A Vector Signal Processing Approach to Color

by

Kah-Kay Sung

Submitted to the Department of Electrical Engineering and Computer Science
on January 10, 1992 in partial fulfillment of the
requirements for the Degrees of
Electrical Engineer
and
Master of Science
in Electrical Engineering and Computer Science

ABSTRACT

Color is a useful visual cue for obtaining surface material information in a scene. Unfortunately, the surface color of an object can sometimes be different from its recorded color in an image because of optical effects like highlight and inter-body reflection. These effects often confuse traditional color algorithms that assume all surfaces in a scene to be perfectly lambertian.

This thesis adopts a signal processing approach to color vision. It represents surface color as a vector signal in a 3-dimensional color space, from which we extract region and boundary information. We immediately face two problems with this approach. The first, which is the same problem that traditional color algorithms face, is that highlight and inter-body reflection effects make image color different from surface color. We use a simple but effective method based on the theory of polarizing filters and electromagnetic wave reflection to correct for these effects on dielectric material surfaces in the scene. The second problem is to augment traditional scalar signal processing tools for 3-dimensional color vector signals. We linearize color by defining a notion of color similarity and difference, and use these measures in place of their traditional scalar counterparts in our vector signal processing algorithms.

The main contribution of this thesis is the systematic approach we propose that allows us to extend scalar signal processing tools and concepts into a multi-dimensional vector domain. We also discuss some ways of integrating surface color information with grey level intensity information.

Thesis Advisor: T. Poggio
Title: Uncas and Helen Whitaker Professor of Brain and Cognitive Sciences

Acknowledgments

Over the past three years, many people have contributed to the writing of this thesis in some way or other. I would like to take this opportunity to acknowledge them for their help and support.

First and foremost, I must thank Tomaso Poggio, my thesis supervisor, for the truly memorable learning experience I have had with him. I benefited much from his vast knowledge and intellectual generosity which gave rise to many of the ideas in this thesis. It was indeed magnanimous of him to have given me his best advice on many occasions, while still granting me the freedom to exercise my creativity. He also compensated for my reluctance to consult the literature.

Among my friends at the lab, many have contributed to the contents and completion of this thesis. In particular, I must thank the following: Brian Subirana, with whom I jointly worked to produce the material on Color Reference Frames presented in Chapter 7. We had many interesting discussions together and I really enjoyed working with him. I am also grateful for the help and advice he generously gave while I was preparing for my Oral Qualifying Examination. Tanveer Syeda, for the different ideas and approaches to Color Vision that we had been constantly exchanging. Each discussion with her was thought-provoking and enriching to me. Tao Alter, for all the time we spent talking about computer vision, MIT life and other things too numerous to mention. I always had company in my office, be it late at night or during the weekends. Sandy Wells, for the amazing library of technical books and journals he has been maintaining, among other things. He had all the references I needed on his shelves!

To my friend, Hui-Lin Lai, thank you for your constant prayers and concerns over my thesis writing. Also, thank you for reminding me that there is more to life than just getting a graduate degree. I treasure your friendship.

To my parents and sister, who have always unconditionally loved and supported me through all my endeavors, I love you all.

This report describes research done at the Artificial Intelligence Laboratory at MIT. This research is sponsored by the Artificial Intelligence Center of Hughes Aircraft Corporation (LJ90-074); by Office of Naval Research contract N00014–89–J–3139 under the DARPA Artificial Neural Network Technology Program; and by NSF and DARPA under contract 8814612-MIP. Support for the A.I. Laboratory's artificial intelligence research is provided by ONR contract N00014–91–J–4038.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Our ability to perceive color is indeed a great asset to our visual system. In everyday life, we often seem to rely heavily on color cues to help us detect things quickly. My blue research note book stands out prominently among the heaps of white paper scattered on my desk. A ripe yellow banana looks very different from an unripe green one in a bowl of fruit. I immediately release the accelerator of my car and depress the brake pedal when a bright red light suddenly flashes in front of me. I can afford to turn my attention away from the video game I am playing for a few seconds, because I do not see any green monsters lurking around on the display.

The extent to which we rely on color for vision becomes especially obvious when we try to identify and locate objects in cluttered black-and-white drawings or images. For example, a normal person finds it a lot harder to trace a route on a black-and-white photocopied highway map, than on a normal colored highway map. In the same way, a ripe yellow banana does not look very different from an unripe green one in a black-and-white photograph. Although we are still able to recognize objects and features based on shape, intensity and surface texture alone, our recognition task would, in most cases, be a lot simpler, faster and more pleasant, if we had the added advantage of sensing with color.

1.1 Color in Machine Vision

Machine vision can be described as a process that converts a large array of intensity or color data into a symbolic description of objects in the scene. Most natural scenes contain many different *physical features*, such as edges, surfaces and junctions. Early Vision operations try to identify and locate scene features that may be due to a number of different causes. How well a vision system finally describes a scene partly depends on how accurately it interprets the *cause* of each physical feature present in the scene, which in turn depends on how well its Early Vision modules can differentiate between the various *types* of observed features. Table 1.1 summarizes the types of

Discontinuity Type	*Physical Cause(s)*
Depth discontinuity Orientation discontinuity	Occlusion Surface Orientation
Optical flow discontinuity	Occlusion
Surface color discontinuity	Material
Texture discontinuity	Orientation Material
Intensity discontinuity only	Shadow Orientation Material

Table 1.1: Discontinuity Types and Physical Causes

discontinuities that can arise in a scene and their possible physical causes[1].

Because most features in a scene show up in an intensity image, intensity information is perhaps the most useful cue in machine vision. Unfortunately, intensity information alone does not always allow us to differentiate between the various types of physical effects that can occur in a scene. For example, most physical discontinuities in a scene show up as intensity discontinuities in the image, so intensity edge detection [Canny 83] is a useful tool for detecting and locating physical discontinuities in general. However, intensity edge detection alone cannot differentiate between the various types of scene discontinuities. That is, it does not provide us with additional information to determine whether an observed discontinuity is a material discontinuity, an illumination discontinuity or an orientation discontinuity. Furthermore, it is also true that physical features in a scene need not always show up in an intensity image. For example, when two isoluminant surfaces are placed side by side in a scene, we may not detect any discontinuities along their common boundary in the intensity image, even though there is definitely a physical discontinuity at the boundary.

In order to fully detect and interpret the different types of physical effects present in a scene, other early level processes are needed to provide the vision system with additional image understanding cues. For example, stereopsis combined with surface interpolation can identify depth and orientation discontinuities [Grimson 80]. Motion analysis is useful for finding occlusion boundaries and recovering 3D structure. [Hildreth 83]. Likewise, color [Rubin and Richards 81] [Rubin and Richards 84] and texture [Voorhees 87] [Voorhees and Poggio 87] modules can provide useful information for inferring surface material properties.

[1]Adapted from *Table 1.1* of [Voorhees 87]

1.2 Surface Spectral Reflectance and Image Irradiance

When dealing with color vision, it is important to distinguish clearly between the two notions: surface color (surface spectral reflectance) and image color (image irradiance). This section formally defines the two concepts in terms of the color image formation process. For the time being, we shall just confine our discussion to *lambertian* surfaces, whereby all the light that the surface reflects comes only from the interaction of incident light with the color pigments within the surface body (body reflection). In the next section, we shall talk about highlight or surface reflection on *non-lambertian* surfaces and how it affects the image color of the surfaces.

When light from an energy source falls on a surface, part of the incident light energy interacts with the color pigments within the surface body and gets reflected off into a color light sensor, such as a color camera or my eye. The lambertian *Color Image Irradiance Equation* describes the relationship between light at different wavelengths falling on the image plane (the *image irradiance*), and the physical properties of the lambertian surfaces being illuminated. In most natural situations, the image irradiance from a particular surface in the scene is proportional to the product of the amount of light incident on the surface (the *surface irradiance*) and the fraction of incident light that the surface reflects (the *surface spectral reflectance*). A simplified form of the Color Image Irradiance Equation may be written as follows:

$$\mathbf{I}(\lambda, \mathbf{r_i}) = \rho(\lambda, \mathbf{r_s})\mathbf{F_b}(\mathbf{v}, \mathbf{n}, \mathbf{s})\mathbf{E}^*(\lambda, \mathbf{n}, \mathbf{s}), \tag{1.1}$$

where λ is wavelength; $\mathbf{r_s}$ is the spatial coordinate of the surface patch being imaged; $\mathbf{r_i}$ is the coordinate of the point on the image plane onto which $\mathbf{r_s}$ projects; $\mathbf{n}$ is the unit surface normal vector at the point $\mathbf{r_s}$; $\mathbf{v}$ is the unit vector in the viewer's direction; $\mathbf{s}$ is the unit vector in the illuminant direction; $\mathbf{I}$ is the image irradiance; $\mathbf{E}^*$ is the surface irradiance; ρ is the surface spectral reflectance function and $\mathbf{F_b}$ is a scaling factor that depends on viewing geometry.

Surface irradiance ($\mathbf{E}^*(\lambda, \mathbf{n}, \mathbf{s})$) is a function of wavelength, λ, and it basically describes the intensity and color of the light source illuminating the scene. White light, for example, has an irradiance value that is roughly constant for all wavelengths in the visible spectrum. Red light has large irradiance values at long wavelengths in the visible light spectrum (near $650nm$) and small irradiance values at short wavelengths (near $420nm$).

Surface spectral reflectance ($\rho(\lambda, \mathbf{r_s})$) describes the proportion of incident light that a surface re-radiates at each spectral wavelength. This surface property is invariant under different lighting conditions, and it depends only on the material that makes up the surface. Hence, when we refer to the *surface color* of an object, we are in fact referring to the *spectral reflectance values* of its surfaces. For example, a green pear has a surface reflectance that peaks near the middle of the visible light spectrum. A white sheet of paper has a surface reflectance that is approximately constant for all visible light wavelengths.

Image irradiance ($\mathbf{I}(\lambda, \mathbf{r_i})$)is the amount of light energy that falls onto the image plane at each wavelength. This is the measurable quantity that forms images in visual sensors. Under white light illumination, the image irradiance from a surface is approximately proportional to its surface spectral reflectance, or to its surface color, as can be inferred from the Color Image Irradiance Equation (1.1). That is to say, the color that a sensor registers for an object in white light is a close approximation to the object's true surface color. Under colored lighting conditions, we usually get significant image irradiance components at fewer spectral wavelengths. At these spectral wavelengths, both the surface irradiance (illumination color) on the illuminated surface and the surface spectral reflectance (surface color) of the surface must have significantly large values. A yellow banana thus appears green in green light and black in blue light.

In general, image irradiance values cannot be taken as a scaled approximation for surface color when imaging in colored light. However, for a sufficiently wide range of colored lighting conditions in a lambertian environment, it is usually true that a patch of uniform spectral reflectance in the scene gives rise to a patch of uniform image irradiance in the image, and a spectral reflectance boundary in the scene gives rise to an image irradiance discontinuity in the image. In other words, image irradiance uniformity in a lambertian image often implies surface spectral reflectance uniformity in the scene and image irradiance discontinuities often imply surface spectral reflectance discontinuities.

1.3 Highlight and Image Color

For most real surfaces, the lambertian Image Irradiance Equation 1.1 gives only an approximate description of the color image formation process. For example, in dielectric materials like paints, plastics, paper, textiles and ceramics, we usually see the reflected light as being composed of two distinct colors. The first component results from the interaction of the illuminant with the color pigments in the material body. This component of reflected light is commonly known as the *body component* and its color depends both on the color of the illuminant and the surface spectral reflectance values of the illuminated body. The lambertian Image Irradiance Equation presented in the previous section accounts only for this component of the reflected light.

When incident light passes through air and hits a dielectric surface, it encounters a junction of materials with different refractive indices. The laws of physics and electromagnetism compels that some percentage of the incident light gets reflected directly off the material junction without interacting with the color pigments of the material body. This component of reflected light makes up the *highlight* that we often see on many real surfaces. It is also commonly known as *surface reflection* [Klinker 88], *interface reflection* or *Fresnel reflection* [Wolff 89] in contemporary vision literature. Highlight takes the color of the incident light source since it is simply incident light that is bounced off the illuminated object's surface in an "unperturbed" fashion, like in a mirror.

With the inclusion of *highlight* in the image formation process, the Color Image Irradiance

Equation for non-lambertian surfaces now contains two terms, a *body reflection* term and a *surface reflection* term:

$$\mathbf{I}(\lambda, \mathbf{r_i}) = \rho(\lambda, \mathbf{r_s})\mathbf{F_b}(\mathbf{v}, \mathbf{n}, \mathbf{s})\mathbf{E}^*(\lambda, \mathbf{n}, \mathbf{s}) + \mathbf{F_s}(\mathbf{v}, \mathbf{n}, \mathbf{s})\mathbf{E}^*(\lambda, \mathbf{n}, \mathbf{s}), \tag{1.2}$$

Here, F_s is a scaling factor for *surface reflection* that depends on viewing geometry and F_b is a scaling factor for *body reflection* that also depends on viewing geometry. The other symbols in equation 1.2 are as defined in equation 1.1.

Notice that with this new reflectance model for non-lambertian materials, *image irradiance* can now be described as a mixture of light reflected from the material surface and light reflected from within the material body. Since F_s, the scaling factor for *surface reflection*, is usually very sensitive to viewing geometry and very different in form from F_b, the proportion of highlight and body reflection that a sensor measures can be very different even for relatively nearby points on the same surface. Since the color composition of highlight is, in general, different from the color composition of body reflection, it is therefore possible that points on the same non-lambertian surface in a scene can give rise to image irradiance readings of different color composition. In other words, where there is highlight, we can no longer assume, as we did in the previous section, that a patch of uniform surface color in the scene will give rise to a patch of uniform color (hue[2]) in the image.

1.4 Color as an Ill-Posed Problem

What makes color vision a difficult problem like most other early vision processes ? By trying to recover information about a 3D world from 2D color images, color vision falls into a general class of ill-posed problems [Berter Poggio and Torre 86] known as inverse optics. These problems are inherently under-constrained and have no unique solutions. Let us see why this is so.

If our goal in color vision is to recover the true surface color of an object from its color image, then we have to compute the surface spectral reflectance function (ρ) of the object from its image irradiance function (I). This is an under-constrained problem because even if all the surfaces in the scene were lambertian and equation 1.1 holds, the image irradiance for a point in the scene still depends on two physical quantities, namely the surface spectral reflectance values (ρ) and the surface irradiance distribution (E^*) at the point. Knowing the surface irradiance distribution in the scene still does not help us solve for $\rho(\lambda, r_s)$ completely because there is another unknown scaling factor, F_b, in the Image Irradiance Equation that depends on viewing geometry. In fact, all that we can recover from I in a bottom up fashion is a scaled factor of $\rho(\lambda, r_s)$ at best, unless we can force ourselves to make other assumptions about the scene and the real world.

If our goal in color vision is just to find uniform *surface color* regions or boundaries in an image

[2]This term has not been formally introduced. For now, it is sufficient to think of *hue* as the normalized λ spectrum of a color signal.

without recovering true surface color, then we have a simpler task than the one we had before. The task only requires us to detect uniformity or discontinuities in surface spectral reflectance values (ρ) by examining image irradiance values (I). In fact, for lambertian scenes under uniform or smoothly varying lighting conditions, this problem actually becomes well-posed because from equation 1.1, changes in image irradiance, I, in the image can only be brought about by changes in surface spectral reflectance, ρ, in the scene. In other words, to find uniform surface color regions or boundaries in a lambertian and uniformly lit scene, all that we have to do is to find uniform color regions or boundaries in the image.

Unfortunately, most natural occurring scenes contain non-lambertian surfaces like dielectrics and metals, so equation 1.1 is a poor approximation to the image formation process. If we use equation 1.2 instead of equation 1.1 to account for highlights on non-lambertian surfaces, then we face an under-constrained problem again, having to compute ρ from I. This is because the image irradiance equation now contains two terms, a body reflection term and a surface reflection term, and we cannot determine the contribution of the body reflection term alone in the image by just examining the values of I.

1.5 Why Traditional Color Algorithms Fail

In the past, researchers in color vision have been designing algorithms that perform grouping and segmentation operations on images using image color. We have seen in Section 1.3 that this is in fact a wrong formulation for most surface color problems. Image color, or image irradiance, depends on both the surface and body components of reflection, so its readings may vary considerably even over a patch of material with relatively uniform surface color. Body reflection, on the other hand, remains relatively constant in color where surface color is uniform, so it provides us with a much better cue for analyzing material composition of scenes. Unfortunately, we have seen that extracting the body component of image color from image irradiance readings is a difficult problem and this problem is usually conveniently overlooked in traditional color algorithms. For example, Lightness Algorithms by Land [Land 59], Horn [Horn 74], Blake [Blake 85] and Hurlbert [Hurlbert 89] assume that surface spectral reflectance can be completely specified by image irradiance in each of 3 chromatic channels when computing lightness ratios for surfaces. Since the assumption fails where there are highlights in the scene, these algorithms only work well for recovering surface spectral reflectance values of flat Mondrains. Traditional feature histogram based color segmentation schemes [Ohlander 76] [Shafer and Kanade 82] that group image pixels using image color similarity measures also perform badly in the presence of strong highlight because they ignore the effects of highlight on image irradiance.

It was not until recently that secondary imaging effects like highlights and inter-body reflection were seriously being studied and modeled as components of the image formation process. Klinker, Shafer and Kanade [Klinker Shafer and Kanade 88b] proposed a *dichromatic reflection model* that treats reflected light from di-electric surfaces as a vector sum of a body reflection component and a surface reflection component, as in equation 1.2. They also demonstrated a technique for separating highlight from body reflection in an image by analyzing color histograms of surfaces. Bajcsy, Lee and Leonardis [Bajcsy Lee and Leonardis 89] extended the dichromatic reflection model to accommodate minor changes in hue that are brought about by inter-body reflections. Although a number of surface reflectance models now exist that can better describe the appearance of non-lambertian scenes, these models are usually too complex to be used in traditional color algorithms because of their multiple image irradiance terms.

1.6 Goals of this Thesis

Our primary goal in this thesis is to investigate how image processing concepts that are defined for scalar signals can be extended to process color data.

Current image processing techniques like filtering, edge detection, region growing and surface reconstruction are relatively well established and favorably tested concepts in early vision. Often, a direct application of these algorithms only work for scalar signals, such as grey-level intensity readings or stereo depth maps. Color data, on the other hand, is usually encoded as triplets of grey-level intensities in 3 chromatic channels. Since each pixel in a color image has 3 scalar chromatic components, it seems only reasonable that we should treat color as a vector quantity in some 3 dimensional space. This means that in order to apply the same scalar image processing concepts to color data, we first have to understand and derive their analogous notions in a multi-dimensional space.

Most early vision systems today process color images by working separately in the 3 chromatic channels. Since each chromatic channel is an array of scalar values, we can apply existing scalar image processing algorithms directly to the individual channels. The results from the individual channels can then be combined, if necessary, to form an overall result for the operation. We see three problems with this divided approach:

1. The problem of coupling information from separate channels has always been an issue when dealing with multiple sources of related data. In a Markov Random Field (MRF) formulation, line processes [Gamble and Poggio 87] can be used to integrate region based information from separate visual cues. In the case of color where there are multiple related chromatic channels, Wright's [Wright 89] color segmentation scheme, for example, treats each color channel as a separate visual cue and integrates discontinuities in the channels using line processes. Ideally, since color is a primitive attribute of early vision, we want to avoid the added complexity of having to explicitly integrate information from its separate channels if possible.

2. When some scalar operation is separately applied to each chromatic channel in a color image and the results recombined as color data, we may not get the same intuitive result as what

we might expect if an analogous concept of the same operation were applied to *color* values. The following two examples expound on what we mean.

In Chapter 5, we argue that color does not change across a pure intensity boundary, such as a shadow or an orientation edge. So, color edge detection algorithms should not detect edges at pure intensity boundaries. However, when scalar edge detection algorithms are directly applied to the separate color channels of an image, we usually still find boundarys in all 3 channels where there are just pure intensity edges.

In Chapter 4, we see that when scalar smoothing algorithms are directly applied to the separate chromatic channels of a color image, colors from brighter pixels tend to get weighted more in the smoothing process than colors from dimmer pixels. Unless we are concerned about the *discretization noise* at dimmer pixels, this is probably not what we want when we think about averaging *color* values.

3. There are some scalar concepts that do not have apparent *color* analogies unless we reason about them in a multi-dimensional domain. For example, Chapter 4 introduces the concept of taking *color medians* for *median window filtering*. We see in Chapter 4 that we cannot conceptualize the notion of *median color* by working separately in the 3 chromatic channels of a color sample.

We demonstrate, in this thesis, that the three problems identified above can be naturally overcome when we deal with *color* as an entity and not as scalar values in three separate chromatic channels. We also show how a large number of existing scalar image processing concepts can be easily extended to work in the color domain, through a systematic but simple *linearization* method.

Our other goal in this thesis is to demonstrate an external technique for removing highlight and inter-body reflections from images. Since image irradiance is just a poor approximation of body reflection under non-lambertian imaging conditions, our color algorithms will still face the same surface reflection problems that traditional color algorithms had, unless these effects can be reliably and substantially reduced in the images we process. Our proposed solution uses a linear polarizing filter to attenuate surface components of reflected light from the scene before they enter the color sensor [Wolff 89]. We show that under a wide range of viewing conditions, the light that our sensor receives through a polarizing filter is a much better approximation to the body component of reflected light.

1.7 Method of Investigation

This thesis maintains a qualitative treatment of the color concepts that we are trying to develop. Scalar analogues to most of our color algorithms can be found in contemporary image processing literature. Since most of these analogues have been widely studied and well tested in the scalar domain, their quantitative issues like computational complexity and other performance measures are already well known. Our main purpose here is to show that we can derive similar concepts between single-channel data processing and multi-channel data processing, if an appropriate representation is chosen for the multi-channel signal. Although our research is done within the framework of surface color, the results from this study should also be applicable to other forms of multi-dimensional data, in particular motion fields.

In the following chapters, we demonstrate how our color algorithms work by showing and analyzing the results they produce on a set of real and synthetic images. Where possible, qualitative comparisons are made between the results we obtain and the results obtained by various other methods of working with color.

1.8 Overview and Contributions

The rest of this thesis is organized as follows: Chapter 2 describes the polarizer technique for removing highlight, inter-body reflections and other secondary effects from images. Chapter 3 chooses a representation for color signals and defines the notions of color uniformity and color difference in this representation scheme. Chapter 4 addresses the issue of noise in color images and describes methods for dealing with noise. Chapter 5 looks at the implementation of an edge detection algorithm for color, while Chapters 6 and 7 deal with the notion of color uniformity and present two methods for finding color region features in images.

The main contributions of this thesis are:

1. A vector representation approach to color that naturally integrates information from a color signal's separate chromatic channels. The approach appears to work for other forms of multi-dimensional data as well, in particular motion fields.

2. A surface reflection removal technique based on polarization properties of reflected light (see also [Wolff 89]).

3. Our scalar notions of color similarity and difference, and the general "linearizing" ideas we employ for extending certain useful scalar signal concepts into the color domain (see Chapters 3, 4 and 7).

4. Our detailed quantitative analysis of color noise, whose basic form is a Rayleigh distribution and not a Gaussian distribution.

5. Our color domain extensions of some existing noise reduction, boundary detection, region finding and salient reference frame computation algorithms.

6. The idea of introducing a dynamic line process into an existing local color averaging scheme [Hurlbert and Poggio 88] so that it preserves the sharpness of color boundaries better.

7. A feature integration algorithm that aligns color image boundaries with grey-level intensity boundaries and discards spurious color edge fragments due to noise.

8. A statistical approach for selecting and interpreting segmentation thresholds and free parameters, which we demonstrate in one of our region finding algorithms.

9. A highly versatile reference frame algorithm that operates directly on color image data, without having to first compute color regions or boundaries.

10. The idea of using reference frames as a source of global image information for region finding.

Chapter 2

Recovering Lambertian Components of Reflection

In most natural occurring scenes, a patch of uniform surface color, or uniform surface spectral reflectance, usually corresponds to some physical entity in the viewer's environment. A ripe banana, for example, gives rise to a patch of yellowish surface spectral reflectance in a bowl of fruit, while an unripe banana gives rise to a patch of green surface spectral reflectance. Similarly, surface color discontinuities in a scene normally arise from object boundaries or material boundaries in the environment, like the surface color boundaries that a blue note book creates with a brown wooden table. Although human beings perform remarkably well at isolating uniform surface color patches even in the presence of strong highlight, machine vision systems today still tend to get easily confused by the different causes of color variations that can occur in a scene. Strong highlights on an object can easily be misinterpreted as a separate region of different surface color or a region of higher albedo. The first step for performing machine vision operations on surface color should therefore be one of recovering *body* or *lambertian* reflection from image irradiance.

This chapter describes a method of recovering body reflection from image irradiance that exploits electromagnetic polarization properties of reflected light. The method was first used by Wolff [Wolff 89] to separate surface and body components of reflection from objects. In his experiments, Wolff showed that highlight can be totally removed from a surface by first computing an average *Fresnel Ratio* for all points on the surface and then using the ratio to estimate highlight strength at each pixel. For the purpose of this thesis, a simpler version of the algorithm that does not compute *Fresnel Ratios* still gives body reflection estimates that are sufficiently accurate for our color processes. Although Wolff also made use of polarization effects for other tasks like stereo matching and material classification in scenes [Wolff 89], its greatest use still lies in body reflection recovery where it reliably reduces secondary imaging effects under a wide range of viewing geometries.

2.1 Model Based Separation of Reflection Components

Recent studies by Klinker, Shafer and Kanade [Klinker Shafer and Kanade 88b] have shown that dielectric surface reflection can be adequately described by a *Dichromatic Reflection Model*, and that the model can be used to help split reflected light into its surface and body reflection components. The model states that if color is represented as a vector in a three dimensional space spanned by its chromatic channels, then the color of light reflected from a dielectric surface becomes a linear combination of two color vectors. A *matte* vector whose length depends on the magnitude of body reflection shows the amount of body reflection falling on the color sensor. The color of this vector depends on the material properties of the illuminated surface. A *highlight* vector whose length depends on the strength of highlight coming from the object accounts for the amount of surface or interface reflection falling on the image. The color of the *highlight* vector is assumed to be the same as the color of the illuminating light source.

To split color pixels into their body and surface reflection components, Klinker, Shafer and Kanade use the *Dichromatic Reflection Model* to determine the *matte* and *highlight* vectors of pixels on a surface. The algorithm maps all color pixels from a region onto the 3D color space, where according to the model, the generated cluster would take the shape of a skewed-T. The two linear sections of the skewed-T cluster point in the directions of the region's *matte* and *highlight* vectors. Pixels that map onto the *matte* linear section are classified as matte pixels with negligible highlight components. The color of these pixels in the image is taken to be their body reflection color. Pixels that map onto the highlight linear section are considered highlight pixels whose deviations from the *matte* line are caused by the presence of highlight. The strength of highlight at these pixels is proportional to their color distance from the *matte* cluster, in the direction of the highlight vector. The body reflection color at each pixel can be recovered by projecting the pixel's color along the *highlight* vector onto the *matte* linear subsection.

Klinker, Shafer and Kanade demonstrated their dichromatic model-based algorithm on a number of dielectric images by splitting each of them into an image without highlight and an image of just highlight. Although the algorithm works well for scenes of dielectric objects, the technique fails if the illuminant color gets very close to the color the object. This is because the two linear branches of the skewed-T cluster become almost parallel under this condition and are no longer distinguishable as two linear sections. Also, since the algorithm works by analyzing color variations of regions over a relatively large local neighbourhood, the algorithm would also fail if there are multiple different colored light sources illuminating the same object. The skewed-T cluster hypothesis also does not hold in this case, because each colored light source will give rise to its own highlight branch for the region in the color histogram.

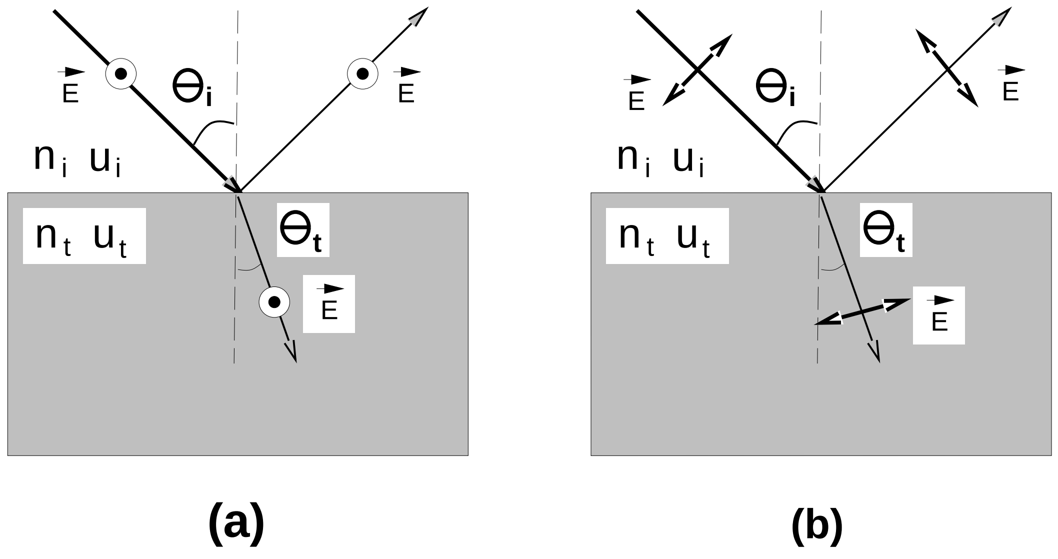


Figure 2.1: (a) An incident light wave with polarization perpendicular to the plane of reflection. (b) An incident light wave with polarization parallel to the plane of reflection.

2.2 Electromagnetic Properties of Surface Reflection

The separation technique that we describe in this thesis is based on the physical property that body reflected light from dielectric surfaces is usually unpolarized, whereas highlight and other forms of surface reflection are usually strongly polarized. Maxwell's electromagnetic field equations and the physics of electromagnetic reflection best explain these polarization effects.

Light incident on a material surface gets partially reflected at the material interface and partly transmitted through the material junction. The transmitted component interacts with color pigments embedded in the material body, giving rise to *body reflection* or *lambertian reflection*. Since color pigments are usually randomly scattered in the material body, *body reflection* emerges in a randomly polarized fashion. For surface reflected light, the electromagnetic fields on both sides of the material junction and across the material junction must satisfy Maxwell's equations and their constitutive relations. This includes having tangential electric field components and perpendicular magnetic flux densities that are continuous across the boundary. The reflected and transmitted components of an incident wave can be derived by *phase matching* the tangential electric and perpendicular magnetic field components so that they are continuous across the material junction. For an incident light wave that is polarized perpendicular[1] to the plane of reflection (see Figure 2.1(a)), we get the following expressions for the magnitudes of the reflected and transmitted *electric* waves after phase matching:

[1]By convention, the *polarization direction* of a light wave is the direction of its electric field.

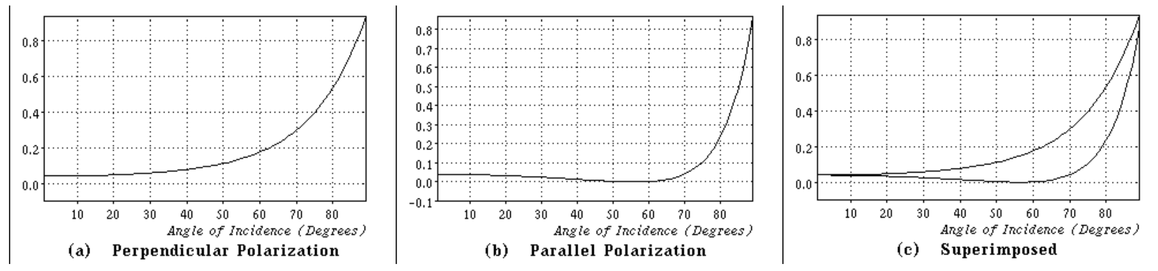


Figure 2.2: Squared coefficients of reflection as a function of θ_i for $n_t = 1.5$. (a) $R_\perp^2$. (b) $R_{||}^2$. (c) $R_\perp^2$ and $R_{||}^2$ superimposed.

$$R_\perp = \frac{\frac{n_i}{\mu_i} cos\theta_i - \frac{n_t}{\mu_t} cos\theta_t}{\frac{n_i}{\mu_i} cos\theta_i + \frac{n_t}{\mu_t} cos\theta_t}, \tag{2.1}$$

$$T_\perp = \frac{2\frac{n_i}{\mu_i} cos\theta_i}{\frac{n_i}{\mu_i} cos\theta_i + \frac{n_t}{\mu_t} cos\theta_t}, \tag{2.2}$$

Here, $R_\perp$ and $T_\perp$ are the reflection and transmission *coefficients* for the perpendicularly polarized light wave, θ_i and θ_t are the angles that the incident and transmitted waves make with the surface normal, n_i and n_t are *refractive indices*, and μ_i, μ_t are the *magnetic permeabilities* of the two media. Note that $R_\perp$ is the quantity that determines the magnitude of interface reflection, and hence the strength of highlight for perpendicular polarization.

For incident light waves whose polarization directions are parallel to the plane of reflection (see Figure 2.1(b)), phase matching of electromagnetic fields yields the following reflection and transmission magnitude relationships:

$$R_{||} = \frac{\frac{n_t}{\mu_t} cos\theta_i - \frac{n_i}{\mu_i} cos\theta_t}{\frac{n_i}{\mu_i} cos\theta_t + \frac{n_t}{\mu_t} cos\theta_i}, \tag{2.3}$$

$$T_{||} = \frac{2\frac{n_i}{\mu_i} cos\theta_i}{\frac{n_i}{\mu_i} cos\theta_t + \frac{n_t}{\mu_t} cos\theta_i}, \tag{2.4}$$

where $R_{||}$ and $T_{||}$ are the reflection and transmission coefficients for parallel polarization, and other symbols are as defined in Equations 2.1 and 2.2. Once again, $R_{||}$ is the quantity that determines the magnitude of surface or interface reflection.

2.2.1 Electromagnetic Field Strength and Irradiance Intensity

The reflection and transmission *coefficients* presented in the previous subsection relate the electric field strengths (magnitudes) of light waves before and after undergoing surface reflection. Normally, an imaging device like a CCD camera or a human eye, senses light proportional to its time average irradiance *intensity*:

$$< I >= \frac{|E|^2}{2\omega\mu},$$

where $|E|$ is the light wave's electric field strength, ω is its chromatic frequency measured in radians per second, and μ is the propagation medium's *magnetic permeability.* For surface reflected light, ω and μ remain unchanged, so the reflected to incident light *intensity* ratios are exactly $R_\perp^2$ and $R_{||}^2$ for perpendicular and parallel polarizations respectively.

2.2.2 Dielectric Materials

Equations 2.1 to 2.4 are unsimplified expressions for the reflection and transmission coefficients of incident electromagnetic waves on material interfaces. These expressions are derived directly from Maxwell's equations, so by first principles, they are valid for all possible pairs of *homogeneous* media with magnetic permeabilities and refractive indices. Often, in the scenes we analyze, most of the objects are dielectrics immersed in air, for which $\mu_i \approx \mu_t \approx \mu_o$ and μ_o is the *magnetic permeability* of free space. Factoring away the *magnetic permeability* terms, Equations 2.1 and 2.3 become:

$$R_\perp = \frac{n_i cos\theta_i - n_t cos\theta_t}{n_i cos\theta_i + n_t cos\theta_t}, \tag{2.5}$$

$$R_{||} = \frac{n_t cos\theta_i - n_i cos\theta_t}{n_i cos\theta_t + n_t cos\theta_i}, \tag{2.6}$$

Using Snell's Law to express one refractive index in terms of the other and then factoring out the refractive indices from the expressions, we get:

$$R_\perp = -\frac{\sin(\theta_i - \theta_t)}{\sin(\theta_i + \theta_t)}, \tag{2.7}$$

$$R_{||} = +\frac{\tan(\theta_i - \theta_t)}{\tan(\theta_i + \theta_t)}, \tag{2.8}$$

The squared surface reflection coefficients for a dielectric with *refractive index* $n_t = 1.5$ are as shown in Figure 2.2. A small *incident angle*, θ_i corresponds to a viewing geometry with the light source almost directly behind the viewer, while a large *incident angle* corresponds to a viewing geometry whereby the light source is almost directly behind the object. Notice that except for the extreme case where $\theta_i > 75°$ or the light source is almost directly opposite the viewer with respect to the object, surface reflections are generally weak, at least for parallel polarization components.

To show that surface reflection is indeed strongly polarized for dielectric objects, we examine the *Fresnel Ratio* for surface reflection, $[R_\perp/R_{||}]^2$, which indicates the *relative intensities* of reflected light in the perpendicular and parallel polarization directions. A ratio much larger than 1 indicates

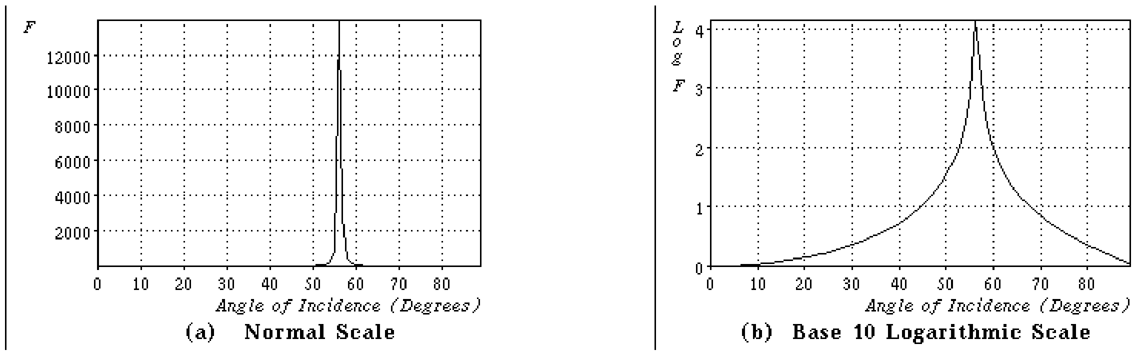


Figure 2.3: Fresnel ratio as a function of θ_i for $n_t = 1.5$. (a) Normal scale for $y - axis$. (b) Base 10 Logarithmic scale for $y - axis$.

that surface reflection is a lot more intense in the perpendicular direction than in the parallel direction. Surface reflected light for $[R_\perp/R_{||}]^2 \gg 1$ should then be strongly polarized, since it contains a large proportion of waves polarized in one particular direction. A ratio that is near unity gives rise to surface reflection whose strength depends very little on polarization angle. Light reflected from the surface should therefore be almost unpolarized, since it contains almost equal proportions of waves polarized at all angles.

We can use Equations 2.7 and 2.8 to derive the Fresnel ratio at a dielectric surface as a function of incident angle, θ_i:

$$\left[\frac{R_\perp}{R_{||}}\right]^2 = \left[\frac{\cos\theta_i\sqrt{1-(\frac{n_i}{n_t}\sin\theta_i)^2}+\frac{n_i}{n_t}\sin^2\theta_i}{\cos\theta_i\sqrt{1-(\frac{n_i}{n_t}\sin\theta_i)^2}-\frac{n_i}{n_t}\sin^2\theta_i}\right]^2. \tag{2.9}$$

Figure 2.3 shows the plot of Equation 2.9 with $n_t = 1.5$. Again, except for extreme viewing geometries where $\theta_i < 25°$ or $\theta_i > 85°$, the Fresnel ratio for surface reflection is at least 2, indicating that surface reflected light from dielectrics is generally strongly polarized.

2.2.3 Metals

The separation technique does not work well for metallic materials because their extremely high surface conductivities (σ) make surface reflection weakly polarized. Surface conductivity affects the *complex permittivity* of a substance as follows:

$$\bar{\epsilon} = \epsilon - j\frac{\sigma}{\omega} \tag{2.10}$$

where $j = \sqrt{-1}$, ω is the frequency of the incident light wave and ϵ is normal permittivity. Using the relationship between *refractive index* (n), *complex permittivity* ($\bar{\epsilon}$) and *magnetic permeability* (μ):

$$n = c\sqrt{\mu\epsilon} \tag{2.11}$$

where c is the speed of light in a vacuum, Equations 2.1 and 2.3 for the perpendicular and parallel surface reflection coefficients of metals can be re-expressed as:

$$R_\perp = \frac{\sqrt{\frac{\epsilon_i}{\mu_i}} cos\theta_i - \sqrt{\frac{\epsilon_t}{\mu_t}} cos\theta_t}{\sqrt{\frac{\epsilon_i}{\mu_i}} cos\theta_i + \sqrt{\frac{\epsilon_t}{\mu_t}} cos\theta_t}, \tag{2.12}$$

$$R_{||} = \frac{\sqrt{\frac{\epsilon_t}{\mu_t}} cos\theta_i - \sqrt{\frac{\epsilon_i}{\mu_i}} cos\theta_t}{\sqrt{\frac{\epsilon_i}{\mu_i}} cos\theta_t + \sqrt{\frac{\epsilon_t}{\mu_t}} cos\theta_i}. \tag{2.13}$$

Since large values of σ give rise to large *absolute* values of ϵ_t in Equations 2.12 and 2.13, we can approximate the coefficients of surface reflection as:

$$R_\perp \approx \frac{-\sqrt{\frac{\epsilon_t}{\mu_t}} cos\theta_t}{\sqrt{\frac{\epsilon_t}{\mu_t}} cos\theta_t} = -1, \tag{2.14}$$

$$R_{||} \approx \frac{\sqrt{\frac{\epsilon_t}{\mu_t}} cos\theta_i}{\sqrt{\frac{\epsilon_t}{\mu_t}} cos\theta_i} = 1. \tag{2.15}$$

This leads to a Fresnel ratio of $[R_\perp/R_{||}]^2 \approx 1$ for metals, which explains why surface reflection is weakly polarized.

2.3 Linear Polarizers and Highlight Reduction

Polarized light can be resolved into linearly independent components whose directions are perpendicular to the wave's direction of propagation. When light is transmitted through a *linear polarizer*[2], the magnitude of the transmitted electric field is proportional to the component of the incident electric field in the polarizer's orientation (see Figure 2.4(a)). For light that is unpolarized or randomly polarized, the electric field components of the wave are almost equal in all directions. So when unpolarized light passes through a linear polarizer, the magnitude and intensity of the transmitted light wave remains almost constant, regardless of polarizer orientation. Light that is partially polarized has electric field components that are stronger in some directions than others. When transmitted through a linear polarizer, the magnitude and intensity of the resulting light wave depends greatly on the orientation of the polarizer. Henceforth, we shall use the term *polaroid filtering* to describe the process of transmitting light through a linear polarizer.

[2]Also known as *polarizers* or *polaroid filters*

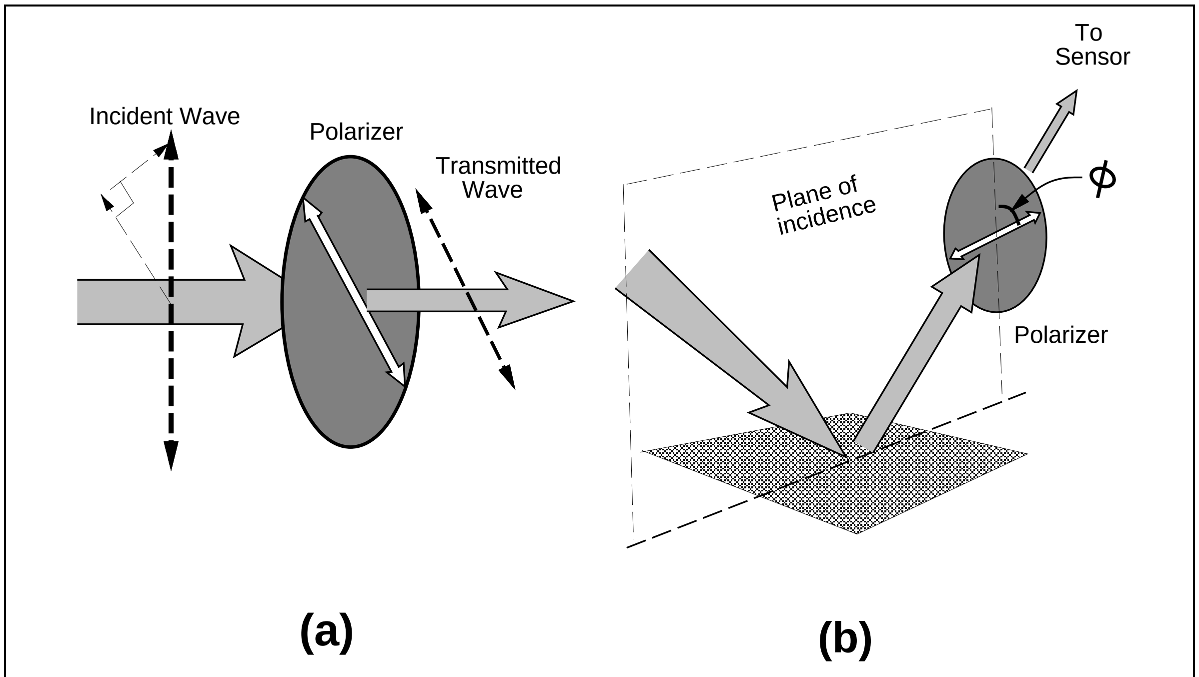


Figure 2.4: (a) Polarized light transmitting through a linear polarizer. (b) Imaging geometry for reducing surface reflection effects in images.

