IMAGE BOUNDARY DETECTION VIA INTENSITY, COLOR AND TEXTURE SEGMENTATION

by

Phyllis R. Chang

Memorandum No. UCB/ERL M93/53

2 July 1993

COLOR SEGMENTATION WHERE INTENSITY SEGMENTATION FAILS

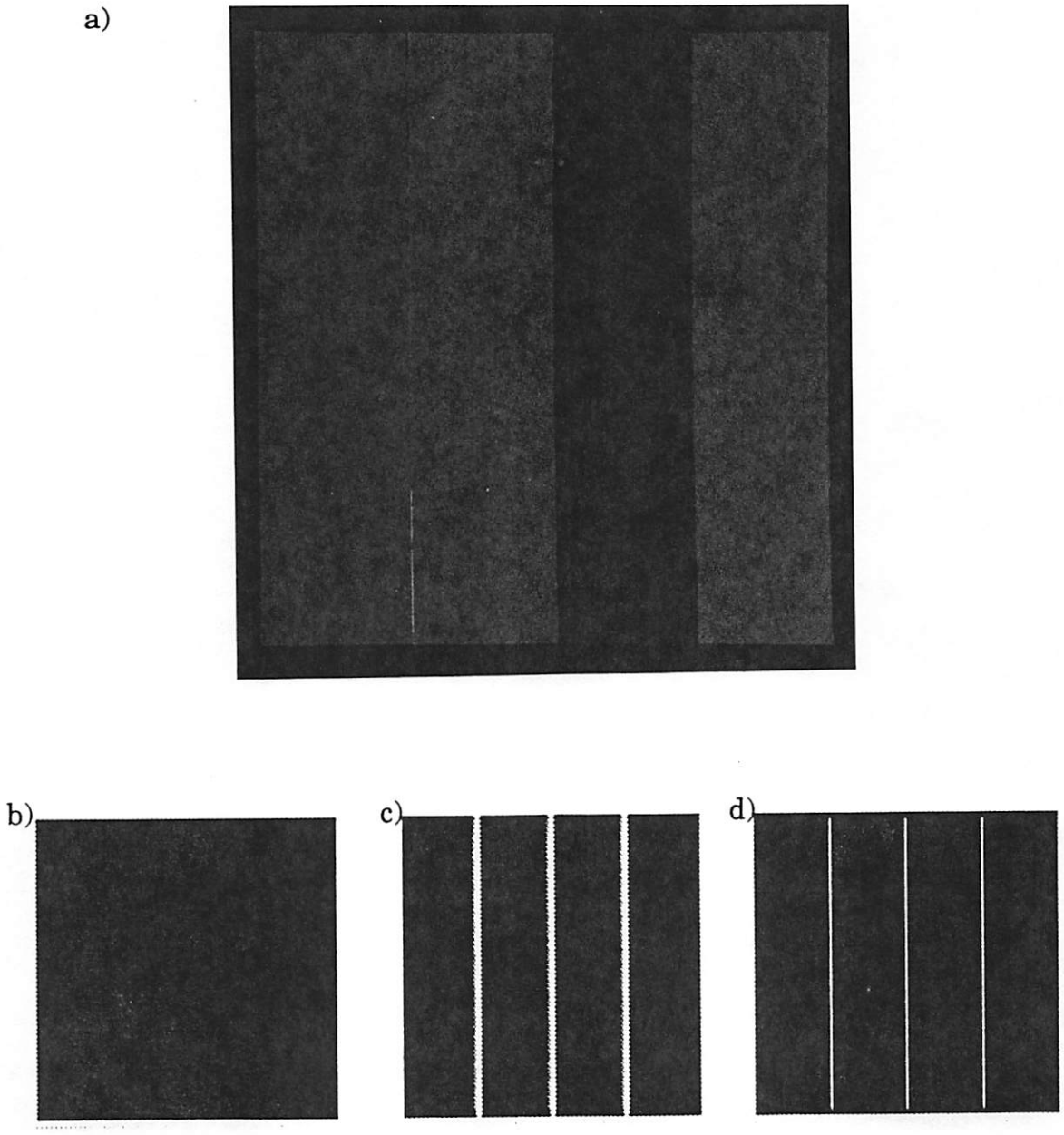


Figure 25: Simplified example of the added usefulness of color edge detection in a boundary detection system.

a) Photograph of the full-color image. The red-green-blue-green stripes are all at equal intensity values.

b) Result of running Canny edge detector on the R+G+B intensity image. No boundaries are detected since there are no intensity changes over the image.

c) The pooled magnitude result of running the Color edge detector using the three bands R, G, and B.

d) The segmentation result of Color edge detector

ANDES: COLOR AND INTENSITY SEGMENTATION OVER SHADOWS

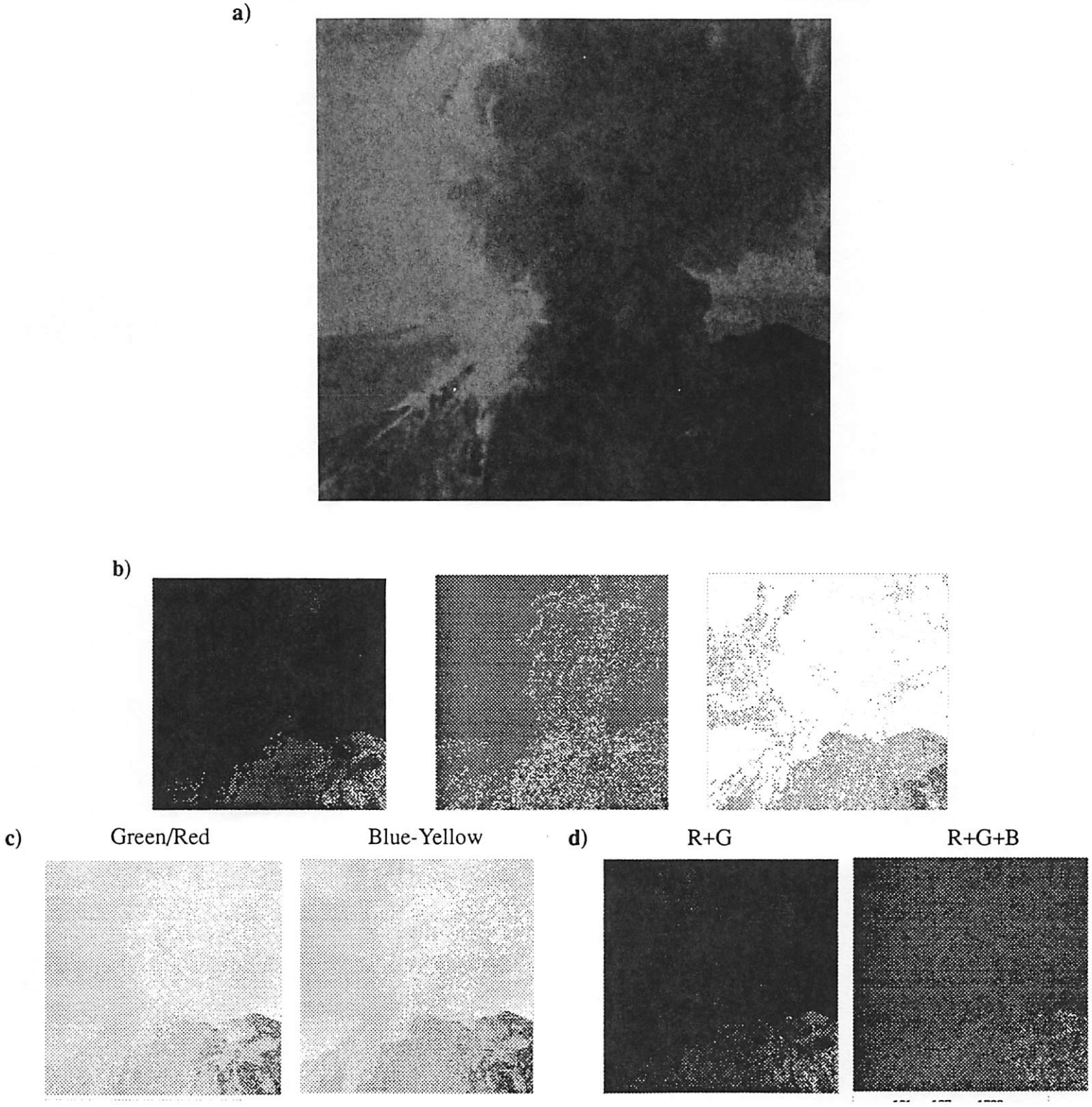
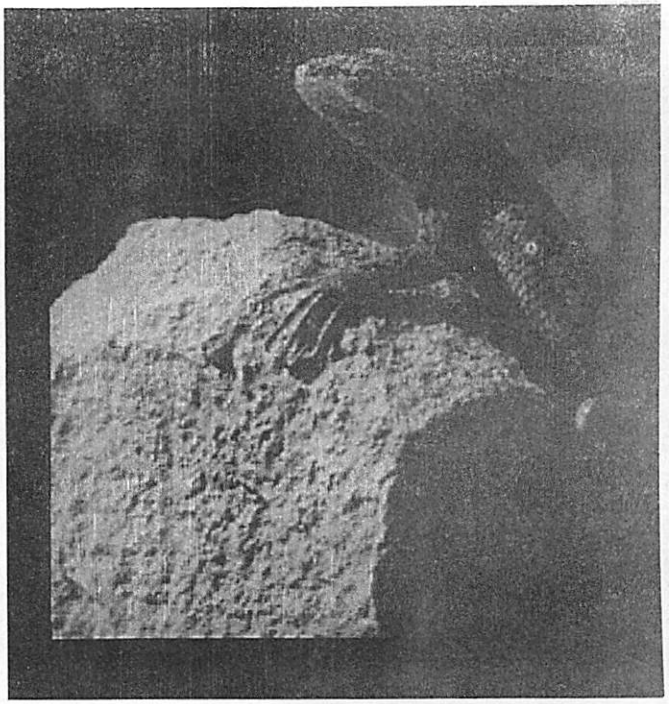


Figure 26:

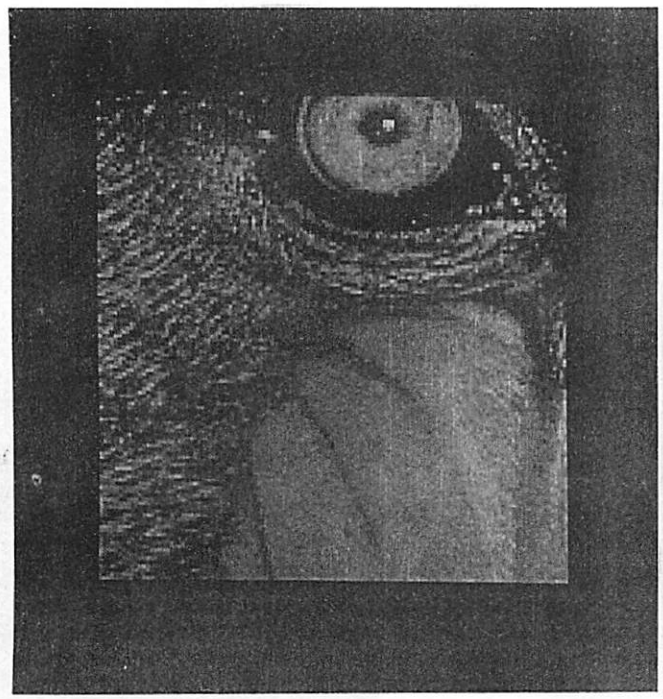
a) Color image of Andes. The white areas are made up of approximately equal amounts of Red, Green, and Blue.
b) The Red, Green, and Blue color bands for the Andes image.
c) The color-opponency bands G-R and B - 1/2(R+G) bands.
d) The intensity images R+G and R+G+B.

a)

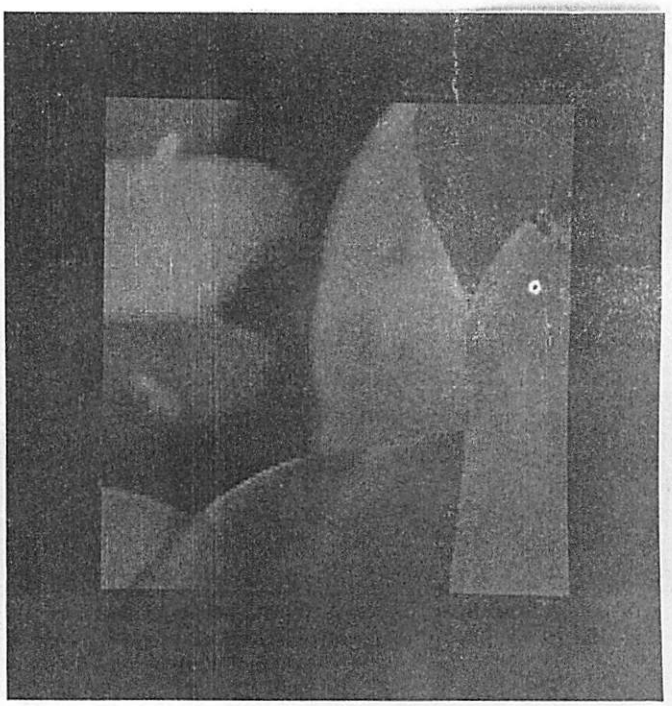
b)



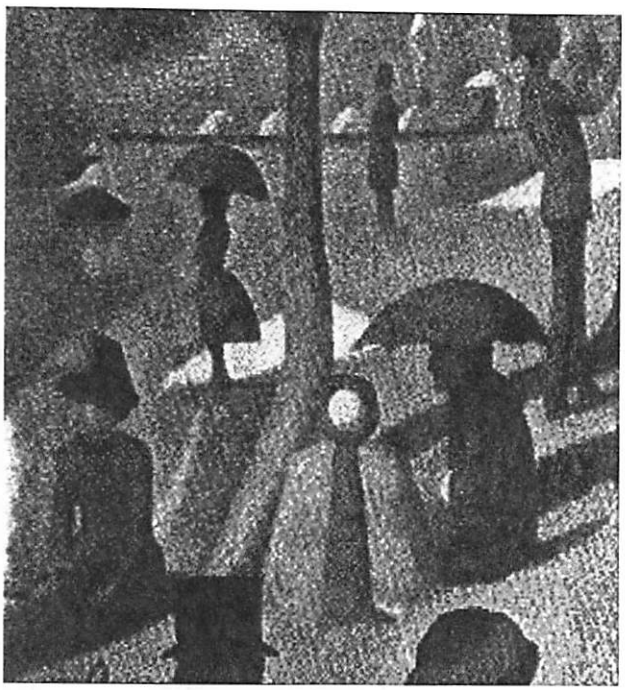
c)



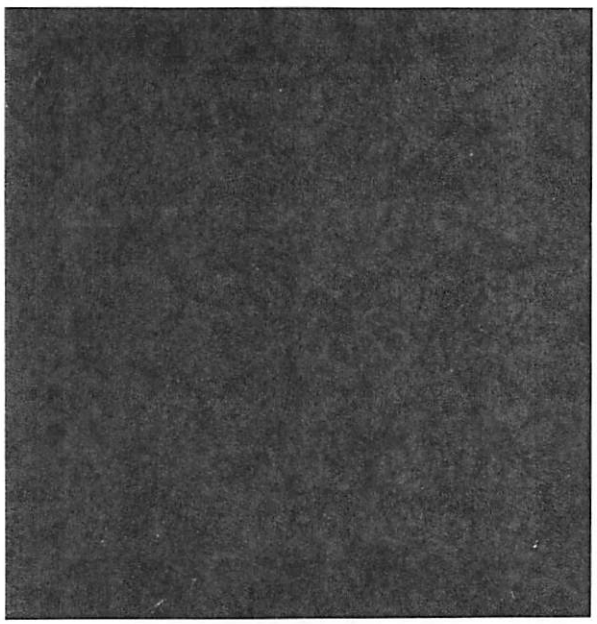
d)



e)



f)



g)

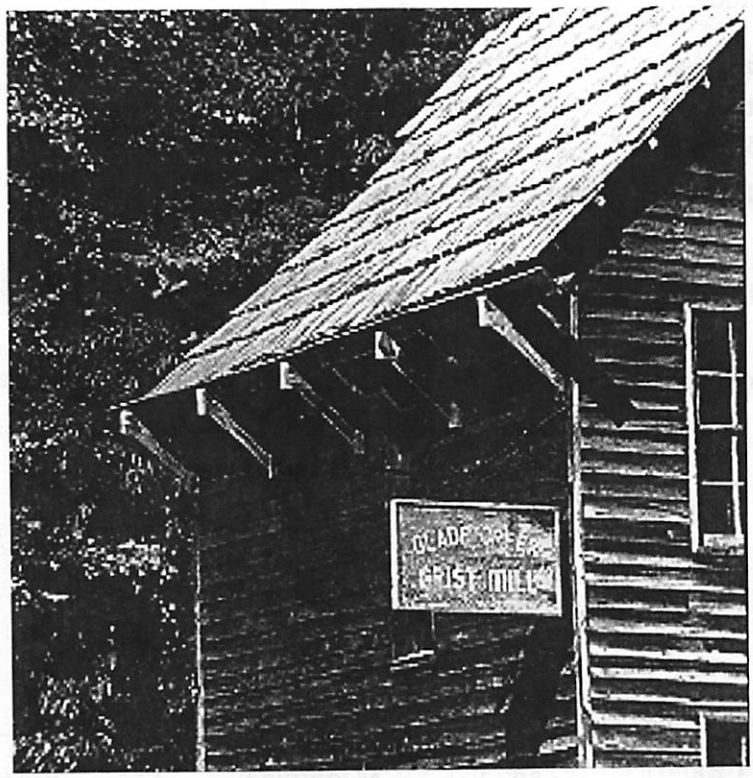


Figure 18: Photographs of color images used in Chapter IV.

Section IV.A. and IV.B. Multichannel approach:

a)Portion of *Lenna*

b)Portion of *Lizard*

c) Portion of *Mandrill*

d) Portion of *Peppers*

Section IV.C. Comparisons Among Intensity, Color, Texture:

e) Portion of *A Day in the Park* by Seurat

f) Yellow foreground texture over blue background, of equal intensities. The three boundary detectors yield very different results.

g) Portion of *Old Mill*. Intensity, Texture, and Color boundary detectors give different results.

color bands are R-G, $\frac{1}{2}$(R+G)-B, R+G+B. The three color bands are analogous to the stage of PIR images in the texture model. The color edge detection steps are:

- for each RGB, smooth the images.
- Calculate x and y derivatives
- Calculate edge magnitude
- local non-maximum suppression to eliminate broad edges.

1. Basic setup: Results for several images

Figure 19 shows the block diagram for this implementation. Initially, the smoothing/derivatives stage (Canny step I) blindly used $\sigma = 8.0$, Figure 21. The color boundary detector did well at

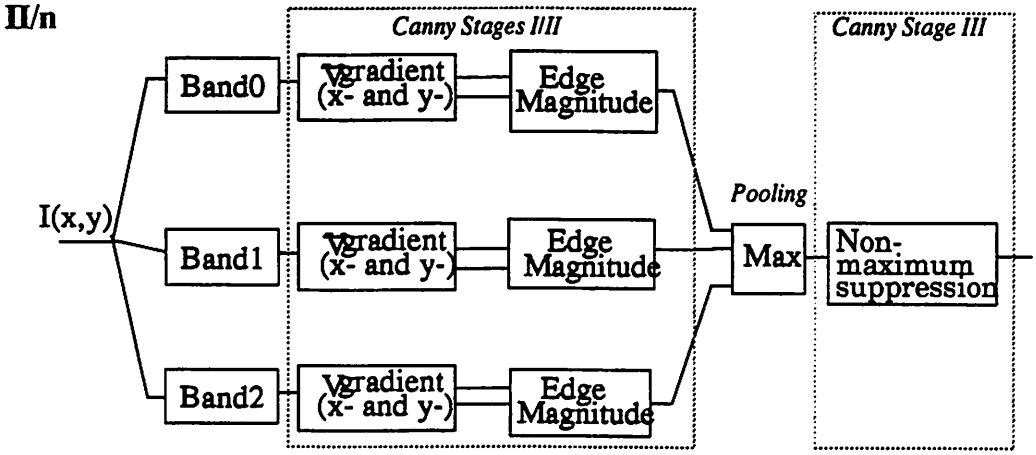


Figure 19: Block diagram for main Color Edge Detection scheme.

defining the color boundaries in the *lenna* image. It missed distinguishing the upper eyelid from the brim of the hat, but this can be explained by the wide smoothing function. It did not give the results expected at first glance for the *lizard* image; in this case, the detector found boundaries between the lizard and the rock, shadows and sunlit areas, but didn't find boundaries between the lizard and the background. This basically shows how well the lizard blends into the background rather than pointing out weaknesses in the algorithm. The *mandrill* image gave boundaries about the eye and the cheek, the two main color areas besides the background (the face). It did miss the extra color rim around the eye and the boundary between the upper nose and face, but these again are due to scale - smoothing blends the two boundaries together. Finally, the

IMAGE BOUNDARY DETECTION VIA INTENSITY, COLOR, AND TEXTURE SEGMENTATION

by

Phyllis R. Chang

Memorandum No. UCB/ERL M93/53

2 July 1993

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Image Boundary Detection via Intensity, Color, and Texture Segmentation

Phyllis R. Chang

Image Boundary Detection via Intensity, Color, and Texture Segmentation

Phyllis R. Chang

Master's Report

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720

Abstract

This report investigates the development of a generalized boundary detector for two-dimensional images. Three types of boundary detection (intensity, color, and texture) together give a complete segmentation which does not depend on the type of input image. The Malik-Perona *texture segmentation* algorithm, which takes a multi-scale, multi-channel filtering approach, forms the initial basis of the texture implementation for this report. Good segmentation results are obtained using this approach. Several variations on the algorithm and their effects on the segmentation are presented, with good results for many variations. The *color segmentation* algorithm follows essentially the same format as the last several steps of the texture algorithm, with the three color bands of an RGB image serving as the individual channels. The results for this method are quite good and often work on images for which intensity differences by themselves are not useful for segmentation. Variations are applied to the basic color algorithm to show that they maintain the effectiveness of the boundary detection. The third type of segmentation finds boundaries based upon intensity differences; the basic *Canny Edge Detector* is used for this purpose on several images. In fact, the Canny Edge Detector is a common denominator among the three boundary detection components, as it serves in a modified form as a back-end edge operator to the texture and color algorithms. Each of the segmentation processes reduces to the detection of edges in an intensity image after preprocessing stages.

