EARLY VISION USING DISTRIBUTIONS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Mark A. Ruzon
April 2000

Abstract

For over thirty years computer vision researchers have been proposing methods for "early vision" tasks such as detecting edges and corners. One key assumption shared by most previous methods is that image neighborhoods are constant in color or intensity, with deviations modeled as noise. Due to computational considerations that encourage the use of small neighborhoods where this assumption holds, these methods remain popular.

This research models a neighborhood as a distribution of colors. Our goal is to show that the increase in accuracy of this representation translates into higher-quality results for early vision tasks on difficult, natural images, especially as neighborhood size increases. We emphasize large neighborhoods because small ones often do not contain enough information. We emphasize color because it subsumes greyscale as an image range and because it limits the number of valid models we should consider; using only greyscale images allows assumptions that do not hold for color.

We start by developing the compass operator, a color edge detector that computes the orientation of the diameter of a circle that maximizes the distance between two color distributions. This distance is computed by using the earth mover's distance, which finds the minimal amount of work needed to transform one distribution into another.

We continue with color corner detection, a generalization of color edge detection in which the two sides no longer have the same size. Extracting corners is more involved than extracting edges because multiple responses to the same corner are not allowed. We show that corners are dependent on edge evidence and, therefore, an edge model is required to confirm the existence of a corner.

Finally, we extend blue screen matting, a technique that extracts an object from a constant color background, to backgrounds that are almost arbitrary. Given coarse knowledge of the boundary between two objects, we compute the color distributions on either side of a potential boundary pixel and estimate alpha, the proportion in which a color from each side mixed to form that pixel's color. As a result, a user can more easily move objects from one image to another while maintaining photorealism.

Acknowledgments

I really would like to thank everyone I have known while at Stanford. We are all very much a product of our interactions with each other, even those that are not so positive. Since I can't thank everyone by name, I will limit this list to my parents, Jim and Dorothy, who allowed me to experience this wonderful thing called life; my fellow students from the vision group, Drago Angelov, Anat Caspi, Aris Gionis, Burak Gokturk, Angela Hodge, David Hoffman, Mike Lin (who provided his implementation of Intelligent Scissors to me), Itay Lotan, Mark Perez, Xiaofeng Ren, Chuck Richards, Brian Rogoff, Donald Tanguay, and John Zhang; Reading Committee members David Heeger and Robert Gray; and Oral Committee members Tom Binford and Glenn Healey. Professor Healey came all the way from UC-Irvine to support me at my defense on 10 December 1999, for which I am especially grateful.

First and foremost on the list of people whose specific contributions I am compelled to mention is my advisor, Carlo Tomasi. My time with him has been one of great freedom: freedom to choose topics that interested me, freedom from the bureaucracy of funding proposals and status reports, and freedom to learn how to do research and make mistakes. He is more generous with his time than any professor I have ever heard of, and I owe him a great deal more than he would ever be willing to acknowledge.

I also thank my long-time officemate, Yossi Rubner. It was he who introduced the earth mover's distance to the group, an idea that sparked much of the research presented here. Being significantly older, he had a lot of wisdom and gave me much advice, some of which I actually took. As for the rest: *Ma pit'om!*

Third on my list to thank is Scott Cohen, who was the first real friend I made at

Stanford. It was a good choice on my part, as I often found myself walking through the corridors of Cedar Hall or the Gates Building to get his help on a technical problem or just vent about the graduate school process. The year since he graduated has seemed longer in some ways.

Finally, I thank my wonderful wife Lesley, to whom I dedicate this work, for all the love she has given me. I cannot say that she helped quicken the time it took to get to this point, but she has certainly made the time pass much more happily.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The goal of computer vision since its inception in the 1960's has been to demonstrate understanding of human vision by constructing systems that can perform visual tasks of the same generality and robustness that humans are capable of. The year 2000 has found only minimal progress made toward this goal; a number of reliable vision systems do exist in industry today, but the range of tasks they perform, such as defect inspection, are very narrow in scope, and they work only in carefully controlled environments. Consumer-level applications, for example visual interfaces, are on the horizon but have not yet made an impact.

The principal difficulty in building robust, general-purpose vision systems is one of representation: what information about the structure and visual appearance of objects is encoded, and how? At the most abstract level we are consciously aware of the names we associate with objects, we can understand the many relationships between objects, and we can form "mental pictures" of objects. At the most concrete level we know a great deal about the types of local stimuli that prompt responses from cells in the retina and parts of the visual cortex.

However, between the two lies a gap that neither scientists nor engineers have yet been able to cross. It is generally believed that one or more intermediate representations exist between what we can directly measure about our physiology and what our conscious minds know. The number of representations, their descriptions, and their relationships to one another are not known; a natural place to look is at the boundaries between objects since they are presumably detectable without knowledge of the objects themselves.

L.G. Roberts, generally considered to have authored the first computer vision paper[1] [81], created an intermediate representation of straight, connected line segments. A scanned image of an object formed the lowest-level input. By applying a derivative filter to the image and connecting the resulting short edges into longer ones, the image was reduced to a line drawing. After finding the corners, convex polygons were extracted, and polygons sharing an edge were compared against a model database. After choosing a model that met both topological and quantitative constraints, the boundary points were used to find the transformation between the frames of reference of the object and the model. In this way, new views of an object could be synthesized. The framework even allowed for an object consisting of multiple parts to be decomposed, with each part being recognized separately.

However, soon afterwards it became clear that, although the approach was (and still is) regarded as reasonable, the real challenge lay in relaxing the simplifying assumptions that were made. In the natural world, backgrounds are not constant, surfaces contain markings, illumination is not diffuse and uniform, and objects do not consist of polyhedra joined together. A new field of study quickly formed.

Much effort has been spent since then trying to understand various aspects of static (single image) and dynamic (image sequence) visual perception, and while a great many milestones have been reached, we are still uncovering the complexity of the problem. Vision research since then can be broadly divided into model-based and image-based approaches.

Model-based approaches to computer vision begin with principles developed in fields as disparate as neurology, physics, and statistics, from which models of perception are formed, implemented, and tested. Historically, most research has followed this path, from purely bottom-up algorithms that extract edges and corners to top-down

[1]Roberts' paper was published in 1965, but *computer vision* became an accepted term only around 1983, before which these techniques were grouped under *image processing*. The difference, at least according to the computer vision community, is that computer vision algorithms output information about the state of the world, while image processing algorithms output images. Examples of the latter include enhancement, restoration, and compression/decompression. There is still much overlap between the two, though.

models of object representation and recognition under different viewing conditions to domain-specific applications such as medical imaging, head tracking, and gesture recognition for sign language.

The image-based approach is more recent and rests on the idea that the principles underlying vision are too complex or too interrelated to be extracted singly. Instead, models are learned directly from image data. For example, image-based rendering uses samples from images to synthesize novel views of objects without first identifying them and estimating their poses. These views appear much more natural than could be done by specifying models manually.

This research is model-based, even though image-based approaches are gaining in popularity. Its primary goal is to answer the question, "What is the best way to represent image neighborhoods and the boundaries between them?" Our answer for representing a neighborhood is a distribution of pixel values, in which the frequency of the dominant colors or intensities is recorded. We intend to show that this representation leads to more accurate results for detecting edges and corners, and in situations more difficult than have been attempted previously. We also consider boundaries, particularly of natural objects like hair, which are not well modeled by edges or corners, and show an image-based rendering application that semi-automatically extracts an object from its background.

The next two sections are devoted to a deeper understanding of the challenges of early vision and our strategy for dealing with them.

1.1 Why is Early Vision a Difficult Problem?

The most interesting challenge in explaining computer vision to the woman or man on the street is convincing them that duplicating human vision is difficult in the first place. Long-term memory does not develop until after a person's visual system is fully formed, so there is no experiential frame of reference: we have always been able to see.

So why is it difficult to build systems that perform visual tasks? We restrict the scope of the question to the scope of this research, which is *early vision*. Early vision refers to the detection of features common to most objects in the world, such as edges, corners, shapes, relative or absolute depth, visual texture (the spatially repetitive patterns of an object's appearance), or motion. These features can ideally be detected and described independently of the objects that give rise to them.

