A Connectionist Algorithm for Corner Detection

TR88-039

August 1988

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 Corner Detection

Cheng-Hong Hsieh

August 3, 1988

This paper[1] describes an algorithm detecting corners in two dimensional grey-scale images. In the following the paper first introduces why a connectionist approach is applied. Then it describes the design considerations of the edge and corner detectors. Next the paper summarizes the simulation results. A discussion on implementation issues and a short conclusion then follow.

1 Introduction

Substantial psychophysical and physiological evidence [Barlow 83, Treisman 86, etc.] indicates that nature has evolved a pre-attentive, highly-parallel, and data-driven early visual system. Furthermore, this early visual system stabilizes in the first several years of an animal's life. Various visual illusions, for example, subjective contour, simultaneous contrast, ..., etc, further demonstrate this. My research is to investigate this stable, highly effective pre-process. It is hoped that the knowledge will contribute to the design of a real time vision machine.

The bounding contours of objects in a two dimensional image are known to be important for recognition. Attneave [54] further emphasized that corners, or discontinuous curvature changes along a contour, are important. Many approaches have been proposed for corner detection. [Asada and Brady 86, Baugher and Rosenfeld 87, Davis 77, etc]. But most methods assume that a given contour exists and the algorithm just decides 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.

[1]The support of NIH Grant R01 CA 39059 is gratefully acknowledged

Moreover, the detection must be fast so that the robot or the organism can interact with its environment effectively. Psychophysical data shows that recognition takes less than 200 milliseconds. So corner detection, supposedly a prerequisite for recognition, must be completed at least within hundreds of milliseconds. What architecture and algorithm would enable the corner detection be performed in real-time?

1. Why a connectionist approach ?

The von Neumann machine is not the answer. The sequential computer is too slow to process the time sequence of two dimensional images. A simple calculation suffices to illustrate this point. Assuming that ten frames of visual input are processed per second, each frame is of one thousand by one thousand pixels. To parse and classify all these pixels requires, say, on average a hundred instructions per pixel. So it takes a dedicated computer of one BIPS to achieve the required performance. This is not likely, or at least not cost-effective, based on the 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 of feeding instructions and data to each processor requires the identification of simultaneously executable portions of a given algorithm. This task of parallelism detection is well known to be difficult. Together with the overhead of inter-processor communication, the multi-processing scheme is not likely to provide the required performance.

So a special-purpose mechanism tuned to process two-dimensional images is needed. Based on the knowledge in the biological visual system, a neural network, or a connectionist approach, seems to be the answer. Contrary to the conventional computer, the connectionist approach investigates how a gross number of local processing units, when properly connected, can perform globally useful functions.

2. Several constraints of the connectionist approach

The most prominent property of a connectionist approach is local wiring, which, as Hubel and Wiesel [77] pointed out, is one of the common characteristics of the neural system. The local wiring distributes the sensory information so the input image is processed in parallel. The local wiring also allows the flexibility of changing the local structure, thus enabling the system to adapt and learn. Lastly the locality assures reliability.

Adaptivity and parallelism have cost 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, there exists 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 can not 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 retina [Phillips 84, Van Essen 83] may serve as a demonstration of this point.

Another problem of the connectionist approach is the costly data representation. Since a neuron can only represent the amount of a specific signal, it takes numerous neurons to represent a quantity. For example, if edge information is to be calculated by a connectionist algorithm, then a neuron is needed for every orientation at every location.

2 The algorithm

Our visual world is highly regular. The imaging condition is consistent through years of evolution; the shape of many objects have common properties. Therefore, when proper constraints are *built-in* in the connection patterns of a neural network, reasonable feature extraction can be achieved in real time. The incorporation of knowledge in the detection mechanism reflects the viewpoint of Gibson's ecological optics. The difference is that nowadays the connectionist approach is within the grasp of computer simulation. This section describes an algorithm for corner detection following this line of thinking.

Edge detectors in this algorithm are defined as filters which combine a smoothing operator with a differentiation operator. As Marr [80] and many others pointed out, a problem of the feature detection approach is that a significant response of such an edge filter does not necessarily indicate the presence of an edge. To cope with this problem, it is assumed that the appearance of an edge can be better justified by the combination of these filter outputs at the same sampling point, while the appearance of a corner can be decided by a neighborhood operation of these edge filter outputs. In some sense, the edge detection is a first-order statistic and corner detection, the second. My simulation results verify that this idea is effective in finding edges and corners.

This section also describes a simple operation, called *artifact cancellation*. It is the counter-interaction of the outputs of edge filters with perpendicular orientations at each sampling location. The essence is that the edge filter outputs contain the artifact due to discrete sampling and finite approximation. The counter-interaction between edge filters of perpendicular orientations effectively discounts this artifact.

2.1 Edge filtering

Given an image, where is the information which best indicates the bounding contours of objects? From the information theoretic sense, the place where image changes most contains the most information. Koenderink's [87] derivative of Gaussian (GD) model provides a mathematical background for detecting these changes. The *n-jet* - the convolution of nth-order derivatives of Gaussian with the image - not only describes the early visual process elegantly, but also provides an efficient computation scheme. The question is what of those components in *n-jet* are *semantically meaningful*?

Torre and Poggio [85, 86] showed that edge detection is an ill-posed problem because the numerical differentiation in the process causes the solution not to depend on the data continuously. To make the problem regular, the physical constraint of smoothness, i.e., that a real edge must have spatial coherence with its neighbors, needs to be included in the edge computation. The result is that the differentiation must couple with a smoothing filter.

For smoothing, among the several possible choices, the Gaussian, giving the minimal uncertainty and being computationally efficient [Koenderink 84 ,Poggio 86, Asada and Brady 86], seems to be the best choice.