I. INTRODUCTION

A. Boundary Detection

Computer vision techniques often seek to accomplish tasks modeling those performed by the human visual system; among these tasks is the perception of boundaries in natural and artificial images. Boundaries provide much of the information needed for humans to recognize and be able to interact with objects. Witness the ability of humans to recognize objects by their outlines alone. The human visual system may effortlessly, or preattentively, discriminate boundaries between different ***textures*** (e.g., a bed of rocks next to a field of grass); it may see a boundary between ***color*** regions; it may infer boundary edges based on ***intensity*** changes in a gray-scale image; or it may find differences due to ***stereo*** vision and optical flow from ***motion*** changes. This paper focuses on the first three of these five visual cues from which boundaries can be

detected. The function of a boundary detector is simply to find perceived edges. The combination, or fusion, of the outputs of various types of boundary detectors can give more complete information over the use of a single boundary detector. A photograph taken of a nighttime scene with a visible-light camera, for instance, and a photograph of the same scene taken with an infrared camera will each differ in informational content; the first might find street lights and the moon, the latter might "see" a car's engine, or an animal. The fusion of the photographic images could give more complete information about the scene than either one alone [1]. More interestingly, however, any single one of the five visual cues alone is often sufficient for gathering information and detecting boundaries without aid from another cue.

B. Applications

Texture analysis in general finds many uses in quality control applications. For example, defects often show up as textural aberrations in textiles, paper, and electronic components. Wood surfaces often have blemishes which are undesirable to the end-user; these can often be detected with vision processes [2]. Recent research [3] utilizes texture segmentation in a quality control setting for automotive finishes. In particular, the paint finish on a car must typically be uniform and aluminum particles which are added to the paint can be used to judge textural quality. While many defects occur as variations within a single texture, such as a worn-out area of carpeting, they can be redefined as a second texture within the first, and segmentation methods can then be applied.

II. INTENSITY EDGE DETECTION

Edges in two-dimensional images can be described as curves across which exist sharp changes in image intensity. This can be a sudden change in intensity occurring on a small scale, or a gradual change in intensity, occurring on a larger scale. The process by which edges are detected is typically to smooth and then to differentiate the image. The smoothing serves to avoid the amplification of noise caused by differentiation; the differentiation amplifies large intensity changes and attenuates small changes. Boundary detection problems can be reduced to the problem of intensity edge detection. By contrast, edge detection has been used in the past as a *front* end to vision models, such as for stereopsis, in order to lessen the amount of information to be processed without removing the relevant information. In the implementations of this report, images are preprocessed to create intensity images, followed by edge detection as a *back* end.

The Canny edge operator [4] lends itself to segmentation of texture, color, or intensity images. Generally, the operator is used to detect boundaries based upon intensity differences among regions of an image. In texture segmentation, the idea behind the use of the Canny edge operator is to pre-process the texture image to the point where *regions* of like texture become sufficiently differentiable in pixel value (intensity) from one another so that a "border" between regions can be found. Obviously, without preprocessing, the canny operator applied to a texture image would yield outlines of the individual texture elements, or texels, rather than a segmentation between regions. The preprocessing as performed in this paper takes a multiscale filtering approach, where the image is convolved with a bank of multiple filters, processed independently and in parallel, and then combined at some point during the Canny stages. It is the approach taken by Fogel and Sagi [5], Turner [6], and Malik-Perona [7,8], for example. This has some foundation in physiological research in early stages of vision which suggests that the human visual system processes information via multiple channels; the filters used in the implementation model several types of visual cells. In color segmentation, the image is separated into its three color bands for processing; in this way, for instance, a boundary will be found between a region of "high" blue value adjacent to a region of "low" blue value.

The Canny operator on an image I(x,y) is composed of three general steps:

I] Calculate directional derivatives in the x- and y- directions on an image I(x,y) which has been smoothed (described as gradients with large standard deviation σ').

The smoothing is necessary to avoid the large spiky responses that occur when differentiating a "noisy" signal. Equivalently, we can convolve I(x,y) with gaussian first-derivatives since the smoothing and differentiating steps can be grouped.

II] Calculate edge magnitude via a peak-finding algorithm. Mark the maximum points which are connected.

III] Eliminate non-edge points through non-maxima suppression.

The Canny edge detector is used as the intensity edge detector. The texture boundary and color boundary detectors require multiple channels (as in the three color bands) and hence the Canny edge operator might be applied multiple times, once to each channel independently of the others; or, the channels may be combined into a single image according to a chosen norm prior to applying the edge operator. The pooling process can occur at many stages of the algorithm. We label the possibilities as follows [9]:

0/n: Combine the channels using an *n*-norm criteria prior to applying all steps of the canny operator to the resulting single image.

I/n: Apply step I of the Canny Operator to each independent channel, then combine using an n-norm, and apply steps II and III to the resulting single image.

II/n: Apply steps I and II of the Canny Operator to each independent channel, then combine using the n-norm, and then apply step III.

III/n: Apply all stages of the Canny Operator to each independent channel, then combine the boundaries detected using the n-norm.

Here, n refers to the norm which is applied pixel-wise whenever two or more images are combined:

1-norm, sum of absolute values; for instance:

$$|r(x,y)| + |g(x,y)| + |b(x,y)| \tag{1}$$

2-norm, Euclidean norm; for instance:

$$\sqrt{r^2(x,y) + g^2(x,y) + b^2(x,y)} \tag{2}$$

∞-norm, max of absolute values; for instance:

$$max(|r(x,y)| + |g(x,y)| + |b(x,y)|) \tag{3}$$

The Perona-Malik paper utilizes II/∞ in its algorithm. This also works quite well for the color edge detector.

III. TEXTURE BOUNDARY DETECTOR

Texture segmentation is generally considered the most difficult task of the three boundary detection types because there is no single definition for texture or how different textures must be in order to be segmented. The criteria often becomes the ability of humans to preattentively (with no effort) segment textures, or is entirely dependent on the application at hand and not subject to generic guidelines.

A. Texture

What constitutes a texture *region* and what differentiates one region from another? Texture does not have one standard definition. Essentially, what may look like a sharp variation in intensity at one scale may recur with some spatial organization such that, at a larger scale, these variations are perceived as a unified texture region. The "sharp variation in intensity" might then be called the texture element, or texel. Various characteristics may distinguish one

texture from another: proximity and spacing of the texture elements, polarity, intensity or brightness, orientation, phase or discontinuities, etc. The criteria on which to base the quality of a segmentation can be task-dependent. A vision system designed to inspect the quality of timber will look for different texture characteristics than one designed to inspect metallic finishes. Another valid answer as to whether one texture region differs from another is to examine empirical results obtained from psychophysical tests on human perception of boundaries between texture regions. Some textures are more difficult to distinguish from other textures. Various theories about differences in texton (line segment) crossings (for instance, an "L" and a "T" are both composed of two lines which cross at different locations), frequency content, mean value, and texel density have been put forth to explain the differences in texture discriminability (e.g., Julesz[10], Beck [11]). The goal in texture segmentation, then, is to at least match the performance of humans in delineating between texture regions. Research has been done in texture classification, but for texture segmentation the properties of the individual textures are not as useful as the differences between regions. Criteria for the goodness of a segmentation include: the smoothness of curves and connections, the lack of spurious edges, no missed edges, and good location of found boundaries.

B. Previous work - multiscale filtering approach

Many useful results in vision research indicate that the multiscale, multichannel approach gives an excellent model of visual perception. In this approach, "channel" refers to the processing results associated with the convolution of the image with a filter of a given type, scale, and orientation. It is called "multichannel" whenever the image is filtered with multiple filters and the individual results processed in parallel. We discuss an interesting model which has been presented by [5] using Gabor filters, which are sinusoids within a gaussian envelope. Gabor filter responses to different types of texture and image variations are highlighted in [6]. In particular, section C details the construction of the model based on the paper by Malik-Perona.

1. Fogel and Sagi

Fogel and Sagi [5] present a model for texture segmentation based upon Gabor filters. Gabor filters, described in 1946 by Gabor and extended to two-dimensions in 1980 by Daugman, have the property of optimal joint resolution in the spatial and frequency domain. This makes them good

texture discriminators since texture elements can be classified as differing in spatial proximity, size/frequency, or density. Their first stage is the convolution of the texture image with an even and an odd Gabor filter to discriminate based upon intensity differences. This is followed by smoothing with a gaussian, thresholding to remove noise, and differentiating using a Laplacian (second derivative) of Gaussian step. The paper addresses the multi-frequency nature of the human visual system, but does not truly integrate this into the algorithm. Instead of utilizing a set of parallel channels, the ideal case is to choose initially the Gabor filters with the correct parameters for a given texture image such that a segmentation is immediately identified. Should this fail, they state, change the parameters and run the algorithm again. This makes the algorithm far from automated, although basic segmentation concepts are included in their model. The paper does provide an iterative procedure rather than a parallel procedure. One other weakness in the algorithm is the use of only four orientation in the filters; better resolution could be obtained by adding a few more orientations. The one-at-a-time approach does seem to have justification for images with only two texture regions; as we find in the experiments in this report, the segmentation of two textures is typically due to a single channel. With increased numbers of regions, the segmentation would be improved by combining results of multiple channels.

2. Turner

Turner [6] shows that Gabor filters respond differently to different textures and identifies what textures may be considered "different". The major contribution from this paper is to categorize the various properties of texture regions which make them discriminable. These properties are:

- *Intensity.* The Gabor filter outputs high values across changes in intensity, but outputs nothing over regions of constant intensity.
- *Phase shift.* When a subjective boundary is formed by shifting a texture, the gabor filter gives high output at the boundary.
- *Orientation.* Tripartite field of L's T's and tilted T's. The Gabor filter produces similar responses to the preattentively similar L and T and different for the tilted T.
- *Second-order statistics*: micropattern regularly spaced in random noise. Differences in second order statistics can be detected with Gabor filters.
- *Regularity:* regularly placed region vs. markov-generated region. Gabor filters give higher responses to regularly placed pixels than to random pixels.

C. Malik Perona Implementation:

Malik-Perona combines the concept of multi-frequency filtering as well as the model of texture vision as bar, blob, and ring detectors, descriptions used in Marr's "primal sketch." This section

Texture Image Set

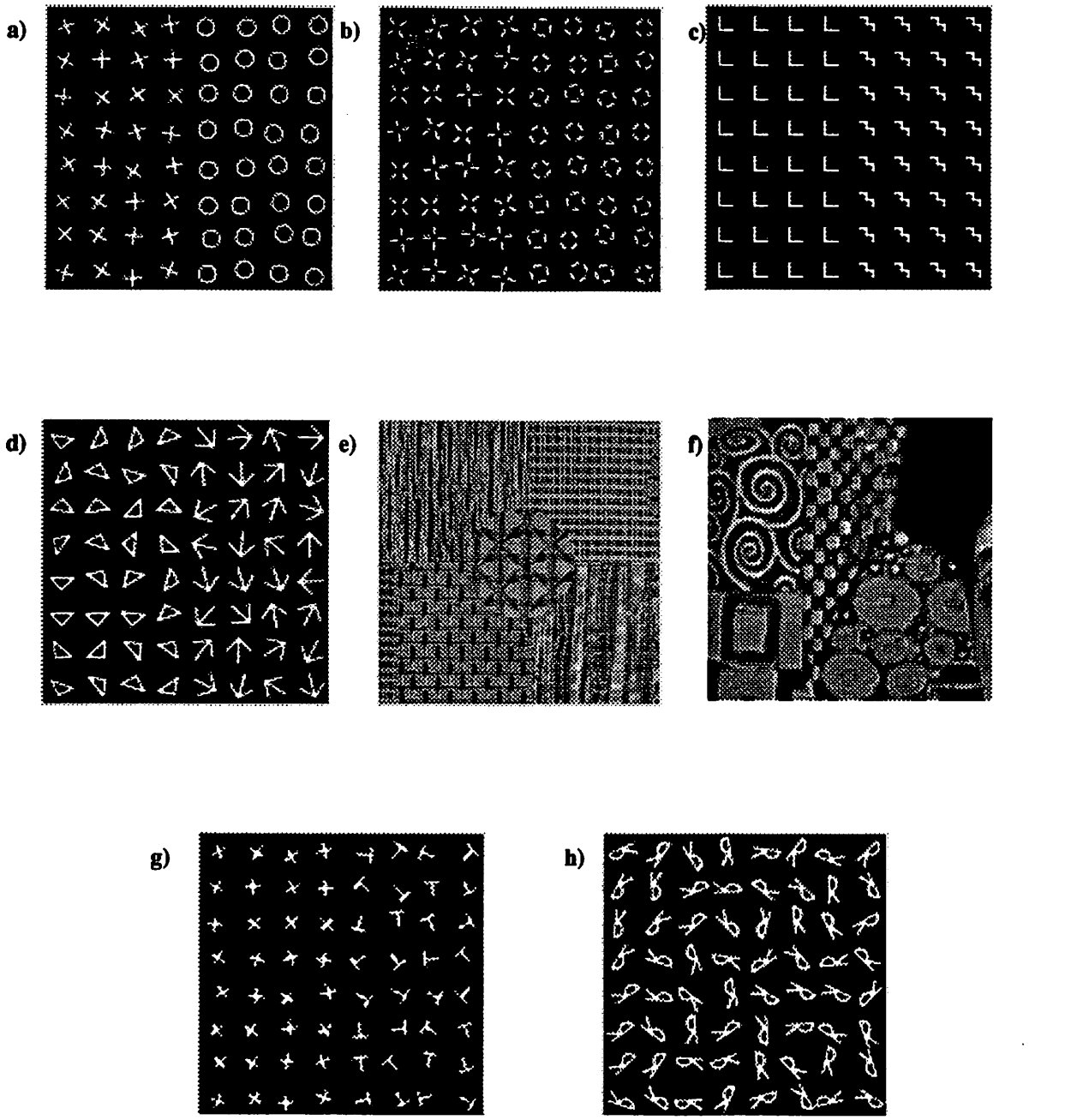


Figure 1: Texture images used for the texture segmentation implementation.

a) *+/o image.*
b) *Crosses/Squares image.*
c) *Arrows/Triangles image.*

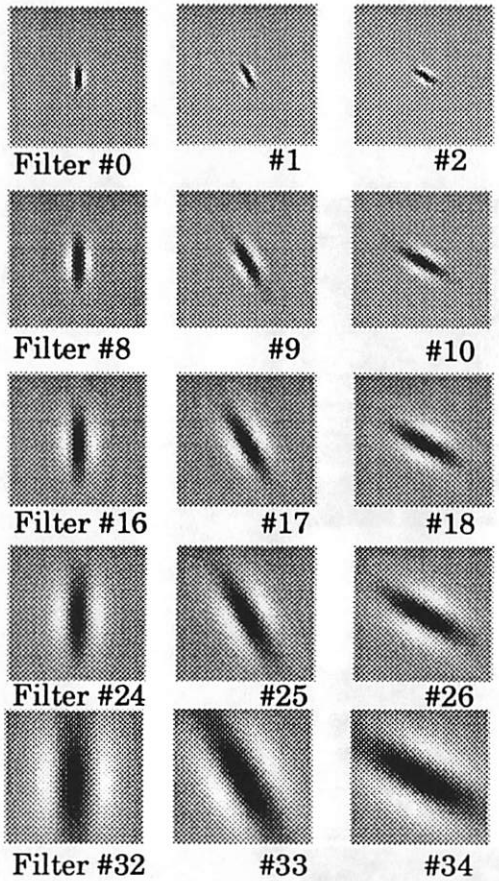
d) *L/M image.*
e) *Patch image.*
f) *Adele image.*

g) *+/T image.*
h) *R/R image.*

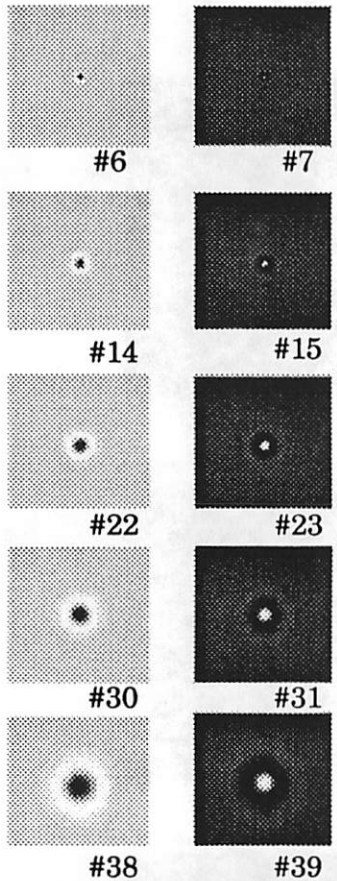
FILTER BANK SET USED IN TEXTURE SEGMENTATION IMPLEMENTATIONS

BASIC SET:

DOOG filters (0,30,60 shown; 90, 120, 150 not shown)



DOG1 filters DOG2 filters



ADDITIONS FOR IMAGE 1:

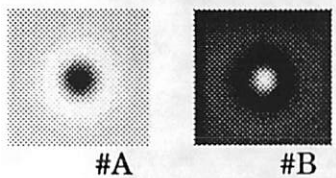


Figure 2. Basic filter set.

includes an implementation and results based upon the Malik-Perona algorithm. Some license may have been taken in portions of the algorithm which were not specifically detailed in [7,8], or were left as open parameters. Discussion of the effects of changing various parameters serves to give insight into the texture segmentation process. Note that no *a priori* knowledge is required about the individual textures, nor of the total number of textures in the image. The set of texture images is shown in Figure 1. The stages of the implementation are as follows:

i. the filtering stage, which inputs one texture image and outputs multiple, parallel *channels*. This stage can be viewed loosely as a bank of matched filters designed to respond to features with certain characteristics.

ii. Two successive nonlinearities applied to each independent channel, one a pointwise nonlinearity and the other an inter-channel nonlinearity, and

iii. Canny edge detection. It is in this last step that the channels can be combined into the final image segmentation.

The implementation in this section follows block diagram (a) in Figure 3.

1. Filtering stage. Although much research has been done on texture *classification*, texture *segmentation* does not necessarily presume, nor look for, knowledge about the individual texture elements. Recent trends in texture segmentation are moving away from the investigation of various properties of individual texture elements, and towards a filter-based approach where global texture properties such as frequency content might differentiate texture regions. When a texture image is convolved with a bank of filters, those filters which are well-matched with one of

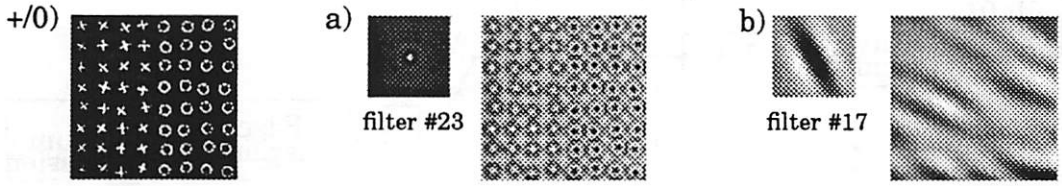


Figure 4: Demonstration of filtering the **+/o** texture image (a) with a filter well-matched to one of the textures (filter 23) and (b) with a filter not matched well with any texture (filter 17). The largest magnitude value in (a) is double that for (b).

the texture regions in terms of shape and scale of texture elements will yield a larger positive or negative response in that region. Those filters which do not match a texture region will give smaller, more uniform responses across the texture boundary. In a sense, the filter bank serves as "matched filters" to features in the texture image shaped like the filters themselves. The filter set used and example convolution responses are shown in Figure 2 and Figure 4, respectively.

TEXTURE SEGMENTATION BLOCK DIAGRAMS

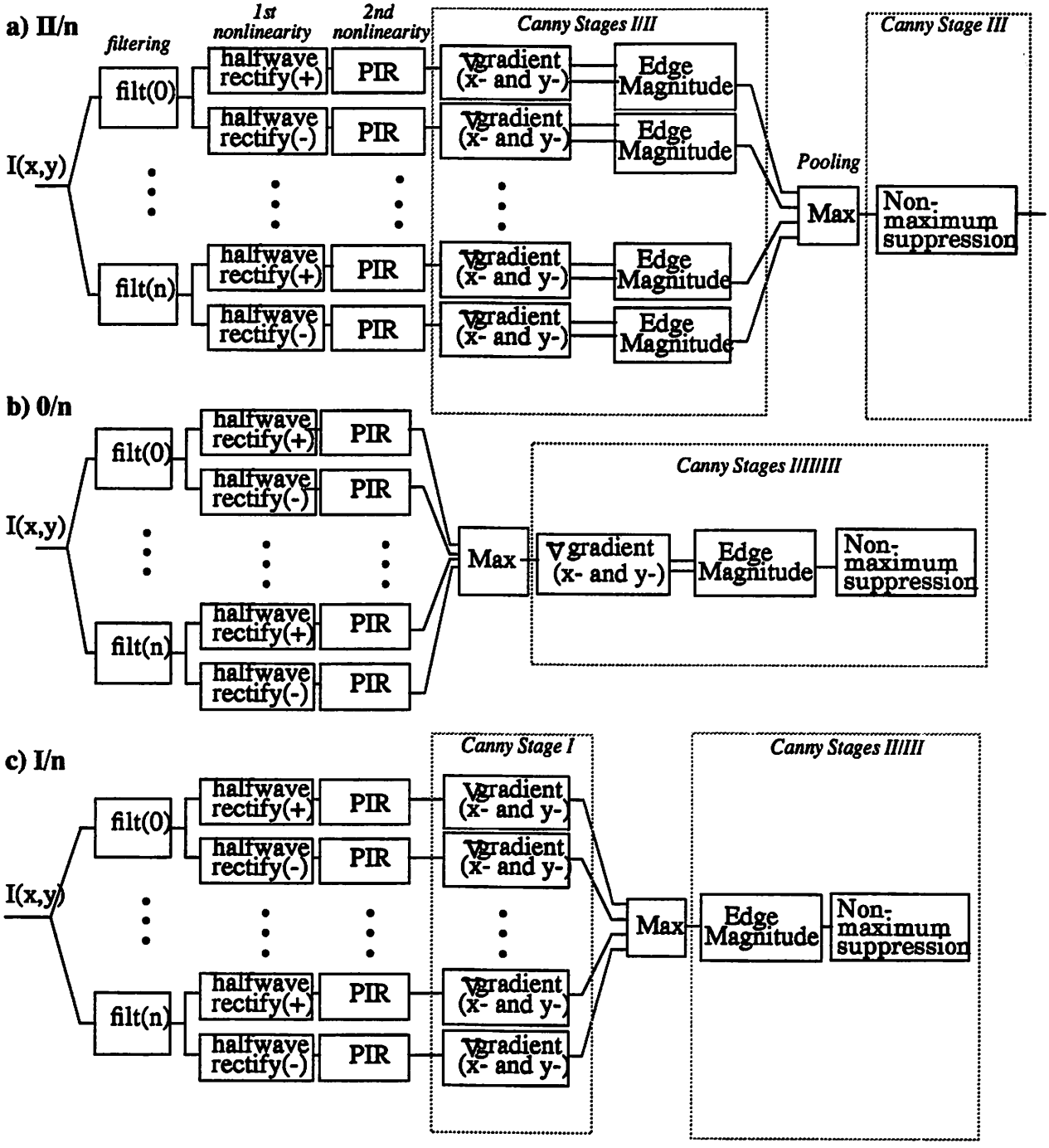


Figure 3: "*Max*" step indicates either norm 1, norm 2, or norm ∞.

a) I/n: apply the first step of the canny operator to each independent channel, then pool the results using an n-norm criterion. The final two steps of the canny operator are applied to the resulting single image.

b) 0/n: combine the PIR images using an n-norm criteria, and then apply all steps of the canny operator to the resulting single image.

c) II/n: apply the first and second steps of the canny operator to each independent channel, and then pool the results using an n-norm criterion. The third and last step of the canny operator are applied to the resulting single image.