But what is the goal of early vision? As we said before, the problem is one of constructing intermediate representations. These representations should satisfy two requirements: they should be smaller than the original representation in terms of the number of bits needed to encode them, and they should preserve the "content" of the image.

Consider the image of a statue in the upper left of Figure 1.1. The image to its right is an artist's rendering, which for hundreds of years was the only intermediate representation of the world that could be effectively and instantly conveyed to other people. Note that the drawing contains the salient features of the image, namely that it is a statue on a pedestal against a background consisting mainly of a building. There are a significant number of differences, however, even though the artist had the original image available throughout the drawing process.

The most evident observation is that the statue is neither the same shape nor size as the original, and it is not in the same location as it would be if it were merely traced. Because the statue is the focus of the image, it has been enlarged. Also, a number of details have actually been added to the statue that are not apparent in the original image. High-level knowledge has been used to incorporate features that the artist believed must be in the world.

Just as important is the fact that many details were left behind. The two main textures, bricks and bush leaves, would be exceedingly tedious to render accurately. Also, the car on the left has disappeared, as well as the shoots of ivy that are crawling up the wall. These details are not central to our understanding and would perhaps distract the viewer from the statue.

We cannot hope to generate such an output automatically due to the preponderance of high-level decisions that went into the drawing. A more realistic intermediate representation for computer vision is a *segmentation*, or a partitioning of an image into regions, each of which corresponds to a different object. The bottom of Figure 1.1

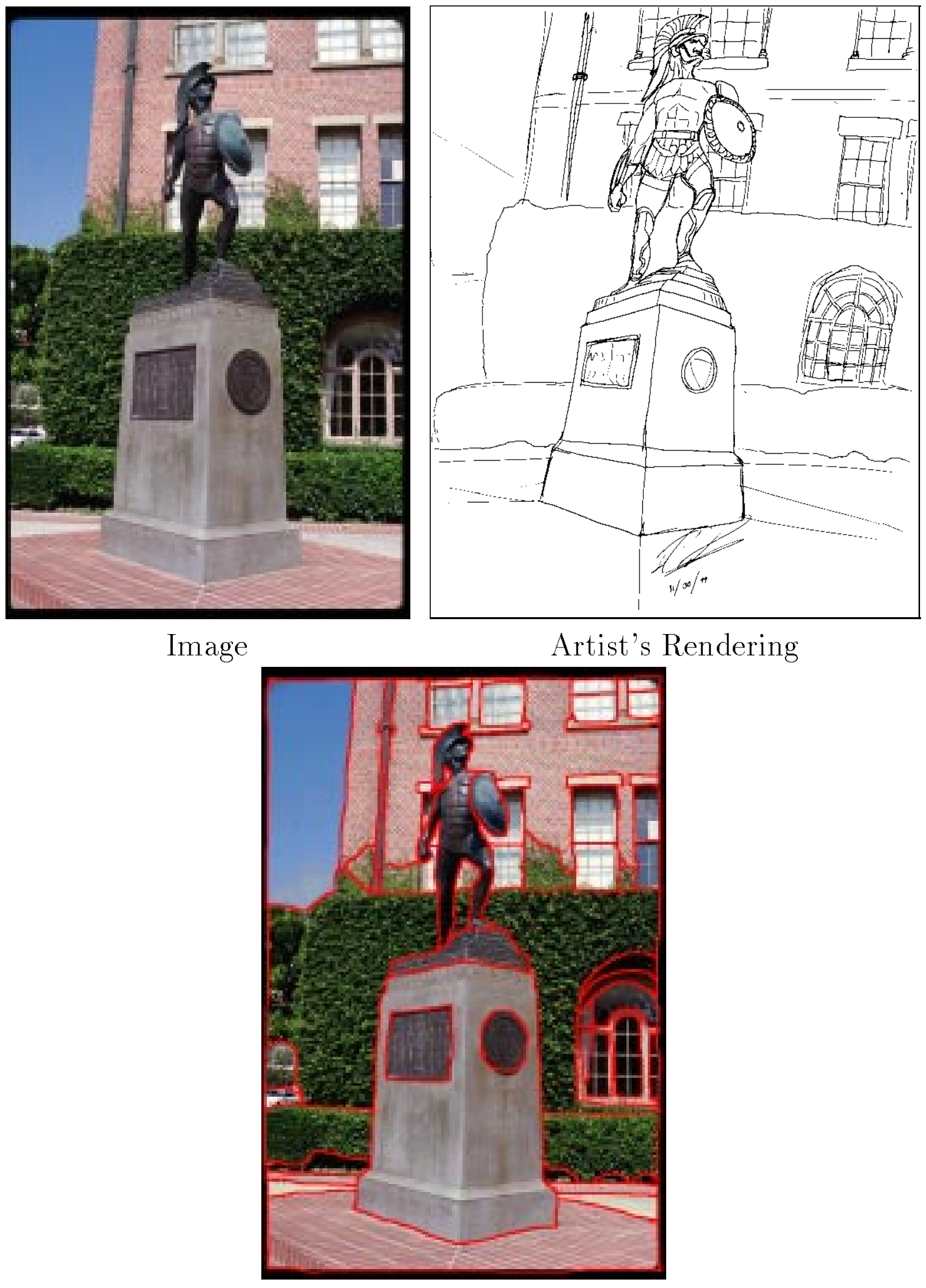


Figure 1.1: Early vision strives to produce intermediate representations such as a line drawing or a segmentation. (Drawing credit: Razael Cortés)

shows a segmentation hand-painted onto the original image.

The segmentation has the advantage of corresponding spatially to the original image, and it also conveys the essence of the image while reducing the amount of memory needed to store it. The proposed segmentation for the statue image is hardly definitive, though. Two possibly important boundaries are missing: the boundary between the ivy and the trees in the background on the left side, and the long, vertical line where the two visible faces of the pedestal meet. The first boundary could hardly be said to exist except for a slight change in texture from ivy to tree and because the car, obviously further away than the statue, leads us to believe that what is above it must also be far away. The second boundary is easier to detect near the bottom of the pedestal than near the top. Adding this boundary to the segmentation makes it easier to recover the 3-D shape of the pedestal, but it might not be necessary to do so to recognize that it is a pedestal in the first place.

Also, one could argue that certain painted boundaries are superfluous. Beneath the hedge are a few regions in shadow. These regions are certainly different in appearance from the hedge, but they do not correspond to different objects, at least in the sense we are comfortable with, and they might interfere with our recognition of the hedge if we proceed by analyzing the shape of the region. We are thus forced to conclude that even segmentation is too unwieldy a goal to pursue.

Furthermore, the state of the art in computer vision does not allow us to produce even one segmentation out of the set of potential segmentations for an image. We spend the rest of this section explaining why.

First, note that the boundaries in Figure 1.1 have a wide range of attributes. Some are short, while others are long; some are sharp, while others are a little blurry. These attributes involve the notion of *scale*, or the size of the neighborhood to be considered. Long, sharp boundaries may be found at a wide range of scales, while short edges can be found only at small scales, and blurry edges require larger scales to be resolved properly. A boundary detection algorithm must be used at multiple scales, but it is not clear how conflicting information from different scales should be combined. We have already seen that some minute details in an image are very important, while others are not, but there is no general way to distinguish between the two.

Solving the scale problem is beyond the scope of this work, but we mention the two prevailing approaches for completeness. One is to consider the set of all edges as a 3-D object, where the scale parameter varies to form the third dimension. An edge now exists over a range of scales, which provides a more complete description. The competing method is local scale control, where image statistics are used to select one scale parameter value at each point, usually the minimum value that produces stable output. Neither method solves the above problems, though.

Another problem concerns the inability of local operators, so named because they analyze only one neighborhood at a time, to produce results that are globally optimal. For instance, if the contrast between two sides of a boundary dips below a pre-specified value, no boundary will be detected, resulting in a gap. The output of local operators is often post-processed to fix the most glaring mistakes, but detecting where all mistakes were made is equivalent to knowing the right answer in the first place.

The natural response to this problem is to make global assumptions that guide the work of a local operator. Unfortunately, even very general assumptions do not hold in all cases. Imagine an algorithm designed to find the boundary of a person's head and shoulders by finding edges and then matching them against a template[2]. Figure 1.2 shows two potential inputs to this algorithm. The relevant edges in the first image are found easily, and any anomalies will be corrected by applying the template. The second image poses problems for the algorithm because the boundary between the woman's hair and the background is not well-defined. Many edges do not match the template, and vital edges are likely to be missed.