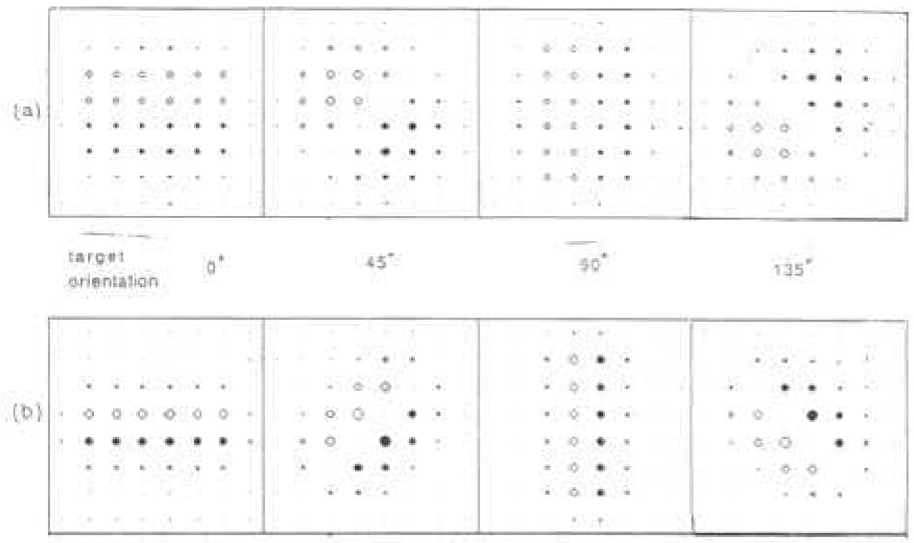


Figure 1: (a) The kernels of an isotropic edge filter with $\sigma = 1.5$ (b) an anisotropic kernel with a different standard deviation in the filter's target orientation and the direction perpendicular to the target orientation . Let the σ and $\sigma_\perp$ stand for these standard deviations. For the current implementation, $\sigma = 1.5$, and $\sigma_\perp = 0.75$. In the figure, the area enclosed by the circles and squares is proportional to the weight of the filter.

For differentiation, Marr and Hildreth's [80] zero-crossing of Laplacian of Gaussian ($\nabla^2 G$) is widely applied in the field of computer vision. The scheme always gives a closed contour, is computationally efficient, and resembles the familiar shape of on-center, off-surround receptive field. However, this approach has its shortcomings too. First, the isotropy of the filter also causes spatial inaccuracy, especially for the sharp corners [Berzins 84]. Next, the second-order differentiation in $\nabla^2 G$, compared with the edge detectors using only first-order derivatives, further amplifies the noise. Lastly, from the viewpoint of connectionist approach, since no orientation is specifically represented, it is very difficult, if not impossible, for this scheme to explain the further usage of edge information. For example, how can subjective contours, which the author conjectures to be an evidence for the preattentive segmentation process, be generated? In summary, the zero-crossings of $\nabla^2 G$ may provide a preliminary information on object boundaries, but, as Torre and Poggio [86] commented, it may be insufficient to account for the segmentation process in early vision.

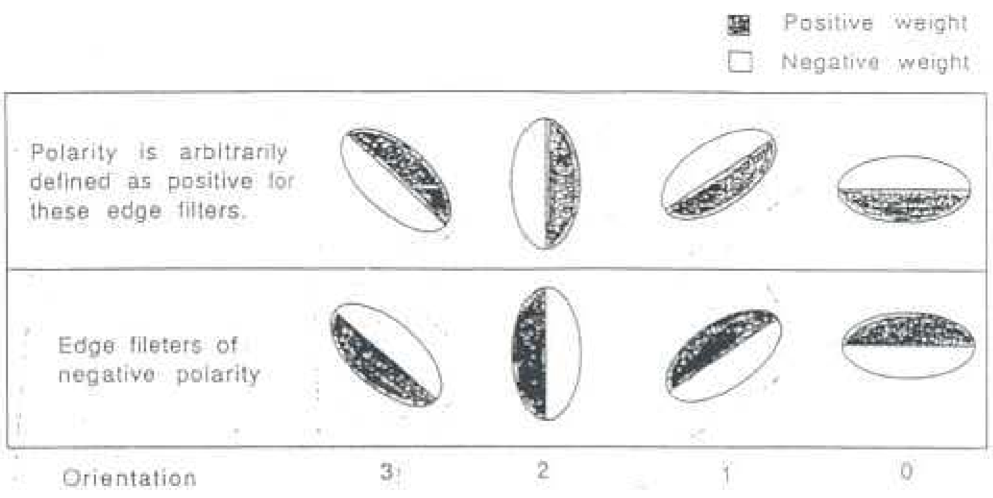

Positive weight
Negative weight
Polarity is arbitrarily defined as positive for these edge filters.
Edge fileters of negative polarity
Orientation
3
2
1
0

Figure 2: The definition of the polarity of edge filters. Note that the filters are arranged from right to left to reflect the usual representation of counter-clockwise rotating angles. Edge filters sensitive to four orientations are applied in this implementation and are labeled by integers from 0 to 3.

The next choice is the first directional derivative of the Gaussian. Though in R^2 derivatives at all directions can be calculated based on two of them, the connectionist scheme needs a neuron to calculate each of the orientations because of the representation constraint. The shape of the kernels of these filters is depicted in figure 1a. The performance of the filter is not satisfactory because, for this algorithm, the edge filters are to detect the real object boundary, while the above-mentioned filters are equivalent to matched filters tuned to detect the edges blurred to a certain degree.

Therefore, the algorithm uses a Gaussian with constant standard deviation in the direction perpendicular to the target orientation (denoted by $\sigma_\perp$) as the edge filter kernel. Figure 1b depicts the shape of these filter kernels, which resembles the receptive fields of Hubel and Wiesel's [77] simple cells. At each sampling point, the algorithm applies four edge filters with each of them sensitive to a different orientation. For each orientation, there are two possible directions of contrast, called polarities, which are illustrated in figure 2.

The remaining question is which $\sigma_\perp$ to use. The result of a preliminary study shows that the sensitivity of the filter with a certain $\sigma_\perp$ is not much affected by the different blurring levels in the input image. This justifies the use of the anisotropic edge filters. How the $\sigma_\perp$ is selected is discussed with other implementation issues.