Hubel and Weisel [12] originally described visual cells as linear feature detectors matched to multiple scales and orientations; Campbell and Robson [13] described multiple channels all processing in parallel. Atkinson and Campbell's work suggests that phase information could be excluded without great detriment to visual perception[14]. This information suggests a model such as the implementation presented here.

In this implementation, a bank of 40 filters (8 classes of filter at 5 scales), operating essentially in parallel, are convolved with the input image to produce 40 channels of processing. The three types of filter implemented in the filter bank (Fig. 2) are referred to as DOOG(θ,σ) (***d***ifference ***o***f ***o***ffset ***g***aussians), DOG1(σ) (***d***ifference ***o***f two concentric ***g***aussians), and DOG2(σ) (***d***ifference ***o***f three concentric ***g***aussians), where θ refers to the orientation and σ refers to the size of the gaussians used. The DOOG filters occur at six equally-spaced orientations per scale. These filters are essentially matched filters to detect *1) bars* at various orientations (six used here), *2) blobs*, and *3) rings*, respectively. These 40 filters are not to be considered a full set, as only 5 scales are implemented due to the computation and time expense and the fact that these 40 filters are quite effective for a large number of texture images. The 40-filter set is referred to here as the "basic" set.

The DOOG and DOG filters were described by Young as a computationally simplified means of implementing gaussian derivative filters [17]; In the limit as the gaussians of the DOG approach the other in size, the DOG becomes $\nabla^2 Gaussian$. Gaussian derivatives model the point response profile of V1 simple cells in the human visual cortex. Young shows that the DOOG filter, a sum of three spatially offset gaussians, match gaussian second-derivative functions extremely well. DOG*n* (number of gaussians = *n+1*) filters match concentric gaussian-type filter profiles; these second-derivatives-of-gaussians are sometimes referred to as Laplacians. These filters are simpler since they are combinations of gaussians rather than of their derivatives. The general equations, where θ is measured from 0 to indicate the rotation of the DOOGs, are:

$$\mathbf{DOOG}(\theta=0,\sigma) = \frac{-2a}{2\pi\sigma_x\sigma_y} exp\left(-\left(\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y-y_a}{\sigma_y}\right)^2\right)\right) + \frac{4a}{2\pi\sigma_x\sigma_y} exp\left(-\left(\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y-y_b}{\sigma_y}\right)^2\right)\right) \quad (4)$$

$$+ \quad \frac{-2a}{2\pi\sigma_x\sigma_y} exp\left(-\left(\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y-y_c}{\sigma_y}\right)^2\right)\right)$$

$$\mathbf{DOG1}(\sigma) = \quad \frac{3a}{2\pi\sigma_x\sigma_y} exp\left(-\left(\left(\frac{x}{\sigma_i}\right)^2 + \left(\frac{y}{\sigma_i}\right)^2\right)\right) + \frac{-3a}{2\pi\sigma_x\sigma_y} exp\left(-\left(\left(\frac{x}{\sigma_o}\right)^2 + \left(\frac{y}{\sigma_o}\right)^2\right)\right) \tag{5}$$

$$\mathbf{DOG2}(\sigma) = \quad \frac{2.075a}{2\pi\sigma_x\sigma_y} exp\left(-\left(\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2\right)\right) + \frac{-4.15a}{2\pi\sigma_x\sigma_y} exp\left(-\left(\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2\right)\right) \tag{6}$$

$$+ \quad \frac{2.075a}{2\pi\sigma_x\sigma_y} exp\left(-\left(\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2\right)\right)$$

$\theta \in \{0, 30, 60, 90, 120, 150 degrees\}$ and $\sigma \in \{1, 2, 3, 4, 6 pixels\}$. The choice the number of filters to implement is based on a tradeoff between computation time and adequate representation of scale and orientation. The initial choice of using 40 filters worked well for most of the artificial textures such as *+/o*, *crosses/squares*, and *arrows/triangles*. Inspection of the largest center-surround (DOG) filters and the *adele* image, however, reveals that these "ring detectors" are not quite large enough to fully respond to the ring-shaped patterns in the image. In the interest of saving computational time, an extra scale of center-surround DOG filters was added and implemented only for this image, especially to test whether an improvement could indeed be found in the segmentation. Similarly, many smaller texture elements will be well-represented in this basic filter set. The choice of filter scale must inherently be based on an assumption of the scale of texture elements which would be encountered; although a finite set of filters may form a basis for representing the image, time is a practical consideration in potentially reducing the set.

2. Nonlinearity 1. Applying both a positive and a negative half-wave rectification on the filtered images preserves sign information which might be discarded from a full-wave rectification

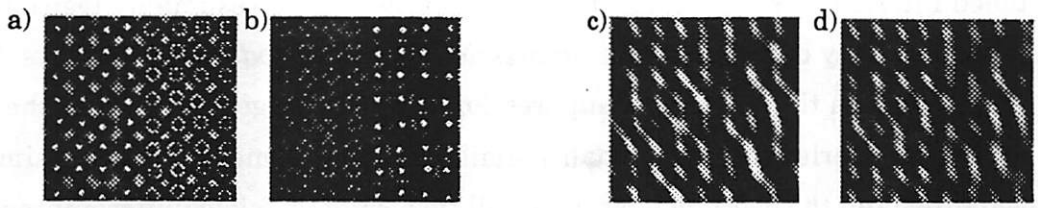


Figure 5: Image *+/O*. a) the positive halfwave-rectified and b) the negative half-wave rectified image corresponding to filter 23. The segmentation ultimately comes from the channel corresponding to (b). Compare with the halfwave-rectifications for filter 17 c) and d); no distinction is made between the two texture regions.

or squaring nonlinearity. This step, as mentioned in [5], is particularly useful for texture regions which contain patterns that are negative images of each other. With half-wave rectification, a separate channel is created for each sign. The 40 channels have been further split into 80 channels. Figure 5 illustrates the information retained by halfwave-rectifying the images of Figure 4.

3. Nonlinearity 2. The implementation in this section follows the Malik-Perona paper in using the following two equations:

$$T_i(x_0, y_0) = \max_j \max_{x,\, y \varepsilon I_{ji}(x_0, y_0)} \alpha_{ji} R_j(x, y) \tag{7}$$

$$PIR_i(x_0, y_0) = \max_{x,\, y \varepsilon S_i(x_0, y_0)} \frac{1}{1 - \alpha_{ii}} [R_i(x, y) - T_i(x, y)]^{+} \tag{8}$$

where

T = thresholds under which responses will be suppressed

PIR = the resulting "Post-Inhibition response"

$I(x_0, y_0)$ = *inhibition neighborhood* about which pixels can affect other pixels; chosen according to Table 1.

α_{ji} = weighting factor for the inhibition; chosen according to type of channels interacting, Table 2.

S = *suppression neighborhood* from which maximum values are chosen to generate each PIR image.

+ = positive halfwave symbol.

In this nonlinearity stage, inter-channel interactions occur but only among certain combinations of channels. Intuitively, the PIR equation suppresses any pixel value which lies below the T, or

threshold, value (as indicated by the positive halfwave-rectification symbol). This is based upon the assumption that spurious responses to non-matched filters occur and are less than responses to matched filters. It also "spreads around" the larger (and presumably legitimate) responses into the local vicinity of radius S, the *suppression neighborhood*, of those points. The best results are obtained when the S radius of suppression is chosen large enough that the circles (Fig. 6) are somewhat overlapping; presumably similar texture elements will give similar responses to a given filter, and the overlapping circles will then create a fairly uniform intensity surface within regions of similar texture elements. Figure 6 shows circles with the S radius a bit too small for the *+/o* image; the next step of finding texture gradients will find sharp discontinuities in locations where texture boundaries do not exist. Also shown in Figure 6 are examples of an ideal radius for the same images. Contrast this with image *arrows/triangles*, where the texture elements are not (radially) symmetrical. The arrows, for instance, when filtered with DOG symmetrical filters, will give a higher response at the intersection of the lines. Since the arrows are pointing in random directions, the high response points will not be evenly distributed. In this case, the best results are obtained with an S radius even larger than for the *+/o* image of Figures 4 and 5 and the *crosses/squares* image, which have very symmetrical texture features.

+/O image:

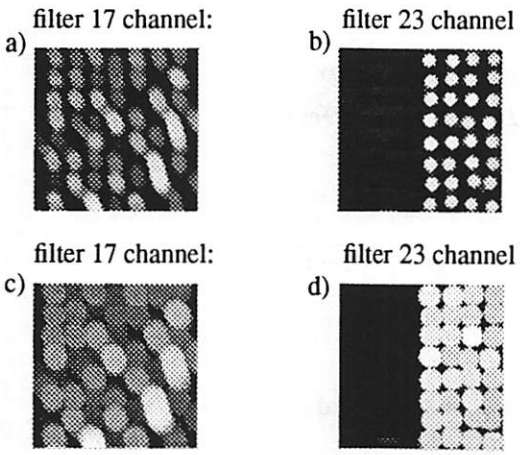


Figure 6: Choice of S, the "suppression radius" for PIRs. A radius which is too small (a, b) will lead to intra-texture gaps; an appropriate radius (c, d) will create a better "tiling" over a texture region, lending itself to edge-finding at correct boundaries due to intensity differences in the next stage. (a) and (c) correspond to filter 17; (b) and (d) correspond to filter 23 which alone contributed to the final segmentation.

Note that the calculations of T(x,y) and PIR(x,y) for a given channel i are not dependent upon all other channels; instead, they are segregated according to scale and orientation. While the description in [9] gives a plausible biological rationale for this segregation, it does not preclude all combinations of segregation. The implementation used here adheres to the following algorithm and variables in Tables 1 and 2 for deciding which channels j are considered in the calculations of T(x,y) for channel i, as follows:

- channel i and j must be the same scale in this implementation.
- if channel i is of type DOOG, channel j must be: a) DOOG of the same orientation and the halfwave pair of i, or b) DOG1 or DOG2.
- if channel i is of type DOG1 or DOG2, channel j may be DOOG, DOG1, or DOG2, as long as $i \neq j$.
- otherwise, channel j is not considered in these calculations for channel i.

Tables 1 and 2 of weighting coefficients and radii for the PIR inhibition neighborhoods are shown here:

Table 1. Weighting coefficients α_{ji} for inhibition, eqn 7.

	channel j		
channel i	DOG1(σ_j)	DOG2(σ_j)	DOOG2(σ_j,r,θ_j)
DOG1(σ_i)	0.2	0.45	0.15
DOG2(σ_i)	0.45	0.25	0.20
DOOG2(σ_i,r_i,θ_i)	0.15	0.20	0.65δ (θ_i, θ_j)

Table 2. Radius for inhibition neighborhoods l_{ji}, eqn 7.

	channel j		
channel i	DOG1(σ_j)	DOG2(σ_j)	DOOG2(σ_j,r,θ_j)
DOG1(σ_i)	$2\sigma_j$	$1.5\sigma_j$	$1.25\sigma_j$
DOG2(σ_i)	$2\sigma_j$	$1.5\sigma_j$	$1.25\sigma_j$
DOOG2(σ_i,r_i,θ_i)	$2\sigma_j$	$1.5\sigma_j$	$1.25\sigma_j$

4. Canny edge detection. As mentioned above, the Canny edge operator can be separated into three stages. Given the multiple channels of processing, there must be, at some point, a pooling

CANNY STAGE I & II EXAMPLES FOR TWO CHANNELS

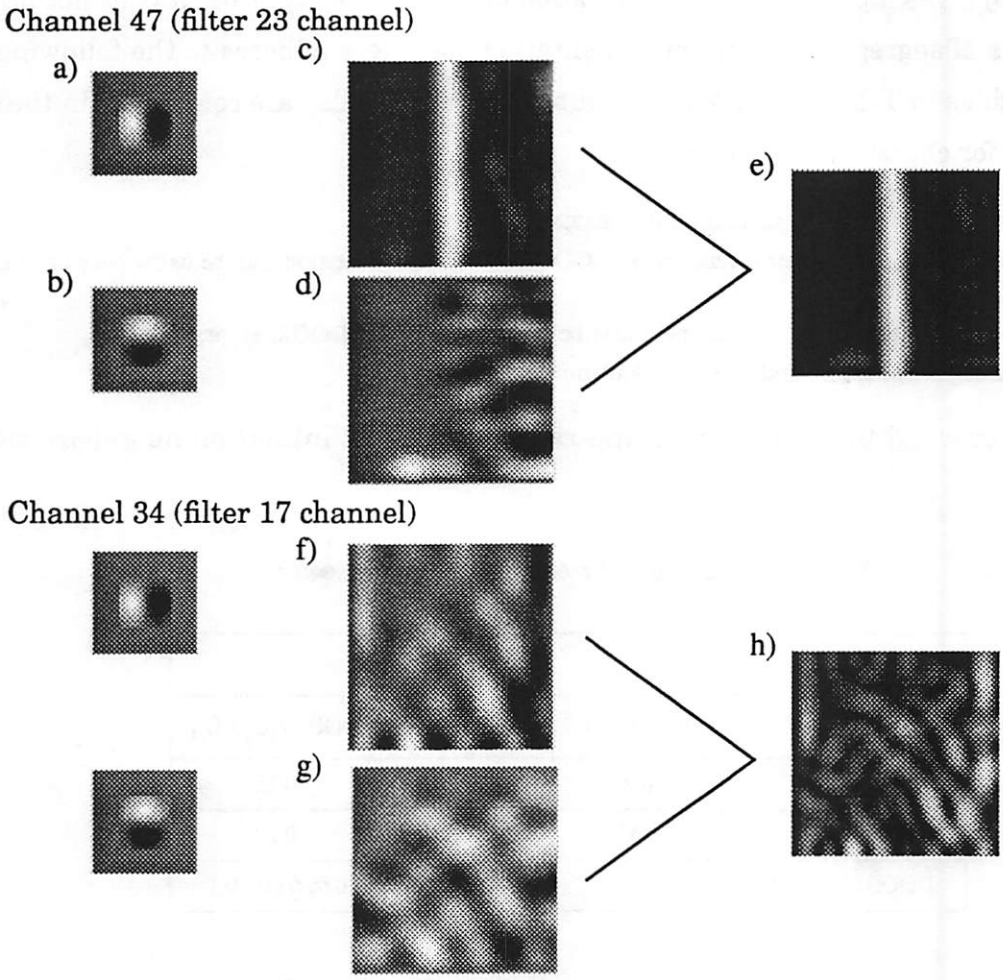


Figure 7: Image +/0. The PIR images in all channels are convolved with *gaussian first-derivative filters* in the x- and y-directions. This is essentially the step-edge detector which gives the largest response at the junction of two regions of differing intensity values, and zero response over constant-intensity regions.

a,b) gaussian derivative filter in x-direction (a), and y-direction (b).

c,d) Channel 47, corresponding to filter 23/negative halfwave-rectification, after stage I of the Canny edge operation. The image is convolved with the gaussian derivative in the x-direction (c) and with the gaussian in the y-direction (d). The largest magnitude value is four times larger in the *top* image than in the *bottom*, as expected.

e) The magnitude image for this channel, obtained by summing the squares of the x and y images.

f,g,h) Channel 34, corresponding to filter 17/positive halfwave-rectification, after stage I of the Canny edge operation. Since the PIR image did not distinguish among the two texture regions, no boundaries will emerge from this step. The largest value in the magnitude for channel 47 (e), is four times the largest value for channel 34 (h).

of responses in order to obtain a single image. This pooling can be done at any of the three canny stages; in this section, the implementation uses II/∞. *Stage I* of the canny operator applies a smoothing function - typically a wide gaussian - followed by a gradient operator. This, in effect, averages out intensity values over fairly large areas giving larger responses at the junction of areas whose "smoothed average values" are different, and near-zero responses over more constant areas. This first step is equivalent to convolving the PIR images in the x- and y- directions with wide first-derivatives-of-gaussians. The resulting output can be considered to be an image whose pixels are vector-valued [x,y]; in *stage II*, various norms (the Euclidean norm is used here) can also be taken of the vector-valued image(s) to obtain magnitudes for each channel; in *stage III*, non-maximum suppression ferrets out the high-magnitude edges, which correspond to the segmentation. Results for two channels are shown in Figure 7.

Two thresholds are used in the implementation of step III. All pixels below the lowest threshold, t1, are eliminated. Points above the upper threshold, t2, are marked as a possible edge starting point. The remaining pixels are designated "edge" points only if they are adjacent to previously marked edge points. The two thresholds were chosen in the experiments as follows: once a pooled magnitude response is found, it is normalized to [0,255] and its histogram taken. Often, there is a double-peak shape to the values. The saddle point is taken to be the value of the lower threshold, since presumably the values above it represent values along the high-valued segmentation border. The upper threshold may be taken to be the value at which there are most pixels. These are scaled back to their original values (equivalently, the normalized pooled magnitude image can be used). This method of choosing thresholds worked quite well in guiding choices leading to the best segmentation.

5. Segmentation Results Summary: The II/∞ pooling is used in this implementation: steps I and II of the canny operation are applied to each independent channel, the channels pooled, then apply step III to the resulting single image. Some results are shown in Figures 8-13. The segmentation results are all based on texture images normalized to 255. This ensures that an adequate range of values were represented in the image. Some research substantiates the ability of humans to discriminate much better when the amount of contrast between figure/ground is higher, as well as being of "opposite" signs (corresponding to dark/light compared to the mean value of the image) [11]. Processing the texture images with normalized gaussians significantly reduces the range of values for the filtered images, and this may be a partial explanation of the need for high contrast.

***+'s and Os** image.* Figure 8 is the magnitude image of the pooled channel responses. c) is the histogram of b, with arrows indicating where low and high thresholds were chosen for finding the edges. d) The segmentation produced for this texture is excellent. The boundary between the texture regions is found, and no spurious responses are generated. Figure e) shows which channel produced the segmentation. In this case, only channel 47, which corresponds with filter 23, contributes to the segmentation. It is informative to compare the texture image with filter 23. Notice that the filter is, in effect, a "matched filter" to the Os texture in both shape and scale.

***Crosses and Squares** image.* Figure 9. The boundary between the two textures has been found, although a spurious edge is also apparent. The true boundary is found from channels corresponding to filters 3 and 23; filter 3 is shown and is an excellent match with the horizontal "bars" in the *crosses* texel. Channel 0 contributes to the spurious line, which may be explained by the predominance of vertical lines in the leftmost column of squares as compared with the second column of squares; filter 0 is essentially a "small vertical line detector."

***Ls and Ms** image.* Figure 10. This segmentation is excellent. The correct boundary has been found at its true location, the line is smooth, and no extra edges are included. The channel leading to the segmentation decision corresponds with filter #0. Inspection shows that this vertical DOOG filter matches to vertical lines of the "L" texture. A diagonal DOOG does match the "M" texture but its response is not as strong.

***Triangles and Arrows** image.* Figure 11. A larger value for the inhibition radius, S=10.0, produced a very good segmentation. Channel 31, or filter 15, is the sole contributor to this segmentation. There are also two spurious edges which are attributable to filters 0 and 3. The upper spurious edge is easily explained with the match of a horizontal line in the arrow to filter 3; the lower edge has no obvious reason for being.

***Patchwork** image.* Figure 12. A larger value for the S radius produced a very good segmentation. Channel 31, or filter 15, is the sole contributor to this segmentation.