2.3.1 Minimizing Secondary Effects

Given that *body* reflected light from dielectric surfaces is almost completely unpolarized whereas *surface reflection* is generally polarized perpendicular to the *plane of incidence*, it is possible to use polaroid filters to help us obtain closer approximations to the color of body reflection from reflected light. Figure 2.4(b) shows the imaging geometry of the highlight reduction technique used in this thesis, where the main idea is to orientate the polarizer such that minimum surface reflection passes through the polarizer into the color sensor.

For the moment, let us assume that we know the surface orientation of all points in the scene, and hence also the plane of incidence for each ray of light entering the color sensor. Also, let us suppose that surface reflection at any point in the scene arises from a single *point* light source. The equation below expresses the intensity of light that the color sensor of Figure 2.4(b) receives, in terms of the phase angle (ϕ) that the polarizer makes with the plane of incidence. A phase angle of 0° indicates that the polarizing orientation is parallel to the plane of incidence, while a phase angle of 90° indicates that the polarizing orientation is perpendicular to the plane of incidence.

$$\mathbf{I}(\phi) = \frac{1}{2} I_b + \left[\frac{R_\perp^2}{R_\perp^2 + R_\parallel^2} \right] I_s \sin^2 \phi + \left[\frac{R_\parallel^2}{R_\perp^2 + R_\parallel^2} \right] I_s \cos^2 \phi \tag{2.16}$$

Equation 2.16 applies for grey-level readings in all three chromatic channels of a color signal.

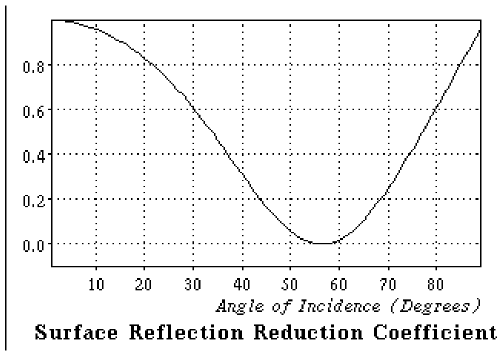


Figure 2.5: Fraction of Remaining Highlight after Polaroid Filtering as a function of Angle of Incidence ($n_t = 1.5$).

The symbols $\mathbf{I}(\phi)$, I_b and I_s represent the intensities of image irradiance, body reflection and surface reflection respectively. $R_\perp$ and $R_{||}$ are the reflection coefficients for perpendicular and parallel polarization as defined in the previous section. Since $R_{||}^2$ is always less than or equal to $R_\perp^2$, we see that the contribution of the surface reflection terms is minimum when the polarizer is oriented at $\phi = 0°$, parallel to the plane of incidence. Equation 2.16 becomes:

$$\mathbf{I}(\phi)|_{\phi=0°} = \frac{1}{2} I_b + \left[\frac{R_{||}^2}{R_\perp^2 + R_{||}^2} \right] I_s. \tag{2.17}$$

Multiplying the result by 2, we get:

$$2\mathbf{I}(\phi)|_{\phi=0°} = I_b + 2 \left[\frac{R_{||}^2}{R_\perp^2 + R_{||}^2} \right] I_s, \tag{2.18}$$

which always gives us a better approximation to the *body reflection* term, I_b, than a direct measurement of image irradiance:

$$\mathbf{I} = I_b + I_s. \tag{2.19}$$

Figure 2.5 shows the fraction of the original highlight intensity that still remains in an image after polaroid filtering, for incident angles in the range of 0° to 90°. Although a relatively large fraction of highlight still remains for small incident angles of $\theta < 35°$, the problem posed is minor for materials with high lambertian reflectances, because the absolute amount of highlight produced at these angles is relatively low. For large incident angles of $\theta > 80°$, we get relatively strong highlight effects whose intensity is only slightly reduced by the filtering technique. Fortunately, in most natural occurring scenes, points like these are almost totally occluded from the viewer's line of sight and make up only a small portion of the entire image.

2.3.2 Multiple and Extended Light Sources

The argument that highlight is minimized at $\phi = 0^\circ$ also holds for extended light sources or multiple point light sources illuminating a single point in the scene. A simple proof for the case of multiple point light sources proceeds as follows: According to the physical laws of reflection, in order for a color sensor to detect highlight at some point on a surface, the *surface normal vector* at the illuminated point must lie on the plane of incidence, which includes the light source, the color sensor and the illuminated point itself. This means that if a color sensor detects a mixture of highlight from a few light sources at some point in the scene, all the contributing light sources must share the same plane of incidence. Aligning a polaroid filter parallel to the plane of incidence would therefore minimize highlight contributions from all the individual light sources, and hence the highlight term as a whole.

Extended light sources can be modeled as a spatially continuous distribution of point light sources, so the proof outlined in the previous paragraph also applies.

2.3.3 Minimizing Secondary Effects for all Pixels

How does one go about determining the polarizer orientations that minimize secondary effects at each pixel in the image? A method that explicitly computes the plane of incidence at each pixel and aligns the polarizer with the plane requires precise and detailed knowledge of imaging geometry. Unfortunately, such information is usually not available to early vision modules operating on unconstrained natural occurring scenes.

A closer examination of the problem and of Equation 2.16 reveals that at $\phi = 0^\circ$, when the polarizer is oriented parallel to the plane of incidence at a given pixel, the pixel's intensity readings in all three chromatic channel are at a global minimum over all values of ϕ. Since our task is just to minimize $\mathbf{I}(\phi)$ for all pixels in the image, a much simpler approach would be to take multiple images of the scene at fixed intervals of ϕ and preserve only the minimum chromatic channel readings at each pixel. To ensure that the minimum values we get at each pixel are reasonably close to their true minimum values, we take 16 images of the scene at 11.25° intervals of ϕ to cover an equally spaced 180° arc for all possible planes of incidence. Our sampling resolution gives us a maximum possible phase mismatch of 5.625° between polarizer orientation and plane of incidence. This allows us to reduce highlight intensity at each pixel to at least 98% of the attenuation factor that we could achieve, when the polarizer and plane of incidence are exactly aligned.

2.4 Results

It has been argued that if image irradiance consisted of only body reflection from objects in the scene, and if all light sources in the scene had the same spectral composition, then a patch of uniform surface spectral reflectance in the scene will give rise to a patch of uniform image color (hue) in the sensor. The value of polaroid filtering in color vision can therefore be measured in this context; that is, a uniformly colored object should appear more uniformly colored in an image with polaroid filtering, than in the same image without polaroid filtering.

This section shows the outcome of polaroid filtering on three natural occurring scenes of dielectric bodies. In each scene, we investigate the performance of the filtering operation on a particular class of secondary imaging effects. The first example in Figure 2.6 demonstrates highlight removal on dielectric surfaces. In this scene, one light source is positioned so that highlights from the main reflecting surfaces have moderate incident angles of 25° to 80°. As predicted by the highlight reduction ratios of Figure 2.5, the operation eliminates a large portion of the highlight on the upper surfaces and handles of the inverted plastic cups. Notice the relative absence of highlight boundaries on the cups in the color edge map[3] of the filtered image, as compared with the cups in the color edge map of the unfiltered image.

Figure 2.7 deals with inter-body reflection in the scene. The painted grey metal cabinet in Figure 2.7 reflects light from a nearby camera tripod into the color sensor. After polaroid filtering, the camera tripod image disappears from the cabinet surface and the boundaries brought about by the tripod disappear completely from the Canny edge map[4].

In Figure 2.8, we see an example of incomplete filtering that does not totally remove strong highlight from a scene. A light source positioned almost directly behind the camera illuminates the plastic cup at a small angle of incidence. Because of the low highlight attenuation factor at small angles of incidence, a relatively large portion of the original highlight on the cup still remains after filtering. Taking vertical *hue* slices down the center of both images in Figure 2.8, we see that polaroid filtering still improves color uniformity for the cup in the image, especially at the highlight pixels.

2.5 Practical Issues

The advantages of polaroid filtering over other methods of separating reflection components are obvious. In principle, the technique is capable of operating at a *pixel* level of resolution, where highlight and other secondary effects at any pixel in the image can be reduced, independent of information obtained from other nearby pixels. No knowledge is required about local color variations in the image. Unlike Klinker, Shafer and Kanade's dichromatic model-based separation algorithm that matches color histogram signatures with model-generated hypotheses, polaroid filtering requires very little computation for recovering lambertian reflection components. All that the operation does is to remember the minimum irradiance value that each image pixel registers as the polarizer orientation changes. Another desirable feature about the technique is that its

[3]Color boundary detection is described in Chapter 5

[4]We are displaying luminance edges instead of color edges for this image because the cabinet and the light source have the same color.

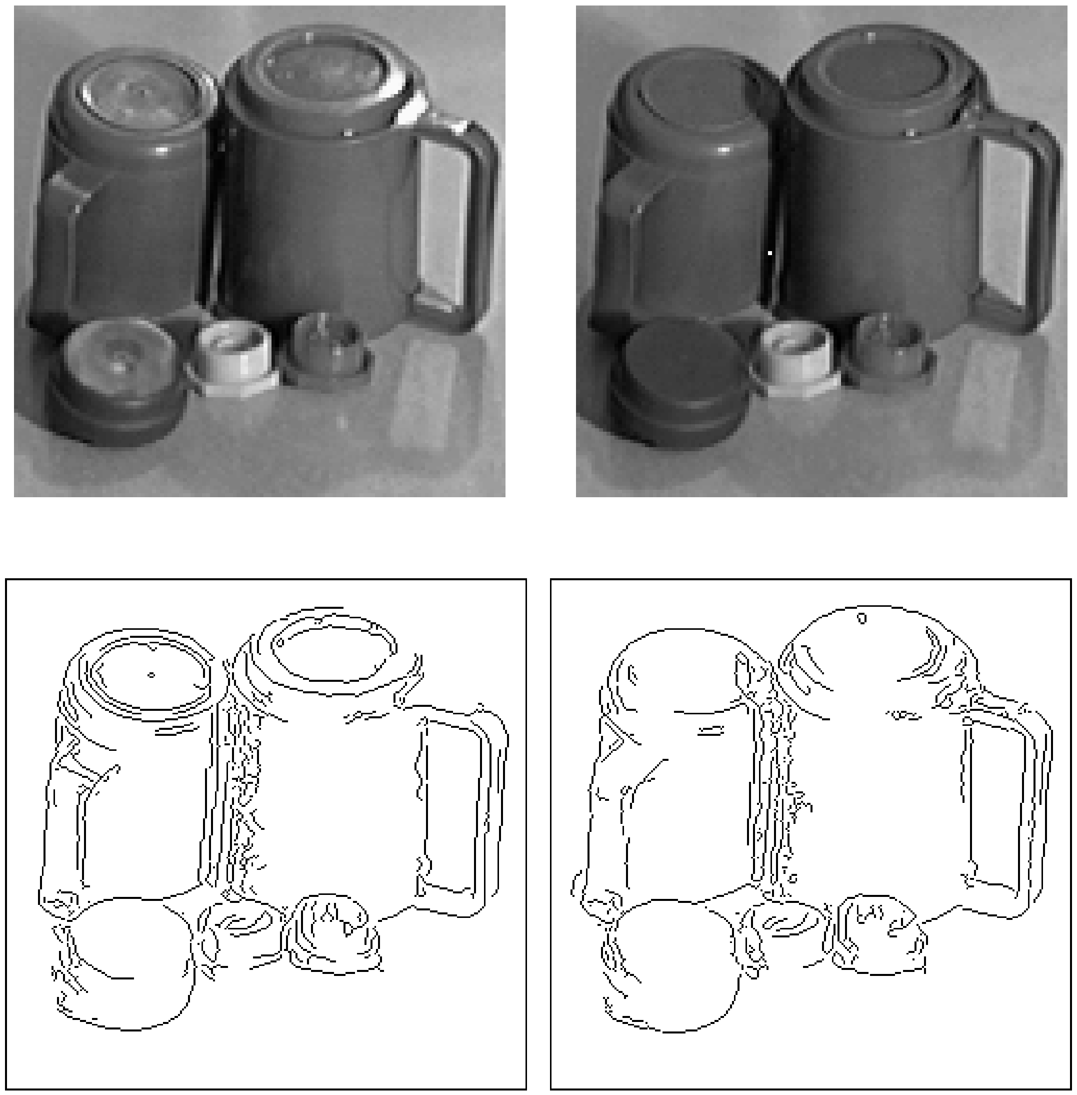


Figure 2.6: Top: Plastic Cup image with highlight, before and after polaroid filtering. Bottom: Color edge maps.

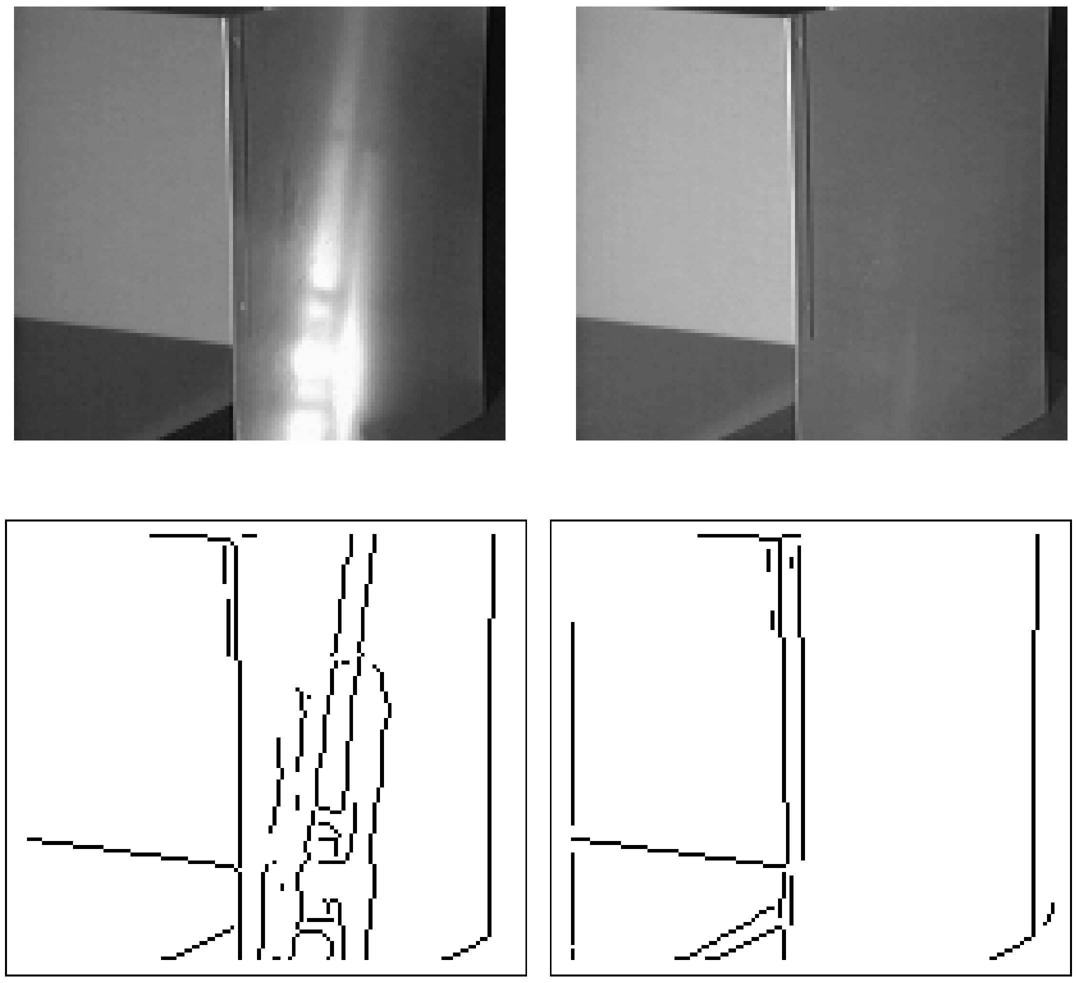


Figure 2.7: (a) Top: Painted grey metal cabinet image with reflection of camera tripod, before and after polaroid filtering. Bottom: Canny (luminance) edge maps.

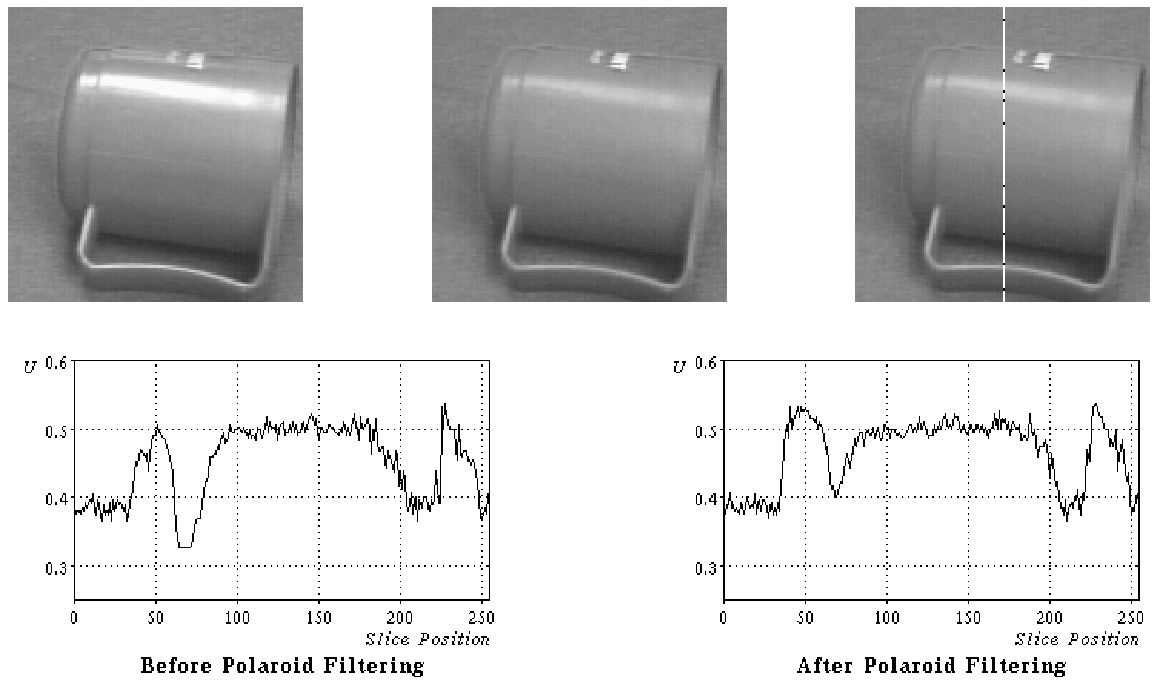


Figure 2.8: Top: Plastic cup image with strong highlight, after polaroid filtering and with vertical hue slice shown. Bottom: Cross section of hue slice before polaroid filtering and after polaroid filtering.

operation is not constrained by the number and spectral distribution of illumination sources in the scene. Filtering works equally well for illuminants of any arbitrary color, as long as surface reflection is sufficiently polarized. The dichromatic model-based scheme, on the other hand, fails when the color of the illuminant gets too close to the surface color of the object, or when there are multiple different colored light sources illuminating the same surface.

2.5.1 Real Time Imaging

In the images we produced, the polarizer attached to the front of the camera lens was manually rotated and frames were grabbed manually at regular phase intervals. A program was executed between frames to update the minimum image irradiance readings seen at each pixel so far. This cumbersome process restricted our choice of test scenes to still life objects in artificial lighting environments, where no changes in the scene could occur between frames.

Real time polaroid filtering requires an automated system to take over the manual chores we performed, so that changes in the scene can be minimized between frames. The problem of making quick and precise changes to polarizer orientation can possibly be solved by mounting a thin film of *liquid crystals* in front of the camera lens. Liquid crystals are substances that behave like electrically controlled polarizers, whose polarization state and orientation depend on the presence and direction of an applied electric field.

An ideal alternative to grabbing and processing multiple frames for each scene is to have image sensors with built in hardware that records only minimum values seen at each pixel over a given time interval. The length of the time window is fixed so that it corresponds to a full 180° phase cycle of the polarizing element.

The real time filtering ideas presented in this subsection are issues in sensor and optical imaging technology, beyond the scope of this thesis. No feasibility studies have been made in this thesis about the proposed solutions.

2.5.2 Finding Better Lambertian Estimates

One natural extension to our current system would be an algorithm that computes even closer estimates to body reflection from available polarization data. Although polaroid filtering removes a substantial amount of secondary imaging effects from the scene over a wide range of incidence angles, Figure 2.5 shows us that in most geometries, some surface reflection still remains after filtering. The ideal extension we seek should therefore allow us to correctly solve for the unattenuated surface reflection components:

$$I_{\perp} = \left[\frac{R_{\perp}^2}{R_{\perp}^2 + R_{\parallel}^2}\right] I_s$$

and

$$I_{||} \quad = \quad \left[\frac{R_{||}^2}{R_\perp^2 + R_{||}^2}\right] I_s$$

of Equation 2.16 at each image pixel, so that further compensation can be made to obtain a better lambertian reflection estimate.

Unfortunately, we cannot solve for $I_\perp$ and $I_{||}$ at each pixel, using only polarization data from the pixel itself. Casting image irradiance readings from three distinct polarizer orientations into a set of simultaneous equations, we get:

$$\begin{bmatrix} \mathbf{I}(\theta_1) \\ \mathbf{I}(\theta_2) \\ \mathbf{I}(\theta_3) \end{bmatrix} = \begin{pmatrix} 1 & \sin^2\theta_1 & \cos^2\theta_1 \\ 1 & \sin^2\theta_2 & \cos^2\theta_2 \\ 1 & \sin^2\theta_3 & \cos^2\theta_3 \end{pmatrix} \begin{bmatrix} I_b \\ I_\perp \\ I_{||} \end{bmatrix}. \qquad (2.20)$$

The rows of the 3×3 matrix are not linearly independent for all possible combinations of θ_1, θ_2 and θ_3, so the matrix is not invertible. Hence, no unique solution exists for the vector $[I_b \; I_\perp \; I_{||}]^T$.

To overcome the problem of getting linearly independent readings for recovering I_b, Wolff [Wolff 89] first defines the physical quantity *Fresnel Ratio* as $I_\perp/I_{||}$, or equivalently $[R_\perp/R_{||}]^2$. Like $I_\perp$ and $I_{||}$, the Fresnel Ratio is a quantity that cannot be computed at a given pixel, using polarization data from the pixel alone. Wolff's algorithm makes use of polarization data from pixels within an entire pre-segmented region to compute an average Fresnel Ratio for the region. It then computes I_b at each pixel in the region using the average Fresnel Ratio value and a re-expressed version of Equation 2.16, with ϕ set to $0°$ and $90°$. Unfortunately, the algorithm relies on a dangerous assumption that Fresnel Ratios are approximately constant within a pre-segmented region. For flat surfaces where incident angles (θ_i) are relatively uniform throughout, the assumption could be valid. However, in the experiments we conducted for objects with curved surfaces, the technique failed badly even when small local patches of 5×5 pixels were used to compute Fresnel Ratios.

Future work in polaroid filtering should focus on combining polarization data with model-based methods of separating reflection components for obtaining better body reflection estimates. Polarization data provides precise and reliable information on surface reflection color. This information can be well used in model-based separation techniques to generate and confirm highlight and inter-body reflection hypotheses.

Chapter 3

Representing Color

To process color as a cue in computer vision, a visual system should first have an appropriate representation for color signals. Often a well chosen representation preserves essential information and gives good insight to the operations that we want to perform. In Chapter 4, we see that a good color representation even allows us to develop and express color concepts that are otherwise not intuitive at first glance. Generally, finding an appropriate representation for some entity involves knowing how data describing the entity is generated and what information is wanted from the data. That is, before we even decide upon a method of representing color, we must first have a clear understanding of what constitutes *color* and what kind of results we expect from our color operations.

As mentioned in Chapter 1, surface color is an excellent cue for extracting material information in a scene. In this thesis, our primary focus is on performing surface color operations. Since surface color, or body color, affects only the *body* component of reflection, we shall limit our discussion on color in the rest of this chapter to the context of *body* reflection alone. This is a reasonable thing to do because in Chapter 2, we showed a method of substantially reducing the effects of surface reflection in a scene using polaroid filters. Our goal here is to first establish a notion of *color* that relates to the material composition of a scene. Thereafter, we shall choose a color representation scheme that best reflects the notions of color similarity and difference in terms of material similarity and difference.

3.1 Color as a Ratio of Spectral Components

Since the time of Isaac Newton, scientists have been studying the perception of color from many different standpoints. Color vision literature can been found in the domain of many modern sciences, including physics, artificial intelligence, psychology, physiology and philosophy. But even until today, many questions about color still remain unresolved. For example, what does one mean by saying that a banana appears yellow? How does one determine if two plastic cups have similar or even identical color?

We believe that in the context of this thesis, color should be treated as a physical measure, specifically, as a *normalized ratio* of light intensities in its separate chromatic wavelengths. One example of such a formulation appears in Land's work on *Lightness Algorithms* [Land 59]. Land defines the surface color of an object as its *relative* surface spectral reflectance, $\rho(\lambda)$, in each of three independent chromatic channels. Similarly, the color of *body reflected light* is defined as its *relative* irradiance, $\mathbf{I}(\lambda)$, in three independent sensor channels, where $\mathbf{I}(\lambda)$ and $\rho(\lambda)$ are as related in the *Image Irradiance Equation* (Equation 1.1) of Chapter 1. Normally, we use the red, green and blue spectral wavelengths as the three independent chromatic channels, corresponding to the three primary colors of visible light. Let us now see why such a physical color formulation is suitable for our goals in this thesis.

3.1.1 A Surface Color Model

Figure 3.1 shows a microscopic model of a dielectric surface that explains the occurrence of surface color and *body* reflection. Similar models have been used by Klinker [Klinker 88], Johnston and Park [Johnston and Park 66], Hunter [Hunter 75] and many others for describing surface and body reflected light. Most dielectric materials consist of a *medium* with *color pigments*. The medium forms the bulk of the material body and is generally transparent to light waves in the visible spectrum. The color pigments embedded in the medium selectively absorb and scatter light rays by a random process of reflection and refraction. Normally, samples of the same material have similar color pigment densities and composition, while samples of different material have different pigment densities and composition. The presence of these pigmented particles in the material medium gives rise to the *surface* color or *body* color that we associate with the dielectric surface. In this model, we can think of *surface* color as the fraction of light energy entering the medium at each chromatic wavelength that does not get absorbed by the color pigments. Because light interacts *linearly* with matter under a wide range of illumination intensities, the fraction of unabsorbed light is usually a constant quantity at each chromatic wavelength.

Body reflection results from light interacting with the color pigments of a surface. When light penetrates a dielectric surface, it travels through the medium, hitting pigmented particles along its path. Each time it interacts with a pigmented particle, certain wavelengths get strongly attenuated by the particle while others get reflected. Eventually, when it exits from the medium, only the unattenuated or slightly attenuated wavelengths remain, giving body reflected light its characteristic color. For most dielectric substances, the color pigments within the medium are uniformly distributed throughout, so the color composition of an emerging light ray does not depend on the path it takes within the medium. Also, since a light ray usually encounters many color pigments and undergoes multiple random reflections before emerging from the medium, the direction it emerges does not depend on the direction it enters the medium. All this suggests why

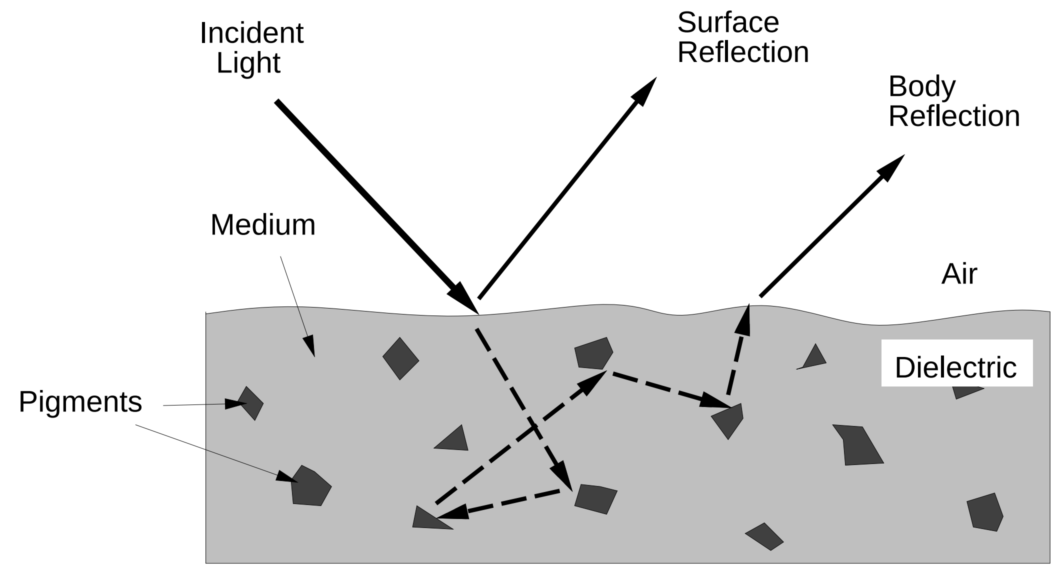


Figure 3.1: Microscopic model of a dielectric surface.

body reflected light is mostly uniform, diffused and highly unpolarized.

3.1.2 Body Reflection Color, Body Color and Material Composition

The dielectric surface model we described suggests that the *surface color* or *body color* of an object is material dependent and does not change under different illumination conditions. The color of body reflected light, defined as its *relative* spectral irradiance intensities, depends on both the material's *body color* and the scene's illuminant color. Body reflected color is also intensity invariant in that it does not change even if the intensity of the light source changes.

Because color pigment density and pigment composition are both relatively constant quantities within a sheet of uniform material, we expect body color to be a relatively uniform quantity as well within the sheet of material. This means that in a scene with no spatially abrupt changes of illuminant color, regions of uniform surface color can only give rise to regions of continuous body reflection color in the image. The converse is also true for most pairs different surface material types. Normally, we expect the color pigment densities and pigment compositions of different material pairs to be sufficiently different, so that significant body color discontinuities can exist at their junctions. In a scene with no spatially abrupt changes of illuminant color, sharp body reflection color discontinuities in the image can therefore only be caused by material boundaries in the viewer's environment.

In summary, treating color as a normalized ratio of spectral intensities provides us with a sensitive measure for inferring color pigment variations in the scene. Very often, variations in color pigment density and composition correlate well with surface material variations.

3.1.3 Color Pigments

It may be interesting to note that color pigments actually exist beneath the surface of many natural and artificially occurring dielectric substances. Furthermore, certain pigments can even be identified by the characteristic colors they cause. Plant tissues for example contain *chlorophyll* that gives rise to the greenish appearance of leaves and young stems. *Melanin* pigments found in most animal tissues and some older plant tissues, such as the bark of trees, give rise to the natural brownish appearance of these substances. Other iron-containing protein pigments, such as hemoglobin, account for the reddish purple color of mammalian blood and certain respiratory tissues of invertebrates.

Organic and inorganic pigments are often introduced into artificial dielectrics to produce colors. For example, in making paints, lead compounds are usually mixed with the solvent to produce white paint, while copper based compounds are usually added for greenish and bluish colors. Metallic pigment compounds can also be found in most natural and synthetic colored gems like rubys, sapphires and emeralds. Textiles are dielectrics that contain fiber like substances as a medium. When dyes are applied to textiles, dye pigments dissolve into the textile fibers to selectively absorb light entering the medium.

3.1.4 Metals

When light falls on a metallic surface, almost all of its incident light energy gets reflected off the interface as surface reflection. Hence, the surface color model that we described here for dielectrics does not apply for metals. Surface reflection is also very much unpolarized for metals, so we cannot use polaroid filtering to get a better estimate of body color. Because "white" metals like silver, zinc and aluminum reflect visible light of all spectral wavelengths equally well, they show up in the scene having the same color as their illuminating source. On the other hand, "brown" metals like copper, gold and brass absorb wavelengths nearer the blue end of the visible spectrum, so shorter light wavelengths get strongly attenuated in their surface reflection. Both classes of metals, however, interact linearly with light, so like dielectric body reflection, their surface reflected color also depends heavily on the color of their illumination source.

3.2 Color as a 3 Dimensional Vector Space

Given the color model that we want to adopt, how does one go about representing color as a *ratio* of spectral intensities? Certainly, an obvious scheme would be to represent irradiant color or surface spectral reflectance as an analytic function of spectral wavelength, λ. This scheme preserves all spectral information of the color signal. Unfortunately, determining such an analytic function for color involves having to compute a function that fits the signal's spectral intensity measurements at each wavelength sufficiently well. Generally, such a function has to be computed from a set of dense spectral intensity samples, and the resulting function can be a polynomial of an arbitrarily high order.

3.2.1 Human Color Perception is 3 Dimensional

One feasible alternative is to represent color as a vector of *net* spectral intensity or spectral reflectance response samples, integrated over λ[1]. Physiological and psychophysical evidence have shown that human color perception can be described by a vector space with only three dimensions. A simple argument is based on the observation that human beings have only three types of color sensing units or cones in their retina. Each type of cone detects light from only a narrow band of the visible spectrum, centered at some fixed wavelength. Even though there are only three types of cones in the human retina, the combined responses from all three types of cones at some retina location can give rise to all the different colors that humans perceive.

Colorimetry methods also reveal the three dimensional nature of human color perception by demonstrating that all human perceivable colors can be uniquely synthesized using three *linearly independent* colored light sources. By three linearly independent colors, we mean no two colors can be linearly combined to produce the third color. A subject faces a large grey surface with two holes, where behind each hole is a white lambertian surface. The surface behind one hole is then illuminated by light of a certain color and the subject's task is to control the intensities of three linearly independent light sources in the second hole, so that the color and intensity in the two holes become indistinguishable. The results obtained indicate that with three linearly independent light sources, there is a unique combination of intensities that produces each match. If only two linearly independent colored sources are used, most colors cannot be matched by any combination of intensities. With four or more colored sources, a given intensity and color can be matched by multiple intensity setting combinations.

3.2.2 The RGB Color Vector Space

Since humans can perceive a sufficiently wide range of different colors and human color perception spans a three dimensional space, we propose using a 3 dimensional *vector* representation for color in this thesis. Each dimension of the vector space encodes the *net* irradiance intensity or the *net* surface spectral reflectance response of a color signal at a selected spectral wavelength. In other words, to represent a color signal in this scheme, we construct a 3 dimensional vector whose components are the signal's *net* irradiance intensities or its *equivalent* surface spectral reflectances in the 3 chosen spectral wavelengths.

[1]The idea is similar to approximating the spectral function of λ with a finite number of basis functions. See, for example, [Yuille 84] for work on Spectral Basis Algorithms. In the human visual system, we have 3 basis functions — the sensitivity functions of the 3 types of cones.

In order to successfully represent all possible humanly perceivable colors in our vector space, the three spectral wavelengths that we choose as vector components must be linearly independent in color: That is, no linear combination of any two chosen wavelengths can form the color of the third wavelength. An obvious choice of three such colors are the three primary colors of visible light, namely the `Red`, `Green` and `Blue` spectral wavelengths, which form a set of orthogonal basis vectors in the human color space. Since most commercially available color cameras also sense in the red, green and blue spectral channels, this choice of vector components allows us to directly encode the red, green and blue channel intensities of a color signal as the red, green and blue components of its color vector. This vector space is commonly known as the `RGB` *Color Vector Space* [Judd and Wyszecki 75] or the `RGB` *Color Cube* [Klinker 88], in terms of its three basis vector components.

3.2.3 A Color Difference Measure

How should color similarity and difference be quantified in our representation scheme? This is a tricky problem because we are seeking a single *scalar* quantity to measure multi-dimensional *vector* differences. Since color encodes intensity components in three independent chromatic channels, and for the time being, if no channel is any more important than the other, our difference measure must be *equally* sensitive to intensity changes in all three channels. Also, in order to truly capture the notion of color as a *ratio* of intensities in three chromatic channels and not as a triplet of absolute intensity values, the difference measure must only respond to *relative* intensity differences and not *absolute* intensity differences. That is to say, the measure should be insensitive to intensity changes alone that are not color changes. In other words, color vectors having the same *relative* channel intensities but different *absolute* channel intensities should be treated as representations of the same color, and have *zero* as their pairwise difference measure.

We shall approach the problem by first considering how similar and dissimilar colors appear as vectors in the `RGB` *Color Vector Space*. If color is defined as its *relative* values in the three sensor channels, then similar colors should map to vectors with similar component ratios in the `RGB` Color Space, while significantly different colors should map to vectors with dissimilar component ratios. Graphically, a pair of vectors with similar component ratios point along the same general direction in the `RGB` Color Space while vectors with dissimilar component ratios point in different overall directions. For example, Figure 3.2(a) shows the vector representations, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, of two similar colors, where both vectors have almost parallel orientations. In Figure 3.2(b), the colors represented by vectors $\boldsymbol{c}_3$ and $\boldsymbol{c}_4$ have dissimilar sensor channel ratios, so the vectors point in significantly different directions. Notice that it is not the *absolute* magnitudes of a color vector's components, but their *relative* magnitudes that determine the vector's orientation, and hence the color it represents.

It appears therefore, that we can use the orientation difference between two color vectors as

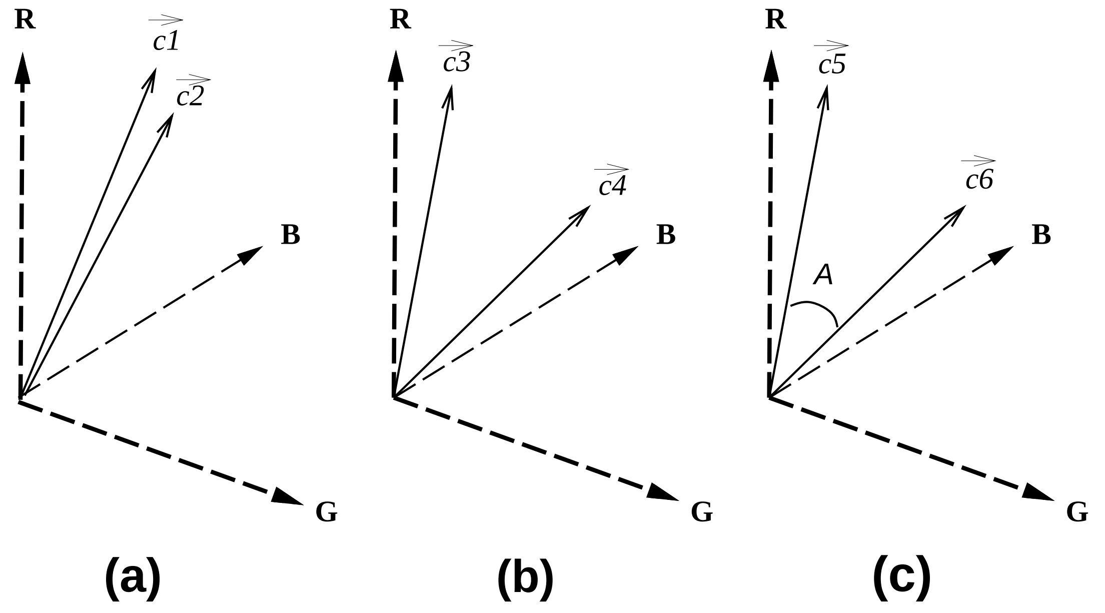


Figure 3.2: (a) Color vectors for 2 similar colors. (b) Color vectors for 2 dissimilar colors. (c) The included angle, A, between 2 color vectors.

their color difference measure, because their orientation difference correlates well with their spectral ratio difference. We propose using the magnitude of the *included angle* between the two vectors to quantify their difference in orientation. Figure 3.2(c) shows what we mean by the included angle, $\mathcal{A}$, between two color vectors, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$. $\mathcal{A}$ can be easily computed from $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ as follows:

$$\mathcal{A} = \arccos\left(\frac{\boldsymbol{c}_1 \odot \boldsymbol{c}_2}{|\boldsymbol{c}_1||\boldsymbol{c}_2|}\right) \tag{3.1}$$

where $\odot$ stands for the vector *dot-product* operation. As desired, this angular difference measure responds only to true color changes and does not favour magnitude changes in any one channel more than the others.

In the forthcoming chapters, we shall incorporate this notion of *angular* color uniformity and difference into our color operations to perform smoothing, edge detection and region segmentation.

3.3 Other Color Spaces and Difference Measures

This section reviews some other approaches to representing color and computing color differences. Where appropriate, comparisons will be made with the RGB vector representation scheme and our angular color difference measure.

3.3.1 UV Normalized Chromaticity Plane

Part of the difficulty in working with color as a ratio is that one has to keep track of irradiance changes in multiple chromatic channels and how the changes relate to one another. *Normalized colors* allows one to perform simple transformations on original sensor readings, like intensities in the `Red`, `Green` and `Blue` sensor channels, to create a new set of readings, say **H**, that changes only at true color boundaries. All that a color algorithm has to do then, is to interpret variations in **H** as color variations in the scene. Lee [Lee 86], Hurlbert and Poggio [Hurlbert and Poggio 88] and many others use the following set of transformations:

$$u = \frac{R}{R+G+B} \tag{3.2}$$

$$v = \frac{G}{R+G+B} \tag{3.3}$$

$$w = \frac{B}{R+G+B} \tag{3.4}$$

to obtain an intensity independent co-ordinate system for color. Normally, only the u and v values of the transformation are preserved and mapped onto a two-dimensional co-ordinate system, known as the *uv normalized chromaticity plane.* Values of w are discarded because they can be easily derived from u and v. This representation elegantly reduces a three-dimensional color signal to a two-dimensional co-ordinate system by mapping all `RGB` triplets of the same color onto the same location of the uv chromaticity plane.

Finding a satisfactory color difference measure becomes a problem in this color representation scheme if we want a measure that responds equally to magnitude changes in all three chromatic channels. Euclidean distances on the uv chromaticity plane tend to magnify irradiance changes in the `Red` and `Green` chromatic channels more than changes in the `Blue` channel. For example, the colors `Red`, `Green` and `Blue` have $R:G:B$ relative channel intensities of $1:0:0, 0:1:0$ and $0:0:1$ respectively, and uv co-ordinates of $(1,0)$, $(0,1)$ and $(0,0)$ in the chromaticity plane. White light has an $R:G:B$ channel ratio of $\frac{1}{3}:\frac{1}{3}:\frac{1}{3}$ and a uv co-ordinate of $(\frac{1}{3},\frac{1}{3})$. Although the $R:G:B$ ratio of *White* light is equally different from the $R:G:B$ channel ratios of `Red`, `Green` and `Blue` light, its uv chromaticity co-ordinate is geometrically closer to the `Blue` co-ordinate than to the `Red` and `Green` co-ordinates. This distance anomaly may be somewhat compensated for if we used *weighted uv* Euclidean distances:

$$||\underline{x}||_W = \underline{x}^T W^T W \underline{x}$$

instead of normalized uv Euclidean distances as a difference measure, where $\underline{x}^T = [\Delta u \Delta v]$ is the uv difference vector and W is a 2×2 matrix of relative weights.

3.3.2 CIE Uniform Color Spaces and Difference Measures

CIE[2] uniform color spaces are designed to predict the magnitude of human perceived differences between pairs of non-matching colors. Judd and Wyszecki [Judd and Wyszecki 75] provides a good overview documentation about CIE color spaces and the problems involved with finding satisfactory color difference formulae.

One of the major problems with computing human perceived color difference is that the observer's judgment varies greatly with the conditions of observation and the nature of the color stimuli. Among the factors affecting the observer's judgment are sizes, shapes, brightness and relative intensities of the test objects and also their surroundings. Although there is currently a large amount of experimental data on human color discrimination, CIE researchers have only been able to design empirical models of color spaces that match the data only under certain experimental conditions.

For standardization purposes among the scientific community, the CIE has proposed the use of two approximately uniform color spaces and their associated color difference measures. Both spaces are intended for normal observation conditions and in some situations, there is even evidence that different sets of coefficients in the color difference formulae may be more appropriate. The **CIE 1976** ($L^* u^* v^*$) **Space** is produced by plotting the quantities L^*, u^* and v^* in rectangular co-ordinates, where L^* is a function of irradiant intensity while ($u^* v^*$) encodes hue and saturation information. The **CIE 1976** ($L^* a^* b^*$) **Space** is another three dimensional rectangular color space where L^* also encodes intensity and ($a^* b^*$) results from a different transformation for hue and saturation values. Both color spaces use Euclidean distances as their color difference measures, which unfortunately does not preserve the notion of color as spectral ratios.