In truth, we have discovered an even more general assumption that has been violated: the assumption that a boundary is always well represented by an edge. For the image of the woman, any boundary drawn will include forest in the foreground, or leave some of her hair in the background.

To sum up, vision algorithms always require assumptions, and no matter how general we believe them to be, there are always cases where they do not hold. Furthermore, there is no consensus as to the goal that is sought after, and so it is difficult to quantify the usefulness of any particular algorithm unless it is designed to perform

[2]Such an approach was actually used by a former Stanford student [43].

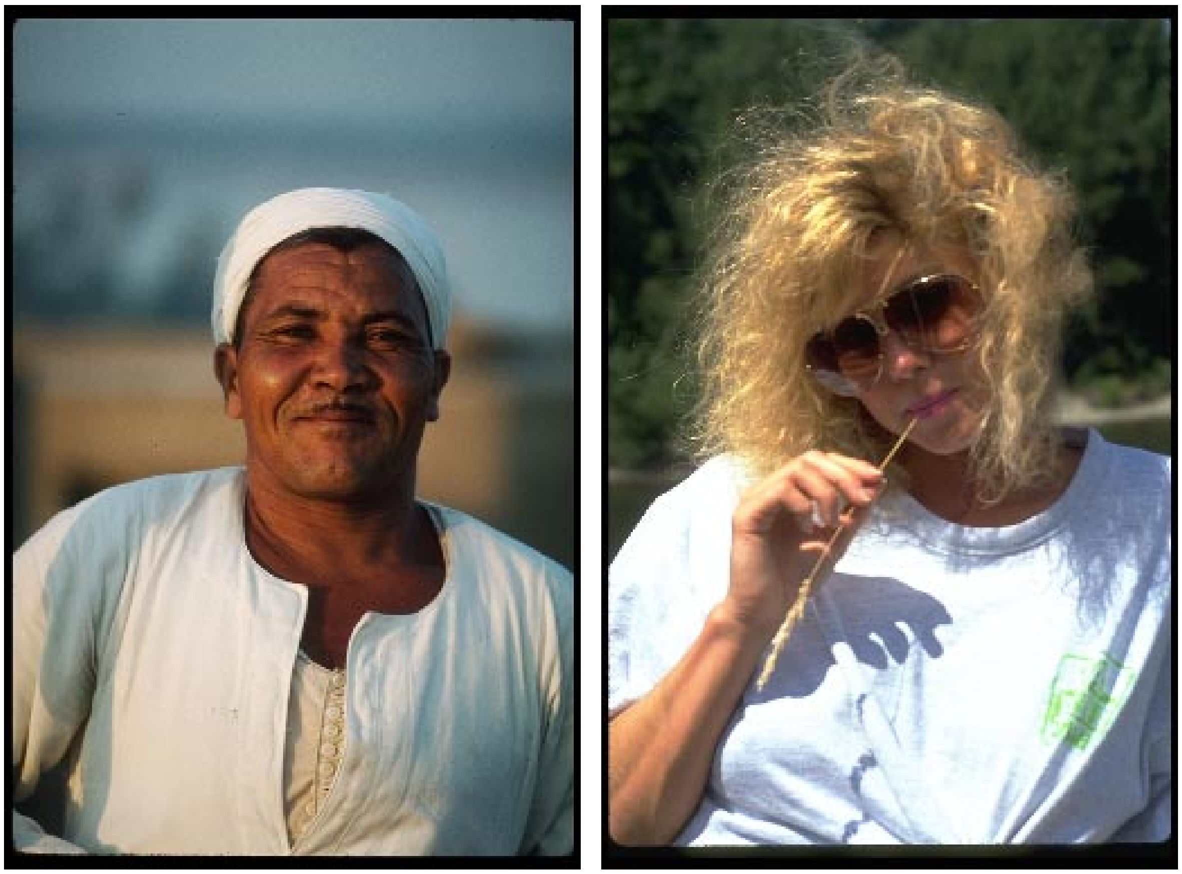


Figure 1.2: The two people in these images have very similar contours, but the man on the left is much easier to extract than the woman on the right because the boundary between her hair and the background is not well represented by a curve.

one of a range of tasks that is very narrow in scope, and only in carefully controlled environments.

1.2 What Lies Ahead

We have painted perhaps an overly grim picture of the state of computer vision today. We may not be able to make significant progress toward the twin goals of full generality and robustness, but if we are allowed to restrict the set of input images for our algorithms, the chances for success are much improved. Progress in the field can be measured by judging one set of assumptions as more general than previous sets, as long as it is accompanied by proof that a system can be built using these assumptions and produce adequate results.

This research is founded on a generalization of the representation of image neighborhoods for detecting boundaries. Most (but not all) detection schemes have relied on the assumption of constancy; in other words, the values of all the pixels on each side of a boundary are ideally the same, and any variations must be due to noise in the imaging process.

We instead represent image neighborhoods as distributions of pixel values. If a neighborhood contains 65% red pixels and 35% yellow pixels, for example, then we use this information in its original form rather than averaging the two to produce some other color. We show that the increase in accuracy that comes from using multiple values produces better edge and corner information than can be otherwise obtained. The resulting framework can handle an image range of any dimensionality, but we restrict our efforts almost exclusively to color. Color is unique in that it is three-dimensional, yet the retina can still perceive it directly.

Using distributions does not solve the problems discussed in the previous section, but we believe it makes solving the scale problem a little easier. The constancy assumption is far more likely to hold at small scales where few pixels are involved, but we require information from large scales as well. The use of distributions increases the quality of results at large scales where constancy cannot be assumed.

The remainder of this section defines the scope of the work pursued in this dissertation and how it is organized.

Chapter 2 outlines the principles upon which this research is based. In addition to examining distributions in more detail, we also consider the notion of flow as a means of comparing or interpolating between two distributions. We also explain our method for computing distances between individual colors, a step that must precede any computation between two color distributions.

In Chapter 3 we consider color edge detection, perhaps the most basic of all computer vision problems. After reviewing previous work, we motivate and develop the compass operator, which finds the orientation of an edge that maximizes the difference between two color distributions. This difference is the result of finding the flow between two distributions that minimizes the amount of work needed to transform one into the other. We also present a greyscale version that runs more quickly.

Corner detection, the subject of Chapter 4, has also been thoroughly studied in the literature. We present what we believe to be the first corner detector specifically for color images. The formulation of our detector is similar to that of the compass operator with the exception that the two color distributions are no longer the same size.

Finally, we present in Chapter 5 an application of distributions to images where boundaries are not well represented by edges or corners. In the film and video industries, the technique of blue screen matting is widely used to extract an object filmed against a constant-color background and to place it in another. We present an algorithm that extends this technique to a much wider set of backgrounds by estimating the color distributions outside a boundary region and measuring the "betweenness" of a color that falls between the two distributions.

Chapter 6 evaluates the contributions of this research and presents avenues for future exploration.

Chapter 2

Principles

If this research was to be compressed into a single sentence, it would be, "Use distributions to represent image neighborhoods." This chapter begins the main work of this thesis by justifying this choice. We begin by examining the improvement in accuracy that results from using distributions, followed by a discussion of flow, the mechanism by which distributions are to be compared to each other. Finally, we justify our use of color images and consider the problem of color distance.

2.1 The Use of Distributions

For the purpose of modeling image neighborhoods, the term *distribution* refers, in its most generic sense, to any representation that allows multiple pixel values along with their relative frequencies. The relative frequencies often sum to 1, in which case this definition is equivalent to that of a probability density function. Distributions have been used for many other purposes in computer vision, such as maintaining multiple hypotheses during motion tracking [41] and representing the color content of images for image retrieval [88], but not for extracting features like edges, corners, and junctions. Planar surfaces have been used to represent neighborhoods, but these are very restrictive types of distributions because they impose spatial constraints on pixel values.