Portion of "Adele." Figure 13. The basic filter set is used to obtain a segmentation; the results are rather good, with the exception of a spurious edge in the lower left half corner. Upon inspection, however, it appears that the basic filter set does not contain radially symmetric filters large enough to accurately match texture elements such as the spirals and the squares. To test if an improvement can be made, two extra filters, DOG1 and DOG2, are added to the set at scale

RESULTS FOR +/*0* IMAGE

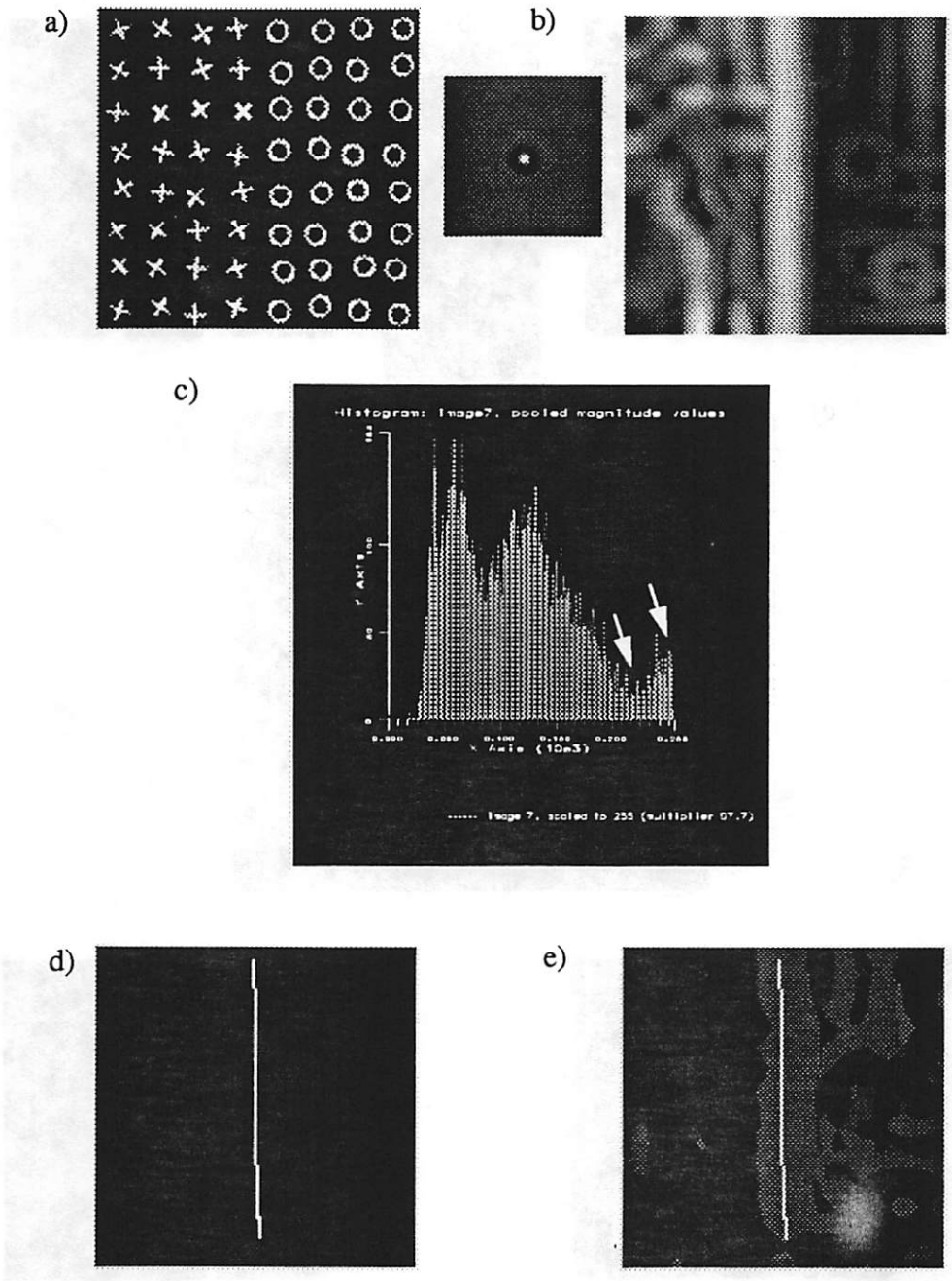


Figure 8:

a) Texture image +/*o*.

b) Magnitude image pooled from all channels for image +/*o*.

c) Histogram of pooled magnitudes with all channels contributing. Arrows indicate x-axis location of upper and lower thresholds. Normalization factor = 97.7.

d) Segmentation results. From histogram, t1 = 2.25, t2 = 2.5.

e) Image which takes on value corresponding to channel number contributing to pooled magnitude image. By masking with the segmentation image, it becomes apparent that only channel 47, i.e., filter 23,contributes to this particular segmentation.

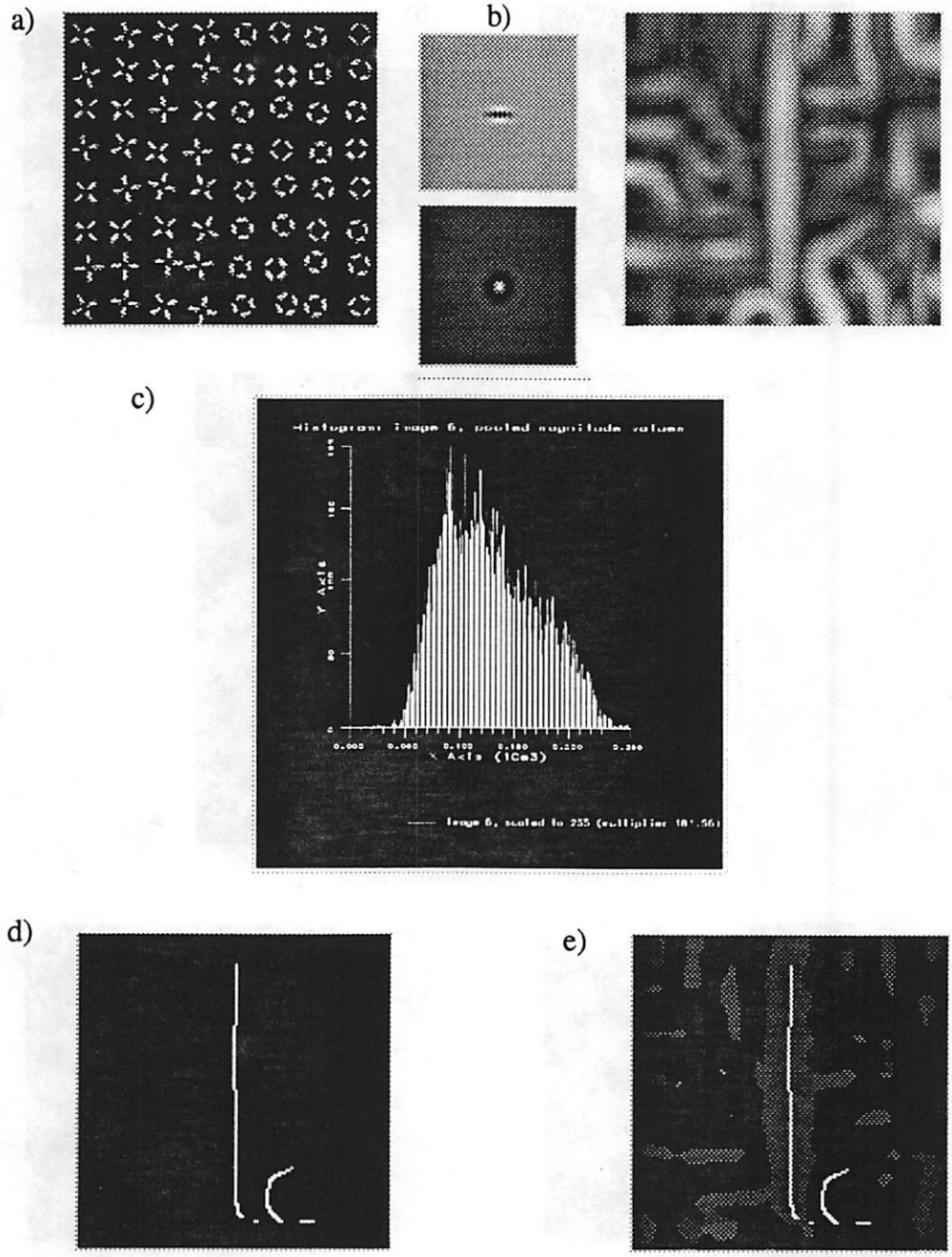


Figure 9:

a) Texture image *crosses/squares* and filters #3 and #23.

b) Magnitude image pooled from all channels for image *crosses/squares*.

c) Histogram of pooled magnitudes of all channels. Arrows indicate x-axis location of upper and lower thresholds. Normalization factor = 181.56.

d) Segmentation results. From histogram, t1 = 1.05, t2 = 1.1.

e) Image which takes on value corresponding to channel number contributing to pooled magnitude image. By masking with the segmentation image, it becomes apparent that channels 6 and 47, i.e., filters 3 and 23,contribute to this particular segmentation.

RESULTS FOR *L/M* IMAGE

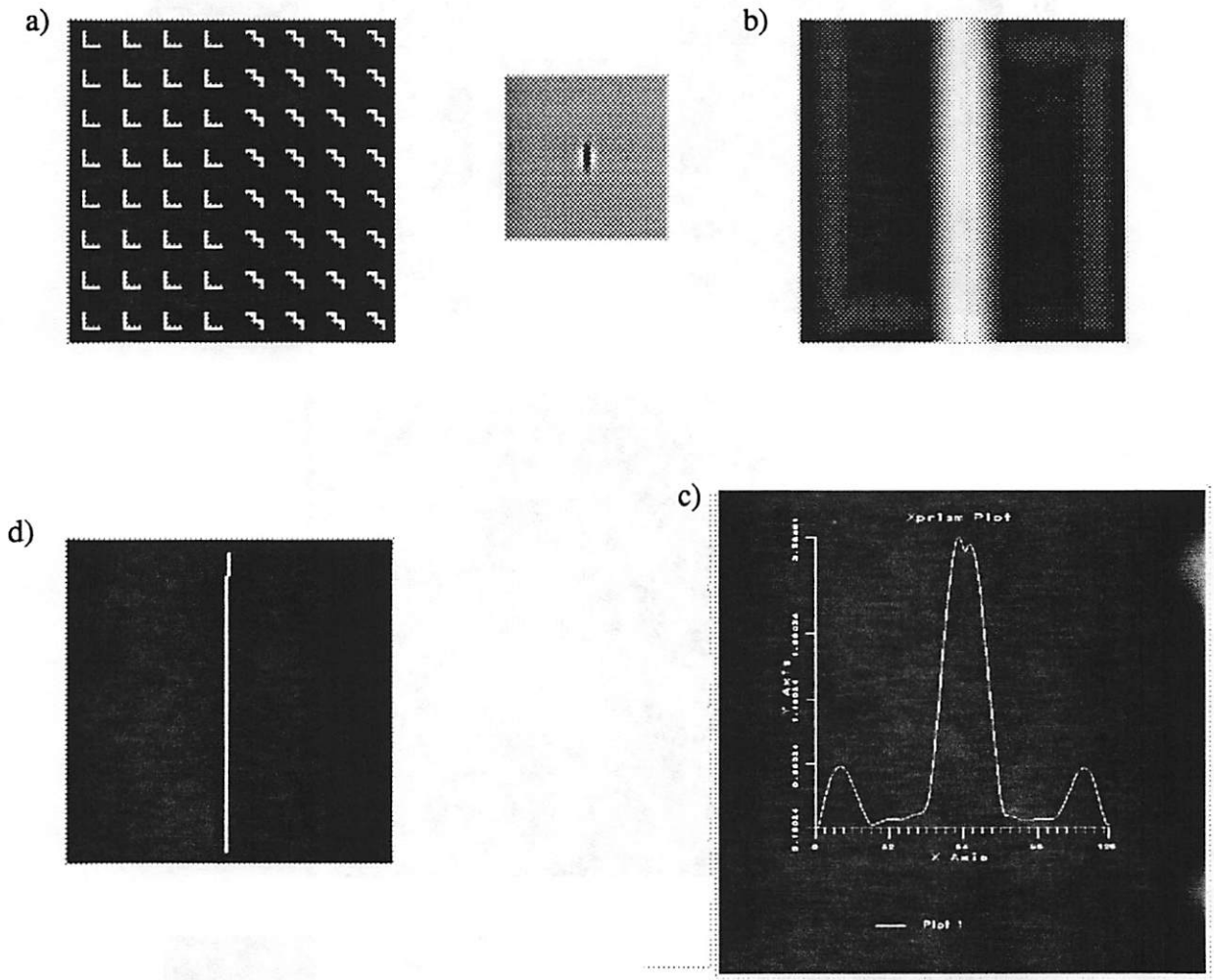


Figure 10:
a) L/M image at σ = 8.0. Filter #0 which leads to segmentation.

b) Pooled magnitude result. square in image1.

c) The average over all rows of (b); The texture boundary is taken to be the pronounced maximum. The peak is larger than the noisy sidelobes by 5.8 dB and larger than the valley points by 10.4 dB.

d) Segmentation result.

RESULTS FOR triangles/arrows IMAGE

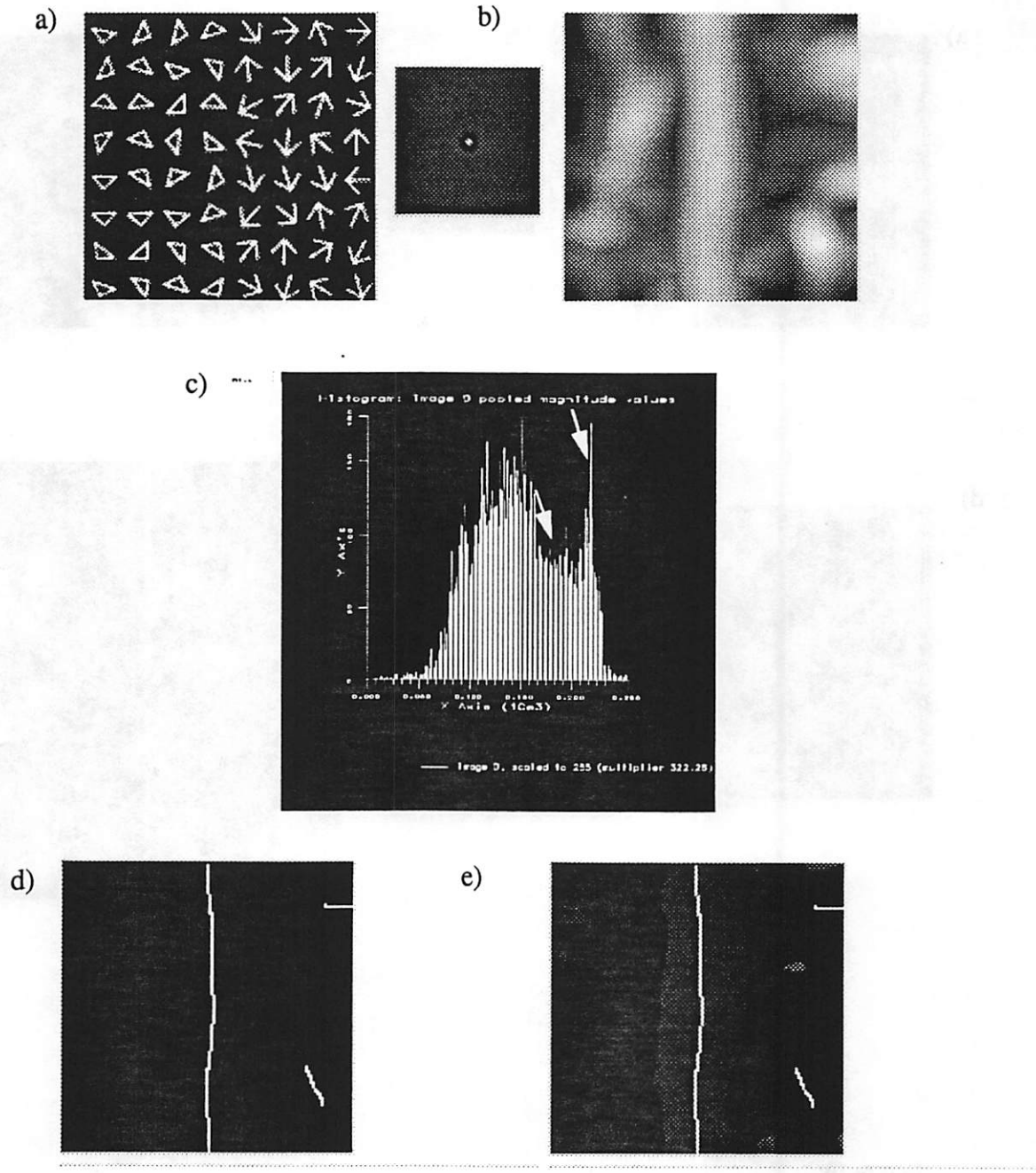


Figure 11:

a) Texture image *triangles/arrows* and filter #15. Notice the asymmetry for which a larger S radius is beneficial.

b) Magnitude image pooled from all channels for image *triangles/arrows*.

c) Histogram of pooled magnitude image from all channels. Arrows indicate x-axis location of upper and lower thresholds. Normalization factor = 322.28.

d) Segmentation results. From histogram, t1 = 0.54, t2 = 0.69.

e) Image which takes on value corresponding to channel number contributing to pooled magnitude image. By masking with the segmentation image, it becomes apparent that channel 31, or filter 15, contributes to this particular segmentation. The spurious lines are attributed to channels 1 and 7, or filters 0 and 3.

RESULTS FOR *PATCHWORK* IMAGE

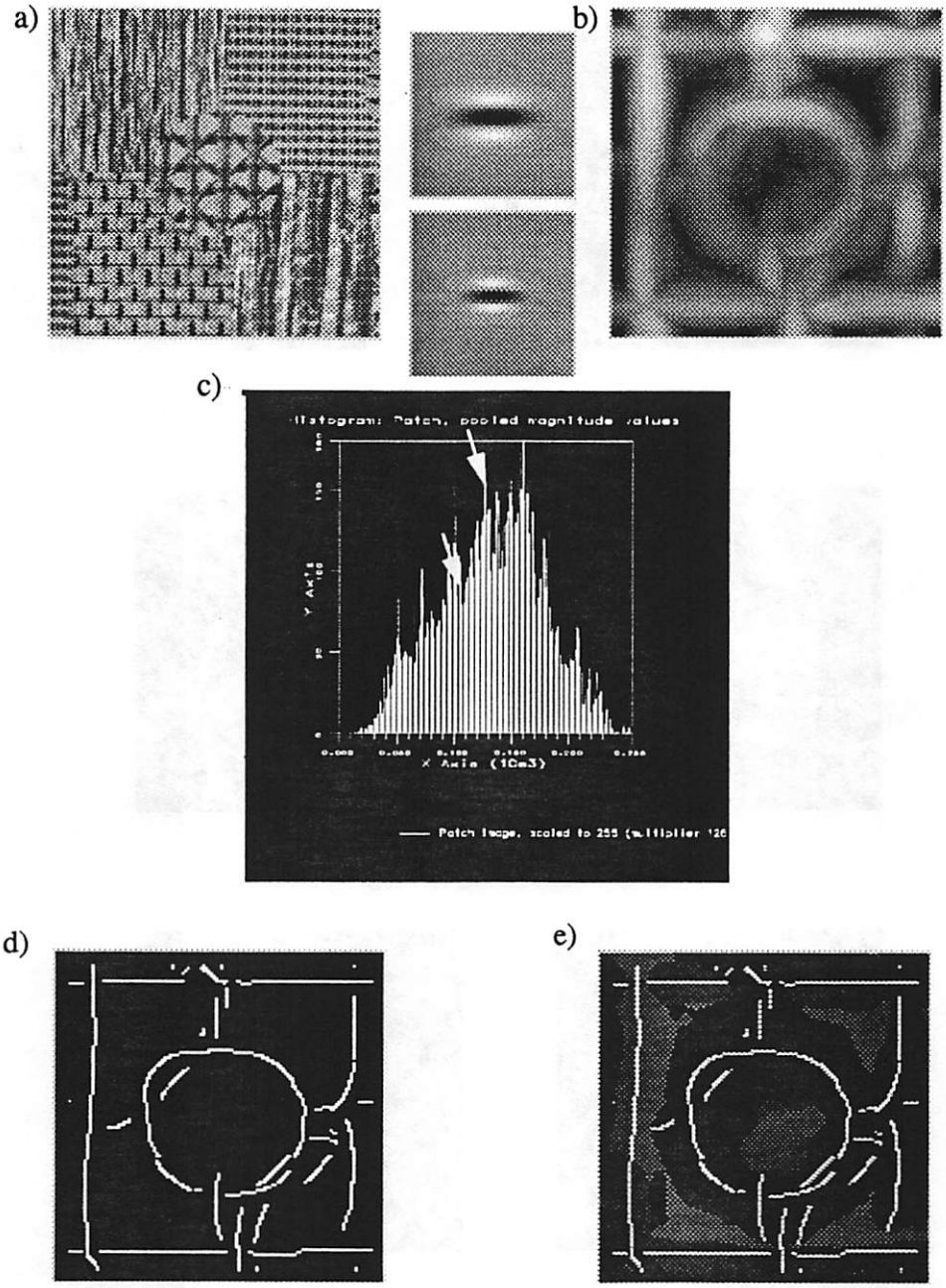


Figure 12:

a) Texture *patchwork* image. and filters #19 and #22.

b) Magnitude image pooled from all channels; PIR suppression radius $S = 10.0$ and gaussian $\sigma' = 8.0$.

c) Histogram of pooled magnitudes. Arrows indicate x-axis location of upper and lower thresholds.

d) Segmentation results.

e) Image which takes on value corresponding to channel number contributing to pooled magnitude image. By masking with the segmentation image, it becomes apparent that channel 39, or filter 19, contributes most to this particular segmentation. Other contributing channels include 22, 23, and 33 (filters 11 and 16).

RESULTS FOR *Adele* IMAGE:

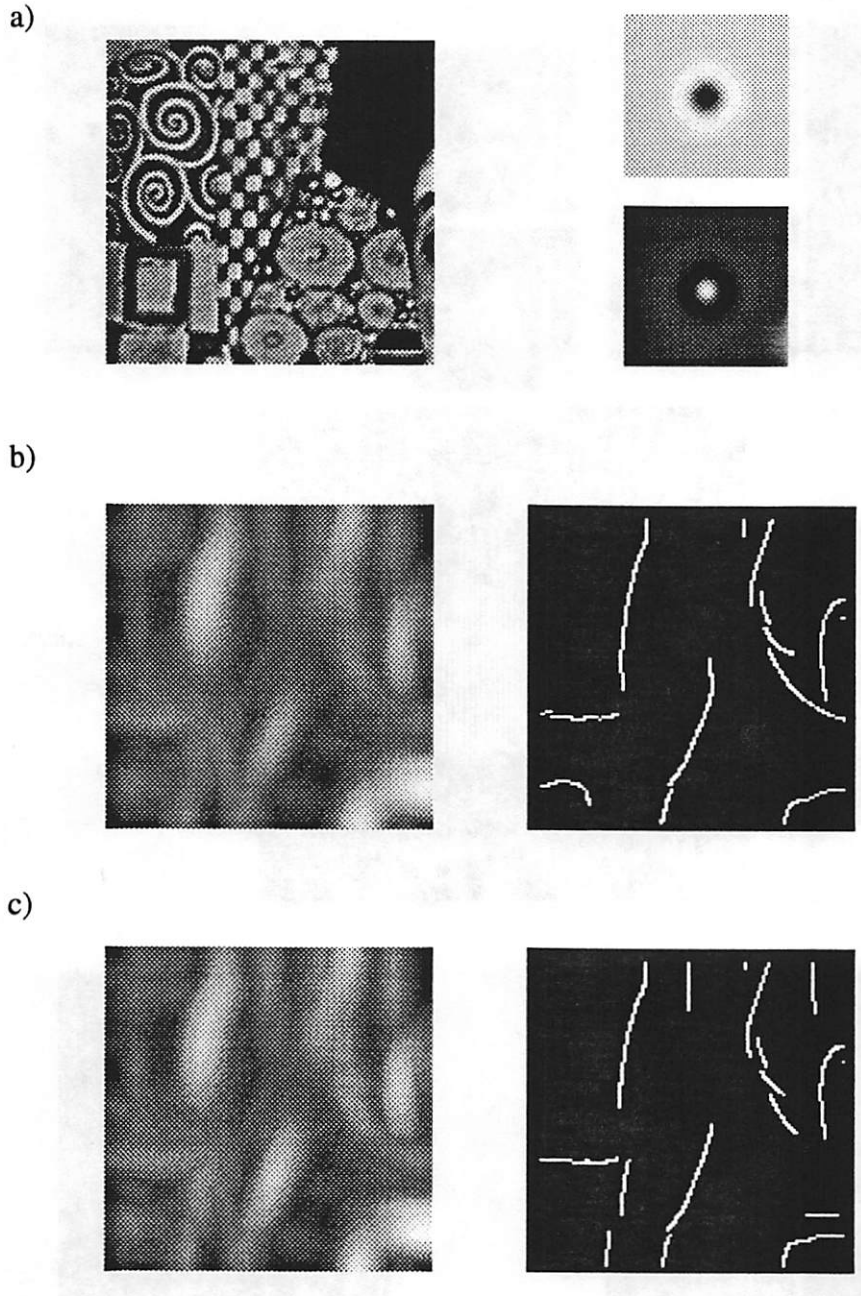


Figure 13:

a) Image of the painting by Gustav Klimt, of Adele image, and additional DOG1, DOG2 filters #A, #B at $\sigma = 8.0$.

b) Pooled magnitude result using basic filter set. Notice the spurious edge in lower left half corner corresponding to the square in image. From Figure 2, notice that the basic set of radially symmetric filters are a bit too small to be a perfect match for those square elements.

c) To test if improvements could be made, the two extra filters shown in (a) are added to the set. Observe the better match to the texture elements in the left half of image. The results are similar, but the spurious edge, lower left, is gone. Notice that a new line has been detected, upper middle. On close inspection, however, one could argue that the checkerboard region is composed of a clear checkerboard pattern, and a smeared checkerboard pattern, and that this line indicates the border.