In summary, let $E(x,y;\theta)$ be the edge filter output of orientation θ at location (x,y),

$$E(x,y;\theta) = K(x,y;\theta) * I(x,y).$$

where, $I(x,y)$ is the input image, and $K(x,y;\theta)$ is the edge filter kernel.

$$K(x,y;\theta) = \{w(x,y) | w(x,y) = k \times \partial G(x,y)/\partial\theta,\ x^2+y^2 \leq \mathit{threshold}\}.$$

where k is a normalization constant such that

$$\sum_{(x,y)\ni w(x,y)<0} k \times w(x,y) = -1 \ \& \sum_{(x,y)\ni w(x,y)>0} k \times w(x,y) = 1.$$

Moreover, Koenderink [87] shows that

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

where the anisotropic Gaussian function, for orientation 0 in figure 2, is

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

2.2 Artifact cancellation

Grossberg and Mingolla [86,87], in designing a connectionist model for their edge-based segmentation scheme, found that the edge filter output gives better results for later processing, if the output of the filter with perpendicular target orientation is subtracted from the edge filter output under consideration. A further examination of this method shows that this scheme is effective to diminish the artifact due to the pixellation and digital sampling. Figure 3 demonstrates this point. Let $E(x,y;\theta)$ and $E(x,y;\theta_\perp)$ be the outputs of the edge filters of perpendicular orientations θ and $\theta_\perp$ at the sampling location (x,y). Since there can not be perpendicular edges at a specific sampling point, the fact that both $E(x,y;\theta)$ and $E(x,y;\theta_\perp)$ are not zero means that there is artifact. A reasonable thing to do is to perform

$$E(x,y;\theta) = \begin{cases} \max(0, |E(x,y;\theta)| - |E(x,y;\theta_\perp)|) & \text{if } E(x,y;\theta) > 0 \\ -\max(0, |E(x,y;\theta)| - |E(x,y;\theta_\perp)|) & \text{if } E(x,y;\theta) < 0 \end{cases}$$

Note that only the output of the filter with target orientation right on edge, as in figure 3a, is not affected. The output strength is adjusted downward for edge filters of all other orientations.

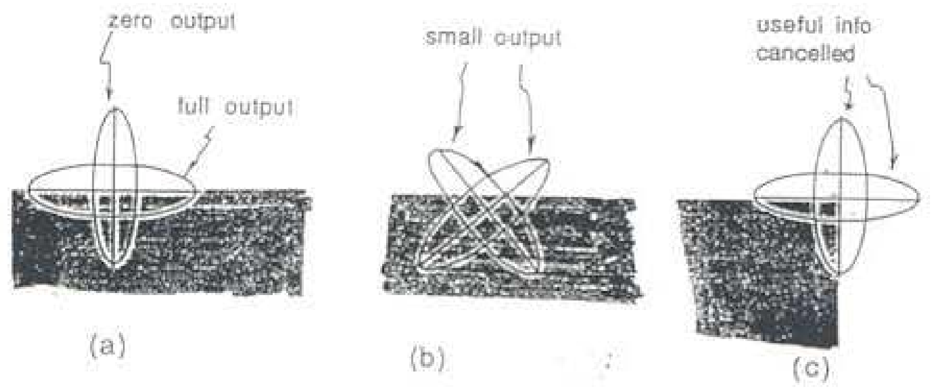


Figure 3: The artifact generated by edge filters. (a) The target orientation is right on edge (b) Both filters with target orientation perpendicular to each other have outputs (c) The situation when the artifact cancellation process may destroy useful information

The only place this operation may destroy useful information is around the right corner as described in figure 3c. However, since the corner detection is devised as second order statistics, i.e., it is based on information at more than one sampling locations, the loss of this information does not affect the detection of a right angle.

2.3 Corner detection

What is a corner? For two-dimensional grey-scale images, a corner can be viewed as the place where two edges meet. A corner can not be detected by template matching as edge filters do for image intensity changes. A simple calculation on required number of neurons shows this: A corner can have various opening angles and various 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 follows. Assuming the resolution of image is 1k by 1k and corners can open to ten different directions and have ten different opening angles. Without counting the polarity change, the required number of corner templates is $10 \times 10 \times 1k \times 1k = 10^8$ which is closed to the total number of neurons in V1 [Wiesel and Hubel, 77].

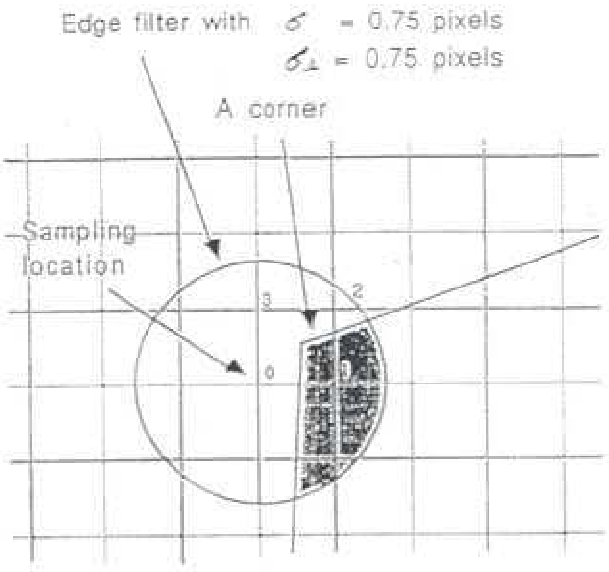

Edge filter with σ = 0.75 pixels
$\sigma_\perp$ = 0.75 pixels
A corner
Sampling location
3
2
0

Figure 4: The edge filtering near a corner on a rectangular sampling grid.

The derivation of the corner detector started from the idea that corner is a local property and thus can be detected from local information. Furthermore, there are evidence that humans detect corners after edges [Blakemore 75, Barlow 83]. A reasonable assumption is that, for a rectangular sampling grid as used in this algorithm, whether a corner exists within a pixel can be decided by the output patterns of the edge filters sampled at the pixel's nearest neighborhood. For example, in a rectangular sampling grid as in figure 4, whether a corner exists within the pixel defined by the four sampling points (indicated by 0 – 3) can be decided by the edge filter outputs at these four sampling points only.