The notion of distributions is more closely related to finding boundaries between different textures. Many different methods have been proposed for accumulating statistics about image texture, the spatially repetitive nature of a region. The biggest problem with texture is that it is a property of a neighborhood, not of a point like color is. Therefore, one is often forced to know the boundaries of a region containing texture before being able to describe it accurately. In addition, an effective texture representation cannot be developed without multi-scale processing, and such information is difficult to compare reliably at different scales. Finally, the very notion of texture is not applicable to all images, such as office scenes or faces in a group picture.

We start with the first and most fundamental choice: whether to use continuous or discrete representations of distributions. Because image values are already represented discretely by the camera or scanner, we use discrete distributions. In Chapter 5, however, we will also use continuous distributions for the purpose of constructing functions that are nonzero throughout color space.

The two main data structures for representing distributions are histograms and signatures. A *histogram* is a sequence of numbers $x_1, x_2, \ldots, x_n$ formed by partitioning a feature space into n bins and summing the weights associated with all data points that fall into each bin. Since each pixel value is treated the same no matter where in a bin it falls, the values are effectively quantized to the same pre-defined value.

A *signature* is more general; it consists of a set of ordered pairs $\{(x_1, \mathbf{v}_1), (x_2, \mathbf{v}_2), \ldots, (x_n, \mathbf{v}_n)\}$, where the $\mathbf{v}_i$'s are points within the feature space to which the weights x_i are assigned. A signature is equivalent to a probability mass function when the x_i's sum to 1. Signatures are superior to histograms because they can adapt to the data; they do not force an arbitrary partitioning of a feature space. On the other hand, they require more computation to create. See [87] for a more complete discussion and comparison.

The constancy assumption, discussed in Chapter 1, is easy to realize in a convolution mask that has a mean value of zero; such masks have often been used for edge and corner detection. It is also easy to show that convolution masks approximate two neighborhoods with one value each. If we consider one application of the mask, where each of the n pixels has a value $\mathbf{v}_i$ associated with a mask weight w_i in a Euclidean space, the operation can be decomposed as follows:

$$\begin{aligned}\|\sum_{i=1}^{n} w_i \mathbf{v}_i\| &= \|\sum_{w_i>0} w_i \mathbf{v}_i + \sum_{w_i<0} w_i \mathbf{v}_i\|, \\ &= \|\sum_{w_i>0} w_i \mathbf{v}_i - \sum_{w_i<0} (-w_i) \mathbf{v}_i\|.\end{aligned}$$

If we fix the sum of the positive weights to 1 and the sum of the negative weights to -1, convolution can be reformulated as computing the distance between the weighted means of each neighborhood[1].

Distributions create a more accurate representation of the image data, and Figure 2.1 illustrates the gain in accuracy. In this small experiment, square windows of different sizes have been randomly chosen from 200 randomly chosen images out of the Corel Image Database, a set of 20,000 natural images. The color and greyscale versions of these images were quantized with 1, 5, and 10 clusters using the binary split algorithm discussed in more detail in Chapter 3. The figure displays the average error in approximating each pixel by its nearest cluster representative.

We note the following trends: (1) quantizing a color image results in more error than quantizing its greyscale equivalent because there are more possible values, (2) quantization error increases logarithmically with the length of the side of the window, and (3) the slope of the error lines decreases with the number of clusters[2]. Of course, Figure 2.1 does not prove conclusively that distributions are superior representations. Feature extraction is ultimately a decision process, and if the error when using the mean is small enough to make the decision correctly, then no advantage is gained. We are not likely to do worse using distributions, however.

There is a second, more philosophical reason for choosing distributions. Because humans organize color in a three-dimensional space, our perception of color is fundamentally different from our perception of grey levels, which exist in a one-dimensional space. The weighted mean of a set of grey level values is "perceptually closed" in

[1]Taking the norm yields edge strength, but the signs of the differences are also important.

[2]Theoretically, the error for the 5- and 10-cluster cases should start at zero since the smallest neighborhoods considered have only 4 pixels. However, our implementation refrains from creating very small clusters; thus the error decreases with window size up to a point because we are able to create more clusters.

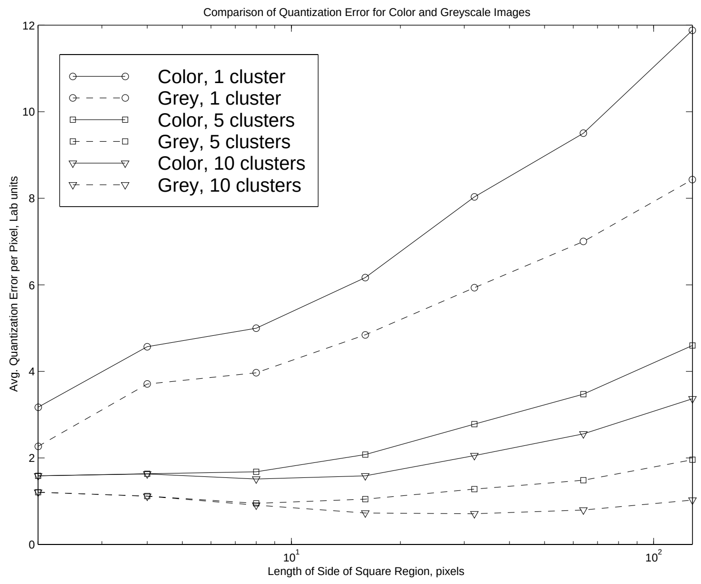


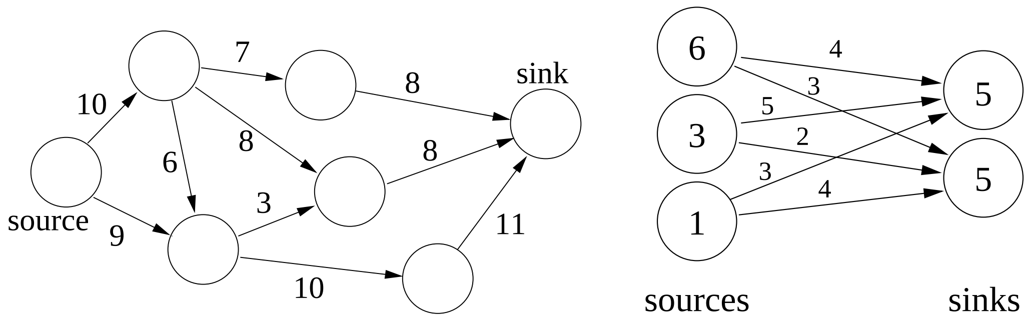
Figure 2.1: Distributions can represent image neighborhoods better than the mean (1 cluster), especially as neighborhood size increases.

the sense that the mean is always perceived to be an intermediate value between the original data points. The mean of a set of colors, however, is perceptually undefined; although the numbers can be averaged in the same way, the resulting color may very well have no perceptual similarity to any of the original colors, violating the informal notion of perceptual closure. Helmholtz [113] noticed that we cannot perceive intermediate colors between red and green, and so creating one seems improper.

2.2 Flow

The reason why distributions have not been used in boundary detection is because computations involving distributions need more time than those involving single values, such as the mean. The nearly continuous increases in processor speed are only now making these computations feasible.

We compare distributions using the concept of *flow*. Flow has already been greatly studied in the literature of graph algorithms [19]. These algorithms take as input a graph whose edges have finite capacities associated with them. They compute the



(a) Maximum Flow Network (b) Transportation Flow Network

Figure 2.2: Two types of flow networks. Maximum flow networks assume an infinite amount of material but only a finite capacity to move the material from source to sink. Transportation flow networks have a finite amount of material, but each edge has a cost instead of a capacity.

maximum amount of material that can be legally transported between two designated vertices, the *source* and the *sink* (see Figure 2.2(a)).

Our notion of flow differs in a few important respects: the sources and sinks can be many and have only a finite capacity, while the edges in the graph, which is bipartite, each have an associated cost rather than a capacity (see Figure 2.2(b)). The goal is to transport the material from the sources to the sinks in a way that satisfies some constraint.

In Section 3.3.2 we will discuss the earth mover's distance, which uses a minimum-cost constraint to compute the distance between two color signatures. This constraint turns our general flow problem into the *transportation problem* (see [67] for an introduction). Later, in Section 5.3, we show how to construct a flow to approximate distance in a three-dimensional manifold inside color space. We are not aware of other attempts to use flow in this way.

2.3 Color

In this final section, we justify our use of color images as a means of exploring the problems of early vision, and in particular how to quantify the difference between colors in a meaningful way.