σ = 8. Observe the better match to the larger texture elements. Indeed the new segmentation appears to be better and the spurious edge is gone. Notice that an extra edge has been detected in the upper middle region. On close inspection, however, one could argue that the checkerboard region is composed of two regions: a clear checkerboard pattern, and a smeared checkerboard pattern, and that this new line indicates the border. Filters which are large enough to exactly match the textures would produce even more improvement in the segmentation.

These results indicate that the current algorithm along with the basic filter set effectively segment many textures. A more complete filter set, however, encompassing both smaller and larger scales than those presently used, will give good results for a wider range of scale of texture elements. The Malik-Perona algorithm, for instance, utilizes 96 filters, more than twice that of this implementation. It is a simple matter to increase the filter set but a compromise between filter number and computational expense sways us towards the reduced filter set.

***+'s and Ts** image:* This texture pair is easily segmentable by humans. This is an example, however, for which the filters in the basic filter set are not quite small enough to distinguish well between the textures. This is an obvious problem which is remedied by adding smaller-scaled filters.

***Rs and backward R**s* image: This texture pair is NOT preattentively segmentable by humans. Similarly, no change in scale of the filter set will cause a segmentation to result from the algorithm. A filter shaped like an "R" or backwards R would cause a segmentation, but this is a ridiculous proposition if the algorithm is to model the human visual system.

D. Variations of Malik-Perona Algorithm

Figure 3 shows block diagram descriptions of the procedures tested. Each of these are a variation on the use of the canny edge operator as a back-end to the algorithm. As mentioned above, they are named I/*n*, II/*n*, III/*n* and 0/*n* where the first number indicates at which step the pooling mechanism is used prior to continuing the remaining canny steps, and the *n* indicates the pooling norm. In addition, one test was done where the PIR step, which is computationally expensive and potentially time-consuming, is replaced by a simpler procedure. This procedure simply takes an image as its input, and replaces each pixel with the largest neighbor value within a given circular vicinity. This spreads out the largest values in the input image and effec-

VARIATION: largest neighbor in place of PIRs
RESULTS FOR *+/0* IMAGE:

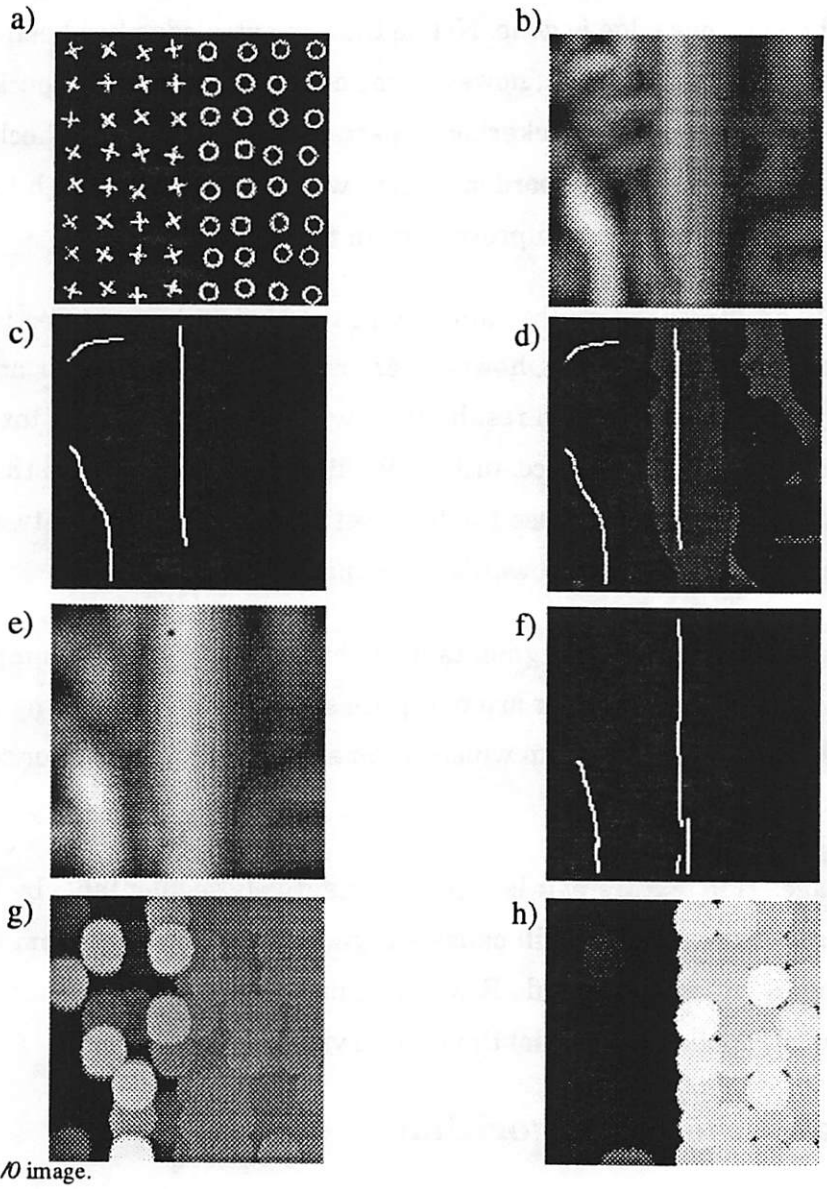


Figure 14: a) *+/0* image.

b) Pooled magnitude result using basic filter set, replacing the PIR second nonlinearity with a simpler step. Without suppressing smaller responses, the largest value found in a neighborhood of each pixel is taken to be the new pixel value. Notice the spurious edge in lower left half corner corresponding to a spurious response which became magnified at the gradient stage.

c) Segmentation produced. The correct texture boundary is found, but so are two non-relevant edges. Largest neighbor radius was 12.0, the gaussian $\sigma' = 12.0$.

d) The true boundary is detected through filter 23 (channel 47), exactly as in the implementation of section c; the upper left spurious edge is due to filter 3 and the lower left spurious edge is due to filter 0 (channel 1), all from negative halfwave-rectified channels.

e) Pooled magnitude result using largest neighbor and adding a normalization of channels. It removes at least the upper spurious edge. Now, channels contributing to texture segmentation include filter 22 (channel 45) also.

f) Segmentations due to largest neighbor replacement plus normalization across channels. The true boundary is found and only one spurious edge remains.

g,h) Channel 1, channel 47, after largest neighbor replacement and normalization of channels. It is easy to spot where the error appears,

tively deletes the smaller values when larger values are nearby. By choosing a radius equal to the suppression radius used in the PIR procedure, the effects are similar..

1. Changing nonlinearities

The Malik-Perona algorithm utilizes two distinct nonlinearity steps in the early processing of texture boundaries. How might the segmentation results change if these nonlinearity steps are changed? The first nonlinearity, halfwave-rectification, seems to be very useful since it retains both positive and negative information in the channels. Fullwave-rectification, another form of nonlinearity, would discard this information.

a. Replacing second nonlinearity with local maximum step

The second nonlinearity (PIRs), which inhibits lesser responses in neighborhoods of larger responses, is replaced by an intuitively similar step. This step simply replaces each pixel value with the largest value found in a circular neighborhood around it; in effect it "spreads out" the large values, but does none of the complicated suppression of lower values found in the PIR step (Figure 14). The result is that in general the boundaries are detected, but several spurious edges are detected as well. The obvious explanation is the lack of thresholding/suppression.

b. Replacing second nonlinearity with squaring operation

Finally the PIR step is replaced by a squaring operation. Vision research indicates that a nonlinearity definitely exists along the visual path and the PIR step is but one hypothesis. If the halfwave-rectified images are called R then the results of this step are simply R^2. The results for the test case on the $+/o$ image are in Figure 15. The value of the suppression variable remained the same at S=*8.0*, but the lack of suppression and "spreading" in this variation meant that a larger smoothing constant is required. The choice of $\sigma' = 16.0$ worked fairly well. The revised texture segmentation algorithm found the essential texture boundary. The quality of the segmentation is lower because the edge is slightly off-center by 4 pixels from the (human) perceived boundary, and a few spurious lines exist. One advantage, however, is the straightness in the detected boundary. The main change in procedure required with the use of this step is to greatly increase the size of the smoothing function of Step I for the Canny operation.

2. changing pooling location

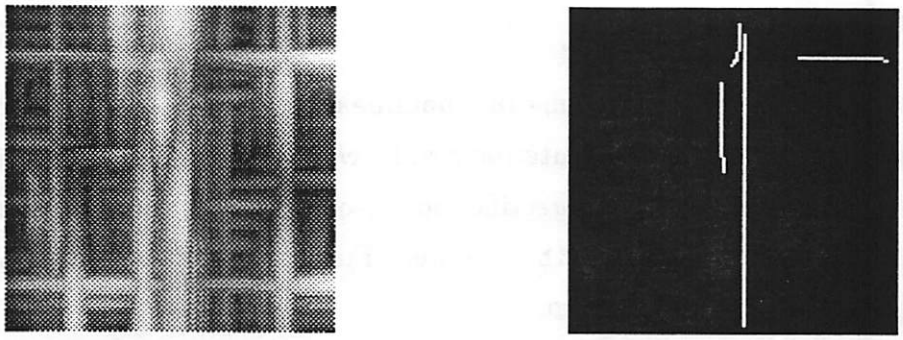


Figure 15:

a) Pooled magnitude image for *+/o*. The basic algorithm is the same, with replacement step PIR = R^2, where R is a halfwave-rectified channel, and larger gaussian $\sigma' = 16.0$ in Canny step I.

As described in the Canny section, there is no set requirement as to when to combine the results of multiple channels into a single pooled image. The Malik-Perona algorithm combines the independent channels after the Step II of the canny operation, or II/norm. The results can significantly change when the pooling location is changed.

a. *0/n* and comparison

Sample pooled PIR results for this change in pooling location are shown in Figure 16. The PIR channels for the *+/o* image are pooled using norm=0, norm=1, and norm=∞. Regardless of the norm used, the pooled image is much less defined along the texture border than the individual PIR images. One expects big differences among the PIR images; combining the PIRs at this stage is premature because this averages a few good channels with many poor but nonzero channels. Because of this loss in definition, the Canny edge operator applied to this pooled image yields poor results. The conclusion is that the pooling location 0/norm is far from optimal.

b. *I/n* and comparison

Sample pooled magnitude images for this pooling location on the +/o image are shown in Figure 17. All three norms were applied, with very different results. The best norm to use in creating the pooled image from the channels is the 2-norm. The ∞-norm gives good results as well; although an extra line is detected, it is nearly negligible. The 1-norm gives a poor magnitude

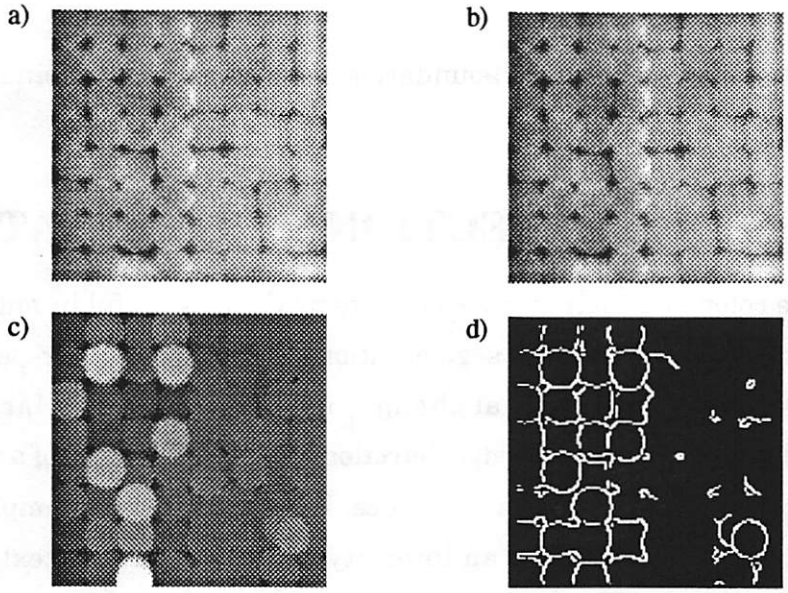


Figure 16: Pooled PIR images for pooling location ***0/n***. All PIR images are pooled according to norm $n \in \{0, 1, \infty\}$, and the Canny operator applied to the resulting single image.

a) Norm 2 resulting PIR for image +/o. Without a big intensity difference in the two regions, no texture boundary can be extracted.

b) Norm 1 resulting PIR. No texture boundary can be extracted from this image.

c) PIR using norm. Canny output shown in d); even with larger σ', the boundary can not be found.

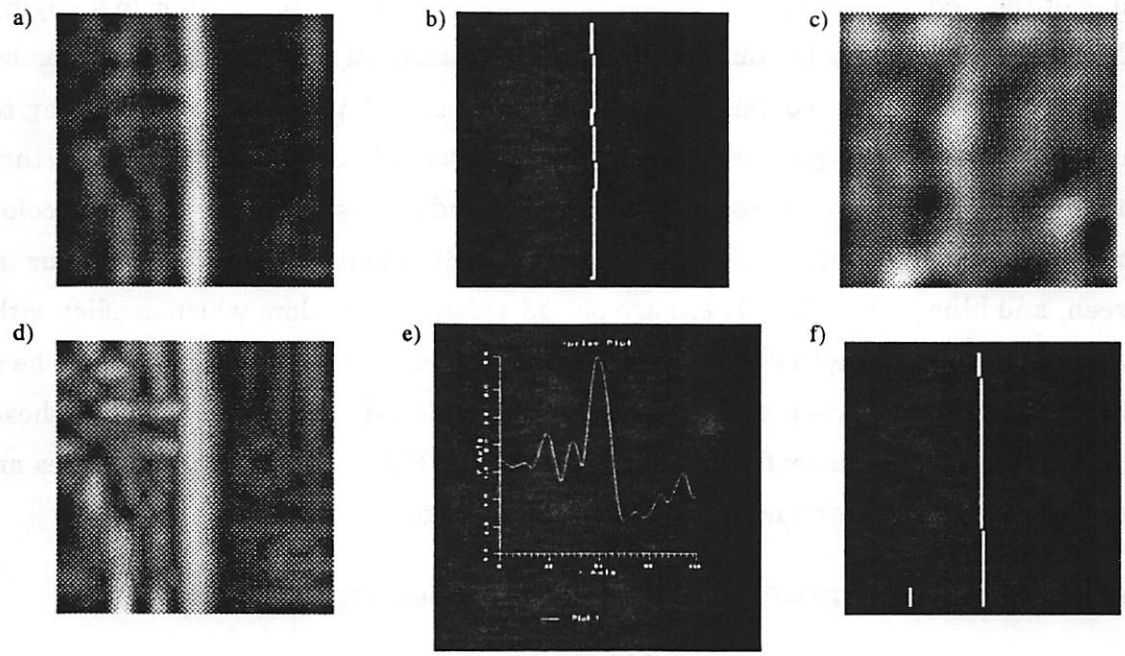


Figure 17: Pooled magnitude images for pooling location ***I/n***. The x-gradient and y-gradient is taken for each channel of PIRs.

a) Norm 2 pooled magnitude image and b) resulting segmentation. This slightly rough segmentation is as good as the standard implementation.

c) The norm 1 pooled magnitude is too nondescript to be of any use.

d) Norm ∞, max-pooled magnitude image, e) the average of all row values for pooled magnitude image; the left sidelobe is 2.5 dB down from the peak while the right sidelobe is 7.7dB down from the peak, and f) resulting segmentation with only a small artifact in the left side.

image which does not segment the texture boundaries. A good pooling combination is the I/2 pair.

IV. COLOR EDGE DETECTION / SEGMENTATION

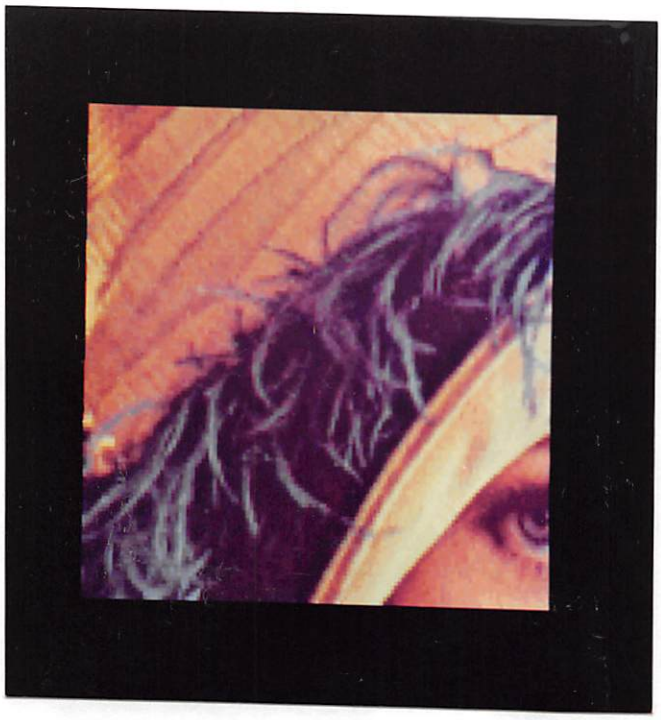
Why should one include color in a computer vision system? Color is useful in multiple applications, such as: 1. object recognition, 2. image segmentation, as we are doing here, and 3. separating highlights and specularity from actual changes in surface. While a large amount of information can be garnered from intensity edge detection and other features of a visual system, some information can only be obtained from color cues. We show with an example that a color edge detector can give improvements over an intensity edge detector or a texture boundary detector alone. Enhanced results occur primarily when the color regions are basically equivalent in intensity levels but differ in hue. This is shown in section IV.C.

The Canny edge detection operator can be applied to color edge detection [9]. For an RGB image, color edge detection often performs better than standard edge detection. Color images can be described with each pixel value being the vector *C=[R G B]*, each component indicating the intensity value of the Red, Green, or Blue bands. The preprocessing required prior to applying the Canny Operator is to separate the image into separate "bands." At least three bands must be used to represent perceived color combinations. Red, Green and Blue bands corresponding to short, medium, and long wavelengths are a common separation scheme, the "trichromacy" theory often attributed to color vision research by Helmholtz and others. A second theory of color vision attributable to Hering, called "color opponency," describes four variables which occur in pairs: red-green, and blue-yellow [20]. These are paired according to colors which conflict with and cancel out one another perceptually; red and green are not often used together to describe a given hue, and blue and yellow are not often used together to describe a given hue. While these two theories may appear to be in conflict, they are both valid; Helmholtz's theory describes an earlier stage of vision and Hering's theory describes a later stage.