Furthermore, assuming that the corner detection follows second order statistic, then there may be fixed patterns between edge filter outputs sampled at each pair of the four locations. After studying edge filter outputs at the six possible pairs of locations, the finding is that, if an edge resides in a pixel, there is at least a pair edge filter outputs showing a certain polarity pattern. Figure 5 summarizes these patterns which can be further categorized into two types. The first kind is indicated by corner types 0, 2, 4, and 6, while the second kind by corner type 1, 3, 5, and 7. For corner type 0, among the 16 edge filters, the ones indicated by the short line segments in the figure should have significant responses. Moreover, the edge filter of 45° at position 0 and the one of 135° at position 3 should both have negative polarity. This is also true for positions 2 and 3, but the responses are

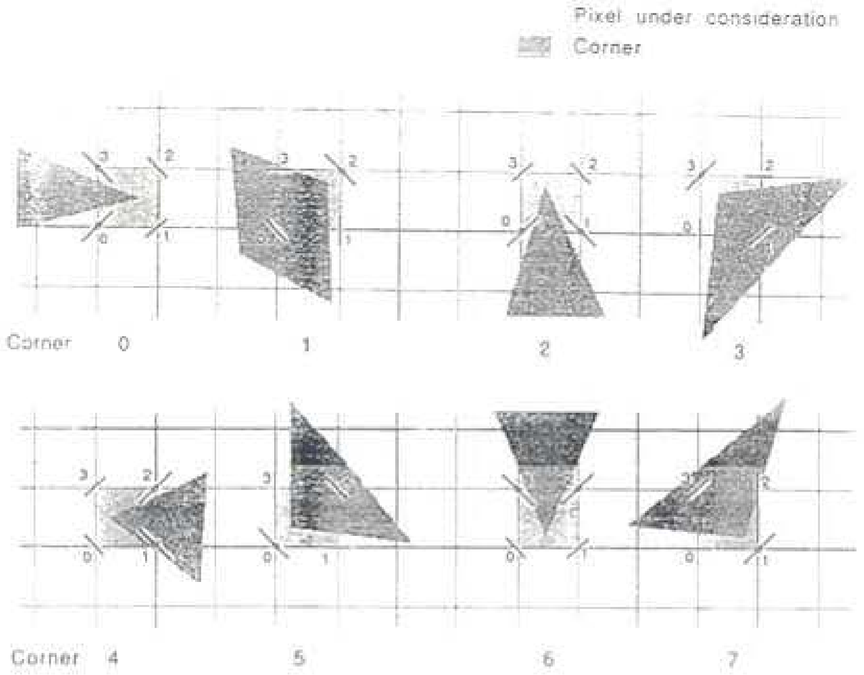


Figure 5: The possible patterns of edge filter outputs near a corner on a rectangular sampling grid. There are eight of them (labeled 0 – 7 in the figure) based mainly on the corner's opening direction.

smaller. For corner type 1, both the pair of edge filters of orientation 135° at location 0 and 2 and the pair of 90° at location 1 and 0° at location 3 have opposite polarities.

The algorithm is tolerant with the definition of these patterns. When I changed the details of the definition, the result does not vary significantly if the definition follows what illustrated in figure 5. An example of the definition for corner type 0 and 1 is described by the following equations.

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

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

where, $C(loc; type)$ denotes the probability of occurrence of a corner (called *corner strength*) at the location surrounded by sampling points $0-3$, *type* specifies the corner type defined in figure 5, $E(loc;\theta)$ denotes the edge filter output of orientation θ at location *loc*, *E_thd* stands for edge threshold, and $f(E1, E2)$ is a simple function giving a pixel's corner strength based on the edge filter outputs. An example of this function is $|\min(|E1|, |E2|)|$.

A program detecting these patterns based on the above definition was implemented. Simulation results show that most of the sampling locations marked by a small circle in figure 5 are satisfactorily detected as corners.

The algorithm is mainly based on the polarity of the edge filter outputs, hence is more stable against noise than the method based on the numeric values of edge filter outputs. Another strength of the algorithm lies in the fact that the algorithm depends only on the local information and is very simple. Not only an efficient implementation on conventional computer architecture is possible, but a construction of a connectionist algorithm is feasible.

Figure 6 shows a possible connectionist implementation of this algorithm. There are four layers. In the input image layer, each circle indicates a receptor which stores a pixel intensity of an image. Each edge filter detects the intensity change at the sampling points in between the pixels of the input image layer. At each sampling point, there are eight edge filters of different target orientations or polarities. The wiring between the edge filter layer and the input image is not shown because of the complication it causes. Each neuron in layer 3 has inputs from edge filters at two sampling locations. An example of the connections between the edge filter layer and the layer 3 (marked with dotted wiring and shaded circle) is illustrated in more detail in figure 6b, which shows that both orientation and polarity contribute to the corner detection. Each neuron in layer 4 has inputs from mainly neurons in layer 3. Its firing 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 4 need to repeat for eight times. Since each connection described above performs only a very simple function, a neural network construction is feasible.

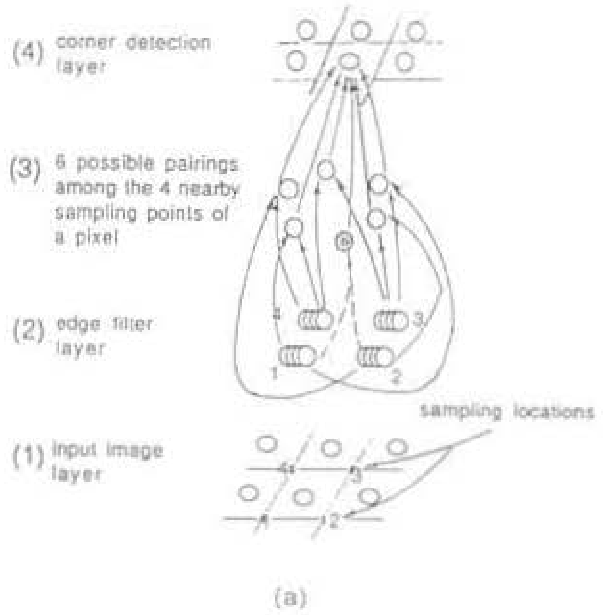
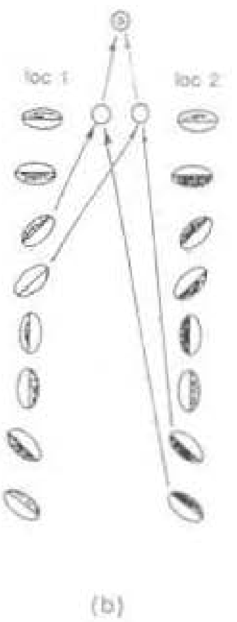

(4) corner detection layer
(3) 6 possible pairings among the 4 nearby sampling points of a pixel
(2) edge filter layer
(1) input image layer
sampling locations
1
2
3
4
(a)


Figure 6: (a) A connectionist implementation of the algorithm. (b) A more detailed description of an example of connections from layer 2 to layer 3.