2.3.1 Why Use Color?

Color plays a key role in our everyday perceptual experience. It allows us to make sense of complex data, such as that presented by maps and graphs. We can often classify objects very easily by noting their color. People make qualitative judgments based on color: shoppers judge the ripeness of a piece of fruit using color cues, and doctors note a patient's skin color as a measure of health. Finally, people associate colors with emotional responses, as evidenced by the planet Mars being named after the Roman god of war due to its red color.

However, these facts do not prove that detecting boundaries in color images is a problem worth considering. For example, Figure 2.3 shows color and greyscale versions of the image used in Figure 1.1. Both images convey the same semantic information needed to understand the scene. Therefore, one could argue that color is unnecessary, and we would make more progress by concentrating on the simpler greyscale image.

Instead, we take the opposite point of view, namely that the equivalence in the boundaries we perceive in the two images requires our models to work as well in color images as in greyscale images. Using color images narrows the set of valid models; many greyscale detection algorithms in the literature can be adapted to color only with significant departures from their original inspiration, if at all. We hold that such models should be reconsidered or rejected as a result.

In addition, there are cases where boundaries exist between two colors whose achromatic components are similar and whose chromatic components are different, resulting in a boundary becoming unresolvable in a greyscale image. This is not likely to happen often in the world, however.

2.3.2 A Perceptual Ground Distance

Having decided that color is important, we must also decide on a method for computing the *ground distance* between two colors, from which we will later compute distances between color signatures. We require this distance measure to be *perceptual*, meaning that the distance between two colors should agree with human perception.

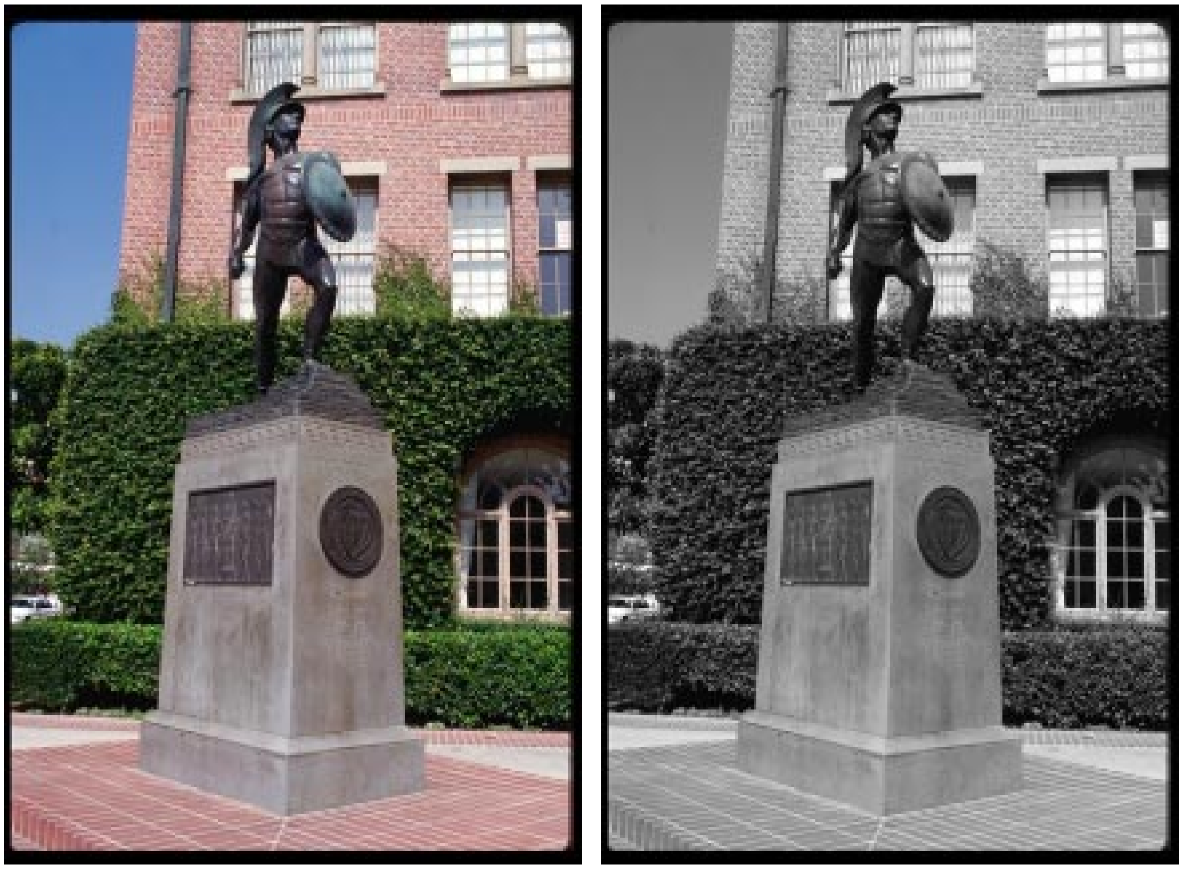


Figure 2.3: All relevant semantic information present in the color image is also present in the greyscale image, calling into the question the usefulness of color for early vision.

To do this we must find the proper combination of a *color space*, an assignment of coordinates to colors, and a distance function on these coordinates.

There is general agreement that the organization of color in our perceptual system is three-dimensional, but the actual assignment of coordinates to colors depends on the task involved. As a result many color spaces exist (see [32] for details on many of them). Few of these spaces were designed to mimic human perception, however. In particular, the Red, Green and Blue (RGB) color space, which is useful for displaying images on computer screens, has hardly ever been advocated as a good space for measuring color distances.

One of the few color spaces that was designed from a perceptual standpoint is the CIE-L*a*b* color space [119]. CIE-Lab (we drop the asterisks) was constructed from the results of color similarity experiments performed by psychophysicists. The Euclidean distance between two nearby colors in this space is intended to be equivalent to their perceptual distance. It is organized according to Helmholtz's opponent-colors theory: L represents lightness, a represents the amount of red or green, and b represents the amount of blue or yellow. CIE-Lab is not ideal; in particular, the experiments that led to its creation used large uniform patches rather than pixel-sized elements, which has a noticeable effect on our perception. However, we have found it to be effective in helping to provide an accurate measure of perceptual color distance.

Unfortunately, the Euclidean distance in CIE-Lab is insufficient for our purposes. An important caveat is that the equivalence between Euclidean and perceptual distances holds for small distances only. For larger distances, the most we can say about a pair of colors is that they are different. For detecting edges and other features this is exactly what is required; once two colors are far enough apart that we can perceive contrast between them, the Euclidean distance between them becomes arbitrary.

This key observation turns out to be independent of our choice of color space. We are not interested in the *physical* properties of the color stimuli; all that we are concerned with is how dissimilar two stimuli are perceived to be. As stimuli become farther apart, their distance should approach unity[3]. We also want our function to be smooth and monotonic in addition to being saturating and a metric. We have chosen:

$$d_{ij} = 1 - \exp\{-E_{ij}/\gamma\}.$$

In other words, the ground distance between color i and color j is an exponential measure, whose steepness is governed by γ (we use $\gamma = 14.0$), of the Euclidean distance E_{ij} between them in CIE-Lab. This function also has the advantage of being roughly linear for small distances, which is why CIE-Lab is still relevant.

Theoretical justification for this measure can be found in the work of Shepard [97], who proposes that an exponential law governs not only color similarity, but also similarity in other perceptual phenomena such as size, shape, pitch, and phonemes. Only γ and the underlying norm (Euclidean, "city-block," etc.) for the physical space

[3]This is reversed from the conventions of psychophysicists, who are usually interested in measuring similarity as opposed to dissimilarity.

changes.

In the next three chapters we apply these principles to the problem of determining boundaries in color images.

Chapter 3

Color Edge Detection

The detection of *edges*, contours separating different parts of an image, has long been a favorite topic in computer vision. In this chapter[1] we analyze the approaches that have been taken to find edges in color images. We then present the compass operator, which uses color signatures as an underlying data structure for detecting step edges[2].