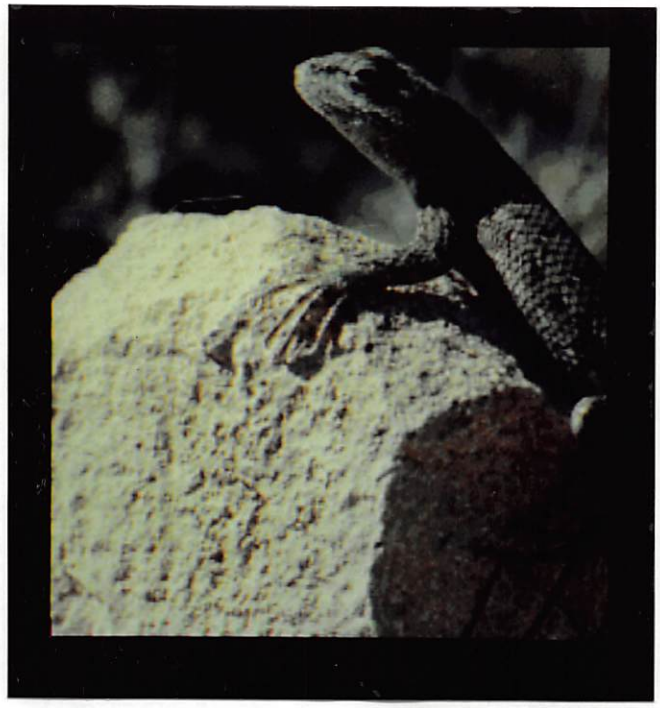
A. Multi-channel implementation - Canny edge operation basis

This implementation utilizes the same combination of edge processing steps as did the texture segmentation implementation. Instead of preprocessing the image by convolving with multiple filters, the image is separated into three channels corresponding to its three color bands. If the three color bands are R, G, B, this corresponds to an earlier stage in the visual model than if the

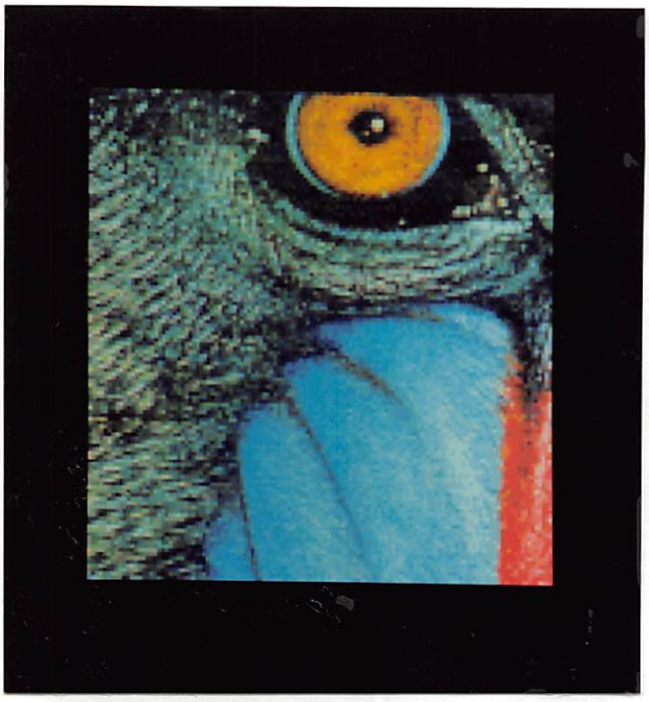
a)



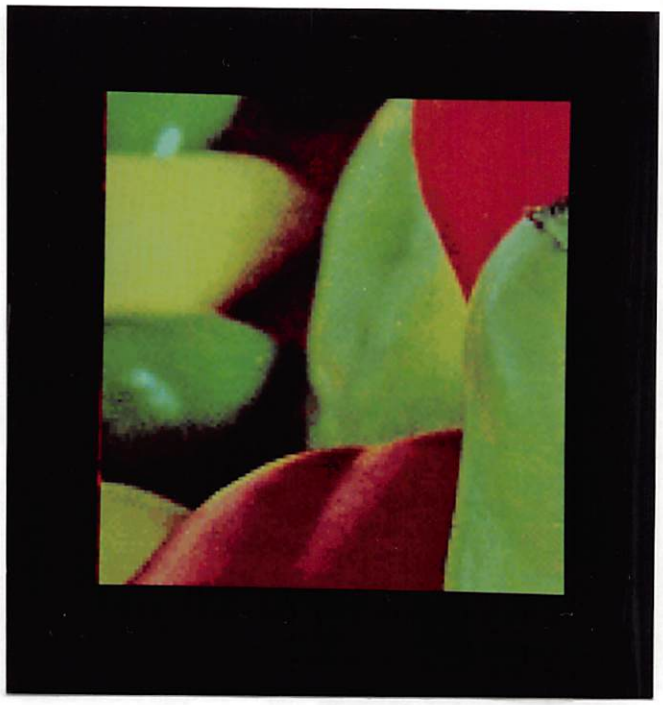
b)



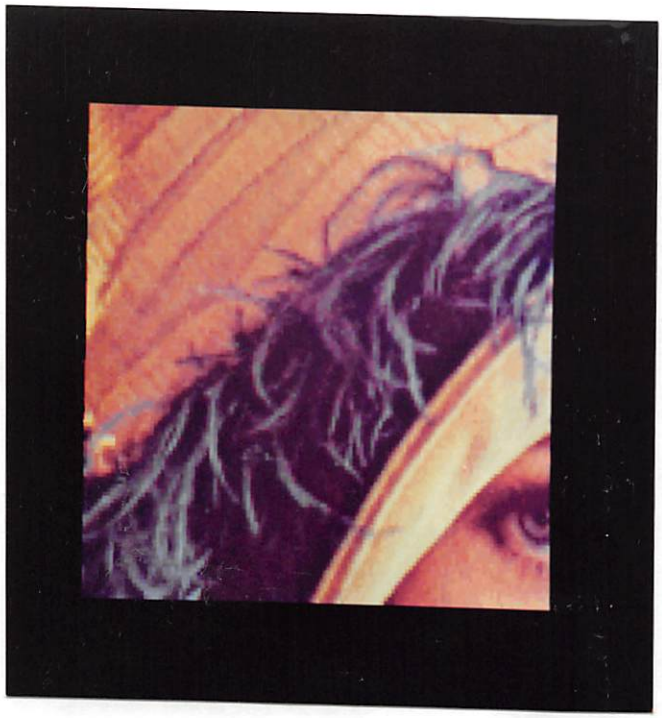
c)



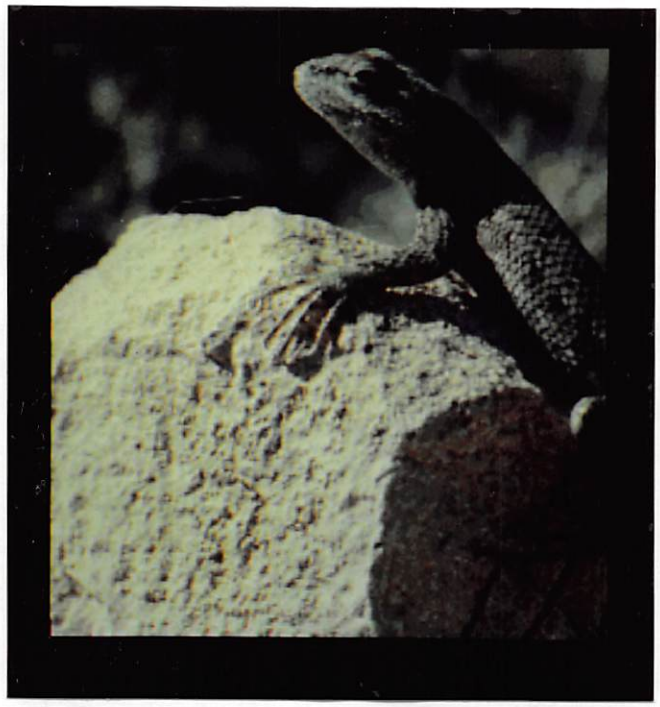
d)



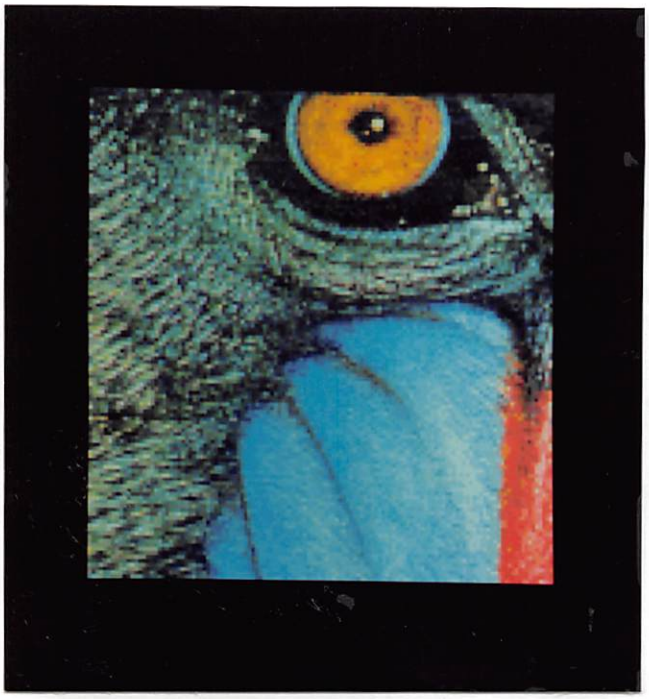
a)



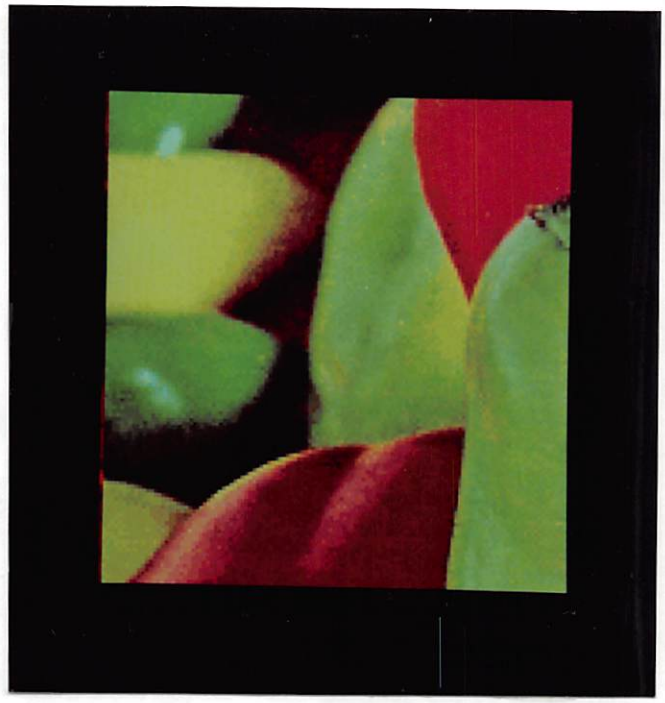
b)



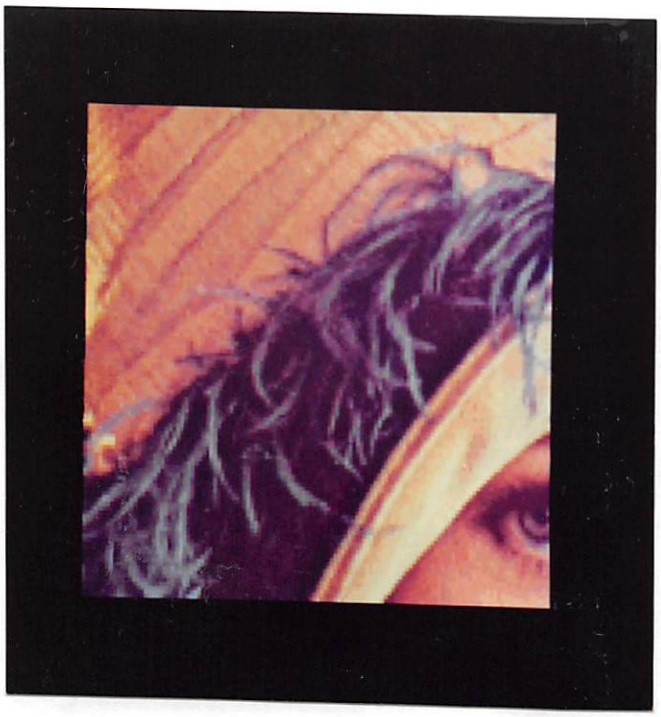
c)



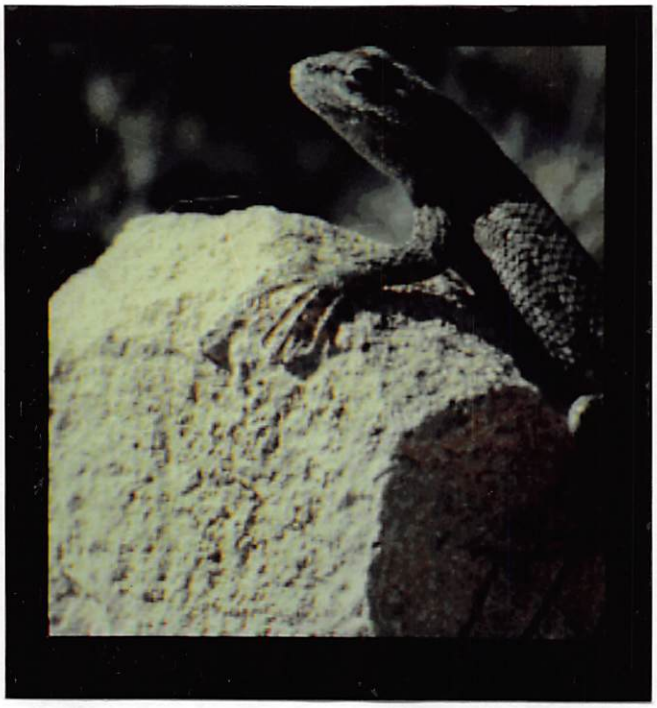
d)



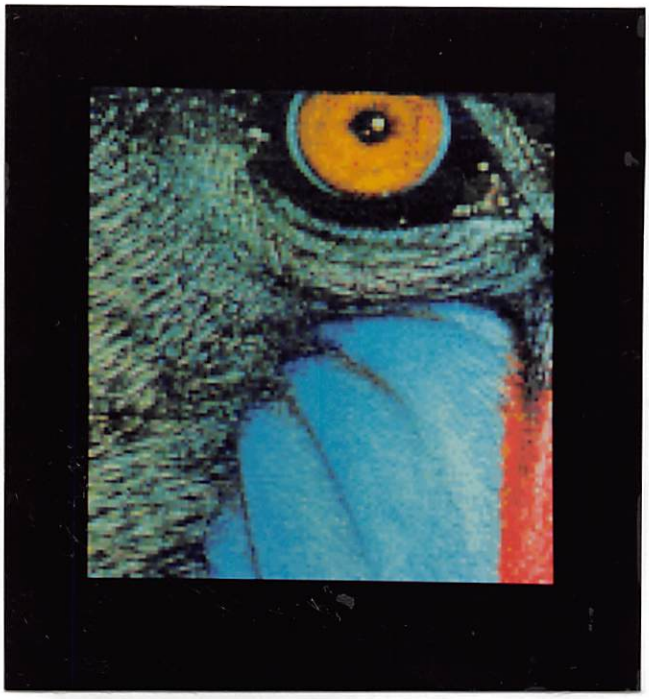
a)



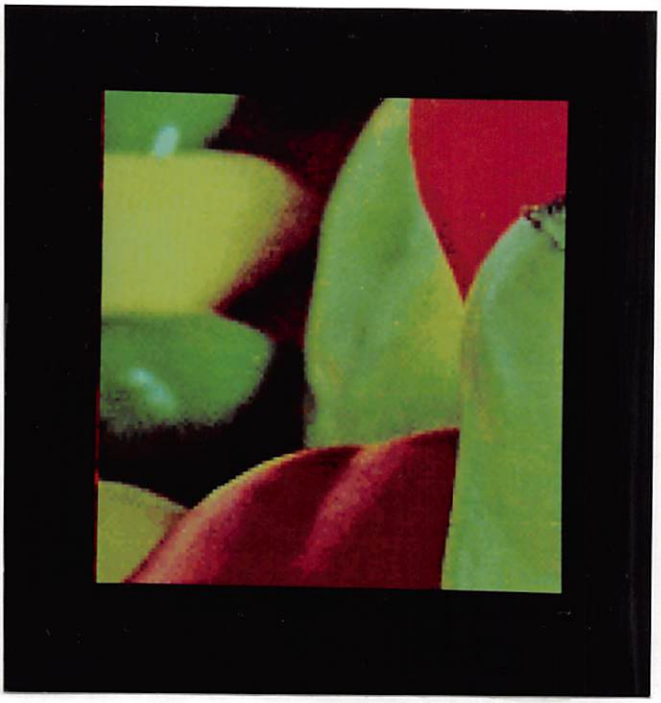
b)



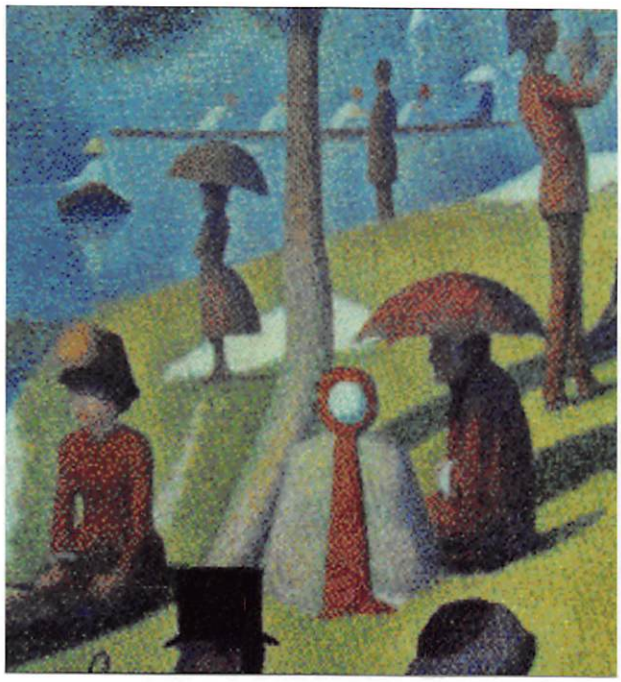
c)



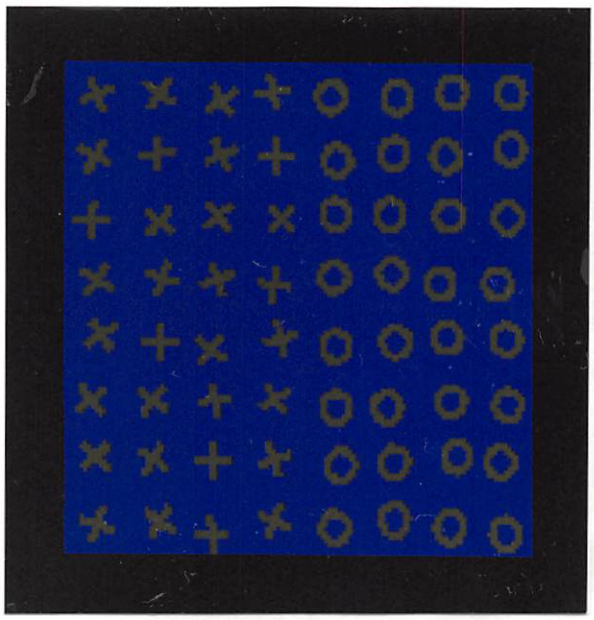
d)



e)



f)



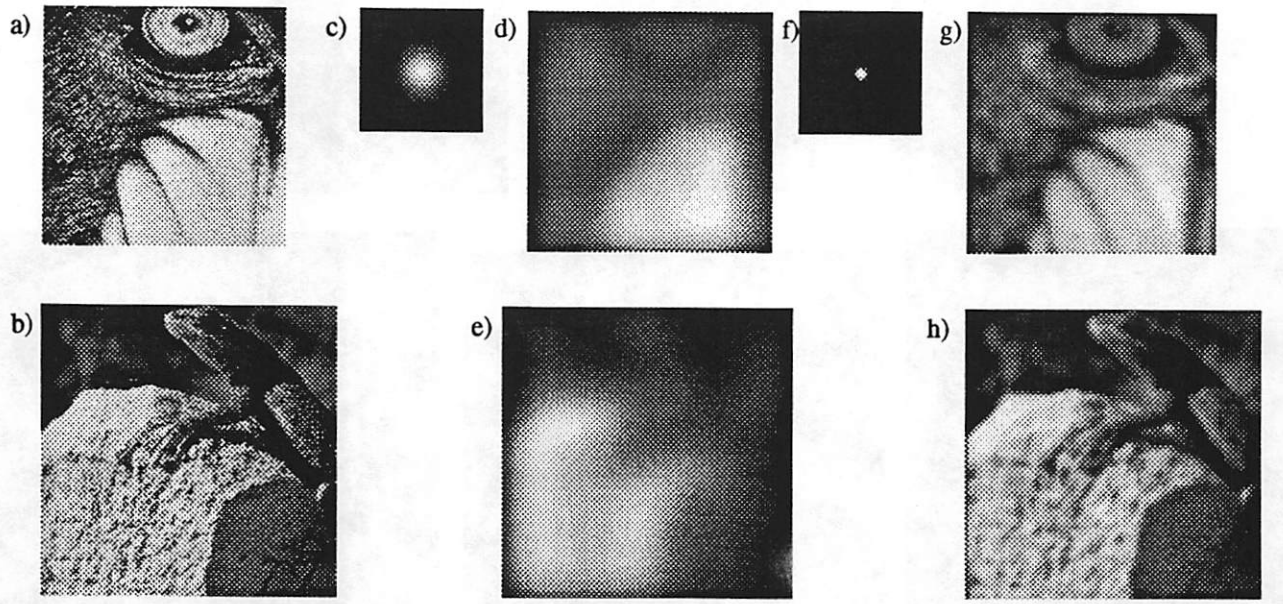


Figure 20: Loss of detail by use of gaussian with scale too large (Mandrill, of Figure 18c)

a,b) The intensity image for mandrill, (a), and lizard, (b)

c,d,e) A gaussian of $\sigma = 8.0$, (c). The result of convolution with gaussian, (d), (e). Notice the loss of detail around the pupil, eye rim, and cheek, and most of the lizard.

f, g, h) A gaussian of $\sigma = 2.0$, (f). The improved result of convolution with smaller gaussian, (g), (h). Notice that more detail is retained while noisy areas have been smoothed.

peppers image is very interesting; it definitely finds the boundaries due to color differences. It does not find boundaries of the individual red and green peppers, as we might do; it does not distinguish among distinct "objects" in the image, only distinct colors. In its function this color boundary detector did quite well. However, given the variety of images and the missed edges in all, a smaller smoothing function should be used, especially if "blindly" rather than chosen by hand. An illustration of this is shown in Figure 20 for $\sigma = 8.0$, and the more appropriate $\sigma = 2.0$.