3.3.3 Other 3D Vector Spaces

Physicists and Psychologists have devised a number of linear and non-linear transformations on RGB channel intensities to obtain other color spaces that model certain psychophysical effects. The YIQ space, for example, attempts to model the *opponent-color theory* of human color vision. Y is a quantity very much like the overall intensity of an irradiance signal. The other 2 quantities, I and Q, are chromaticity values that express color composition using a co-ordinate system whose axes are *opposing color* measures like red and green, yellow and blue. The HSD (Hue, Saturation and Density) space uses white light as a reference signal, from which all colors are encoded according to their peak spectral location in the visible light spectrum (Hue) and spectral purity (Saturation). It is not clear from current computer vision literature whether any of these color spaces or color difference measures are better suited for color discrimination purposes than our RGB vector representation scheme.

[2]International Commission of Illumination.

3.4 Discussion

This chapter presents a color representation scheme and a color difference measure derived from a pigmentation model of dielectric surfaces. The scheme treats color as a ratio of spectral intensities to reflect the physical property of light interacting linearly with material pigments. In a scene with uniform or slowly changing illumination, the scheme predicts sharp changes in body reflection color only at material boundaries, because the different pigment compositions of different materials give rise to their different surface colors. Our angular color difference measure, therefore, detects changes in color pigment content between different regions in the scene by detecting changes in body reflected color.

By ascribing equal weights to irradiance changes in the `Red`, `Green` and `Blue` chromatic channels, the color difference measure assumes that the spectral composition of most natural occurring scenes contains almost equal amounts of energy in the three spectral wavelengths. Unfortunately, this equal weighting feature makes the difference measure a poor approximation to human color difference perception. There is evidence showing that humans are actually a lot more sensitive to spectral changes in the `Red` and `Green` channels than in the `Blue` channel. Some cone densities studies near the fovea of the human retina have even produced red, green and blue cone ratio estimates that are as unequal as $32 : 16 : 1$ [Vos and Walraven 70]! While it appears to be true that most natural occurring substances like plants, animal tissues and geological features have colors that peak near the red and green bands of the visible spectrum, man-made substances are equally likely to come in all possible colors. It is conceivable therefore, that in an environment with man-made objects, our equally weighted angular measure is likely to be a better overall detector of color differences than a measure that tries to emulate human color difference perception.

Because the proper functioning of our representation scheme and color difference measure depends only on color being treated as a spectral ratio, we can make our difference measure more sensitive to changes in certain spectral channels by simply scaling sensor readings in the different channels appropriately. For example, to make a system twice as sensitive to irradiance changes in the `Blue` channel than in the `Red` and `Green` channels, all one needs to do is to double all sensor readings in the blue channel before building `RGB` vector representations. Notice that scaling still maps `(R G B)` triplets with the same channel ratio to parallel vectors in the color space. A major advantage of this feature is that a new set of relative channel sensitivities can always be selected to suit the task being performed.

The scheme can also be generalized to use any suitable set of orthonormal color basis vectors other than `Red`, `Green` and `Blue`. For example, a better set of orthonormal color basis vectors can possibly be derived by performing *principle component analysis* [Young 86] on a representative sample of pigment colors. Although such an analysis is beyond the scope of this thesis, our representation scheme facilitates the use of a "better" set of color basis vectors should such a set of vectors be found.

Chapter 4

Color Noise

When working with natural occurring scenes, one inevitably has to deal with image noise. Noise introduces random errors into sensor readings, making them different from the ideal irradiance values predicted by the *Color Image Irradiance Equations* 1.1 and 1.2 of Chapter 1. These random errors often bring about undesirable side effects in subsequent vision processes. In boundary detection for example, noise can give rise to unexpected irradiance discontinuities within surfaces, or cloud out weak discontinuities between objects, resulting in unwanted spurious edges or missing boundaries in edge maps. Similarly, in region finding or image segmentation processes, noise can perturb sensor readings in ways that lead to over-merging or fragmentation of image regions.

In Chapter 3, we introduced a *surface color* representation scheme based on the physical properties of color image formation. This chapter addresses the next logical problem in a signal processing set-up for color, namely representing and reducing noise effects in the color context. We begin by first discussing what we mean by *noise* in color signals and show how it can be quantified in a way consistent with our color representation scheme. Next, we extend the concept of *smoothing* into the color domain as a means of reducing the strength of random *color* fluctuations in images. The latter half of this Chapter presents two color noise reduction techniques that preserve the sharpness of color boundaries, while smoothing away unwanted noise fluctuations within uniform color surfaces at the same time.

4.1 What is Color Noise?

As mentioned in Chapter 3, color images are usually encoded as scalar irradiances in three independent sensor channels, namely the Red, *Green* and *Blue* chromatic channels. Like grey-level intensity sensors, color sensors can also be affected by noise. This happens whenever there are random fluctuations in the scalar image irradiances recorded by each color channel. Normally, we can expect the strength of *scalar* noise, N, in each color channel to be *White Gaussian* in form; that is, having the following magnitude probability distribution:

$$Pr(\mathcal{N} = n) = G(\sigma, n) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{n^2}{2\sigma^2}}. \qquad (4.1)$$

To simplify our calculations for the rest of this chapter, we shall also assume that sensor noise behaves like pairwise independent random variables in all three chromatic channels, having identical probability distribution functions. That is, all three sensor channels have the same *zero* average noise magnitude and the same noise variance, σ, over the entire image. For any given image pixel however, the noise readings in the three channels are totally uncorrelated, hence giving them a pairwise independent random relationship. Under most imaging conditions with CCD cameras, these are all reasonable assumptions.

4.1.1 Quantifying Color Noise

In order to deal with noise in the color domain, one should first have a proper understanding of noise as a color effect. For example, how should one quantify noise for color values? What does it mean when the *color* of one surface is more *noisy* than the *color* of another surface? Perhaps the most straight-forward treatment of noise in a color signal is to keep track scalar noise magnitudes in the three chromatic channels separately. Existing scalar noise operations can then be applied to the three chromatic channels individually to reduce noise effects. It turns out, however, that such a direct treatment of noise in the color context is in fact inconsistent with our *normalized ratio* definition of color. This is because the same set of sensor channel noise readings that one has, can be translated into color perturbations of different magnitudes, depending on the actual value of the original color signal. We shall see later in this section how this comes about.

We believe that just as the general notion of noise describes the amount of random fluctuation in a given quantity, *color noise* should measure the strength of random *color* fluctuation in a color signal. Because we are adopting a scalar angular measure for quantifying color difference in this thesis, we propose using a similar scalar angular measure for quantifying color fluctuation, and hence color noise as well. Figure 4.1 shows what we mean by an *angular* noise margin in a *color* signal. This approach of representing color noise has the following advantages:

First, the angular measure for color noise is consistent with our intuitive understanding of color and the noise notion in general. Just as we would expect to find large differences in values among members of a noisy scalar distribution, we can also expect to see wide angular spreads of color vectors, and hence large differences in color values among samples of a noisy color distribution.

Second, the approach allows us to compare color noise strengths objectively on an angular magnitude scale. This is important because it provides us with a means of evaluating the noise reduction performance of a color noise operator quantitatively, by comparing absolute signal noise levels before and after the operation.

Third, by using the same units of measurement for color difference and color noise, we can compute *Signal to Noise Ratios* (SNR) for color images easily by taking direct quotients the two

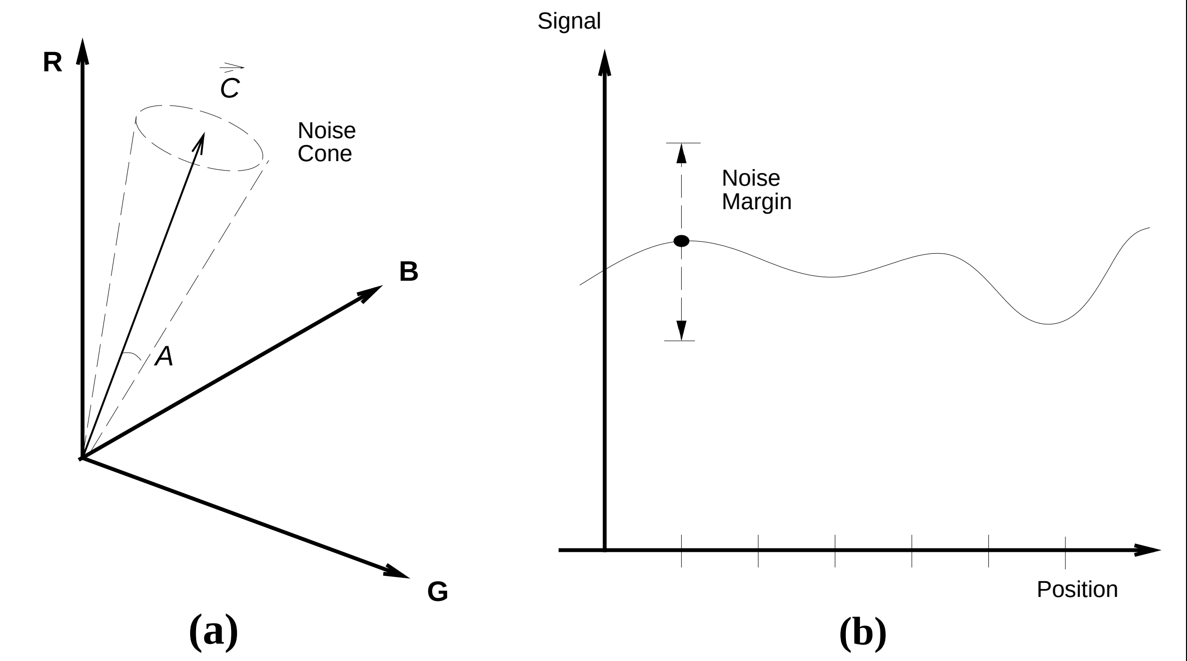


Figure 4.1: Noise margins for (a) a Color signal, quantified as an *angular* measure; and (b) a scalar signal, quantified as a *scalar* measure.

quantities. The SNR is a much better indication of how severe noise effects are in an image than absolute noise strength alone. A high SNR implies that noise induced color fluctuations are small as compared with actual color differences between image regions. This means that by defining appropriate thresholds, we can still reliably tell apart color changes caused by noise from color changes across object boundaries. A low SNR on the other hand implies that noise induced color fluctuations are large as compared with actual surface color differences, and can be easily mistaken as actual color changes because of their magnitude. Under such circumstances, a vision system might have trouble producing reliable edge-maps and region segmentations for the image.

4.1.2 Sensor Channel Noise and Color Noise

The rest of this section examines quantitatively how scalar noise in a color sensor contributes to angular noise in a color signal. More precisely, we want to derive an angular probability distribution function, $P_A(A)$, for color noise, in terms of scalar sensor noise magnitudes, σ, similar to the White Gaussian scalar noise probability distribution function given in Equation 4.1.

We shall proceed by considering first the idea of color noise as color *difference vectors.* Suppose we represent noise in the 3 channels of a color pixel as a 3D *perturbation vector* in an RGB color space. Let us also define the *magnitude* of a perturbation vector as:

$$p(r,g,b) = \sqrt{r^2 + g^2 + b^2} \tag{4.2}$$

where r, g and b are the scalar noise perturbation magnitudes in the *Red*, *Green* and *Blue* chromatic channels respectively.

Since we are assuming that all three chromatic channels have identical noise distribution functions of strength σ, we can expect the the perturbation vector to have a spatial probability distribution function that is spherically symmetric about its origin, or in other words, one that depends only on the value of p. The perturbation vector's *magnitude*, p, therefore has the following probability distribution function:

$$Pr_p(p) = \int_{-p}^{p}\int_{-\sqrt{p^2-r^2}}^{\sqrt{p^2-r^2}} Pr_r(r)Pr_g(g)[Pr_b(\sqrt{p^2-r^2-g^2}) + Pr_b(-\sqrt{p^2-r^2-g^2})]dgdr$$

where $Pr_r(n) = Pr_g(n) = Pr_b(n) = G(\sigma,n) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{n^2}{2\sigma^2}}$ are all White Gaussian noise distributions with variance σ^2.

Substituting Gaussian expressions for the channel noise distributions Pr_r, Pr_g and Pr_b of the double integral and performing the integration step, we eventually get:

$$Pr_p(p) = (\frac{1}{\sqrt{2\pi}\sigma})^3 4\pi p^2 e^{-\frac{p^2}{2\sigma^2}}. \tag{4.3}$$

Figure 4.2 plots the probability distribution function of Equation 4.3 for $\sigma = 20$. Unlike the zero-mean scalar noise distribution functions of individual sensor channels that peak at zero, the function here peaks near $p = 28$, or in general, at values of $p = \sqrt{2}\sigma$. What is more interesting however, is that at $p = 0$, the value of the probability distribution function equals zero. This suggests that as long as there is some non-zero scalar noise distribution in each individual channel of a color sensor, all image pixel readings from the sensor will almost certainly be corrupted by noise. One subtle point to note about Equation 4.3 before we proceed: For a given p value, the expression evaluates the *combined* probability densities of all length p vectors in the perturbation space. To obtain each length p perturbation vector's probability density value, we must divide the expression by $4\pi p^2$, the total "number" of length p vectors in the perturbation space. It is crucial that we clearly understand what $Pr_p(p)$ quantifies when performing the mathematical derivations below.

To compute the angular color noise distribution, $Pr_A(A)$, of a signal from its perturbation magnitude distribution (Equation 4.3), we shall first use the geometric configuration illustrated in Figure 4.3 to determine $Pr(\phi \leq A)$, the system's *cumulative* angular noise distribution function. Differentiating $Pr(\phi \leq A)$ with respect to A would then yield $Pr_A(A)$, the result we want. If the unperturbed color vector signal has length L, and $p(x) = \sqrt{L^2 + x^2 - 2Lx\cos A}$ is the magnitude of the *particular* color perturbation vector, then the color noise *cumulative* angular probability

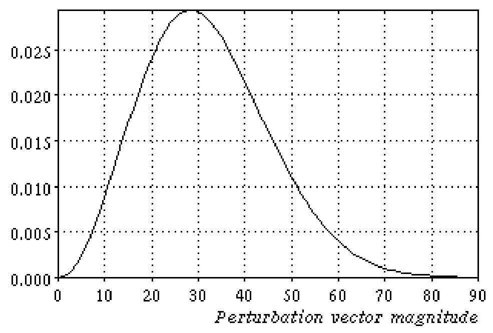


Figure 4.2: Perturbation magnitude (Equation 4.3) for $\sigma = 20$. In general, the function peaks at $p = \sqrt{2}\sigma$.

distribution would be:

$$Pr(\phi \leq A) = \int_{x=0}^{\infty} \int_{\phi=0}^{A} \frac{Pr_p(p(x))}{4\pi p(x)^2} 2\pi x^2 \sin \phi d\phi dx$$

Substituting the expression for Pr_p from Equation 4.3 into the integral above and expressing p in terms of x as we did in the previous paragraph, we get:

$$\begin{aligned} Pr(\phi \leq A) &= \int_{x=0}^{\infty} \int_{\phi=0}^{A} 2\pi (\frac{1}{\sqrt{2\pi}\sigma})^3 x^2 e^{-\frac{x^2+L^2+2Lx\cos A}{2\sigma^2}} \sin \phi d\phi dx \\ &= K - \frac{1}{2} \cos A e^{-\frac{L^2 \sin^2 A}{2\sigma^2}} (1 + erf(\frac{L \cos A}{\sqrt{2}\sigma})), \end{aligned}$$

where K is a term without A and $erf(t) = \sqrt{\frac{2}{\pi}} \int_0^t e^{t^2}\, dt$ is the *error function*. Finally, differentiating $Pr(\phi \leq A)$ with respect to A yields:

$$\begin{aligned} Pr_A(A) &= \frac{1}{2} \sin A e^{-\frac{L^2 \sin^2 A}{2\sigma^2}} (1 + \frac{L^2 \cos^2 A}{\sigma^2})(1 + erf(\frac{L \cos A}{\sqrt{2}\,\sigma})) \\ &\quad + \frac{1}{4} \sin 2A \sqrt{\frac{2}{\pi}} \frac{L}{\sigma} e^{-\frac{L^2}{2\sigma^2}}. \end{aligned} \tag{4.4}$$

Equation 4.4 clearly shows that the angular noise distribution of a color signal depends on both the color sensor's scalar noise strength (σ) and the signal's vector *magnitude* (L). Since color differences are quantified as angles in this thesis, it follows that for a fixed sensor channel noise level, σ, colors of brighter image pixels (larger L's) tend to get less severely affected by sensor noise than colors of dimmer pixels (smaller L's). This explains the assertion we made earlier in this section that the same set of sensor channel noise readings can be translated into *color* perturbations of

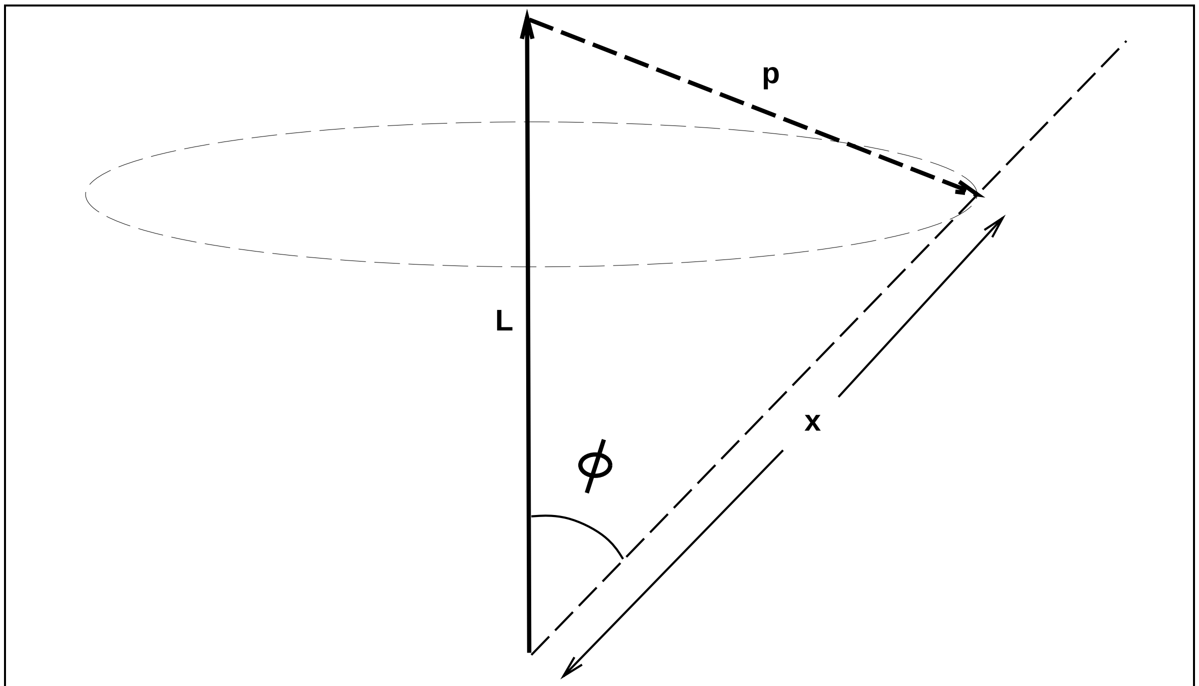


Figure 4.3: Geometric configuration for computing angular color noise distribution from color perturbation magnitude distribution.

different magnitudes.

Under normal lighting conditions where $L \gg \sigma$ for most image pixels, the second term in Equation 4.4 diminishes because of the $e^{-\frac{L^2}{2\sigma^2}}$ factor and only the first term dominates. Furthermore, considering only small values of A where the value of $Pr_A(A)$ is significant, we can make the following additional approximations: $\sin A \approx A$, $\cos A \approx 1$ and $(1 + erf(\frac{L \cos A}{\sqrt{2}\,\sigma})) = 2$. Equation 4.4 thus reduces to:

$$Pr_A(A) \approx Ae^{-\frac{L^2 A^2}{2\sigma^2}}(1 + \frac{L^2}{\sigma^2}), \tag{4.5}$$

which takes the form of a *Rayleigh Distribution*: $K\,xe^{-\frac{x^2}{2\sigma_r^2}}$ where $\sigma_r = \frac{\sigma}{L}$. Figure 4.4 plots our approximate angular color noise distribution (Equation 4.5) against our actual angular color noise distribution (Equation 4.4) for several values of L/σ. Notice how well the approximate distribution conforms with the actual distribution, even for the relatively small L/σ values we use.

A final but interesting observation to make about Equation 4.5 is that $Pr_A(A = 0) = 0$ as in a Rayleigh distribution. In other words, as long as scalar channel noise exists in a color sensor, we can expect the *color* of *all* image pixels to be corrupted and different from their original values. Also, the expected angular color displacement at each pixel is approximately $\sqrt{\frac{\pi}{2}}\sigma_r = \sqrt{\frac{\pi}{2}}\frac{\sigma}{L}$ radians.

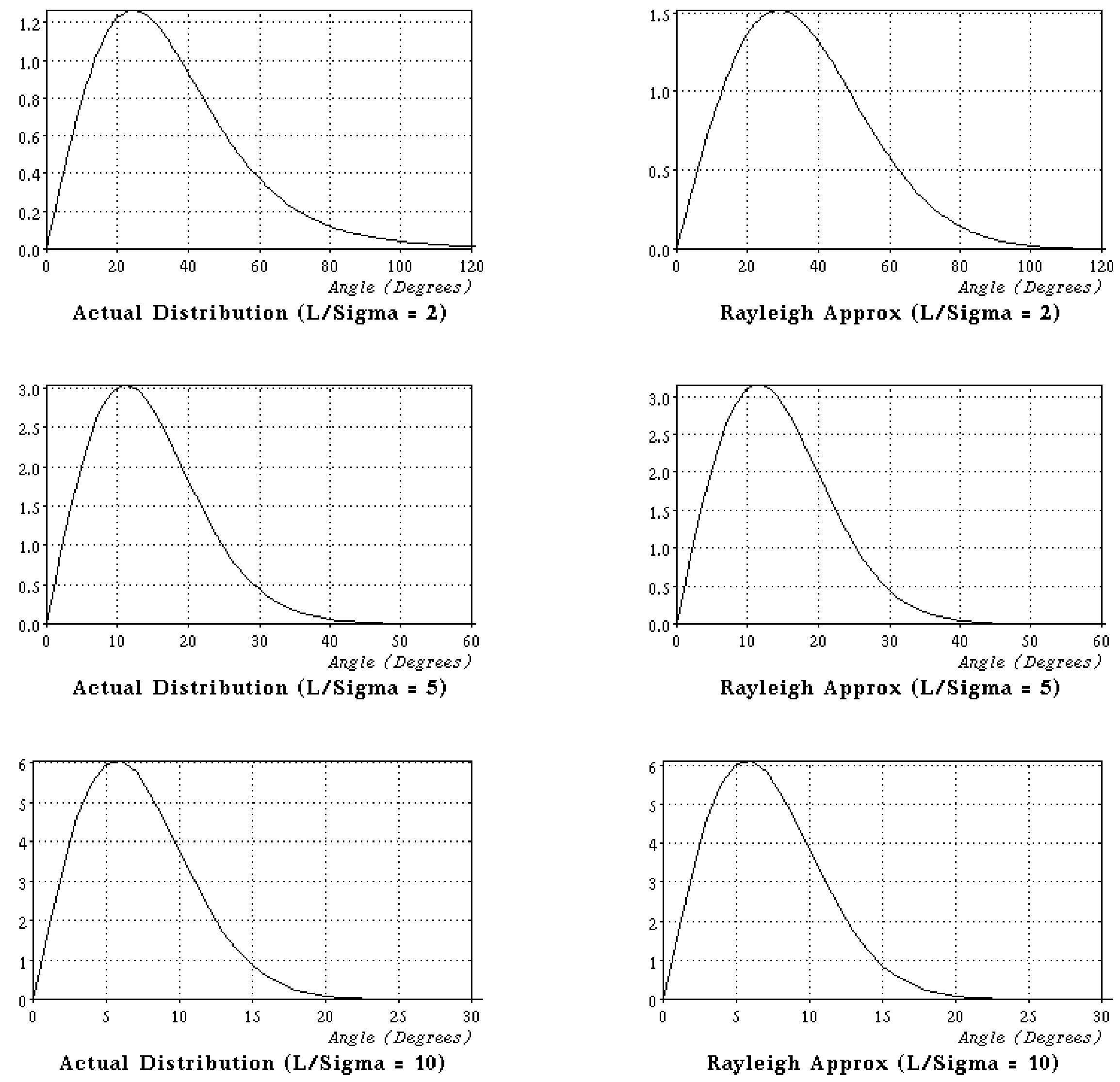


Figure 4.4: Actual and Approximate angular noise distributions for three values of L/σ. Top row: $L/\sigma = 2$. Center row: $L/\sigma = 5$. Bottom row: $L/\sigma = 10$. Notice that the approximation improves as the L/σ ratio increases.

4.2 Color Averaging

To reduce the strength of noise in an image, one normally uses an *averaging* or *smoothing* operation to filter away random fluctuations due to noise. As the name suggests, *averaging* computes local *mean* values at all image pixels to produce output readings that are statistically closer to the original uncorrupted image signal. For scalar signals bathed in zero-mean White Gaussian noise with variance σ^2, it can be analytically shown that taking the mean of a signal over k samples reduces the signal's noise variance to $\frac{\sigma^2}{k}$, or $\frac{1}{k}$ its original value (see for example [DeGroot 86]). The average noise strength, σ, in the original signal is thus also reduced to $\frac{\sigma}{\sqrt{k}}$, or $\frac{1}{\sqrt{k}}$ its initial value in the smoothed output.

4.2.1 The Color Mean

We shall now extend the concept of *averaging* as a noise reduction technique into the color domain. Like scalar averaging, this process involves computing an unknown quantity called the *mean* value of a sample of *colors*. Since the main purpose of averaging is one of reducing random noise fluctuations in an image, *color averaging* should therefore also seek to reduce random *angular* fluctuations in a color signal. Ideally, this means that the *mean* color vector we find should be a suitable "representative" of its *color* sample in some sense.

Most machine vision systems today treat *color representativeness* as having representative intensity values individually in the three separate chromatic channels. The formula below computes the arithmetic mean, μ, of a scalar valued sample: $x_1, \ldots, x_n$:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{4.6}$$

Suppose we use the traditional notion of *color representativeness* to compute the mean value of a *color* sample: $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$, where each $\boldsymbol{c}_i$ is a 3D vector $[r_i\ g_i\ b_i]^T$ of *Red*, *Green* and *Blue* channel intensities. Then the sample's *color mean* can be expressed as the color vector:

$$\begin{aligned} \boldsymbol{\mu} &= \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{c}_i \\ &= \frac{1}{n}\sum_{i=1}^{n} \begin{bmatrix} r_i \\ g_i \\ b_i \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n} r_i \\ \frac{1}{n}\sum_{i=1}^{n} g_i \\ \frac{1}{n}\sum_{i=1}^{n} b_i \end{bmatrix}. \end{aligned} \tag{4.7}$$

Unfortunately, this channel based definition of *color mean* is inconsistent with our ratio color representation in this thesis, because it allows two color samples containing the same *color* elements to produce mean vectors that represent different *colors*. The following example shows how this definition fails.

Since we are treating *color* as a normalized ratio of sensor channel intensities, we can replace any element, say $\boldsymbol{c}_1$, in the color sample above, with a scaled multiple of itself, $k\boldsymbol{c}_1$, and still get back a color sample with the same *color* elements. Computing the new sample's *color mean* using the approach in Equation 4.7, however, now yields:

$$\begin{aligned} \boldsymbol{\mu}_{new} &= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{c}_i + (k-1)\boldsymbol{c}_1 \\ &= \boldsymbol{\mu} + (k-1)\boldsymbol{c}_1. \end{aligned} \tag{4.8}$$

Clearly, the two mean vectors $\boldsymbol{\mu}_{new}$ and $\boldsymbol{\mu}$ can point in different directions, and hence represent different *colors*.

Rightfully, our color mean algorithm should compute "representative" *colors* as a whole and not just representative channel intensity values. It particular, it should be able to use color vectors having identical channel ratios interchangeably, because they all represent instances of the same *color*. This is possible if we adopt an angular interpretation of "color representativeness", similar to an important mathematical property that the arithmetic mean exhibits. It can be shown that for a scalar sample: $x_1, \ldots, x_n$, the arithmetic mean (μ) is the number, d, that minimizes the sample's *Mean Squared Error*:

$$E[(x_i - d)^2] = \frac{1}{n}\sum_{i=1}^{n}(x_i - d)^2. \tag{4.9}$$

This is one property that makes the arithmetic mean a good "average" measure, because it maintains as small a value difference as possible between all elements of the sample and itself.

The *mean* value $(\boldsymbol{\mu})$ of a color sample: $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$, can therefore be similarly defined as the vector, $\boldsymbol{d}$, that minimizes the expression:

$$E[\mathcal{A}^2(\boldsymbol{c}_i, \boldsymbol{d})] = \frac{1}{n}\sum_{i=1}^{n}\mathcal{A}^2(\boldsymbol{c}_i, \boldsymbol{d}). \tag{4.10}$$

where $\mathcal{A}(\boldsymbol{i}, \boldsymbol{j})$ is the angular color difference measure between vectors $\boldsymbol{i}$ and $\boldsymbol{j}$. The condition is analogous to that of Equation 4.9 except it now minimizes the *angular* Mean Squared Error between $\boldsymbol{\mu}$ and all of $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$. Physically, if each of $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$ were normalized into a vector of given length and replaced by an equal length rod of uniform density in the 3D space, then we can imagine $\boldsymbol{\mu}$ to be the system's axis of least inertia.

Equation 4.7 can be modified to compute *color mean* vectors in the minimum mean squared error sense. Minimizing squared angular differences is equivalent to weighing each color vector's *ratio* equally when computing the direction of $\boldsymbol{\mu}$. This can be done by normalizing the channel components of each color vector by the vector's own magnitude as in the summation below:

$$\begin{aligned} \boldsymbol{\mu} &= \frac{1}{n}\sum_{i=1}^{n}\frac{\boldsymbol{c}_i}{|\boldsymbol{c}_i|} \\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\sqrt{r_i^2+g_i^2+b_i^2}}\begin{bmatrix} r_i \\ g_i \\ b_i \end{bmatrix}, \end{aligned} \tag{4.11}$$

which we shall henceforth adopt as our *color mean* definition.

4.2.2 Color Averaging and Noise Reduction

The following mathematical analysis describes quantitatively how *angular noise* gets reduced by computing *color mean* vectors. Let us consider an averaging operation of n color vectors, $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$, whose lengths are $L_1, \ldots, L_n$ respectively. All n vectors are immersed in White Gaussian sensor channel noise of strength σ. Using Equation 4.5 together with our length and channel noise parameters, we get an approximate *angular* noise distribution for the color vector $\boldsymbol{c}_i$:

$$Pr_i(A) \approx Ae^{-\frac{L_i^2A^2}{2\sigma^2}}(1+\frac{L_i^2}{\sigma^2}), \tag{4.12}$$

for small values of A.

We shall now use Equation 4.11 to derive a similar expression for the *angular* noise distribution of the sample's color mean, $\boldsymbol{\mu}$. The normalization step in the summation yields, for each $\boldsymbol{c}_i$, a color vector of length 1 with individual sensor channel noise distributions $G(\frac{\sigma}{L_i}, n)$. Applying the *Linear Combination Property* for Gaussian distributed variables (see for example Chapter 5.6 of [DeGroot 86]) to the summation and division by n steps, we can show that the channel noise distribution for $\boldsymbol{\mu}$ will still be Gaussian in form and have a variance of $\frac{\sigma^2}{n^2}[\frac{1}{L_1^2}+\cdots+\frac{1}{L_n^2}]$. The resulting magnitude of $\boldsymbol{\mu}$ will also have an expected value of 1.

To translate $\boldsymbol{\mu}$'s channel noise and length parameters into an *angular* noise distribution, we substitute 1 for L and $\frac{\sigma}{n}\sqrt{\frac{1}{L_1^2}+\cdots+\frac{1}{L_n^2}}$ for σ into Equation 4.4. This gives us:

$$\begin{aligned} Pr_{\boldsymbol{\mu}}(A) &= \frac{1}{2}\sin Ae^{-\frac{L_\mu^2\sin^2 A}{2\sigma^2}}(1+\frac{L_\mu^2\cos^2 A}{\sigma^2})(1+erf(\frac{L_\mu\cos A}{\sqrt{2}\,\sigma})) \\ &\quad + \frac{1}{4}\sin 2A\sqrt{\frac{2}{\pi}}\frac{L_\mu}{\sigma}e^{-\frac{L_\mu^2}{2\sigma^2}}, \end{aligned}$$

where $\frac{1}{L_\mu} = \frac{1}{n}\sqrt{\frac{1}{L_1^2} + \cdots + \frac{1}{L_n^2}}$, which we can rewrite as: $L_\mu = \Theta(\sqrt{n}\, L_i) \approx \sqrt{n}\, L_i$, assuming the best possible scenario whereby all the original unnormalized color vectors have approximately the same length. Finally, considering once again only the small values of A where $Pr_{\boldsymbol{\mu}}(A)$ is significant, we get:

$$\begin{aligned} Pr_{\boldsymbol{\mu}}(A) &\approx Ae^{-\frac{L_\mu^2 A^2}{2\sigma^2}}(1 + \frac{L_\mu^2}{\sigma^2}) \\ &= Ae^{-\frac{nL_i^2 A^2}{2\sigma^2}}(1 + \frac{\sqrt{n}L_i^2}{\sigma^2}). \end{aligned} \tag{4.13}$$

Comparing the Rayleigh distributions of Equations 4.12 and 4.13, we see that before averaging, the *angular* noise distribution, $Pr_i(A)$, of the system, peaks at $A = \frac{\sigma}{L_i}$ with *mean* $\sqrt{\frac{\pi}{2}}\frac{\sigma}{L_i}$. After averaging, the resulting *angular* noise distribution, $Pr_{\boldsymbol{\mu}}(A)$, peaks at $A = \frac{\sigma}{\sqrt{n}L_i}$ with *mean* $\sqrt{\frac{\pi}{2n}}\frac{\sigma}{L_i}$. Color averaging over n samples thus reduces the strength of *angular* noise by a factor of $\sqrt{n}$ at best.

4.3 An Averaging Algorithm

Because unconstrained color averaging operates on image color at all pixel locations, actual color discontinuities get smoothed by the averaging process as well. This smoothing process degrades color boundaries by slowing down color changes across them. This in turn creates undesirable side effects for certain color operations, like edge detection, that assume sharp color changes across material boundaries.

Even if we were to consider only *Signal to Noise Ratios* and not absolute signal strengths at color boundaries, unconstrained color averaging generally does not help us enhance color discontinuities either. Suppose we choose a non-directional $n \times n$ pixel square kernel for our color averaging operation. Using results presented in the previous section, we can show that taking an n^2 sample color mean reduces the strength of angular color noise by a factor of n (or to a level of $\frac{1}{n}$ times its original value). Figure 4.5 summarizes next the operator's effect on the "slope" of a color boundary. If the discontinuity were ideal, then smoothing longitudinally across it with the same kernel transforms its *step* like profile into a *ramp* whose increments are $\frac{1}{n}$ times the magnitude of the original step difference. So, unconstrained averaging with a 2D filter attenuates both the signal and noise strengths of a color image by the same proportion, namely $\frac{1}{n}$.

4.3.1 A Markov Random Field Formulation

In this section, we present a different approach to color averaging that helps us overcome the boundary smoothing problem. The approach treats each color image as a 2-dimensional lattice of color val-

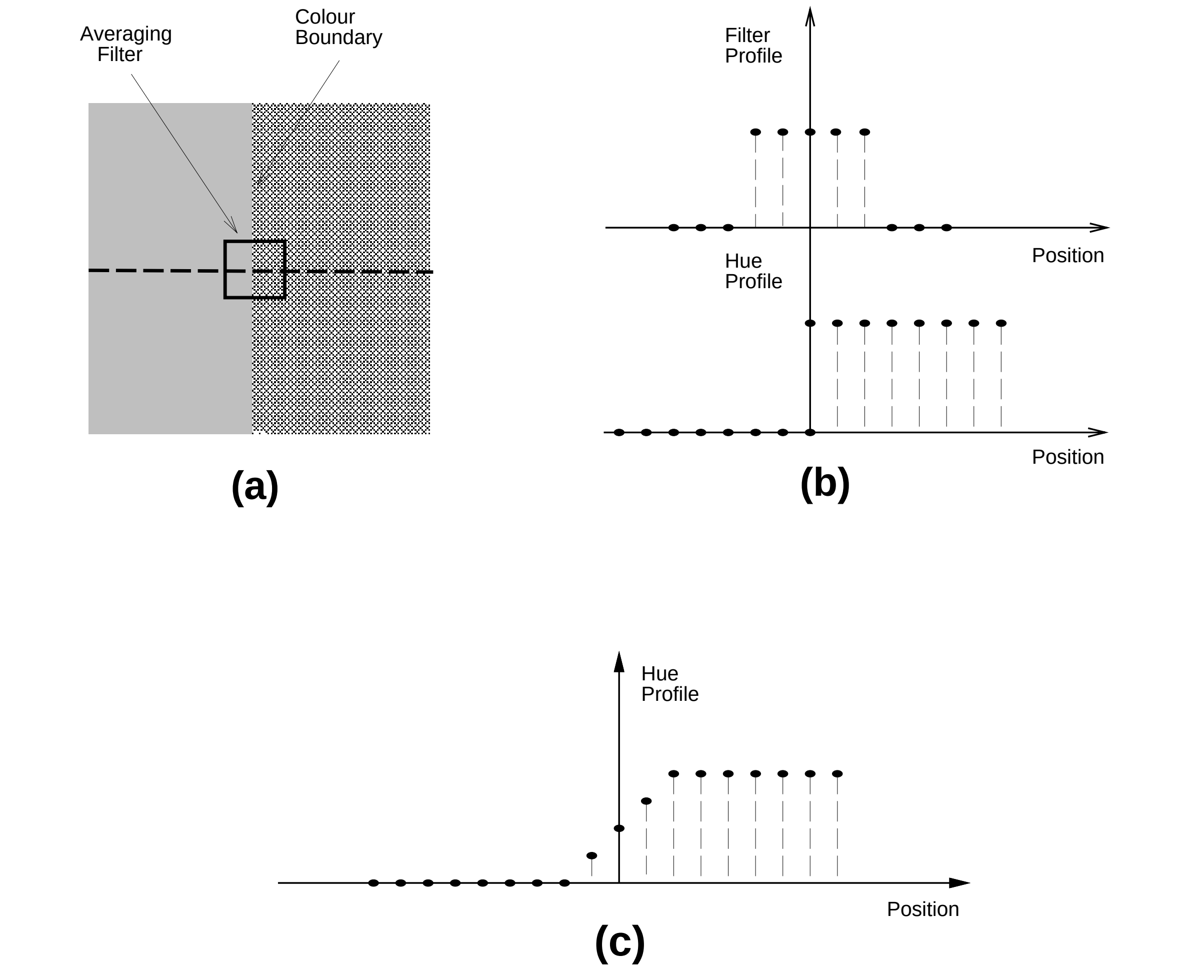


Figure 4.5: (a) Square averaging filter in the vicinity of a color boundary in 2 dimensions. (b) Discrete convolution of square filter and color hue profile along a longitudinal cross section of the color discontinuity (the dotted line in (a)). (c) Result of convolution. A ramp profile whose value increments are $\frac{1}{n}$ times the magnitude of the original step difference.

ues and attempts to fit piecewise smooth color hyper-surfaces to the image data within certain constraints. All this is done through a process known as *regularization* [Poggio Voorhees and Yuille 84], using a computational framework called Markov Random Fields (MRFs). Several others, notably Poggio et. al. [Poggio et. al. 85] [Poggio et. al. 87], Gamble and Poggio [Gamble and Poggio 87], Geman and Geman [Geman and Geman 84] and Blake and Zisserman [Blake and Zisserman 87], have employed similar techniques before to reconstruct image surfaces of other visual cues, such as depth and motion fields, from sparse and noisy data.

Briefly, MRFs associate an energy potential with each possible solution to a surface reconstruction problem. This energy function depends only on local pixel interactions within the reconstructed image surface. That is, the amount of energy each pixel contributes to the system's total energy function depends only on the final image values assigned to the pixel itself and to its immediate neighbours. The system's total energy is the sum of the energy potentials contributed by all pixels in the image. Normally, this quantity is small if all pixels in the reconstructed image have values that are similar to their immediate neighbours', and close enough to their original values. Finding a minimum energy solution to the system therefore amounts to smoothing pixel values for the image, while still maintaining the overall shape of the original surface. More importantly, MRF techniques are also able to integrate edge based information with region based smoothing because they contain a mechanism, called a *line process*, that curbs pixel-wise interaction across plausible boundaries. So, if we exploit this *line process* mechanism intelligently in our color averaging task, we can reduce noise fluctuations within image regions, while preserving sharp color changes across most color boundaries at the same time.

We now describe the MRF technique in greater detail as it applies to our color averaging problem. Given a color image, we first transform the image into a color vector field C_{xy} on a regular 2 dimensional lattice of points (x, y). Our goal then is to determine a solution field $\tilde{C}_{xy}$ which is

1. "close" enough to the original color field, C_{xy}, by some measure, and
2. locally "smooth" by some measure except at locations that correspond to plausible color boundaries in the image.

To formalize the above constraints, we introduce an energy function that the desired solution field, $\tilde{C}_{xy}$, should minimize. For each lattice location (x, y), the full energy potential is:

$$\begin{aligned} E_{xy} \;=\; & \mathcal{A}^2(\tilde{C}_{xy}, C_{xy}) \\ & + \alpha\,[(1-h_{xy})\,\mathcal{A}^2(\tilde{C}_{x+1,y}, \tilde{C}_{xy}) + (1-h_{x-1,y})\,\mathcal{A}^2(\tilde{C}_{xy}, \tilde{C}_{x-1,y}) \\ & \quad + (1-v_{xy})\,\mathcal{A}^2(\tilde{C}_{x,y+1}, \tilde{C}_{xy}) + (1-v_{x,y-1})\,\mathcal{A}^2(\tilde{C}_{xy}, \tilde{C}_{x,y-1})] \\ & + \gamma\,(h_{xy} + v_{xy} + h_{x-1,y} + v_{x,y-1}), \end{aligned} \tag{4.14}$$

where as before, A denotes the angular color difference measure of Chapter 3. h and v are the horizontal and vertical *line processes* that we hinted about earlier. They have a values 0 or 1 at every (x, y) lattice location, where 1 indicates the presence and 0 the absence of a color boundary. Qualitatively, Equation 4.14 can be interpreted as follows: The first term enforces "closeness" by penalizing solutions that are very different from the original data value. The second term favours "smooth" color transitions between neighbouring pixels, except across plausible color boundaries where the *line processes* h or v have values of 1. The third term introduces a penalty for each line process created, so that discontinuities are only declared where color changes are sharp enough to have arisen from color boundaries. Equation 4.15 gives us the total energy function of the entire system, which equals the sum of energy potentials over all lattice locations.

$$E_{total} = \sum_{x,y} E_{xy}. \tag{4.15}$$

In the past, only stochastic algorithms based on *Monte Carlo* and *simulated annealing* techniques were available to actually solve Equations 4.14 and 4.15 for their minimum energy configuration $\tilde{C}_{xy}$ [Marroquin 85] [Marroquin Mitter and Poggio 85]. Unfortunately, these methods are computationally very expensive and non-deterministic in behaviour. Also, they do not guarantee convergence although this problem hardly arises in reality. It was not only until recently, when Geiger and Girosi [Geiger and Girosi 89] developed a *mean field theory* based deterministic approximation to the stochastic algorithms above, that the MRF approach towards color averaging finally became a lot more feasible.