3.1 A Brief History of Edge Detection

Figure 3.1 shows the basic steps carried out by all edge detectors. The first step produces a continuous measure of the degree to which the data in each image neighborhood matches the model of an edge. This step also includes computing model parameters such as edge orientation. The second step is a discrete one where the results of the previous stage at each point are compared with neighboring points, and a decision is made to declare each point "edge" or "non-edge." There are many possible edge models, and here we mention only the most well-known.

The simplest edge detectors are those based on fixed-size zero-mean convolution masks. Roberts' cross [81] was the first, followed by Sobel [77] and Prewitt [79]. These masks varied in size from 4 to 16 pixels and were usually attuned to different orientations. If two orientations were used, the masks computed derivatives in

[1]Much of this work has been published in [89].

[2]Other edge types include *delta* edges or *lines*, which are valuable but less common, and *roof* or *crease* edges, which are even less common.

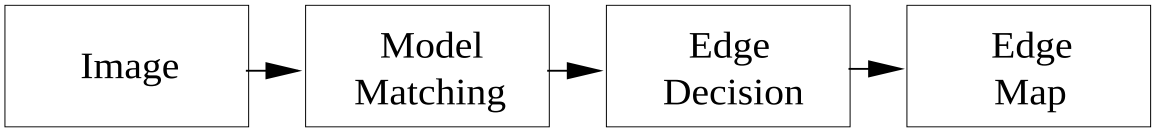


Figure 3.1: Flowchart of edge detection algorithms

orthogonal directions and combined them into an estimate of the gradient. If more than two orientations were used, the maximum response indicated the edge strength and orientation. Edge points were found by suppressing non-maximal responses [86] and thresholding.

Soon afterward Hueckel [38] developed a more complex approach. Coefficients for eight low-frequency continuous basis functions were computed and used to fit the data to a model of a circle split into two neighborhoods of different intensity. Some equivalence between this detector and convolution masks was later shown [2, 85].

A radically different edge detector was proposed in 1980 by Marr and Hildreth [55]. Rather than looking for maxima in first derivatives, they used an isotropic operator, the Laplacian-of-Gaussian, that was based on second derviatives. Zero-crossings marked the location of edges, and these edges formed closed curves. It led to a debate on the merits of isotropic and anisotropic operators that was eventually won by the latter. The Marr-Hildreth operator is still used, though, especially when closed curves are desirable.

The preceding operators were all popular, but their ad hoc nature left enough dissatisfaction to spark a new class of optimal edge detectors, the most famous being those of Canny [10] and Deriche [22]. Because these operators were designed by mathematically deriving definitions of principles such as robustness, localization, and minimal multiple responses, they were more powerful, adapted easily to different scales, and provided better results.

All the above operators invoked the constancy assumption, assuming that any variations were due only to noise. Non-constancy in one dimension was examined early on by Nalwa and Binford [66], who assumed that intensity was constant along the edge and was modeled as a hyperbolic tangent surface across it. Leclerc and Zucker [47] modeled one-dimensional neighborhoods as polynomials up to degree 3,

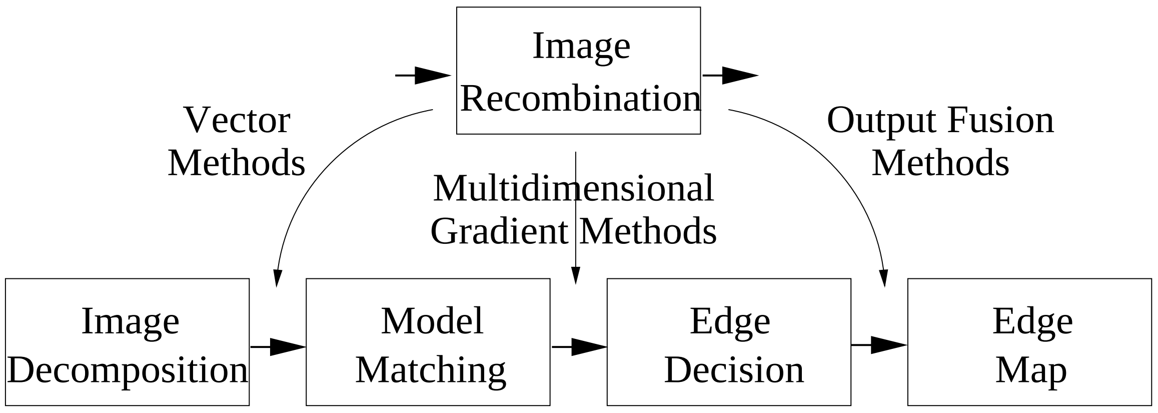


Figure 3.2: Flowchart of color edge detection algorithms. Placing the image recombination step at different points results in output fusion methods, multidimensional gradient methods, or vector methods.

again assuming that intensity was constant along the edge. The later edge detectors of Wang and Binford [115] and Binford and Chiang [8] compensated for shading effects by modeling image neighborhoods as planes with arbitrary surface normals.

3.2 A Review of Color Edge Detectors

The use of color images does not change the nature of edge detection, but it does add one important step, image recombination, shown in Figure 3.2. Color images are decomposed into three component images, and some set of operations is performed on each component separately (including the null set). The intermediate results are then combined into a single output. The point at which recombination occurs is key to understanding the different categories of color edge detection algorithms: output fusion methods, multidimensional gradient methods, and vector methods. We consider each category in turn.

3.2.1 Output Fusion Methods

In output fusion methods, greyscale edge detection is carried out independently in each color component; combining these results yields the final edge map. Computationally, these algorithms are distinguished by a lack of operations that can be written out in vector form.

Nevatia [68, 69] developed the first output fusion method. He computed edges by running Hueckel's edge detector on the luminance component and on two chromaticity components. Each was independent, but the orientation at each point was constrained to be the same. The three separate orientations were weighted to give a final orientation, after which all other parameters were recomputed.

A number of other approaches involve adding the results of a computation on each component together. Shiozaki [99] weighted the results of his entropy operator by the relative amounts of red, green, and blue at a pixel. Malowany and Malowany [54] added absolute values of Laplacian outputs. Two articles by Carron and Lambert [11, 12] investigated the HSI (Hue, Saturation, and Intensity) color space. The first one computed edge strength using a weighted sum over each component and the second was an extension using fuzzy sets. Weeks and Myler [116] averaged together outputs in the HSL (Hue, Saturation, and Lightness) color space.

A more sophisticated approach came from Alberto Salinas *et al.* [4]. They proposed regularization as a way to fuse the outputs of three separate edge maps found by using Canny's edge detector while at the same time introducing "well-posedness" to the inherently ill-posed problem of edge detection. The final edges minimized a functional that summed the perturbations between the final edge map and each components's edge map, plus a curvature measure.

3.2.2 Multidimensional Gradient Methods

Multidimensional gradient methods are characterized by a single estimate of the orientation and strength of an edge at a point. The first such method belongs to Robinson [82, 83], who also appears to have published the first paper on color edge detection. He computed 24 directional derivatives (8 neighbors $\times$ 3 components) and chose the one with the highest magnitude as the gradient.

However, it was Di Zenzo [24] who wrote the classic paper on multidimensional gradients. His method was derived algebraically, but it is perhaps better explained in terms of matrices. A 2×2 matrix is formed from the outer product of the gradient vector in each component. These matrices are summed together and the square root of the principal eigenvalue (also known as the principal singular value) becomes the magnitude of the gradient. The corresponding eigenvector yields the gradient direction.

Di Zenzo showed how to compute this gradient using the Sobel operator, but he did not detect edges directly. Two articles led by Cumani [20, 21] were the first to apply true multidimensional gradients to detecting edges and describing images. Drewniok [28] applied the concept to multispectral satellite images, and Saber *et al.* [92] used it in a segmentation scheme. Chapron [14, 15] used the Cauchy-Deriche gradient in each component.

Others have also developed multidimensional approaches besides Di Zenzo. Two works whose authors include Moghaddemzadeh and Bourbakis [60, 61] used a normalized hue contrast in the HSI color space to compensate for low saturations. Tsang and Tsang [110, 111] used the Sobel gradient in the Hue, Saturation, and Value (HSV) color space, choosing the H or V component depending on other quantities, to suppress edges caused by specular reflection. Macaire *et al.* [52] performed relaxation on the normalized Sobel gradient to classify pixels. Finally, Scharcanski and Venetsanopoulos [93] averaged color vectors together before computing directional derivatives and a gradient.