These color edge detector results missed closely aligned edges in particular and did not generate spurious edges. This problem is indeed fixed by lowering the σ used, which could generate bigger problems in the texture boundary detector rather than here. Examine Figure 22 which shows the effects of changing the sigma size; more and more of the small-scale detail comes out as we smooth with a lesser-sized gaussian; at $\sigma = 1$, the smallest used, finally some non-essential edges are detected. Therefore smaller smoothing functions are beneficial for color edge

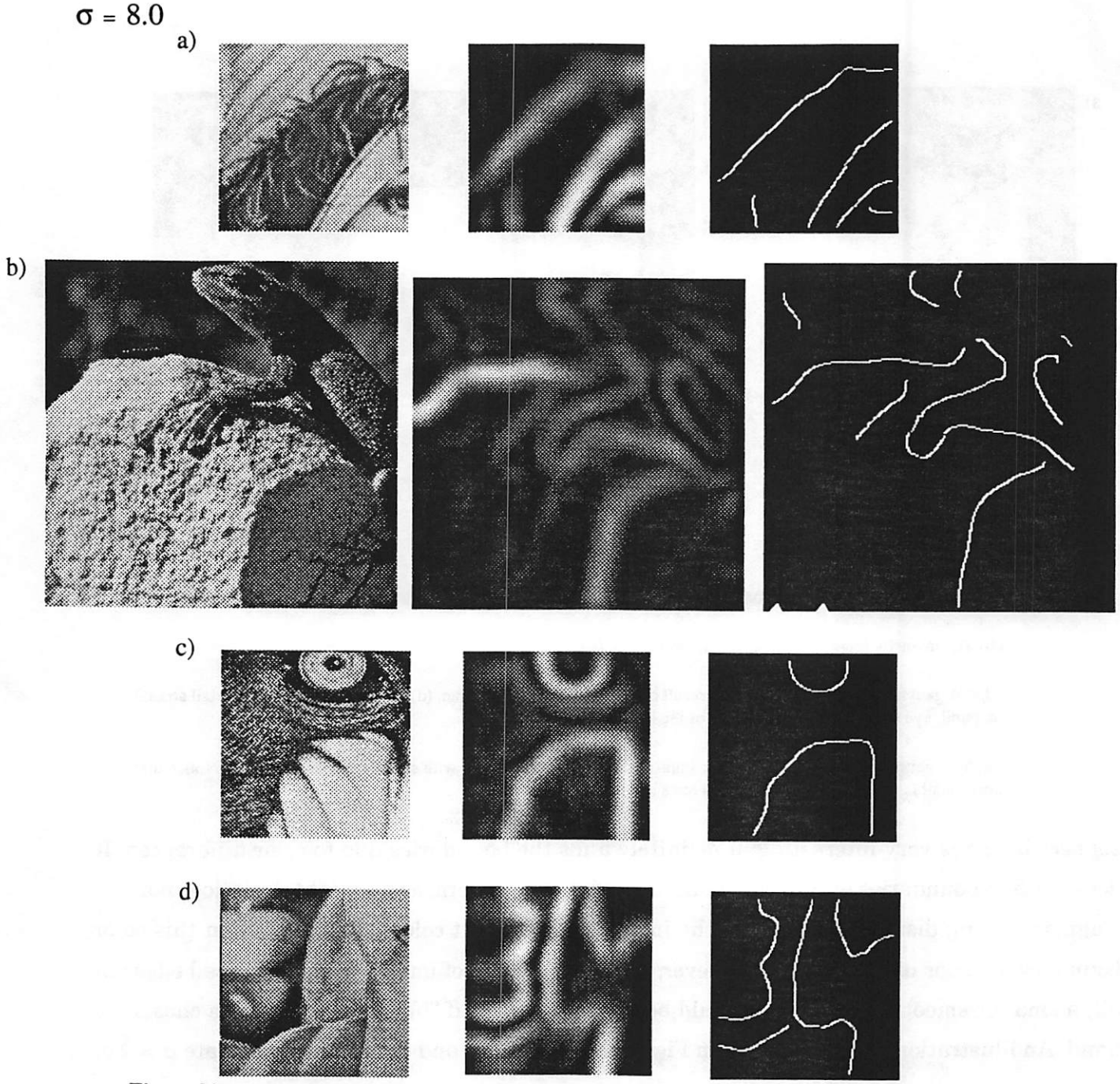


Figure 21: Effects of an excessively large smoothing gaussian. Shown are the R+G+B intensity images, the pooled magnitude result, and the segmentation result. Color photographs shown in Figure 18.

a) *Lenna* image.

b) *Lizard* image.

c) *Mandrill* image.

d) *Peppers* image. Notice that the regions are indeed separated by color. The upper left corner has three green peppers which are slightly highlighted in the magnitude image, but do not readily appear in the segmentation without tweaking of thresholds.

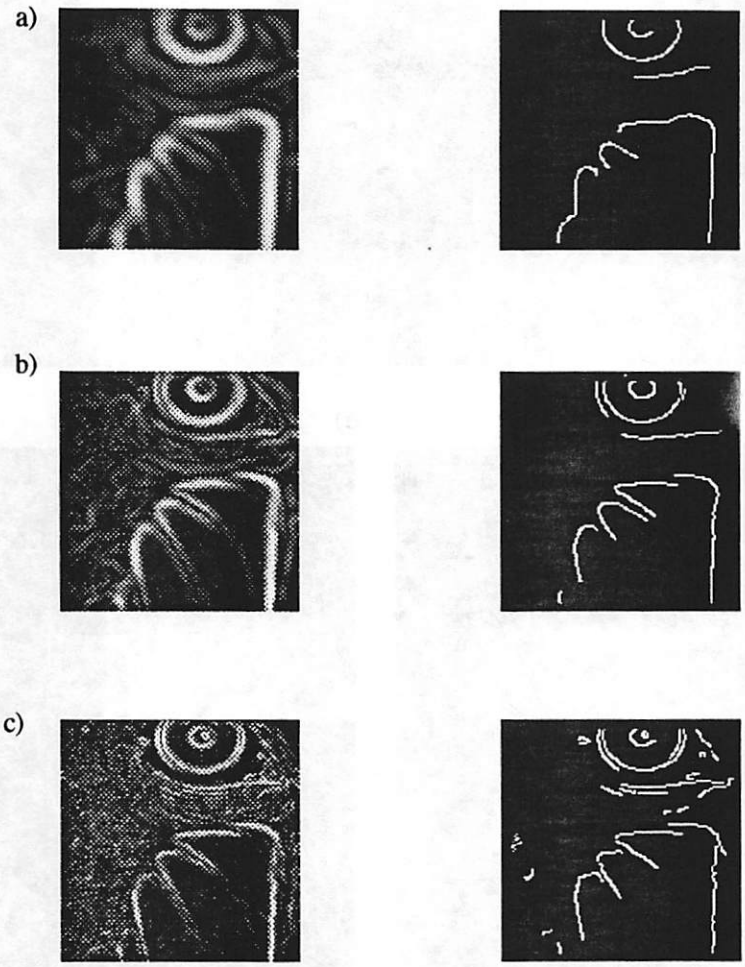


Figure 22: Refer to Figure 18c for color photograph.

a) $\sigma = 4.0$. Some improvement can already be seen over the previous $\sigma = 8.0$ results.

bl $\sigma = 2.0$. This scale finds many essential details in the mandrill image, including the pupil and the curves about the nose.

c) $\sigma = 1.0$. This scale is small enough to define details but with the side effect of picking up non-essential color edges. Note the rim of the eye and the speckle in the pupil.

COLOR SEGMENTATION RESULTS USING COLOR-OPPONENCY BANDS

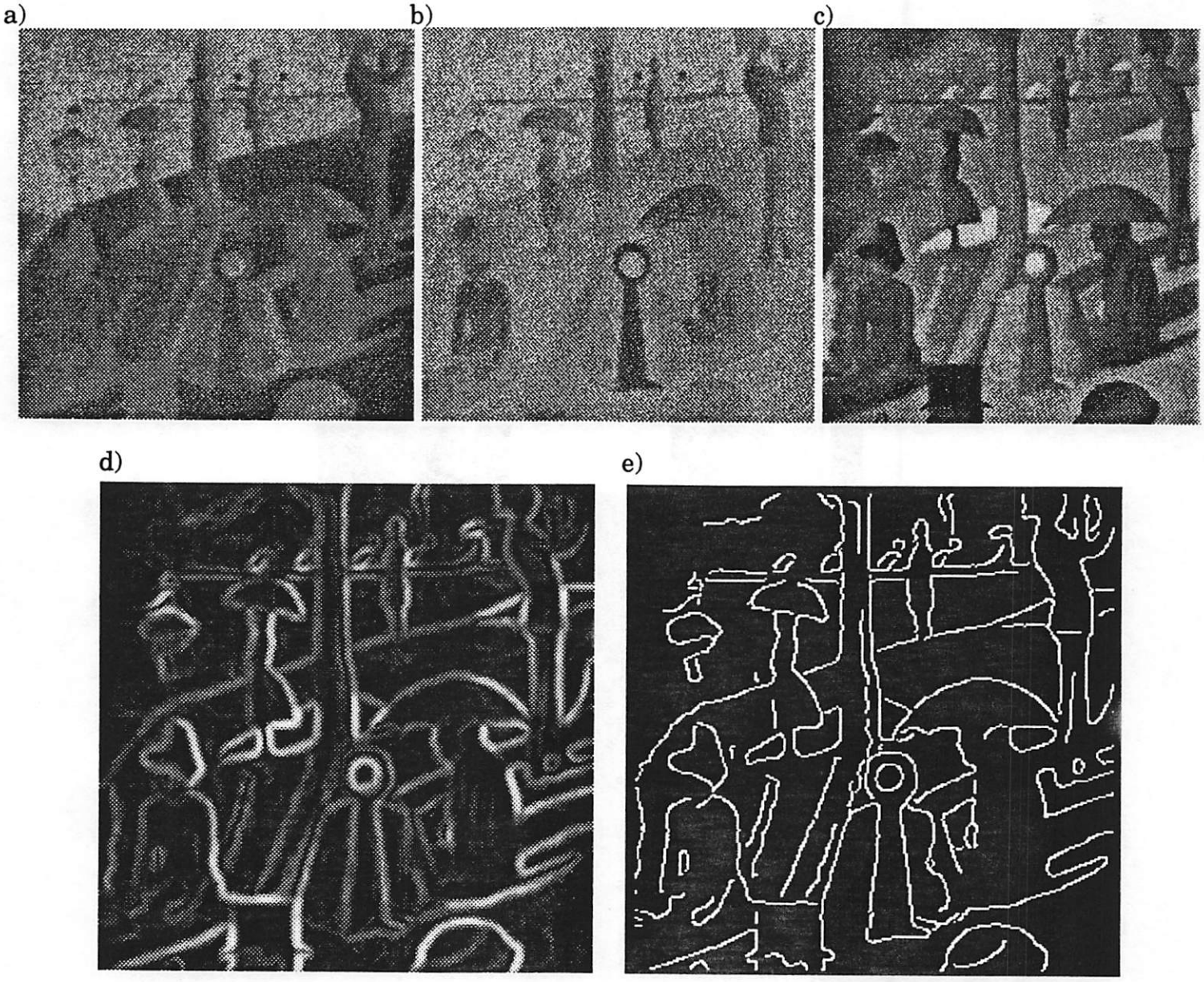


Figure 23: Color edge detection using opponent coding bands for "A Day in the Park" by Seurat. Photograph in Figure 18e.

a) The blue-yellow band. Positive (light) areas indicate higher blue, while negative (dark) areas indicate higher yellow.

b) The green-red band. Positive areas indicate higher green, while negative areas indicate higher red. Notice the alternating light and dark dots due to the pointillist style of the painting in which red and green colors are side-by-side.

c) The intensity image formed by R+G. Blue is a smaller proportion of human visual color cells.

d) The pooled magnitude image from the color edge detector.

e) The color segmentation produced by the color edge detector.

detection; it is commonly accepted that a value of $\sigma = 2$ may be used with impunity in most applications. Hence this value, $\sigma = 2$, is used in all subsequent experiments for this paper.

B. Variations on this process and Results

As with the texture model, the color algorithm can be modified in several ways and yet perform quite well. Here we show the effects of pooling at locations 0/∞ and I/∞ in the Color edge detector. As with the texture case, pooling prior to applying any of the three steps of the Canny operator is premature and gives relatively poor segmentation results. In the color case, the pooled image is too much like an intensity image to find several pertinent boundaries. Pooling at I, however, gives a good color segmentation (Figure 24), especially at the waterline.

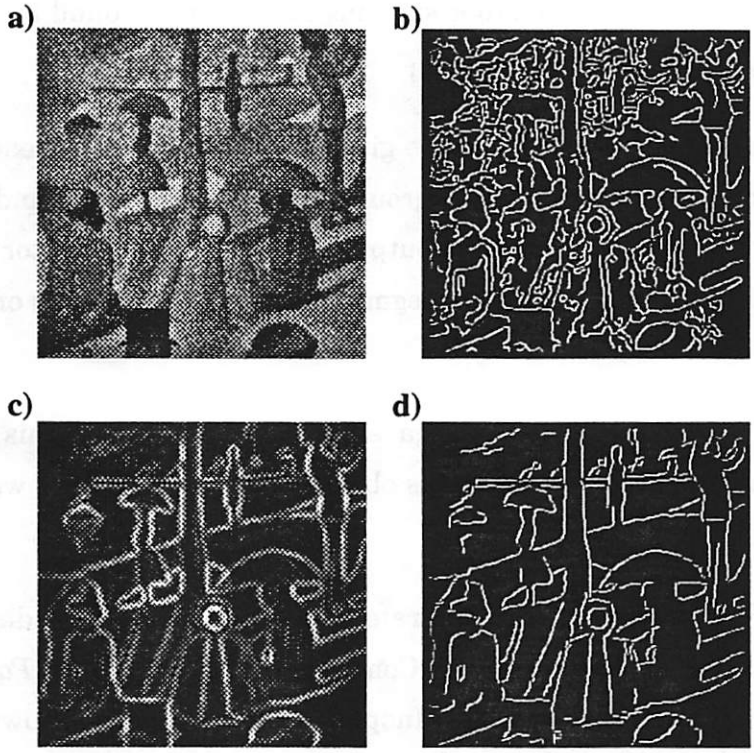


Figure 24: Comparison with pooling locations I and III: norm ∞ used in all cases.

a,b) 0/∞: The pooled image is simply max(R,G,B)(x,y) for each pixel point. The Canny edge operator is applied to this image. The pooling gives an image much like the intensity image, so the expectation is a similar, and not very good, segmentation.

c,d) I/∞: The x- and y- gradients on the three input bands are first pooled, and then the magnitude and segmentation are produced. This gives very good results.

C) Comparison With Other Two Boundary Detectors

Figure 25: *Equal-intensity color bands.* The intensity image produces no segmentation while the Color edge detector finds the boundaries. This is a simplified demonstration of the effectiveness of including a Color edge detector scheme in a generalized boundary detector.

Figures 26, 27: *A view of the Andes.* The billowing smoke is particularly interesting, as it appears to be multicolored across the shadowed and lit regions. True shadows have no color changes (images e and g), but do have intensity changes (image c).

Figure 28: *A Day in the Park.* This demonstrates the same ideas as Figure 25 with a more interesting image. There are several regions of the painting which differ in color but are of similar intensity. Notice especially the waterline; the intensity edge detector misses this edge while the color edge detector finds it easily. Several other such instances can be found by examining the two segmentation results, images b and d.

Figure 29: *Colored texture image +/os.* This image gives different results for each of the three boundary detectors. It is composed of a blue background and a yellow foreground, both of equal intensities. The intensity edge detector gives no output. The color edge detector finds the individual texture element outlines; but the texture segmentation algorithm finds only the texture boundary.

Figure 30: *The Old Mill.* The color edge detector gave the clearest edges for this image. This is somewhat unexpected, because the image contains obvious texture regions as well as shadowy changes in intensity.

Figure 31: *A Day in The Park,* revisited. Demonstrates the usefulness of including a color edge detector in a general boundary detection scheme. Comparison of *A Day in the Park*: Figure [a]: segmentation using texture boundary detector. Inspection of this image shows there are no clearly definable texture regions. Hence one would expect that the texture segmentation would not produce boundaries which closely match human detection of boundaries. This is not the case. While the main locations of the fisherman, the woman, and the tree are found, the boundaries are not very accurate; in addition, some relevant regions of the image have been missed.

C) Comparison With Other Two Boundary Detectors

Figure 25: *Equal-intensity color bands.* The intensity image produces no segmentation while the Color edge detector finds the boundaries. This is a simplified demonstration of the effectiveness of including a Color edge detector scheme in a generalized boundary detector.

Figures 26, 27: *A view of the Andes.* The billowing smoke is particularly interesting, as it appears to be multicolored across the shadowed and lit regions. True shadows have no color changes (images e and g), but do have intensity changes (image c).

Figure 28: *A Day in the Park.* This demonstrates the same ideas as Figure 25 with a more interesting image. There are several regions of the painting which differ in color but are of similar intensity. Notice especially the waterline; the intensity edge detector misses this edge while the color edge detector finds it easily. Several other such instances can be found by examining the two segmentation results, images b and d.

Figure 29: *Colored texture image +/os.* This image gives different results for each of the three boundary detectors. It is composed of a blue background and a yellow foreground, both of equal intensities. The intensity edge detector gives no output. The color edge detector finds the individual texture element outlines; but the texture segmentation algorithm finds only the texture boundary.

Figure 30: *The Old Mill.* The color edge detector gave the clearest edges for this image. This is somewhat unexpected, because the image contains obvious texture regions as well as shadowy changes in intensity.

Figure 31: *A Day in The Park,* revisited. Demonstrates the usefulness of including a color edge detector in a general boundary detection scheme. Comparison of *A Day in the Park*: Figure [a]: segmentation using texture boundary detector. Inspection of this image shows there are no clearly definable texture regions. Hence one would expect that the texture segmentation would not produce boundaries which closely match human detection of boundaries. This is not the case. While the main locations of the fisherman, the woman, and the tree are found, the boundaries are not very accurate; in addition, some relevant regions of the image have been missed.

C) Comparison With Other Two Boundary Detectors

Figure 25: *Equal-intensity color bands.* The intensity image produces no segmentation while the Color edge detector finds the boundaries. This is a simplified demonstration of the effectiveness of including a Color edge detector scheme in a generalized boundary detector.

Figures 26, 27: *A view of the Andes.* The billowing smoke is particularly interesting, as it appears to be multicolored across the shadowed and lit regions. True shadows have no color changes (images e and g), but do have intensity changes (image c).

Figure 28: *A Day in the Park.* This demonstrates the same ideas as Figure 25 with a more interesting image. There are several regions of the painting which differ in color but are of similar intensity. Notice especially the waterline; the intensity edge detector misses this edge while the color edge detector finds it easily. Several other such instances can be found by examining the two segmentation results, images b and d.

Figure 29: *Colored texture image +/os.* This image gives different results for each of the three boundary detectors. It is composed of a blue background and a yellow foreground, both of equal intensities. The intensity edge detector gives no output. The color edge detector finds the individual texture element outlines; but the texture segmentation algorithm finds only the texture boundary.

Figure 30: *The Old Mill.* The color edge detector gave the clearest edges for this image. This is somewhat unexpected, because the image contains obvious texture regions as well as shadowy changes in intensity.

Figure 31: *A Day in The Park,* revisited. Demonstrates the usefulness of including a color edge detector in a general boundary detection scheme. Comparison of *A Day in the Park*: Figure [a]: segmentation using texture boundary detector. Inspection of this image shows there are no clearly definable texture regions. Hence one would expect that the texture segmentation would not produce boundaries which closely match human detection of boundaries. This is not the case. While the main locations of the fisherman, the woman, and the tree are found, the boundaries are not very accurate; in addition, some relevant regions of the image have been missed.

C) Comparison With Other Two Boundary Detectors

Figure 25: *Equal-intensity color bands.* The intensity image produces no segmentation while the Color edge detector finds the boundaries. This is a simplified demonstration of the effectiveness of including a Color edge detector scheme in a generalized boundary detector.

Figures 26, 27: *A view of the Andes.* The billowing smoke is particularly interesting, as it appears to be multicolored across the shadowed and lit regions. True shadows have no color changes (images e and g), but do have intensity changes (image c).

Figure 28: *A Day in the Park.* This demonstrates the same ideas as Figure 25 with a more interesting image. There are several regions of the painting which differ in color but are of similar intensity. Notice especially the waterline; the intensity edge detector misses this edge while the color edge detector finds it easily. Several other such instances can be found by examining the two segmentation results, images b and d.

Figure 29: *Colored texture image +/os.* This image gives different results for each of the three boundary detectors. It is composed of a blue background and a yellow foreground, both of equal intensities. The intensity edge detector gives no output. The color edge detector finds the individual texture element outlines; but the texture segmentation algorithm finds only the texture boundary.

Figure 30: *The Old Mill.* The color edge detector gave the clearest edges for this image. This is somewhat unexpected, because the image contains obvious texture regions as well as shadowy changes in intensity.

Figure 31: *A Day in The Park,* revisited. Demonstrates the usefulness of including a color edge detector in a general boundary detection scheme. Comparison of *A Day in the Park*: Figure [a]: segmentation using texture boundary detector. Inspection of this image shows there are no clearly definable texture regions. Hence one would expect that the texture segmentation would not produce boundaries which closely match human detection of boundaries. This is not the case. While the main locations of the fisherman, the woman, and the tree are found, the boundaries are not very accurate; in addition, some relevant regions of the image have been missed.

(ANDES CONTINUED)
RESULTS: CANNY EDGE DETECTION AND COLOR EDGE DETECTION

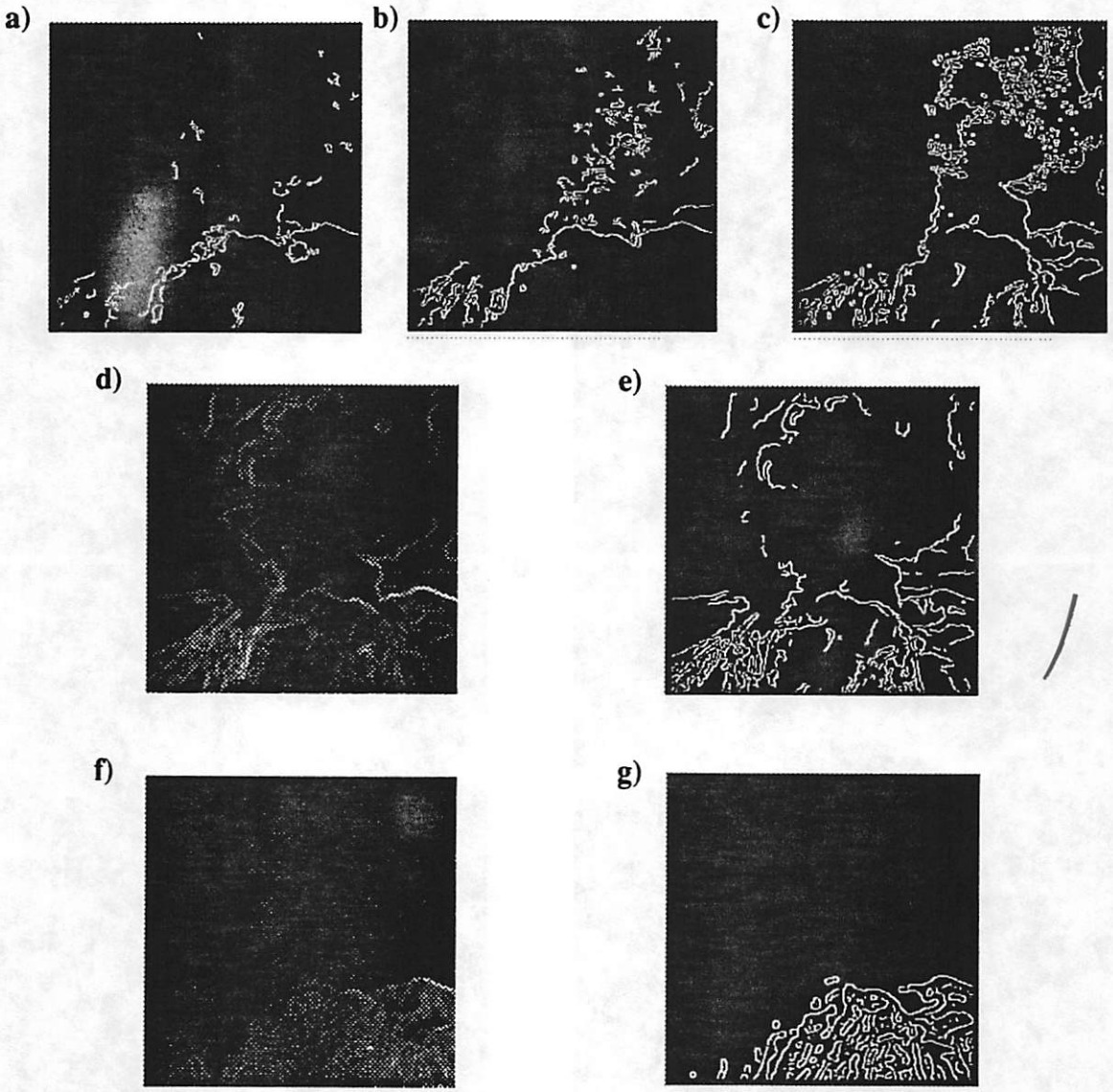


Figure 27:
CANNY EDGE DETECTOR RESULTS ON VARIOUS BANDS
The individual R, G, B bands do not produce edges; this is because of the general whiteness of the image, implying equal amounts of the three hues.
a) For Green-Red band, the only real dividing line is along the mountain edge.
b) For Blue-Yellow band, much of the difference is found in the billowing smoke.
c) This is the R+G intensity image; notice the delineation between lit and shadowed areas.