4.3.2 A Deterministic Approach

Our solution to the minimization problem above is based on another MRF deterministic approximation technique, developed by Hurlbert and Poggio for segmenting scalar hues [Hurlbert and Poggio 88] [Hurlbert 89]. Here, we are extending their algorithm to smooth 3D color vector fields. Basically, we discard the "closeness" term, $\mathcal{A}^2(\tilde{C}_{xy}, C_{xy})$, of Equation 4.14 from the energy potential, and incorporate the third (*line process*) term into the second ("smoothness"). The new energy function becomes:

$$\begin{aligned} E_{xy}^{New} \;\; = \;\; & \alpha[(1-h_{xy})\,Q(\mathcal{A}(\tilde{C}_{x+1,y}), \tilde{C}_{xy}) + (1-h_{x-1,y})\,Q(\mathcal{A}(\tilde{C}_{xy}, \tilde{C}_{x-1,y})) \\ & + (1-v_{xy})\,Q(\mathcal{A}(\tilde{C}_{x,y+1}, \tilde{C}_{xy})) + (1-v_{x,y-1})\,Q(\mathcal{A}(\tilde{C}_{xy}, \tilde{C}_{x,y-1}))], \end{aligned} \tag{4.16}$$

where $Q(x)$ is a zero-centered quadratic function, truncated to a constant when $|x|$ is above a certain value (see Figure 4.6).

Hurlbert and Poggio derived, for the scalar field case, an iterative algorithm that uses gradient descent to find stable minimum energy hue configurations. For color vector fields, the following

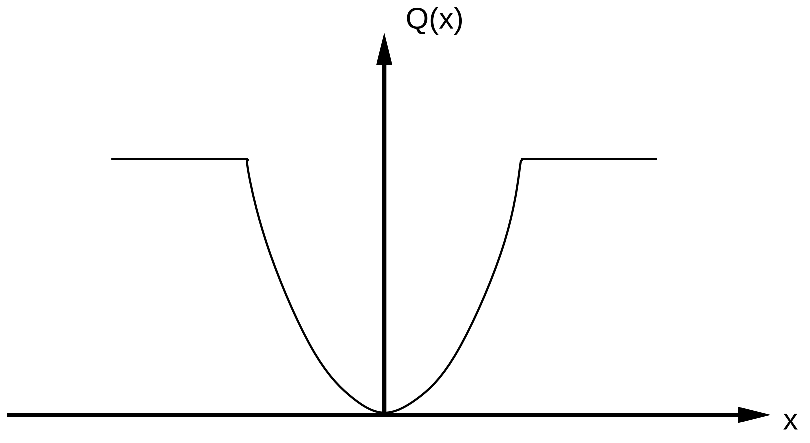


Figure 4.6: Quadratic function that implements the energy potential for deterministic MRF smoothing.

differential equation (the 3D analogue of Hurlbert and Poggio's scalar equation) governs successive changes in C_{xy}:

$$\frac{dC_{xy}}{dt} = -\alpha \frac{\partial E_{xy}^{New}}{\partial C_{xy}}. \tag{4.17}$$

Assuming none of location (x, y)'s immediate neighbours are separated from location (x, y) by line processes, then choosing an appropriate value for α and solving the system for values of C_{xy} at discrete times t yields:

$$C_{xy}^{t+1} = \texttt{MEAN}(C_{x+1,y}^{t}, C_{x-1,y}^{t}, C_{x,y+1}^{t}, C_{x,y-1}^{t}). \tag{4.18}$$

where `MEAN` denotes the *color mean* operation that we defined earlier.

Equation 4.18 describes an iterative algorithm that repetitively replaces the *color* of each image pixel with the *mean color* of its four immediate neighbours. Intuitively, the *mean* operation suppresses random angular noise fluctuations and propagates uniform *color* values across the image at the same time. To prevent smoothing across probable region boundaries, Hurlbert and Poggio's algorithm takes as input one or more edge maps, which it uses as line processes to contain averaging. In addition, it also disallows averaging between pixels whose hue differences are greater than some free threshold, T. The modified updating algorithm uses only neighbouring pixels having the same edge label as the central pixel, and whose colors are similar enough to the central pixel's color to compute new *color means*. This edge label and color similarity restriction, together with the original four nearest neighbour updating scheme, helps ensure that averaging only takes place among pixels that lie on the same side of a physical color discontinuity.

4.3.3 Simulating a Dynamic Line Process

Because Hurlbert and Poggio's averaging scheme uses the same input edge maps and threshold values to curb averaging throughout its entire iteration stage, we expect its final solution field to exhibit sharp discontinuities only where there are recorded edges, or where the field discontinuities are already sufficiently steep to begin with. This feature would be ideal if our edge map inputs were perfect or if suitable threshold values could be determined in advance, since the outcome would then be a set of smoothly colored regions, fully bounded by sharp color discontinuities. Unfortunately, most edge detection techniques today are not perfect and the edges they find often contain broken fragments [Beymer SM]. The task of determining suitable boundary threshold values for an image is also a difficult problem that still has not been satisfactorily solved. So, inter-region smoothing can still occur where there are edge gaps, and this only weakens the color contrast of the gaps as the number of iterations increases beyond a certain limit.

Traditional MRF algorithms (eg. *Monte Carlo* and *simulated annealing* techniques), on the other hand, tend to be less badly affected by initial edge defects because they contain a mechanism that dynamically updates line processes. New line processes at weak boundaries can subsequently be enabled if their discontinuities get sharper as the computation progresses. Similarly, existing spurious line processes due to data noise can be subsequently disabled, should their random fluctuations later diminish. The mechanism is therefore expected to "seal up" most edge gaps until the lattice of values attains a stable configuration, which is ideal for controlling the spread of uniform *color* values.

A similar "line process updating" mechanism can be easily incorporated into Hurlbert and Poggio's iterative averaging framework as follows: We interleave the operation of an edge detector with the existing *color* update process. The edge detector computes a new edge map for the current *color* field each time it is invoked. The algorithm then uses this new edge map as its updated "line process" for subsequent iterations until the next edge detector call.

We have implemented the full deterministic averaging algorithm above that approximates MRF smoothing with dynamic line processes. The edge detector used is a Canny *color* edge finder that responds to color *ratio* changes. A detailed description of the edge detector can be found in Chapter 5. Figures 4.7 and 4.8 show the changing edge maps that the algorithm produces on two natural occurring images. The algorithm computes a new edge map for each image once every 5 iteration updates. Although we ran the update procedure on each image for only 100 iterations, the results clearly confirm 2 important facts:

1. Local averaging does indeed smooth away random color fluctuations in an image, as the temporally decreasing number of spurious edges in the edge maps suggest. On surfaces with weak specularities, like the cups of Figure 4.8, the algorithm is able to smooth away the specularities as well.

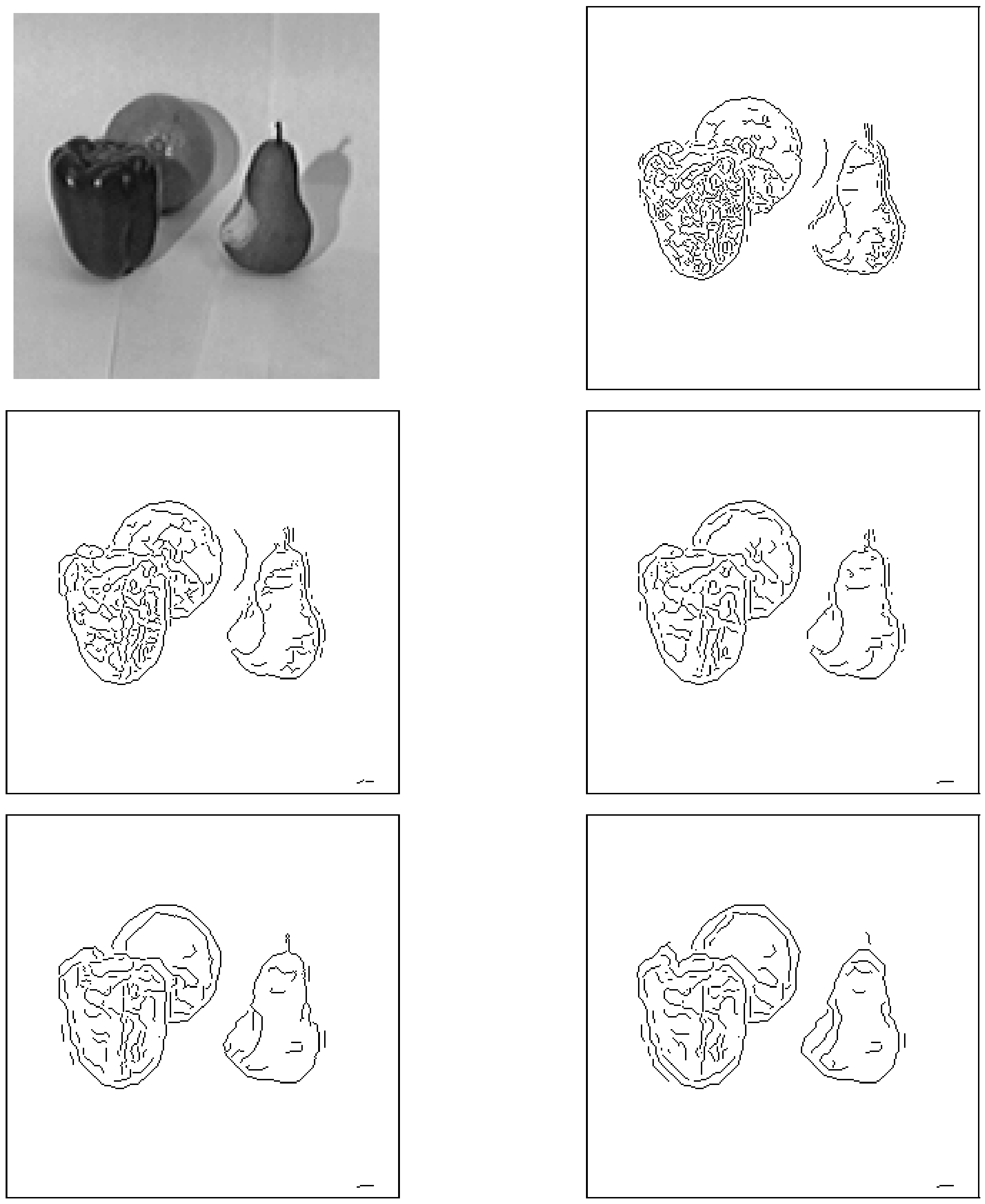


Figure 4.7: Changing edge maps produced by our deterministic MRF-like averaging scheme with a "line process updating" mechanism. Top Left: Original image. Top Right: Initial edge map. Center Left: Edge map after 10 iterations. Center Right: After 20 iterations. Bottom Left: After 50 iterations. Bottom Right: After 100 iterations.

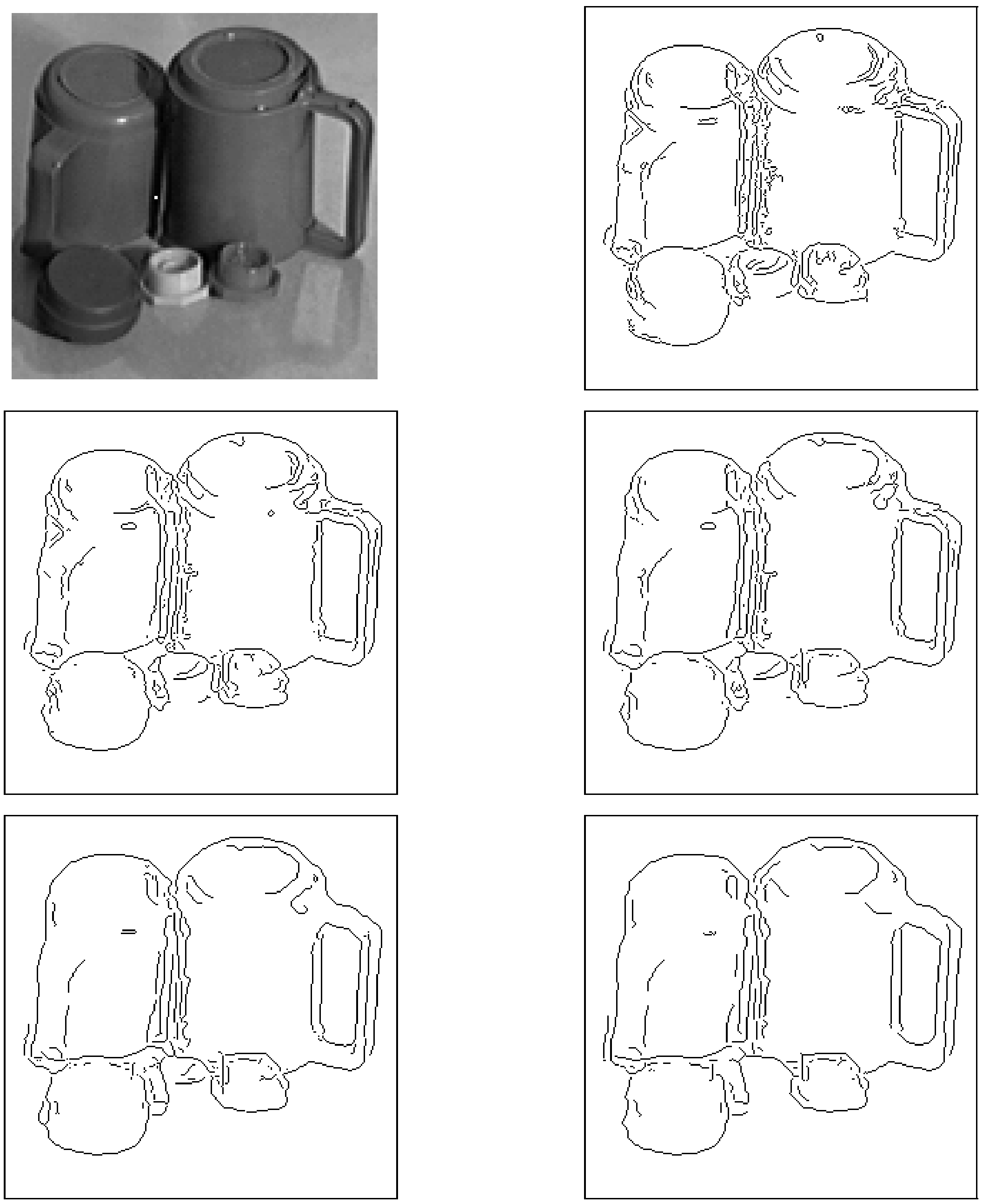


Figure 4.8: Changing edge maps produced by our deterministic MRF-like averaging scheme with a “line process updating” mechanism. Top Left: Original image. Top Right: Initial edge map. Center Left: Edge map after 10 iterations. Center Right: After 20 iterations. Bottom Left: After 50 iterations. Bottom Right: After 100 iterations.

2. Our simulated "line process updating" mechanism performs well in sealing up most broken edge fragments and maintaining sharp color contrasts across color boundaries. The edges that correspond to actual object boundaries in the image are extremely stable only because no smoothing could have taken place across them.

4.4 Color Median Filtering

Ideally, a machine vision system should be able to determine its own operating parameters automatically under most imaging conditions. Otherwise, it might have to depend on human help to recalibrate itself for different image inputs. This becomes a serious problem if we want to build vision systems that guide autonomous robots and vehicles, because these "autonomous" machines cannot be truly independent of human control.

So far, our color averaging approach assumes two sets of free parameters, namely mask dimensions (or in the deterministic MRF approximation case, the number of update iterations) and boundary thresholds. We shall defer the problem of determining suitable mask sizes for our color filters till Chapter 6, where we derive a formal relationship between image noise and signal detectability. As for now, we will just focus our attention on the color difference thresholds that help us define "line processes" for smoothing. In the deterministic MRF approximation scheme we implemented, these boundary threshold parameters appear absent because they actually reside within the color edge detection routines that update line processes. In the full MRF smoothing framework, these thresholds become the γ coefficient of Equation 4.14, a parameter that trades off color smoothness with boundary formation. Unfortunately, for most natural occurring images, choosing a suitable set of boundary thresholds can be a very difficult problem that involves considerable trial and error. Furthermore, a suitable set of thresholds for one part of an image may not even be suitable for another part of the same image because of differences in appearance.

4.4.1 Why Median Filtering?

This section introduces an alternative approach to color averaging, called *color median filtering*, that implicitly seeks its own natural boundary thresholds at different parts of the image. The technique replaces each pixel's value with the *median* value of its local neighbourhood, where mathematically, the median m of a set of values is such that half the members in the set are "greater" than m and half are "less" than m. Like mean averaging, median filtering also produces a "smoothed" image output in which the strength of random noise is reduced. More importantly however, the technique also exhibits a desirable side effect absent in mean averaging, namely, it preserves the sharpness of image boundaries and other image line features.

The effect of median filtering on noise and image features can be best illustrated with a 1 dimensional scalar signal processing example as in Figure 4.9. Here, we are convolving the signal

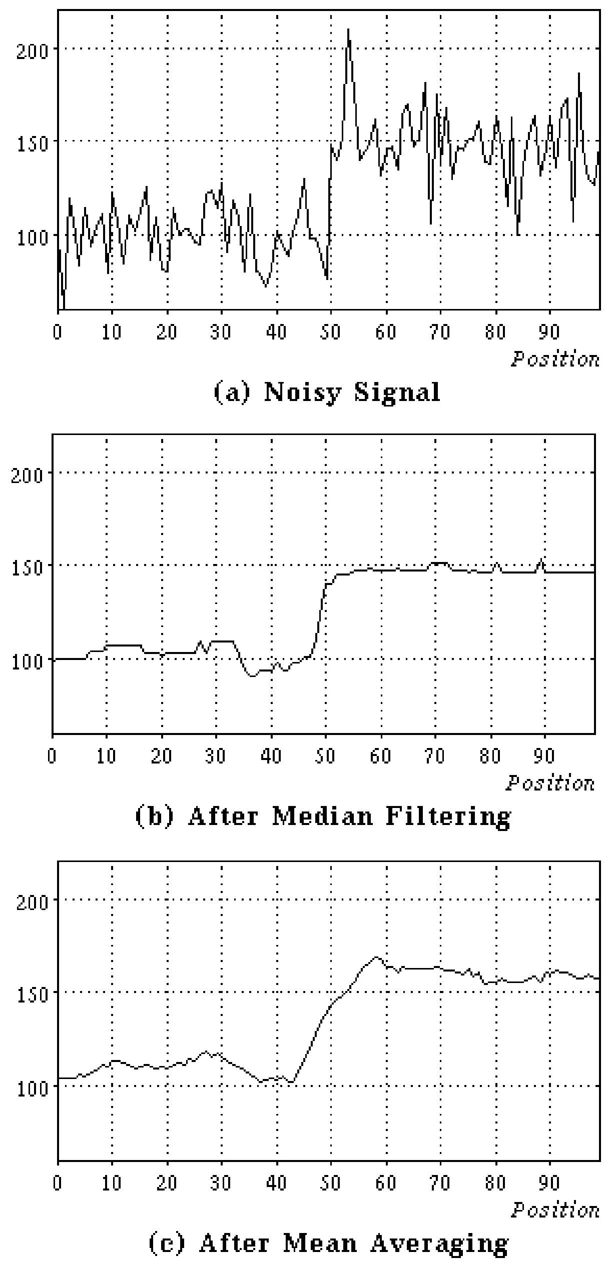


Figure 4.9: (a) Original noisy scalar signal. (b) Result after Median window filtering with a size 13 mask. Notice how outlying data points get rapidly smoothed and how boundary sharpness gets preserved by the operation. (c) Result after mean averaging with a size 13 mask.

with a median window filter of size 13; or in other words, we are replacing each pixel's value with the median value of itself and its 12 nearest neighbours. Notice two important features about the operation: First, the technique smoothes away arbitrarily large spikes like those near the step boundary very rapidly. The spikes' magnitudes do not affect the operator's smoothing effect much, because occasional data outliers do not change the value of the median. Second, the operation preserves the sharpness of edges. Since the median of a sample must be one of the sample's members, the median value at a boundary location must be an existing value on either side of the boundary. No "in between" values can be created at the junction, so we can still expect a slope that is as steep as the original boundary gradient.

4.4.2 Scalar Median Statistics

For scalar signals, we can show that median filtering is almost as effective a noise reduction technique as mean averaging. Statistically, both the local mean and median values of a noisy signal estimate the true value of the signal almost equally well. More precisely, the expected errors of the 2 quantities with respect to the signal's true value differ only by a small constant multiplicative factor, namely $\frac{\pi}{2}$. We present some relevant results leading to this conclusion in the following 2 paragraphs.

Suppose we use a size n local neighbourhood to determine the true signal value, S_{xy}, at each location (x, y) of an image. Let us also assume as before that the image is bathed in white Gaussian noise, $G(\sigma, s)$. The measurable signal at image location (x, y) will therefore be a symmetric probability distribution function:

$$f(s) = S_{xy} + G(\sigma, s),$$

centered at S_{xy}, because $G(\sigma, s)$ is itself a zero-centered symmetric probability distribution function. Assuming further piecewise constancy, such that all n local neighbourhood locations have the same signal distribution function $f(s)$, then as we have seen before, the size n sample mean, $\bar{S}^n_{xy}$, at image location (x, y) will be an S_{xy} centered Gaussian distribution function with variance $\frac{\sigma^2}{n}$.

To determine the size n sample median distribution, $\tilde{S}^n_{xy}$, at image location (x, y), we shall use a theorem on large sample properties of the median (see Section 9.8 of [DeGroot 86]). If the n nearest neighbours of location (x, y) have the same Gaussian distributed values, $f(s)$, as location (x, y) itself, then for large values of n, it can be shown that $\tilde{S}^n_{xy}$ will also have a Gaussian distribution with mean S_{xy} and variance $\frac{1}{4nG^2(\sigma,0)} = \frac{\pi\sigma^2}{2n}$.

We shall perform a similar analysis later to compare the smoothing effects of median window filtering and mean averaging for three dimensional color data.

4.4.3 The Color Median

Our earlier definition of the *sample median* implicitly assumes some total ordering on all sample members. Obviously, the same definition fails for color data, because no general total ordering notion exists for vector quantities. To conclude this section, we propose an alternative interpretation for the *color sample median*, which we will use in this thesis to synthesize color median filters.

Intuitively, the color median should exhibit the following 2 properties to fulfill its role as an averaging operator and a boundary preserver:

1. It must be a representative measure of its color sample in some sense. Only then can it have a smoothing effect on data like the sample mean.

2. It must have the same value as some member of the sample. Just as the scalar sample median preserves sharp step profiles by not introducing any "in between" scalar values at junctions, the color sample median can also preserve sharp color changes by not introducing any "in between" color values.

A reasonable *color median* interpretation follows from an algebraic property of the scalar sample median, similar to that of Equation 4.9 for the scalar sample mean. It can be shown that for a scalar sample: $x_1, \ldots, x_n$, the sample median is the member, m, that minimizes the *Mean Absolute Error* (M.A.E.) term:

$$E[|x_i - m|] = \frac{1}{n}\sum_{i=1}^{n} |x_i - m|. \tag{4.19}$$

We shall similarly define the *median* value of a color sample: $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$, as the sample member, $\boldsymbol{c}_m$, that minimizes the expression:

$$E[\mathcal{A}(\boldsymbol{c}_i, \boldsymbol{m})] = \frac{1}{n}\sum_{i=1}^{n} \mathcal{A}(\boldsymbol{c}_i, \boldsymbol{m}), \tag{4.20}$$

where as before, $\mathcal{A}(\boldsymbol{i}, \boldsymbol{j})$ denotes the angular color difference measure between vectors $\boldsymbol{i}$ and $\boldsymbol{j}$. Notice how both requirements are met in this definition of the color median. An $O(n^2)$ algorithm exists for finding a pixel's size n local neighbourhood color median vector. Basically, we compute the system's Mean Absolute Error for each sample member, $\boldsymbol{c}_i$, using Equation 4.20, and select the sample member, $\boldsymbol{c}_m$, that produces the smallest result.

Figure 4.10 compares the smoothed output of a Gaussian weighted color averaging filter with that of an equivalent[1] color median filter on a simple test image. The results are self explanatory. Notice from the hue maps how color median filtering preserves sharp hue changes in the original image, while smoothing away random hue fluctuations at the same time.

[1]The equivalent color median filter has a mask radius equal to the Gaussian averaging filter's standard deviation (σ).

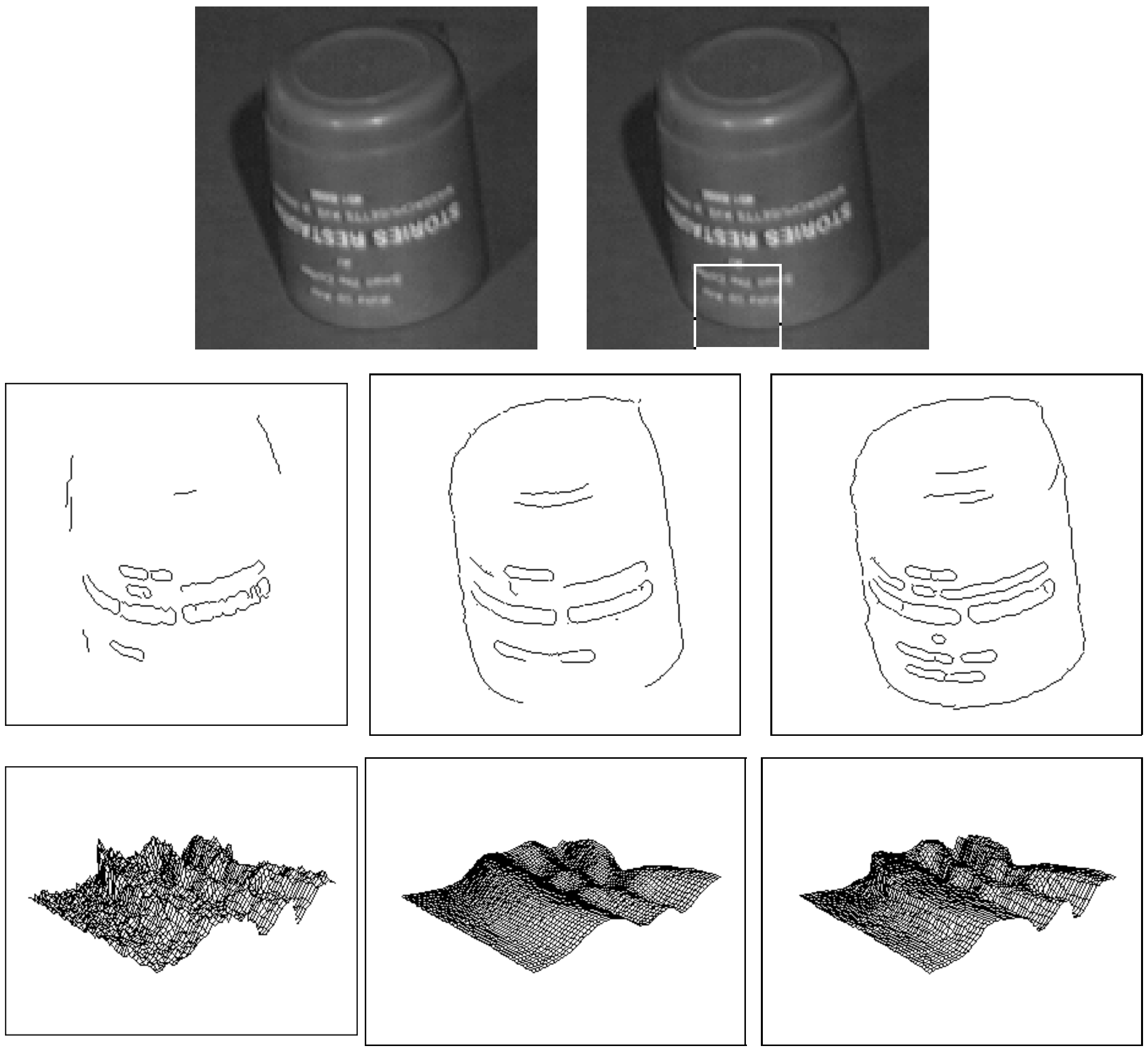


Figure 4.10: Effects of 2 passes of weighted color mean averaging ($\sigma = 3$) and color median filtering (window radius = 3) on a plastic cup image. Top Left: Original Image. Top Right: Region whose hue profile we are displaying. Second Row: Color edge maps of original image, image after color mean averaging and image after color median filtering. Bottom Row: `U` channel hue profile of bordered region in original image, in image after color mean averaging and in image after color median filtering.

4.4.4 Color Median Statistics

We conclude this chapter by informally analyzing the effectiveness of our color median algorithm as an angular noise reduction technique. Although our intention here is to seek an *average case* noise reduction measure for color median filtering, we are unfortunately unable to present such a result at this time because we still do not fully understand the 3-dimensional statistical nature of the color median. Instead, we shall just derive a result that may be interpreted as a *best case* noise reduction measure for color median filtering. Without further insight into the color median notion and its statistical behaviour, we can only guess for now that the *average case* color noise reduction measure differs from this derived *best case* measure by only a constant multiplicative factor.

Suppose we want to compute the color median of a noisy size n image patch whose individual pixel *colors* are $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n$. Our analysis assumes that the color median is the pixel *color*, $\boldsymbol{c}_i$, whose ratio is closest to that of the sample's true color mean[2]. Geometrically, $\boldsymbol{c}_i$ is the pixel *color* whose angular noise perturbation is smallest among all the sample *colors*: $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n$.

Recall from Equation 4.5 that a pixel's angular color noise perturbation, A, has the following approximate distribution:

$$Pr_A(A) \approx Ae^{-\frac{L^2A^2}{2\sigma^2}}(1 + \frac{L^2}{\sigma^2}),$$

with *mean*: $\bar{A} \approx \sqrt{\frac{\pi}{2}}\frac{\sigma}{L}$. Assuming that the ratio $\frac{L}{\sigma}$ is sufficiently large, we can further approximate the distribution above as:

$$Pr_A(A) \approx Ae^{-\frac{L^2A^2}{2\sigma^2}}\frac{L^2}{\sigma^2}. \tag{4.21}$$

From Equation 4.21, we can now derive the probability distribution function for A_n, the *minimum* value of n independent angular color perturbation (A) measurements:

$$Pr_{An}(A_n) \approx A_n e^{-\frac{nL^2A_n^2}{2\sigma^2}}\frac{nL^2}{\sigma^2}. \tag{4.22}$$

The distribution above has a *mean* value of $\bar{A_n} = \sqrt{\frac{\pi}{2n}}\frac{\sigma}{L}$, or $\frac{1}{\sqrt{n}}$ times the expected angular color perturbation of a single pixel. So, like color mean averaging, color median filtering over n pixels reduces the strength of angular color noise by a factor of $\sqrt{n}$ at best.

[2]In reality, this need not always be true, so the result we obtain for this analysis will be overly optimistic.

Chapter 5

Color Boundary Detection

Boundary detection has often been regarded as one of the key early level vision modules in many computer vision applications, for example model based object recognition [Grimson and Lozano-Perez 85a] [Grimson and Lozano-Perez 85b], shape from motion computation [Ullman 79] [Hildreth 83] and stereo image analysis [Marr and Poggio 79] [Grimson 81]. A critical step in abstracting image signals into symbolic tokens involves identifying and locating physical discontinuities in the scene. Although intensity edge detection, or the detection of grey level luminance changes, is useful for finding most physical discontinuities in the viewing environment, it alone cannot distinguish between the various types of physical discontinuities because almost all discontinuity types can bring about image intensity changes (see Table 1.1 of Chapter 1). Other visual cues are needed to classify the different types of discontinuities present.

Previous work by Rubin and Richards [Rubin and Richards 81] [Rubin and Richards 84] have shown that color is a useful cue for differentiating material changes from other types of scene discontinuities. Material boundaries are interesting because they outline object entities or meaningful parts of an object in the real world. As such, they can be used to speed up higher level vision processes, in particular object recognition, by grouping together portions of an image that most probably belong to a single world entity.

In Rubin and Richards' edge algorithm, color (or material) boundaries are detected by first finding image irradiance discontinuities in the three separate chromatic channels. The algorithm then performs a *Spectral Crosspoint* check and an *Ordinality* test at each edge location to determine whether the irradiance edge actually corresponds to a color or material boundary, details of which can be found in [Rubin and Richards 81]. It should be noted, however, that even in an image without specularities and other secondary effects, neither the *spectral crosspoint* condition nor the *ordinality violation* condition need to occur at material boundaries. Most color edge detection algorithms today [Hurlbert 89] [Lee 86] detect material boundaries by finding grey level discontinuities in image hue values:

$$\begin{aligned} u &= \frac{R}{R+G+B}, and \\ v &= \frac{G}{R+G+B}. \end{aligned}$$

These values are known to be relatively stable within regions of uniform material origin and different across material boundaries.

This chapter describes an autonomous color boundary detection technique that is based on the *normalized color ratio* representation scheme of Chapter 3. By autonomous, we mean that the technique is able to detect color boundaries independent of other visual cues, such as intensity boundaries. To quantify image color changes, the technique uses the *angular* color difference measure of this thesis, which we have reasoned to be the most sensitive and "well balanced" color discriminating measure for our chosen color representation scheme and the *pigmentation model* of material composition. We can therefore expect the technique to produce optimally "correct" edge responses, where "correctness" means successfully marking true color (or material) boundaries and not wrongly marking false boundaries, like pure intensity discontinuities due to shadows or orientation changes.

Because color data is sensed separately as grey level intensities in the three chromatic channels and combined multiplicatively as ratios, noise effects tend to get amplified when computing color, as discussed in Chapter 4. This usually results in color boundaries that are more badly broken and less well shaped than their intensity counterparts of the same image. Fortunately, most color discontinuities in real scenes also correspond to intensity discontinuities that can be independently detected by intensity edge detectors. The last section of this chapter describes a mechanism that uses intensity edges to reconstruct the color edges detected by our color edge finder. The algorithm takes as input a color edge map with its intensity edge counterpart, and produces an integrated color edge map whose edges are better connected and localized due to the intensity cues.

5.1 A One-Dimensional Boundary Detection Formulation

Most edge detection techniques today take place in several stages, the first of which usually enhances probable image boundary locations in some fashion. Subsequent stages are then used to mark and link together discontinuities locations in the image to form linear edge features. The theoretical emphasis of our work will be on this first stage of *color* boundary detection, namely the problem of enhancing probable color image boundaries. Although most later color edge detection stages are also interesting processes that deserve special attention in their own right, they are very similar in nature to their boundary detection counterparts of other visual cues, and will therefore not be discussed here in any detail. It is this first color boundary enhancement stage that actually

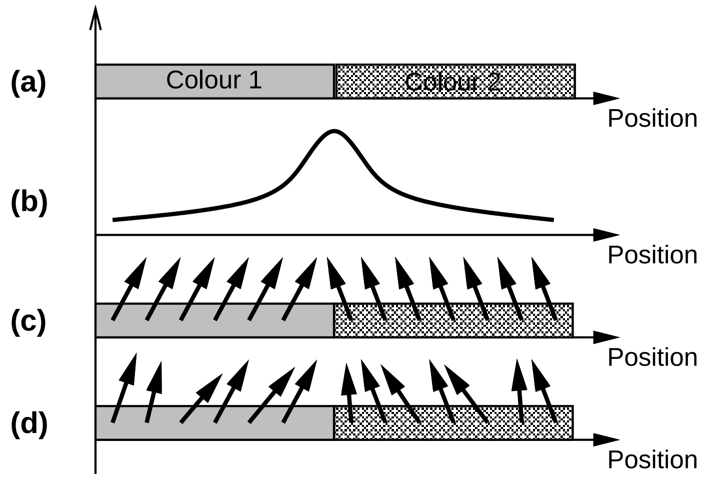


Figure 5.1: (a) Symbolic description of a color step edge. (b) Desired form of response after convolution of spatial operator with color step edge profile. (c) A linear array of color vectors, representing a color step edge profile. (d) Effect of sensor noise on the color vector array representation of (c).

distinguishes our color boundary detection technique from other color edge detection algorithms, or for that matter, edge detectors of any other visual cue.

The basic design problem is as illustrated in Figure 5.1, where we want to highlight sudden color changes in an image, like the "step" color profile of part (a). Let us assume, for the time being, that the "step" profile is ideal, or in other words, the *colors* are perfectly uniform on both sides of the discontinuity. To perform the discontinuity enhancement task, we "convolve" the image signal with some spatial operator to produce a scalar response that peaks at the color junction. Our objective is to derive such a spatial operator, that upon "convolving" with a *color* field, produces a *scalar* output pattern somewhat similar in form to Figure 5.1(b).

The rest of this section describes how color "convolution" can be transformed into a mathematically computable operation on vector arrays. Because we are treating *color* as 3-dimensional vector values in the RGB color space, the color profile of Figure 5.1(a) can be quantitatively represented as a linear array of 3D color vectors (see Figure 5.1(c)). For ideal "step" profiles, vector orientation changes should only occur at color boundaries. The desired convolution operator should therefore be one that produces scalar response peaks only where there are sudden orientation changes in the color vector field. Notice that in this formulation, pure grey-level intensity changes that only affect the *magnitude* of color vectors do not give rise to color boundaries.

Under non-ideal imaging conditions, sensor noise corrupts the color profile of Figure 5.1(a), so that the image does not register perfectly uniform color values on both sides of the "step" edge. Computationally, these noise effects appear as small directional perturbations in the profile's color vector field representation, see Figure 5.1(d). It is important that the desired convolution operator ignores these noise induced vector orientation changes in the image signal, otherwise the output will be cluttered with many false color boundaries.

5.2 A Color Edge Operator

Most existing edge detection algorithms choose either the first or second difference between neighbouring image pixels as an appropriate quantity for accentuating intensity changes. Basically, the difference operators transform the image into a more convenient representation for extracting discontinuities. A significant intensity change gives rise to a peak in the first difference and a zero-crossing in the second difference, both of which can be straightforwardly identified by simple algorithms. An excellent survey of these difference based measures for edge detection can be found in [Hildreth 84].

5.2.1 Enhancing Color Discontinuities

The color edge operator presented in this section emulates the performance of a grey-level first difference operator on scalar image fields. First differences are appropriate for our particular problem formulation because they produce outputs that peak at image discontinuities — the type of response that our subsequent edge processes expect. Also, it turns out that the color *first difference* computation framework can be easily extended to emulate the behaviour of other "first difference like" operators, for example the Gaussian first-derivative operator. Intensity edge detection results have shown that the Gaussian first-derivative operator produces very stable edge detection results even in the presence of noise.

The theory behind using first differences for enhancing discontinuities can be best explained in the continuous scalar spatial domain. Following through the derivation steps leads to an analogous discontinuity enhancement process for color data. Given a scalar function $f(x)$, differentiating with respect to space (x) yields the *gradient* of f:

$$f'(x) = \lim_{dx \to 0} \frac{f(x+dx) - f(x)}{dx}, \tag{5.1}$$

whose *absolute value* is locally maximum where the slopes are steepest, namely at sharp discontinuities. For spatially discrete functions like image arrays, the *gradient* expression of Equation 5.1 can be approximated by the function's first difference:

$$f_1(x) \approx f'(x)\big|_{dx=1} = f(x+1) - f(x), \tag{5.2}$$

which, for later convenience, we shall re-express as:

$$f_1(x) = sign[f(x+1) - f(x)]|f(x+1) - f(x)|. \tag{5.3}$$

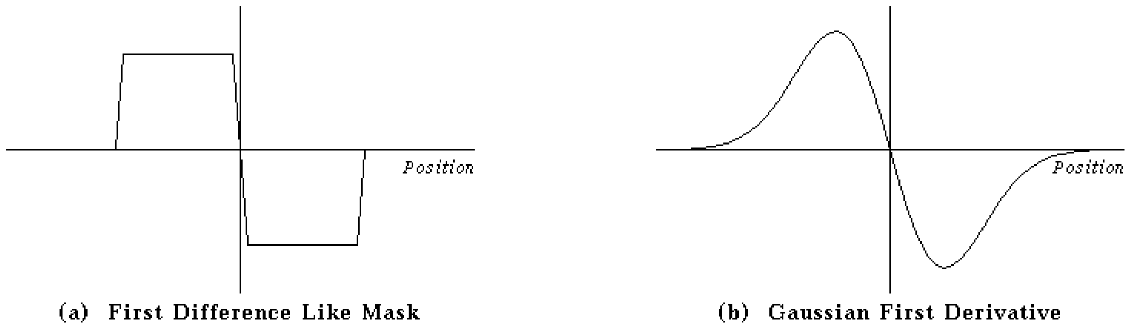


Figure 5.2: (a) A basic *first difference* profile. (b) The Gaussian first derivative — a "first difference" like edge mask with a near optimal detectability-localization product.

Equation 5.3 defines the scalar first difference operator as a product of two terms: A *sign* term, $sign[f(x+1) - f(x)]$, indicating the direction of the local slope, and a *magnitude* term, $|f(x+1) - f(x)|$, showing the amount of local intensity change in f. Only the *magnitude* term, $|f(x+1) - f(x)|$, is needed for enhancing discontinuities, since the *absolute* first difference is always a local maximum at all discontinuity locations. In the color domain, this *absolute* first difference translates into the angular color difference measure of Chapter 3, which quantifies *absolute* differences between *colors*. Therefore, to enhance all color discontinuities in an image, $\boldsymbol{f}(x)$, we can compute the following quantity:

$$|\boldsymbol{f}_1(x)| = \mathcal{A}(\boldsymbol{f}(x+1), \boldsymbol{f}(x)), \tag{5.4}$$

which is conceptually equivalent to computing an *absolute* gradient for color values.

5.2.2 Extensions for "First Difference Like" Mask Patterns

Although the first difference operator outputs very precise image discontinuity locations in its edge maps, its small local support makes it very sensitive to random signal fluctuations. This often gives rise to many spurious edge fragments throughout the image. Edge operators with large local supports, on the other hand, tend to produce results that still remain stable under image noise, but at the expense of more precise localization. Ideally, we would like to extend the *color first difference* concept to emulate other "*first difference* like" mask patterns as well, so that color boundary detection can be made into a more stable process. Figure 5.2(a) shows the basic profile of a scalar *first difference* operator, which is a zero-centered odd symmetric function of 2 rectangular boxes. We shall consider a mask pattern to be "first difference like" if it is also an odd symmetric function with only one zero-crossing at the origin. Generally, "first difference like" mask patterns make suitable operators for enhancing step signal discontinuities.

Figure 5.2(b) shows a "first difference like" Gaussian profile:

$$G_1(x,\sigma) = \frac{x}{\sqrt{2\pi}\sigma^3} e^{-\frac{x^2}{2\sigma^2}} \tag{5.5}$$

which was proven by Canny [Canny 83] to be a very close approximation to his optimal[1] step edge mask. For scalar signals, $f(x)$, each application of the mask at some image point, $x = x_o$ computes a weighted sum of the signal values under the mask window.

$$G_1(x,\sigma) * f(x)|_{x=x_o} = \sum_{i=-\infty}^{\infty} G_1(i,\sigma) f(i - x_o), \tag{5.6}$$

Since the left half of the mask is totally positive and the right is totally negative in value, (and the image signal is always non negative), the computation can also be performed by taking the *absolute* weighted sums of the two halves separately and then subtracting one result from the other. In other words:

$$G_1(x,\sigma) * f(x)|_{x=x_o} = \left|\sum_{i=1}^{\infty} G_1(i,\sigma) f(i - x_o)\right| - \left|\sum_{i=-\infty}^{0} G_1(i,\sigma) f(i - x_o)\right| \tag{5.7}$$

which upon closer inspection, equals:

$$\sum_{i=1}^{\infty} |G_1(i,\sigma)| f(i - x_o) - \sum_{i=-\infty}^{0} |G_1(i,\sigma)| f(i - x_o). \tag{5.8}$$

The expression has an *absolute value* of:

$$\left|\sum_{i=1}^{\infty} |G_1(i,\sigma)| f(i - x_o) - \sum_{i=-\infty}^{0} |G_1(i,\sigma)| f(i - x_o)\right|, \tag{5.9}$$

which, like the *magnitude* component of Equation 5.3, is a difference of 2 weighted-sum terms separated at the mask center. Both quantities produce local response peaks at signal discontinuities.

Equation 5.9 suggests a possible interpretation for a "first difference like" *color* edge mask. Because the two summation terms total (and in some sense, normalize) signal values on both halves of the mask, they can be replaced by similar *color weighted average* terms, a description of which can be be found in Chapter 4. The subtract operation that computes *absolute differences* between the 2 summations corresponds to the *angular difference measure* in the color domain. The application of a "first difference like" mask to a color signal, $\boldsymbol{f}(x)$, can therefore be computed as follows:

[1]Canny's optimality criterion is a detection-localization product.

$$|G_1(x,\sigma) * \boldsymbol{f}(x)|_{x=x_o} = \mathcal{A}(\sum_{i=1}^{\infty} |G_1(i,\sigma)| \frac{\boldsymbol{f}(i-x_o)}{|\boldsymbol{f}(i-x_o)|}, \sum_{i=-\infty}^{0} |G_1(i,\sigma)| \frac{\boldsymbol{f}(i-x_o)}{|\boldsymbol{f}(i-x_o)|}). \tag{5.10}$$

Intuitively, the result measures weighted average color differences across adjacent pixel patches, instead of simple color differences between adjacent pixels.