3 Simulation results

The algorithm is implemented by programs in C on a Sun workstation. Figure 7 shows the output after each stage of the algorithm. Figure 7a is the sampled input of a triangle. 7b shows the result of edge detection. Note that, at each sampling location, edge filter outputs of four different target orientations are shown. 7c is the result after artifact cancellation. 7d demonstrates that the three corners are successfully detected. What follows describes the strengths and limits of the algorithm.

1. The algorithm works under various situations.

Figure 8 shows four corners of various opening angles. They are selected from a collection of simulated results on artificial, ideal corners. All of these corners are located at (8, 8) of a (16, 16) grid. For the corner of 15°, the location detected is shifted because of the inadequacy of the rectangular sampling, which is clearly shown in the sampled test input. For a flat edge, there is no corner detected as expected. There are also corners detected near the boundary of the frame.

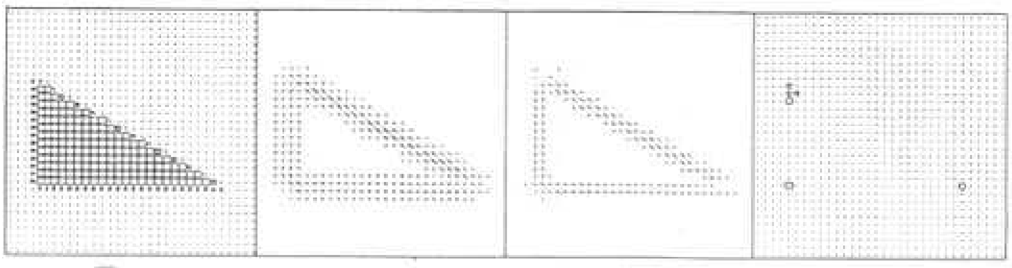


Figure 7: For test pattern of a triangle of 30°, 60° and 90°. (a) the sampled input (b) the result of edge filtering (c) after artifact cancellation (d) the detected corners

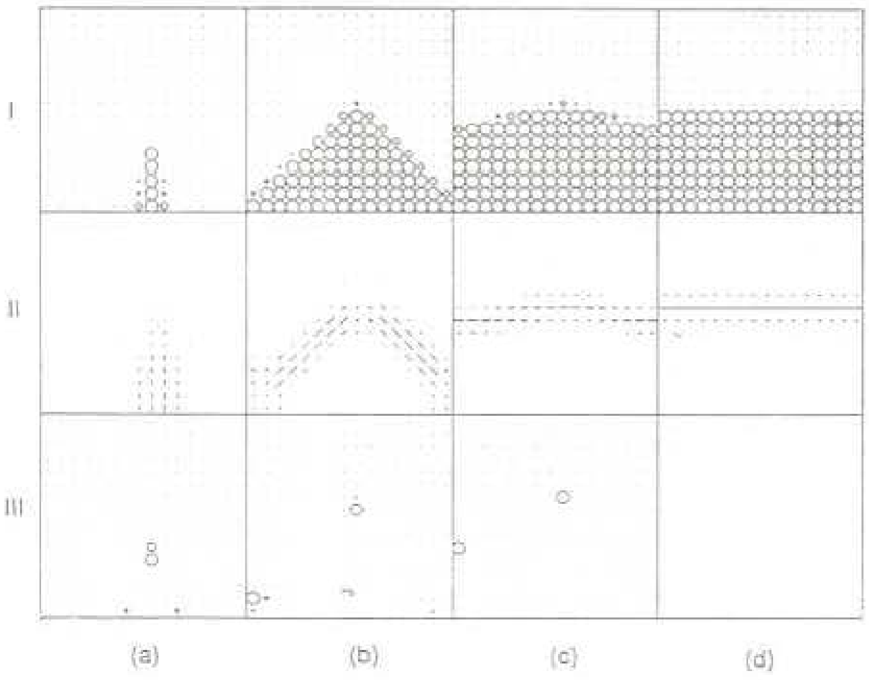


Figure 8: Corners of various opening angles. (a) 15° (b) 90° (c) 150° (d) 180°. Row I shows the input patterns. Row II shows the results after artifact cancellation. Row III is the result of corner detection.

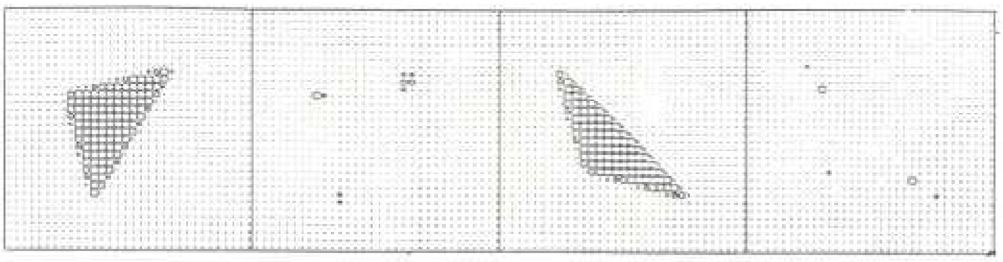


Figure 9: Two triangles of arbitrary orientations. (a) a triangle of 45°, 45°, and 90° (b) a triangle of 30°, 30°, and 120°.