3.2.3 Vector Methods

In vector methods, the decomposition and recombination steps nullify each other; the vector nature of color is preserved throughout the computation. The work of Machuca and Phillips [53] is ostensibly the first vector method for color edge detection. However, their model of color reduces to one dimension, as they felt that color was useful only where greyscale edge detection fails.

Huntsberger and Descalzi [39] assigned each color a set of fuzzy membership values based on distances to color clusters. Edges occurred where these values straddled the 50% threshold for a color. Pietikainen and Harwood [76] used histograms of vector differences to find the most salient edges. Yang and Tsai [121] projected each color vector in an 8×8 block onto another vector to convert an image to greyscale. Vector projection was also used by Tao and Huang [105], but they projected vector differences onto the $n(n-1)/2$ vectors connecting n color clusters in an image. Djuric and Fwu [25] found edges using the MAP (maximum a posteriori) rule.

Perhaps the most compelling work in vector methods so far has been that of Trahanias and Venetsanopoulos [107, 108]. Their method used the median of a set of vectors, which is the vector in that set whose distance to all other vectors is minimized. Once the vector median has been determined, vectors can be sorted by increasing distance from the vector median, and various statistics can be measured and used for edge detection.

3.3 The Compass Operator

We believe that algorithms that incorporate more vector operations are preferable to those with fewer. Even though the mechanics of color perception involve three separate processes, the sensation of color is unitary [113]. Therefore, our perception (and detection) of edges should be equally unitary; an edge between a yellow region and a green region would be detected mainly in the red component (assuming RGB), yet we do not perceive it to be a "red edge." Also, removing the assumption of constancy means that we cannot average color components together. The notion of "average color" is undefined when two or more different colors exist in a neighborhood.

Output fusion methods have the least amount of vector processing and are usually the most ad hoc. It is not clear why adding three gradient magnitudes, for instance, leads to an accurate characterization of the overall gradient magnitude. Multidimensional gradient methods are the most principled, and they give good results because they are often based on greyscale edge detectors that are optimal in some way. Vector methods are the most appealing on an abstract level, but the methods generally try to reduce the dimensionality of the data to one so that established algorithms can be applied.

We propose a method for edge detection that preserves the vector nature of color throughout the computation while comparing favorably in both a conceptual and a practical sense to multidimensional gradient methods. Imagine a circular "compass" placed over a neighborhood of the image, and as the "needle" (a diameter of the circle) spins, a scalar measure of distance between the color distributions in each semicircle is computed. The needle will eventually point along the orientation that maximizes this distance.

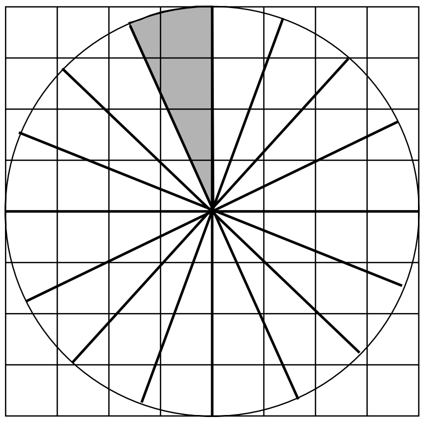
Using color distributions implies a more accurate representation, as each pixel's value is perturbed less than when only averages are used. The ability to model more types of variances than low-order polynomial surfaces corrupted by noise is a significant extension of the cases that can be handled reliably. What is left to show is that edge strength and orientation can be reliably computed under this method. In the following sections, we discuss how color signatures are constructed and how distances are computed between them, and we will show results of applying the operator to natural images.

3.3.1 Creating Color Signatures

We noted in Chapter 2 that a color signature is a data structure consisting of a set of point masses in color space. In this section we discuss how to find and describe these point masses.

In theory every distinct color vector in a neighborhood could become a point mass. However, we prefer more compact signatures for two reasons: (1) comparing large signatures is much more computationally expensive than comparing small signatures, and (2) humans cannot consciously perceive dozens or hundreds of different colors in a neighborhood anyway.

Therefore, we use vector quantization methods (see [33] for an overview) to produce compact signatures. In particular, we use the binary split algorithm of Orchard and Bouman [72] since it was developed for color images and allows control over the



.32	.96
.07	.96
	.62
	.21

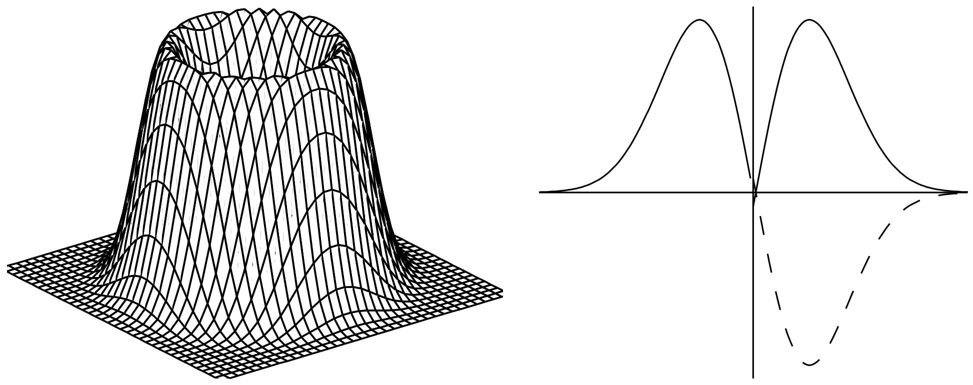
Figure 3.3: Using eight orientations for the compass needle divides this circle of radius four pixels into wedges. The area of intersection of each pixel with the shaded wedge is shown on the right. Updating the color signatures is more efficient using wedges.

signature's size. It is a greedy algorithm that, at each step, splits the cluster whose covariance matrix has the largest principal eigenvalue. A cluster is split by using the plane containing the cluster mean that is normal to the direction of the principal eigenvector. We terminate the algorithm if the maximum number of clusters (usually 10) has been reached, or if the largest eigenvalue falls below a threshold.

We perform vector quantization once for each circular neighborhood in the image. As the orientation of the diameter varies, the amounts of each color in the semicircles will change, but the colors themselves remain constant. The amount that each pixel contributes to one or both signatures depends on how much of the pixel's area falls inside each semicircle and its relative position in the circle.

The area is computed by modeling each pixel as a unit square and calculating how much of each square falls within the circle. Furthermore, using a fixed number of equally-spaced orientations divides the circle into wedges (see Figure 3.3). Computing the area of each pixel in each wedge makes updating the signatures efficient. Note that the center of the circle does not coincide with a pixel center.

In addition to the area, we also factor in the relative importance of each pixel to the computation. We model this importance as a Rayleigh distribution based on the distance from the center of a pixel to the center of the circle. Rayleigh distributions



(a) 2-D Rayleigh distribution (b) 1-D cross section

Figure 3.4: The weighting function for the compass operator is a Rayleigh distribution. By taking its cross section and reflecting half of it, we see the similarities to the edge detectors of Canny and of Deriche.

are expressed in polar coordinates as:

$$f(r) = r\exp\{-r^2/2\sigma^2\},$$

where σ is the scale parameter. The radius of the circle is 3σ.

The shape of this function is illustrated in Figure 3.4(a). It follows some of the intuitive notions set forth by Canny and by Deriche, such as little weight near the center where pixel colors are presumably changing rapidly as we move across the edge, and little weight near the periphery where pixels are, on average, far away from any hypothesized edge.

Figure 3.4(b) shows a cross-section of a Rayleigh distribution. The dashed line is a reflection of half the curve across the x-axis. The edge detectors of Canny and Deriche have weighting functions that are similar in shape to the Gaussian derivative we have just created.

One important difference, however, is that both Canny and Deriche used anisotropic weighting functions in 2-D. In other words, all pixels near a hypothesized edge have zero weight in their schemes. Since our function is isotropic, we are deviating significantly from these edge detectors. The reason for this deviation is computational efficiency; the weights at each pixel would otherwise change as the orientation of the diameter changes. Because we do not assume that the gradient is normal to the edge orientation, we must use more than two orientations, and thus the added computational cost. Also, some neighborhoods might have more than one edge going through the circle's center and we would like to detect all of them.

