, and object recognition can then be done.

A possible usage of the multiple edge filter outputs is to verify the "edgeness" at the sampling point. The following observations on the relationships among the 16 edge

filter outputs at each sampling point (4 sizes with each of 4 orientations) may help.

- For the four edge filters of different orientations and the same size, the polarities of the filter outputs should be coherent.
- Two edge filters of successive sizes should give edge strengths of similar edge direction and magnitude.
- For an edge segment shorter than the kernels of edge filters, the corresponding edge strengths should decrease smoothly with the increase of edge filter size.

Note that the above heuristics can help in identifying a real edge, but they are not necessarily true for a noisy image or for the sampling points near a more complicated image structure like a corner. In my implementation the edge filter outputs were averaged. Running the algorithm against several test patterns, this simple scheme gave good segmentation on many images.

Chapter 4

Corners

This chapter deals with the problem that directional edge filters cannot correctly indicate the object boundary near more complicated image structures like corners, T-junctions, and cross-junctions, which are important for object boundary detection. To cope with this problem, I propose a scheme which detects these image structures based on patterns of edge strengths.

This chapter first introduces the problem of corner detection and then describes a solution. The results of applying this scheme to several test patterns are then presented. The chapter concludes with a discussion of possible improvements to the algorithm.

4.1 Introduction to Corner Detection

Corners are contour points with infinite curvature, where

$$curvature = \lim_{\Delta s \to 0} |\frac{\Delta \theta}{\Delta s}|, \tag{4.1}$$

with θ being the inclination angle of the tangent line and s being distance along the curve.

Corners are important characteristics of an object-bounding contour in a two-dimensional image. Besides the psychophysical evidence [Attneave, 1954], this point can be justified from two directions: the inadequacy of edge filtering near a corner and the insufficiency of using context-free edge strengths to define object boundaries. Edge filtering near a corner causes problems because an edge filter in this algorithm is a combination of a smoothing operator and a differentiation operator. The smoothing causes a sharp corner to be rounded, and then the first-order directional derivative can not detect the corner. Figure 4.1 illustrates the edge filtering near a corner. At sampling location 0 in the figure the edge strengths of all four orientations are nonzero. There is no way to determine the object boundary based on the edge filter outputs at a single sampling point.

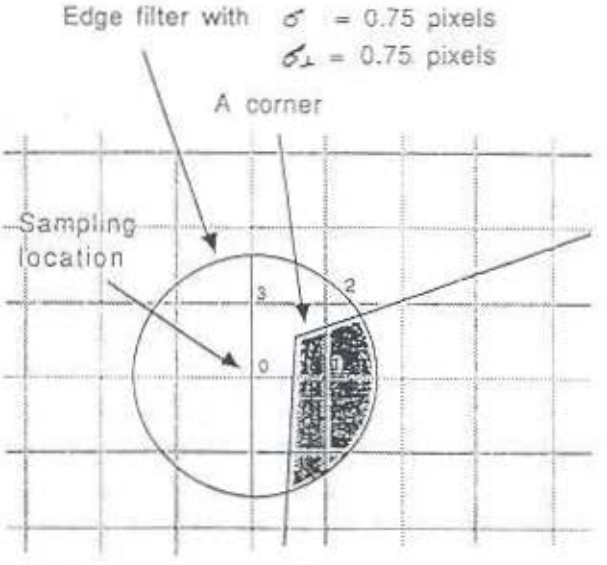


Figure 4.1: Edge filtering near a corner.

The second problem is that edges alone are unlikely to define a closed contour properly. If corners are not specifically detected, connectivities between nearby edges are hard to define.

Figure 4.2 – 4.5 illustrate various kinds of corners, T-junctions, and cross-junctions with the corresponding edge filter outputs. Means of detection of these image structures will be given later in this chapter, and the results of segmentation will be illustrated in Chapter 5. Figure 4.2 shows the edge filter outputs for corners of opening angles 15°, 30°, 60°, 90°, and 150°.

In each of the input patterns in Figure 4.2I, there is an ideal corner with the tip at (8,8) on a 16 x 16 grid. The sizes of the circles in the diagram indicate the portion of the pixel covered by the corner. It is obvious that there exists the problem of discrete representation: in Figure 4.2Ia the corner of 15° cannot be precisely represented, and the lost information cannot be recovered by edge filters. Moreover, the pattern of edge filter outputs near a corner are different from those near a linear edge. It follows that when integrating edge strengths to indicate an object boundary, different rules need to be used for locations near a corner and locations near a linear edge.

Figure 4.3 shows the edge filter outputs for T-junctions, and Figures 4.4 and 4.5 show the edge filter outputs for various cross-junctions. Near these image structures the edge filter outputs deviate from those of a linear edge owing to the symmetry that

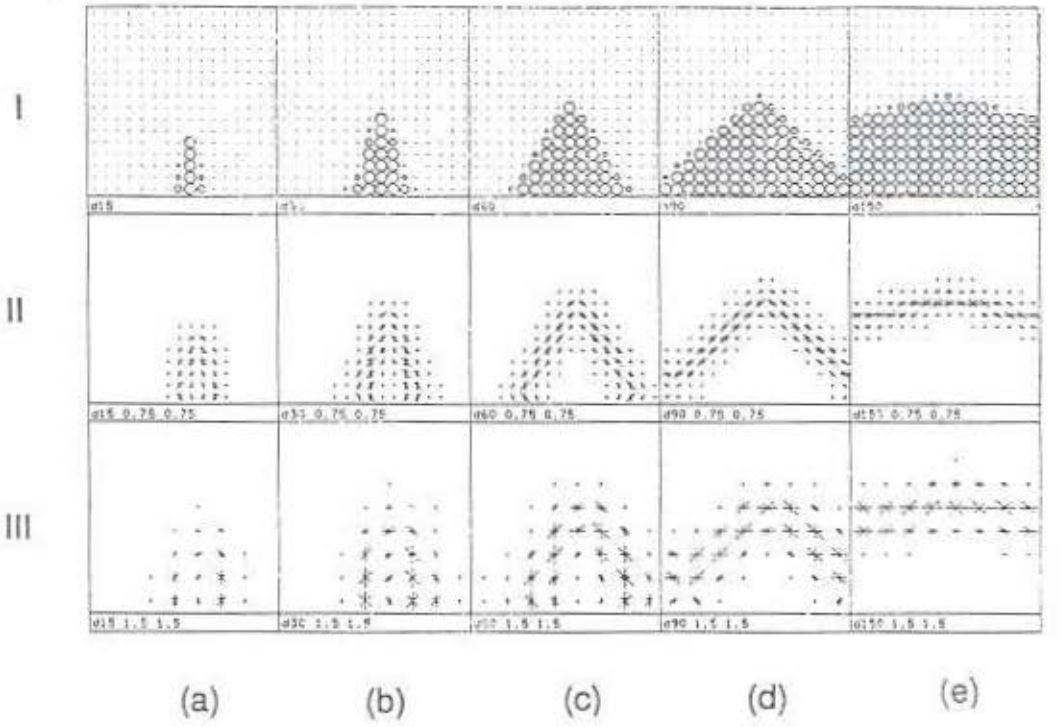


Figure 4.2: The results of edge filtering near corners. Row (I): input patterns; Row (II): output of edge filters with $\sigma = \sigma_\perp = 0.75$; Row (III): output of edge filters with $\sigma = \sigma_\perp = 1.5$; Column (a)(b)(c)(d)(e): corners of 15°, 30°, 60°, 90° and 150°, respectively.

these image structures impose on the left and right halves of the edge filter kernel. This phenomenon is especially evident for Figure 4.5d, wherein the edge filter outputs at (8,8) vanish.

It should be clear that corners are important for defining the object boundary, but how can they be detected based on the input of the visual system — a two-dimensional intensity array? Many approaches have been proposed for corner detection [Asada and Brady, 1986; Baugher and Rosenfeld, 1987; Davis, 1977; Dobbins, Zucker, and Cynader, 1988], but most of these methods assume that a given contour exists and the algorithm determines where along the contour the corners reside. From the viewpoint of a vision machine, the problem is more difficult. The contours of objects in the scene are not given, yet the corners need to be located.

4.2 Detection of Corners and Related Image Structures

Corners, T-junctions, and cross-junctions are detected by separate parallel processing mechanisms based on edge information. This section first elaborates on the derivation of the mechanism for corner detection. Then it describes how a similar method can be applied to detect T-junctions and cross-junctions.

4.2.1 Corners

Corners can be detected by template matching, but this approach is impractical. A simple calculation on the number of processing units required for a possible corner template scheme shows this point: A corner can have various opening angles and opening directions. Since there is no way to know where there will be a corner in the image, there must be a corner template of every possible opening angle and opening direction everywhere in the visual field. A conservative estimate is based on the assumptions that the resolution of the image is 1000 by 1000 and corners can open to ten different directions and have ten different opening angles. Without counting polarity, the required number of corner templates is $10 \times 10 \times 10^3 \times 10^3 = 10^8$, which is near the total number of neurons in area V1 of the visual cortex [Wiesel and Hubel, 1977].

The derivation of my corner detection scheme is based on two properties of a corner: a corner is a local property and corner detection is a second-order property of edges. As regards the locality, since template matching is improbable, a natural question is *can the*

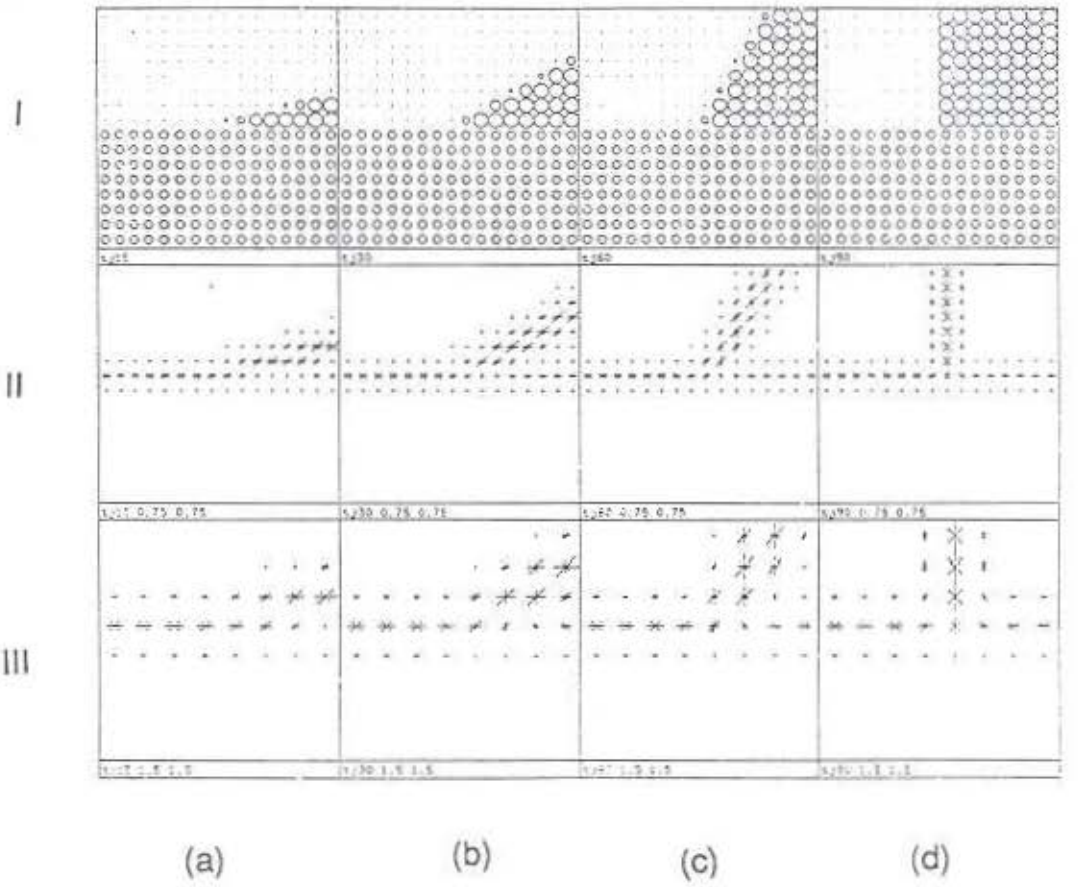


Figure 4.3: Edge filter output near T-junctions. Row (I): input patterns; Row (II): output of edge filters with $\sigma = \sigma_\perp = 0.75$; Row (III): output of edge filters with $\sigma = \sigma_\perp = 1.5$; Column (a)(b)(c)(d): T-junctions of 15°, 30°, 60°, and 90°, respectively.

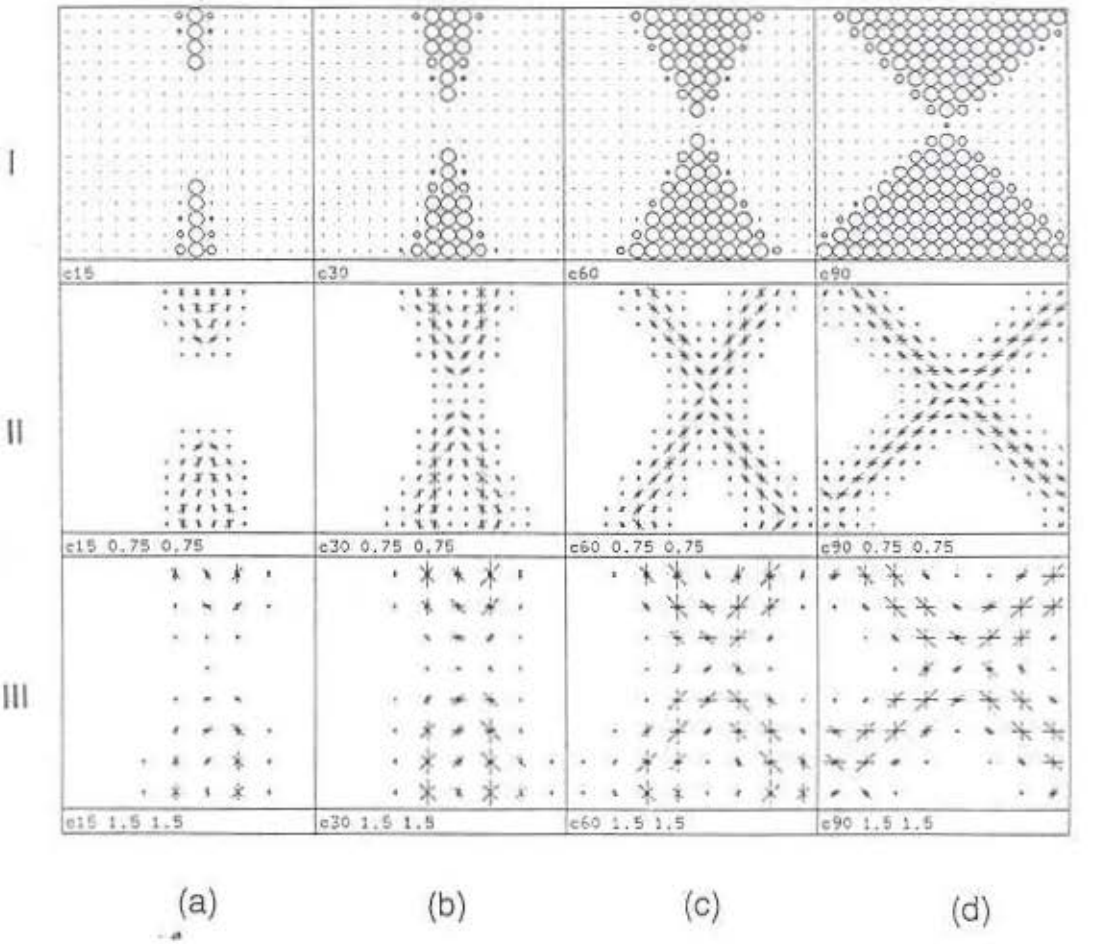


Figure 4.4: Edge filter output near cross-junctions. Row (I): input patterns; Row (II): output of edge filters with $\sigma = \sigma_\perp = 0.75$; Row (III): output of edge filters with $\sigma = \sigma_\perp = 1.5$; Column (a)(b)(c)(d): cross-junctions of 15°, 30°, 60°, and 90°, respectively.

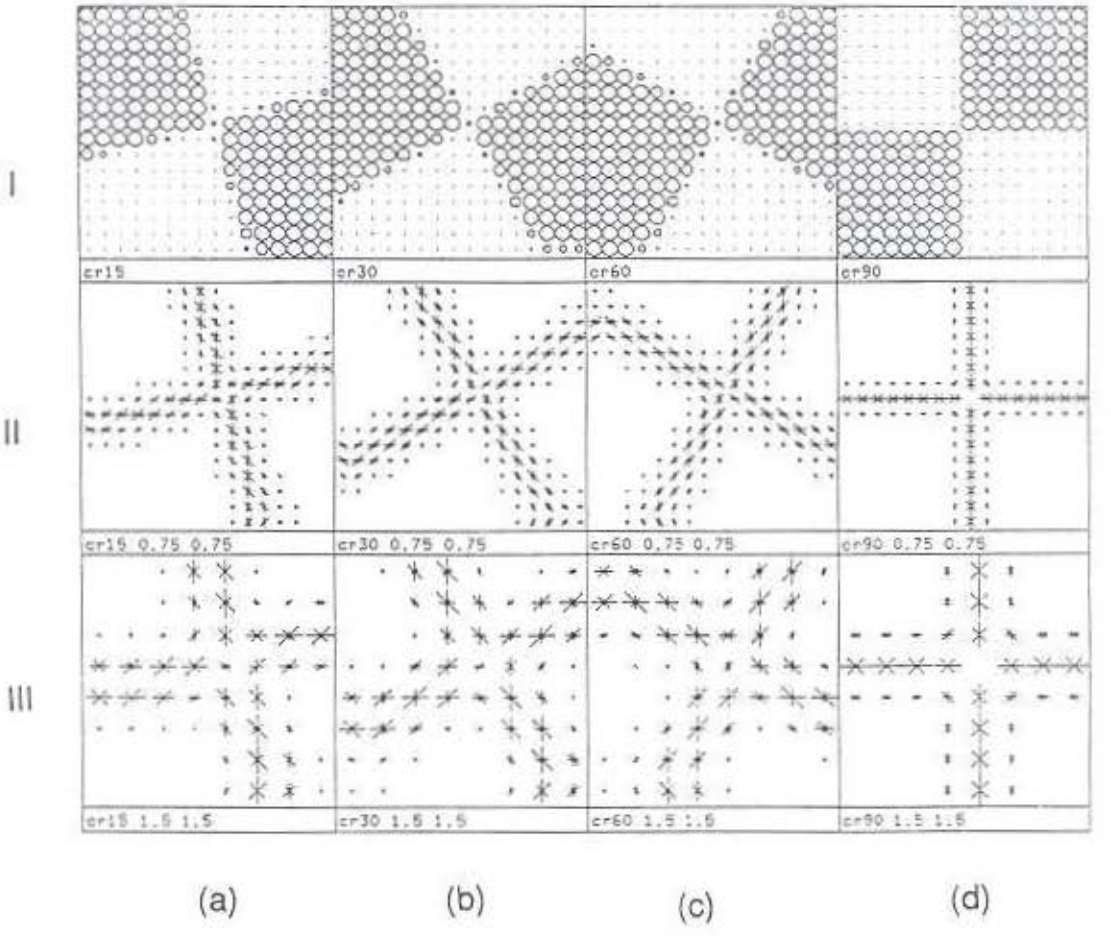


Figure 4.5: Edge filter output near oblique cross-junctions. Row (I): input patterns; Row (II): output of edge filters with $\sigma = \sigma_\perp = 0.75$; Row (III): output of edge filters with $\sigma = \sigma_\perp = 1.5$; Column (a)(b)(c)(d): cross-junctions rotated by 15°, 30°, 60°, and 90°, respectively.

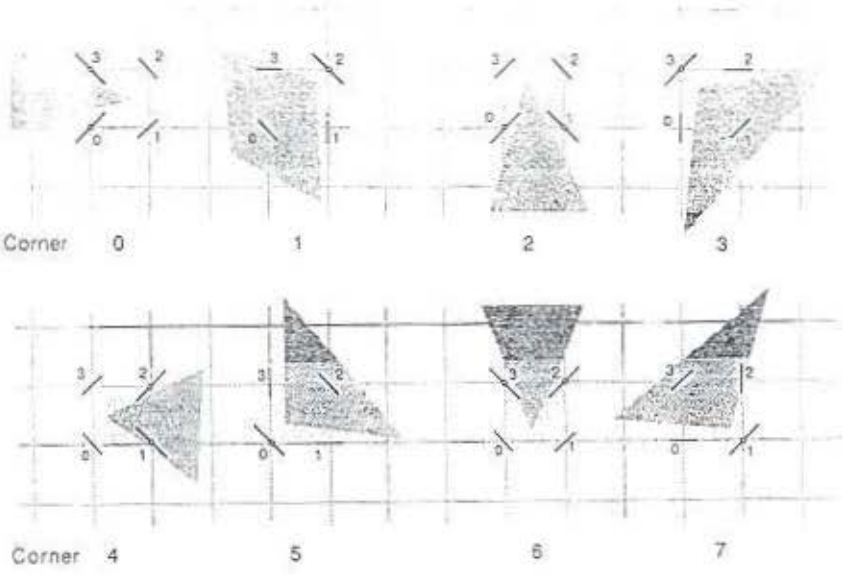


Figure 4.6: The edge strength patterns near a corner. They are labeled by 0 – 7 in the figure and differ mainly in the corner's opening direction.

corners be detected from edge information? Some psychophysical evidence indicates that humans detect corners after edges [see Barlow, 1983; Frisby, 1980]. So, if a corner is a place where two edges meet, whether a corner exists within a pixel can be decided by edge filter outputs in the pixel's neighborhood.

Corner detection can be based on a second-order statistic of edges. For a corner residing within a pixel, the polarities of the edge strengths sampled at the neighboring points have a certain pattern of relationships. For example, let $E(loc; k)$ represent significant edge strength of orientation k (ranging from 0 to 3) at location loc; in the diagram of corner type 0 in Figure 4.6, $E(0;1)$ and $E(3;3)$ have opposite polarities.

Studying the edge filter outputs at the four corner points of a pixel, we see that there are only two kinds of edge strength patterns if a corner is within the pixel: one with the corner pointing to the direction between two sampling points and the other with the corner pointing directly to a sampling point. Each kind has four possible arrangements as illustrated in Figure 4.6, where the eight possible edge strength patterns are labeled by 0 – 7 according to the direction in which the corner points. To analyze a pixel for containment of a corner, the edge filter outputs at the 4 sampling locations on the pixel, marked by 0 – 3, are checked. A small line segment at each of these sampling points

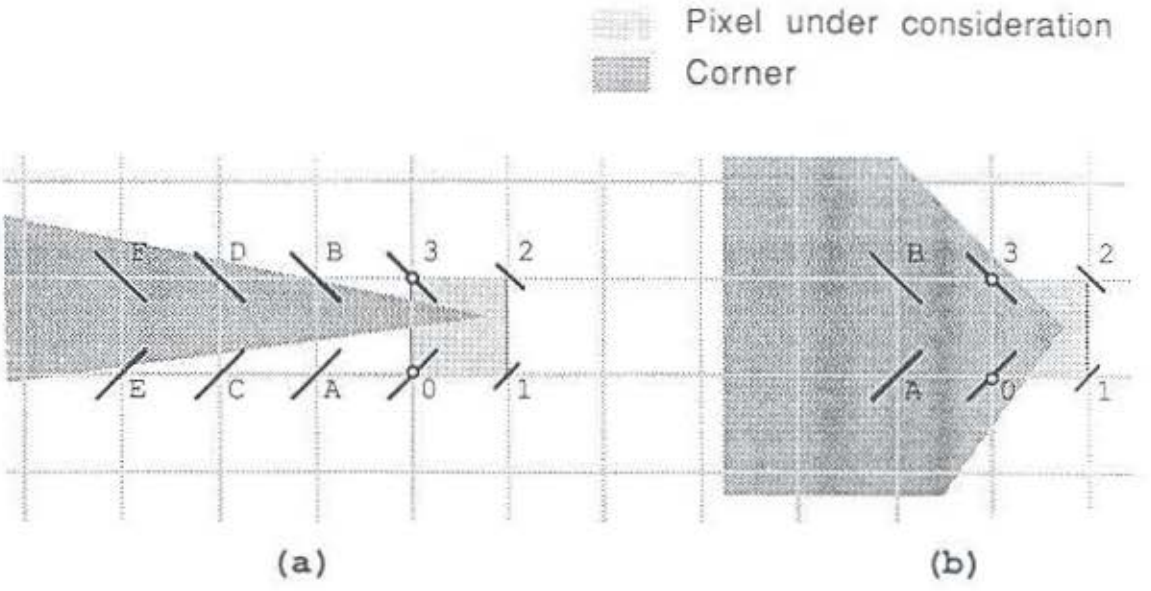


Figure 4.7: The edge strengths near a type 0 corner shown in more detail.

indicates a significant edge strength at the orientation of the line segment. For each of the eight edge strength patterns shown, the locations marked with a small circle are deemed to be corners.

The two kinds of corners in Figure 4.6 are labeled by odd and even numbers, respectively. Let us take corner type 0 as an example of a horizontal or vertical (even numbered) corner and corner type 1 as an example of a diagonal (odd numbered) corner. The following argument about type 0 and 1 corners, respectively, generalizes to the other corners in their category.

For corner type 0 the pair of edge strengths $E(0;1)$ and $E(3;3)$ should have opposite polarities, and so should the pair $E(1;1)$ and $E(2;3)$. However, for a type 0 corner shown in Figure 4.7 (especially for a long, sharp one as in 4.7a), the polarity relationship described above holds for several pairs of edge strengths along the 0° direction, e.g., $E(A;1)$ and $E(B;3)$, $E(C;1)$ and $E(D;3)$, and $E(E;1)$ and $E(F;3)$ in 4.7a. Which locations should be recognized as corner points? A solution is to take into account the magnitude of the edge strengths — the edge strengths at the desired corner points 0 and 3 are usually the largest compared with those at locations 1, 2 and A, B. To accomplish this we define a function $S(E_1, E_2)$ of the strength of a pair of edges with strengths E_1 and E_2. For example, S may be the maximum function.

But how many pairs of edge strengths should be considered to find the pair with the largest magnitude? To avoid comparing the edge strengths at several pairs of locations, an approximate scheme based on the following observation was adopted: considering the strengths of edge pairs in Figure 4.7a, from left to right the strengths tend to increase slowly until they decrease abruptly just beyond the corner. Therefore, a subtraction and a test of sign, applied only on the four edge strengths around the pixel under consideration, was applied — at each pixel where $E(0;1) \times E(3;3) < 0$ and $E(1;1) \times E(2;3) < 0$ the scheme checks that whether $\max(E(0;1), E(3;3)) - \max(E(1;1), E(2;3)) > 0$. If so, the pixel is deemed as a corner point. The scheme works because beyond the tip of the corner the edge strengths of proper orientation drop quickly. Hence the multiplication for the polarity check gives 0 and prohibits the acceptance of the pixel.