5.3 Implementation Details

We have implemented a serial version of our color boundary detection algorithm for the Symbolics Lisp Machines and a parallel version for the Connection Machine. The task essentially involves incorporating a color boundary enhancement front end into the framework of an existing Canny intensity boundary detector. The following is an abstract account of the design decisions we made and some heuristics we adopted in our implementation.

5.3.1 Operating in Two Dimensions

Until now, all our analysis in this chapter has assumed that images are one dimensional *color* signals of RGB vector values. For two dimensional color images, $\boldsymbol{f}(x,y)$, an edge point also has an orientation in addition to its two-dimensional position co-ordinate. We shall define the terms *gradient direction* to mean the image direction where the *color gradient* is steepest, and *edge orientation* as tangent to the gradient direction. To consistently enhance image locations such that steeper edges always appear stronger in the output than gentler edges regardless of orientation, we must be able to compute the *color gradient* at each edge point in its gradient direction. An inexpensive method of doing this is to resolve *color gradients* into spatially orthogonal components as if they were intensity gradients.

If we assume that color changes are reasonably smooth near color boundaries, then we can approximate the magnitude and direction of *color gradients* using color changes in two fixed directions. Our implementation uses a one dimensional "first difference like" mask to compute *partial* color gradients, $\boldsymbol{f}_x(x,y)$ and $\boldsymbol{f}_y(x,y)$, in the horizontal and vertical image directions, where:

$$\begin{aligned} \boldsymbol{f}_x(x,y) &= \mathcal{A}(\boldsymbol{f}(x+1,y), \boldsymbol{f}(x-1,y)) \\ \boldsymbol{f}_y(x,y) &= \mathcal{A}(\boldsymbol{f}(x,y+1), \boldsymbol{f}(x,y-1)). \end{aligned}$$

We can then determine the color gradient magnitude, $\boldsymbol{f}_1(x,y)$, from its partial components as follows:

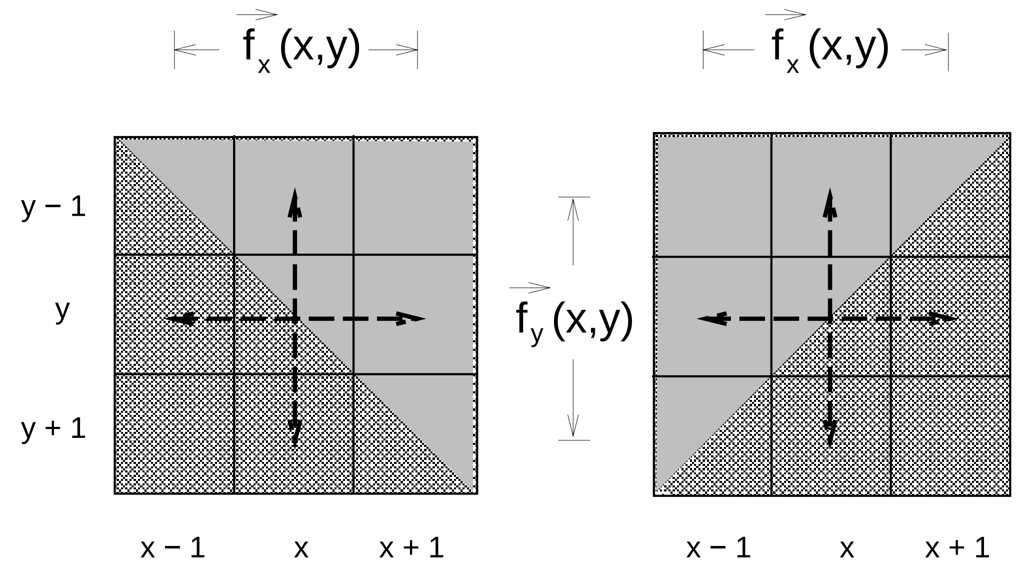


Figure 5.3: The same pair of partial color gradient components can be generated by gradients of the same magnitude in two possible directions.

$$\boldsymbol{f}_1(x,y) = \sqrt{(\boldsymbol{f}_x(x,y))^2 + (\boldsymbol{f}_y(x,y))^2}. \tag{5.11}$$

A slightly more complicated process is needed for determining color gradient and edge directions. Figure 5.3 shows an example of how two different color edge profiles can give rise to the same *partial gradient* pair, f_x and f_y. We cannot distinguish between the two edge profiles by just examining the values of f_x and f_y, because f_x and f_y are both *absolute* values. In other words, there is no such notion as positive or negative value changes in color, as there is in intensity and other scalar quantities. To resolve the ambiguity, our implementation computes both possible gradient directions from the absolute components f_x and f_y. It then checks the color difference along both directions and takes the "steeper" direction as the gradient direction.

5.3.2 Directed Color Gradients

The operations described so far detect color discontinuities in images. Ideally, these operations should produce edge maps that mark only those pixels that lie along the boundaries between *color* regions. Real images, however, often contain noise that fragments true boundaries and creates spurious discontinuities in edge maps. Thus, edge detection operations are typically followed by edge linking procedures, designed to join together edge pixels into continuous line features.

Most edge linking techniques today make use of *directed gradients* as one of the principle properties for establishing similarity between adjacent edge points (see for example Chapter 7.2 of [Gonzalez and Wintz 87]). For intensity images and maps of other scalar quantities, the *directed*

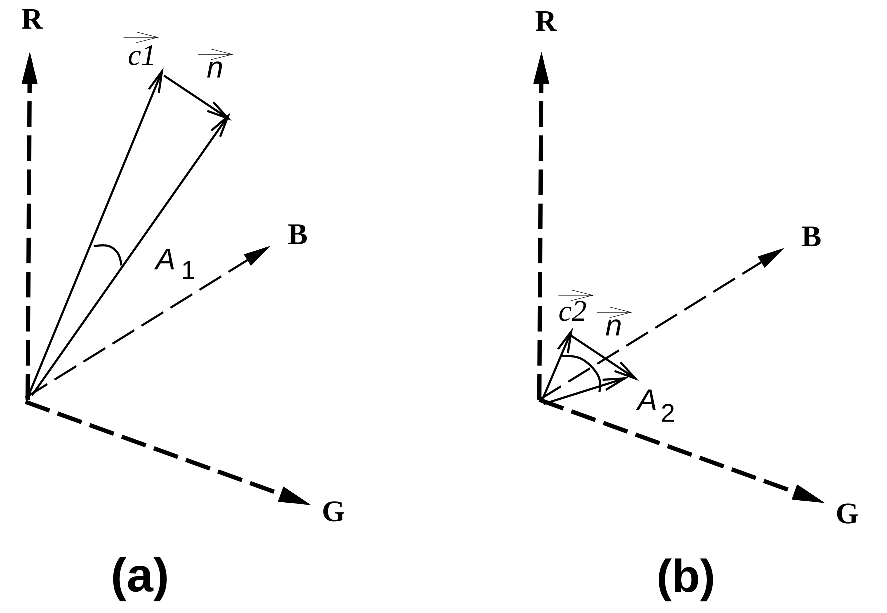


Figure 5.4: (a) Noise displaces a bright color vector, $\boldsymbol{c}_1$, by a small angle A_1. (b) The same noise signal displaces a dim color vector, $\boldsymbol{c}_2$, by a large angle A_2, where $A_2 \gg A_1$.

gradient points "downhill" where the "slope" is steepest. That is to say, it points in the gradient direction with an additional constraint that it runs from its higher value end to its lower value end.

Our treatment of *color* has nothing truly equivalent to the scalar *directed gradient* notion, because there is no "greater than" relationship between *color* values. In order to make use of existing edge linking techniques for our color edge detector, we adopt the convention that *directed color gradients* point from color regions of higher image intensity values to color regions of lower image intensity values, in the direction of greatest color change. Since real images rarely contain adjacent isoluminant color regions, it turns out that our heuristic works reasonably well.

5.3.3 Magnitude Scaling

Because white noise is usually uniformly distributed in strength throughout an image, dim color vectors tend to experience larger noise induced directional perturbations than bright color vectors. Figure 5.4 illustrates why this is so. Since we are using angular differences as a color difference measure, we can expect to detect erroneously large *color* differences due entirely to white noise in dim image areas. These large color perturbations get wrongly enhanced by our difference operators during edge detection, giving rise to undesirable spurious line fragments in the final color edge map.

Our current implementation explores heuristics for ignoring these false color discontinuities. Basically, we want to de-emphasize angular color differences between dim color vectors, while preserving color discontinuities between bright image pixels. To do this, we scale the angular

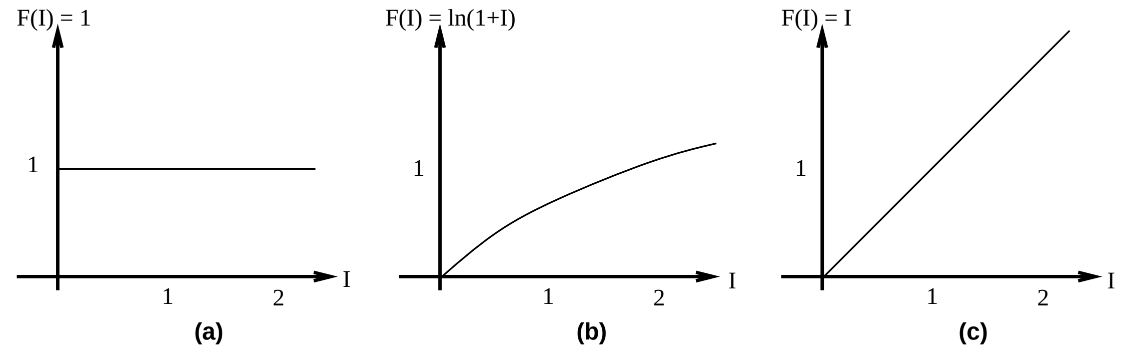


Figure 5.5: Scaling functions for compensating noise effects: (a) $F(I) = 1$, or no scaling. (b) $F(I) = \ln I$, logarithmic scaling. (c) $F(I) = I$, linear scaling.

difference output of the edge enhancement stage by some non-decreasing local intensity function, so that difference measurements in bright areas get weighted more than difference measurements in dim areas. Figures 5.6 and 5.7 display the edge results we get using three different scaling functions. In the two test images, we obtained best results in terms of preserving true edges and eliminating false responses using a logarithmic scaling function, as shown in Figure 5.5(b). Qualitatively, a logarithmic function seems to be ideal for our scaling purpose because they strongly penalize color perturbations where intensities (I) are low, but do not overly accentuate color differences in high intensity regions at the expense of color differences in moderate intensity regions.

5.4 Luminance Edges – Integrating Other Visual Cues

The ability to intelligently integrate information from different visual cues is perhaps one of the key reasons why biological vision systems today are still a lot more robust than current artificial vision systems. While early vision processes give independent information about physical discontinuities and surfaces in the scene, visual integration combines information provided separately by different visual modules to construct scene descriptions that are more complete and more reliable. Researchers like Gamble and Poggio [Gamble and Poggio 87] have devoted much work to the problem of integrating information from different visual cues like depth, motion and shading. In this section, we explore the possibility of integrating boundary based information from intensity and color data.

5.4.1 The Role of Luminance Edges

As noted by Hurlbert [Hurlbert 89], luminance or intensity edges are excellent visual cues for enhancing and locating *color* discontinuities. Because noise effects in the three chromatic channels combine multiplicatively as ratios when computing color, color boundaries found by segmenting

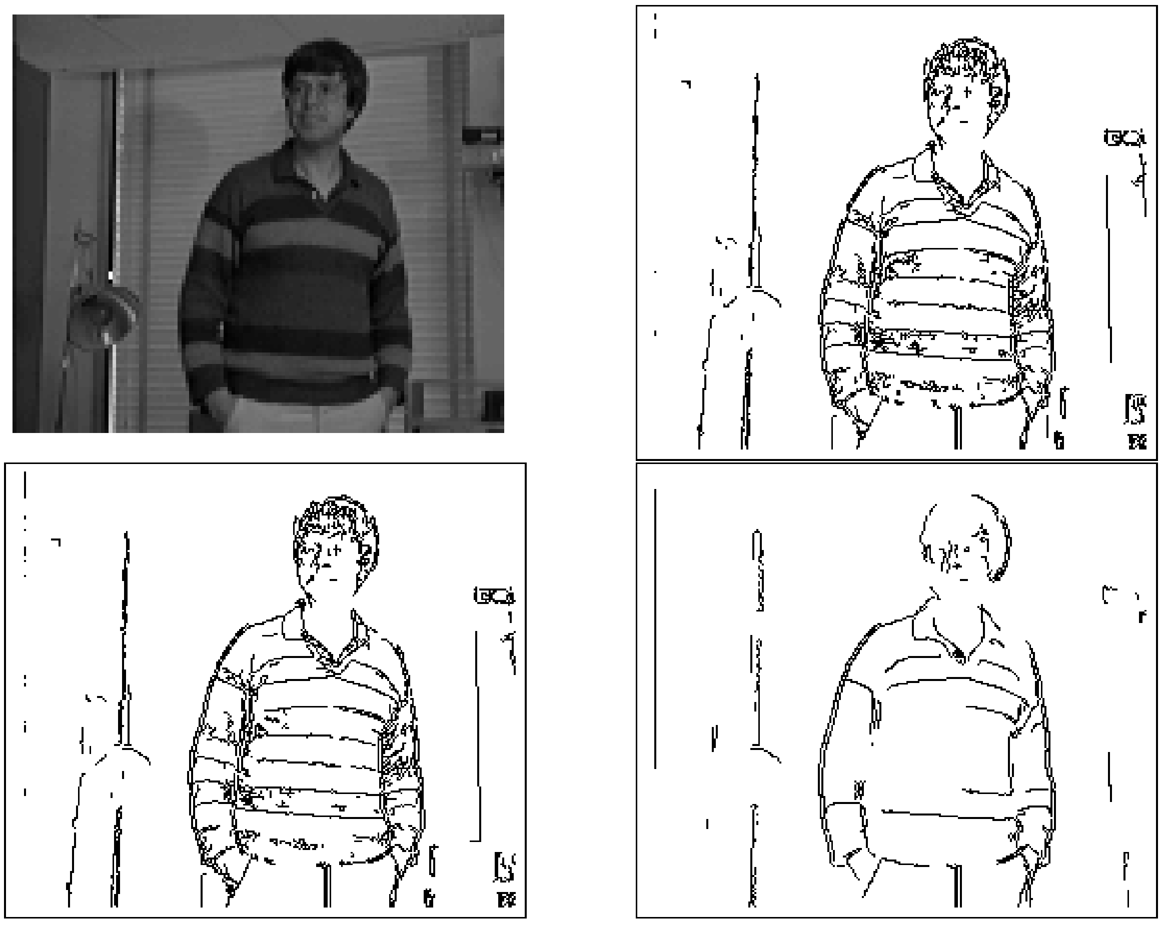


Figure 5.6: Edge maps produced with different scaling functions: Top Left: Original Image. Top Right: $F(I) = 1$, or no scaling. Bottom Left: $F(I) = \ln I$, logarithmic scaling. Bottom Right: $F(I) = I$, linear scaling.

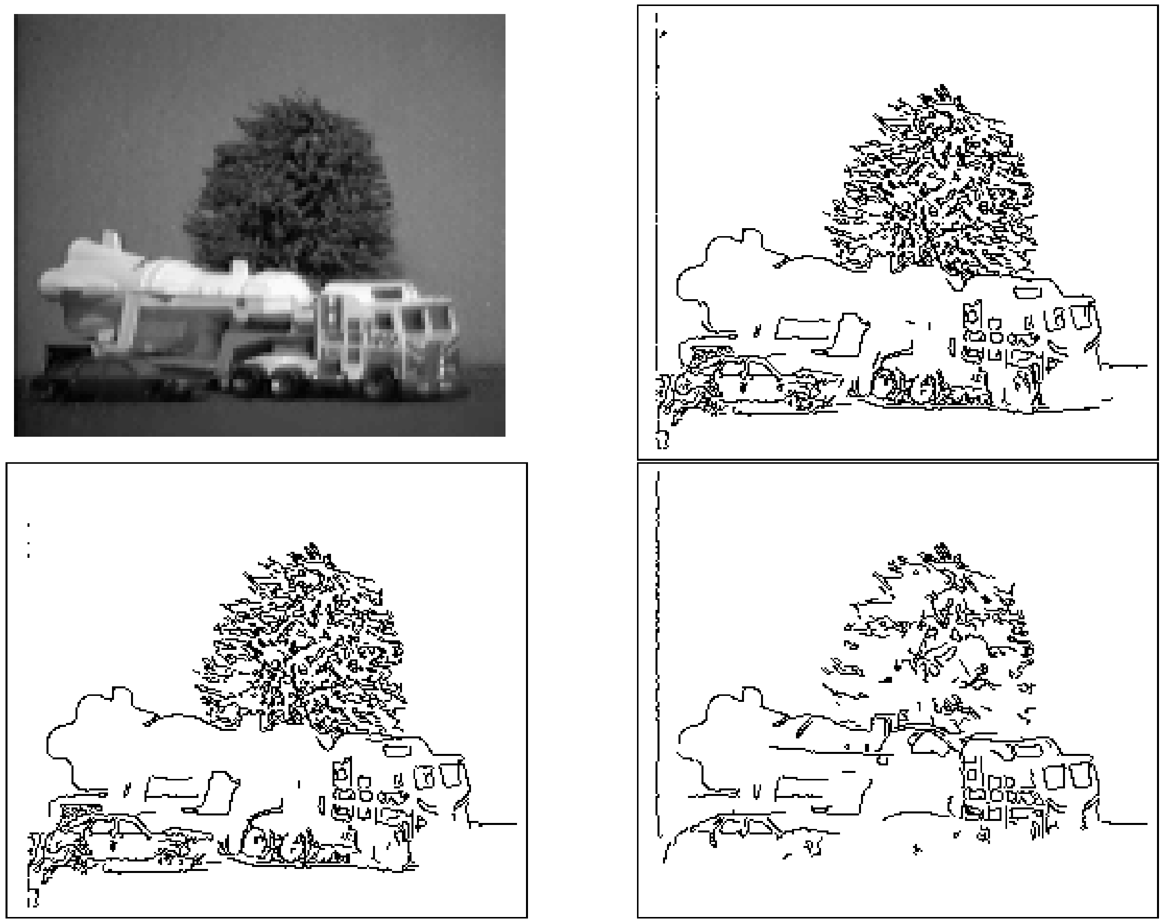


Figure 5.7: Edge maps produced with different scaling functions: Top Left: Original Image. Top Right: $F(I) = 1$, or no scaling. Bottom Left: $F(I) = \ln I$, logarithmic scaling. Bottom Right: $F(I) = I$, linear scaling.

channel ratios alone tend to be more fuzzy with poorly formed outlines than their intensity counterparts. Psychophysically, this agrees with the observation that isoluminant color boundaries also appear fuzzy to humans, while high luminance contrast color edges capture and contain color regions well, even if actual hue differences between the adjacent regions are small.

Except in synthetic images where isoluminant boundaries can be artificially created, most *color* image discontinuities also give rise to intensity discontinuities. Color boundaries appearing as edge features in color edge maps are therefore also very likely to appear as luminance discontinuities in corresponding intensity edge maps. We shall take advantage of this close spatial correspondence between color and luminance discontinuities by matching and aligning them together to produce better connected and better localized *color* edge outputs. This is possible because luminance edge features are in general better formed and better localized than color edge features of the same image.

5.4.2 Overview of Algorithm

We describe, in the following paragraphs, our boundary feature integration technique for improving *color* edge outputs. The algorithm has access to a set of color edges and a set of luminance edges (usually extracted from their respective edge maps), from which it produces an integrated edge map whose features correspond more accurately to actual scene color discontinuities. First, a note on terminology before we proceed: We shall use the term "edge (or boundary) feature" to describe a full length unbroken chain of edge pixels in an edge map. The term "edge (or boundary) segment" refers to a continuous chain of edge pixels that makes up part of an edge (or boundary) feature.

The full integration process takes place in three stages:

1. **Matching and aligning color and intensity boundary features:** At the very least, this stage preserves existing edge features from the original color edge set, so that the output does not exclude color discontinuities that have already been detected. To improve poorly formed *color* edges in the output, we repetitively search for "close spatial matches" between existing color edge features and luminance edge features. If a "close match" is detected, we assume that the "matching" portions (or segments) of the two edge features arise from the same physical discontinuity in the scene. Assuming further that the luminance edge segments are better localized than their matching color counterparts, we can partially reconstruct the color edge feature by replacing it with its matching luminance edge segments in the output. The stage terminates when all reconstructable color edge features have been replaced by their matching intensity segments.

2. **Extending and sealing reconstructed color edges:** We extend the partially reconstructed color edge outputs of Stage 1 to seal up possible edge gaps left behind by the alignment process. A partially reconstructed color edge feature can appear as a few disjoint

edge segments from the luminance edge map. Likewise, two or more color edge features could have been reconstructed as segments arising from the same luminance edge feature. In both cases, we would like to link together these disjoint edge segments because they are likely to have arisen from the same physical color discontinuity.

3. **Discarding spurious edge fragments:** In general, it is very difficult for a computer vision system to reliably tell apart actual edge features from noise induced markings in a single edge map. With two or more visual cues, it is possible to synthesize a relatively reliable decision procedure that discards false edge markings using some simple heuristics. This stage determines which unmatched color edge features are most likely due to noise and removes them from the final color edge output. It bases its decision only on information found in the original color and luminance edge maps.

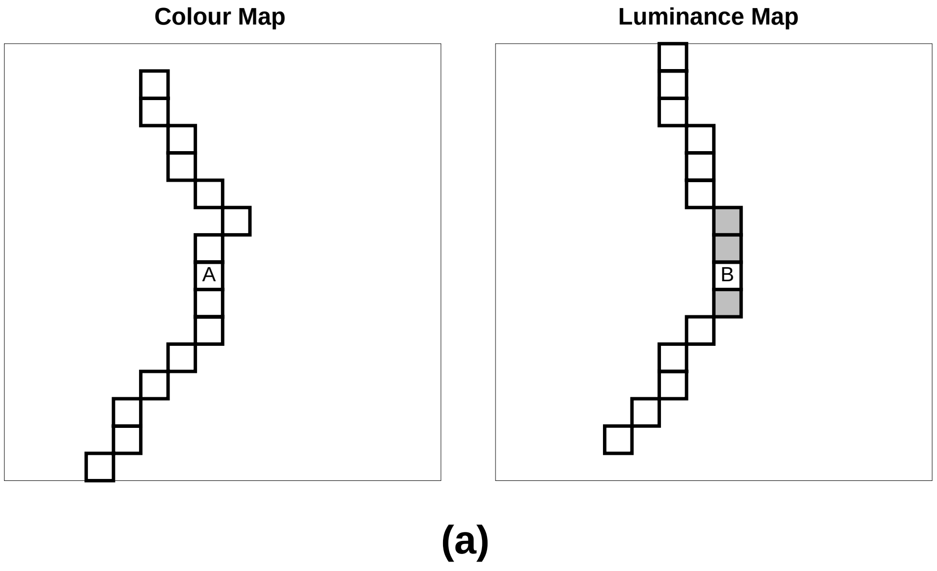
5.4.3 Matching and Aligning Color Edges with Intensity Edges

We say that part of a color edge *matches* part of a luminance edge if every pixel on the matching color edge segment *corresponds* to one or more pixels on the matching luminance edge segment and vice-versa. A color edge pixel corresponds to a luminance edge pixel if their locations are within some small distance δ[2] of each other in their respective edge maps, and if their local orientations differ by less than some small angle θ[3]. Notice that pixel-wise correspondence, as we have defined, does not have to be unique; that is, it need not be a one-to-one mapping relationship between color and luminance edge pixels.

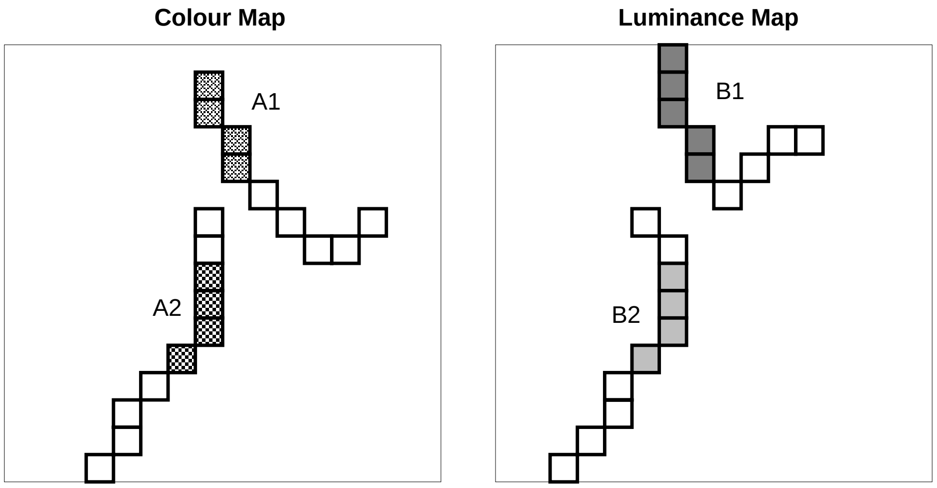
Figure 5.8 illustrates our *matching* and *correspondence* concepts with some specific examples. In part (a), pixel A of the *color* edge segment corresponds to pixel B of the *intensity* edge segment because they both satisfy the proximity and local orientation constraints. Since correspondence can be a one-to-many or a many-to-one relationship, pixel A also corresponds to all the shaded edge pixels near pixel B. In part (b), color edge segment A_1 matches luminance edge segment B_1 because all of A_1's pixels correspond to one or more of B_1's pixels and vice-versa. The same correspondence relationship holds between the pixels of segment A_2 and B_2. In general, edge segments A_1 and A_2 can both be part of the same edge feature, as could edge segments B_1 and B_2. They can even be overlapping edge segments for that matter.

We consider a color edge feature *reconstructable* if a significant fraction[4] of its pixels belong to *matching* edge segments. Figure 5.9 explains the rationale behind our reconstructability criterion. Suppose the physical cause of a color edge feature, like F_1, also gives rise to one or more luminance edge features in the intensity image. We can reasonably assume that F_1's matching edge segments make up a very large fraction of its total edge length, because the color and luminance edge features

[2]We set δ to be 3 pixels' length in our implementation.
[3]30° in our implementation.
[4]75% and above, for our implementation



(a)



(b)

Figure 5.8: (a) An illustration of pixel-wise correspondence. See text for explanation. (b) An illustration of matches between color and luminance edge segments. See text for explanation.

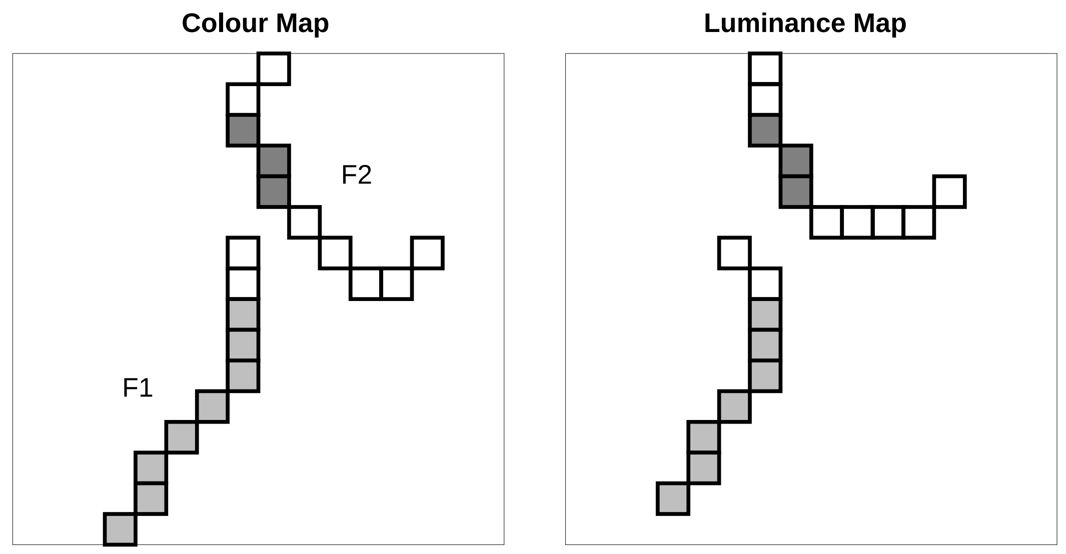
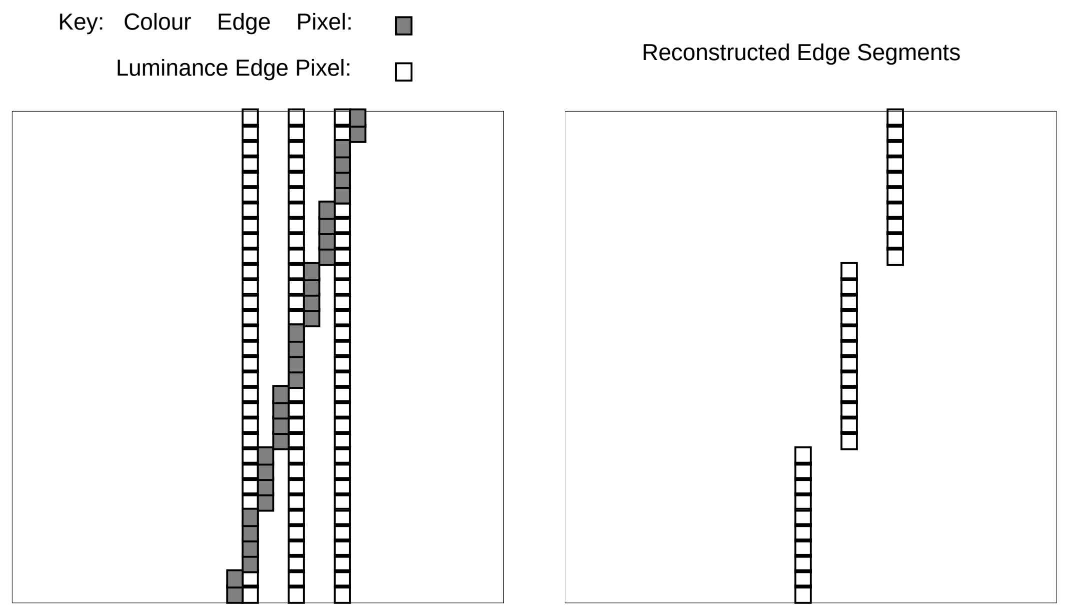


Figure 5.9: A reconstructable color edge feature, F_1, and a non-reconstructable color edge feature, F_2. The shaded areas are matching edge segments.

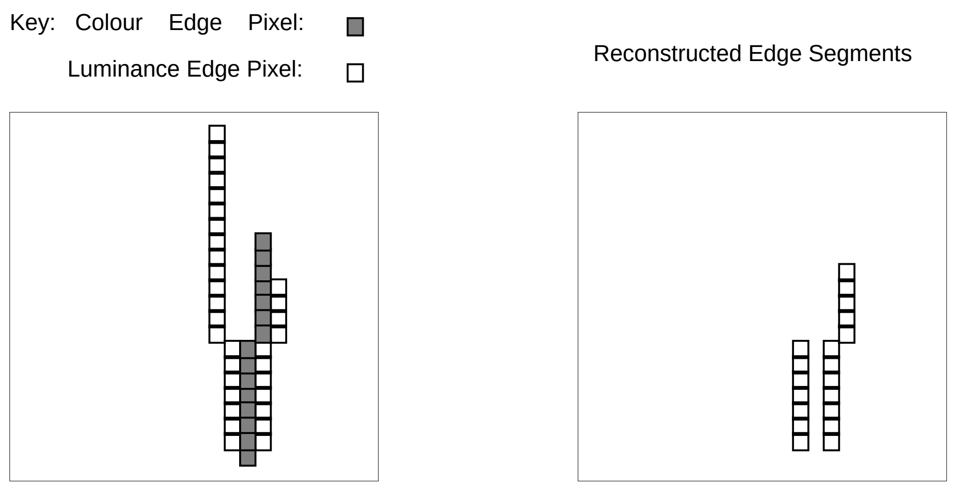
should overlap each other very closely in this case. The converse is usually also true for color and luminance edge features that do not arise from the same physical discontinuity, as for example edge feature F_2 and its false match.

To reconstruct a color edge feature, we improve its overall localization by *aligning* its matching edge segments with their corresponding intensity counterparts, and discarding the remaining unmatched edge portions. Basically, this involves replacing the entire original color edge feature with its matching *luminance* edge segments. In Figure 5.9 for example, the reconstruction process would transform the entire color edge feature, F_1, into the lightly shaded *luminance* edge segments of the *intensity* edge map at the end of the edge matching and alignment stage.

Because of its simple design, the matching and alignment stage can sometimes be easily deceived to produce inconsistent results with certain color and luminance edge configurations. Figure 5.10 illustrates two such examples. Part (a) shows a color edge feature intersecting several closely spaced luminance edges. Although it is clear from the drawing that the color edge feature does not coincide with any of the luminance edge features, it still produces matching segments on all of them. The result is a jagged partial reconstruction as shown on the top right map. Part (b) describes an instance where a single color edge segment gives rise to multiple matching segments in the luminance edge map. For this example, one possible solution might be a checking mechanism that prevents the same color edge segment from producing matches on two or more luminance edge features. We contend, however, that degenerate input configurations like these rarely occur in real world images, so our integration scheme still works well most of the time.



(a)



(b)

Figure 5.10: Two examples where the matching and alignment technique produces inconsistent results. On the left are superimposed maps of the color and luminance edge features. On the right are the aligned edge segments. (a) A color edge feature running across closely spaced parallel luminance edges. (b) A color edge feature producing matching segments on two closely spaced luminance edge features.

5.4.4 Extending and Linking Partially Reconstructed Edge Segments

The matching and alignment process leaves behind an intermediate edge map, whose color edge segments are generally well localized but mostly disconnected. In this stage, we try to extend and link together sets of disconnected intermediate edge segments to form longer and more continuous color edge outputs. We consider a pair of edge segments *linkable* as part of a single greater color edge feature, if they jointly satisfy one of two possible conditions (see Figure 5.11):

1. They are both partially reconstructed edge segments produced by matches against the same original color edge feature. They need not be part of the same luminance edge feature in the original intensity edge map.

2. They are both partially reconstructed edge segments from the same greater luminance edge feature in the original intensity edge map. Their matching color edge segments need not have arisen from the same original color edge feature. If, however, their matching color segments did actually arise from different color edge features, we require that the break between them (the two partially reconstructed edge segments) be no larger than some small distance[5].

A simple argument explains our *linkability* criteria for edge segments. Basically, it is reasonable to assume that edge segment pairs satisfying either of the two conditions above actually arise from the same real world physical color discontinuity, and so may be linked together to form single continuous output color edges. The rationale behind the first condition is obvious. Since both edge segments match the same color edge, and single continuous edge features in the original color map mostly arise from single stretches of physical color discontinuities in the real world, we can safely conclude that the two edge segments are indeed disjoint portions of the same physical color boundary feature. We can justify the second condition as follows: If the two partially reconstructed edge segments arise from the same original intensity edge, it is possible that the two edge segments, together with the unmatched intensity portion between them, coincide with a single continuous physical color discontinuity in the real world. If both segments also match the same color edge feature, the first condition holds and we can be very certain of our hypothesis. Even if the two edge segments do not match the same original color edge, we can still be highly certain of our hypothesis if there is only a small unmatched portion between the edge segments, because spurious breaks in the color edge map are common due to noise.

Our actual edge extending and linking process works as follows: For the first *linkability* condition, we want to join together two luminance edge segments whose matching color counterparts both arise from the same original *color* edge feature. As for now, we shall just consider the case where the two luminance edge segments do not arise from the same *intensity* edge. Notice that the other case where the two edge segments do belong to the same greater luminance edge also

[5]In our implementation, this is half the average length of the two partially reconstructed edge segments.

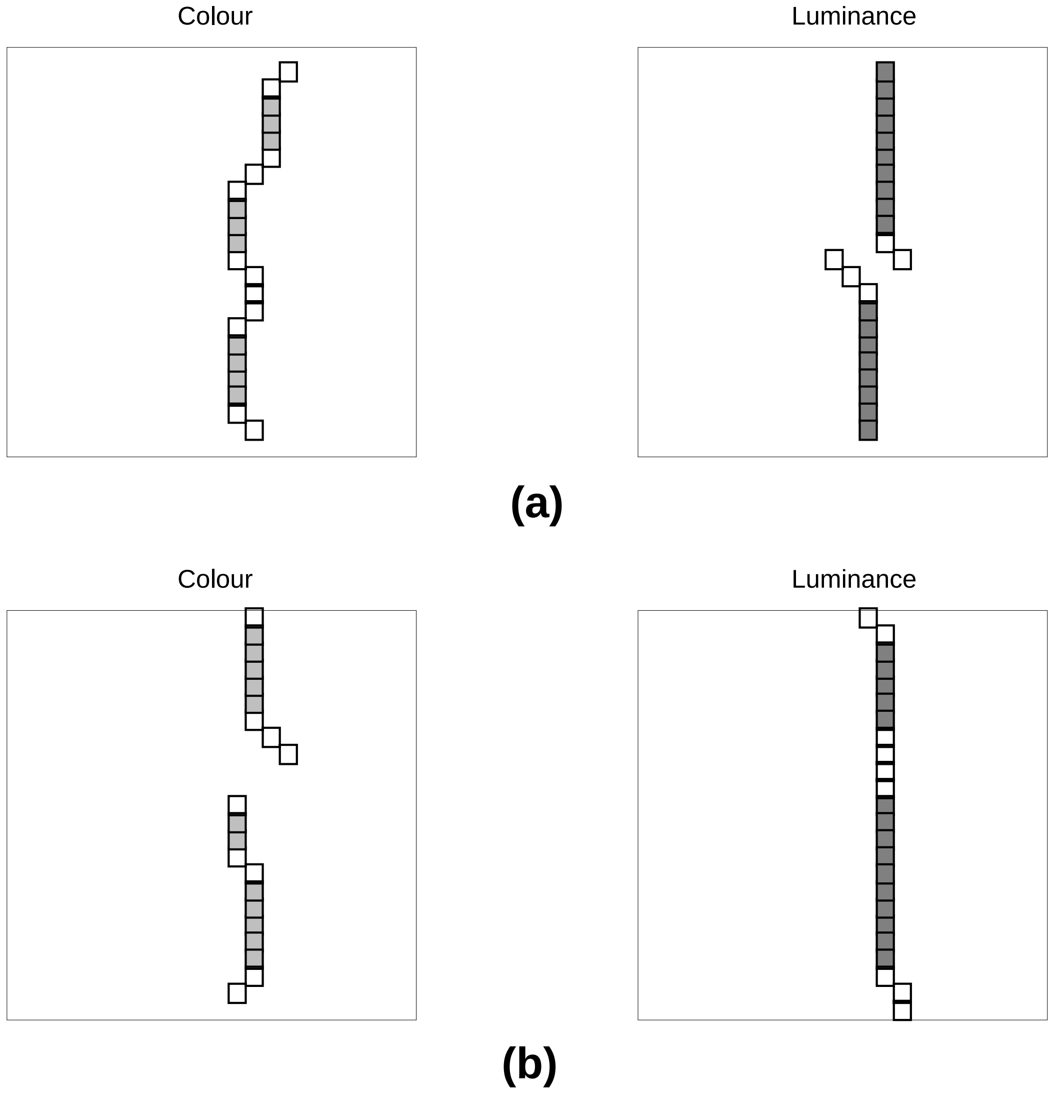


Figure 5.11: Conditions where two partially reconstructed output edge segments may be linked to form part of a single greater color edge feature. The segments under consideration are darkly shaded in the luminance edge maps (right). Their matching color edge segments are lightly shaded in the color edge maps (left). (a) Both segments have matching segments on the same original color edge feature. (b) Both segments arise from the same original intensity edge feature and satisfy some spacing constraints.

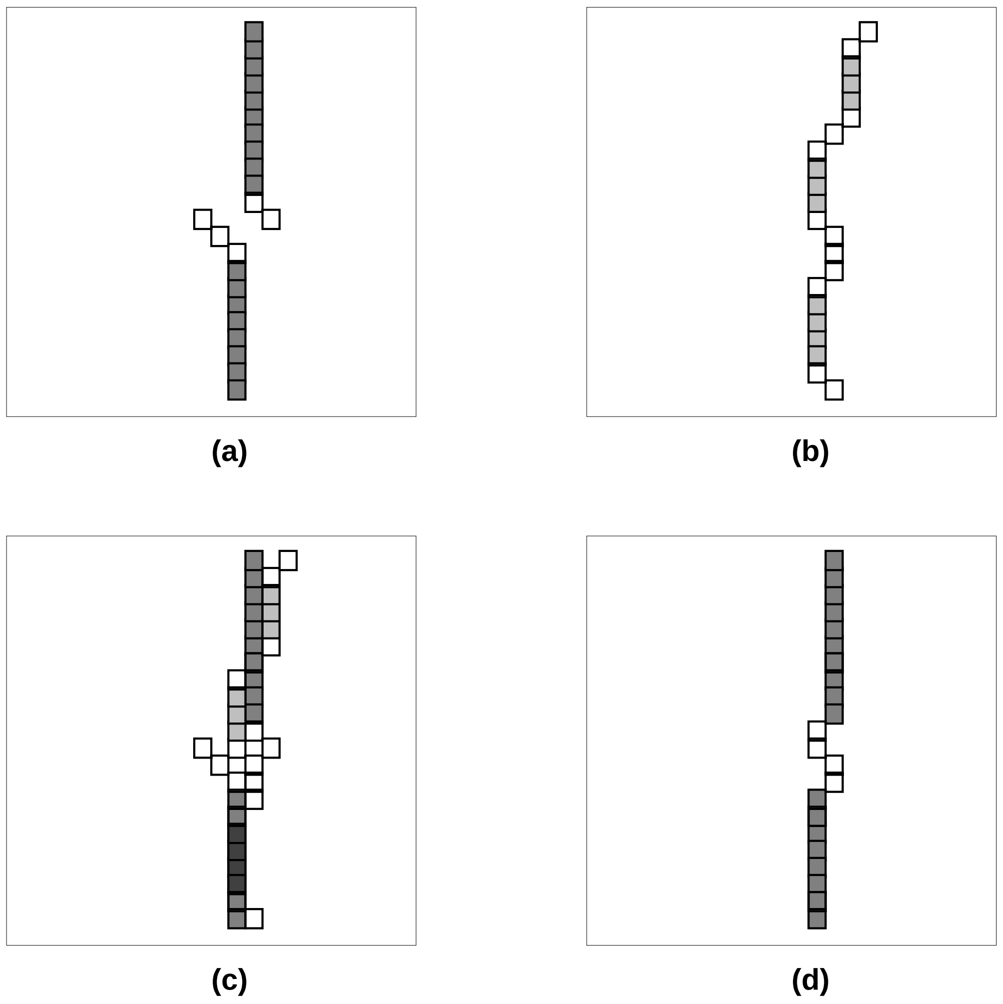


Figure 5.12: Linking procedure for two partially reconstructed output edge segments satisfying the first *linkability* condition. (a) Output edge segments to be linked (darkly shaded). (b) Matching color edge segments (shaded) in original color edge map. (c) Map of (a) and (b) superimposed. (d) Portion of color feature used to link partially reconstructed output edge segments.

satisfies the second *linkability* condition, and will be treated separately together with the second condition for greater convenience. Because we do not have a reliable stretch of luminance edge pixels between the two disjoint segments to help us better localize the missing color boundary, our algorithm must construct the edge link based on information contained in the original color edge map alone. Figure 5.12 shows how this can be achieved. Basically, the idea is to use the original color edge feature's central unmatched portion as a link for the two disjoint segments. If gaps still exist between the end points of the link and the edge segments, we simply seal the gaps by directly connecting the link and edge segments end points together. Since we are using edge points from the original color map to extend our color output boundaries, we can expect the extended boundaries introduced by this process to be as well localized as the original color edge features.

The second *linkability* condition deals with pairs of disjoint edge segments that belong to the same greater luminance edge feature. In order to fully satisfy this condition, we argued earlier that the two edge segments must overlap a single real world color discontinuity that coincides spatially with their greater luminance edge. Therefore, to link together the two segments, we can simply extend them along their greater luminance edge feature's path until they meet. Because we are using intensity edge points to extend our color output boundaries, and luminance edge features tend to be spatially well aligned with their physical causes, we can expect very well localized and less fragmented color edge results from this process.

5.4.5 Discarding Spurious Edge Fragments

The integration processes that we have described so far make use of luminance edge features to realign and reconnect broken color edges. All original color edge features that have not been improved upon by the previous two stages are still being preserved in the algorithm's output at the end of the second stage. A little analysis will show that it should also be possible to identify and discard false color edge features from the output edge map, by comparing and combining color and intensity edge based information.