COLOR SEGMENTATION RESULTS ON VARIOUS BAND COMBINATIONS
d,e) The three R,G,B bands were used in the texture segmentation algorithm The boundaries around the smoke and the snowy mountains are found.
f,g) The three opponent-coding bands were used in the texture segmentation algorithm. Notice that areas which "look different" in color because of shadows are not found in this segmentation. In particular, note the smoke above the mountain; the lefthand-side of the smoke looks much brighter than the front-side, simply because the sun is lighting one side. No color changes, only intensity changes, occur across a shadow. The mountainous regions do have color changes between snowy and bare slopes.

Effectiveness of Color Edge Detector over Intensity Edge Detector for same-intensity color boundaries

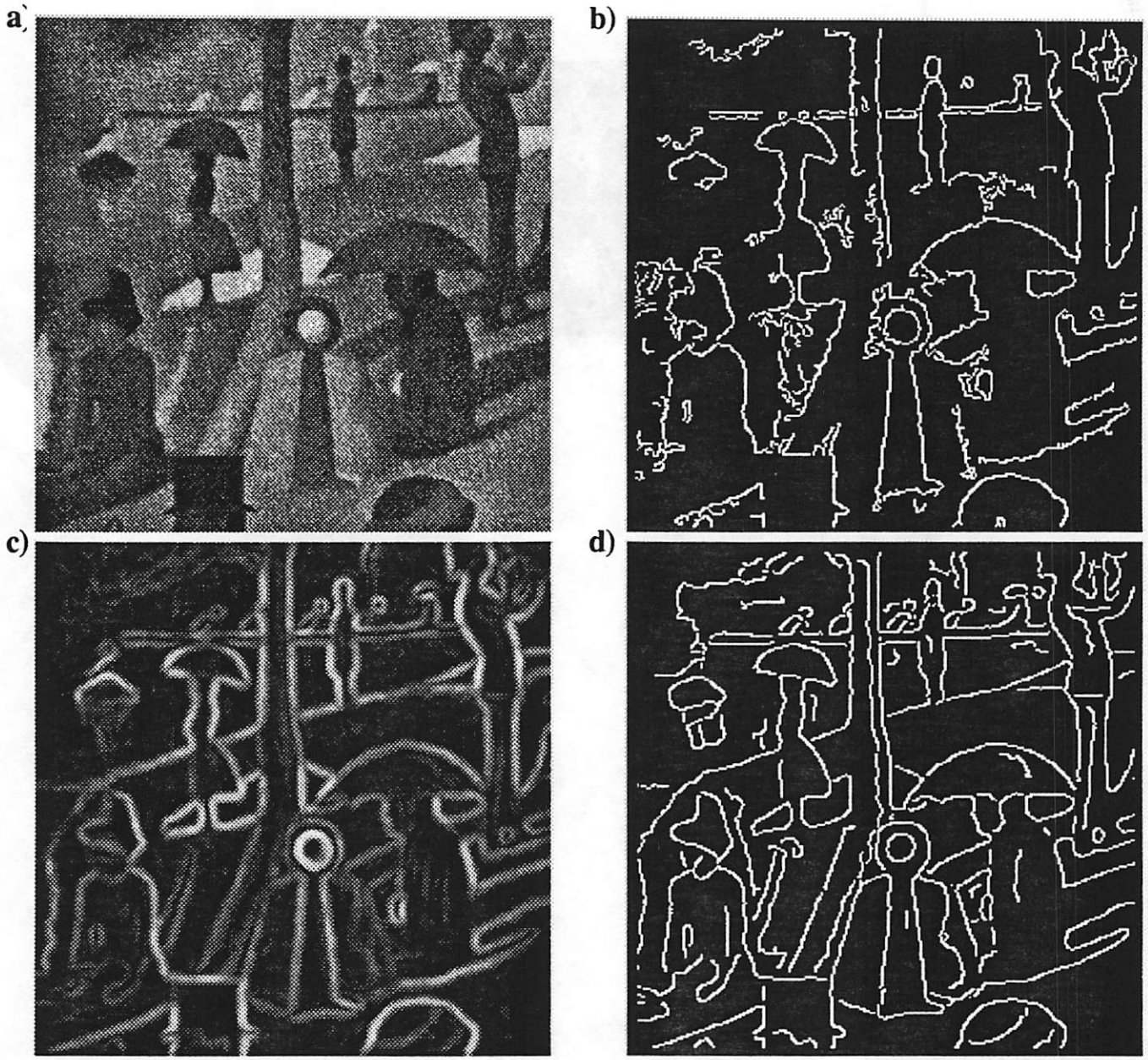


Figure 28: Refer to Figure 18e for color photograph.

a) Intensity image. Portion of "A Day in the Park." The intensity image is formed by adding the R,G, and B bands.

b) The output of the Canny Edge Operator. Notice only the edges due to sharp intensity changes are detected.

c) The pooled magnitude image from the Color Edge Detector.

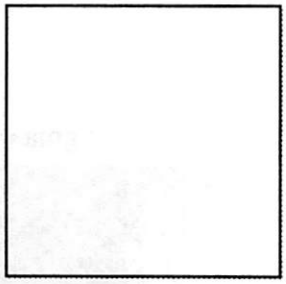
d) The output of the Color Edge Detector. Notice that perceived boundaries due to color differences are easily found by this detector; notice that the waterline, for instance, is found here whereas the intensity edge detector missed it since only the color, not the intensity, changes over that boundary. Several other instances are observable in comparing edge outputs b and d.

UNIQUENESS OF TEXTURE SEGMENTATION ON A COLOR IMAGE

Yellow features against blue background

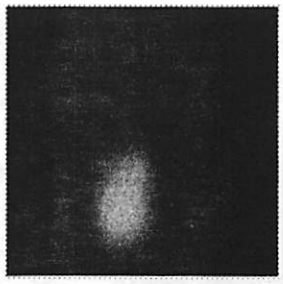
ORIGINAL IMAGE AND RESULTS OF: CANNY EDGE DETECTOR

a)



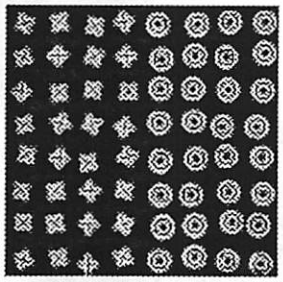
R+G+B intensity image

b)

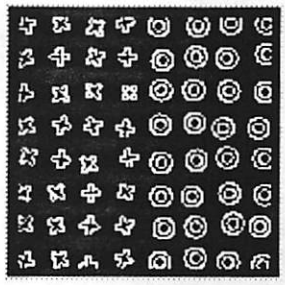


COLOR EDGE DETECTOR

c)

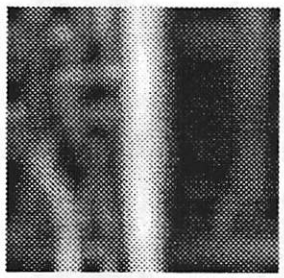


d)



TEXTURE BOUNDARY DETECTOR

e)



f)

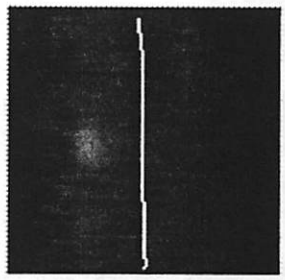


Figure 29: Comparison of intensity, texture, and color segmentation on an interesting case.

a) Texture/Color image: yellow features against blue background.This is the R+G+B image for which intensity values are constant for each pixel, hence appearing blank, with value 100 = white.

b) Canny Edge Detector finds nothing since there are no intensity changes.

c) Color Edge detector pooled magnitude result.

d) Color Edge Detector output; result is essentially as expected.

e) Texture Boundary Detector pooled magnitude result.

f) Texture Boundary Detector result. Only this algorithm found the texture boundary rather than the individual elements.

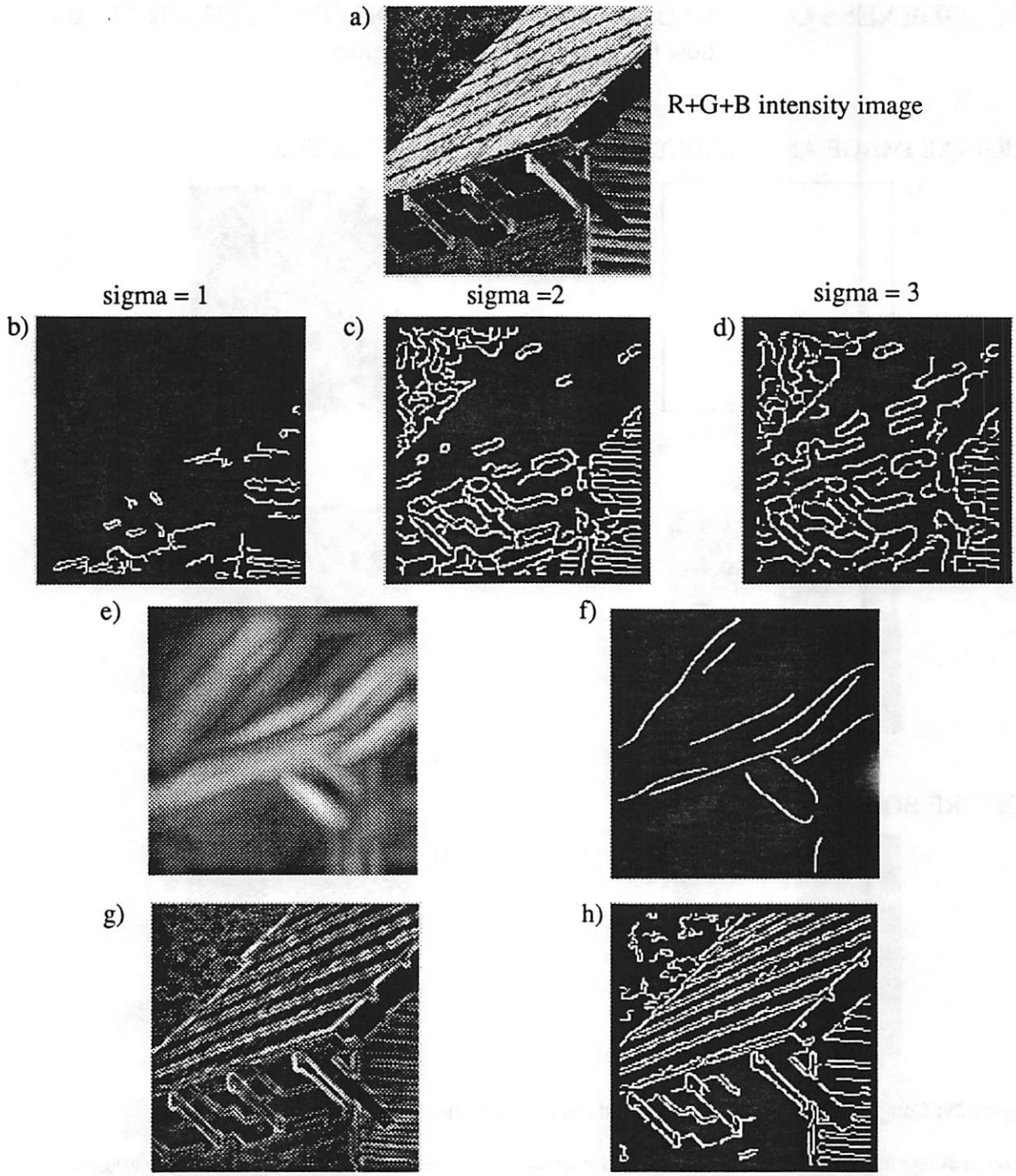


Figure 30: Refer to Figure 18g for color photograph.

a) Mill R+G+B intensity image
Canny Edge Detector result with b) $\sigma = 1.0$, c) $\sigma = 2.0$, d) $\sigma = 3.0$

e) Texture Boundary Detector pooled magnitude image

f) and Resulting Texture Segmentation

g) Color Edge Detector pooled magnitude image.
h) Color Edge Detector results.

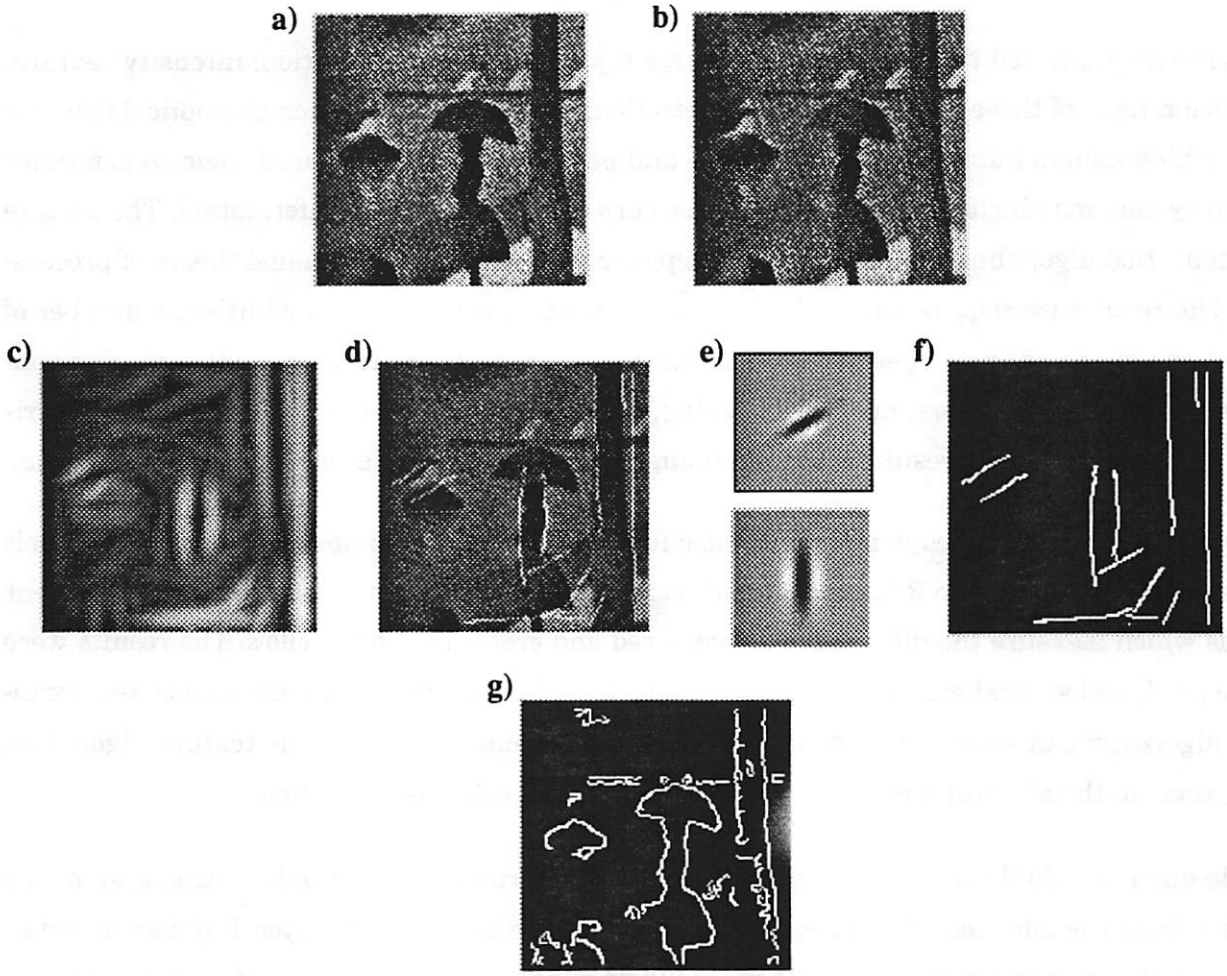


Figure 31: Refer to Figure 18e for color photograph.

a) The R+G+B intensity image, and

b) the R+G intensity image. There is less definition in the latter image.

c) The texture segmentation algorithm is applied to the image. The result shows that a few key outlines are found. This image does not have clearly delineated texture regions. The tree trunk, the woman's outline, the fisherman, and a small portion of the waterline are found, but little else.

d) Superposition of found boundaries with image.

e) Two of the filters leading to the segmentation: They match well with the "step edges" in intensity along the tree, the woman, and the fisherman.

f) The texture segmentation.

g) The Canny edge detection result for comparison.

VI. CONCLUSION

This report examined the development of three types of boundary detection: intensity, texture, and color. Each of these kinds of boundary detection, as well as stereo vision and optical flow, are cues which enable humans to interact with and perceive the world around them. A computer vision system may include some or all of these cues to gather essential information. The texture segmentation algorithm implemented in this paper followed the multichannel theory of processing. The results were quite successful for the sample textures tested. In addition, a number of variations in processing stages will still lead to good segmentation. These include replacing nonlinearity steps with others, such as squaring, and halfwave-rectification. Indeed, the experiments showed that good results could be obtained with multiple variations in a basic algorithm.

The color segmentation implementation also followed the multichannel model. Three channels were used, with either the R,G, and B bands serving as input images, or with the color-opponent bands which measure the differences between red and green, blue and yellow. The results were quite good, and several examples serve to demonstrate the additional benefits a color segmentation algorithm can provide to a boundary detection scheme. Similar to the texture algorithm, variations in the algorithm can lead to equally successful color segmentation.

While any individual visual cue provides a wealth of information, a computer vision system can benefit from the addition of an intensity, color, or texture boundary detector. Further considerations of the integration of these methods will help define a generalized boundary detector.

REFERENCES

1. P.J. Burt, "A Gradient Pyramid Basis for Pattern-Selective Image Fusion," *SID 92 Digest*, pp. 467-470, 1992.
2. R. W. Conners, C. W. McMilin, K. Lin, and R. E. Vasquez-Espinosa, "Identifying and Location Surface Defects in Wood: Part of an Automates Lumber Processing System," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 6, pp 573-583, 1983.
3. F. Farrokhnia, "Multi-Channel Filtering Techniques for Texture Segmentation and Surface Quality Inspection," Ph.d dissertation, Michigan State University, 1990.
4. J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986.
5. I. Fogel and D Sagi, "Gabor Filters as Texture Discriminator," *Biological Cybernetics*, Vol. 61, pp 103-113, 1989.
6. M.R. Turner, "Texture Discrimination by Gabor Functions," *Biological Cybernetics*, Vol. 55, pp 71-82, 1986.
7. J. Malik and P. Perona, "Preattentive Texture Discrimination with Early Vision Mechanisms," *Journal of the Optical Society of America*, Vol. 7, No. 5, pp 923-932, May 1990.
8. J. Malik and P. Perona, "A Computational Model of Texture Segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, pp 326-332, June 1989.
9. T. Kanade, "Image Understanding Research at CMU," *Image Understanding Workshop Proceedings*, Los Angeles, CA., pp 23 -25, Feb 1987.
10. B. Julesz, "Texton Gradients: The Texton Theory Revisited," *Biological Cybernetics*, Vol. 54, pp. 245-251, 1986.
11. N. Graham, J. Beck, and A. Sutter, "Nonlinear Processes in Spatial-frequency Channel Models of Perceived Texture Segregation: Effects of Sign and Amount of Contrast," Vision Research, Vol. 32, No. 4, pp 719-743, 1992.
12. D. Hubel and T. Weisel, "Receptive Fields, Binocular Interaction and Functional Architecture in the Cats's Visual Cortex," *Journal of Physiology*, London, Vol.160, pp 106-154, 1962.
13. F. Campbell and J. Robson, "Applications of Fourier Analysis to the Visibility of Gratings," *Journal of Physiology*, London, Vol. 197, pp 551-556, 1968. .
14. J. Atkinson and F. Campbell, "The effect of Phase on the Perception of Compound Gratings," *Vision Research*, Vol. . 14, pp 159-162, 1974.
15. H.R. Wilson and S. C. Giese, "Threshold Visibility of Frequency Gradient Patterns," *Vision Research*, Vol. 17, pp 1177-1189, 1977.
16. D. Marr and E. Hildreth, "Theory of Edge Detection," *Proceedings of the Royal Society of London*, Vol. 207, pp 187-217, 1980.
17. R. Young, "Gaussian Derivative Theory of Spatial Vision: Analysis of Cortical Cell Receptive Field Line-Weighting Profiles," *GM Research Report*, Warren,, MI. May, 1985.
18. E. Zrenner et al, "Color Perception," pp 164-185.
19. J. Malik, CS280 Computer Vision class notes, Spring 1992.
20. P. J. Burt and R. J. Kolczynski, "Enhanced Image Capture Through Fusion," *Proc. of ICCV 93 Conference*, Germany, pp. 173-182.
21. D. Marr, *Vision*. New York: W. H. Freeman and Company, 1982.
22. J. S. Lim, *Two-Dimensional Signal and Image Processing*. Englewood Cliffs, New Jersey: Prentice-Hall, 1990.

23. S. G. Mallat and S. Zhong, "Complete Signal Representation With Multiscale Edges," Technical Report No 483, Dept. of Computer Science, New York University, Dec. 1989.

24. A. K. Jain and F. Farrokhnia, "Unsupervised Segmentation Using Gabor Filters," *Int'l Conference on Systems, Man, and Cybernetics*, Los Angeles, CA., 1990.