In summary, the corner strength $C(loc;type)$ of corner type, *type*, at the location, *loc*, can be calculated based on the following equation.

$$C(loc;0) = \begin{cases} \max(0, \max(E(0;1), E(3;3)) & \text{if } E(0;1) \times E(3;3) < 0 \ \& \\ \qquad - \max(E(1;1), E(2;3))) & \quad E(1;1) \times E(2;3) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Note that the above equation implies that the edge strengths $E(1;1), E(2;3), E(0;1)$, and $E(3;3)$ should also be significant, i.e. larger than an edge threshold (E_thd); otherwise either $E(0;1) \times E(3;3) = 0$ or $E(0;1) \times E(3;3) = 0$.

For corner type 1, obviously $E(0;3)$ and $E(2;3)$ should have opposite polarities.

$$C(loc;1) = \begin{cases} \max(E(0;3), E(2;3)) & \text{if } E(0;3) \times E(2;3) < 0 \ \& \\ & \quad E(1;2), E(3;0) > E\_thd \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

By symmetry, similar statements can be made about corner types 2, 4, and 6, and 3, 5, and 7. By appropriate rotation of one of the two cases above, an expression can be given for the other corner types.

A program detecting these patterns based on the above definition was implemented. At each sampling point only the edge strengths from the smallest edge filter (e.g., $\sigma = \sigma_\perp = 0.75$ for the finest scale) were used for corner detection because being a corner is a local property whereas the convolution results of a bigger edge filter show the property of a more global area and are hence improper for locating corners. Application of the

scheme to several test patterns shows that most of the sampling locations marked by a small circle in Figure 4.6 are satisfactorily detected as corners.

The algorithm has two major advantages. First, the algorithm is defined on the polarity of the edge filter outputs, so it is more stable against noise than the method based on the numeric values of edge filter outputs. Second, the algorithm depends only on local information and is simple. Not only is an efficient implementation on a conventional computer architecture possible, but also a construction of a connectionist implementation is feasible.

Figure 4.8 shows a possible connectionist implementation of this algorithm. There are four layers: input representation, edge filtering, edge pair comparison, and corner representation. In the input layer each circle indicates a pixel intensity of the image; each cross indicates where edge filtering is to apply. Edge filters measure the local intensity changes over the image regularly. At each sampling point there are eight edge filters differing in target orientation or polarity. The wiring between the edge filter layer and the input layer is not shown to simplify the figure. Each neuron in layer 3 has inputs from edge filters at two sampling locations. An example of the connections between layer 2 and layer 3 (marked with dotted wiring and a shaded circle) is illustrated in more detail in Figure 4.8b, which shows how both orientation and polarity contribute to corner detection. Each neuron in layer 4 has inputs from neurons in layer 3. The firing of the layer-4 neuron indicates that there is a corner within the pixel surrounded by sampling points $1 - 4$ in the input image layer. Since there are eight corner types, the neurons shown in layer 4 need to be replicated 8 times.

4.2.2 T-junctions

Figure 4.3 shows that edge filters do not give proper edge strengths at the object boundary near a T-junction. Studying the edge strength patterns shown in Figure 4.3, i.e., a horizontal boundary intersected by another one at a certain angle (other T-junctions give similar results), the intersecting boundary causes the horizontal edge filters near the T-junction to give smaller outputs, while vertical edge strengths appear. As illustrated in Figure 4.9a, this effect peaks at the position of intersection and decreases smoothly with increasing distance to the junction. Another way to describe this behavior is that, if the edge strengths at a single sampling point defines a local *edge direction*, near a T-junction

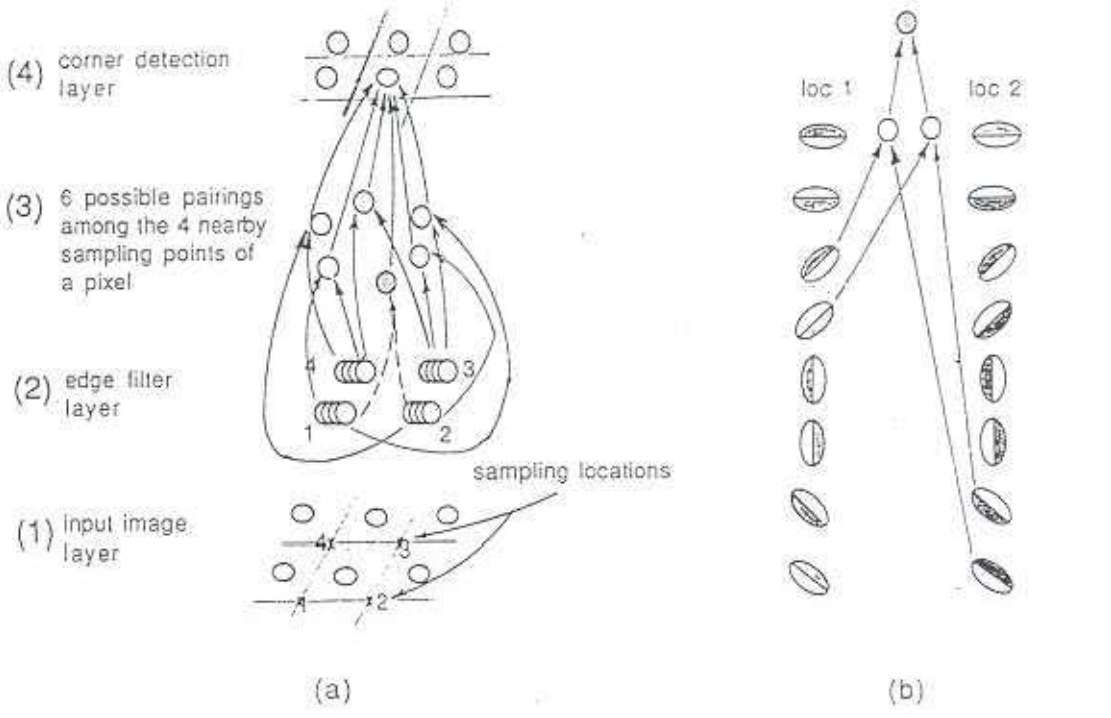


Figure 4.8: (a) A possible connectionist implementation for corner detection. (b) A more detailed illustration of connections from layer 2 to layer 3.

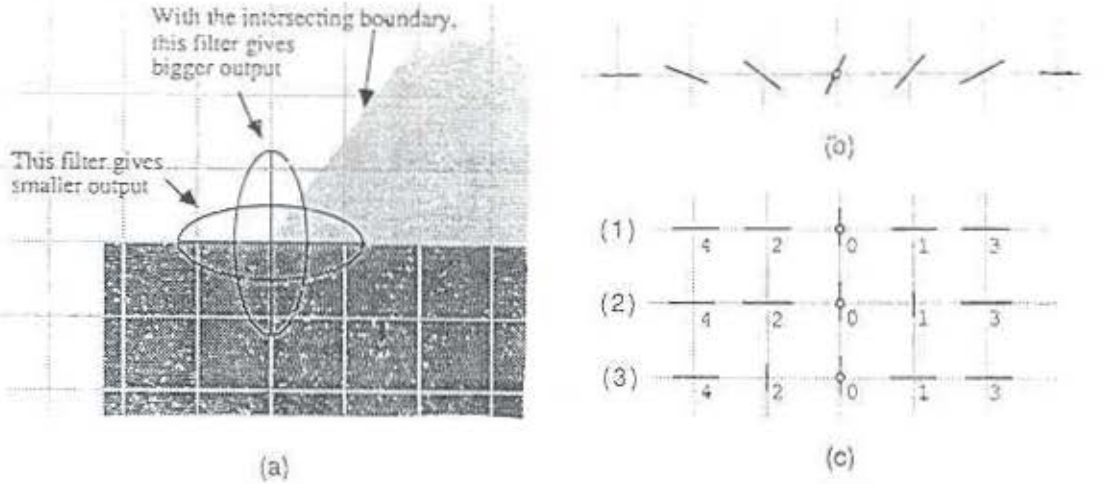


Figure 4.9: Edge strength patterns for a T-junction. (a) Edge filtering near a T-junction. (b) The effect of the intersecting boundary on edge strengths along the main boundary (horizontal for now). (c) The edge strength patterns used to detect a T-junction shown in (a).

the edge direction changes with a pattern shown in Figure 4.9b, wherein a small circle indicates the location of intersection. Furthermore, a smaller intersecting angle causes a wider spread of this direction variation.

This direction variation pattern can be used to detect T-junctions. There are two important points to consider. First, how big should the area be in which the algorithm checks for the direction variation pattern? Evidently, an area of radius of 1 pixel as for corner detection is insufficient. Since, for efficiency, the pixels under consideration should be as few as possible, the current implementation checks the edge strengths within an area of radius equal to 2 pixels.

Second, how can the direction variation pattern be detected? An intuitive answer is to match the pattern of Figure 4.9b directly. This approach is unlikely to work because, given two edge strengths of successive orientations at a sampling point, the calculation of the edge direction requires an interpolation among orientations, and interpolation involves a division. Hence it is difficult to accomplish using a connectionist architecture. Moreover, the storage of the edge direction requires many neurons.

An alternative is to match the pattern of all the edge strengths at the positions near the junction. But this scheme is difficult because if, at two successive positions, the same two edge strengths are significant, to decide if the edge directions at these positions are in proper sequence requires the comparison of the magnitudes of these edge strengths, which is also costly for a connectionist approach. Moreover, comparison of absolute values, here edge directions, is noise-sensitive.

A closer look at Figure 4.9a suggests that an approximation scheme can be devised based on how the intersecting boundary disrupts the edge strengths along the main boundary. As mentioned before, the edge strengths along the boundary are diminished near the junction and are sometimes exceeded by the perpendicular ones. Thus after artifact cancellation the edge strengths along the main boundary may disappear; instead the perpendicular ones occur near the junction.

Based on the above arguments, the three edge strength patterns illustrated in Figure 4.9c are used to detect the direction variation pattern of Figure 4.9b. In the figure numerals 0 – 4 are used to indicate the locations under consideration. At a sampling point (indicated by a small circle and labeled by "0") if one of the edge patterns in 4.9c is detected, the location is deemed to be a T-junction. Note that the current scheme considers only the T-junctions in which the intersecting boundary affects the edge strengths at no more than two sampling points along the main boundary. Also, the edge strengths of diagonal orientations should not be considered because a significant diagonal edge strength can occur not only due to a T-junction but also to a wavy boundary. However, one or two vertical edge strengths embedded in a series of horizontal ones may well indicate the existence of a more complicated image structure.

In summary, the *T-junction strength*, $T(0)$ at location 0 in Figure 4.9c, can be calculated by

$$T(0) = \begin{cases} f(E(1;0), E(2;0), & \text{if } E(3;0), E(4;0), E(0;0) > E\_thd \;\&\; \\ \quad E(3;0), E(4;0)) & \quad ((E(2;0) > E_{thd} \;\&\; E(1;0) > E\perp hd) \text{ or} \\ & \quad (E(2;2) > E_{thd} \;\&\; E(1;0) > E\perp hd) \text{ or} \\ & \quad (E(2;0) > E_{thd} \;\&\; E(1;2) > E\perp hd)) \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where the function $f(E_1, E_2, ..., E_n)$ can be, say, a maximum-finding operation of its arguments. The scheme can easily be generalized for T-junctions composed of boundaries

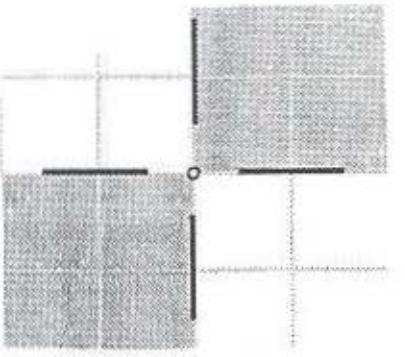


Figure 4.10: Edge strength patterns indicating a cross-junction.

of arbitrary orientations. Applying this approximation scheme to Figure 4.3 shows the scheme is effective.

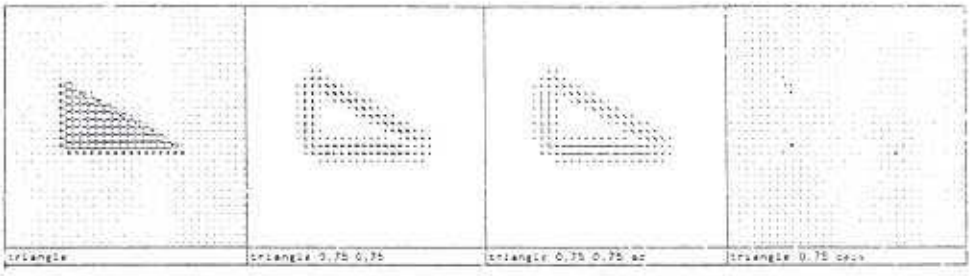
4.2.3 Cross Junctions

The edge strengths near a cross-junction are weakened by the complicated image structure. Further study on segmentation (described in Chapter 5) shows that only the cross-junction in Figure 4.5d seriously affects the segmentation. Because the pattern is perfectly symmetric for the edge filters at that junction, all edge strengths vanish there. For all other cases reasonable segmentation can be achieved with the edge and corner information.

The edge strength pattern shown in Figure 4.10 can be used to identify the cross-junction in Figure 4.5d. That is at a sampling point, the *cross-junction strength*, $Cr(x,y)$, can be represented by

$$Cr(x,y) = \begin{cases} f(E(x+1,y;0), E(x-1,y;0), & \text{if } E(x+1,y;0) > E\_thd \;\&\; \\ \quad E(x,y+1;2), E(x,y-1;2)) & \quad E(x-1,y;0) > E\_thd \;\&\; \\ & \quad E(x,y+1;2) > E\_thd \;\&\; \\ & \quad E(x,y-1;2) > E\_thd \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

The simple scheme successfully detects the pattern in Figure 4.5d.



(a) (b) (c) (d)

Figure 4.11: The corner detection of a test pattern, *triangle*. (a) the sampled input (b) the result of edge filtering (c) after artifact cancellation (d) the detected corners

4.3 Strengths and Limits of the Scheme

The corner detection algorithm described above was implemented by programs in C on a Sun workstation. Figure 4.11 shows the output after each processing stage of the algorithm: Figure 4.11a is the sampled input of a triangle, 4.11b is the result after edge detection, 4.11c is the result after artifact cancellation, and 4.11d demonstrates the successful detection of the three corners. What follows describes the strengths and limits of this corner-detecting algorithm.

4.3.1 Successful Detection of Corners

Figure 4.12 shows five corners of different opening angles, which are selected from a collection of simulated results on a set of ideal corners. All corners are located at (8, 8) of a 16 × 16 grid. For the corner of 15° in Figure 4.12a, the location detected is shifted by two pixels because of aliasing in the digitization process. In Figure 4.12b a corner is correctly detected at (8,8). For each corner in the image the algorithm does not give a single location; instead several positions near the exact location of the corner are indicated. This behavior may be improved by mutual inhibition, and in my algorithm this problem is addressed by the spatial coherence check, described in Chapter 5. Note that Figure 4.12IV shows that the same corner detection scheme works for both scales used in the figure. The segmentation results in these two scales will be shown in Chapter 5.

Figure 4.13 shows the corners detected for two triangles arbitrarily oriented. Both 4.13a and 4.13b show 2 diagrams — the left one is test input, and the right is the result of corner detection.

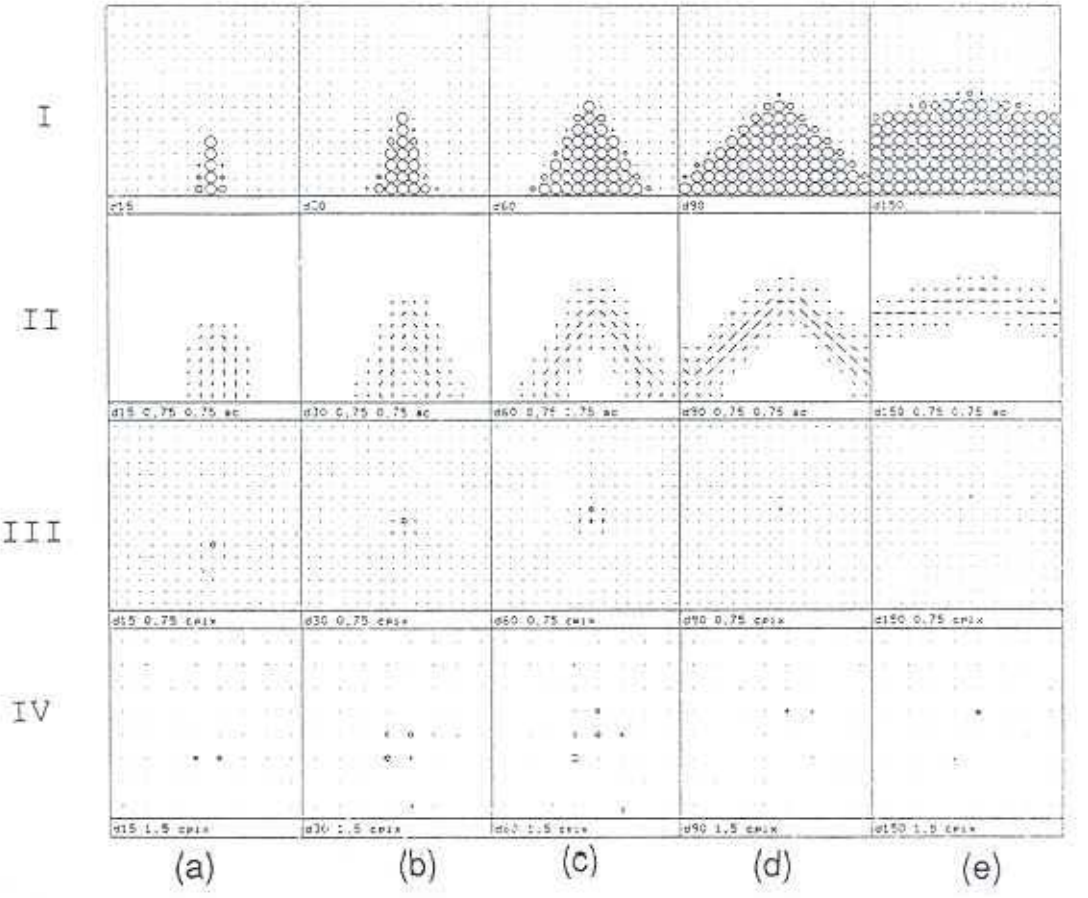


Figure 4.12: Corner detection for corners of various opening angles. Row (I): Input patterns; Row (II): results of artifact cancellation; Row (III): results of corner detection of the finest scale. Row (IV): results of corner detection of the next finest scale. Five columns indicate different angles: (a) 15° (b) 30° (c) 60° (d) 90° (e) 150°.

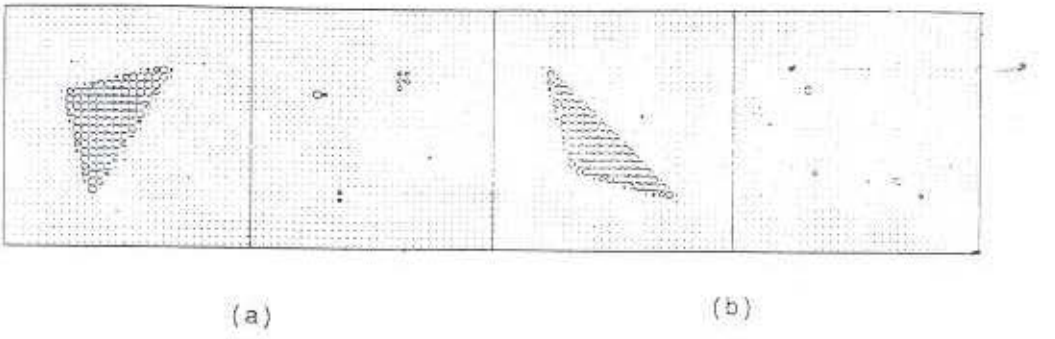


Figure 4.13: Corner detection for two triangles of arbitrary orientations. (a) a triangle of 45°, 45°, and 90° (b) a triangle of 30°, 30°, and 120°.

Figure 4.14 shows the corners detected on a portion of a real scene. Since a natural scene is usually complicated, causing the result to be difficult to see, this test pattern is intentionally selected for its relative simplicity. Compare the locations of detected corner with the input image; the result is satisfactory.

Figure 4.15 shows the detection of T-junctions shown in Figure 4.3. The results of edge filtering with artifact cancellation, the input to the T-junction detection scheme, are also shown in Figure 4.15II. Figures 4.15III and IV show that at both scales the existence of a T-junction was successfully detected in each input pattern. For the patterns with smaller intersecting angles the detected location for the T-junction shifts. This is in part due to aliasing in the digitization process as occurred for a sharp corner.

Figure 4.16 shows that a cross-junction as shown in Figure 4.5d is successfully detected. Note that the scheme described previously works only for a special kind of cross-junction. Further study is required to develop schemes for detecting general cross-junctions.

Figure 4.17 shows how the algorithm works on two thin narrow objects, each 2 pixels wide. The line ends are indicated by a cluster of corners. This is understandable since a corner also defines the end of an object.

4.3.2 Capability Limits of the Algorithm

The corner-detecting algorithm described above has capability limits in regard to sharp and nearly flat angles, noise, and blurring. These capability limits are discussed in the following, and methods for extending the performance beyond these limits are described in the next section.

The algorithm does not work as well near almost flat corners because the edge filters used are 45° apart and hence the edge strengths of orientations other than 0° are small near an almost flat corner. It is difficult to tell a corner from a straight edge. Again, we reach a limit due to aliasing in the digitization process. Figure 4.18 shows the corners detected in a triangle of 15°, 15°, and 150°.

Sharp corners are usually less accurately located. One reason besides aliasing is that edge filtering is more seriously affected by noise there. Since corner detection is based on

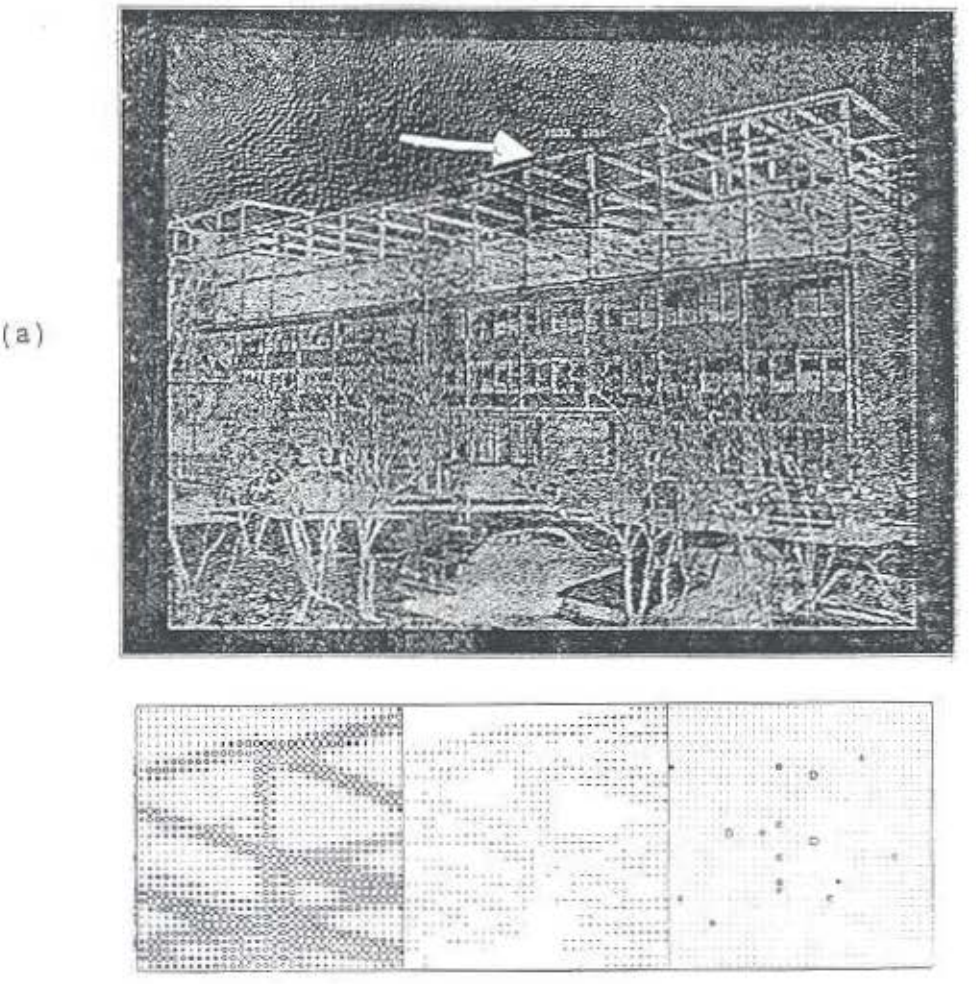


Figure 4.14: Corners detected in a real scene. (a) is an image of a building, portion in the frame is extracted as test patterns, (b) shows the test pattern with area of circle representing the intensity, (c) is the output after artifact cancellation, and (d) shows the important corners detected.