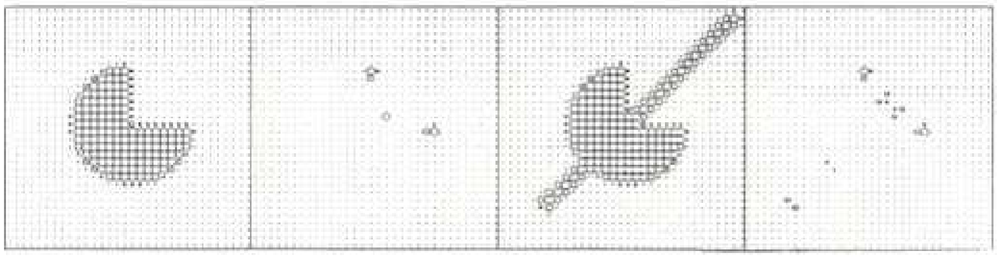


Figure 10: Corners not formed by two linear edges. (a) a pacman (b) a pacman with a line segment.

Figure 9 demonstrates the simulation results of two triangles arbitrarily oriented. 7b and 7d are the simulation results of test patterns 7a and 7c respectively. Note that, for each corner, the algorithm does not give a single location. Instead several pixels near the exact location of corner are indicated. This is mainly due to the digital sampling and can be improved by mutual inhibition.

Figure 10 shows the detection of corners not formed by two linear edges. 8b also shows that the algorithm can detect the corners caused by the overlap of two shapes.

Figure 11 shows the simulation result on a portion of a real scene. Since a natural scene is usually very complicated, causing the simulation result to be difficult to see, this test pattern is intentionally selected by its relative simplicity. The locations of detected corners are compared with the image.

The result is satisfactory.

2. Line ends are detected naturally.

The algorithm detects line ends naturally. If a line segment is too wide, i.e., it becomes a rectangle, then four corners will be shown. Figure 12 shows the simulation result.

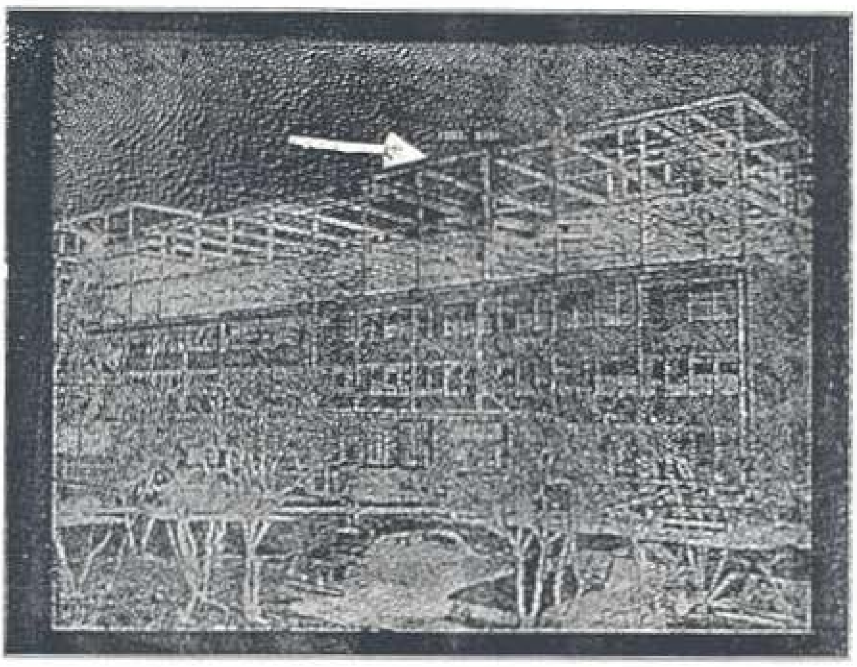
3. The algorithm has its capability limits.

This subsection describes the capability limits of the algorithm. The effect of corners of sharp and nearly flat angles, noise, and blurring are respectively discussed. Possible solutions are discussed in the next section.

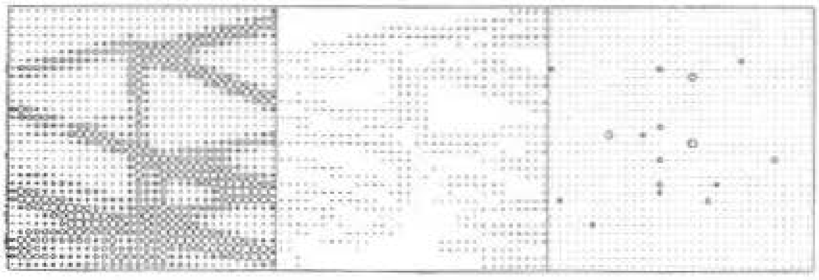
The algorithm does not work as well on sharp and nearly flat corners. Figure 13 shows the result on corners of 15° and 150°. The reason is that edge filters are 45° apart. For corners of flat angles, they can not tell if it is near a corner or is a perfect edge. For corners with sharp angles, the output of edge filters are more seriously affected by the noise and the artifact of pixellation. Since the corner detection is based on the edge filter outputs, it is accordingly affected.

Noise affects the performance of the algorithm. This is expected since a corner, as a property of *2-jet*, is more sensitive to noise than edges. In figure 14, the results are not thresholded. It seems that the performance is heavily influenced for random noise of signal to noise ratio below 1.67.

Figure 15 describes the effect of blurring. The algorithm does not behave as well when the σ of the blurring Gaussian exceeds 3 pixels. Figure 15b and 15c show another problem: blurring smooths the corner, but the number of corners shown increases with the increasing blurring levels. The reason is that, at those situations, besides decreasing the intensity gradients at a real edge, blurring also causes the intensity gradients to spread out. Since the threshold for the edge filters is small (1 in 10) and unchanged through these simulations, the detected edges spread out and the corner detection is affected. When the blurring level increases as in figure 15d, the corners disappear.



(a)



(b) (c) (d)

Figure 11: Corners detected in a real scene. (a) an image of a building, portion in the frame is extracted as test patterns (b) test patterns with area of circle representing the intensity (c) the output after artifact cancellation (d) important corners are correctly detected.