We adopt the following heuristics for differentiating real color edge features from false color map markings due to image noise: If a color edge feature matches well with a luminance edge feature, or if its length is sufficiently large by some appropriate measure[6], then from a probabilistic standpoint, we can reasonably assume that the color edge feature is indeed real, and corresponds to some material change in the physical world. The assumption makes sense because: (1) it is fairly unlikely that a randomly formed false color edge actually aligns itself sufficiently well with a physical luminance edge feature, so as to produce a decent match; nor (2) is it likely that noise induced hue differences in a color image can actually be regular enough to produce long and perceivable color discontinuities. Our differentiation scheme therefore treats all unmatched color edge fragments below a certain length threshold as noise markings to be discarded from the final color edge output.

[6]In our implementation, we fixed this measure at a constant length of 15 pixels.

Psychophysically, the scheme apparently agrees well with human visual characteristics, which tends to readily overlook small color image patches with isoluminant boundaries.

5.4.6 Results

For efficiency reasons, we implemented a reduced version of the boundary based integration scheme on a serial Symbolics Lisp Machine, and tested the algorithm on a few image examples. Our implementation differs from the original intended design in one major aspect, namely, it does not take into account the local orientation constraint when matching pixels. Also, the implementation merges Stages 1 and 2 of the original algorithm together into a single, approximately equivalent procedure. It is still our intention to eventually have a working implementation of our full integration scheme at a later date.

Two of our test scenes are shown in Figures 5.13 and 5.14. For each test case, we obtain a set of luminance edge features by running a Canny intensity edge finder [Canny 83] through the scene's grey-level intensity image. We then compute the color edge map using our color boundary detection algorithm described in the earlier sections. Notice how the luminance edge features of both images, are on the whole, much better localized and much better connected than their corresponding color boundaries. Notice also, that these well formed luminance edge features finally replace and connect together their matching but poorly localized color edge fragments in the algorithm's output. The algorithm's third stage can be best appreciated by comparing results where the color images are most noisy, namely within the subject's hair region for Figure 5.13, and near the rear ends of the vehicles for Figure 5.14. The stage produces an overall cleaner edge map by correctly discarding most of the short, unmatched color edges from the final output.

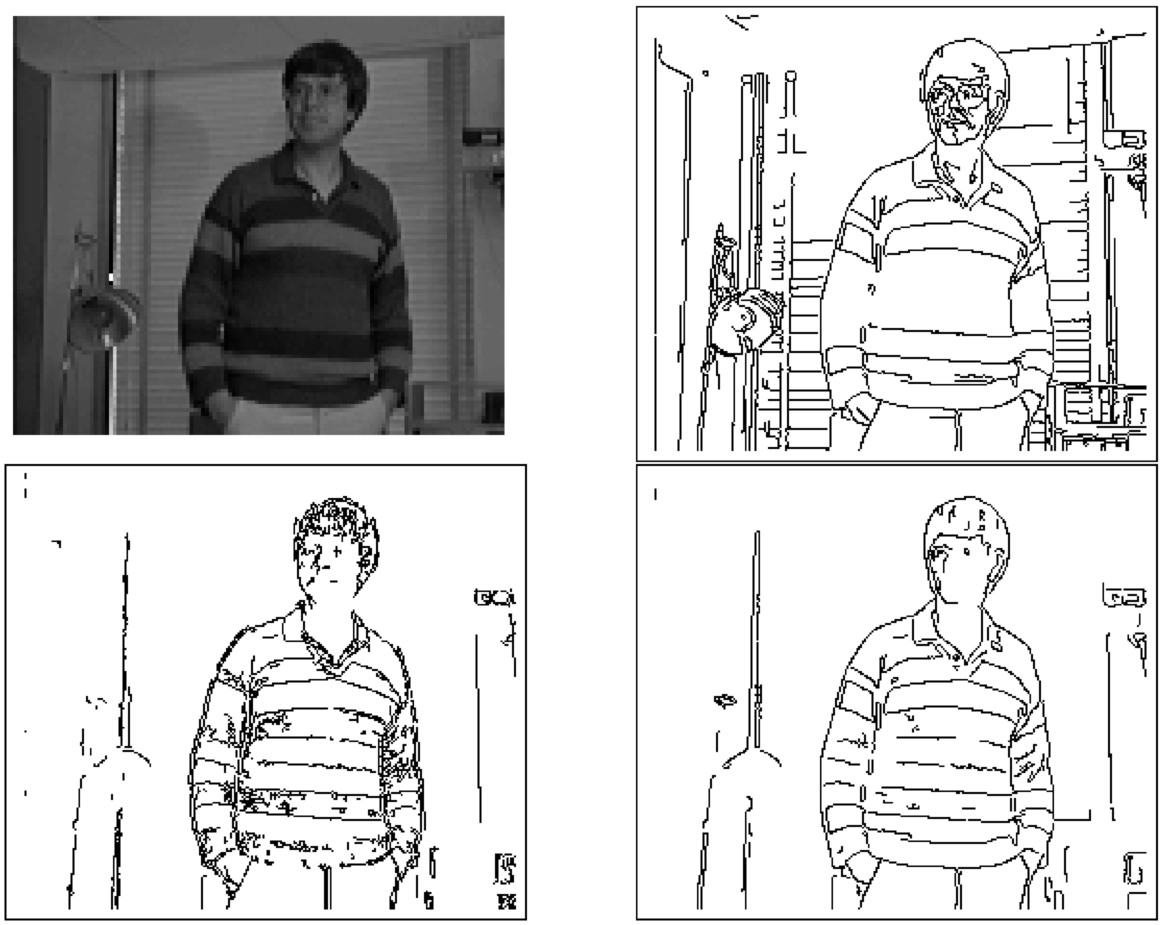


Figure 5.13: First test image example. Top left: Original image. Top right: Luminance edge map. Bottom left: Color edge map. Bottom right: Reconstructed result.

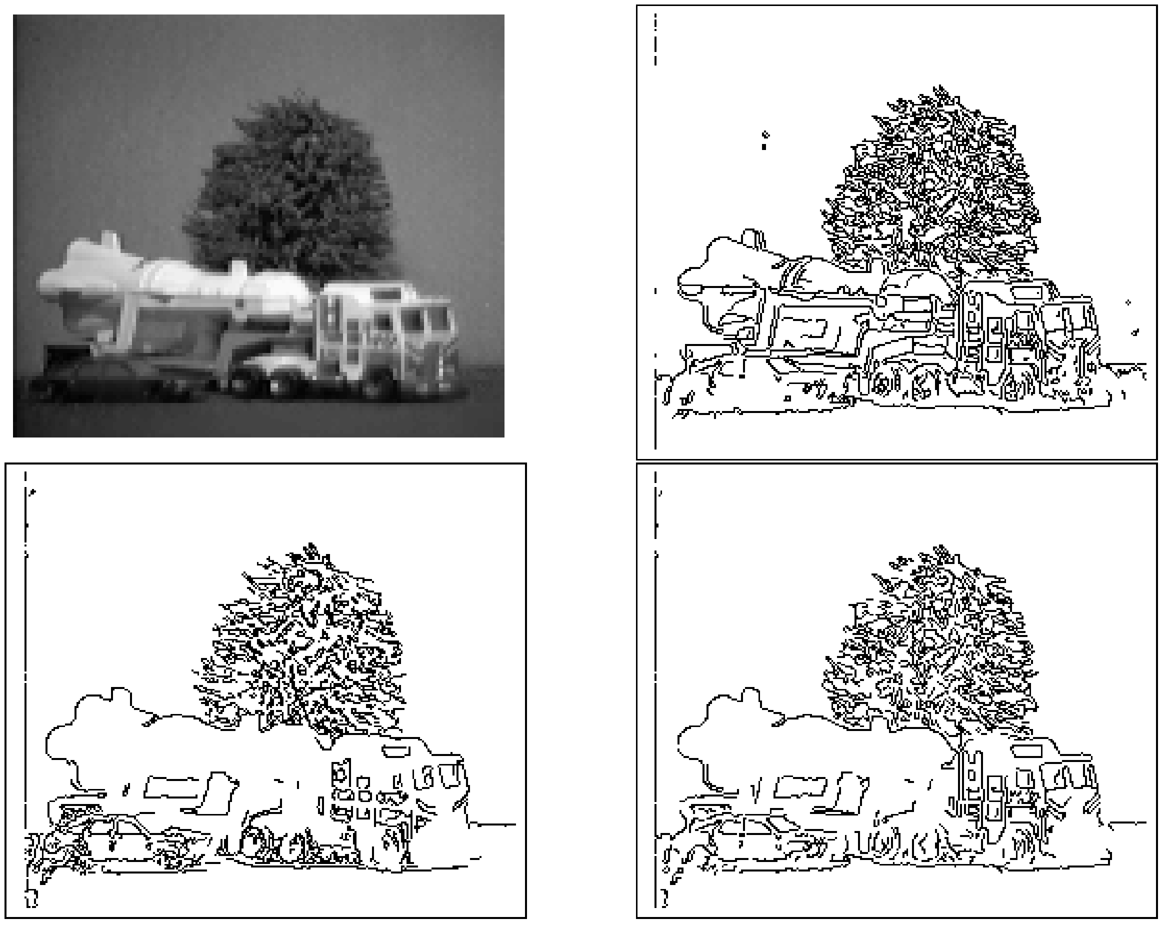


Figure 5.14: Second test image example. Top left: Original image. Top right: Luminance edge map. Bottom left: Color edge map. Bottom right: Reconstructed result.

Chapter 6

Finding Color Regions

Image segmentation algorithms have generally been based upon one of two basic image value properties, namely discontinuity and similarity. In the previous chapter, we addressed the problem of color boundary detection, where an image is partitioned into separate regions based on abrupt color changes. Our motivation there was to detect edge and line features in the image. This chapter deals with the dual problem of color boundary detection, called *color region finding*, that segments images into separate regions based on color uniformity. Our goal here is to group individual pixels in an input image into sets of connected pixels sharing some common physical property (surface color in this case) to form surface features. Ideally, each of these features should either correspond to a full world object or a meaningful part of one in the scene.

A little thought will reveal that given perfect boundary maps, region finding becomes trivially solvable because we can simply "fill in" image boundaries to form regions. Despite its seemingly redundant nature with respect to boundary detection, region finding has still been widely recognized as one of the key early level computer vision processes for several reasons. First, boundary maps of real images are seldom perfect and often fragmented because of sensor noise and other operator design limitations. Applying these "filling in" procedures naively to real images can result in "bleeding" effects that gives rise to erroneously overmerged regions. An independent process must therefore be derived to perform region finding, which may in turn be combined with boundary detection to produce better segmentation results [Milgram and Kahl 79] [Haddon and Boyce 90].

Second, although one might eventually want "perfect" boundary and region maps from segmentation algorithms, certain vision applications today can still work fairly well with more "conservative" region finding results, whereby not every image pixel maps to some region, and not every output region corresponds to an entire image surface. These "conservative" region estimates can often be easily obtained using uniformity based region finding techniques, but not discontinuity based boundary detection techniques. An excellent example of such an application is in region based boundary feature grouping, similar in idea to REGGIE of [Clemens 91].

Third, it can be argued that although boundary based information is very useful for accurately localizing objects in an image, region based information can sometimes be much better suited for identifying and confirming the presence of interesting objects in the scene. For example, I might be able to identify my research notebook in a heavily cluttered environment, not because I am able to see its outline clearly among other objects in the scene, but because I can confidently recognize the distinctive color of its surface. Similarly, a hunter might be able to spot a leopard hiding behind a bush by just recognizing the markings on its coat without actually having seen the whole animal. As mentioned earlier, a computer vision system can emulate these object identification processes more naturally by using a direct uniformity based region approach to image segmentation instead of an indirect discontinuity based boundary approach.

6.1 Color Regions and their Computation

Often, what constitutes a "region" will depend on the particular task at hand. We shall propose a general and somewhat less restrictive notion of region finding, which we believe, meets the requirements of many middle and higher level computer vision applications.

6.1.1 A Basic Formulation

Let I represent the entire image region. We can view *region finding* as a process that marks out within I, subregions: $\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_n$, such that:

1. $\bigcup_{i=1}^{n} \mathcal{R}_i \subseteq \mathcal{I}$,
2. Each $\mathcal{R}_i$ is a connected set of pixels,
3. $\mathcal{R}_i \cap \mathcal{R}_j = \{\}$, for all $i \neq j$,
4. $H(\mathcal{R}_i) = \texttt{TRUE}$, for $i = 1, \cdots, n$,
5. $H(\mathcal{R}_i \cup \mathcal{R}_j) = \texttt{FALSE}$, for $i \neq j$ and $\mathcal{R}_i, \mathcal{R}_j$ have a common boundary.

The boolean predicate, $H(\mathcal{R})$, is sometimes known as a *homogeneity function*, and is defined over all image pixel within the region R. It establishes a set of uniformity criteria for grouping image pixels into regions. For *color* regions, we want $H(\mathcal{R}) = \texttt{TRUE}$ if and only if all pixels within R have *color* values that are "similar enough" to have arisen from the same physical surface. Much of this chapter concerns deriving a suitable homogeneity function for color image values.

Notice that Condition 1 does not require all input image pixels to be included in some output region. This is useful because it allows the region finder to exclude outlying points that do not fit well into any region from the final segmentation. For color images, such points tend to occur frequently near region boundaries and within poorly illuminated areas of the image where *color ratios* can be extremely noisy.

6.1.2 Region Finding Algorithms

We describe, in the following paragraphs, the general structure of a uniformity based region finder to summarize previous work done in this area, and to outline the color region finding algorithm that we will be presenting subsequently. Typically, a uniformity based region algorithm proceeds in three stages:

1. **Mark out initial image locations (or patches) to start region growing:** We lay down a set of "seed" points (or patches) in the image from which we grow regions. Ideally, each "seed" should be entirely contained within a single image region so that the growing patch it generates can also be entirely contained within a single image region. Each image region that we wish to find should house at least one "seed".

2. **Grow initial patches:** We increase the size of each "seed" by appending to it those neighbouring pixels with "similar" physical attributes, for example luminance, texture or surface color. Using the basic region finding formulation we established earlier, if S is a growing "seed" and p is a neighbouring image pixel, then the growing process appends p to S if and only if $H(\mathcal{S} \cup \{p\}) = \texttt{TRUE}$. The *homogeneity* check ensures that all growing "seeds" stay within bounds of their enclosing regions.

3. **Merge adjacent patches that can be combined:** When two or more growing "seeds" meet, we merge them together into a single larger patch if their attributes are "similar" enough to have arisen from the same physical surface. More formally, if $\mathcal{S}_i$ and $\mathcal{S}_j$ are two growing "seeds", then the merging process joins them together if and only if $H(\mathcal{S}_i \cup \mathcal{S}_j) = \texttt{TRUE}$. Again, the *homogeneity* check ensures that only "seeds" arising from the same physical entity may be merged, so each larger image patch that the merging process produces still falls within a single image region.

Steps 2 and 3 are usually performed in parallel and the algorithm terminates when no more "seeds" can be further grown or merged. The set of final patches we get form a segmentation of the image.

Clearly, the main challenge in Step 1 is to design an algorithm that reliably generates sets of image "seeds" that meet the topological requirements above. It would also be desirable to generate sufficiently large "seeds" that adequately describe the surface attributes of their enclosing regions even before further growing. This condition is critical for good region finding results, because these "seed" attributes are key inputs to the *homogeneity tests* of Steps 2 and 3.

Traditional region finding techniques make use of a partitioning process, called *splitting*, to produce maximally large initial image "seeds" [Hanson and Riseman 78] [Rosenfeld and Kak 76] [Horowitz and Pavlidis 74]. Basically, the idea of *splitting* is to recursively divide an image into smaller sub-regions, until each sub-region falls entirely within a single image entity. The process begins by examining the entire input image for signs of attribute disuniformity, which, if detected, indicates the presence of different surface entities in the scene. For images with non-uniform attributes, it then divides the image into smaller sub-regions, and recursively applies the *splitting* procedure to each sub-region until no more sub-regions can be further divided. The set of resulting sub-regions make up the "seeds" that start the subsequent *growing* and *merging* stages. A popular *attribute uniformity* test paradigm for *splitting* uses *feature histograms* to estimate the number of different image entities in an image sub-region [Prewitt and Mendelsohn 66] [Chow and Kaneko 72], where each histogram mode indicates the presence of one image entity. This test paradigm has also been extended to use multi-dimensional feature histograms for multi-dimensional image attributes, like intensity gradients [Bracho and Sanderson 85] and color [Ohlander 76], where the presence of multiple modes in any histogram dimension indicates the presence of multiple region entities. Generally however, feature histogramming methods cannot reliably separate image regions whose modes peak around the same histogram location, especially if one mode is significantly smaller than the other in size. This happens fairly often in reality when we process images with sufficiently noisy attributes, and also those with a wide range of region sizes.

A somewhat different "seed" generating approach starts by performing attribute uniformity tests on small surface patches throughout the input image, after which it links together adjacent image patches with uniform attributes to form initial "seeds". Since each locally uniform patch contains no attribute boundaries by definition, and pairs of adjacent uniform patches do not contain separating boundaries between them[1], "seeds" generated in this fashion should fall entirely within a single image region if the attribute uniformity test paradigm is reliable. Some recent work by Klinker, Shafer and Kanade [Klinker Shafer and Kanade 88a] [Klinker 88] use a similar local "seed" generating technique in a dichromatic model-based segmentation scheme. To test a patch of image pixels for attribute uniformity (color uniformity in this case), the technique hashes pixel RGB values into a 3 dimensional *color* histogram and checks the resulting distribution for a *matte* signature. It then links together adjacent *matte* patches as initial region finding "seeds" if their combined color histogram distribution is also *matte* in form. On the whole, local techniques like the above are far less likely to overlook small regions in the input than *splitting* approaches do, because by examining small image patches for attribute uniformity instead of large ones, they do not allow very large regions in the image to cloud out smaller nearby regions. What most implementations like [Klinker Shafer and Kanade 88a] and [Klinker 88] lack, however, is a general method for determining suitable test patch sizes.

For Steps 2 and 3, the *growing* and *merging* tasks also often reduce to finding an appropriate homogeneity function for combining pixels with "seeds" or pairs of image "seeds". Like the uniformity test paradigms of Step 1, the simplest homogeneity functions are also attribute based, and work by locally comparing pixel values from the pair of image elements to be appended. Usually, this includes testing the pair of elements for similarity between their *mean* attribute values,

[1]We are assuming here that adjacent patches overlap partially.

their attribute *variances* and possibly other higher order attribute moments. Klinker, Shafer and Kanade's dichromatic model-based segmentation scheme uses a set of more versatile similarity conditions to account for the presence of secondary effects like specularities in the image. The scheme takes advantage of the property that surface reflected light consists of two components — a *matte* component and a *highlight* component, both of which appear as vectors in the *RGB* color space. To compare two image elements for color similarity, the scheme attempts to geometrically infer and match their *matte* components by operating on their *RGB* histogram distributions. Other more sophisticated homogeneity functions use domain dependent heuristics, like knowledge about probable boundary shapes and perimeters, to help in their test decisions. Some examples can be found in [Brice and Fennema 70] and [Feldman and Yakimovsky 74].

6.1.3 Segmentation Thresholds and Parameters

It is reasonable to infer that most variants of the above uniformity based region algorithm will contain at least a few free operating thresholds and parameters. In Stage 1 for example, we expect at least one free parameter from the "seed" generation process, such as a uniformity threshold that controls region splitting, or a mode discrimination threshold for classifying histogram distributions. In Stages 2 and 3, we can also expect to find some attribute uniformity and element size thresholds, embedded within the "seed" growing and merging procedures.

As in many other early level computer vision processes, one of most difficult problems in region finding concerns choosing a suitable set of free thresholds and parameter values that work well for a wide range of real images. We shall examine this problem more closely in the color algorithm we design.

6.2 Tests of Confidence and Significance — A Statistical Formulation for Color Region Finding

In this section, we introduce a statistical approach to image "seed" generation and attribute uniformity testing using color. Specifically, we shall devise statistical tests and answers to the following questions: (1) How large must a uniform image patch be so that its *color* can be "measurable" in a noisy image ? (2) Is a given image pixel located within the interior of a color region ? (3) Are two adjacent color image patches part of the same greater color region ? Our answers to these questions will help us determine suitable free thresholds and parameter values in a traditional region finding framework, which we shall describe in the next section.

The kind of tests that we will be performing are commonly known as *confidence* and *significance* tests (see for example [DeGroot 86] and [Frieden 83]). The general form of a *confidence test* appears as follows:

$$Prob\,[|x_d - x| \leq \epsilon] \geq C, \tag{6.1}$$

where x_d is the derived value of a measured noisy quantity, x is the actual value, ϵ is an adjustable error bound, and C is a free parameter with value between 0 and 1, known as the *confidence coefficient*. Equation 6.1 describes the "goodness" of a certain measurement, x_d, as a combination of two factors: Its *accuracy* with respect the actual value, as indicated by the error bound ϵ, and its *certainty factor*, as reflected by the *confidence coefficient* C, which denotes the probability that the measured value, x_d, falls within the indicated error bound ϵ of the actual value x.

A *significance test* is similar in spirit to a *confidence test*, except that it is used for verifying hypotheses about systems instead of determining the accuracy of measurements. Given a system, we postulate a *hypothesis* (H_0) about it, and obtain a set of *observations* (U) to determine if the hypothesis is valid. We also associate with the procedure an adjustable parameter, α, known as the *level of significance* for the test. An observation U is *statistically significant* if its chance of being produced by a system obeying H_0 is smaller than α, which may in turn by interpreted as strong evidence against H_0. Conversely, U is *statistically insignificant* if its chance of arising from a system obeying H_0 is greater than α, which may suggest evidence supporting H_0. These concepts about *significance testing* should become clearer later on in this section.

We see two advantages of applying the above mentioned statistical methods to a traditional region finding framework:

1. **A more insightful interpretation of free threshold and parameter values:** At first glance, our statistical approach does not help us overcome the difficult problem of selecting suitable thresholds and parameters values, because it merely replaces one set of free region finding thresholds and parameters (those used by the "seed generation" and uniformity test paradigms) with another set of free parameters (the C, ϵ and α parameters of the *confidence* and *significance* tests). A closer examination will show, however, that this new set of free parameters provides greater insight to the system's characteristics, in terms of its attribute sensitivity (ϵ) and reliability (C and α). We contend that this new set of parameters is more desirable as a set of adjustable system variables, because they give the user direct control over the interesting system characteristics.

2. **A natural implementation of adaptive thresholding and parameter selection:** In real images, image properties like noise strength, region density and contrast, can be very different at different image locations. Some traditional region finding algorithms have made use of adaptive thresholding and parameter selection techniques to account for these image property differences. Our statistical approach performs these adaptive adjustments automatically because it computes, for each image location, a set of region finding thresholds and parameter values that locally meets the user specified sensitivity and reliability levels.

6.2.1 The Size of Detectable Image Patches

We shall begin by deriving a result that helps us determine a lower size limit for "detectable" noisy image regions, where our notion of "detectability" will be defined shortly. Suppose we want to segment a perfectly noiseless image, whose entities are all piecewise uniform color surfaces, into a set of uniform surface color regions. Because the image is noiseless, we can measure the exact *color ratio* at each pixel, and so we can isolate color regions that are as small as a single pixel in size. In a noisy image, there is a non-zero *color* noise variance at each pixel which degrades our ability to isolate single pixel image regions. Although we can still obtain arbitrarily good *color* estimates in a noisy image by performing local *color* averaging as in Chapter 4, these averaging results are valid only if the averaging neighbourhood falls entirely within a single image region. That is, we can only obtain reliable *color* estimates for noisy regions that are sufficiently large in size.

We consider a uniform color image patch "detectable", if we can determine its true *color ratio* to within an *angular* error cone of ϵ radians, at a *confidence level* of C. Given the *angular* color noise distribution in an image, we want to find the smallest possible image patch size, N, (measured in number of pixels) that satisfies the "detectability" condition above. Intuitively, we expect the value of N to increase as C increases and ϵ decreases.

We shall approach the problem as a *confidence test* of accuracy in a mean, where the region's true color ratio is $\hat{c}$, and our task is to quantify the *color sample mean's* accuracy as a function of the image patch size N. Assume that the *angular* noise distribution at each pixel, $A = \mathcal{A}(\boldsymbol{c}_n,\ \hat{c})$, is approximately as given in Equation 4.4:

$$Pr_A(A) = \frac{L^2}{\sigma^2} A e^{-\frac{L^2 A^2}{2\sigma^2}}, \tag{6.2}$$

where A is the *angular* color difference measure and $\boldsymbol{c}_\mu$ is the *color sample mean.* From Chapter 4, the color sample mean of a size N neighbourhood is:

$$\boldsymbol{c}_\mu = \sum_{n=1}^{N} \frac{\boldsymbol{c}_n}{|\boldsymbol{c}_n|}, \tag{6.3}$$

and its *angular* error distribution, $E = \mathcal{A}(\boldsymbol{c}_\mu,\ \hat{c})$, is approximately:

$$Pr_E(E) = \frac{NL^2}{\sigma^2} E e^{-\frac{NL^2 E^2}{2\sigma^2}}. \tag{6.4}$$

To achieve for $\boldsymbol{c}_\mu$ an error cone of ϵ radians at a confidence level of C, we want a value of N such that:

$$Prob\,[\mathcal{A}(\boldsymbol{c}_\mu,\ \hat{c}) \leq \epsilon] \geq C, \tag{6.5}$$

or equivalently:

$$\int_0^{\epsilon} Pr_E(E)dE \geq C. \tag{6.6}$$

Substituting Equation 6.4 into the above and performing the integral, we get:

$$1 - e^{-\frac{NL^2\epsilon^2}{2\sigma^2}} \geq C, \tag{6.7}$$

which eventually reduces to:

$$N \geq \frac{2\sigma^2}{L^2\epsilon^2} \ln(\frac{1}{1-C}). \tag{6.8}$$

Equation 6.8 expresses the minimum "detectable" patch size, N, for a designated C and ϵ pair, as a function of an *angular* noise strength term, $\frac{\sigma}{L}$. We can derive the value of $\frac{\sigma}{L}$ by measuring $\bar{\delta_A}$, the local average *angular* color difference magnitude between two adjacent image pixels. Let δ_A be the angular color difference magnitude between two adjacent pixels from the same region, in an image whose angular noise distribution is given by Equation 6.2. Since δ_A sums the errors of two independent $\boldsymbol{c}_n$ readings, we can easily show that it has the following distribution:

$$Pr_{\delta_A}(\delta_A) = \frac{L^2}{2\sigma^2} E e^{-\frac{L^2E^2}{4\sigma^2}}, \tag{6.9}$$

whose mean, $\bar{\delta_A}$, equals $\frac{\sigma}{L}\sqrt{\pi}$. So, to compute $\frac{\sigma}{L}$ at a given image point, we simply measure $\bar{\delta_A}$ for the pixel's local neighbourhood and multiply the result by $\frac{1}{\sqrt{\pi}}$.

6.2.2 The Insideness of Image Points

Our next test result determines whether or not an image point is well contained within the interior of a color region. One way of generating large image "seeds" that do not cross region boundaries, is to consider only those non-boundary pixels in the image when marking "seeds". The decision procedure for classifying pixels is not as difficult as it might seem, because we can still generate large valid "seeds" by conservatively excluding some interior pixels as boundary pixels, as long as we do not wrongly include any boundary pixels as interior pixels.

The decision procedure uses a *significance test* paradigm, with a simplifying assumption that surface color around an image pixel can only be uni-modal or bi-modally distributed. The former applies when the pixel is located sufficiently deep within a uniformly colored region, while the latter holds true when the pixel is near a color boundary. Notice that we are essentially ignoring cases where the pixel is near a multi-region color junction.

Suppose we establish a hypothesis, H_0, that a given pixel lies inside a uniformly colored image region. For now, let us also assume without any loss of generality that if H_0 is `false`, the pixel lies on a color boundary of known orientation, and the pixel's local neighbourhood color distribution appears as in Figure 6.1(b). We can test for H_0 by computing the *mean* color ratio on both halfs of

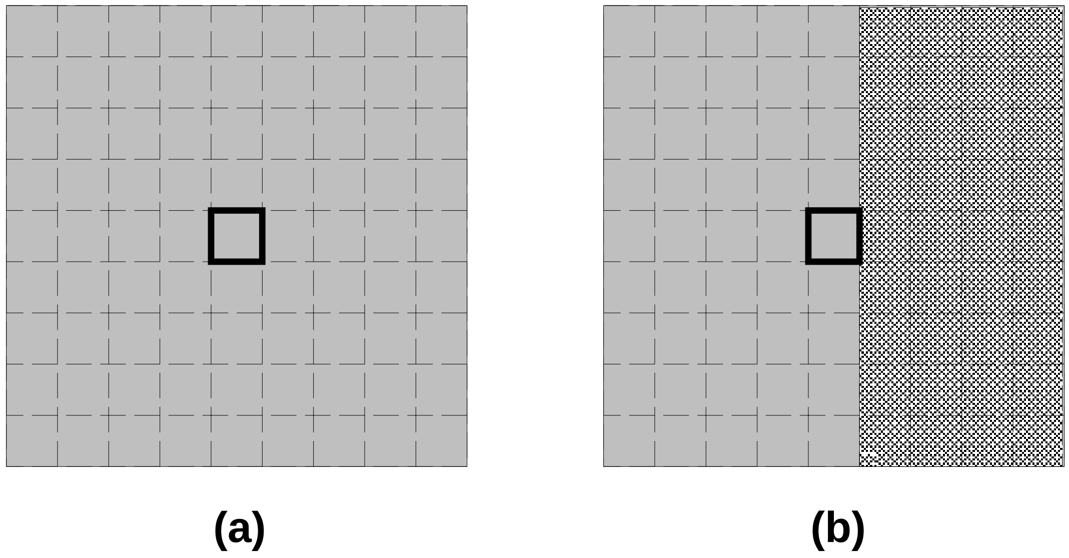


Figure 6.1: (a) Local neighbourhood color distribution of a pixel deep inside a uniformly colored image region. (b) Local neighbourhood color distribution of a color boundary pixel. Other color edge orientations are also possible.

the pixel's local neighbourhood, and checking that they are not *significantly different* for a desired *significance level* α.

To compute the left and right *mean* color ratios, $\boldsymbol{c}_\mu^L$ and $\boldsymbol{c}_\mu^R$, we average *colors* using Equation 6.3 over all neighbouring pixels within the respective halfs. Assuming as before that the *angular* color noise distribution at each pixel is given by Equation 6.2, we can easily derive the following *angular* color noise distribution for the left and right half *mean* vectors respectively:

$$\begin{aligned} Pr_{EL}(E_L) &= \frac{N_L L^2}{\sigma^2} E_L e^{\frac{E_L^2 N_L L^2}{2\sigma^2}}, \text{ and} \\ Pr_{ER}(E_R) &= \frac{N_R L^2}{\sigma^2} E_R e^{\frac{E_R^2 N_R L^2}{2\sigma^2}}, \end{aligned}$$

where N_L and N_R are the number of pixels in the left and right neighbourhood halfs. Furthermore, if the true left and right half *color means* are equal, we can show that their measured angular difference, $D_\mu = \mathcal{A}(\boldsymbol{c}_\mu^L, \boldsymbol{c}_\mu^R)$, obeys the following angular probability distribution:

$$Pr_{D\mu}(D_\mu) = \frac{N_L N_R L^2}{(N_L + N_R)\sigma^2} D_\mu e^{-\frac{D_\mu^2 N_L N_R L^2}{2(N_L+N_R)\sigma^2}}. \tag{6.10}$$

We consider H_0 (the region interior hypothesis) plausible if the D_μ value we measure is *insignificant* relative to the desired α level. That is, if:

$$\int_0^{D_\mu} Pr_{D\mu}(D)dD \leq (1-\alpha). \tag{6.11}$$

This reduces to the following inequality:

$$D_\mu \leq \sqrt{\frac{2(N_L+N_R)\sigma^2}{N_L N_R L^2} \ln \frac{1}{\alpha}}, \tag{6.12}$$

which may be interpreted as the range of D_μ values that are too *insignificant* to suggest evidence against H_0, the region interior hypothesis.

6.2.3 Color Similarity between Image Elements

Our final test paradigm determines if two image elements have color distributions that are "similar" enough to have arisen from the same color world entity, where an image element refers to either a single image pixel or a patch of image pixels. For this thesis, we shall use a *color similarity* test that compares only *mean* color ratios and local *color variances* of the two image elements being combined. In the case where an image element is a single pixel or a very small patch, we define its *color mean* and *color variance* to be the *color mean* and *variance* of its local neighbourhood, whose size is determined by Equation 6.8. More sophisticated extensions to our similarity test paradigm may compare higher order *color moments* as well.

To determine if two image elements have the same *color mean*, we use a color *difference of mean* significance test, similar in form to the region interior test for image pixels, described in the previous subsection. Qualitatively, the two test procedures are totally identical, except for the contexts in which they are being applied. Instead of computing and comparing *mean* color ratios for a pixel's two local neighbourhood halfs, our similarity test computes and compares *color means* for a pair of image elements. We shall not elaborate on the *difference of mean* test procedure any further, since it has already been adequately described earlier on.

We define the *angular color variance*, S, of a size N image patch to be:

$$S = \frac{1}{N}\sum_{i=1}^{N}[\mathcal{A}(\boldsymbol{c}_i, \boldsymbol{c}_\mu)]^2, \tag{6.13}$$

where $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$ are the individual pixel *colors* and $\boldsymbol{c}_\mu$ is the *mean* patch color. Assuming again that each pixel's angular noise distribution, $A = \mathcal{A}(\boldsymbol{c}_i, \boldsymbol{c}_\mu)$, obeys Equation 6.2, we get the following exponential form for the squared angular noise distribution, $A_2 = [\mathcal{A}(\boldsymbol{c}_i, \boldsymbol{c}_\mu)]^2$:

$$Pr_{A^2}(A_2) = \frac{L^2}{2\sigma^2} e^{-\frac{L^2 A_2}{2\sigma^2}}. \tag{6.14}$$

Taking the average of N independent A_2 measurements yields the following *Order N Erlang* distribution for S:

$$Pr_S(S) = \frac{1}{(N-1)!}(\frac{NL^2}{2\sigma^2})^N S^{N-1} e^{-\frac{NL^2 S}{2\sigma^2}}, \tag{6.15}$$

which can be closely approximated using an equivalent Gaussian of *mean* $\frac{2\sigma^2}{L^2}$ and *variance* $(\frac{2\sigma^2}{\sqrt{N}L^2})^2$, even for relatively small patch sizes, N. (See Figure 6.2).

We shall use a *difference of variance* significance test to determine if two image elements have the same *angular color variance*. Suppose the two image elements have sizes N_1 and N_2, and *measured* angular color variances S_1 and S_2 respectively[2]. Let H_0 be the hypothesis that the two image elements actually have the same *true* angular color variance, $S_T = \frac{2\sigma^2}{L^2}$. Using the equivalent Gaussian approximation for an N^{th} *order Erlang* to simplify our calculations, and assuming that H_0 is `true`, we get:

$$\begin{aligned} Pr_{S1}(S_1) &\approx \frac{1}{\sqrt{2\pi}\sigma_{S1}} e^{-\frac{(S_1 - S_T)^2}{2\sigma_{S1}^2}} \\ &\text{and} \\ Pr_{S2}(S_2) &\approx \frac{1}{\sqrt{2\pi}\sigma_{S2}} e^{-\frac{(S_2 - S_T)^2}{2\sigma_{S2}^2}}, \end{aligned}$$

where $\sigma_{S1} = (2/\sqrt{N_1})(\sigma_1/L_1)^2$ and $\sigma_{S2} = (2/\sqrt{N_2})(\sigma_2/L_2)^2$ are the derived standard deviations for the equivalent Gaussian distributions. Combining $Pr_{S1}(S_1)$ and $Pr_{S2}(S_2)$ yields a Gaussian *difference of angular variance* statistic, $S_D = S_1 - S_2$, with:

$$Pr_{SD}(S_D) \approx \frac{1}{\sqrt{2\pi}\sigma_D} e^{-\frac{S_D^2}{2\sigma_D^2}}, \tag{6.16}$$

where $\sigma_D = \sqrt{\sigma_{S1}^2 + \sigma_{S2}^2}$.

We say that two image elements have the same angular color variance if their measured S_D value is *insignificant* for the given α level. Graphically, Figure 6.3 shows the range of S_D measurements supporting the equal angular variance hypothesis, H_0. Notice that in this statistic, the "ideal" S_D value for H_0 is 0, so S_D is significant at very positive and very negative values. In other words, to verify H_0, we test for:

$$\int_{-S_D}^{S_D} Pr_{SD}(S)dS \leq (1-\alpha), \tag{6.17}$$

which, after some algebraic manipulation, may be re-expressed as:

[2]We can directly measure the angular color variance of an image patch by first computing its *mean* color vector and then applying Equation 6.13 over all pixels in the patch.

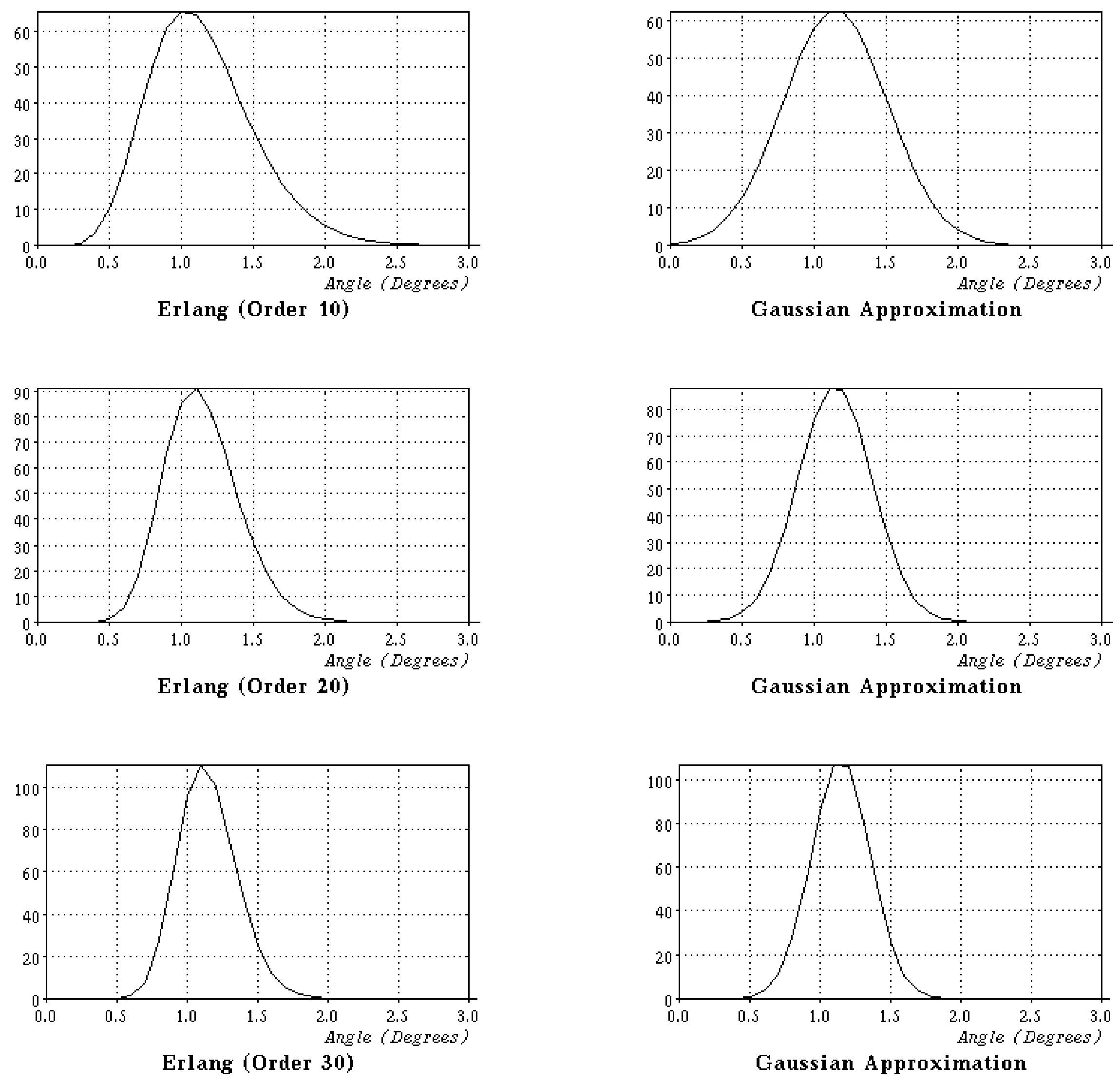


Figure 6.2: Approximating the Order N Erlang distribution with Gaussians for selected values of N. Top row: $N = 10$. Center row: $N = 20$. Bottom row: $N = 30$. Notice that the approximation gets better as N increases.

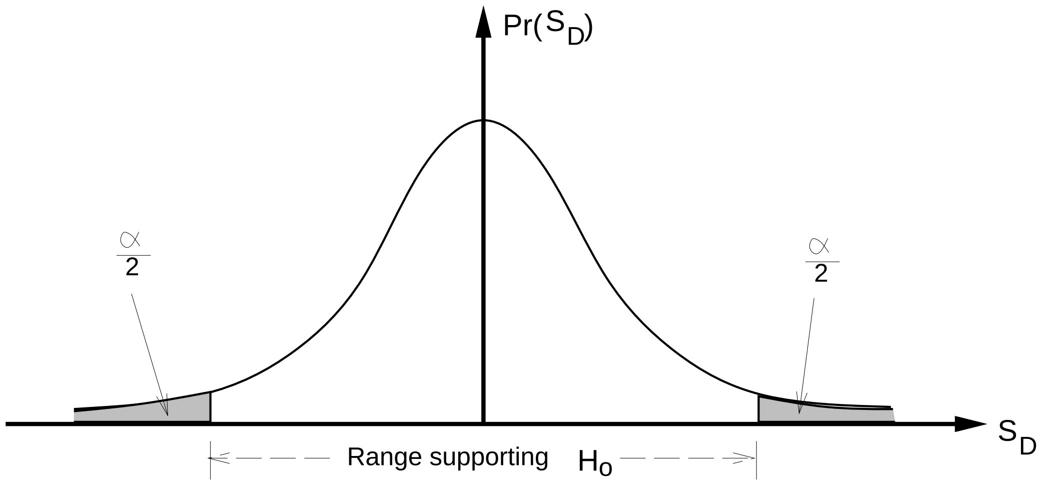


Figure 6.3: Range of *insignificant* S_D values supporting the equal angular color variance hypothesis, H_0, for a given significance level α.

$$\Phi(\frac{S_D}{\sigma_D}) \leq \frac{1-\alpha}{2}. \tag{6.18}$$

Here, $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{x^2}{2}} dx$ is the *cumulative distribution function* for the unit Gaussian probability distribution function, whose values can be easily obtained from standard mathematical tables.

6.3 The Overall System

This section lists the steps that make up our statistical region finding algorithm. Its purpose is twofold: First, it serves as a summary for the detailed confidence and significance tests we described in the previous section. Second, it gives the reader a clearer overview of how our statistical techniques fit into a traditional "seed" based region finding framework. Since it is also possible to replace our statistical tests with more sophisticated threshold and parameter setting techniques, our description below also provides a modular framework for possible improvements.

1. **Estimate the "detectable" patch size at each image location.** We compute the local $\frac{\sigma}{L}$ ratio at each image point by averaging angular color differences between adjacent pixels over a small fixed neighbourhood[3], and multiplying the result by $\sqrt{\frac{1}{\pi}}$. We then use the pixel's local $\frac{\sigma}{L}$ value with Equation 6.8 to obtain N, the local minimum "detectable" patch size. Our implementation uses a $\frac{1}{2}^{\circ}$ sensitivity level for ϵ and a 99.5% confidence level for C.

[3]In our implementation, we used a fixed radius of 3 pixels.

2. **Identify region interior pixels.** For each pixel, we examine the color distribution of its N nearest neighbours to determine whether or not it lies inside a color region. We apply the "insideness" significance test of Subsection 6.2.2 to check for color edges in four orientations, namely the horizontal, vertical and the two diagonal directions. If none of the four tests reveal the presence of a color edge, we consider the pixel an "interior pixel". Otherwise, we treat it as a "boundary pixel". To ensure that no "boundary pixels" get wrongly classified as "interior pixels", we must choose a significance level, α, that is not too small. A value of 0.01 works well in our implementation.

3. **Link adjacent interior pixels to form "seeds".** At the end of Step 2, each connected patch of "interior pixels" lies entirely within a single color region, and may be used as an image "seed" for color region finding. We consider two "interior pixels" *connected* if they are one of each others' eight nearest neighbours. Because Step 2 classifies pixels by examining their N nearest neighbours' color distribution, we expect that for a size N or larger color region, the "insideness" test should at least correctly label some of its interior pixels. So, regions that are size N or larger should contain at least one "seed", and should therefore appear in the segmentation output.