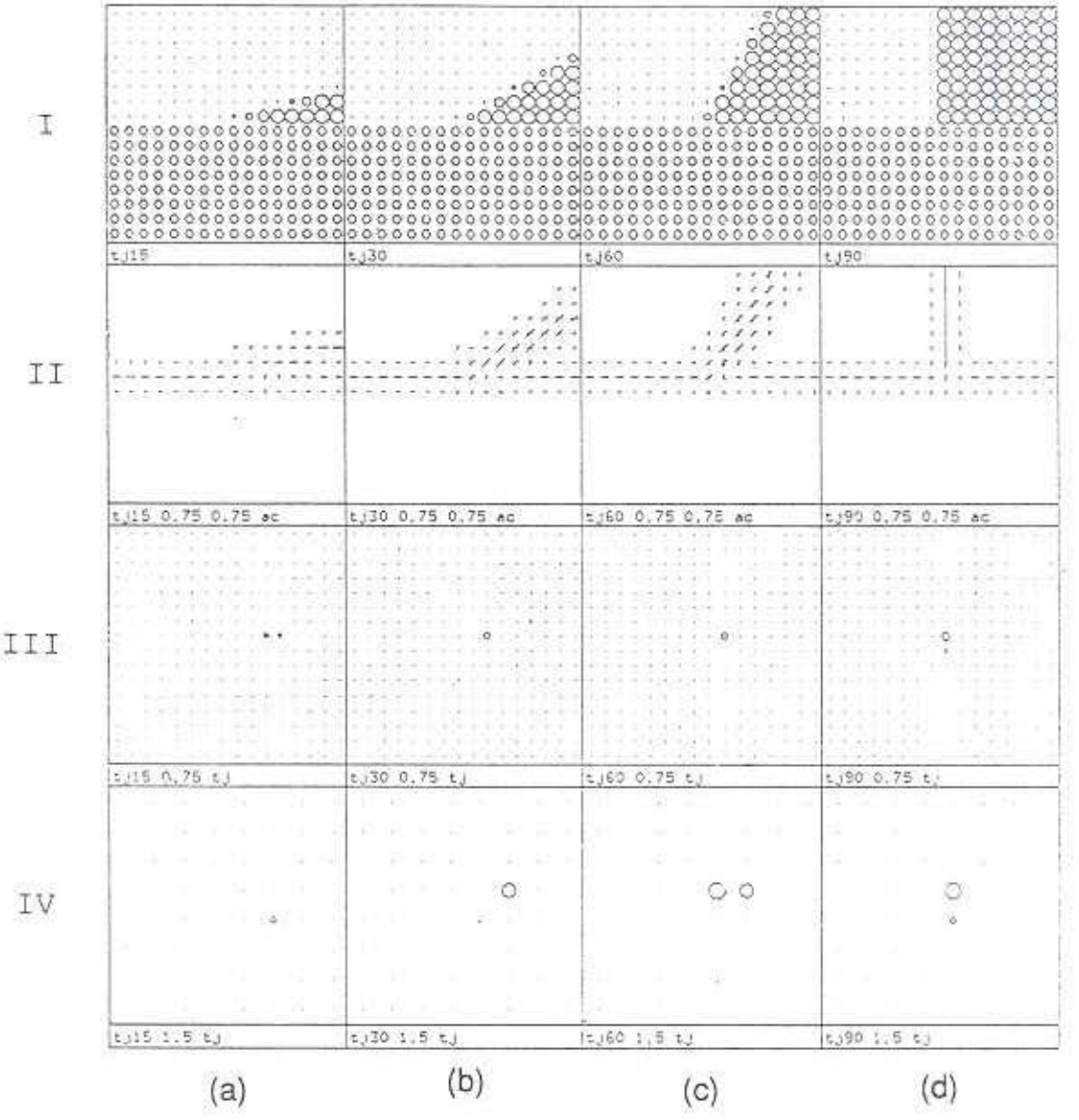
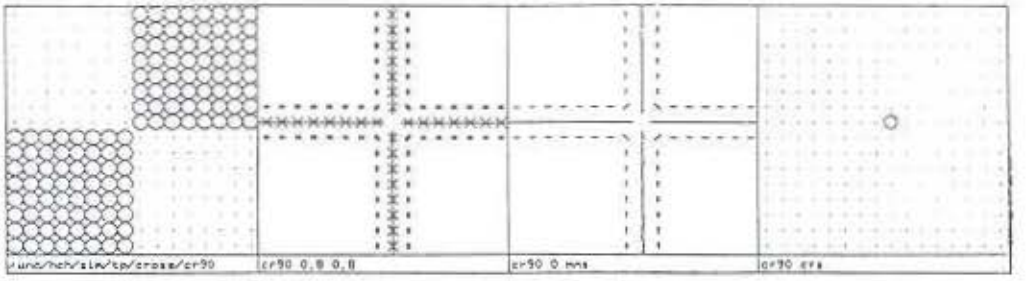
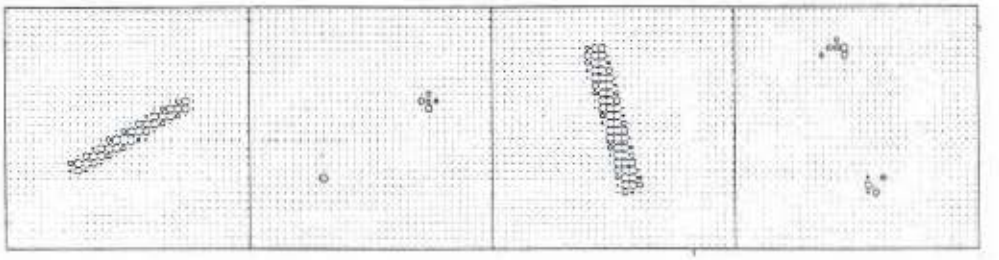


Figure 4.15: The detection of a T-junction. Row (I): Input patterns; Row (II): results of artifact cancellation; Row (III): results of T-junction detection of the finest scale. Row (IV): results of T-junction detection of the next finest scale. Four columns indicate different angles: (a) 15° (b) 30° (c) 60° (d) 90°.



(a) (b) (c) (d)

Figure 4.16: The detection of a cross-junction. (a) input pattern (b) edge filter output (c) after artifact cancellation (d) cross-junction detected



(a) (b)

Figure 4.17: Two examples of line end detection.

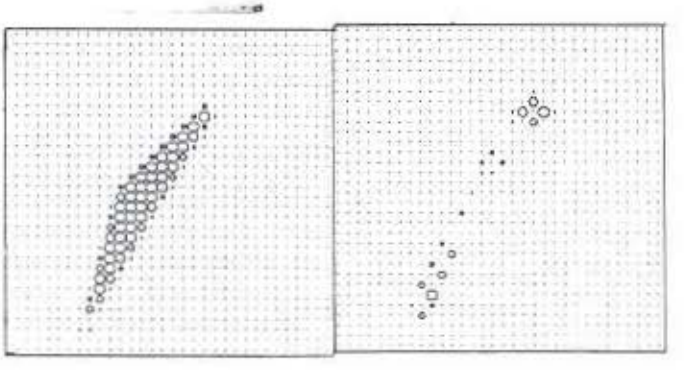


Figure 4.18: Corner detection for a triangle of 15°, 15°, and 150°.

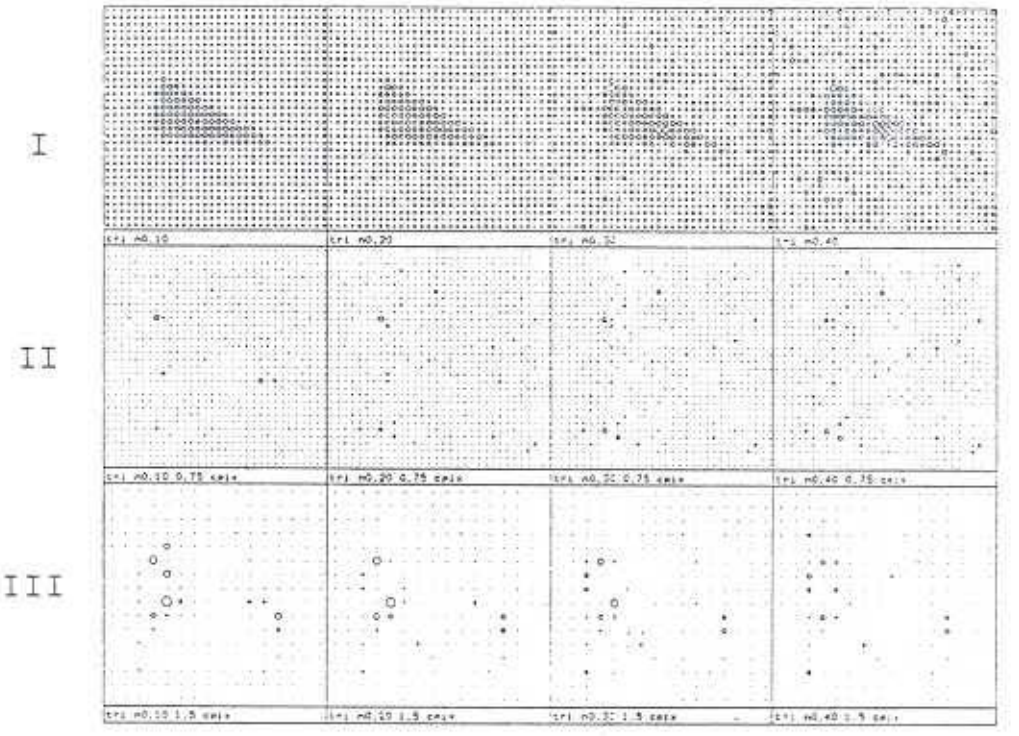


Figure 4.19: The effect of random noise on corner detection. The noise-to-signal ratio is (a) 0.1 (b) 0.2 (c) 0.3 (d) 0.4.

the edge filter outputs, it is accordingly affected. For similar reasons, the detection of T-junctions has problems when one boundary intersects another at a small angle.

Noise affects the performance of the algorithm as expected because a corner is a place where two edges meet and is hence more sensitive to noise than a single edge. In Figure 4.19I Gaussian noise (measured by its standard deviation, σ_{noise}) is inserted in each of the test patterns. The term "nx" in the subtitle of each diagram indicates that the noise-to-signal ratio of the test pattern is x, where

$$noise - to - signal\ ratio = \frac{\sigma_{noise}}{I_{sig} - I_{bkg}} \tag{4.6}$$

and I_{sig} and I_{bkg} are the average intensities of signal and background, respectively. The corners detected by the two finest scales are shown in Figure 4.19II and 4.19III. The coarser scale gives better performance. This point will be clearer when the segmented results of these noisy images are discussed in Chapter 5.

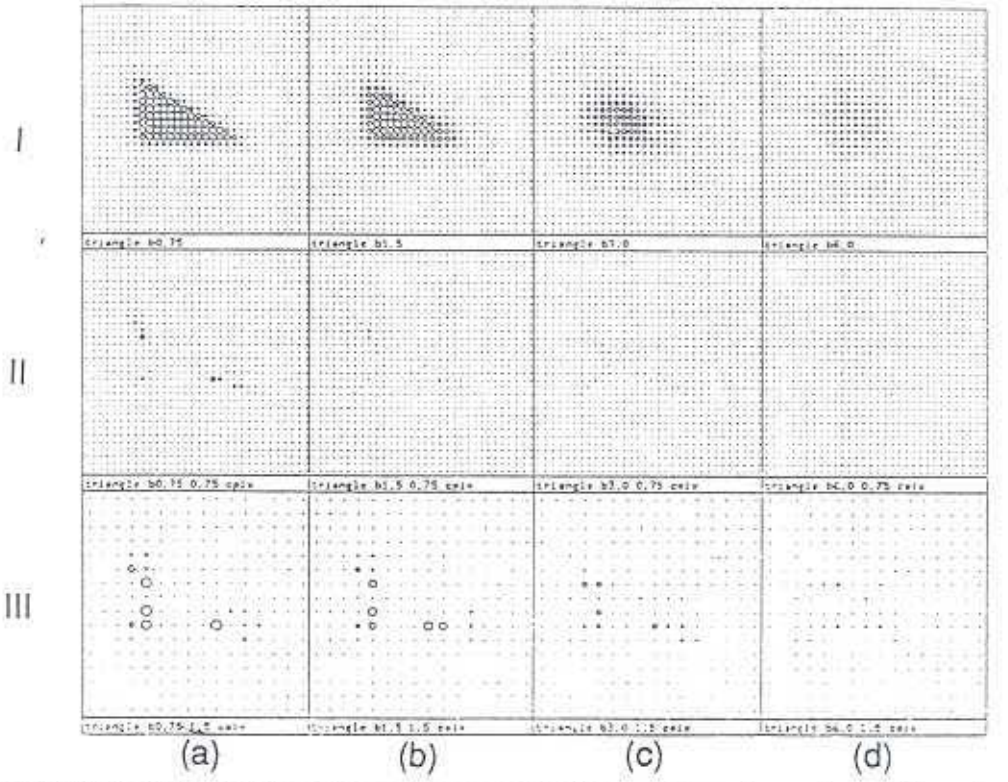


Figure 4.20: The effect of blurring on corner detection. A two dimensional Gaussian is used as the blurring function. The σ is (a) 0.75 (b) 1.5 (c) 3.0 (d) 6.0 pixels. Column (I): input; (II): corners detected in the finest scale; (III): corners detected in the next finest scale.

Figure 4.20 illustrates the effect of blurring. Two-dimensional Gaussians with σ = 0.75, 1.5, 3.0, and 6.0 were convolved with the test pattern *triangle*. When the blurring level increases, the edge strengths spread out; hence the number of detected corners increases as shown in Figure 4.20. When the blurring level is substantial, the corners are smoothed out as shown in Figure 4.20.

Finally, the algorithms for detecting the T-junctions and cross-junctions are not complete. What are the edge strength patterns for general T-junctions and cross-junctions? The answer awaits further investigation.

4.3.3 Remedies to Weaknesses and Several Implementation Issues

There are two possible remedies to the above-mentioned weaknesses at the cost of more resources. Figure 4.21a demonstrates that this algorithm can be extended by considering the neighboring 16 sampling points instead of only 4. All locations marked

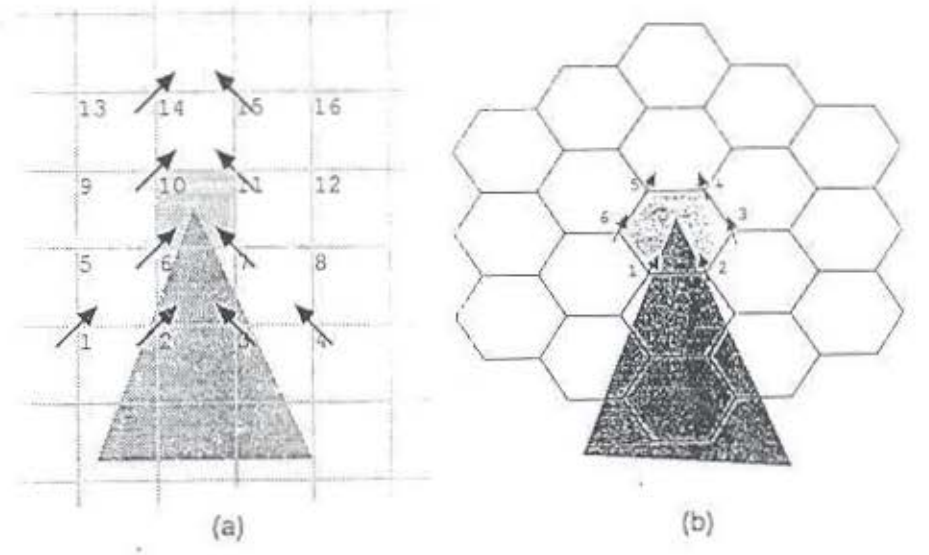


Figure 4.21: The possible schemes to improve the algorithm. (a) more sampling points are considered (b) a hexagonal sampling grid.

with an arrow in the diagram instead of only locations 6, 7, 10, and 11 can be used to estimate the probability of the existence of a corner in the shaded pixel. For example, if the edge filter pair at location 1 and 4 with marked orientation fire substantially and have opposite polarity, the probability should increase. Another scheme to remedy the weaknesses is to use the hexagonal sampling grid as shown in Figure 4.21b.

Though the algorithm has its shortcomings, so do all visual systems because they all have limited processing power and limited processing time. Yet the visual environment is arbitrarily complicated. Unpredictable situations beyond the visual system's detection capability may always occur. Therefore it is impractical to always search for perfect solutions; rather one should design systems the components of which are tolerant to inconsistencies in their inputs and make economic use of resources.

The resources required for the corner detection are briefly analyzed as follows. For the current implementation, assuming the resolution of the input image is $m \times n$ and the number of orientations is k, then we need $m \times n \times k$ edge filters, $6 \times m \times n \times k$ corner detectors, plus a summary layer and other intermediate connections on the order of $m \times n$. Note that each neuron has only local connections, say, of upper bound c. Then the number of connections in the network is in $O(cmnk)$. If the algorithm is simulated on a von Neumann machine, this figure also indicates the order of computation time. For the

current implementation, with an image of 64 x 64 and 50 neural connections on average, the computation load is about 1.5×10^8 operations. With a 10 MIPS machine, such as a Sun 4, it takes minutes to run. Evidently, with a connectionist implementation or a multiprocessing architecture, real-time performance could be expected.

4.4 Summary

This chapter described an edge-based corner detector. It has been shown that the intuitively simple scheme successfully detects corners under various conditions. The essence of the scheme lies in the locality of the detection algorithm. Thus a connectionist model can be built and an efficient implementation based on a multiprocessing architecture is conceivable.

The design of the model is based on knowledge of the human visual system, the constraints imposed by our visual world, and functional analysis of the computational needs of visual tasks.

Chapter 5

Segmentation

This chapter describes how to integrate the edge and corner information to give object contours. Without noise, the edge and corner detection methods described previously properly detect the object boundaries in an image. Nevertheless, noise is inherent in the imaging process. A boundary may be contaminated; a feature detector may indicate nonexisting features. How should one cope with this problem? One component of a solution is to consider contextual information — if an edge or a corner is a part of the object boundary, then there must exist nearby edges or corners. This operation, called *spatial coherence check*, is described in detail in Section 5.1.3.

The chapter first discusses the design of this segmentation algorithm and then summarizes the results obtained by applying this method to various test patterns.

5.1 Design of a Connectionist Segmentation Algorithm

This section first introduces the background for my segmentation scheme. Then it describes several design decisions. Lastly the section summarizes the algorithm and discusses why the algorithm is effective.

5.1.1 Background

It has been argued in Chapter 1 that a visual system cannot function at the pixel (receptor) level; a fast, data-driven information reduction is necessary. Image segmentation is the separation of an image into regions, where each region is approximately uniform in some property, e.g., color, motion, depth, texture, or intensity. Thus segmentation is a mechanism reducing the representation of an object from an intensity value at every pixel to a boundary and a description of the uniform surface property of the region within the boundary. This research operates on two-dimensional grey-scale images, and the grey-scale intensity is the only segmentation cue considered. The information reduction aspect

of image segmentation is true for both region-based and edge-based algorithms: for the former, the region grows until the property under examination is different — an object boundary is found; for the latter, the abrupt changes of the property in the image are first detected, and then the edges connect with each other to form a closed boundary — an object region is enclosed.

For an edge-based algorithm, how can a connectionist approach, which applies only simple functions on local information, cope with the difficult problem of edge following? An indication comes from Grossberg's *Cooperative-competitive loop*, which was briefly described in Chapter 2. Since the bipole layer is the key to the success of this process, the equations that Grossberg used to calculate the neural responses of *bipole cells* are summarized in the following.

Let $B(x,y;k)$ represent the response of a neuron with bipolar receptive field at location (x,y) with orientation k, and $B_l(x,y;k)$ and $B_r(x,y;k)$ the weighted sum from its left and right bipole field, respectively. Then at equilibrium state

$$B(x,y;k) = g(B_l(x,y;k)) + g(B_r(x,y;k)),$$

where g is a normalization function and $B_l(x,y;k)$ and $B_r(x,y;k)$ are calculated based on the edge strengths ($E(i,j;r)$ of orientation r at location (i,j)) and the connection weights ($F^{r,k}_{i,j,x,y}$: from neuron at $(i,j;r)$ to neuron at $(x,y;k)$):

$$\sum_{(i,j)\in R_l \text{ or } R_r,\, r\in D_k} [E(i,j;r) - E(i,j;r_\perp)]F^{r,k}_{i,j,x,y},$$

where R_l and R_r are collections of pixels in the left and right bipole subfield for B_l and B_r, respectively, D_k is the set of orientations near k, r is an orientation, and $r_\perp$ is the orientation perpendicular to r. Note that $E(i,j;r) - E(i,j;r_\perp)$, instead of $E(i,j;r)$ alone, is used in the above calculation. Grossberg and Mingolla [1986, 1987] found that boundary completion is more satisfactory with this operation. The counter-interaction between perpendicular edge strengths is called *artifact cancellation* in my algorithm and was discussed in Chapter 3.

The *cooperative-competitive loop* has two problems. First, to complete the gaps of a test pattern such as *Kanizsa square* (Figure 2.1b) of various sizes, different sizes of bipole fields are necessary, causing the network configuration to be image-dependent. Second, it is improper to apply the bipole field, which has a linear shape, near corners. What are possible remedies?

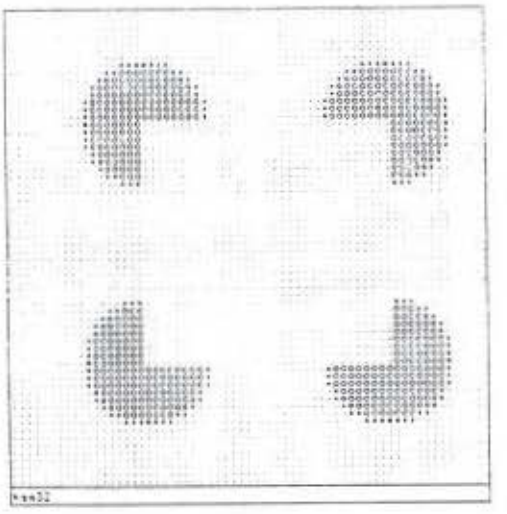
To cope with the problem of image dependence, the bipole field should not vary in size but should be based on a fixed support. As for the size of the support, a small support not only means less processing (i.e., fewer connections), but also gives a better-connected boundary. For example, if the coherence for an edge is checked based on 2 pixels on each side, then a boundary may be deemed connected when there is actually a one-pixel gap on the boundary. A problem of using a small support is that if noise destroys edge strengths along the boundary for more pixels than the size of the support, the boundary cannot be completed. In my algorithm a small support is used for the coherence check, and gaps on contours are completed by the edge strengths generated by elongated edge filters, as discussed in Chapter 3.

As regards the improper response of the bipole field near corners, a specific corner detector has been devised in my algorithm, and specific coherence rules for corners will be defined. A problem with this idea is the incompatible representation of edges and corners: corner detection shows whether a corner resides within a pixel in an image, while edge filtering is applied at the four corner locations of a pixel. How can the corner and edge information be combined? The answer comes from a closer look at the corner detection scheme described in Chapter 4: the corner detection mechanism not only detects the existence of a corner but also differentiates the corner type. Hence a corner can be represented by its forming edges.

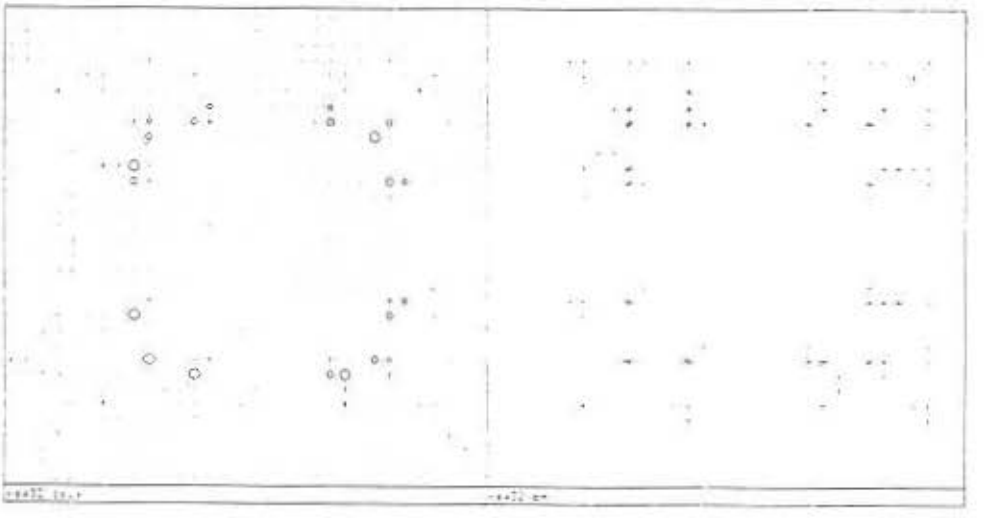
Figures 5.1b and 5.1c show the detected corners in two different representations for test pattern *Kanizsa square*: one indicates in which pixels corners reside and the other shows the edge strengths forming corners. Note that a corner in Figure 5.1c is indicated by edge strengths of 3 orientations at a sampling position. The purpose of using more than two edge strengths to indicate a corner is to make the spatial coherence check for edges easier, as will be seen more clearly in Section 5.1.4. Three edge strengths are also used to indicate a T-junction; for a cross-junction shown in Figure 4.5d, the edge strengths of all four orientations are used.

5.1.2 Design Decisions in my Segmentation Scheme

Based on the above idea, a segmentation scheme was devised. The design decisions concerning edge polarity, stopping criteria, and the handling of incoherent edges are discussed in the following. The spatial coherence rules for edges and corners are then discussed in the next subsection.



(a)



(b) (c)

Figure 5.1: The detected corners in test pattern *Kanizsa square* shown in two representations. (a) input pattern (b) corners indicated by circles at the pixels in the image (c) corners represented by the edge strengths forming the corners.

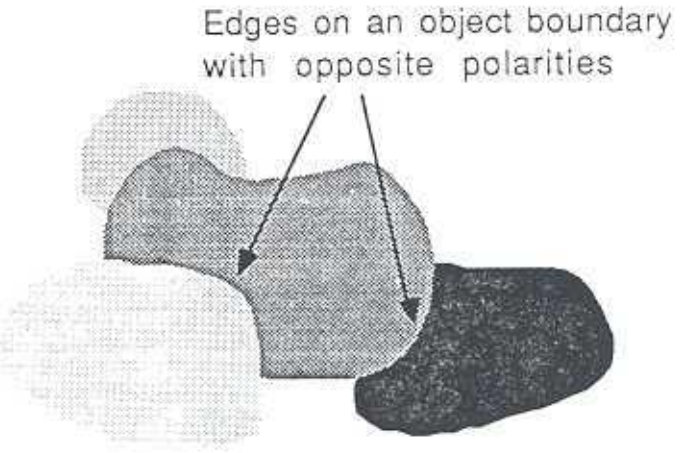


Figure 5.2: An object contour may contain edge strengths of opposite polarities.

Should the edge polarity be considered in the definition of spatial coherence rules? The answer is "no" because, as illustrated in Figure 5.2, an object may occlude objects of different surface intensities and hence its boundary may consist of edges of opposite polarities. Besides, Figure 2.1b shows that a subjective contour can also be formed by edges of opposite polarities. Therefore, the spatial coherence rules defined in the next subsection consider only orientation and location.

The spatial coherence check rejects incoherent image structures through an iterative process. An important design decision, then, is to select appropriate stopping criteria for the iterative loop. In the current implementation, the iteration continues until no edge strengths changes. In practice, the number of edge strengths modified during an iteration usually drops to under 0.1% of the total number of edge strengths after 3 iterations.

Viewing an edge strength as proportional to the probability of having an edge of a certain orientation at the location, how should an edge strength be adjusted? There are several possibilities, e.g., the magnitude of an incoherent edge strength is decreased by a certain ratio. In my current implementation the incoherent edge strengths are set to 0 for simplicity and efficiency. This scheme will be further discussed in Section 5.1.4.