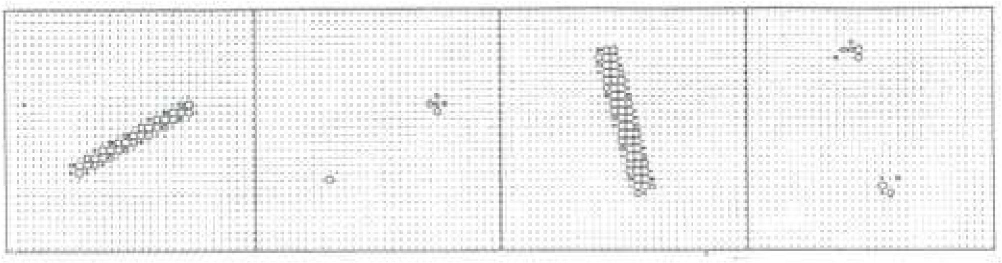


Figure 12: Two examples of line end detection.

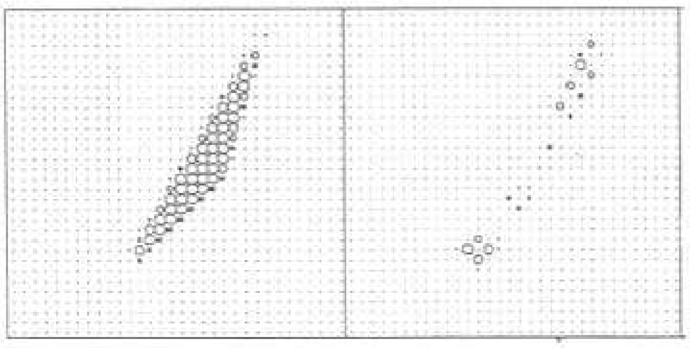


Figure 13: A triangle of 15°, 15°, and 150°.

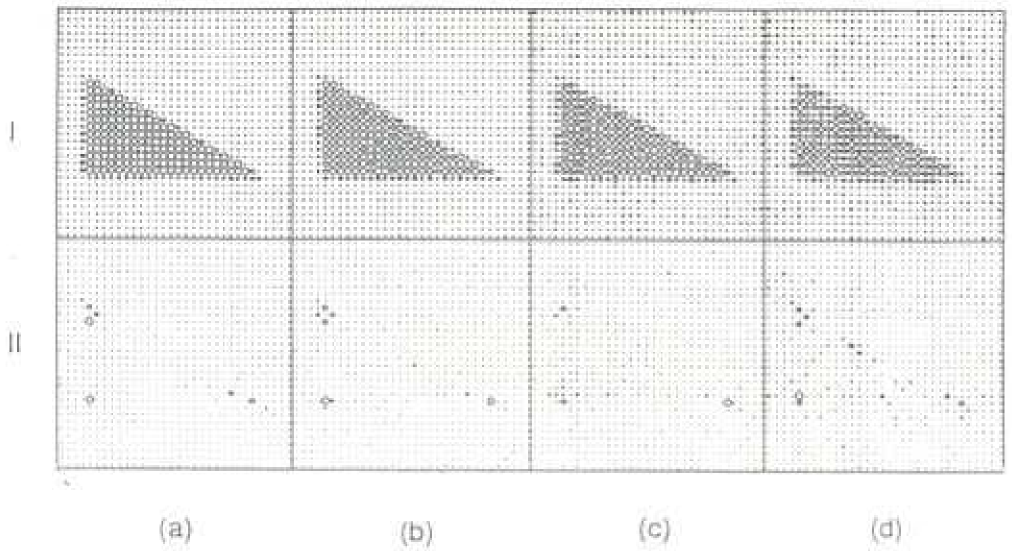


Figure 14: The effect of random noise. The signal to noise ratio is (a) 5 (b) 2.5 (c) 1.67 (d) 1.25.

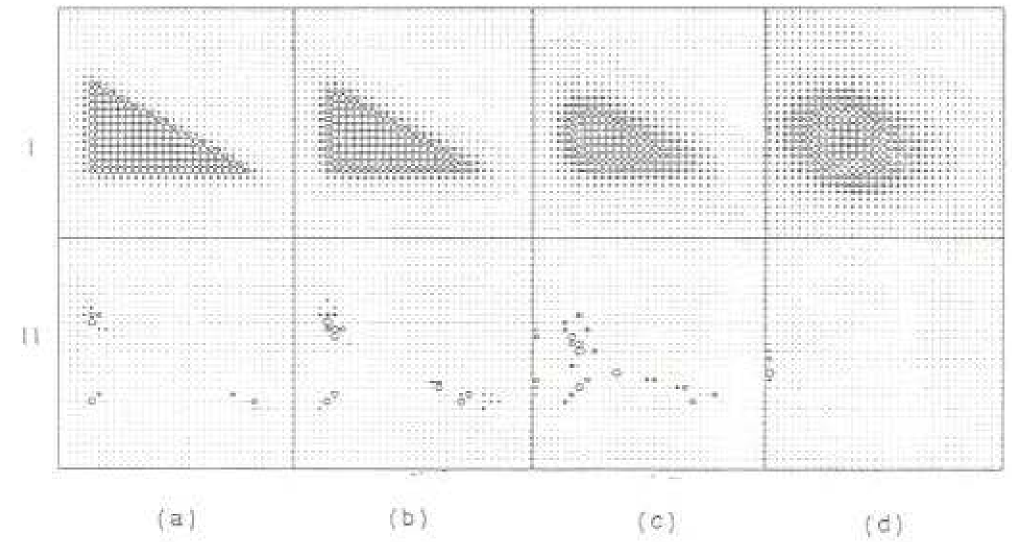


Figure 15: The effect of blurring. A two dimensional Gaussian is used as the blurring function. The σ is (a) 0.75 (b) 1.5 (c) 3.0 (d) 6.0 pixels.

4 Discussion

As Tsotsos commented [87],

> *computer simulation is different from mathematical modeling because the problem of pixellation, sampling, and computational complexity must be considered.*

In this section these problems related to input representation, edge filtering, and corner detection are described. The section ends with a brief discussion on the computational complexity of the algorithm.

The first problem is the representation of the input image. Since a biological visual system performs an imaging process just as a vision machine does, the sampling problem is inevitable. For convenience, the input representation is chosen to be a two dimensional array of real numbers ranging from zero to one. For artificial test patterns, the portion of a pixel covered by a figure is calculated and the pixel intensity is accordingly adjusted.