4. **Grow initial "seeds".** To consistently merge free neighbouring image pixels with growing "seeds", we must first check that both the free pixels and the "seeds" have similar color properties. We use the *difference of mean* and *difference of variance* significance tests described in Subsection 6.2.3 to check for color similarity between the pixels and "seeds". Recall that for an individual image pixel or a very small "seed", we define its color *mean* and *angular variance* to be the color mean and angular variance of its local size N neighbourhood. Our implementation uses a significance level of $\alpha = 0.005$.

5. **Combine adjacent "seeds" with similar patch colors.** We apply the same *difference of mean* and *difference of variance* significance tests of Step 4 when combining adjacent image "seeds". The algorithm repeats Steps 4 and 5 until no more "seeds" can be further grown or combined.

6.4 Results and Conclusions

6.4.1 Two Examples

We implemented a parallel version of our color region finder on the Connection Machine and tested it on a few real images. Figures 6.4 and 6.5 summarize the results we obtained on two test cases: a striped sweater image and a plastic scene. For ease of reference, we also include some intermediate region finding results that our algorithm produces.

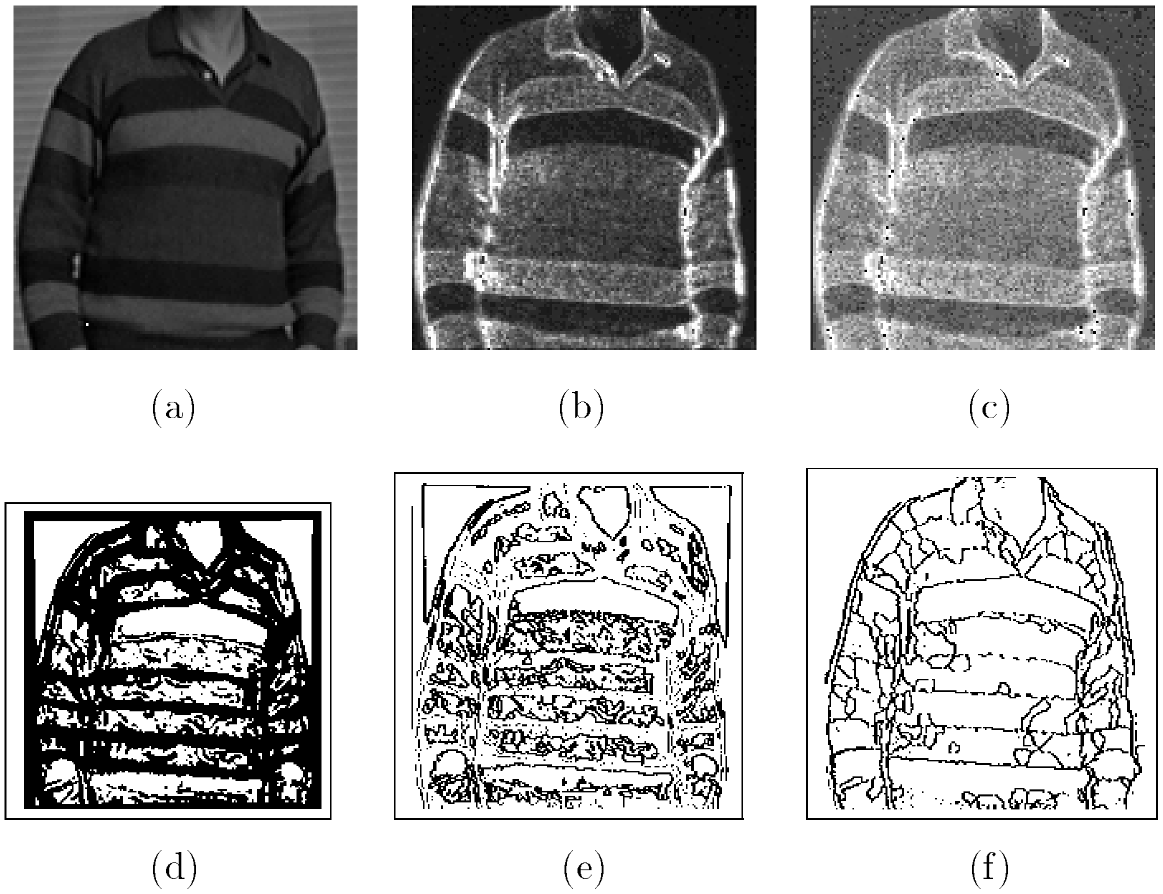


Figure 6.4: Segmentation results obtained from our color region finding algorithm. (a) Image of a color striped sweater. (b) Local angular color noise strength. Brighter patches indicate noisier pixels. (c) Relative minimum "detectable patch" widths. Brighter pixels indicate wider patches. (d) Interior pixels. (e) Image "seeds" obtained from linking adjacent "interior pixels". (f) Final regions.

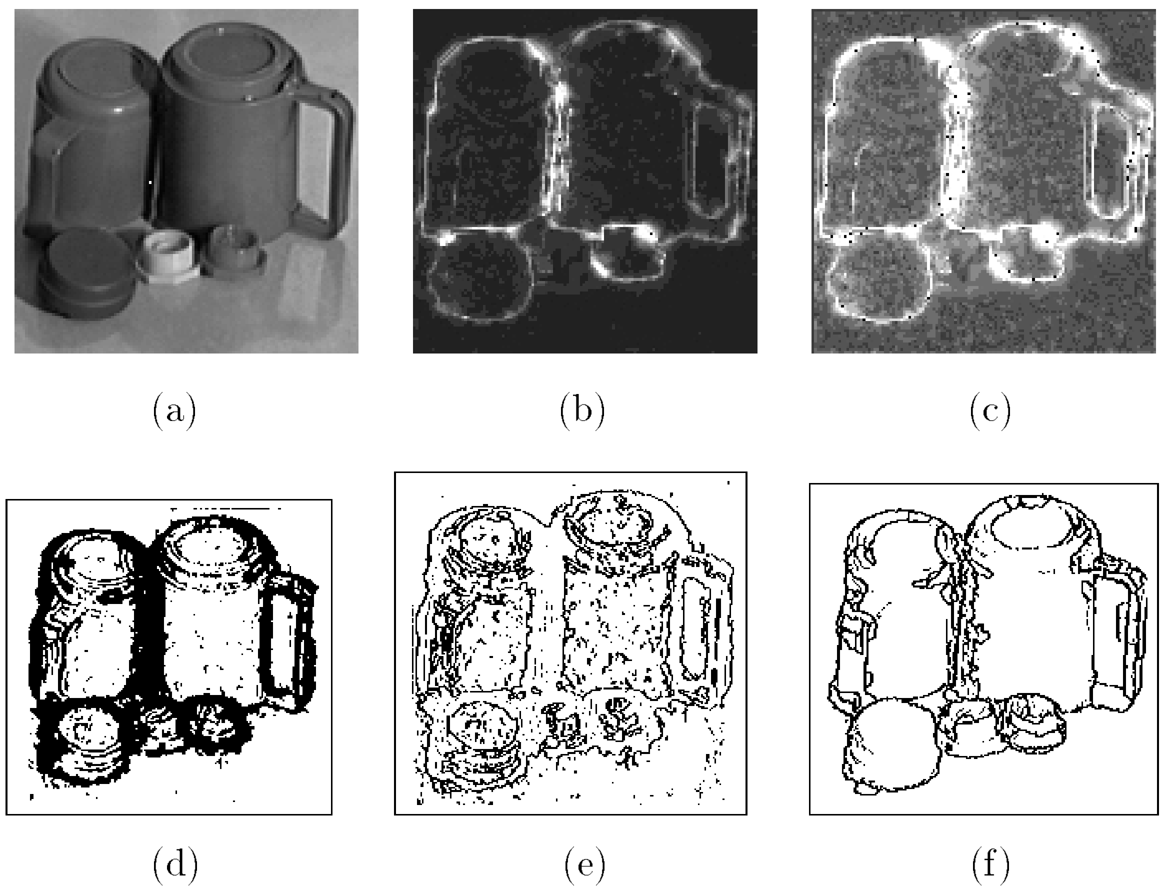


Figure 6.5: Segmentation results obtained from our color region finding algorithm. (a) Image of a plastic scene. (b) Local angular color noise strength. Brighter patches indicate noisier pixels. (c) Relative minimum “detectable patch” widths. Brighter pixels indicate wider patches. (d) Interior pixels. (e) Image “seeds” obtained from linking adjacent “interior pixels”. (f) Final regions.

The algorithm's first step estimates the strength of local angular color noise at each pixel, using the pixel's local neighbourhood color distribution. We display the output in Figures 6.4(b) and 6.5(b), where we encode noisier pixels as brighter patches. Notice that brighter pixels on the noise map correspond to darker locations in the image, as one might expect. Figures 6.4(c) and 6.5(c) map the derived minimum "detectable patch" widths at each pixel on a gray scale, where again, brighter pixels indicate wider patches. In Step 2, the algorithm classifies each image pixel as either an "interior pixel" or a "boundary pixel". Figures 6.4(d) and 6.5(d) display the classification results with "interior pixels" being highlighted. Step 3 connects adjacent "interior pixels" to form initial "seed", as shown in Figures 6.4(e) and 6.5(e). In Figures 6.4(f) and 6.5(f), we present the final color regions that our algorithm finds after iteratively applying Steps 4 and 5 on the image "seeds".

6.4.2 System Evaluation

We conclude this chapter by briefly evaluating our statistical region finding approach in the context of some previous work. Our approach is primarily motivated by the difficult task that traditional region finding algorithms face in selecting suitable segmentation thresholds and free parameters. To circumvent this problem, we developed a statistical region finding formulation that helps us determine suitable segmentation thresholds and parameter values automatically. Although the approach merely replaces one set of free thresholds and parameters with another set of free statistical parameters, we contend that the new set of parameters is more desirable as a set of adjustable system variables, because they provide greater insight to some of the system's important segmentation characteristics. Also, the approach intrinsically implements adaptive thresholding and parameter selection — a powerful technique for processing images with variable noise distribution.

Like most traditional uniformity based region finding techniques, our approach also suffers from *fragmentation* and *over-merging* effects. Fragmentation occurs when one wrongly divides a single color image entity into multiple final output regions. Over-merging does the opposite; it wrongly merges pixels from two or more color image entities into a single final region. We can find instances of both region *fragmentation* and *over-merging* in Figures 6.4(a) and 6.5(a), our final output maps for the two example images.

We believe that region finding algorithms will always suffer from *fragmentation* and *over-merging* effects, as long as one continues using only *local* decision procedures to operate on image elements. *Local* decision procedures are those that examine only *local* image attributes when performing segmentation tasks. Most computer vision researchers today believe that successful region finding algorithms must make use of *global* decision procedures as well, which includes taking into account general image information, like overall attribute distribution in the image as a whole. Unfortunately, the problem of integrating *global* and *local* image information for early level vision processes is still a poorly understood computer vision subject.

In the next chapter, we shall demonstrate a different region finding technique that makes use of intermediate image structures, called *region skeletons*, as a source of global image information. The technique produces better region finding results than our current statistical approach.

Chapter 7

Color Ridges and Salient Frames of Reference

7.1 Reference Frames — an Overview

For certain recognition and knowledge representation tasks, it is sometimes convenient to use a set of stick-like curves that run along local region centers or axes of symmetry, to geometrically describe the shape of image regions [Duda and Hart 73] [Pavlidis 78]. Such a description can serve as shape *reference frames* for making explicit certain geometric features present in the scene. These geometric features can then be used as important cues for detecting and inferring the presence of interesting image objects. Figure 7.1 shows two binary image region examples with their stick-like shape reference frames superimposed. Notice how each reference frame spans its region interior in describing its region's shape. Henceforth, we shall use the terms *reference frames* and *skeletons* interchangeably, because of the amazingly close structural resemblance between the two concepts.

Reference frames are an interesting notion in computer vision and robotics because of their wide range of possible applications. Voronoi diagrams [Canny and Donald 87], or distance reference frames, are helpful computational geometry tools for robot path planning and configuration space computation. In some object representation schemes, reference frames can serve as versatile primitive constructs for describing complex physical structures, an example being the use of generalized cylindrical reference frames for modeling objects with joints [Russell Brooks and Binford 79] [Nevetia and Binford 77] [Marr and Nishihara 78]. When dealing with elongated flexible objects, reference frames are useful as a means of finding stable canonical shape descriptions, because curved elongated shapes can be "straightened" into their canonical form by "unbending" them along their skeletons.

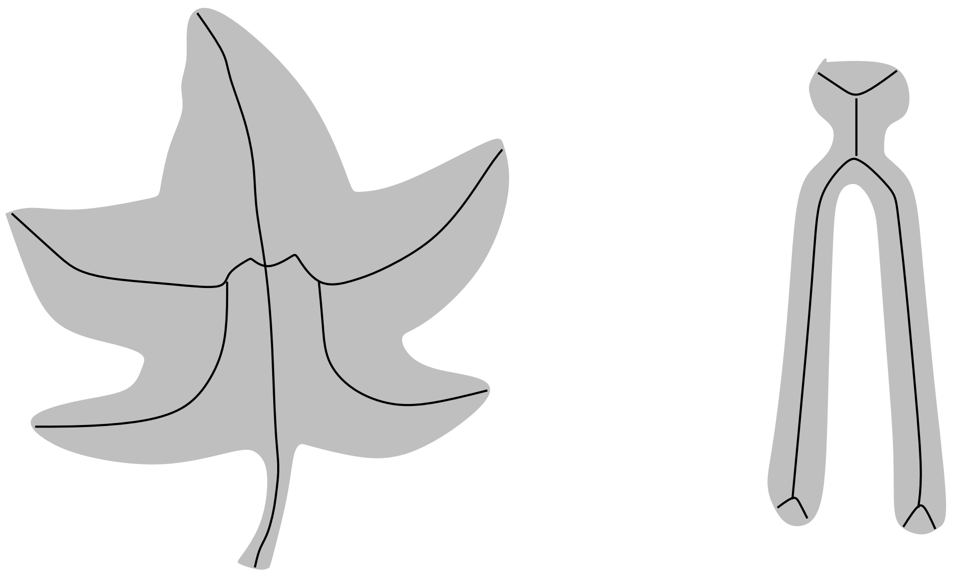


Figure 7.1: Two binary images and their reference frames [Cordella Arcelli and Levialdi 81].

7.1.1 Computing Reference Frames

Most reference frame algorithms today fall into one of two categories, both of which operate on either line drawings or binary images. The first class of algorithms focuses on preserving symmetry information in image regions. Skeletons found by this class of algorithms are normally computed using symmetry measurements of points from region boundaries, and their shapes are generally unaffected by small outline perturbations or minor curvature irregularities along the region boundaries. Examples of algorithms in this category include Smoothed Local Symmetries (SLS) [Connell and Brady 87] and region thinning algorithms [Tamura 78]. A second class of algorithms known as Symmetric Axis Transforms (SAT) [Blum 67] [Blum and Nagel 78] preserves original shape information of regions at the expense of skeletal structure smoothness. One way of computing SAT reference frames is by a method nicknamed the "brushfire" algorithm, details of which can be found in [Blum 67]. SAT skeletal maps are generally very sensitive to irregularities in the region outline.

Because all the above mentioned algorithms are edge-based methods that take distance or symmetry measures from image boundaries, reference frames cannot be computed until edge detection has been successfully performed. We see a major disadvantage in this approach, namely in its heavy dependence on edge detection results. Figure 7.2 shows an example of unstable edge detection results, where certain edge segments of an image can disappear and then re-appear again across scales. Since we usually do not know a-priori what an appropriate edge detection scale might be for each part of an image, we can expect edge detection algorithms in general to miss finding discontinuities along some physical edges. This in turn gives rise to poorly formed reference frames

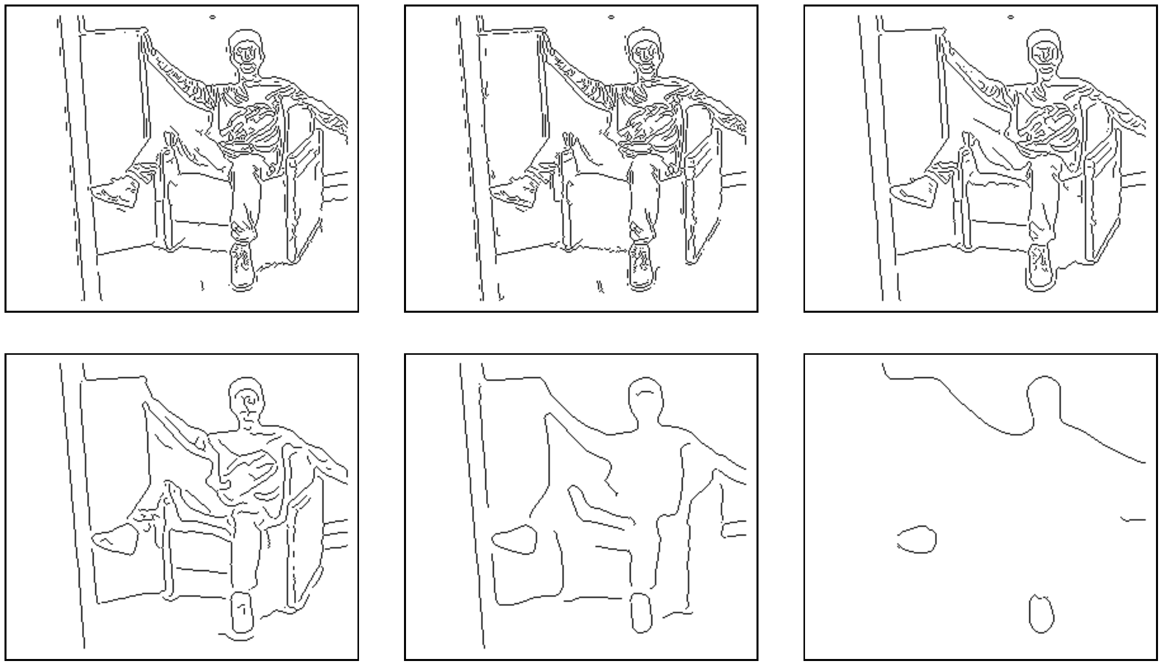


Figure 7.2: Canny edge maps of an image at 6 different scales. Notice how certain edges can disappear and re-appear across scales.

for the affected regions. Since humans are capable of providing reasonably close shape descriptions even for regions with fuzzy boundaries, this suggests that in principle, skeletal maps can still be computed without edge-based information.

7.1.2 Color Reference Frames

In this chapter, we demonstrate how reference frames can be computed directly from color image irradiances, without having to make use of any edge-based information. Our intention here is twofold: First, we want to show that reference frames can indeed be computed using a purely region-based approach, hence avoiding the problem of relying on edge-detection results altogether. Second, we want to extend the concept of *reference frames* into the *color* domain, just as other physical concepts, like regions and boundaries, have been used with color data. The reference frame algorithm that we design will therefore operate on *color* values instead of intensity values, and the skeletons we find will be uniform surface color skeletons instead of intensity region skeletons. The main advantage we get here is one of producing better semantic shape descriptors for our images. As alluded to in Chapter 1, color regions tend to correspond much better to physical entities in the image than intensity regions, so color reference frames should also serve as much better shape descriptors for objects in the scene than intensity skeletons.

The rest of this chapter will be organized as follows: First, we introduce a new color notion, called a *color ridge*, that we use as a model of *color uniformity* for detecting uniform color regions. Next, we describe an algorithm for detecting color ridges and ridge centers in an image, which we use as a means of locating uniform color region centers and axes of symmetry for constructing skeletons. Then, we modify an existing edge-based reference frame algorithm to compute skeletons of uniform color image regions in a purely region-based fashion. Finally, we show some skeleton finding results together with an interesting application of color skeletons in color region growing. We shall see that the segmentation results we get here are in fact superior to those produced by the statistical methods of Chapter 6.

7.2 Color Ridges

This section formally presents the *color ridge* notion as a color image feature. What is a *color ridge*? How does the feature model uniform color regions? What makes the feature "ridge like" in the conventional sense? These are some of the issues that we will address.

7.2.1 A Color Uniformity Model

To compute color reference frames using a *region-based* approach, we must first have an appropriate model of *color uniformity* so that uniform color regions can be treated as interesting image features just like edge segments or T-junctions. We can then think of the reference frame problem as a dual problem to boundary detection, where our task here is to find and link together salient stretches of color uniformity features to form skeletons. Figure 7.3(a) shows a 1-dimensional description of our *color uniformity* model, called a *color ridge*. Its structure can be described as a central band of uniformly colored pixels surrounded by bands of different color. A cross section of its scalar analogue, an intensity ridge, appears in Figure 7.3(b).

We will be using the *color ridge* feature of Figure 7.3(a) to model uniform color regions in 1D as follows: The central band of uniformly colored pixels corresponds to points within the uniform color region itself. The two differently colored side bands set spatial limits to the extent of color uniformity in the region. They help us estimate the image width of the uniform color region, and also the location of its center which we use for constructing region skeletons.

How then is the *color ridge* feature of Figure 7.3(a) related to the scalar ridge notion of Figure 7.3(b)? To answer this question, let us first consider a commonly accepted understanding of the *ridge* concept, which is a central elevated surface that rises sharply above two adjacent side surfaces. In this definition, one implicitly assumes that the quantity being measured is scalar, and that points on a ridge surface have greater values than points off the ridge platform. For grey-level intensity images, the conventional *ridge* concept relates easily to physical regions in an image having uniformly high intensity values relative to their local neighbourhoods. With color data however, the *ridge* notion becomes less clearly defined, because color is not a scalar value and

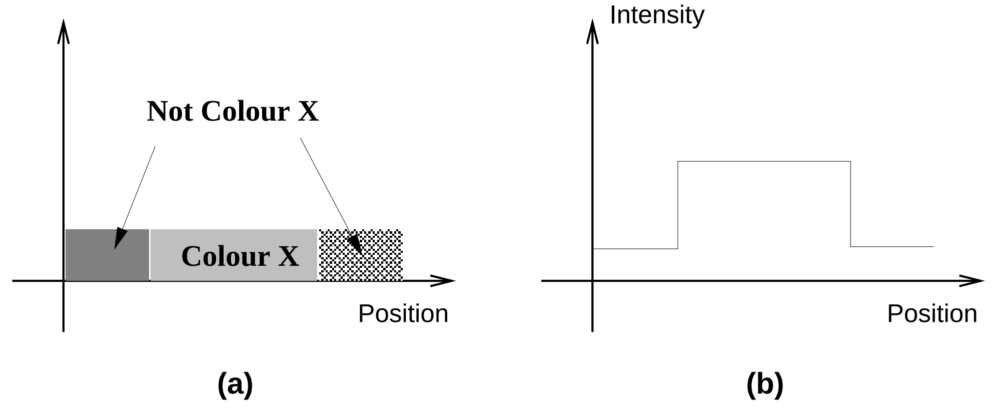


Figure 7.3: (a) Semantic description of a color ridge. (b) Cross sectional plot of an intensity ridge.

there is no clear "greater than" relationship in color. In order to have *color ridges*, as we do in Figure 7.3(a), it appears that we must first adopt a different *ridge* description that does not depend on the existence of a "greater than" relationship in the quantity being measured. Alternatively, we can linearize the color space, so that regions with pixel colors that map uniformly and sharply high onto an "absolute color scale" can be treated as "ridges" in the color sense. We shall see that both approaches in fact successfully reconcile our notion of a *color ridge*, as depicted in Figure 7.3(a), with the traditionally accepted *ridge* notion, as shown in Figure 7.3(b).

7.2.2 An Alternative Ridge Description

A helpful way of envisioning *color ridges* is to describe *scalar ridges* using only primitive relationships that are also defined in the color domain. We shall introduce an alternative ridge description that does not depend on the existence of a "greater than" operator. This new description only makes use of *difference measures*, a relationship that is well defined both in the scalar domain and in the color domain.

Figure 7.4(a) shows the cross sectional profile of a scalar ridge feature. In Figure 7.4(b), we have a different graphical representation of the *ridge* feature in Figure 7.4(a), which we shall refer to as a *difference profile*. The difference profile of a window displays difference measures between points in the window and a chosen reference value. It can be computed by first choosing a reference value for the window from points near its center, and then plotting *relative* differences between the reference value and values of points on the profile. For a well centered *ridge* feature, points on the raised platform should have very small difference profile readings, because their values are very close to the window's reference value. Points off the ridge platform should have large difference readings because their values are very different from the window's reference value. In short, the *difference model* describes a *ridge* feature as a central uniform band of points with low difference

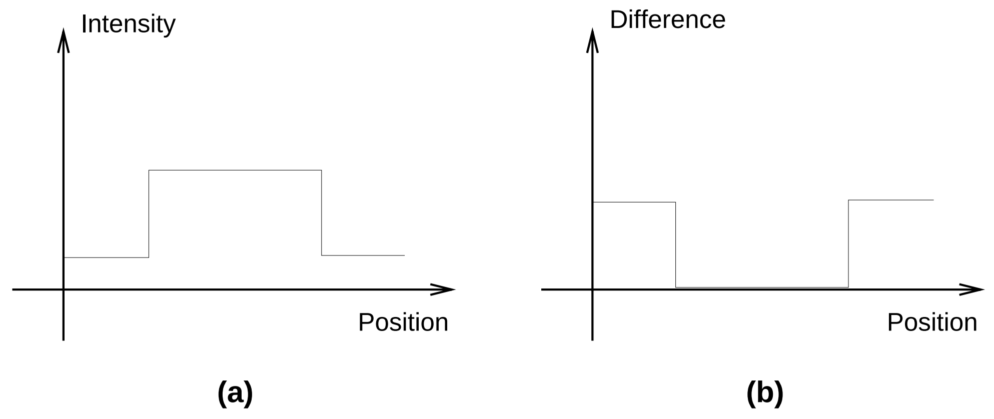


Figure 7.4: (a) Cross sectional profile of a ridge. (b) Cross sectional *difference* profile of ridge in (a)

values, surrounded by points with high difference values. Notice that the *difference representation* conveniently gets rid of the need for a "greater than" relationship in the quantity being described.

The *difference model* is general enough to describe our proposed notion of a *color ridge*, if the ridge feature is well centered within the reference window. If we choose a representative color from the central band of our color ridge as a reference value, and use the *angular* measure developed in Chapter 3 to compute color differences, then all points within the central band should produce low difference profile readings because they all have colors that are very close or identical to the window's reference color. Likewise, points outside the central ridge band should have high difference profile readings because their colors are very different from the reference color. A color ridge can therefore be described as a central band of uniformly colored points whose colors are very similar to the "ridge color", surrounded by points whose colors are very different from the "ridge color".

7.2.3 Absolute Colors

To reason about *color ridges* as scalar ridges, we need a means of quantifying color in some "absolute" sense, so that we can have a "greater than" relationship for color values. One possibility is to use a context dependent linearizing transform that assigns scalar similarity measures to colors in the color space, based on the color of the ridge we are detecting. Colors that are very similar to the "ridge color" get assigned high similarity values while colors that are different from the "ridge color" get mapped to low values. Equation 7.1 presents a suitable similarity measure ($\mathcal{S}_{\odot}$) for the linearizing transform, where $\boldsymbol{c}$ is the color vector being compared, $\boldsymbol{c}_R$ is the context dependent reference color and $\odot$ stands for the *vector dot product* operation.

$$\mathcal{S}_{\odot}(\boldsymbol{c}) = \frac{\boldsymbol{c} \odot \boldsymbol{c}_R}{|\boldsymbol{c}||\boldsymbol{c}_R|} \tag{7.1}$$

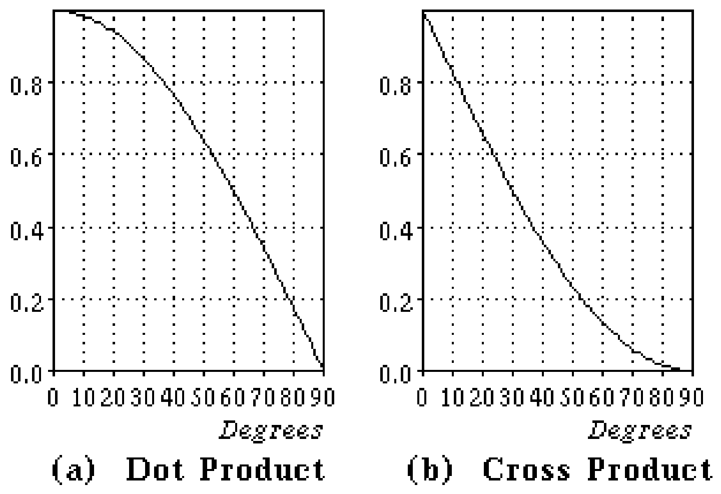


Figure 7.5: Similarity Measure as a Function of Angular Difference for using: (a) Normalized Vector Dot Products, and (b) Normalized Vector Cross Products.

Equation 7.2 presents another similarity measure ($\mathcal{S}_\otimes$) that responds more sensitively to color dissimilarities near the reference color. The symbols are as defined in Equation 7.1 with $\otimes$ denoting the *vector cross product* operation.

$$\mathcal{S}_\otimes(\boldsymbol{c}) = 1 - \frac{|\boldsymbol{c} \otimes \boldsymbol{c}_R|}{|\boldsymbol{c}||\boldsymbol{c}_R|} \tag{7.2}$$

Both similarity measures are decreasing functions with respect to the angular color difference measure of Chapter 3. They assign a maximum value of 1 to colors that are identical to the reference "ridge color", $\boldsymbol{c}_R$, and a minimum value of 0 to colors that are orthogonal to $\boldsymbol{c}_R$ in the RGB vector space. Figure 7.5 shows the relationship between the two similarity measures and the angular color difference measure. It is easy to see that the overall linearizing operator transforms color ridges into scalar ridges that look like the profile in Figure 7.3(b).

We can summarize the analogy between color ridges and scalar ridges as follows: When defining ridges of a certain "ridge color", we use a "goodness" scale that ranks colors according to how similar they are to the given "ridge color". As in defining scalar ridges, we then seek a uniform central platform of points whose colors rank high on the "goodness" scale, surrounded by points whose colors rank low on the scale. Since color ridges can be of any color in general, our "goodness" scale must change each time we want to define a ridge of a different "ridge color".

7.3 Color Ridge Detection

How do we detect color ridge features in an image? In this section, we present a color ridge algorithm that helps us locate uniform color region centers and axes of symmetry for finding color skeletons. We shall proceed by considering first the results we want from our ridge detection process, following which we determine how we can best go about performing the actual ridge detection task. Although work like this falls under a general class of computer vision problems known as *feature detection*, we shall not attempt to address any optimality or efficiency issues concerning color ridge detection here, because similar issues have already been addressed by others in work done elsewhere [Canny 83]. Also, it turns out that we do not really need very high quality ridge detection results to compute acceptable color reference frames. Instead, our main purpose is to show how the *ridge* concept can be extended into the color domain, and how *color ridges* can be treated and successfully detected as an image feature through a simple ridge detection technique.

For the sake of simplicity, we shall just analyze the color ridge detection process in a one-dimensional domain, and derive filters that are purely one-dimensional in form. Although images generally contain two-dimensional entities, our simplified analysis does not affect our reference frame algorithm in any way, because as we shall see later on, all the color ridge detection tasks that we have to perform are purely one-dimensional in nature. Even if our tasks require us to perform operations in 2D, the one-dimensional operators that we derive can still be effectively employed as directional 2D operators.

7.3.1 The Operator Output

Because a one dimensional ridge feature has two spatial components, namely a *width* component and a *location* component, our ridge detection process must be able to recover both spatial components of a color ridge for the sake of completeness. Unfortunately, as in the case of intensity ridge detection [Canny 83], we cannot synthesize a single operator mask that responds positively to ridges of all possible sizes. To detect color ridges of all possible widths, we need to adopt a multi-scale approach that uses a family of masks to account for ridge features over the entire scale space. For most of this section, we shall only focus our attention on building an operator that detects color ridges of a single fixed width. To generalize the approach for ridges of other widths, we can simply use a similar operator design with its linear dimensions appropriately scaled. Ideally, our ridge operator should exhibit the following two characteristics:

1. If the width of a ridge feature matches the width of the operator mask, then the magnitude of the operator output should peak near the center of the ridge platform and decrease rapidly towards the sides of the platform. In other words, the operator output should behave like a probability measure for estimating the center of the ridge platform. We shall define the *location* of the ridge feature as the peak output location on the ridge platform.

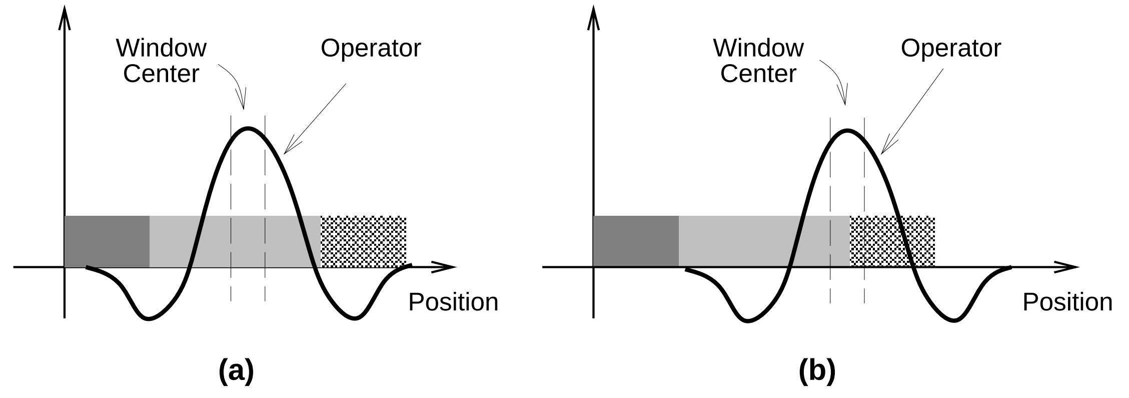


Figure 7.6: An operator window that is centered (a) within a color ridge and (b) near the edge of a color ridge. In case (a), points near the center of the window give a good estimate of the overall "ridge color". In case (b), points near the window center give a poor estimate of the overall "ridge color".

2. When a color ridge is processed by masks of different sizes, the strongest peak response at the center of the ridge feature should come from the filter whose width best matches the width of the ridge. That is to say, if we do not know the *width* of a ridge, we can deduce its *width* by measuring the linear dimensions of the filter mask that produces the strongest peak response when applied to the ridge.

7.3.2 A Scalar Approach

As far as possible, we shall design our color ridge operator to emulate the behaviour of a scalar ridge detector. Each application of the operator on some part of the image proceeds in two steps. The first step linearizes the operation by transforming local colors around a pixel into scalar values. The second step involves convolving a scalar mask with the linearized data, so that points lying near the center of a color ridge will produce high output values.

During the first step, we need to determine a suitable linearizing transform for pixel colors within the operator window. This task amounts to "guessing" an appropriate "ridge color", to be used as a reference vector for our similarity transform. Suppose the center of our operator window is reasonably well aligned with the center of a color ridge (see Figure 7.6(a)), we can get a fairly good estimate of the "ridge color" by averaging pixel colors near the center of the window. This is because points near the center of the window map to points on the ridge platform whose colors are reasonably close to the overall "ridge color". If our operator window is centered near a color boundary off the center of a ridge platform (see Figure 7.6(b)), we cannot get a reasonable "ridge color" estimate using the same averaging method, because we are likely to average pixel colors from both sides of the color boundary.

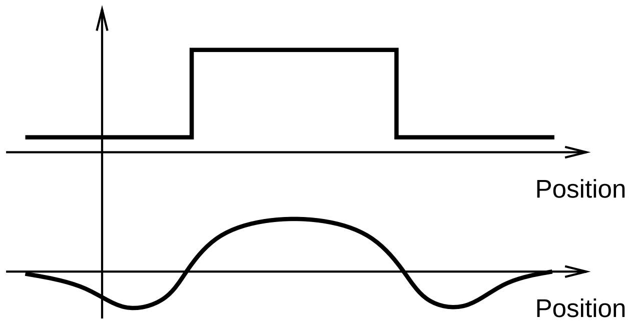


Figure 7.7: A scalar ridge profile and its optimal operator.

To ensure that the operator responds favourably only when we have a reliable "ridge color" estimate, our approach checks that the window is sufficiently well centered within a color ridge while "guessing" the "ridge color". It performs the check by computing a local color gradient at the window center to determine if the window center is indeed sufficiently far away from a color boundary. If the local color gradient is large, it assumes that the window is centered too near a color boundary for a reliable color estimate, and so it introduces a heavy penalty on the final operator output. If the gradient is small, it assumes that the window is indeed well centered within a color ridge, and "guesses" a reference color by averaging pixel values within a small fixed distance[1] from the window center. We use Equation 7.2 to linearize local pixel colors within the operator window. For a well centered window, we expect the local linearized data within the window to appear as a scalar ridge profile.

The second step convolves a scalar ridge mask with the linearized data to produce an operator output. Intuitively, we want a mask pattern that has the following properties: When centered on a ridge feature, the portion of the mask containing the ridge platform should be positive, so that high similarity values on the ridge can contribute positively to the operator output. Portions of the mask off the ridge platform should be negative so as to favour low similarity colors off the ridge platform.

Figure 7.7[2] shows a scalar ridge profile together with its optimal operator, obtained by applying numerical optimization on the operator's impulse response. For the purpose of generating inertia maps in this thesis, it suffices to use a more easily computable mask pattern that still detects ridge features fairly well. One possibility is to use a normalized Gaussian second derivative mask whose distance between zero-crossings (2σ) equals the width of the ridge we are trying to detect:

$$G_{2D}(x) = (1 - \frac{x^2}{\sigma^2})\frac{1}{\sigma^3}e^{-\frac{x^2}{2\sigma^2}} \tag{7.3}$$

[1]We use $\frac{1}{4}$ the radius of the filter's central lobe.
[2]From Chapter 4 of [Canny 83]

The mask outline is similar to the optimal ridge detector profile, but has an added advantage of being easily computable and scalable in real time. Figure 7.8 shows the results we get by using Gaussian second derivative masks with our color ridge algorithm for detecting an ideal and a noisy color ridge. Mask sizes ranging from a width of 20 to a width of 120 were applied at each ridge pixel location and the maximum output value across scales is recorded as the final result. The final result is fairly stable under additive white noise and has a signal of the desired form — one that peaks near ridge centers and diminishes towards color boundaries.

7.3.3 Non-Linear Convolution

As this point, there are still two improvements that we can make to our mask design for better operator results. The first improvement has to do with narrowing operator response widths, so that sharper peaks can be formed at color ridge centers. Although the Gaussian second derivative produces output patterns that peak near ridge centers, we want our output readings to decrease even more rapidly toward color boundaries, so that ridge centers can be more precisely located even in the presence of noise. Sharper peaks near region centers in turn give rise to better formed reference frames that keep closer to region axes of symmetry.

An easy and effective way of sharpening ridge outputs can be accomplished by using *non-linear* operator masks. In Figures 7.9(a) and 7.9(b), we see a ridge feature being convolved with two masks of mismatched widths. Although it is clear from the figure that both masks are not well aligned with respect to the ridge center, we get, in both cases, an incurred output penalization that arises only from the misalignment between the right half of the mask pattern and the ridge feature. To increase misalignment penalties up to twice the original amount, we use a technique that separately convolves the left and right portions of a mask with a ridge feature, so that we can have access to the results of both "half convolutions". At each spatial location, the technique compares the two convolution results and outputs twice the smaller of the two values. So, if $R(x)$ is the ridge feature and $G_{2D}(x)$ is the equation of a Gaussian second derivative mask, the "convolution" output can be mathematically expressed as:

$$NL(x) = 2\min\left(\sum_{i<x} R(x-i)G_{2D}(i), \sum_{i>x} R(x-i)G_{2D}(i)\right) + G_{2D}(0)R(x) \qquad (7.4)$$

The min operator in Equation 7.4 makes the resulting convolution *non-linear*. From the overall process description, we can see that if the mask is centrally aligned with a symmetric ridge, the output we get will be the same as the output we obtain from a normal linear convolution, because both halves of the *non-linear* convolution produce identical results. That is, the *non-linear* convolution process does not affect the peak response that we can get from a symmetric ridge feature. For masks located off ridge centers however, we do get a greater output attenuation than what we would otherwise see, because we are taking min values for the two "half convolutions". In the best

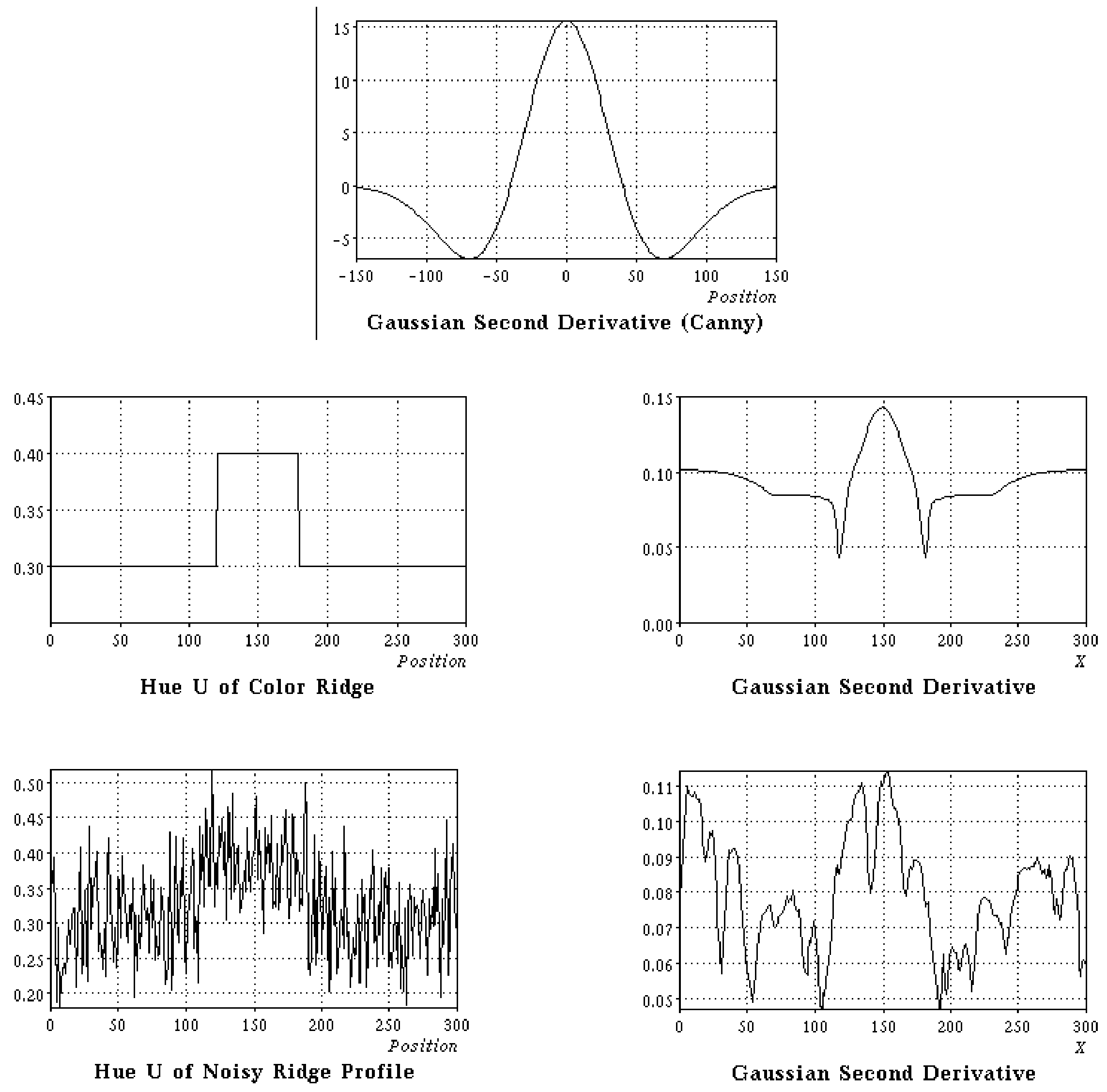


Figure 7.8: Top Row: Second Derivative of Gaussian mask for ridge profiles. Middle Row: Hue U channel of an ideal color ridge with color ridge operator response. Bottom Row: Hue U channel of a noisy color ridge with color ridge operator response.