5.1.3 Spatial Coherence Rules

The spatial coherence rules describe how an edge or a corner should be connected locally and is essential for this segmentation scheme. What should the spatial context be for an edge, a corner, a T-junction, and a cross-junction?

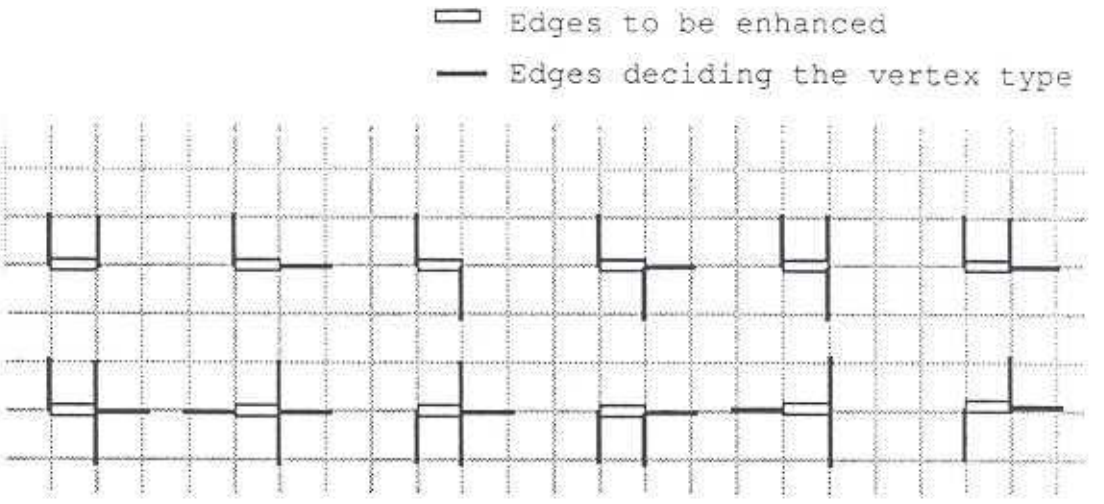


Figure 5.3: Several allowable local connections in Prager's algorithm.

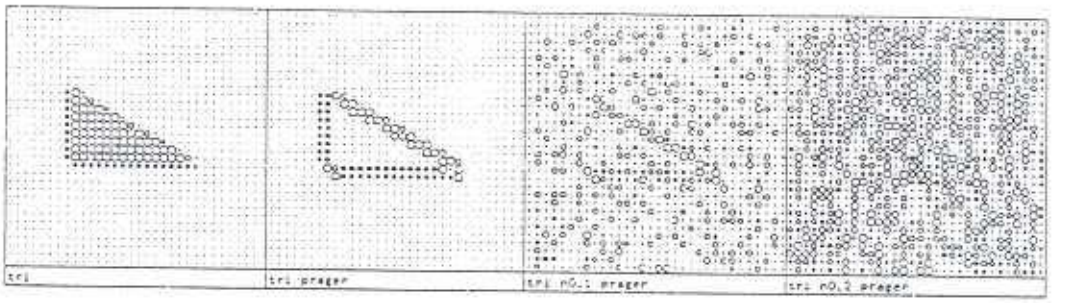
A clue comes from Prager's work on edge relaxation, wherein an edge strength is adjusted according to the connectedness of its two vertices [Prager, 1980]. An implementation of this algorithm in our laboratory shows that the algorithm behaves reasonably well on many images. However, several points on the design and performance of that algorithm are worth further consideration:

1. Only the vertical and horizontal edge gradients are used to decide the vertex type. Furthermore, the edge gradients are usually calculated on a small support and are hence sensitive to noise.

2. Corners are not explicitly detected but are accepted implicitly in the definition of the allowable local connections. Thus the definition of allowable local connections (a portion of them illustrated in Figure 5.3) needs to allow 90° turns. The above two factors cause a small amount of noise to result in bad segmentation. Figure 5.4 illustrates this point.

3. The algorithm is not guaranteed to give closed contours, causing difficulties for further processing.

My algorithm differs from Prager's mainly in two respects. First, not only are edges and corners specifically detected, but also the types of these features are determined. Thus coherence rules become easy to define — the necessary spatial context required to form a closed contour from an edge with known orientation and a corner with known forming



(a) (b) (c) (d)

Figure 5.4: The segmentation results of Prager's algorithm on noisy images.

edges is straightforward to define. Second, my algorithm rejects incoherent edge strengths and does not increase others, while in Prager's algorithm an edge strength can be generated or enhanced.

The following describes the spatial coherence rules for edges, corners, T-junctions, and cross-junctions, respectively.

Edges

Figure 5.5 lists the spatial coherence rules for various edges. As described in Chapter 3, it is impossible to have more than two edge strengths at a sampling location after *artifact cancellation*. In particular, the edge strengths, as shown by the line segments with a circle in them in Figure 5.5, have only eight possible orientations or combinations thereof, as labeled by $0, 1, 2, 3, (0,1), (1,2), (2,3), (3,0)$.

The coherence rules for edges are defined separately for the two categories of edges: for one category the edge is oriented in one of the four principal directions (numbered by 0 - 3 in Figure 5.5) and for the other category the edge is oriented between two successive principal directions (numbered by two numerals). For the former, unless there is an edge or a corner at the positions indicated by small line segments at both sides of an edge (e.g., positions A, B for edges with orientation 3), the edge strength under consideration is set to 0. For the latter, besides the 8 closest pixels, one more location is checked. For example, for the edge oriented in direction $(3,0)$ in Figure 5.5, the coherence rule requires

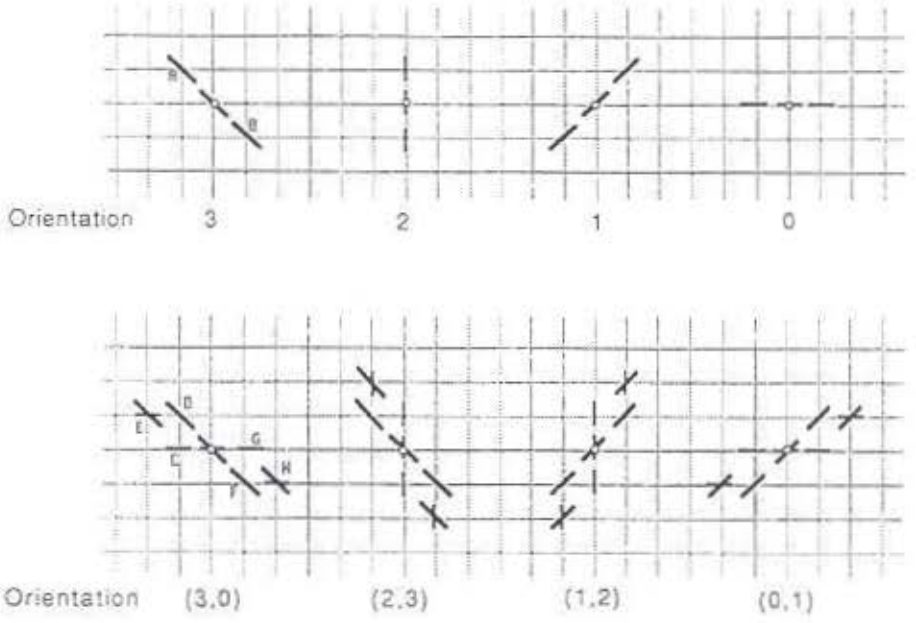


Figure 5.5: The spatial coherence rules for edges.

that there be corners or significant edges of indicated orientations at both sides — at least one of the locations C,D, and E has an edge or a corner, and so does one of locations F,G, and H.

In summary, the edge strength after spatial coherence check, $M(x,y;k)$ at location (x,y) of orientation k (ranging from 0 to 3), is calculated by

$$M(x,y;k) = \begin{cases} E(x,y;k) & \text{if } E(x,y;k) > E_{thd} \text{ and } R(x,y;t) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

where $R(loc;t)$ represents the result of the spatial coherence check at location loc for type t and t can be any of the eight indicated in Figure 5.5. As before, let $E(loc;k)$ be the edge strength at location loc for orientation k. Then the following shows by example how the values $R(x,y;t)$ and $R(x,y;(h,l))$ can be computed (see Figure 5.5).

$$R(x,y;3) = \begin{cases} 1 & if\ E(A;3) > E_{thd}\ \&\ E(B;3) > E_{thd} \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

and for edge type (3,0),

$$R(x,y;(3,0)) = \begin{cases} 1 & if\ \ (E(C;0) > E_{thd} \text{ or } E(D;3) > E_{thd} \text{ or } E(E;3) > E_{thd}) \\ & \text{and } (E(F;3) > E_{thd} \text{ or } E(G;0) > E_{thd} \text{ or } E(H;3) > E_{thd}) \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

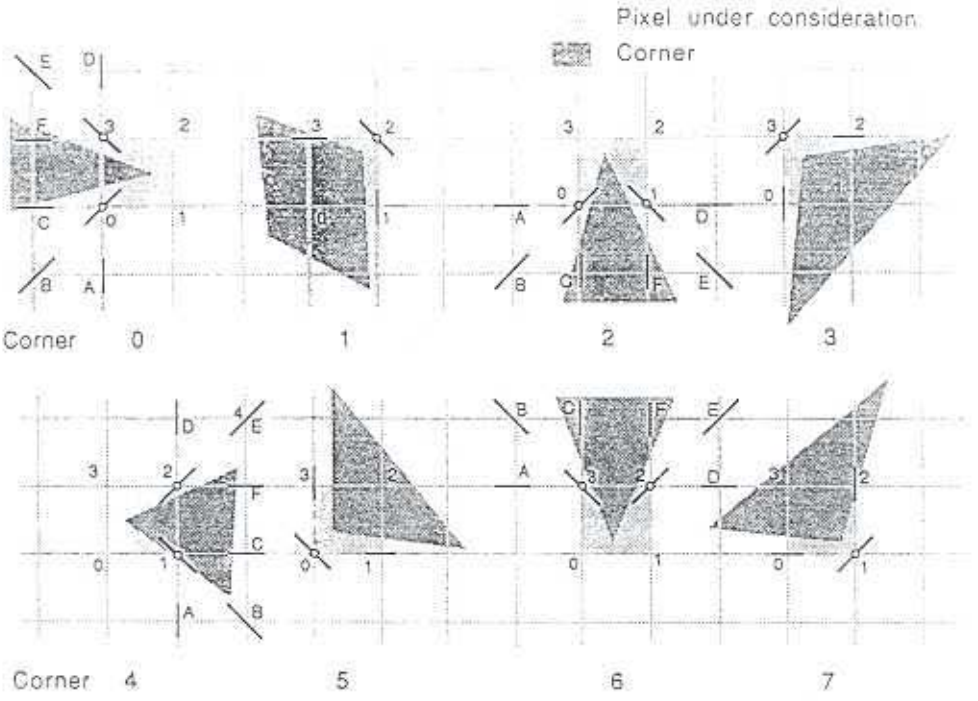


Figure 5.6: The spatial coherence rules for corners.

Note that since a corner is represented by its forming edges and edge strengths of three orientations are used to indicate a corner point, the above equations do not need to test for corners explicitly. For edges of other orientations similar rules apply.

Corners

Figure 5.6 illustrates the coherence rules for the eight corner types. In the figure the sampling locations around the pixel under consideration are labeled with numerals 0 – 3. The edge strengths forming the corner are indicated by small circles. The spatial locations considered by the spatial coherence rules are labeled by letters A – F.

The coherence rules for corners are similar to those for edges. At each of the two sides of the corner there must be an edge or a corner. For diagonal corners (odd numbered in Figure 5.6) the rules are simple: using corner type 1 as an example, $E(2;3)$ is verified if $E(1;2)$ and $E(3;0)$ are significant. For a horizontal or vertical corner, the situation is more complicated. Figure 5.7 illustrates that different edge strengths should be checked for corners of different angles. Hence in the current implementation a type 0 corner is verified if there is at least one significant edge with the indicated orientation at each side of the corner. Let $R_c(loc;t)$ represent the result of spatial coherence check for a type t

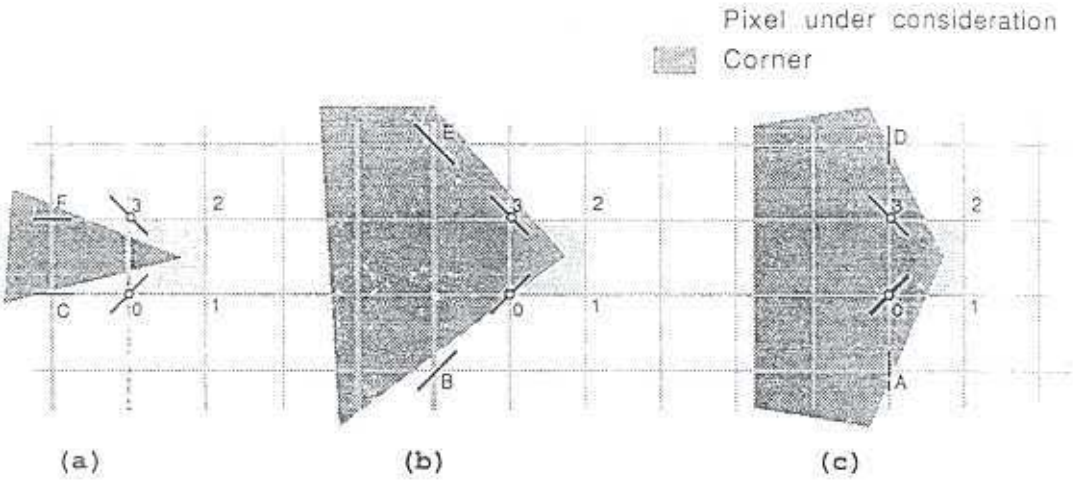


Figure 5.7: The spatial coherence rule for various corners of type 0.

corner at location *loc*. The expressions for two corner types are listed below; for other corner types similar expressions can be obtained by symmetry.

$$R_c(0;0) = \begin{cases} 1 & if \quad (E(A;2) > E_{thd} \text{ or } E(B;1) > E_{thd} \text{ or } E(C;0) > E_{thd}) \\ & \text{and } (E(D;2) > E_{thd} \text{ or } E(E;3) > E_{thd} \text{ or } E(F;0) > E_{thd}) \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

and

$$R_c(2;1) = \begin{cases} 1 & if\ E(1;2) > E_{thd}\ \&\ E(3;0) > E_{thd} \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

T-junctions and Cross-junctions

The spatial coherence rules for T-junctions and cross-junctions are simply the conditions for detecting these image structures as described in Section 4.2.2 - 4.2.3. To be precise, the result of spatial coherence check for the T-junction at location *0* in Figure 4.9, $R_T(0)$, can be calculated by

$$R_T(0) = \begin{cases} 1 & \text{if } E(1;0) > E_{thd}\ \&\ E(2;0) > E_{thd}\ \&\ E(3;0) > E_{thd}\ \& \\ & \quad E(4;0) > E_{thd}\ \&\ E(0;2) > E_{thd} \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

Note that the condition is simpler than that in equation 4.7 because the edge strength along the main boundary is activated for a detected T-junction.

The result for a cross-junction, $R_{cross}(x,y)$ at location (x,y), is

$$R_{cross}(x,y) = \begin{cases} 1 & \text{if } E(x+1,y;0) > E_{thd} \;\&\; E(x-1,y;0) > E_{thd} \;\&\; \\ & \quad E(x,y+1;2) > E_{thd} \;\&\; E(x,y-1;2) > E_{thd} \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

5.1.4 Summary of the Segmentation Algorithm

This subsection summarizes my segmentation algorithm and discusses the algorithm's edge-strength-adjusting scheme.

Figure 2.5, repeated in this section, summarizes this segmentation algorithm. The algorithm consists of parallel, multiple segmentation mechanisms, each segmenting the input image into boundaries at different levels of detail. There are two major differences among scales. First, the edge filters in different scales have different $\sigma_\perp$'s. The current implementation mainly uses three scales with $\sigma_\perp = 0.75, 1.5$, and 3 pixels, respectively. Values of $\sigma_\perp > 3$ pixels are not used because the highest resolution of the test patterns applied is only 128×128, and an edge filter of $\sigma_\perp > 3$ causes most test results to be too blurred for accurate feature detection. an uninteresting resolution. Second, different sampling intervals are used by different scales. As discussed in Chapter 3, for the scales with $\sigma_\perp = 0.75, 1.5$, and 3 pixels, the sampling intervals are 1, 2, and 4 pixels, respectively.

Within each scale, multiple edge filters first extract edges from the image. The edge integration stage then summarizes edge filter outputs. Corners, T-junctions, and cross-junctions are detected based on the outputs of the smallest edge filter. For each image structure a corresponding spatial coherence rule is applied. For locations where two or more image structures are detected, the more complicated image structure is selected. (The priority sequence is cross-junction, T-junction, corner, and edge.) The iteration of the coherence check assures that all edge strengths remaining are spatially coherent and form a closed contour.

The current implementation of my algorithm does not generate or increase edge strengths to complete a boundary. This approach is different from other relaxation boundary-finding schemes [Grossberg, 1985; Prager, 1980] in which edge strengths at a position can be adjusted both upward and downward. It is hard to define good heuristics to extrapolate an object boundary based on local information. Smoothness is the most common one used; the bipole field of Grossberg's *boundary contour system* is an example.

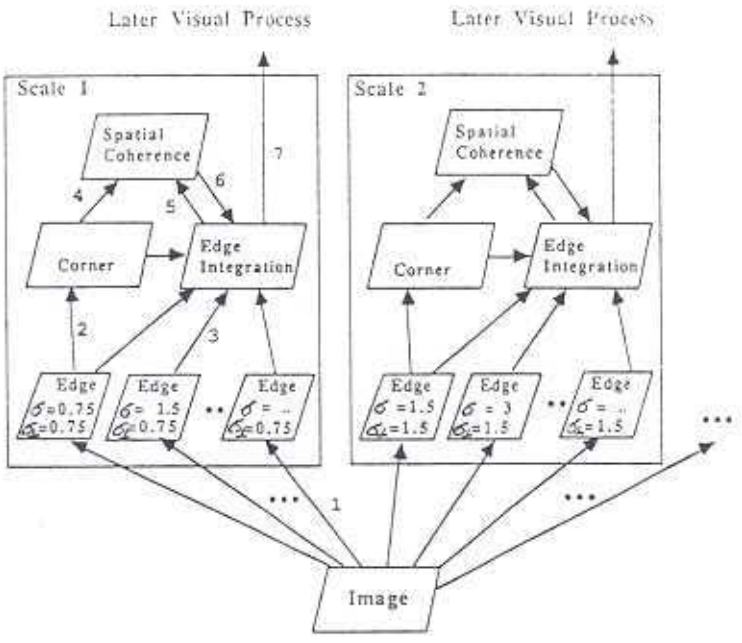


Figure 2.5: A block diagram describing the processing stages of my algorithm.

Unfortunately, the gaps that can be completed depend on the particular heuristics used, causing the system configuration to be image-dependent. More importantly, object contours in the image are not always smooth curves — there are corners, T-junctions, and cross-junctions, too. If a more tolerant coherence rule is used to cope with this problem (as with Prager's algorithm), the algorithm becomes sensitive to noise. My algorithm avoids using such heuristics; instead the propagated edges are generated by the elongated edge filters. Thus the extra edge strengths required to complete the gaps are obtained from the detection of nearby linear edge segments in the image instead of from a strength-adjustment scheme.

Propagated edge strengths can cause trouble by forming contours not indicating object boundaries in the image. This problem is lessened in part because corner detection is based only on the outputs of the smallest edge filter. Hence most propagated edge strengths, if not connected with correct edge strengths and forming a closed contour, will disappear after iteration. This arrangement prevents the occurrence of many undesired contours but enables the completion of gaps between two nearly linear edge segments.

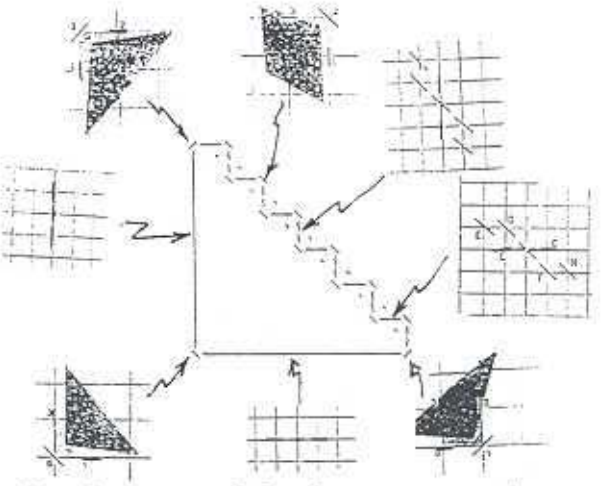


Figure 5.8: Closed contours defined on a rectangular sampling grid.

At the same time, gaps on boundaries can be closed, if not at one scale, then at another. Examples illustrating this gap-completion effect are shown in Figures 5.17 – 5.18. Though in my algorithm edge filters do not interact across scales, it is conceivable that the edges in different scales might be combined by certain mechanisms. The investigation of the principle and usage of this inter-scale interaction requires future research.

The coherence rules cause definition of a closed contour on a rectangular sampling grid — a closed contour is composed of edge strengths which are above a certain edge threshold and locally follow the spatial coherence rules. Therefore, starting from an arbitrary position and following the spatial coherence rules according to the local edge strengths, we can follow the boundary until we return to the original position. With this definition, if the later processing is performed on a sequential computer, a chain code can be used to represent the boundary, and the statistics about the region can be calculated. Figure 5.8 illustrates a closed contour; the coherence rules invoked by different edge strengths are also indicated.

5.2 Test Results and Evaluation

This section describes the results of my segmentation procedure on a collection of test images. In order to evaluate the results of my algorithm, criteria for judging the quality of a segmentation are needed. The following subsections first discuss my opinions on what is a *good* segmentation and then introduce the test patterns. The test results are lastly presented.

5.2.1 Evaluation Criteria for Segmentation Algorithm

What is a *good* segmentation? Unfortunately, the answer depends on how useful the segmented information is to the visual system. Poggio, Torre, and Koch [1985], among others, showed that image segmentation is an ill-posed problem because of the information loss in the imaging process. There is generally not a unique correct or measurable segmentation for an image.

What features, derived from the input data without interaction with a knowledge base, are important for recognition? Evidence from biological vision suggests that bounding contours, corners, surface properties, and the spatial relationship among the parts of an object are important. Segmentation is a data reduction process, but a good segmentation for object recognition should extract the above-mentioned information to feed higher visual processes.

Section 1.1.3 listed several properties of segmentation. The following further discusses these properties by listing the conditions for an acceptable segmentation:

1. Each segmented contour is closed so that the surface properties of a region can be computed. Each region can be an object or a portion of an object.
2. The corners are detected because corners are important for recognition.
3. The curvature magnitude and curvature pattern of each contour segment are specified with satisfactory accuracy.
4. The orientation of each contour segment is specified with satisfactory accuracy so that, for example, a diamond and a cube can be distinguished.
5. An object boundary is specified with satisfactory spatial accuracy so that the spatial relationship among objects and among parts of an object is preserved.

name	resolution	description	main purpose
bldg	128 × 128	Natural scene dithered	Corners
c..	16 × 16	Synthesized cross-junctions	Cross junctions
ct0	128 × 128	Synthesized CT images	Segmentation
d..	16 × 16	Synthesized corners	Corners
dsa	128 × 128	Digital subtraction angiogram	Segmentation
ksq	64 × 64	Synthesized Kanizsa's square	Illusory contour
ladder	32 × 32	Synthesized figure of ladder	Multiresolution
line	32 × 32	Synthesized line	Corner, segmentation
owl	64 × 64	Natural scene dithered	Segmentation
sqr	32 × 32	Synthesized square	Noise
tex	64 × 64	Synthesized figure of a texture	Multiresolution
tj..	16 × 16	Synthesized T-junctions	T-junctions
tri	32 × 32	Synthesized triangle	Corner
wheel	64 × 64	Synthesized circle with gaps	Multiresolution

Table 5.1: A list of test patterns.

5.2.2 Test Patterns

The input patterns used to test the algorithm can be categorized into three types: synthesized figures, medical images, and natural scenes. Table 5.1 lists the test patterns. In the figures illustrating segmentation results, shown later in this section, the *name* of each test pattern is shown in the subtitle. The dots in *c..*, *d..*, and *tj..* indicate numbers which stand for test patterns having different angles. The pattern *ct0* is an artificial image generated from real computerized tomography images.