For edge filters, the first question is what should be the σ and $\sigma_\perp$. The trade-off is that a bigger σ gives a better signal to noise ratio; while a smaller one more accurately indicates the location of the detected feature. A clue comes from the psychophysical data. Bergen and Wilson [79] showed that four scales with σ approximately equal to 3, 6, 12, 23 pixels suffice to explain many psychophysical phenomena. Marr, Poggio, and Hildreth [80] further pointed out that a smaller scale is likely. Based on these considerations, σ and $\sigma_\perp$ are selected to be 1.5 and 0.75 pixels respectively. Another consideration is from the sampling theory. The Nyquist frequency for the first-order differential of Gaussian with σ equal to 1 is about 1.2 cycles per pixel. So the filter selected as above causes aliasing error. However, the corner detection is based mainly on the orientation and polarity of edges at neighboring sampling locations, so it is not sensitive to the small sampling error. The trade-off of spatial accuracy is well worth it. Another set of simulation based on kernels with both σ and $\sigma_\perp$ equal to 0.75 pixels generally shows even better performance.

Another question about the edge filter is that, at each sampling location, how many orientations shall we apply the edge filter? Since sampling is on a rectangular grid, four orientations with each of them having both polarities

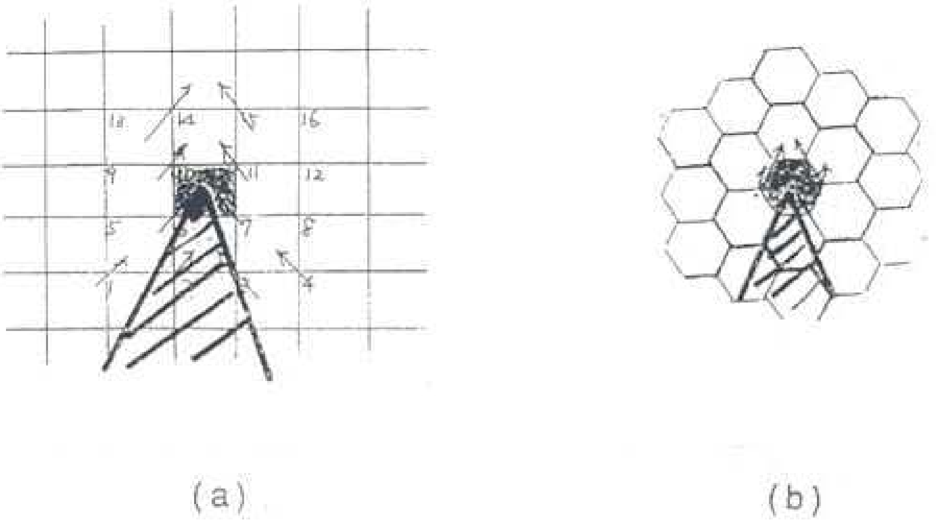


Figure 16: Two possible schemes to improve the algorithm: (a) more sampling points are considered (b) a hexagonal sampling grid.

suit the purpose. I investigated the possibility of applying edge filters on more orientations. It does not increase the algorithm performance by much, but causes the programming and the construction of a connectionist model to be more difficult.

As described in the last section, the proposed corner detector does not work as well for sharp and nearly flat corners. Its performance is also affected by noise and blurring. There are two possible remedies at the cost of throwing in more resources. Figure 16a demonstrates that an example of how this algorithm can be extended to taking into account the neighboring 16 sampling points. Instead of considering only locations 6, 7, 10, 11, all locations marked with an arrow can be used to either increase or decrease the estimation of the probability of the existence of a corner in the shaded area. For example, if the edge filter pair at location 1 and 4 with marked orientation fire substantially and have opposite polarity, then the probability should increase. Another scheme is by using the hexagonal sampling grid as depicted in figure 16b. A version of the program based on this scheme is now under construction. I believe that this scheme will improve the algorithm performance.

Another observation on the performance of corner detection is that it really depends on the edge filters. If the edge filters are not working properly, corner detection will not work either. In this sense, the edge filter can be viewed as a part of the corner detector.

Though the algorithm has its shortcomings, the author does not consider these as a serious drawback. The point is that all visual systems have limited resources - 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 seek for perfect solutions. The economic use of resources is important.

The resources required for this algorithm is briefly analyzed as follows. Note that the estimation is based on the current implementation. Assuming the resolution of the input image is $m \times n$, the number of orientations is k, the number of corner types is c, then we need $m \times n \times k$ edge filters, $(6 + c) \times m \times n$ corner detectors, plus other intermediate connections in the order of $m \times n$. Note that each neuron has only local connections, say, of upper bound l. Then the number of connections in the network is in $O(lmn(6 + k + c))$. If the algorithm is simulated on a von Neumann machine, this figure indicates the order of required computation time. For the current implementation, with image of 128×128, neural connections of 50 on average, the computation load is about 1.5×10^8. With a 10 MIPS machine, a Sun 4 for now, it takes minutes to run. Evidently, with a connectionist implementation or a multi-processing architecture, real-time performance can be expected.

5 Conclusion

This paper describes an edge-based corner and line end detector. It has been shown that the intuitively simple scheme successfully detects corners under various conditions. The essence of the scheme lies on the fact that the detection is based only on the local information. Hence a connectionist algorithm can be built and an efficient implementation on a multi-processing architecture is likely.

The design of the algorithm is based on the knowledge in human visual system, the constraints imposed by our visual world, and the functional analysis on the computational needs of visual tasks. Besides the design of a better computer vision system, it is hoped that the knowledge thus attained will help understanding the working principles of the biological visual system.

There are many problems left, for example, how to cope with noise and blurring in the image? How will the hexagonal sampling grid help detecting corners? Answers to these questions still await further investigation.

Acknowledgement

I thank my advisor Stephen Pizer for his help and guidance; Prof. Stephen Grossberg for enlightening me on the neural network approach to vision; Prof. Jan Koenderink for his direction and encouragement; my colleague John Gauch for his reading my draft and helping me in editing, and my friends Shie Jue Lee and Ming Ouh-young for their helpful discussions.