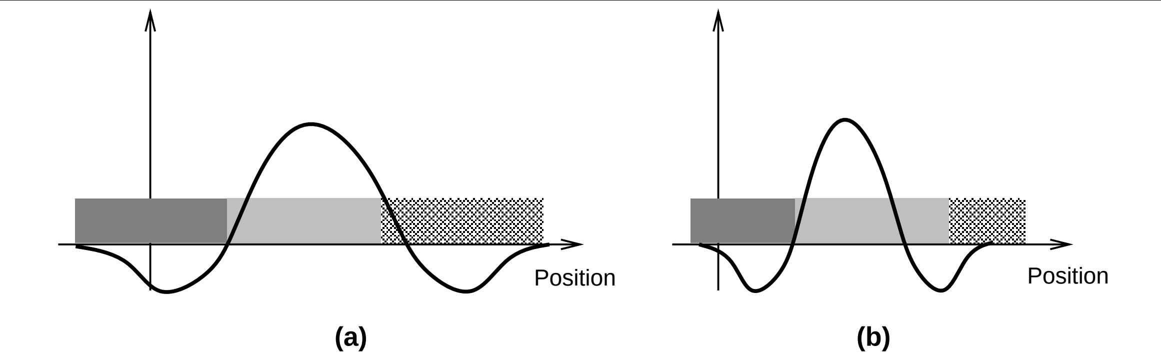


Figure 7.9: A mask that is (a) too big, (b) too small for the ridge. Only the right half component of the convolution gets penalized in both cases.

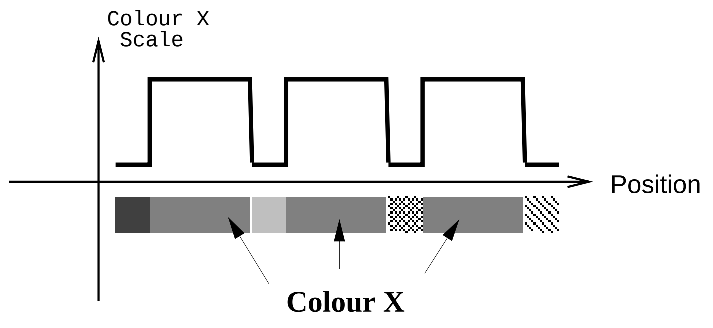


Figure 7.10: Wide color ridges with narrow valleys, corresponding to alternating wide and narrow color bands.

case, we get twice the attenuation of a linear mask when only one half of the linear convolution experiences attenuation, as in the examples of Figure 7.9. Our ridge outputs therefore decrease more rapidly away from ridge centers, towards color boundaries.

7.3.4 Operator Support

Until now, we have assumed that color ridges are spatially well separated enough, so that we can use operator supports that are as large as a few ridge widths. In Figure 7.7 for example, the operator support for the ideal ridge mask is three times the width of its target ridge feature. To get near zero DC-components with a Gaussian second derivative mask, we also need an operator support that is at least three times the zero-crossing distance of the mask, or three times the targeted ridge width! In real images where adjacent color regions can have very different widths, as in the example cross section of Figure 7.10, a large operator support can result in poorly formed peaks, especially if the alternate wider ridges have very similar “ridge colors”.

The second improvement adapts our mask pattern for detecting wide ridges better in the pres-

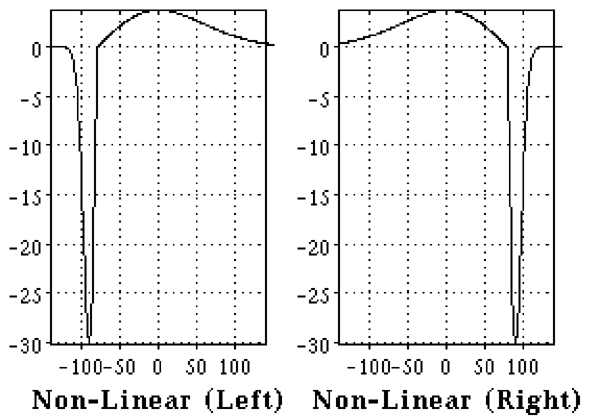


Figure 7.11: Left component and right component of improved mask pattern for a color ridge of width 150. Each central-lobe is a normalized Gaussian first derivative of $\sigma = 75$ and each side-lobe is a Gaussian first derivative of $\sigma = \frac{75}{8}$.

ence of narrow valleys. The main idea is to use narrower and deeper side-lobes in our mask pattern, so that we can still achieve near zero DC output components with a much smaller operator support. At the same time, we also do not want side-lobes that are too narrow, otherwise the operator output will be too sensitive to minor color fluctuations near the sides of the mask. In our implementation, we get relatively stable results using *normalized* side-lobes whose standard-deviations (σ) are one-eighth the main-lobe standard-deviation. It requires an operator support of only $1\frac{1}{2}$ times the ridge width.

To combine the two improvements discussed above, we use a pair of *Gaussian first derivative-like* masks as shown in Figure 7.11 to perform our non-linear convolution operation. The mask pair achieves the same overall qualitative effect as two Gaussian second derivative halfs with compressed side-lobes. Figure 7.12 shows some examples of the sharper and undistorted results we get using the non-linear mask pair instead of the Gaussian second derivative mask.

7.4 Finding Color Reference Frames

This section describes a reference frame algorithm that operates on region-based color data. It models uniform color regions as 1D color ridges and performs color ridge detection to locate region centers.

7.4.1 A Saliency Network Approach

Our skeletal map algorithm is a modified version of Subirana-Vilanova's saliency based scheme that finds image reference frames from line drawings or edge maps [Subirana-Vilanova 90b] [Subirana-Vilanova 90a].

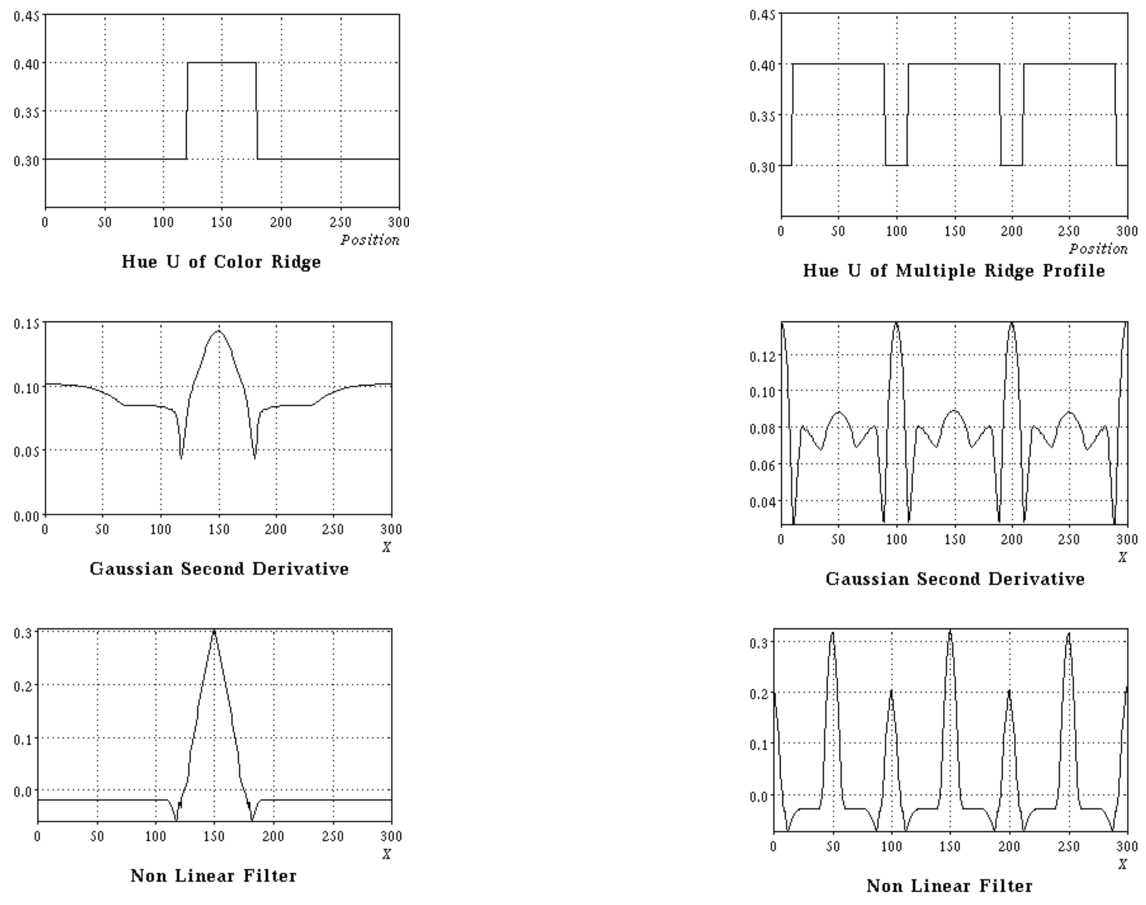


Figure 7.12: Top Row: Color ridge profiles. Center Row: Ridge operator response with Gaussian second derivative mask. Bottom Row: Ridge operator response with improved non-linear mask.

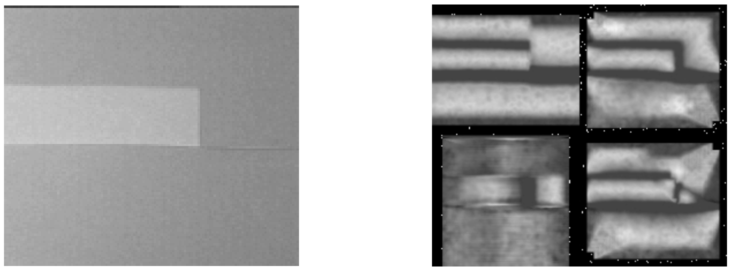


Figure 7.13: Left: Image of 3 rectangular regions. Right: "Insideness" or Inertia maps for the rectangular regions. From the top-left in a clockwise direction, the reference directions are: North-South, East-West, Northeast-Southwest and Northwest-Southeast. Brighter points have higher *inertia* values.

Saliency nets [Sha'ashua and Ullman 88] are dynamic programming algorithms that operate within a graphical framework for finding curves that maximize a certain quantity in an image. For the purpose of finding region skeletons, this quantity can be a combined measure of a curve's smoothness and its "insideness" with respect to the boundaries of some enclosing image region. Subirana-Vilanova demonstrated experimentally that curves found using this maximizing criterion usually turn out to be excellent reference frames for the regions they traverse.

To quantify the "insideness" of points within an enclosing region, Subirana-Vilanova proposes a directional local symmetry measure, called *inertia value*, that shows how deep an image point is within its enclosing region. Points lying deep inside a region near a *local* axis of inertia have high *local* symmetry measures, and so have high *inertia values* perpendicular to the axis direction. Points lying near the boundary of a region, far away from an axis of symmetry, have low symmetry measures perpendicular to the axis direction, and therefore have low *inertia values*. For any orientation, a local *inertia* maximum indicates that the point lies exactly at the center of its enclosing region on the axis of symmetry. Figure 7.13 shows the "insideness" maps or *inertia surfaces* of an image with three distinct regions. The inertia surfaces are computed in the four cardinal directions, parallel to the sides and diagonals of the image. Brighter points on the maps have higher "insideness" or *inertia* values than darker points.

To control the smoothness of a curve, Subirana-Vilanova defines another positional and directional quantity, called *tolerated length*, which penalizes the "goodness measure" of curves that bend excessively relative to the shape of their enclosing regions. The *tolerated length* for a curve at a point is a function of both the curve segment's local curvature and the enclosing region's local width perpendicular to the curve (see Figure 7.14). High curvatures and wide enclosing regions give rise to low tolerated lengths, which in turn give rise to high penalization factors. Essentially, this constraint only allows high curvature segments to exist along narrow sections of an enclosing region, just as narrow bodies tend to be more flexible than thick bodies.

The full skeleton finding process proceeds in three steps:

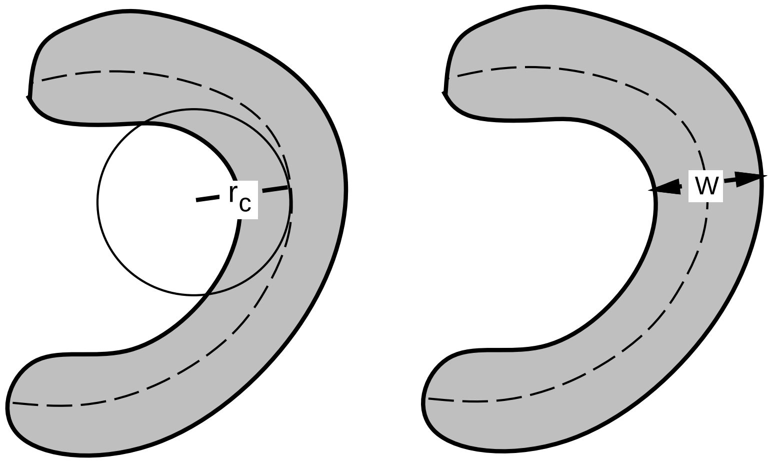


Figure 7.14: Radius of curvature (r_c) and local region width (W) — factors affecting a curve's tolerated length.

1. **Compute inertia surfaces and region widths:** Inertia values and local region widths are computed over the entire image at a fixed number of equally spaced orientation intervals. At each point and for each orientation, the algorithm uses the computed inertia value as an initial "goodness" estimate for finding skeletons that pass through the point in the given direction.

2. **Perform local network saliency computations to generate curves:** The saliency network consists of a two dimensional grid of processors where each processor holds all the local state information of an image pixel. The computation finds for every image pixel and everyone of its outgoing orientations, the most salient curve starting at that pixel with that local orientation. During each network iteration, each processor updates its own state by examining its own "goodness" value and inheriting some of its neighbours' "goodness" values. At the end of the n^{th} iteration, the network stores the "goodness" measure of the most salient size n curve that leaves each image pixel in each direction. Local widths are used to compute *tolerated lengths* at each pixel direction for controlling line segment curvatures. Only long smooth curves that stay within the interior of image regions become salient at the end of this stage.

3. **Extract region skeletons from the saliency network:** The curve with the highest saliency or "goodness" measure is identified in the network. Usually, such a curve can traverse a few image regions, with the first region contributing most to its saliency value. We extract the portion of the curve that lies within the first region as a skeleton for the region. The remainder of the curve is discarded because the curve could have entered subsequent regions in a highly asymmetric or non-central manner, hence forming unsatisfactory region skeletons.

In Subirana-Vilanova's test examples where a region is either an isolated line drawing or a binary image pattern, the task is relatively straight-forward because we can simply truncate a curve where it first crosses a line boundary or a binary threshold. This process is repeated for the next most salient curve until a sufficiently large portion of the image has been accounted for.

A more detailed description of the Saliency based reference frame algorithm can be found in [Subirana-Vilanova 90b] and [Subirana-Vilanova 90a].

7.4.2 Modifications for Region-Based Color Data

Subirana-Vilanova's algorithm computes *inertia surfaces* and width estimates from edge-based information by taking direct distance measurements from the line drawings or edge maps it works with. To augment the system for region-based color data, two extensions must be made to the existing implementation.

First, we need a means of computing inertia surfaces and local widths directly from image color. Our method should be *region-based* and not *edge based.* That is, we do not want to use color edges anywhere within our computation. Instead, we want to be able to generate inertia surfaces and local width estimates directly by detecting and measuring uniformly colored image regions.

Second, we need a different heuristic for truncating salient curves in Step 3. Since our truncation procedure has direct access only to color image irradiances, it must be able to truncate curves reliably by examining only color changes along their paths. In particular, it must be able to break curves where color differences are consistently large enough to be caused by region boundaries, and not truncate curves within regions, even though local color differences may be large because of noise.

It turns out that we can use the *color ridge* notion and the *color ridge detection* process we developed earlier to implement the 2 extensions above. We have seen in Chapter 3 that unlike intensity irradiance or depth values, surface color readings tend to be relatively constant within uniform material regions, even across sharp orientation changes or lighting shadows. So when traversing a path in an image, we can expect to see very similar surface color values along portions of the path within the same image region, and sharp surface color changes where the path crosses region boundaries. In other words, we can expect the surface color profile along any arbitrary image path to appear very "ridge like", where each *color ridge* corresponds to a portion of the path that lies entirely within a single color region.

Our two extensions can therefore be recast as the following *color ridge* problems: To compute a point's *inertia value* in some direction, we first take a local image cross section at the point in the given direction, which should appear as a *color ridge* feature. We then determine how deep the point is within its enclosing region by performing ridge detection on the *color ridge* profile. The point will be assigned a high *inertia* value if it is located well within the ridge interior and a low *inertia* value if it is located near an edge. So, to compute directional inertia surfaces for an image, we simply perform *directional* color ridge detection using the 1D color ridge operator we developed earlier on all points in the image. The output we get is the inertia surface for the chosen operator direction.

To extract region skeletons from salient curves, we "unbend" each salient curve into a 1 dimensional color profile and treat its color distribution as a sequence of color ridges. Each ridge belongs to a separate region of the image that the curve traverses. Since we only want a reference frame for the first region the curve traverses, we simply identify the portion of the "unbent" curve that belongs to the first ridge profile and truncate it at the end of the ridge profile. This amounts to performing color ridge detection on the curve's "unbent" color profile, and cutting the curve where the output is lowest between the first and the second ridge peaks.

7.5 Skeletons and Regions

7.5.1 Implementation Details

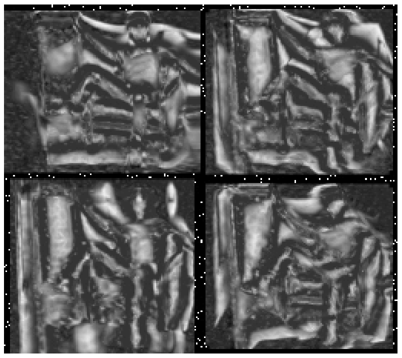
The color ridge detector and the modified 3-stage region-based reference frame algorithm described in the previous sections were implemented on a Connection Machine.

During the first stage, we perform directional color ridge detection at multiple scales to compute inertia surfaces and local width maps at a few different orientations. In our implementation, we detect ridges for widths of 20 to 150 pixels at steps of 2. This is done at 4 different orientations, namely the N-S, E-W, NE-SW and NW-SE directions, so as to produce inertia surfaces and local width maps for the 8 closest neighbour directions.

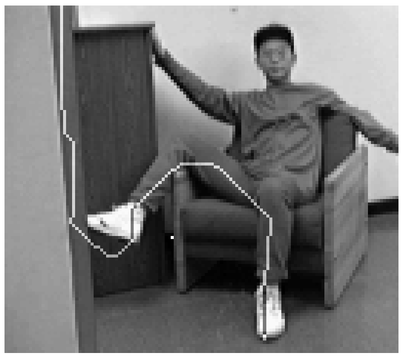
The second stage computes the saliency measure and direction of the most salient curve starting at each point in the image. Because we only compute inertia surfaces and local widths for the 8 cardinal directions, the most salient curve's direction at each point is also limited to one of the 8 closest neighbour directions in our current implementation. The number of network iterations we perform is set to the maximum dimension of the image measured in pixels.

In Stage 3, we find skeletons from salient curves by again performing multiple scale color ridge detection along each curve. We use ridge scales ranging from a width of 20 pixels to the full length of the curve measured in pixels. Currently, we truncate a curve manually by eyeballing its ridge detector response for the truncation point – the lowest output location between the first 2 *significant* ridge peaks. With suitable thresholds for determining *significance*, it is possible to design a simple algorithm for taking over the manual curve truncation task. Our skeleton extraction process works serially beginning with the most salient image curve, that is it find a region skeleton from the most salient curve first before searching for the next most salient curve to find the next region skeleton.

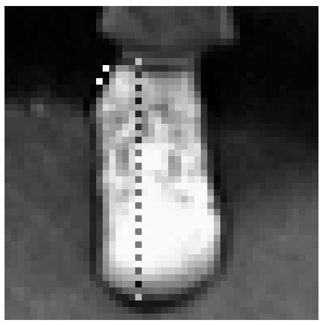
(a)



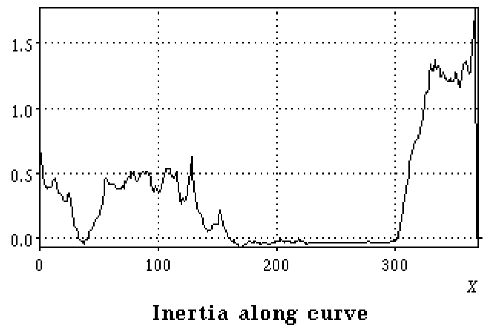
(b)



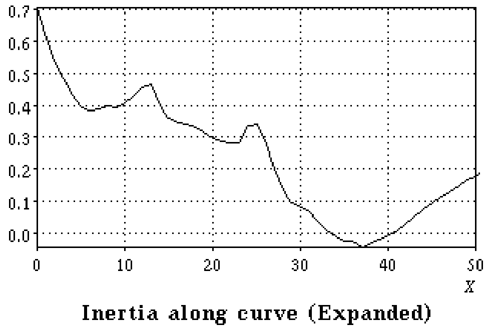
(c)



(d)



(e)



(f)

Figure 7.15: How region skeletons are computed. (a) The color image. (b) Inertia surfaces for the image computed using directional color ridge detection. (c) A salient curve the saliency network finds from the inertia surfaces. (d) The resulting skeleton for the first region. (e) The result of color ridge detection on the curve's color profile. We truncate the curve at the point where the curve first crosses a region boundary, which corresponds to the lowest output point between the first 2 significant ridge peaks. (f) The ridge detection profile expanded to focus on the first ridge, which corresponds to the skeleton for the shoe.

7.5.2 Results

We have successfully tested our directional color ridge finder and the modified region-based reference frame algorithm on a few real images. Figures 7.16 and 7.17 summarize the results we obtained from two test images. The first test case (Figure 7.16) is an image of a blurred object in a uniformly colored background. Both intensity and color edge detection operations produce very poor edge maps for this image at most scales, so we can expect traditional edge-based reference frame algorithms to produce poor skeletons for the image too. The second example (Figure 7.17) is a natural occurring image of the author in a moderately complex indoor environment. It demonstrates the system's ability to find good reference frames for regions of different shapes and sizes.

The figures also display some of the skeletons we find in the two images together with their associated color regions. Since each skeletal curve generally occupies a good spatial sample of points from within its enclosing region, its color distribution can be treated as a "representative" color sample for its enclosing region. We can therefore make use of color skeletons as intermediate color region descriptors for color region finding.

Our color region finding algorithm grows skeletons into complete image regions by working with the color distribution of each skeleton within a traditional *Split and Merge* segmentation framework [Ballard and Brown 82] [Horowitz and Pavlidis 74]. The algorithm first computes the *average color* (see Chapter 4) and the *angular color standard deviation*, σ_a (see Chapter 6), for each region skeleton. It then recursively examines pixels bordering the skeleton and merges them with the skeleton to form a larger region if their colors are close enough to the skeleton's *average color*. Because the regions in both test images are indeed relatively uniform in color, we can use a simple closeness measure for merging pixels with skeletons that depends only on the skeleton's *average color* and its σ_a value. In both test cases, we allow merging to occur if the pixel's color and the skeleton's *average color* are within $2\sigma_a$ of each other. The segmentation results for most regions still remain unchanged even after increasing the merging threshold from $2\sigma_a$ to $3\sigma_a$.

7.6 Summary and Extensions

This chapter introduces a new notion of color uniformity and presents a technique for detecting uniform color features directly from color image readings. The color uniformity modeling scheme seeks an analogous relationship between uniform color regions and scalar 1 dimensional ridge features, so that traditional scalar ridge detection methods can be employed to detect and locate uniform color regions in the image. In a scene whose physical surface entities are actually uniformly colored or just slightly varying in color, the modeling scheme can transform a reasonably well centered color cross sectional profile of some image region into a scalar 1D ridge profile. The ridge profile's high central plateau corresponds to the portion of the color cross section that lies within the region itself. Detecting and locating feature centers for these color ridges thus amounts to highlighting

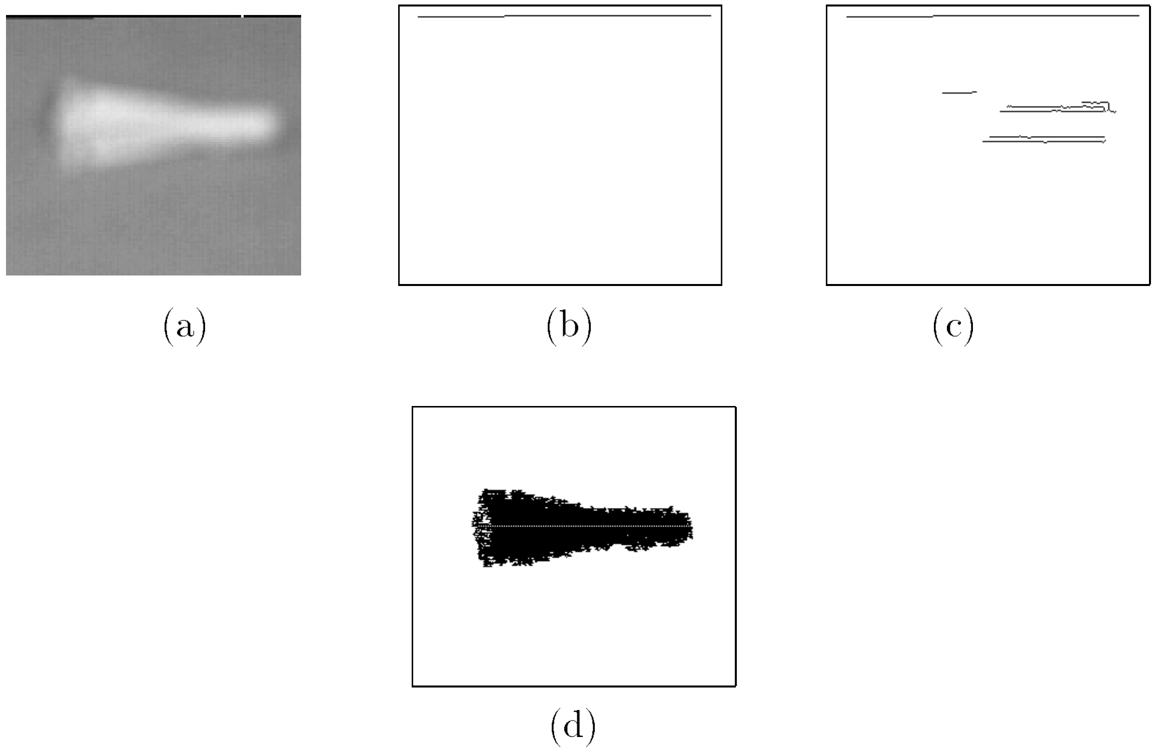


Figure 7.16: (a) An image of a blurred object in a uniformly colored background. (b) Color boundaries using a mask size of $\sigma = 2$. (c) Intensity boundaries using a mask size of $\sigma = 2$. (d) Skeleton and color region corresponding to the blurred object.

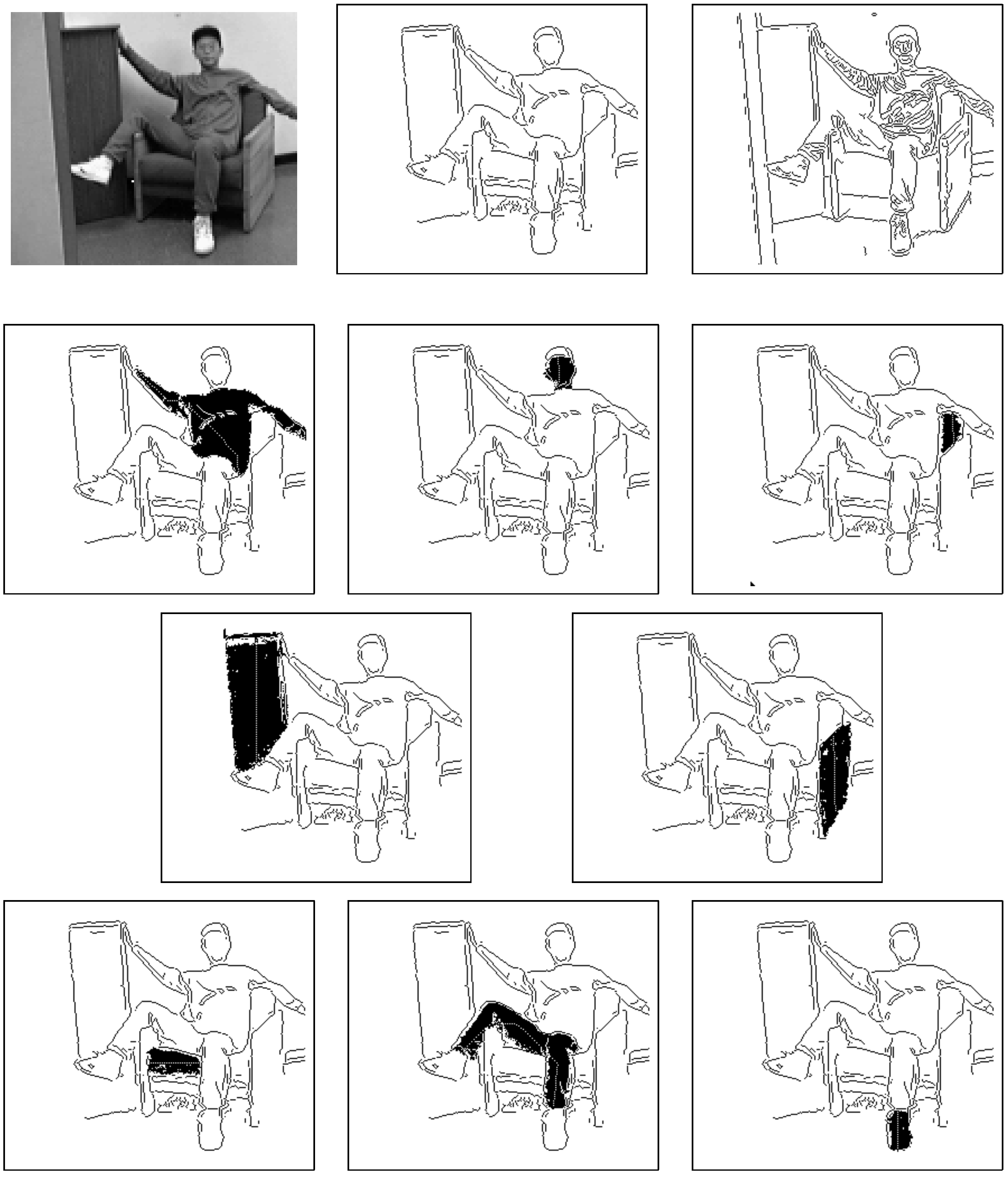


Figure 7.17: Top Row: An image of the author in a moderately complex indoor environment, color boundaries and intensity boundaries. Other Rows: Some color skeletons and regions found by our algorithm.

shape centers and local axes of symmetry for uniform color regions in the image.

By successfully linking together continuous stretches of region centers and region local axes of symmetry to form skeletons, we have in fact demonstrated an independent approach that computes the dual problem of edge linking in boundary detection. Here, the results we get are salient stretches of line features that lie within and describe the shapes of uniform color regions in the image. The advantage of having a skeleton finding approach independent of boundary detection is obvious, as vision systems can now infer the overall shapes of image regions even when their region boundaries are poorly formed. Because region skeletons occupy a good spatial sample of points within their enclosing image regions, we see that they can serve as helpful intermediate region descriptors and ideal start points for region finding operations.

7.6.1 Ridges as a General Uniformity Notion

Section 7.2 extends the notion of scalar ridges into color, a vector domain, and uses the *color ridge* concept to model color uniformity. In Section 7.3, we then generalize the traditional scalar ridge detection process to detect color ridge features. We believe that the ridge concept presented in this chapter is in fact a more universal descriptive notion that can be extended to model uniformity in other forms of data as well, such as texture uniformity. All that one needs for making the ridge extension into another cue is to define an appropriate *quantitative* similarity or difference measure for the cue. In the case of texture, this quantitative measure can be a *Maximum Frequency Difference* (MFD) statistic that Voorhees [Voorhees 87] used for texture discrimination. A *texture ridge* can then be described as a 1 dimensional texture profile with a central block of highly similar texture points, suitable for modeling uniform texture regions in 1D. To detect and locate texture ridge centers in images, we can use an approach analogous to the color ridge detection process that first converts texture profiles into scalar ridge profiles using texture similarity measurements before performing scalar ridge detection.

7.6.2 Skeletons and Region Growing

The previous section briefly describes a practical application of color reference frames in region finding. Currently, our skeleton based region finder only makes use of skeletons for their representative color values within a *Split and Merge* region growing framework. A natural extension to the existing system would be to make use of the geometric properties that skeletons poses as well. We have seen at the beginning of this chapter that skeletons are good shape descriptors for their enclosing regions because they run through region centers and local axes of symmetry. Since color ridge detection also gives us width estimates of regions centered about their skeletons, it makes sense to use the shape of skeletons and their local width estimates as an alternative means of geometrically controlling the extent of region growing. Future research in this area should try to integrate the geometric constraints of regions captured by their skeletons together with our current skeleton based region growing technique to produce more robust color region finders.

Bibliography

[Bajcsy Lee and Leonardis 89] Ruzena Bajcsy, Sang Wook Lee, and Ales Leonardis. Color image segmentation with detection of highlights and inter-reflections. To appear in International Journal on Computer Vision, 1989.

[Ballard and Brown 82] Dana H. Ballard and Christopher M. Brown. *Computer Vision*, chapter 5, pages 155–160. Prentice-Hall, 1982.

[Berter Poggio and Torre 86] M. Berter, Tomaso Poggio, and V. Torre. Ill-posed problems in early vision. Technical Report AIM–924, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1987.

[Beymer SM] David J. Beymer. Junctions: Their detection and use for grouping in images. Master's thesis, Massachusetts Institute of Technology, 1989.

[Blake and Zisserman 87] Andrew Blake and Andrew Zisserman. *Visual Reconstruction.* The MIT Press, Cambridge, MA, 1987.

[Blum 67] H. Blum. A transformation for extracting new descriptors of shape. In Walthen Dunn, editor, *Models for the perception of speech and visual form*, pages 362–380. MIT Press, Cambridge, MA., 1967.

[Blum and Nagel 78] H. Blum and R. N. Nagel. Shape descriptors using weighted symmetric axis features. *Pattern Recognition*, 10:167–180, 1978.

[Bracho and Sanderson 85] Rafael Bracho and Arthur C. Sanderson. Segmentation of images based on intensity information. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 341–347, 1985.

[Brice and Fennema 70] C. Brice and C. Fennema. Scene analysis using regions. *Artificial Intelligence*, 1(3):205–226, 1970.

[Cordella Arcelli and Levialdi 81] L. Cordella C. Arcelli and S. Levialdi. From local maxima to connected skeletons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(PAMI-3):134–143, 1981.

[Canny and Donald 87] J. Canny and B. Donald. Simplified voronoi diagrams. Technical Report AIM–957, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, April 1987.

[Canny 83] John F. Canny. Finding lines and edges in images. Technical Report TM-720, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1983.

[Chow and Kaneko 72] C. Chow and T. Kaneko. Automatic boundary detection of the left ventricle from cineangiograms. *Computers and Biomedical Research*, 5:388–410, August 1972.

[Clemens 91] D. Clemens. *Region-based feature interpretation for recognizing 3D models in 2D images*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1991.

[Connell and Brady 87] J. H. Connell and M. Brady. Generating and generalizing models of visual objects. *Artificial Intelligence*, 31:159–183, 1987.

[DeGroot 86] Morris H. DeGroot. *Probability and Statistics, Second Edition.* Addison-Wesley Publishing Company, New York, 1986.

[Duda and Hart 73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*, chapter 6, pages 211–257. John Wiley and Sons Inc., 1973.

[Feldman and Yakimovsky 74] J. Feldman and Y. Yakimovsky. Decision theory in artificial intelligence: I. a sementics-based region analyzer. *Artificial Intelligence*, 5(4):349–371, 1974.

[Frieden 83] B. Frieden. *Probability, Statistical Optics and Data Testing.* Springer-Verlag, New York, 1983.

[Gamble and Poggio 87] Edward Gamble and Tomaso Poggio. Visual integration and detection of discontinuities: The key role of intensity edges. A.I. Memo No. 970, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Ma., October 1987.

[Geiger and Girosi 89] Davi Geiger and Federico Girosi. Parallel and deterministic algorithms for mrfs: surface reconstruction and integration. A.I. Memo No. 1114, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Ma., June 1989.

[Geman and Geman 84] Stuart Geman and Don Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[Gonzalez and Wintz 87] Rafael C. Gonzalez and Paul Wintz. *Digital Image Processing (Second Edition).* Addison Wesley, Reading, MA, 1987.

[Grimson 80] W. Eric L. Grimson. *Computing shape using a theory of human stereo vision.* PhD thesis, Massachusetts Institute of Technology, 1980.

[Grimson 81] W. Eric L. Grimson. *From Images to Surfaces.* MIT Press, Cambridge, Mass., 1981.

[Grimson and Lozano-Perez 85a] W. Eric L. Grimson and Tomas Lozano-Perez. Model-based recognition and localization from sparse range data. In A. Rosenfeld, editor, *Techniques for 3-D Machine Perception.* North-Holland, Amsterdam, 1985.

[Grimson and Lozano-Perez 85b] W. Eric L. Grimson and Tomas Lozano-Perez. Recognition and localization of overlapping parts from sparse data. In T. Kanade, editor, *Three-Dimensional Vision Systems.* Kluwer Academic Publishers, Amsterdam, 1985.

[Haddon and Boyce 90] J. F. Haddon and J. F¿ Boyce. Image segmentation by unifying region and boundary information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):929–948, 1990.

[Hanson and Riseman 78] A. R. Hanson and E. M. Riseman. Segmentation of natural scenes. *CVS*, 1978.

[Hildreth 83] Ellen C. Hildreth. *The Measurement of Visual Motion.* MIT Press, Cambridge, Mass., 1983.

[Hildreth 84] Ellen C. Hildreth. Edge detection. Technical Report AIM–858, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Ma., 1985.

[Horn 74] B.K.P. Horn. Determining lightness from an image. *Computer Graphics and Image Processing*, 3(1):277–299, December 1974.

[Horowitz and Pavlidis 74] S. L. Horowitz and T. Pavlidis. Picture segmentation by a direct split and merge procedure. In *Proceedings IJCPR*, pages 424–433, August 1974.

[Hunter 75] R. S. Hunter. *The Measurement of Appearance.* John Wiley and Sons, New York, 1975.

[Hurlbert 89] Anya C. Hurlbert. *The Computation of Color.* PhD thesis, Massachusetts Institute of Technology, 1989.

[Hurlbert and Poggio 88] Anya C. Hurlbert and Tomaso Poggio. A network for image segmentation using color. Technical report, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Ma., 1988.

[Johnston and Park 66] R. M. Johnston and R. E. Park. Color and appearance. *Color Engineering*, 4(6):14, 1966.

[Judd and Wyszecki 75] D. B. Judd and G. Wyszecki. *Color in Business, Science and Industry (Third Edition).* Wiley, New York, 1975.

[Klinker 88] Gudrun J. Klinker. *A Physical Approach to Color Image Understanding.* PhD thesis, Carnegie Mellon University, 1988.

[Klinker Shafer and Kanade 88a] Gudrun J. Klinker, Steven A. Shafer, and Takeo Kanade. Image segmentation and reflection analysis through color. In *Proceedings Image Understanding Workshop*, pages 838–853, Cambridge, MA, April 1988.

[Klinker Shafer and Kanade 88b] Gudrun J. Klinker, Steven A. Shafer, and Takeo Kanade. The measurement of highlight in color images. *International Journal on Computer Vision*, 2:7–32, 1988.

[Land 59] Edwin H. Land. Colour vision and the natural image. *Proceedings of the National Academy of Sciences*, 45:115–129, 1959.

[Lee 86] Hsien C. Lee. Method for computing scene-illuminant chromaticity from specular highlight. *Journal of the Optical Society of America*, 3:1694–1699, 1986.

[Marr and Nishihara 78] D. Marr and H. Nishihara. Representation and recognition of the spatial organization of three dimensional structure. In *Proceedings of the Royal Society of London B*, volume 200, pages 269–294, 1978.

[Marr and Poggio 79] D. Marr and T. Poggio. A theory of human stereo vision. In *Proceedings of the Royal Society of London B*, volume 204, pages 321–328, 1979.

[Marroquin Mitter and Poggio 85] J. Marroquin, S. Mitter, and Tomaso Poggio. Probabilistic solution of ill-posed problems in computational vision. In *Proceedings Image Understanding Workshop*, pages 293–309, Miami Beach, FL, December 1985.

[Marroquin 85] Jose Marroquin. *Probabilistic solution of inverse problems.* PhD thesis, Massachusetts Institute of Technology, 1985.

[Milgram and Kahl 79] D. L. Milgram and D. J. Kahl. Recursive region extraction. *Computer Graphics and Image Processing*, 9:82–88, 1979.

[Nevetia and Binford 77] R. Nevatia and T. Binford. Description and recognition of curved objects. *Artificial Intelligence*, 8:77–98, 1977.

[Ohlander 76] R. Ohlander. *Analysis of Natural Scenes.* PhD thesis, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[Blake 85] T. Ottoson and S. Zeki. *Central and Peripheral Mechanisms of Colour Vision*, pages 45–49. Macmillan, 1985.

[Pavlidis 78] T. Pavlidis. A review of algorithms for shape analysis. *Computer Graphics and Image Processing*, 7:243–258, 1978.

[Poggio et. al. 85] Tomaso Poggio and the staff. Mit progress in understanding images. In *Proceedings Image Understanding Workshop*, pages 25–39, December 1985.

[Poggio et. al. 87] Tomaso Poggio and the staff. Mit progress in understanding images. In *Proceedings Image Understanding Workshop*, pages 41–54, Los Angeles, CA, February 1987.

[Poggio Voorhees and Yuille 84] Tomaso Poggio, Harry Voorhees, and Alan L. Yuille. Regularizing edge detection. Technical Report AIM–776, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1984.

[Prewitt and Mendelsohn 66] J. Prewitt and M. Mendelsohn. The analysis of cell images. *Ann. N. Y. Acad. Sci*, 128:1035–1053, 1966.

[Russell Brooks and Binford 79] G. Russell R. Brooks and T. Binford. The acronym model based vision system. In *Proceedings IJCAI*, pages 105–113, 1979.

[Rosenfeld and Kak 76] A. Rosenfeld and A. Kak. *Digital Picture Processing.* Academic Press, New York, 1976.

[Rubin and Richards 81] John M. Rubin and W. A. Richards. Colour vision and image intensities: When are changes material. Technical Report AIM–631, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1981.

[Rubin and Richards 84] John M. Rubin and W. A. Richards. Colour vision: Representing material categories. Technical Report AIM–764, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1984.

[Sha'ashua and Ullman 88] A. Sha'ashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *Proceedings of the International Conference on Computer Vision*, pages 321–327, Dec 1988.

[Shafer and Kanade 82] S. A. Shafer and T. Kanade. Recursive region segmentation by analysis of histograms. In *Computer Vision, Graphics, and Image Processing*, pages 22:145–176, 1982.

[Subirana-Vilanova 90a] J. Brian Subirana-Vilanova. Curved inertia frames and the skeleton sketch: Finding salient frames of reference. In *Proceedings of the International Conference on Computer Vision*, pages 702–708, Osaka, Japan, 1990.

[Subirana-Vilanova 90b] J. Brian Subirana-Vilanova. The skeleton sketch: Finding salient frames of reference. In *Proceedings Image Understanding Workshop*, pages 614–622, Pittsburgh, Pennsylvania, 1990. Morgan Kaufmann Publishers.

[Tamura 78] H. Tamura. A comparison of thinning algorithms from digital geometry viewpoint. In *Proceedings Int. Conf. on Pattern Recognition*, pages 715–719, Kyoto, Japan, 1978.

[Ullman 79] Shimon Ullman. *The Interpretation of Visual Motion.* MIT Press, Cambridge, MA, 1979.

[Voorhees and Poggio 87] Harry Voorhees and Tomaso Poggio. Detecting textons and texture boundaries in natural images. In *Proceedings of the International Conference on Computer Vision*, pages 250–258, London, England, June 1987.

[Voorhees 87] Harry L. Voorhees. Finding texture boundaries in images. Master's thesis, Massachusetts Institute of Technology, 1987.

[Vos and Walraven 70] J. J. Vos and P. L. Walraven. On the derivation of the foveal receptor primaries. *Vision Research*, 11:799–818, 1970.

[Wolff 89] L. B. Wolff. Using polarization to separate reflection components. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 363–369, June 1989.

[Wright 89] W. A. Wright. A markov random field approach to data fusion and colour segmentation. *Image and Vision Computing*, 7:144–150, 1989.

[Young 86] Richard A. Young. Principle-component analysis of macaque lateral geniculate nucleus chromatic data. *Journal of the Optical Society of America*, 3:1735, October 1986.

[Yuille 84] Alan L. Yuille. A method of computing spectral reflectance. Technical Report AIM–752, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1984.