5.2.3 Results

Segmentation of Objective Contours

Figure 5.9 shows the segmentation results for the more complicated image structures: corners, T-junctions, and cross-junctions whose input patterns were shown in Figure 4.2 – 4.5. Note that due to artifact cancellation an edge is indicated by at most two line segments; thus a location marked with more than two line segments indicates a more

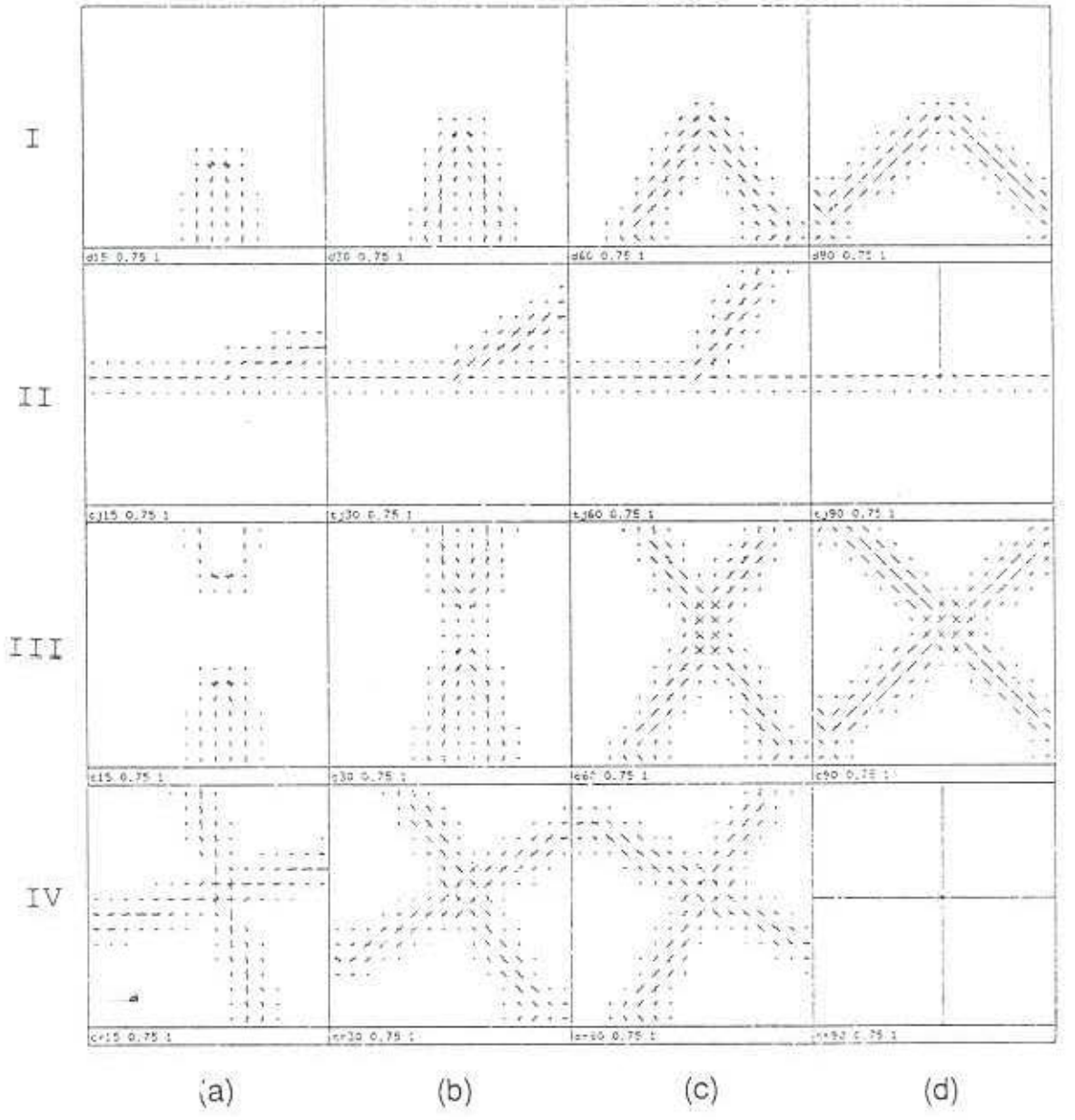


Figure 5.9: Segmentation results for more complicated image structures shown in Figure 4.2 – 4.5. Row (I) are corners. (II) are T-junctions, and (III) (IV) are cross-junctions. Columns (a) (b) (c) (d) show four different kinds of each image structure.

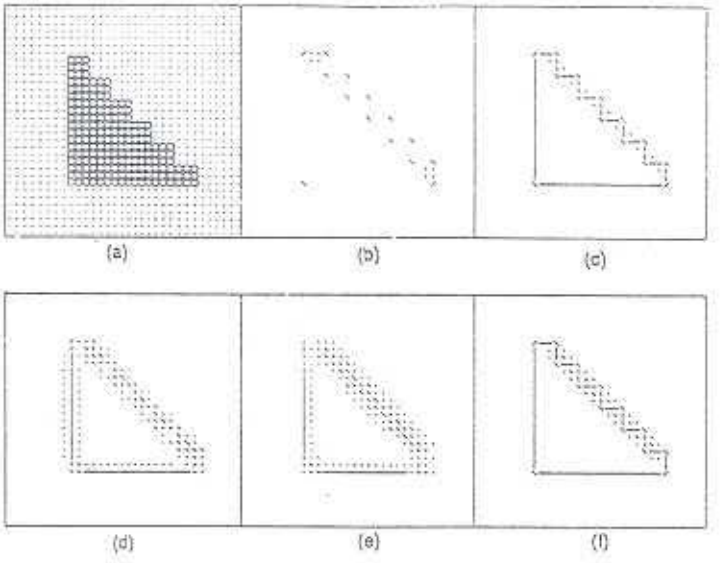


Figure 5.10: The effect of multiple edge filters within a single scale. (a) is test pattern *ladder*, (b) shows detected corners, (c) (d) (e) (f) are segmentation results for edge filters with (c) $\sigma = \sigma_\perp = 0.75$, (d) $\sigma = 1.5, \sigma_\perp = 0.75$, (e) $\sigma = 3.0, \sigma_\perp = 0.75$, (f) $\sigma = 0.75, 1.5, 3.0$; $\sigma_\perp = 0.75$.

complicated structure. Figure 5.9IId shows a T-junction, and 5.9IIId a cross-junction. Without the detection of these structures, the spatial coherence rule would reject the vertical edges in 5.9IId and all edge strengths in 5.9IIId.

Figure 5.10 shows the effect of the multiple edge filters within each scale. Since the test pattern *ladder* has no noise and no sampling error, the algorithm gives proper segmentation using each edge filter separately. Figure 5.10c, 5.10d, and 5.10e show this point. Figure 5.10b shows the corners detected. Note that a corner is represented as a combination of edges. Figure 5.10f is the segmentation result from the finest scale with the the outputs from the three edge filters averaged. Figure 5.10c gives the contour with the best spatial accuracy, but other segmentation results are also reasonable.

The algorithm performs well on more complicated images. The segmentation of test patterns *ct0* and *owl* are shown in Figure 5.11 – 5.12. Both images are processed by the finest scale with edge filters of $\sigma = 0.75, 1.5, 3.0$ and $\sigma_\perp = 0.75$.

(a)

(b)

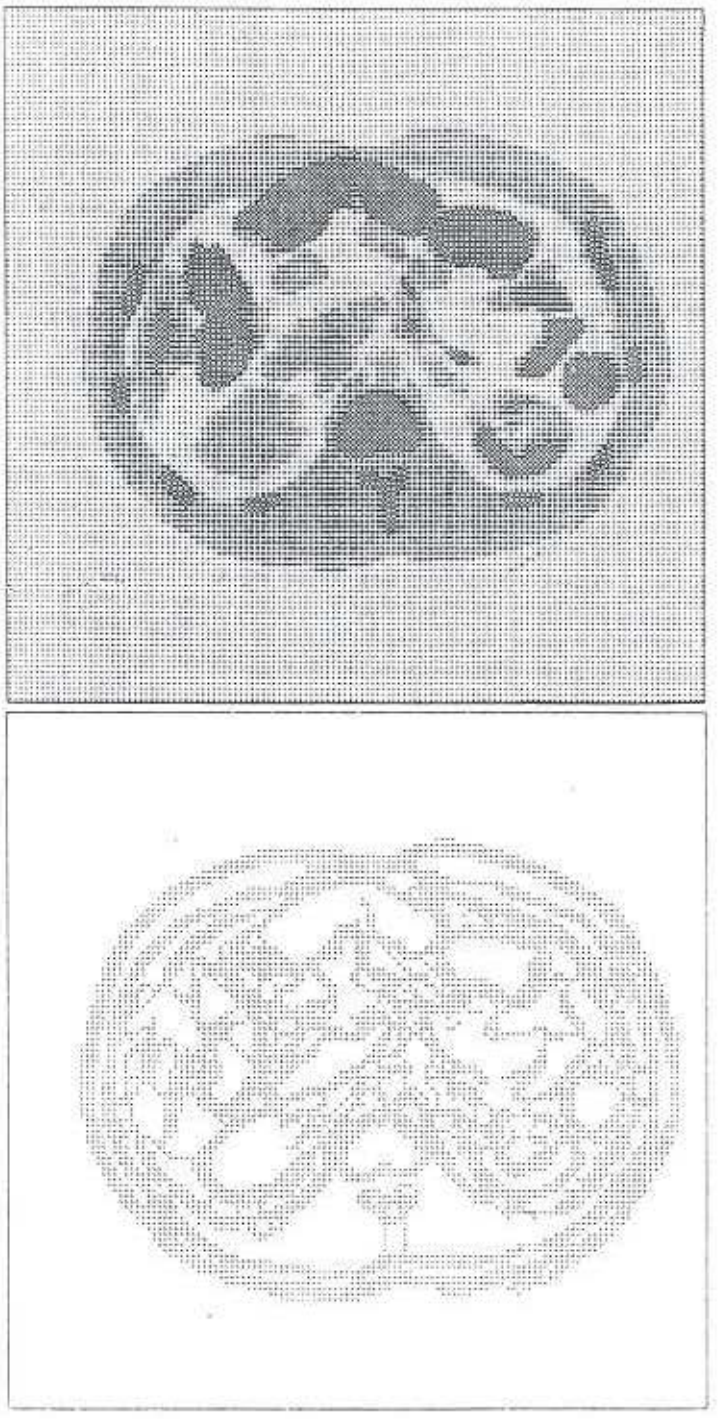
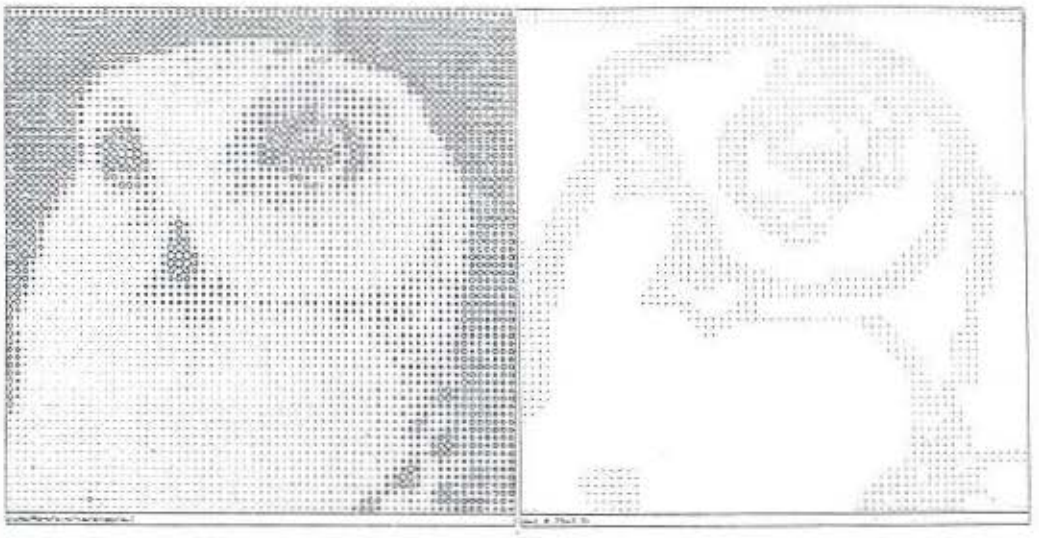


Figure 5.11: The segmentation of the test pattern *ct0*. (a) input (b) segmented result



(a) (b)

Figure 5.12: The segmentation of the test pattern *owl*. (a) input (b) segmented result

The Effect of Random Noise

The patterns *ksq* and *sqr* with various degrees of Gaussian random noise were input to the algorithm. The algorithm gives reasonable segmentation for low levels of noise. When the noise increases, either a portion of the boundary disappears or extra boundaries are generated. The behavior in the presence of noise will be discussed further in Chapter 6. Figure 5.13 gives two examples of successful segmentation of test pattern *ksq* and *sqr* with noise, respectively.

The Results of a Blurred Image

Figure 5.14 shows the segmentation results for images of various blurring levels. Compare these results with the results of corner detection in Figure 4.20. Though corners are smoothed out with heavier blurring as expected, reasonable segmentation is obtained for both scales.

The Segmentation Results at Different Scales

Different scales can locate object contours at different levels of detail. Figure 5.15 shows the segmentation results of test pattern *ladder* by the two finest scales separately. A single edge filter was used in each scale. This scheme can be expected to perform well because there is no noise in the input image. The results show that the two scales segment contours into different levels of detail. Similarly, Figure 5.16 shows the segmentation of

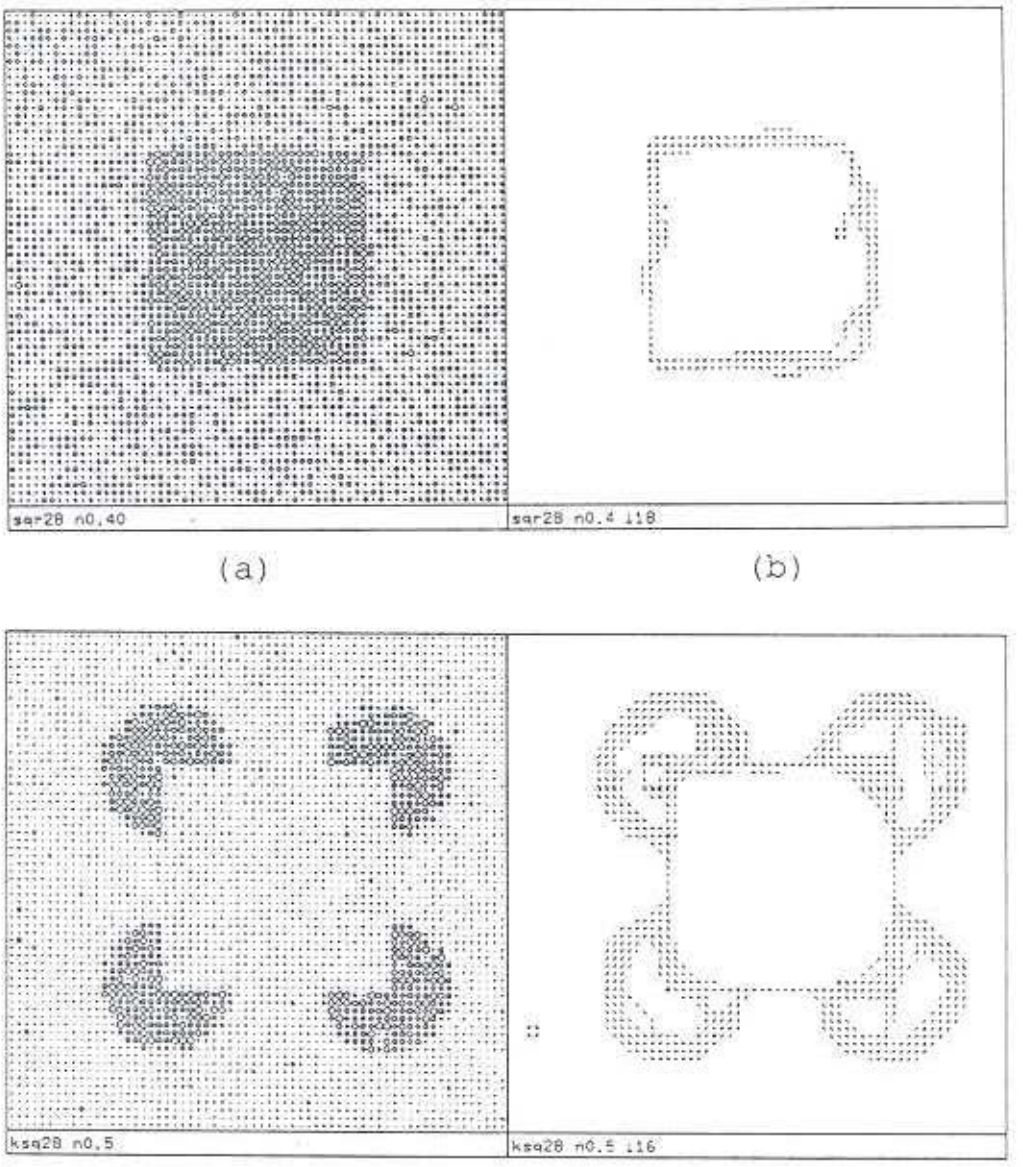


Figure 5.13: Examples of segmentation against noisy images. (a) Test pattern *sqr* with noise (b) The segmentation result against (a) for edge filters $\sigma = 0.75, 1.5, 3.0; \sigma_\perp = 0.75$ (c) Test pattern *ksq* with noise (d) The segmentation result against (c) for edge filters $\sigma = 0.75, 1.5, 3.0; \sigma_\perp = 0.75$

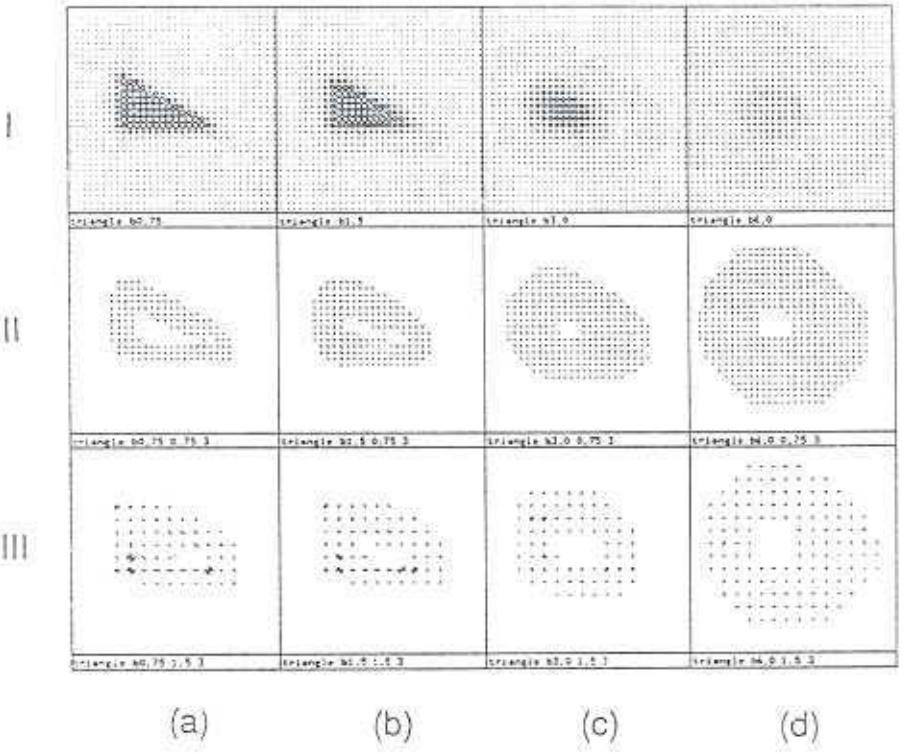


Figure 5.14: The segmentation results of blurred images by the two finest scales. Row (I) are input patterns, (II) are results of scale of $\sigma_\perp = 0.75$ pixels, and (III) are results of scale of $\sigma_\perp = 1.5$ pixels. Columns (a) (b) (c) (d) show four images blurred by an two-dimensional Gaussian of $\sigma = 0.75, 1.5, 3.0$, and 6.0, respectively.

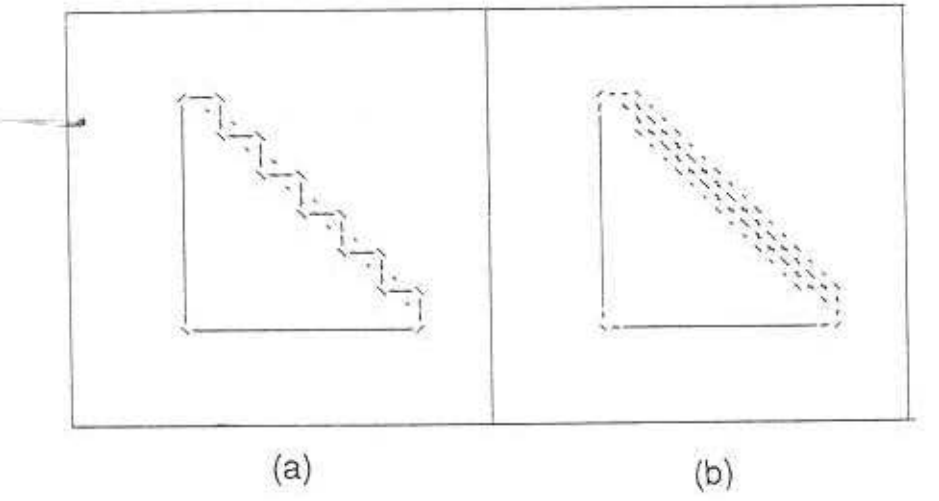


Figure 5.15: The segmentation of the test pattern *ladder* at two scales. (a) $\sigma = \sigma_\perp = 0.75$ in the finest scale (b) $\sigma = \sigma_\perp = 1.5$ in the second finest scale.

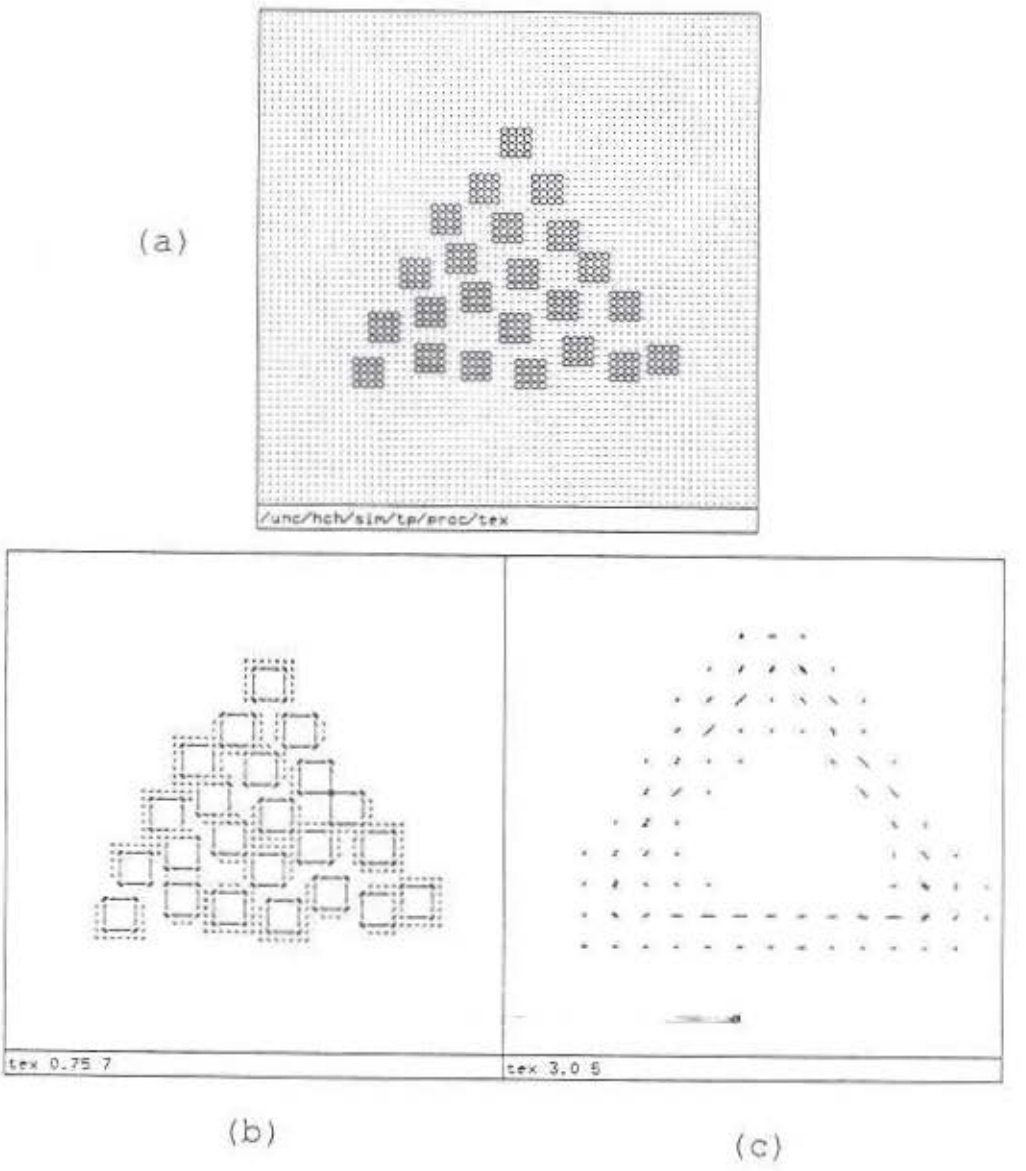
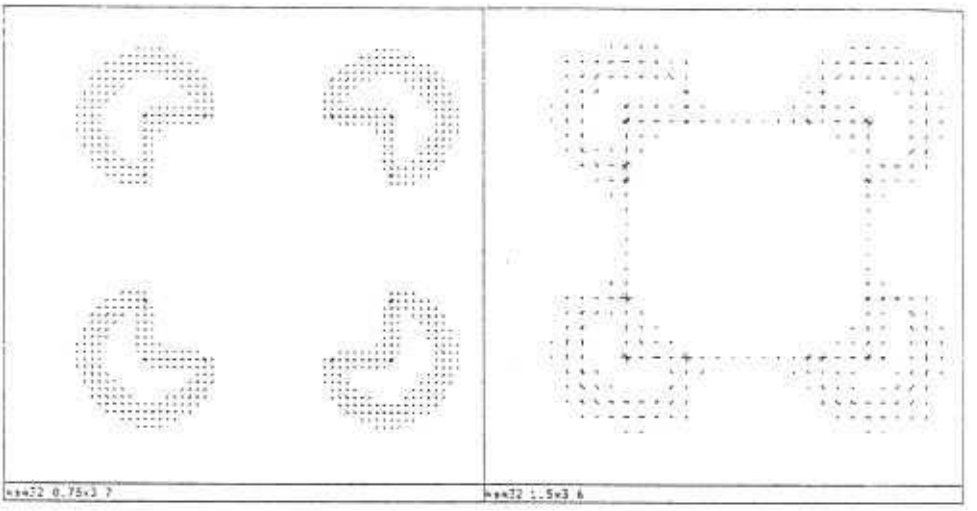


Figure 5.16: The segmentation of the test pattern *tex* at two different scales. (a) with edge filters (b) $\sigma = \sigma_\perp = 0.75$ (c) $\sigma = \sigma_\perp = 3.0$



(a) (b)

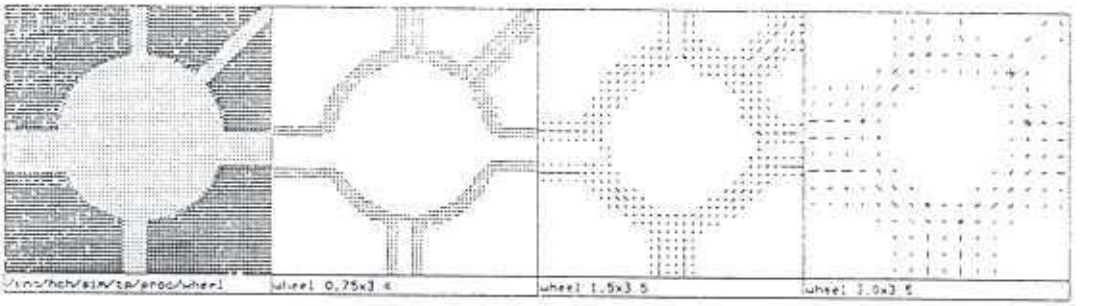
Figure 5.17: The completion of *Kanizsa square* by two scales. (a) scale of $\sigma_\perp = 0.75$ (b) scale of $\sigma_\perp = 1.5$.

the test pattern *tex* in two different scales. The coarser scale has resolution of only 1/16 of the finer scale and segments a more global triangle from the image. Figure 5.16a also shows that a cross-junction and several T-junctions were detected.

As mentioned in Chapter 3, with a fixed number of edge filters in each scale, a subjective contour not completed in a finer scale may be completed in a coarser one. Figure 5.17 shows that a Kanizsa square with the gap twice as long as the inducing edge (8 pixels) cannot be completed at the scale of $\sigma_\perp = 0.75$ but is completed at the scale of $\sigma_\perp = 1.5$ with each scale using 3 edge filters.

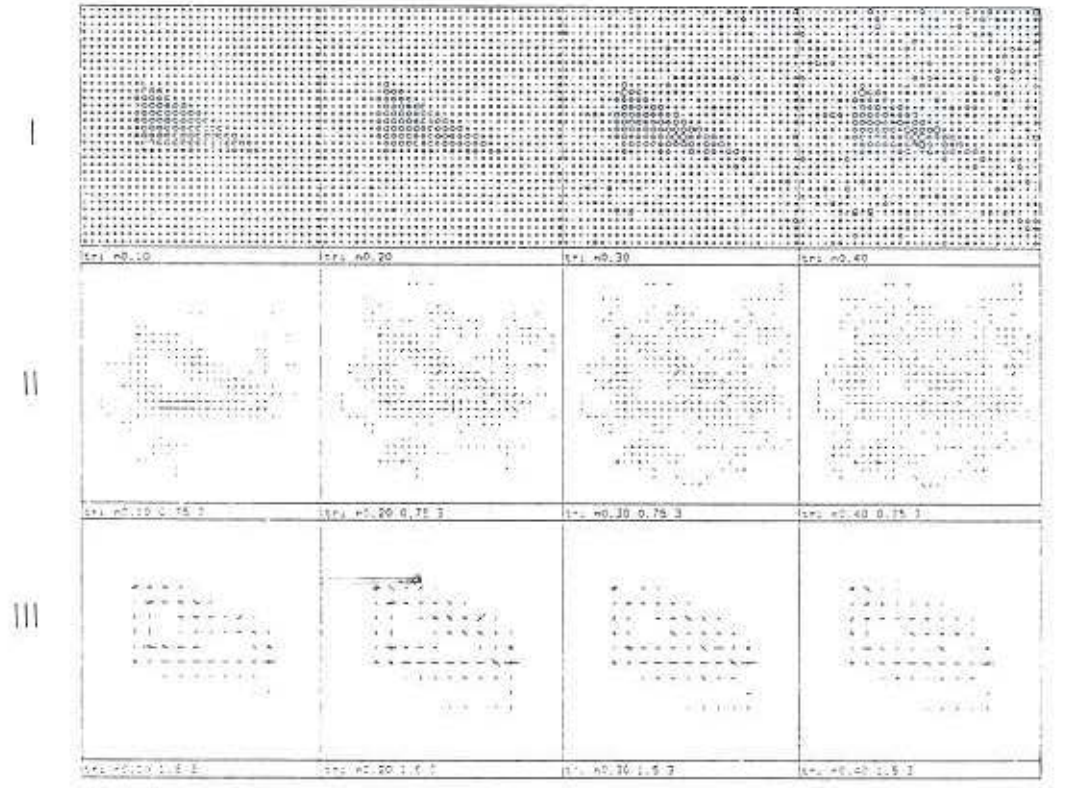
Another example is on the test pattern *wheel* of curve boundary as shown in Figure 5.18. The scale of $\sigma_\perp = 0.75$ completes the gaps of 2, 4, and 6 pixels wide, but not the gaps of 8 and 10 pixels. These bigger gaps are, however, completed in scales of $\sigma_\perp = 1.5$ and 3.0 pixels.

Figure 5.19 shows the segmentation results of images with different levels of noise. Evidently, the finer scale of $\sigma_\perp$ is heavily affected for noise-to-signal ratio above 0.3, while the coarser scale still gives a reasonable segmentation. This figure should be compared



(a) (b) (c) (d)

Figure 5.18: The completion of the gaps in a curve boundary on test pattern *wheel* in three scales. (a) input pattern (b) scale of $\sigma_\perp = 0.75$ (c) scale of $\sigma_\perp = 1.5$ (d) scale of $\sigma_\perp = 3.0$.



(a) (b) (c) (d)

Figure 5.19: The segmentation results of noisy images by two finest scales. Row (I) are input patterns, (II) are results of scale of $\sigma_\perp = 0.75$ pixels, and (III) are results of scale of $\sigma_\perp = 1.5$ pixels. Columns (a) (b) (c) (d) show four images with different noise-to-signal ratios: 0.1, 0.2, 0.3, and 0.4.

with the results of corner detection in Figure 4.19 and the results of Prager's algorithm in Figure 5.5.

5.3 Conclusion

In summary, my segmentation algorithm performs reasonably under a variety of conditions. The algorithm is robust against random noise and behaves as expected under blurring. Different scales in the algorithm can segment an object contour into different levels of detail; a coarser scale can complete bigger gaps and has better noise immunity.

The algorithm consists of only local wiring and is intuitively simple. Locality and simplicity contribute to the high processing speed. The segmentation of a 128×128 image with four edge filters in a single scale takes a Sun 4 a few minutes to compute. The algorithm demonstrates reasonable performance on vastly different input patterns.

In conclusion, my edge-based algorithm detects the intensity changes which are locally significant and globally form a closed contour. Compared with other algorithms for early visual tasks, this segmentation algorithm does not detect both *what* and *where*, but tries to answer a simpler question — given the observed intensity distribution alone, where is there something in the visual field?

Chapter 6

Evaluation

This chapter describes the evaluation of my algorithm. It first compares the algorithm's performance with Grossberg's *boundary contour system* and then with human performance on a specific task — object detection in the presence of random noise.

6.1 Comparison with Grossberg's Algorithm

This section describes the difference between my algorithm and Grossberg's *boundary contour system*. The comparison consists of two portions: the difference in the design and the difference in the resulting performance.

6.1.1 Differences in Algorithm Design

The differences in algorithm design between my algorithm and Grossberg's *boundary contour system* are briefly summarized in the following.

1. A different edge filter is used. Instead of a Difference of Box (which works fine with proper length and width selections), my algorithm applies the elongated first-order directional derivative of Gaussian. Also, only 4 edge orientations are sampled at each location instead of 8 or 12 in Grossberg's implementations.

2. The artifact cancellation operation is treated as a follow-up step of edge filtering. It is not performed in the iterative loop (cooperative-competitive loop) as in Grossberg's design.

3. Corners are separately detected in my algorithm.

4. To complete gaps of different sizes, the bipole field of Grossberg's configuration is image-dependent. In my algorithm the purpose of bipole cells is replaced by a spatial coherence check, which is defined on a small fixed support.

5. Bipole cells in Grossberg's cooperative layer can generate new edge strengths between two separate linear edge segments. In my algorithm incoherent edge strengths are rejected.

6. In Grossberg's model there are two on-center off-surround connections: one from the edge filter to the first competitive layer, and the other from the cooperative layer to the first competitive layer. A function of these connections is edge thinning. There is no corresponding processing in my algorithm. The parallel edge segments between two neighboring regions indicate that the intensities vary smoothly between the regions and are deemed as useful information.

7. The concept of multiscale is essential for my algorithm. With multiple elongated edge filters within each segmentation mechanism, each with fixed $\sigma_\perp$ and different σ, noise effects are minimized and gaps on boundaries are closed. With multiple segmentation mechanisms, each with different $\sigma_\perp$ and sampling interval, more global contours are labeled, and bigger gaps are completed. Grossberg mentioned multiple-sized edge filters and multiple-sized bipole fields as possible extension to the *boundary contour system* [Grossberg and Mingolla, 1987] but did not show results.

8. My algorithm does not depend on an analog neural formalism as does Grossberg's model. Moreover, my algorithm uses only one parameter, edge threshold, instead of eleven and applies only local operations; global operations like normalization are avoided.

6.1.2 Differences in Performance

The performance of Grossberg's boundary contour system was evaluated by implementing an approximation of his model [Grossberg and Mingolla, 1986] and applying the program to the test patterns listed in Table 5.1. The implementation deviates from the original model in two main respects: a different edge filter was applied and an approximation scheme was used to calculate the neural responses. Also, the anisotropic first-order derivative of Gaussian was used as the edge filter instead of the difference-of-box function.

It is assumed that the time derivative terms in the nonlinear differential equations that model the behavior of the neurons rapidly converge, and therefore, the response of each neuron at the stable state can be calculated by algebraic manipulations instead of a solution of a system of differential equations. For example, the neuron at (i, j)

of orientation k, y_{ijk}, in the second competitive stage obeys the shunting on-center off-surround equation [Grossberg and Mingolla, 1985]

$$\frac{d}{dt}y_{ijk} = -Dy_{ijk} + (E - y_{ijk})O_{ijk} - y_{ijk}\sum_{m \neq k} O_{ijm} \tag{6.1}$$

where O_{ijk} is the neural response at (i,j) of orientation k in the previous layer and A, B are constants. Setting $\frac{d}{dt}y_{ijk} = 0$ gives

$$y_{ijk} = \frac{EO_{ijk}}{D + O_{ij}} \tag{6.2}$$

where $O_{ij} = \sum_m O_{ijm}$ for all orientations m at (i,j). The above equation shows that the neural response thus calculated does not depend on the transient behavior of the circuit but rather on the connection patterns. Mingolla contends that using the transient is essential to the performance of the Grossberg/Mingolla model, but the problems of basing detection on the transient seems insurmountable: On the one hand, the instability of the transient response would lead to dependence of time period in which a neuron gives proper response. On the other hand, using the transient presents the problem of synchronization among many mutually-interacting neurons.

For each test pattern the parameters of the *boundary contour system* were tuned to obtain a reasonable result (Recall the discussion of evaluation criteria in Section 5.2.1). All the test patterns listed in Table 5.1 served as inputs to Grossberg's algorithm. The results are shown in Figure 6.1 – 6.2. The effect of various levels of random noise is measured for two simple patterns, a square and an illusory square. These results are shown in Figure 6.3.

In brief, with the parameters properly selected, Grossberg's *boundary contour system* gives a reasonable segmentation for all the test patterns listed in Table 5.1. With multiple edge filters the system also gives different levels of detail for an object contour, which is shown in the results of test patterns *ladder* and *tex*. Compared with my algorithm described in Chapter 5, there are several differences in performance:

1. The configuration of my algorithm has only one parameter, edge threshold, and is not image-dependent, whereas Grossberg's model [1985] has 11 parameters among which the ones defining the bipole field are image-dependent.

2. My algorithm more explicitly defines a closed contour because, after iteration, all edge strengths and corners remaining are spatially coherent and form a portion of a closed contour.

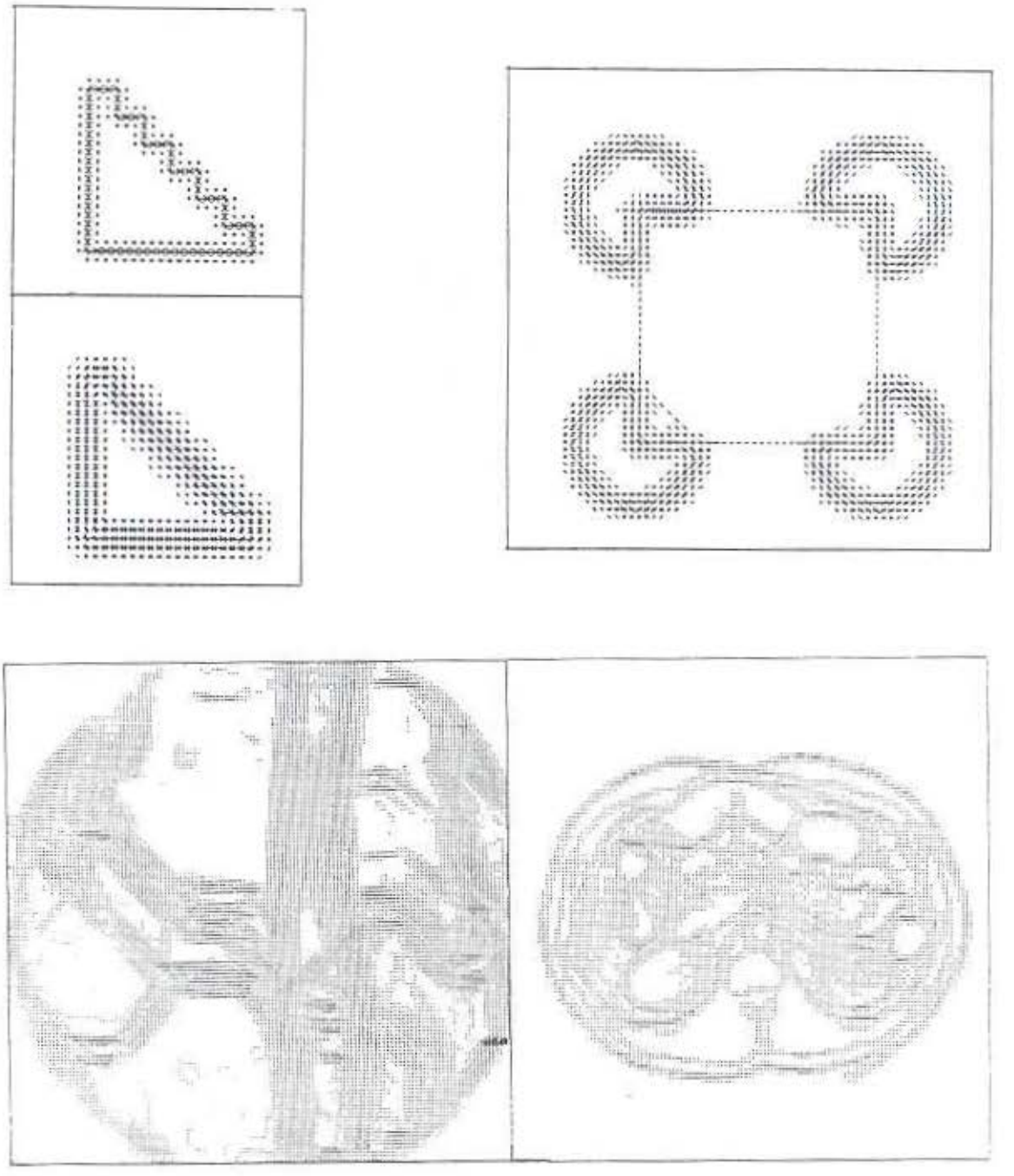


Figure 6.1: The segmentation results for various test patterns by Grossberg's *boundary contour system*.

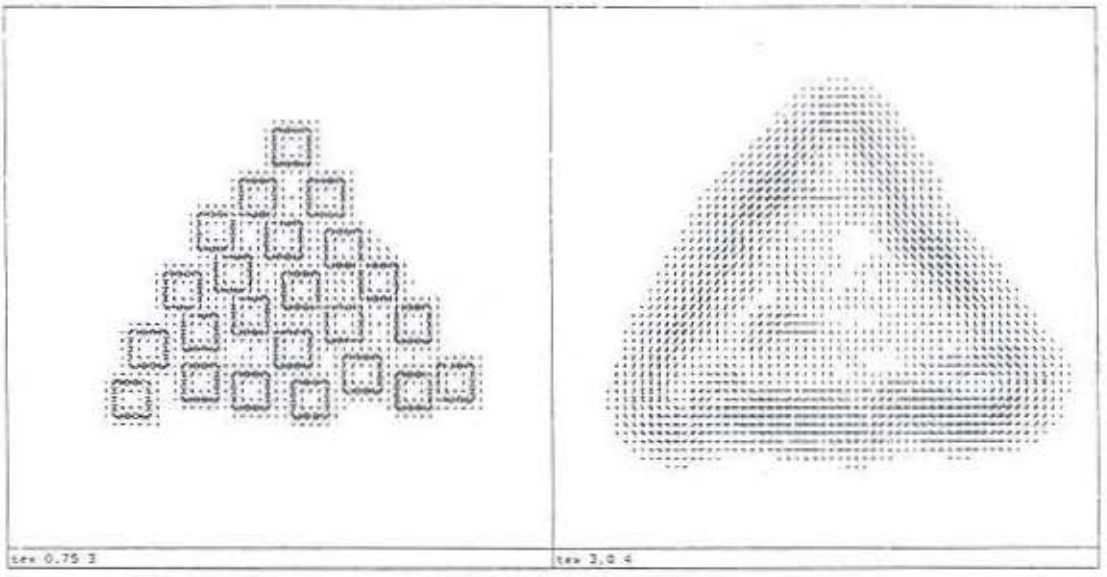


Figure 6.2: The segmentation results for test pattern *tex* by Grossberg's *boundary contour system*. The pattern to the left is with a small edge filter ($\sigma = \sigma_{\perp} = 0.75$); the right segments a bigger structure with edge filter of $\sigma = \sigma_{\perp} = 3.0$.

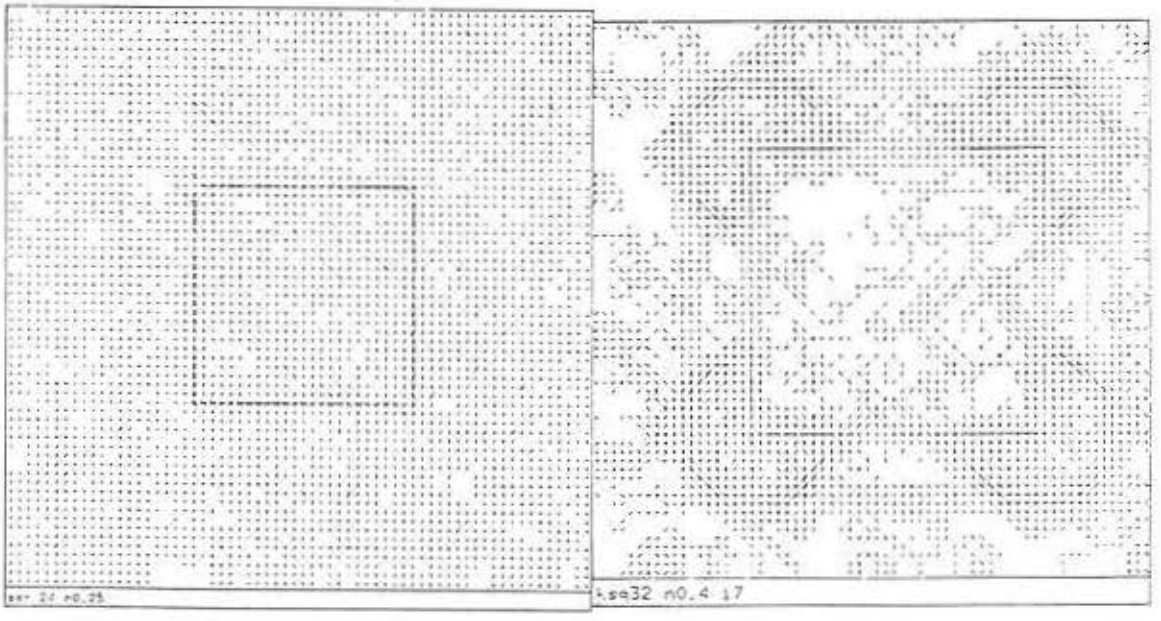


Figure 6.3: The effect of random noise on the performance of the *boundary contour system*. The subtitles of the patterns show that left pattern is with noise-to-signal ratio equal to 0.25; the right pattern has noise-to-signal ratio of 0.4.

3. Grossberg's model handles line ends and may explain the Ehrenstein illusion; my algorithm does not.

4. My algorithm has better noise immunity. Compared with Figures 5.12 and 6.8, which show the segmentation results of noisy images by my algorithm, Figure 6.3 shows that since the *boundary contour system* adds more edge strengths to the edge map, the result becomes very noisy when the noise level is above a certain value. (In the figure, this value is where noise-to-signal ratio = 0.4.)

6.2 Human Object Detection vs. Noise

My algorithm was designed in part based on knowledge of the biological visual system, so it is worth comparing the performance of my algorithm with that of human vision. A comparison between the characteristics of the two systems may help to improve the artificial visual system defined by my algorithm. Unfortunately, biological vision is so complicated that it is impossible to perform a thorough analysis in a short period of time. As a compromise, a very specific task concerning the effect of random noise on object detection was selected as the basis of comparison. The following section describes the experiment.

6.2.1 Objective

The objective of the experiment was to measure the threshold noise level at which a human subject fails to detect an object in the stimulus. The independent variable in this experiment is the random noise level; the dependent variable is the subject's object detection capability at that noise level. It follows that the task of the experiment is for the subject to state whether the subject detects an object in the stimulus or not.

To assure that results from the psychophysical experiment are comparable with that of my algorithm, the stimuli shown to the subject need to be the same as the images input to my algorithm. The environment must be carefully controlled; the task must be specific so that the effect of factors other than the independent variable are minimized.

Detection targets with both subjective and objective contours were used for the experiment. It is well-known that humans see objects consisting of subjective contours, and the perception of visual illusions may suggest aspects of the underlying architecture

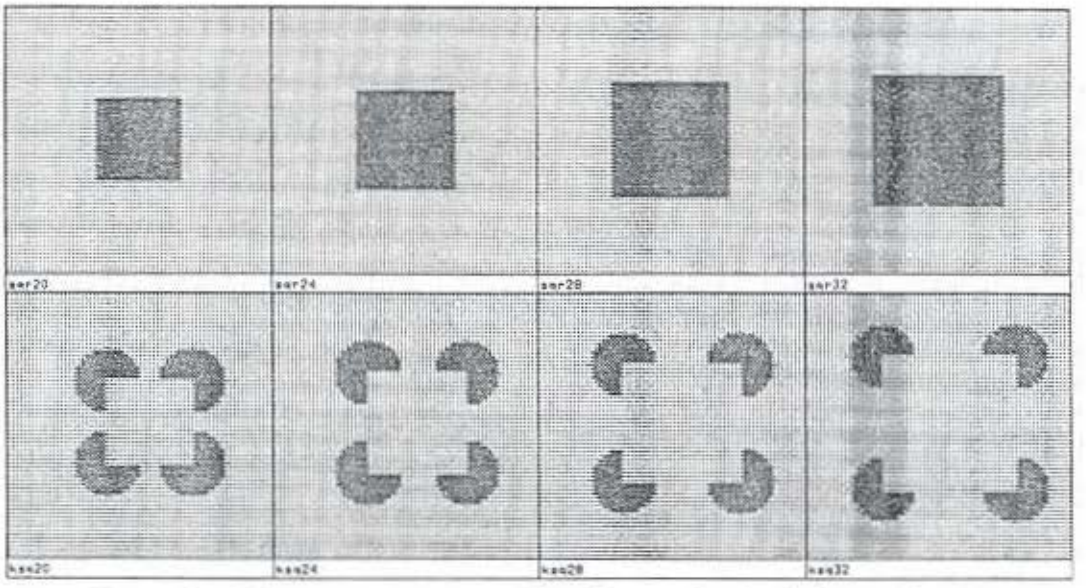


Figure 6.4: The test patterns used for the psychophysical experiment

of the visual system. Hence a comparison between the effect of random noise on human detection capability of a Kanizsa square and on my algorithm's segmentation capability with that test pattern seemed useful.

The comparison between the performance of a human and since my algorithm can be based on the absolute values of the threshold noise level or on a certain trend, e.g., how the threshold noise level varies with the sizes of the detection target. Since the biological visual system applies knowledge and multiple segmentation cues, and my algorithm is by no means a complete model of the human visual system, it is not very informative to compare the absolute values. A comparison of some trends is more reasonable. Therefore, four sizes were selected for each of the two test patterns, *sqr* and *ksq*, and each subject ran through test patterns of multiple sizes of each detection target. The hypothesis is that the way random noise affects the algorithm's performance on objects of different sizes is similar to the way it affects human object detection. To be more specific, the profiles of threshold noise level versus object sizes should be similar for my algorithm and human subjects.

6.2.2 Method

Experimental Paradigm

Several experimental procedures can be used for point estimation, e.g., *method of constants*, *method of limits*, *up-down procedures*, and *maximum-likelihood estimation*. For

this experiment, an *up-down method* was used because of its simplicity, high efficiency, small-sample reliability, and relative freedom from restrictive assumptions [Levitt, 1970].

The noise level in this experiment is specified as the noise-to-signal ratio, as defined in equation 4.6, and the measurement of the threshold noise level proceeds as follows. Each test pattern was shown to the subject with various levels of random noise was shown to the subject, and to each the subject answered whether an object was perceived or not. A positive answer increased the noise-to-signal ratio in the test pattern for the next trial, and a negative one decreased the noise-to-signal ratio. After a certain number of changes of response type, i.e., from *yes* to *no* or *no* to *yes*, the program stopped and the threshold noise level was estimated based on the resulting data.

The step size for the noise level adjustment was fixed through the experiment and was decided based on the results of a preliminary study: A bigger step size allows faster convergence and hence a shorter run, while a smaller step size gives better accuracy of the final result. To compromise, the noise in the initial image was set at a value near the possible threshold value, and reasonably small step sizes were selected. This step size is 0.05 (in noise-to-signal ratio defined in equation 4.6) for test pattern *ksq* and 0.1 for test pattern *sqr*.

An adjustment of the noise level was performed after each trial. However, it was found that if the same step size was used for both increment and decrement of the noise level, a subject easily noticed that convergence had been reached and might lose concentration for the later stimuli. A standard technique to cope with this problem is to integrate several tests into one run and interleave among different test patterns. Unfortunately, the integration also lengthens a run of the experiment. Each test of this experiment took 30 to 40 trials. A combination of, say, 4 tests might bore the subjects, resulting in lower data accuracy. Therefore, I decided to use a simpler scheme: the decrement of the noise level for each negative response (the subject does not see the object) was three times that of the increment. A result of this scheme is shown in Figure 6.5. The experiment can be viewed as a combination of alternating up and down sequence; each sequence is the result of an application of the *method of limits*. Note that the up and down sequences have different step sizes.

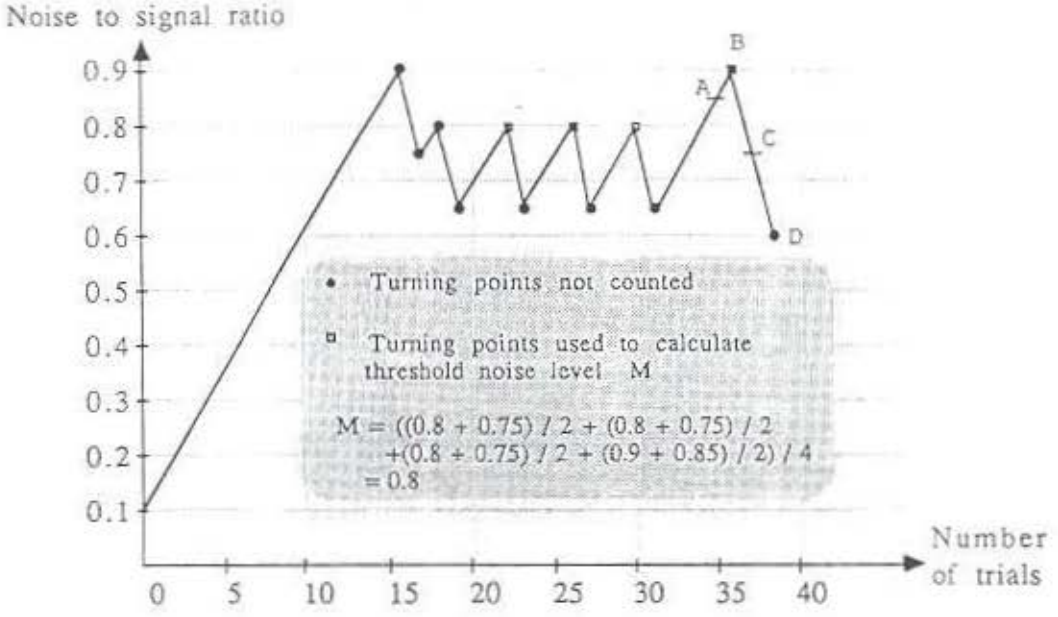


Figure 6.5: A set of data obtained from the experiment with the estimation method illustrated. The expression uses the Wetherill [1963] estimate of the 50% threshold based on the last 4 upper turning points. For each of the turning point the estimate of threshold is midway between the turning point level and the preceding level.

Given a set of data, the procedure for estimating the threshold value was discussed thoroughly by Wetherill et al. [1963]. In this experiment the turning points indicate the 50% point of a subject's psychometric function. There are two sequences and hence two kinds of turning points as shown in Figure 6.5. Let U be the threshold suggested by the up-sequence, in particular the average of the values at point A and B on Figure 6.5. Let V be the threshold suggested by the down sequence, in particular the average of the values at points C and D. Then there are several ways to estimate the threshold noise level based on U's and V's. For this experiment only the trend of the data points was considered; the absolute value of thresholds was not essential. Therefore, this simple estimation scheme was adopted. After studying the possible combinations of U's and V's for many sets of data, I decided to use the average of U's at the last 4 upper turning points as the threshold noise level. The first 2 upper turning points are discarded as suggested by Brownlee et al [1953]. This scheme is equivalent to successive applications of the *method of limits* with up sequence only.

The decision criterion of a subject could change over time during the experiment. This potential inconsistency was checked by the insertion of catch trials. Before the ex-

periment two patterns, one with small noise and the other with large noise were generated and shown to the subject. The experiment began after the subject agreed that there was an object in the pattern with small noise and none in the other. During the experiment, the catch trials of these two test patterns were randomly inserted in the trial sequence. If there was more than one error for the catch trials in a run of the experiment, the data were deemed inconsistent and were discarded. Five out of 34 sets of data failed the test.

Equipment

The images were displayed on a Comtal 10/24 imaging system. The host computer was a Vax 11/730 running the Berkeley 4.3BSD UNIX[1] operating system. The grey-scale monitor was calibrated and converged prior to the experiment. The luminance range of the monitor was 0.004 footlamberts for the driving level of 0 and 50 footlamberts for the driving level of 255.

The Comtal 10/24 system has three 1024 × 1024 8-bit frame buffers which allow fast display of a sequence of test patterns. The 64 × 64 test pattern was at the center of the upper left quarter of the monitor. A zoom factor of 2 × 2 was performed by hardware to make the test pattern occupy the center region of the screen.

The subjects entered the answers through a keyboard. The four consecutive keys of *1*, *2*, *3*, and *enter* at the lower right corner of the keyboard were the only ones needed after the program started.

Software for the Experiment

Programs in the C programming language were written to display the sequence of test patterns and collect data. The start-up time of the program was about 80 seconds. The inter-trial interval was 6 seconds on average. No subject felt the procedure uncomfortable, and all of them understood the operation of the program after a brief explanation and several minutes of practice. All stimuli and the user answers were recorded by the computer.

[1]UNIX is a trademark of AT&T Bell Laboratories.

Environmental Layout

The subject was seated approximately 85 centimeters from the display with viewing direction perpendicular to the screen. Each side of the test pattern extended for 27 mm on the monitor and thus subtended 2° of visual angle. The keyboard rested on a side table, which the subject could comfortably reach.

The perception laboratory provided good environmental control in the experimental area. The ambient light was set to 4 footlamberts. The black-painted room minimized undesired reflections on the monitor.

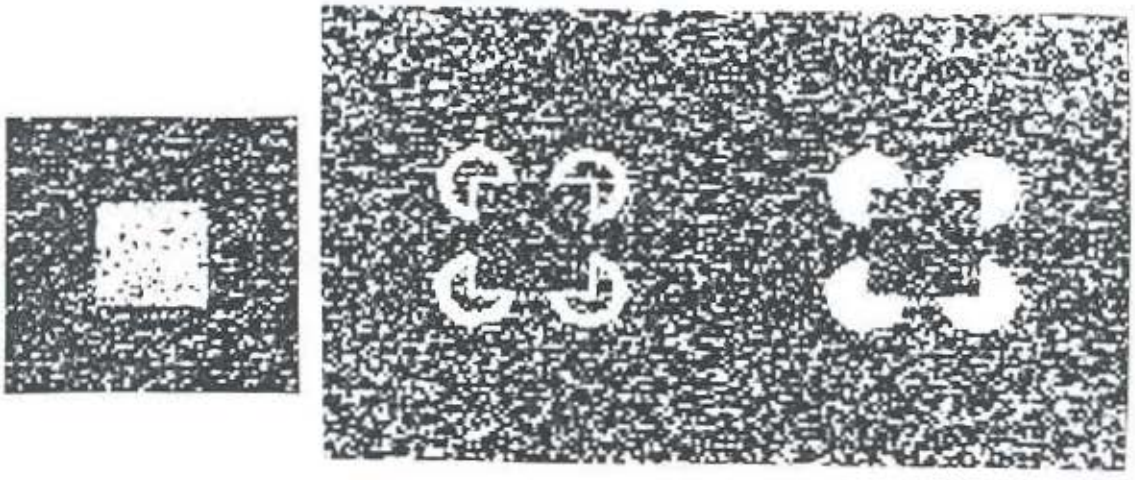
Test Images

There were two sets of test patterns: *sqr* and *ksq*, each with spatial sampling of 64×64 pixels. The test pattern *ksq* was composed of four pacmen with the inducing edges properly aligned to form an illusory square. The opening angles of the pacmen were 90°, and the radius was eight pixels. A test pattern generation program put the four pacmen at the proper locations according to the size of the illusory square. All inducing edges were sharp. The sizes of the square were selected to be 20, 24, 28, and 32 pixels so that the illusory contour was substantial in each case. The test patterns of *sqr* were simpler. There were again 4 sizes of 20, 24, 28, and 32 pixels. The boundaries of the test pattern *sqr* were step edges.

The test pattern was shown on a grey-scale monitor with dynamic range of 0 – 255 driving levels. The range of pixel intensity of the test pattern without noise was 88 – 168. This decision was a compromise between the need for a contrast to show the signal and the need for an intensity range to show the noise. Note that when the noise-to-signal ratio is above 0.33, the thresholding at driving levels 0 and 255 occurs. This operation might affect the subjects' performance though the current data do not indicate so.

Gaussian white noise was added to the image. Remember that the standard deviation, σ, of the Gaussian was defined as the strength of noise, and the difference between the average intensities of the background and the signal was the strength of signal. The noise-to-signal ratio was defined in equation 4.6,

$$noise - to - signal\ ratio = \frac{\sigma_{noise}}{I_{sig} - I_{bkg}},$$



a b

Figure 6.6: An example of the the test stimuli. (a) *sqr24* (b) *ksq28*, where the number indicates the size of the square.

and served as the independent variable of the experiment. Note that when the σ was large, the intensities were truncated at 0 and 255, respectively. This truncation occurred when the noise-to-signal ratio was above 1/3, and the number of pixels with truncated intensities reached 1% of the total number of pixels when noise-to-signal ratio was above 0.5. Moreover, for reasons of display speed all the data points were obtained with the same noise pattern. Further studies are needed to find out the effect of different noise patterns.

A control pattern was shown beside the test pattern *ksq* to help the subject decide whether the central region was darker than the surrounding region or not. The control pattern was composed of four pacmen with only outlines (shown in Figure 6.6). The central squares of the control and test patterns had the same sizes. A single noise pattern was added to both patterns. Since the control pattern was known not to generate the subjective contour, when the difference between the central region and the surrounding region of the test pattern is small, the pattern can help a subject to decide whether there is indeed a brightness difference. Moreover, to avoid the possible interference from the dark background, the substantial area outside the two patterns was filled in with random noise.

The Task

For the set of stimuli of an objective square, the question for a subject to answer was: *do you see an object in the stimulus*? The question for the stimuli of the subjective contour was not as obvious. A psychophysical experiment on the Kanizsa square [Halpern, 1987] suggested that the question of "do you see a difference between the brightness in the central region and the surrounding region?" was reasonable. This question was adopted in my experiment. The subjects were instructed to compare the brightness difference between the central region and surround in the control pattern when the answer to the above question was unclear.

Subjects

Eight naive subjects (5 female, 3 male) participated in the experiment. All subjects were graduate students or had attained a graduate degree recently. Most of them did not have previous experience in visual psychophysical experiments. The observers were guaranteed that the result would not be associated with each individual and were informed that it was more important to be consistent than to be competitive with respect to the threshold.

Procedure

Each subject ran through the experiment for 2 sessions on two separate dates. During each session a subject did 8 runs of experiment, each for a different test pattern. The runs for the same kind of test pattern were performed consecutively, but in a random sequence.

Before running the experiment, a subject was first seated in front of the monitor and an explanation of the experimental setup and the stimuli were given. Before and during the experiment, the subjects were allowed to ask the experimenter to give the explanation again.

The experimental procedure is summarized in the following. Two patterns were first shown to the subject. One was with small amount of noise and the other with large noise. These noise levels were selected from a preliminary study which showed that one could detect the signal in the less noisy pattern but could not in the other. All subjects confirmed this before the experiment. During the experiment each test pattern was shown

subject id	size 20	size 24	size 28	size 32	mean
hh	0.6750	0.6750	0.3875	0.1875	0.4813
hh	0.5750	0.3500	0.3375	0.2250	0.3719
hs	0.2750	0.0875	0.1500	0.0750	0.1469
hs	0.2375	0.1500	0.1625	0.1125	0.1656
hw	1.0875	0.7750	0.9000	0.4875	0.8125
hw	1.0875	0.8500	0.9125	0.6750	0.8813
js	0.8000	0.7125	0.6000	0.4375	0.6375
js	0.8000	0.6125	0.6750	0.6125	0.6750
lm	0.8750	0.4625	0.4125	0.3500	0.5250
lm	1.0750	0.7500	0.8625	0.6250	0.8281
mc	1.0250	0.6750	0.7875	0.6125	0.7750
mc	0.8250	0.6500	0.6000	0.5125	0.6469
sj	0.9750	0.8500	0.8000	0.6625	0.8219
sj	0.6875	0.7000	0.5625	0.5000	0.6125

Table 6.1: The experimental results of test pattern *ksq*.

until the subject responded. If the subject perceived the signal, the key *1* was entered, otherwise *2*. The subject could change each decision until a *3* was entered. Each positive response increased the noise level by a fixed unit, and each negative one decreased the noise by three units for the next stimulus. After 12 turnings of the data, i.e., 12 up sequences and down sequences in total, the program stopped and estimated the threshold noise level based on the values of the last 4 upper turning points [Wetherill et al., 1963].

The two test patterns first shown were used in the catch trials. Each test pattern was inserted into the trial sequence randomly. The answers to the catch trials were separately recorded and printed.

In each session, the subject ran through eight test patterns consecutively. The first run served as a trial run to allow the subject to become familiar with the experiment. After the subject completed all four sizes of the *ksq*, the test pattern of *sqr* then followed. The eight runs took about one and a half hours.

subject id	size 20	size 24	size 28	size 32	mean
hh	1.4000	1.4000	1.5500	1.5750	1.4812
hh	1.3000	1.2750	1.4750	1.6250	1.4187
hs	1.7000	1.5500	1.7500	1.9000	1.7250
hs	1.7000	1.7000	1.8000	2.1500	1.8375
hw	1.2250	1.2000	1.4000	1.4250	1.3125
hw	1.1500	1.3250	1.4000	1.3000	1.2937
lm	2.3000	1.6000	2.3500	2.8000	2.2625
lm	2.2750	2.7000	2.5750	2.4750	2.5063
lw	1.8000	1.6250	1.7750	1.7000	1.7250
lw	1.8750	1.7250	1.8750	2.0000	1.8688
mc	1.0250	1.1750	1.0500	1.2250	1.1187
mc	0.8750	1.0250	1.1000	1.1000	1.0250
sj	1.1250	1.2750	1.3250	1.4250	1.2875
sj	1.2750	1.4750	1.4250	1.5750	1.4375

Table 6.2: The experimental results of test pattern *sqr*.

Results

The experimental results are listed in Tables 6.1 and 6.2. Each row in the table shows the four threshold noise levels that a subject gave for a detection target of a certain size. Note that each threshold noise level was obtained from one run of the experiment and the 4 thresholds in the row was recorded consecutively. If any of the 4 data points did not pass the consistency test, that row of data were discarded. Among the eight subjects for each experiment, there were 2 out of 16 rows discarded for each test pattern.

Both tables show that the data obtained on two separate dates for a single subject do not differ much — the average of the difference between the results obtained during the two separate sessions is 0.1589 with standard deviation 0.2011 for test pattern *sqr* and 0.1375 with standard deviation 0.1153 for test pattern *ksq*. The subjects were consistent. Though there was substantial variation across people, since the goal was to analyze the trend between the two sets of data from humans and from the algorithm, this variation was an error term and was handled by the analysis method described in section 6.4. Tables 6.3

subject id	size 20	size 24	size 28	size 32
hh	0.6250	0.5125	0.3625	0.2063
hs	0.2563	0.1188	0.1563	0.0938
hw	1.0875	0.8125	0.9062	0.5813
js	0.8000	0.6625	0.6375	0.5250
lm	0.9750	0.6063	0.6375	0.4875
mc	0.9250	0.6625	0.6938	0.5625
sj	0.8312	0.7750	0.6813	0.5812
mean	0.7857	0.5929	0.5822	0.4339
SD	0.2753	0.2318	0.2460	0.1994

Table 6.3: The average threshold noise levels of each subject for test pattern *ksq*.

and 6.4 give the means of the threshold noise levels for each subject on each test patterns, as well as the means and sample standard deviations (SD) across all subjects.

6.3 My Algorithm vs. Noise

This section describes how random noise affected the segmentation performance of my algorithm. In the following the procedure for determining the threshold noise level is briefly described.

The same test patterns as used for the psychophysical experiment were used as the input to the algorithm except that the test patterns of *ksq* were presented without the control pattern and with a smaller background region. A single configuration of four edge filters with $\sigma_\perp = 0.75$ and $\sigma = 0.75, 1.5, 3.0, 6.0$, and a single edge threshold of 0.03 was applied to all the noisy test patterns. After various levels of noise was added to the images, each of these test patterns was input to the segmentation algorithm.

The noise level at which the algorithm fails to segment the test pattern properly is recorded as the threshold noise level for the algorithm. For this experiment I use an operational definition of the failure of segmentation: if the the area of the segmented region differs from that of the target region by more than 25%, then the segmentation is deemed as failure. Empirically, there are two ways a failure can occur: either a portion

subject id	size 20	size 24	size 28	size 32
hh	1.3500	1.3375	1.5125	1.6000
hs	1.7000	1.6250	1.7750	2.0250
hw	1.1875	1.2625	1.4000	1.3625
lm	2.2875	2.1500	2.4625	2.6375
lw	1.8375	1.6750	1.8250	1.8500
mc	0.9500	1.1000	1.0750	1.1625
sj	1.2000	1.3750	1.3750	1.5000
mean	1.5018	1.5036	1.6321	1.7339
SD	0.4631	0.3481	0.4457	0.4919

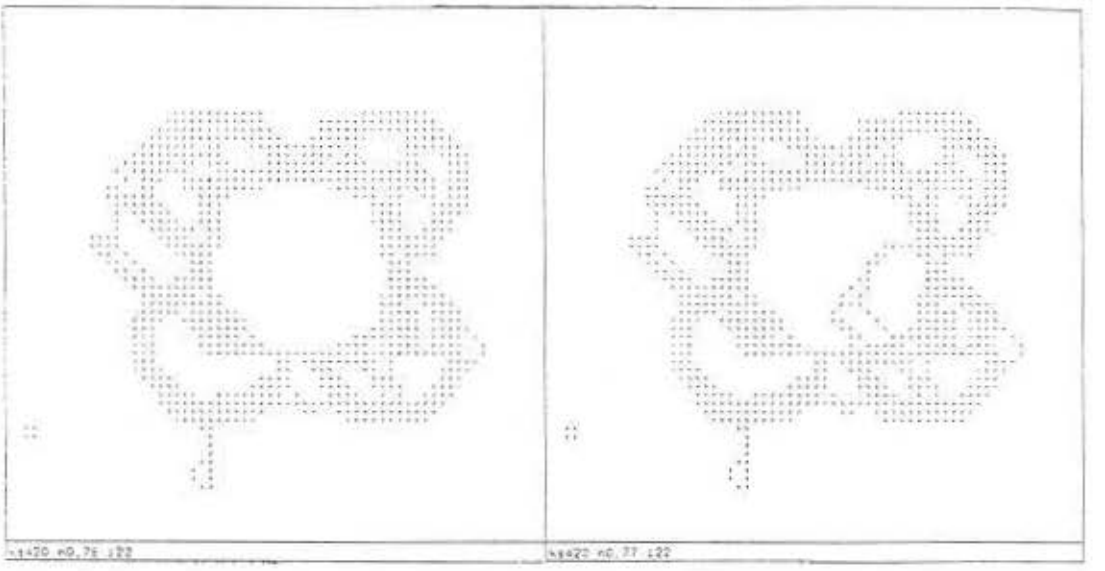
Table 6.4: The average threshold noise levels for each subject of test pattern *sqr*.

test pattern	size 20	size 24	size 28	size 32
Kanizsa square	0.765	0.745	0.550	0.095
square	0.325	0.465	0.405	0.485

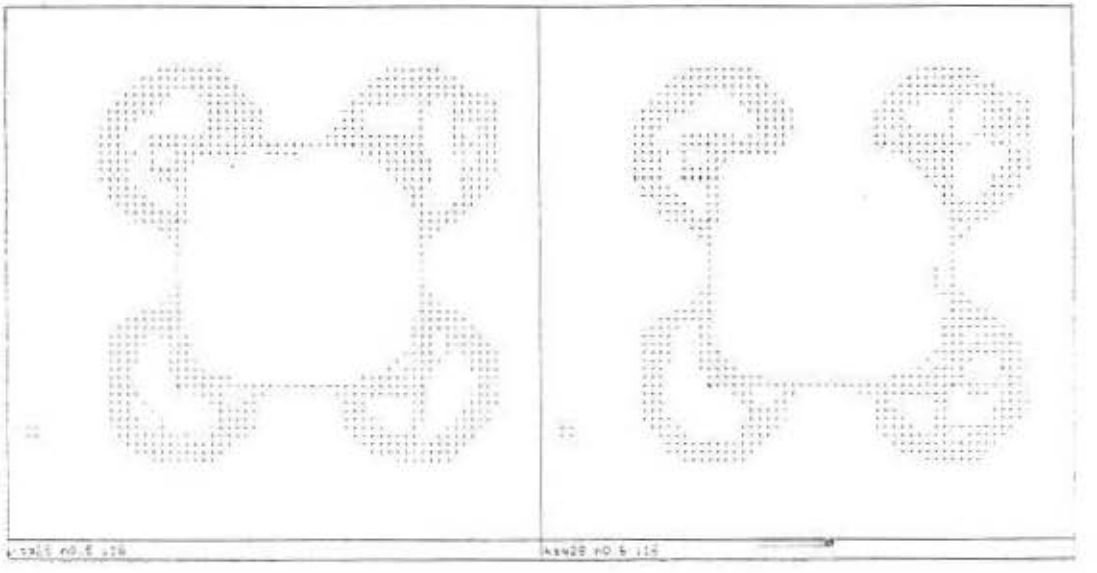
Table 6.5: The threshold noise levels measured by my algorithm.

of the boundary of the detection target disappears or extra contours appear. Figures 6.7 and 6.8 demonstrate the segmentation results with noise levels at which the algorithm was deemed to give improper segmentation.

The determination of the threshold noise level proceeded as follows. Since only a single noise level could be tested at one time, the range of possible noise levels was first determined empirically. A series of images, with fixed difference in noise-to-signal ratio, was created. These images were fed to the algorithm in a sequence similar to a binary search to find a range including the threshold noise level. This procedure was repeated until the desired accuracy was obtained. Also, a fixed random number seed was used to add noise to all the test images. The effect of different noise patterns requires further research. The results are shown in Table 6.5.



(a) (b)



(c) (d)

Figure 6.7: Two examples of the segmented results showing the threshold noise level for test pattern *ksq*. Top row are with central region of 20×20 pixels, while that of the bottom row are 28×28 pixels. Pattern (a) has noise-to-signal ratio of 0.76 and segmentation is deemed as good, (b) has noise-to-signal ratio of 0.77 ad is deemed as bad. Similarly, patterns (c) and (d) are with noise-to-signal ratio of 0.5 and 0.6 and are deemed as good and bad segmentation, respectively.

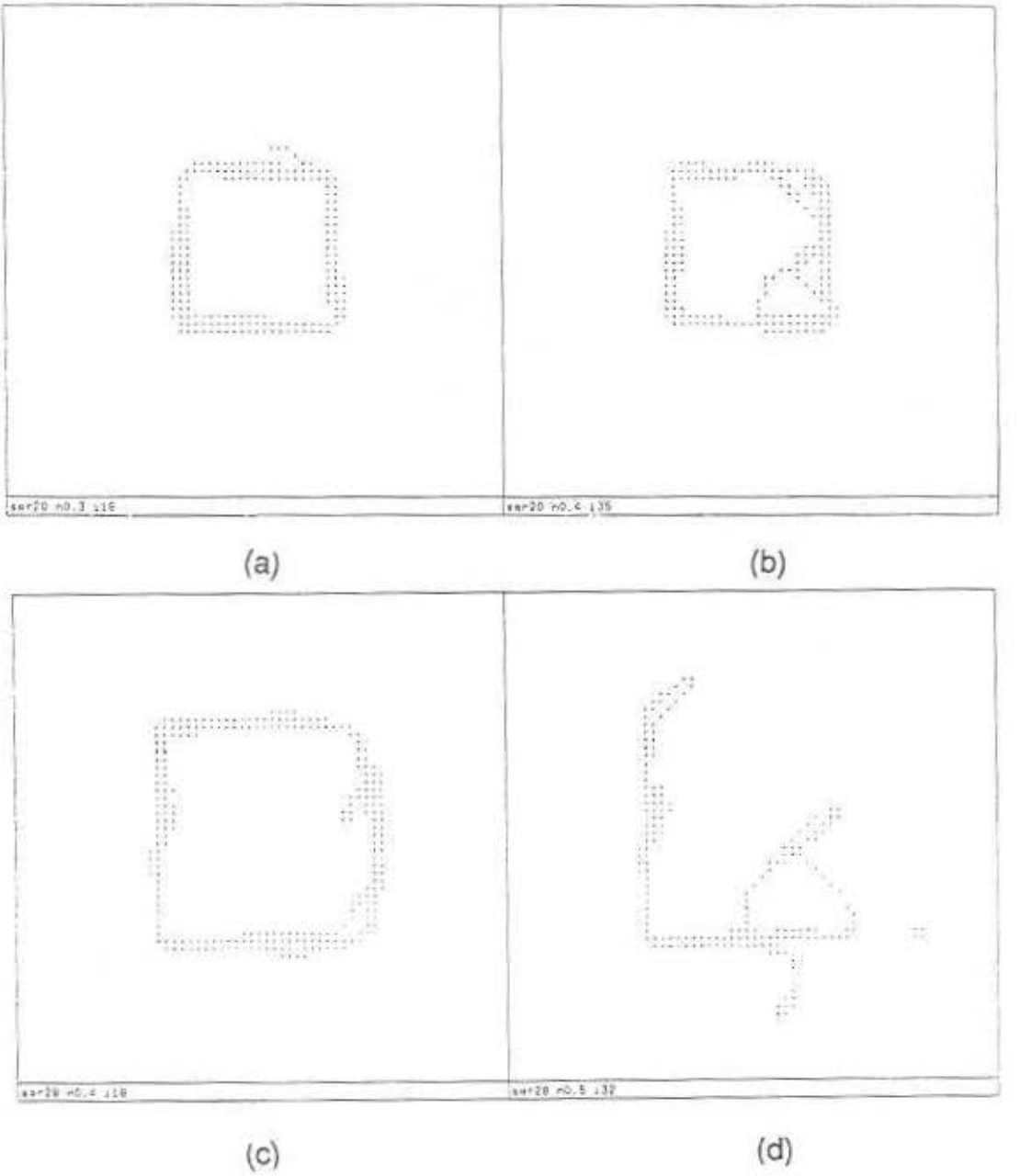


Figure 6.8: Two examples of the segmented results showing the threshold noise level for test pattern *sqr*. Top row squares are 20 × 20 pixels, while that of the bottom row are 28 × 28 pixels. Pattern (a) has noise-to-signal ratio of 0.3 and segmentation is deemed as good, (b) has noise-to-signal ratio of 0.4 ad is deemed as bad. Similarly, patterns (c) and (d) are with noise-to-signal ratio of 0.4 and 0.5 and are deemed as good and bad segmentation, respectively.

6.4 A Comparison of Human Vision and My Algorithm

The Statistical Method

This subsection briefly describes the method of data analysis. A profile analysis for two independent groups was used to compare the parallelism between the trend of the psychophysical data and the data obtained by my algorithm. The analysis was in the form of hypothesis testing. The *null hypothesis*, H_0, was that the random noise affected the human object detection and algorithm segmentation in a similar way. More precisely, let $\overline{X} = (\overline{x_1}, \overline{x_2}, \overline{x_3}, \overline{x_4})$ be the vector of the sampled mean of the experimental results and $\overline{x_1}, \overline{x_2}, \overline{x_3}, \overline{x_4}$ be the average of N_1 observers' responses for test pattern of size 20, 24, 28, and 32, respectively. Likewise, let $\overline{Y} = (\overline{y_1}, \overline{y_2}, \overline{y_3}, \overline{y_4})$ be the vector of the sampled mean for N_2 results obtained by running the algorithm against test inputs with noise.

$$H_0 : C\overline{X} = C\overline{Y}, \tag{6.3}$$

where

$$C = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{6.4}$$

The *alternative hypothesis* was that the two profiles were not parallel. According to [Morrison, 1976], Hotelling's T^2 is a reasonable statistic for the test.

$$T^2 = \frac{N_1 N_2}{N_1 + N_2}(\overline{X} - \overline{Y})'C'(CSC')^{-1}C(\overline{X} - \overline{Y}), \tag{6.5}$$

where CSC'/N is the sampled covariance matrix. T^2 is related to the F distribution by

$$F = \frac{N_1 + N_2 - p}{(N_1 + N_2 - 2)(p-1)}T^2, \tag{6.6}$$

where p is the number of elements in the mean vector. For this experiment, $p = 4$. H_0 would be rejected at the level of significance, α, if the observed F exceeded $F_{\alpha;p-1,N_1+N_2-p}$.

Results

According to Table 6.3 and 6.4, N_1 for both *ksq* and *sqr* are 7, $N_2 = 1$, and $p = 4$. Thus the critical values to reject H_0 for various α's are listed below.

α	0.1	0.05	0.025	0.01
$F_{\alpha;3,4}$	4.19	6.59	9.98	16.69

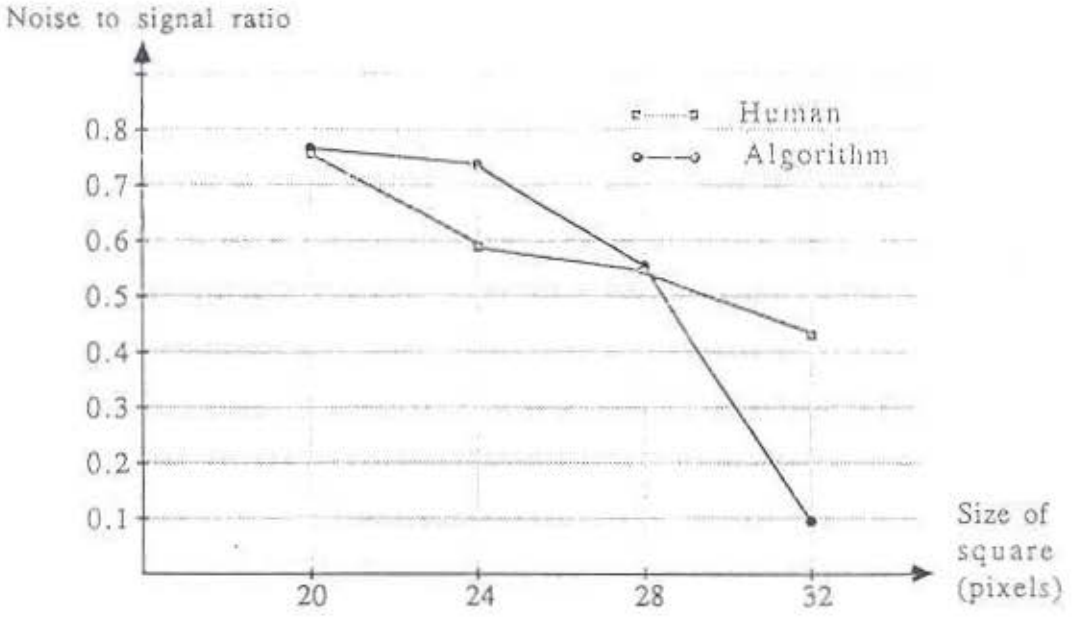


Figure 6.9: A plot of the results from psychophysical experiments and test of the algorithm on test pattern *ksq*.

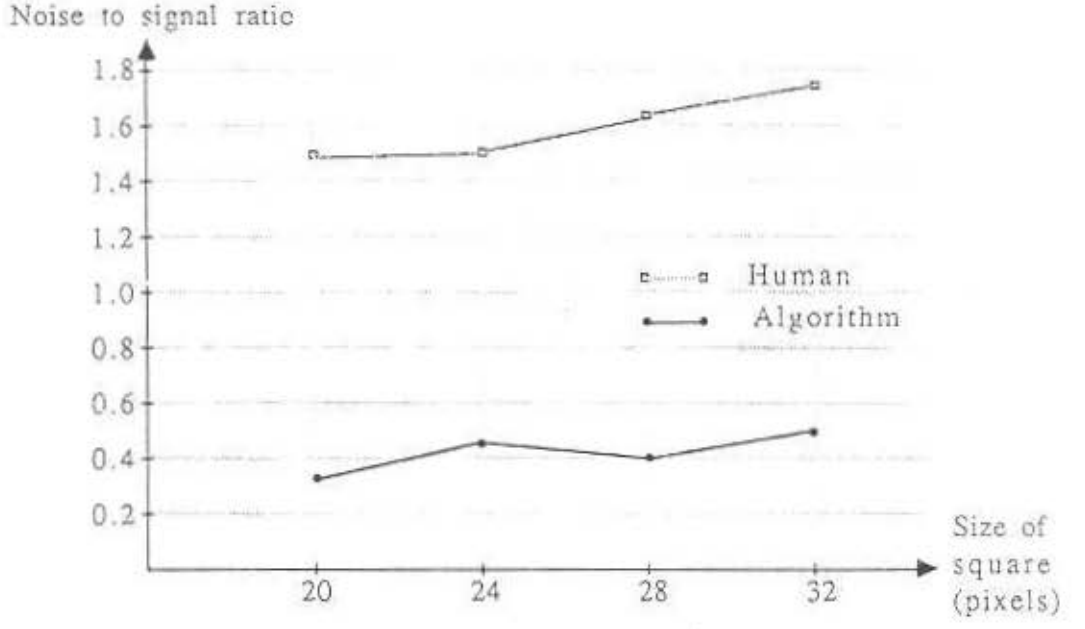


Figure 6.10: A plot of the results from psychophysical experiments and test of the algorithm on test pattern *sqr*.

The means of the psychophysical data, with the results obtained from successively applying the algorithm to the test inputs, are plotted in Figures 6.9 and 6.10. For the illusory contour, $T^2 = 27.3855, F = 6.0857 > 4.19$. Hence H_0 is rejected for $\alpha = 0.1$. For $\alpha = 0.05$, $F = 6.0857 < 6.59$; the data fail to reject H_0. For the square of physical contour, $T^2 = 3.1350, F = 0.6967 < 4.19$. Hence for the physical square the data fail to reject H_0.

Although the psychophysical data reject the hypothesis of parallel profiles for the test pattern *ksq* for $\alpha = 0.1$, Figure 6.9 shows that, for a bigger object less noise is required for both human and my algorithm to discount the effect of subjective contour than that for a smaller object. It is encouraging that both curves are monotonically decreasing.

For the test pattern *sqr*, the psychophysical data shows that more noise is needed for a bigger object as expected, and the results obtained by applying my algorithm to the test inputs of various sizes show a similar trend. The profile analysis does not reject H_0, confirming this point.

Chapter 7

Conclusions

This chapter summarizes the considerations behind the design of my algorithm, its implementation, and evaluation. It concludes with a discussion about possible future research directions.

7.1 Summary of the Algorithm

My algorithm is based on the assumption that an effective vision algorithm should be based on a connectionist architecture because of the strict speed constraint of visual tasks. The effectiveness of the connectionist approach is possible because our visual world is highly regular — the shape of many objects have common properties. Therefore, if the world constraints, viewed as segmentation cues, are built into the detection hardware, reasonable segmentation can be obtained in real-time. In this sense the definition of connections is really the process of including knowledge about our visual world into the physical architecture of the visual system.

The special-purpose circuits provide high processing speed but have their constraints too: they can implement only local operations; their functions are rigid and their capabilities limited. Hence side effects may occur. For example, the need of special-purpose circuitry for fast information reduction in early vision causes optical illusions. One way to cope with this deficiency is to have repetitions of the circuitry with each unit covering a different range of spatial and temporal frequencies. This notion is the basis underlying multiple segmentation mechanisms and multiple edge filters within each segmentation mechanism.

My edge-based segmentation algorithm consists of three stages: edge filtering, corner detection, and a spatial coherence check. All the later stages depend on edge filtering. After studying several functions for edge filter kernels, the multivariate Gaussian filters

at four orientations and with σ and $\sigma_\perp$ following certain relationships were selected. A problem with this template scheme is the possibility of false alarms — significant response in an edge filter does not necessarily indicate an edge. Several measures were taken to cope with this problem. First, multiple elongated edge filters were applied at each sampling point to handle the various levels of noise in the image. Second, artifact cancellation was used to discount the sampling artifacts based on the edge strengths at a single sampling location. Third, a scheme for detecting corners in the image was used because edge filtering does not give proper information about object boundaries there. Fourth, a spatial coherence check was developed to verify the local information based on spatial context.

7.2 Summary of the Implementation and Evaluation

A program implementing this edge-based segmentation algorithm was developed, and this program was applied to many test patterns to investigate the properties of the algorithm. It was found that the algorithm gives reasonable segmentation under various conditions, the multiscale mechanism in the algorithm can segment an object contour at different levels of detail, and the algorithm behaves reasonably well against random noise. The algorithm has its problems too: the current implementation does not work well for lines in the image and has difficulty with a sharp angle or a T-junction with a small intercepting angle.

A version of Grossberg's *boundary contour system* was also implemented, and the same input patterns used to test my algorithm were input to this program. The results show that, with the parameters properly selected, the *boundary contour system* also gives reasonable results for all test patterns within a few iterations. Compared to my algorithm, the *boundary contour system* handles lines and may explain the Ehrenstein illusion but has the following shortcomings relative to my algorithm:

- The system configuration is image-dependent.
- The performance depends on many system parameters.
- The model does not work well for noisy images.

The effect of random noise on my algorithm's segmentation capability was further investigated to compare with data from human subjects. Psychophysical experiments measuring the noise level at which a subject fails to detect an object in the stimulus were

designed and performed. The results obtained from the experiment and my algorithm showed a similar trend.

It may be concluded that an algorithm with multiple elongated edge filters, explicit corner detection, and explicit local spatial coherence checking, when applied at many resolutions with corresponding spatial sampling, can detect both real and subjective contours successfully. The performance is competitive with Grossberg's method and comparable in certain ways to human performance.

7.3 Future Work

My algorithm is by no means a complete model of the human visual system, but it is an example showing that computer simulation provides a useful means for studying vision. Besides, the knowledge thus obtained can be used to design an effective vision machine. This section first lists several points which I found, during this research, to be either possible alternatives to the current design decisions or require further investigation. Lastly the section outlines the directions in which research on this connectionist method might expand.

7.3.1 Design Decisions Worth a Second Thought

Effect of Noise. The same noise pattern with different magnitude was inserted in the input patterns for testing both my algorithm and human vision. The effect of different noise patterns should be further studied.

Coherence Rules for Edges and Corners. The coherence rules for edges and corners may be improved. For example, the coherence rules for diagonal corners consider a location beyond the neighboring 8 pixels of the position under consideration. Therefore, a contour may have a one-pixel gap on it. What is the probability of this situation? Can we leave out this position in the definition?

T-junctions and Cross-Junctions. The T-junctions and cross-junctions are places where object boundaries interact; as mentioned in Chapter 4, the detection of these features and the design of corresponding coherence rules require more thorough study.

On the Observer Experiment. The results attained from the observer experiments were interesting but not decisive: the task was specific; the effect of noise pattern and intensity thresholding needs further investigation; a fixed viewing condition (e.g., visual angle covered by a test stimulus) was used through the experiment; the test patterns applied did not cover the whole range of possible object sizes; the threshold noise levels obtained for test patterns *sqr* vary by a small amount and might need more experimental data to confirm the trend. More work is required.

7.3.2 Future Directions

Line Detection. As pointed out before, the algorithm does not work well on lines or thin objects in the image. Special filters and follow-up processing are necessary to obtain satisfactory behavior for such objects. Second-order directional derivatives of a two-dimensional Gaussian may serve as line detectors as physiological evidence suggested. A different spatial coherence rule is required for lines. Moreover, the combination of the detected lines and edges is also worth further investigation.

More Precise Corner Detection. As suggested in Chapter 4, a corner may be detected based on the edge strengths in a bigger area. Furthermore, edge filtering can be applied on a hexagonal sampling grid. Will these measures improve the algorithm performance?

Interaction with Other Segmentation Cues. Biological vision applies multiple segmentation cues. Besides intensity differences, there may be color, texture, motion, and depth differences. What is the mechanism to extract these segmentation cues from an image? How can the results from different segmentation cues be combined to give a segmented image?

Connections from Learning. Adaptivity enables a connectionist model to incorporate knowledge into its parallel distributive architecture. There have been many learning paradigms in the field of neural networks. It would be interesting to see that if the connection patterns described in my algorithm can be learned from an initially random network.

Multiscale mechanism. Lastly, as mentioned in Chapter 5, proper multiscale organization, integrating individually function-limited units into a globally effective system,

is the key to the success of this algorithm. There are many related problems to be solved. For example, during edge filtering, the support of the edge filter, when applied at two successive sampling points, overlap with each other. What is the optimal relationship between the edge filter size and the sampling interval? What is the optimal number of scales? How should the outputs at different scales be combined? Answers to these questions await further investigation.

Besides the functional aspect, the multiscale mechanism also provides the necessary redundancy for reliability — the visual system is elegantly built: a unit, when it is normal, enhances the system performance, and when it is not, it does not cause the system to fail. I conjecture that this mechanism is not only essential for vision but for the brain as well. There is still much to learn.

REFERENCES

Asada, Haruo and Michael Brady, *The Curvature Primal Sketch*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-8**(1):2-14, 1986.

Ashkar, G.P. and J.W. Modestino, *The Contour Extraction Problem with Biomedical Applications*, Computer Graphics and Image Processing, **7**:331-355, 1978.

Attneave, F. , *Some Informational Aspects of Visual Perception*, Psychological Review, **61**:183-193, 1954.

Babaud, Jean, A.P. Witkin, M. Baudin, Richard O. Duda, *Uniqueness of the Gaussian Kernel for Scale-Space Filtering*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-8**(1):26-33, January 1986.

Ballard, Dana H. and Christoper M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

Barlow, H.B., *Understanding Natural Vision*, in Physical and Biological Processig of Images, O.J. Braddick and A.C. Sleigh (ed.), Springer-Verlag, Berlin, 2-14, 1983.

Barlow, H.B., *Why Have Multiple Cortical Areas*, Vision Research, **V26**(N1):81-90, 1986.

Baugher S. and A. Rosenfeld, *Corner Detection and Localization in a Pyramid*, Technical report CAR-TR-298, Center for Automation Research, University of Marryland, June 1987.

Bergholm, Fredrik, *Edge Focusing*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-9**(6):726-741, November 1987.

Berzins, V., *Accuracy of Laplacian Edge Detectors*, Computer Graphics and Image Processing, **27**:195-210, 1984.

Blakemore C. and R. Over, *Curvature Detectors in Human Vision ?*, Perception, **3**:3-7, 1975.

Bracewell, Ronald, N., *The Fourier Transform and its Applications*, McGraw-Hill Inc., 1986.

Brownlee, K.A., J.L. Hodges and M. Rosenblatt, *The Up-and-Down Method with Small Samples*, Journal of American Statistics Association, **48**:262-277, 1953.

Campbell, F.W. and J. Robson, *Application of Fourier analysis to the visibility of gratings*, Journal of Physiology (London), **197**:417-424, 1977.

Canny, John, *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-8**(6):679-698, November 1986.

Chien, Y.P. and K.S. Fu, *A Decision Function Method for Boundary Detection*, Computer Graphics and Image Processing, **3**:125-140, 1974.

Coggins, James M., Fredric S. Fay, and Kevin E. Fogarty, *Development and Application of a Three-Dimensional Artificial Visual System*, Computer Methods and Programs in Biomedicine, **22**:69-77, 1986.

Cohen, Michael A. and Stephen Grossberg, *Neural Dynamics of Brightness Perception: Features, Boundaries, Diffusion and Resonance*, Perception & Psychophysics, **36**(5): 428-456, 1984.

Cornsweet, Tom N., *Visual Perception*, Academic Press, New York, 1970.

Crowley, James L. and Alice C. Parker, *A Representation for Shape Based on Peaks and Ridges in the Difference of Low-Pass Transform*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-2**(2):156-159, March, 1984.

Dobbins, Allan, Steven W. Zucker, and Max S. Cynader, *Endstopping in the Visual Cortex: a Neural Substrate for Calculating Curvature*, Technical Report, Computer Vision and Robotics Laboratory, McGill Research Centre for Intelligent Machines, McGill University, 1988.

Davis, Larry S., *Understanding Shape: Angles and Sides*, IEEE Transactions on Computers, **C-26**(3):236-242, 1977.

Duda, O. Richard and Peter E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, Inc. 1973.

Feldman, J.A., *Connections*, Byte, April 1985.

Fischler, Martin A. and Robert A. Elschlager, *The Representation and Matching of Pictorial Structures*, IEEE Transactions on Computers, January 1973.

Frisby, John P., *Seeing, Illusion, Brain and Mind*, Oxford University Press, 1980.

Fukushima, Kunihiko, *Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern recognition*, Neural Networks, **1**(2):119-130, 1988.

Ginsburg, Author P., *Visual Information Processing Based on Spatial Filters Constrained by Biological Data*, Dissertation for Ph. D., University of Cambridge, England, 1977. Also published as Air Force Aerospace Medical Research Laboratory Technical Report AMRL-TR-78-129, December, 1978.

Gross, C.G., C.E. Rocha-Miranda, and D.B. Bender, *Visual Properties of Neurons in Inferotemporal Cortex of the Macaque*, Journal of Neurophysiology, **35**:96-111, 1972.

Grossberg, Stephen and Ennio Mingolla, *Neural Dynamics of Perceptual Groupings: Textures, Boundaries, and Emergent Segmentations*, Perception & Psychophysics, **38**(2): 141-171, 1985.

Grossberg, Stephen and Ennio Mingolla, *Computer Simulation of Neural Networks for Perceptual Psychology*, Behavior Research Methods, Instruments, & Computers, 601-607, 1986.

Grossberg, Stephen and Ennio Mingolla, *Neural Dynamics of Surface Perceptions: Boundary Webs, Illuminants, and Shape-from-Shading*, Computer Vision, Graphics, and Image Processing **37**:116-165, 1987.

Halpern, Diane F., *The Functional Equivalence of Objective and Illusory Brightness Enhancement*, Chapter 18 of The perception of Illusory Contours, Perry, Susan and Glenn E. Meyer (ed.), Springer-Verlag New York Inc., 1987.

Hildreth, E.C., *Implementation of a Theory of Edge Detection*, MIT Artificial Intelligence Laboratory, Technical Report AI-TR-579, 1980.

Hildreth, E.C., *The Detection of Intensity Changes by Computer and Biological Vision*, Computer Vision, Graphics, and Image Processing, **22**:1-27, 1983.

Hong, Tsai-Hong, K.A. Narayanan, Shmuel Peleg, Azriel Rosenfeld and Teresa Silberberg, *Image Smoothing and Segmentation by Multiresolution Pixel Linking: Further Experiments and Extensions*, IEEE Transactions on Systems, Man, and Cybernetics, **SMC-12**(5):611-622, September 1982.

Hubel, D.H., and T.N. Wiesel, *Functional Architecture of Macaque Monkey Visual Cortex*, Procceedings of Royal Society of London, **B 198**:1-59, 1977.

Hubel, D.H., and T.N. Wiesel, *Brain Mechanisms of Vision*, Scientific American, September 1979.

Hueckel, Manfred H., *A Local Visual Operator Which Recognizes Edges and Lines*, Journal of Association for Computing Machinery, **20**(4):634-647, October 1973.

Kanizsa, G., *Organization in Vision*, Praeger, New York, 1979.

Kass, Michael, Andrew Witkin, and Demetri Terzopoulos, *Snakes: Active Contour Models*, International Journal of Computer Vision, **1**(275-000):321-331, 1987.

Koenderink, J.J. and van Doorn, A.J., *Invariant Features of Contrast Detection, an Explanation in Terms of Self-Similar Detector Arrays*, Journal of Optical Science of America, **72**:83-87, 1982.

Koenderink, J.J., *The Structure of Images*, Biological Cybernetics, 1984.

Koenderink, J.J., *Image Structure*, Mathematics and Computer Science in Medical Imaging, NATO ASI Series, Series F: Computer and Systems Sciences, **39**:67-104, MA Viergever-A Todd-Pokropek (ed.), Springer-Verlag, 1988.

Korn, Axel F., *Toward a Symbolic Representation of Intensity Changes in Images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-10**(5):610-625, September, 1988.

Land, Edwin H. and John J. McCann, *Lightness and Retinex Theory*, Journal of the Optical Society of America, **61**(1):1-11, January 1971.

Levitt, H, *Transformed Up-Down Methods in Psychoacoustics*, Journal of the Acoustic Society of America, **49**(2-2):467-477, 1971.

Liftshitz, Lawrence M., *Image Segmentation via Multiresolution Extrema Following*, Dissertation, Department of Computer Science, University of North Carolina at Chapel Hill, 1987.

Levine, Martin D., *Vision in Man and Machine*, McGraw-Hill, 1985.

Linsker, Ralph, *From Basic Network Principles to Neural Architecture: Emergence of Orientation Columns*, Proceedings of National Academy of Science of USA, **83**:8779-8783, November 1986.

Marr, David, *Vision*, San Francisco, Freeman 1982.

Marr, David and Tomaso Poggio, *A Theory of Human Stereo Vision*, Proceedings of Royal Society of London, B **204**:301-428, 1979.

Marr, David, Tomaso Poggio and Ellen C. Hildreth, *Smallest Channel in Early Human Vision*, Journal of the Optical Society of America, **70**(7):868-870, July 1980.

Marr, David and Ellen C. Hildreth, *Theory of Edge Detection*, Proceedigns of Royal Society of London, B **207**:187-217, 1980.

Marroquin, J., S. Mitter, and Tomaso Poggio, *Probabilistic Solution of Ill-posed Problems in Computer Vision*, Journal of the American Statistical Association, **82**(397):76-89, March 1987.

Mead, Carver A. and M.A. Mahowald, *A Silicon Model of Early Visual Processing*, Neural Networks, **1**(1):91-97, 1988.

Minsky, Marvin and Seymour Papert, *Perceptrons: An Introduction to Computational Geometry* , The MIT Press, 1968.

Mishkin, Mortimer and Tim Appenzeller, *The Anatomy of Memory*, Scientific American, June 1987.

Morrison, Donald F., *Multivariate Statistical Methods*, McGraw-Hill Book Company, New York, 1976.

Nilsson, Nils J., *Principle of Artificial Intelligence*, Tioga Publishing Company, 1980.

Oliver, William R., *Personal Communication*, University of North Carolina at Chapel Hill, 1987.

Phillips C.G., Zeki, S. and Barlow, H.B., *Localization of Function in the Cerebral-cortex-Past, Present and Future*, Brain, **V107**:328-361, March 1984.

Pizer, Stephen M., *Lecture Notes for Course on Image Processing*, The University of North Carolina at Chapel Hill, 1987.

Pizer, Stephen M., *Lecture Notes for Course on Solid Shape*, The University of North Carolina at Chapel Hill, 1988.

Pizer, Stephen M., William R. Oliver, and Sandra H. Bloomberg , *Hierarchical Shape Description via the Multiresolution Symmetric Axis Transforms*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-9**(4):505-511, July 1987.

Pizer, Stephen M., *Multiscale Methods and the Segmentation of Medical Images*, to appear in The Formation, Handling and Evaluation of Medical Images, Springer-Verlag, Berlin, 1989.

Prager, J. M., *Extracting and Labeling Boundary Segments in Natural Scenes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-2**(1):16-27, January 1980.

Poggio, Tomaso, Vincent Torre, and Christof Koch, *Computational Vision and Regularization Theory*, Nature, **317**(26):314-319, September 1985.

Rosenblatt, F., *Principles of Neurodynamics*, Spartan Books, 1962.

Rosenfeld, A. and A.C. Kak, *Digital Picture Processing*, New York: Academic Press, 1976.

Rumelhart, David E., James L. McClelland, and the PDP research group, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Volume I and II, MIT Press, Cambridge, Massachusetts, 1986.

Torre, Vincent and Tomaso A. Poggio, *On Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-8**(2):147-163, 1986.

Tsotsos, John K., *A "Complexity Level" Analysis of Vision*, IEEE proceedings of International Conference of Computer Vision, 346-355, 1987.

van Essen D.C. and J.H.R. Maunsell, *Hierarchical Organization and Functional Streams in the Visual Cortex*, Trends in Neuroscience, 370-375, September 1983.

Wetherill, G.B., H. Cohen and R.B. Vasudeva, *Sequential Estimation of Quantal Response Curves: A New Method of Estimation*, Biometrika, **53**(3 and 4):439-454, 1966.

Wilson, Hugh R. and James R. Bergen, *A Four Machanism Model for Threshold Spatial Vision*, Vision Research, **19**:19-32, 1979.

Witkin, Andrew P., *Scale-space Filtering*, in Proceedings of International Joint Confonference on Artificial Intelligence, Karsruhe, West Germany, 1019-1021, 1983.

Yarbus, A.L., *Eye Movements and Vision*, Plenum Press, New York, 1967.

Zucker, Steven W., *The Compufational Connection in Vision: Early Orientation Selection*, Behavior Research Methods, Instruments, and Computers, **18**:608-617, 1986.

Chapter 7

Conclusions

This chapter summarizes the considerations behind the design of my algorithm, its implementation, and evaluation. It concludes with a discussion about possible future research directions.

7.1 Summary of the Algorithm

My algorithm is based on the assumption that an effective vision algorithm should be based on a connectionist architecture because of the strict speed constraint of visual tasks. The effectiveness of the connectionist approach is possible because our visual world is highly regular — the shape of many objects have common properties. Therefore, if the world constraints, viewed as segmentation cues, are built into the detection hardware, reasonable segmentation can be obtained in real-time. In this sense the definition of connections is really the process of including knowledge about our visual world into the physical architecture of the visual system.

The special-purpose circuits provide high processing speed but have their constraints too: they can implement only local operations; their functions are rigid and their capabilities limited. Hence side effects may occur. For example, the need of special-purpose circuitry for fast information reduction in early vision causes optical illusions. One way to cope with this deficiency is to have repetitions of the circuitry with each unit covering a different range of spatial and temporal frequencies. This notion is the basis underlying multiple segmentation mechanisms and multiple edge filters within each segmentation mechanism.

My edge-based segmentation algorithm consists of three stages: edge filtering, corner detection, and a spatial coherence check. All the later stages depend on edge filtering. After studying several functions for edge filter kernels, the multivariate Gaussian filters