3.3.2 The Earth Mover's Distance

The problem of measuring the distance between two signatures is an instance of the general problem of measuring distance between probability density functions, which was first considered over 30 years ago. Results from ergodic theory, probability theory, and information theory have combined to produce many different distance measures. A primitive distance measure that fits this category was developed by Levenshtein [50] for two binary strings that need not have the same length.

The constraints of computational efficiency and matching human perceptual similarity have prevented many of these measures from being applied to vision. Within vision, the reliance on histograms has tended to produce algorithms that do not fully take into account the need to use the distances between bins as well as the amounts of mass within each bin. For example, the histogram intersection method [104] is less useful in cases where a bin partition splits a cluster of similar pixels. Cross-bin measures have been developed (*e.g.*, [70]) but also fail to measure similarity accurately [87]. The Hausdorff distance has led to a family of distance measures based on local correspondence between points [40], but we desire a global correspondence.

The Earth Mover's Distance (EMD) overcomes these limitations. The EMD formulates the distance measurement as an instance of the transportation problem in which one signature is considered to be piles of dirt and the other to be a set of holes; the minimum amount of work needed to move the dirt into the holes is the EMD.

More formally, the EMD computes a set of flows f_{ij} between two signatures $X = \{(x_1, \mathbf{v}_{x_1}), (x_2, \mathbf{v}_{x_2}), \ldots, (x_n, \mathbf{v}_{x_n})\}$ and $Y = \{(y_1, \mathbf{v}_{y_1}), (y_2, \mathbf{v}_{y_2}), \ldots, (y_m, \mathbf{v}_{y_m})\}$ that minimizes

$$\sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} f_{ij} \ ,$$

where d_{ij} is the ground distance between $\mathbf{v}_{x_i}$ and $\mathbf{v}_{y_j}$ in the feature space. The following constraints (adapted from [88]) ensure that as much dirt is moved into holes as possible:

$$\begin{array}{rcll} f_{ij} & \geq & 0 & \forall i,j, \\ \sum_{i=1}^{n} f_{ij} & = & y_j & \forall j, \\ \sum_{j=1}^{m} f_{ij} & \leq & x_i & \forall i. \end{array}$$

A few notes are in order. This formulation assumes that the total mass of X is greater than or equal to the total mass of Y. If this constraint is not true, simply switch the two. If X and Y have equal mass, the last constraint becomes an equality, and the EMD then belongs to a family of related measures that includes $\bar{d}$ [29], $\bar{\rho}$ [34], and Wasserstein[3] [26]. Finally, the EMD is a metric whenever the ground distance function is also a metric, which is true for the perceptual distance function used here (see [87] for a proof).

The final distance is normalized by the sum of the flows:

$$\mathrm{EMD}(X,Y) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij} f_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}} \, .$$

The EMD was first used by Rubner, Tomasi, and Guibas [88] to compute the perceptual distance between images by comparing their color signatures. However, it should be noted that in the vision community a one-dimensional form of the EMD known as the *line feature distance* was used by Shen and Wong [96]. It was later renamed the *match distance* and extended to multidimensional histograms by Werman, Peleg, and Rosenfeld [117].

Compared to previous measures used in computer vision, the EMD is characterized by its generality and its robustness. Because the EMD can handle signatures with arbitrary numbers of clusters, it can adapt to arbitrary complexity. Robustness stems from the fact that perturbations in cluster centers or even small changes in the number of clusters do not have disproportionate effects on the result.

The key feature that separates the EMD from similar measures in ergodic and probability theory is that it can find distances between signatures of unequal total

[3]This Russian transliteration is the accepted one in the literature, but the cited article uses "Vashershtein."

mass. Its forumulation as a combinatorial optimization problem allows it to find the subset of the larger signature that corresponds best to the smaller signature. In Chapter 4, this property will become important because, when detecting corners, the signatures naturally have unequal total mass.

The final reason for choosing the EMD is that previous work applied to color and to texture [87, 17] shows it to be an effective tool for matching human perception under a wide range of conditions. For signatures that are already similar to each other, measures such as χ^2 statistics or the Jeffrey divergence (a symmetric form of relative entropy) give slightly better results [87, Ch. 6]. We are more interested in dissimilarity, however, so comparing such signatures accurately is not relevant.

To use the EMD to find edges, we must supply the cost of moving a unit mass between every pair of colors. This is accomplished by using the perceptual distance measure from Chapter 2.

3.3.3 Extracting Edge Information

We can now explain the basic algorithm for local edge estimation with the compass operator: a circular neighborhood is clustered in CIE-Lab, and for each value of θ, the orientation of the diameter, we compute the EMD between the color signatures of the resulting semicircles. The result at each neighborhood is a function $h(\theta), 0 \leq \theta < 180$, and this section explains how edge information is extracted from this function. Afterwards, we extract edges by comparing this information at neighboring points.

3.3.3.1 Strength and Orientation

Figure 3.5 shows an application of the compass operator on parts of two images, the first an ideal step edge and the second taken from an image containing a bush and a tree. The plots show the EMD for each orientation of the diameter, which is spaced $15°$ apart (the default). The *strength* of the edge is the maximum value of $h(\theta)$ and the *orientation* of the edge is the argument that produced the strength.

We do not expect the true orientation of the edge always to be one in the sampled range, however. The vertex of the parabola that contains the three highest EMD

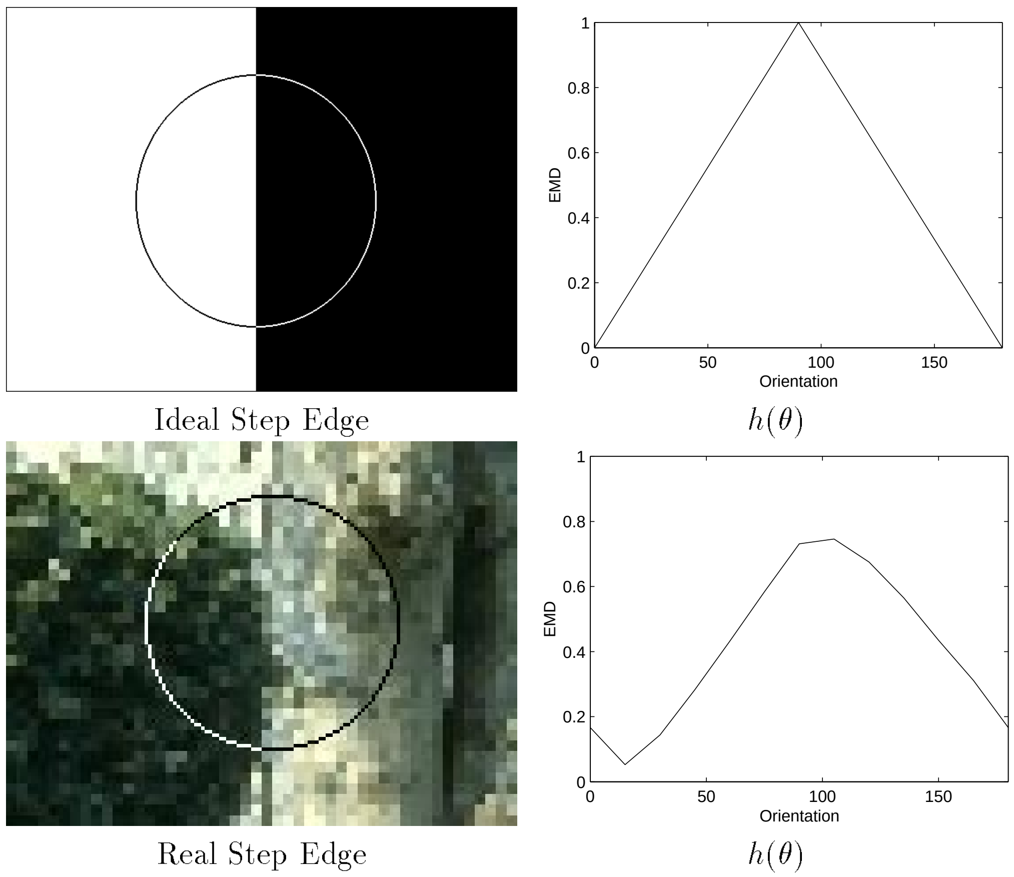
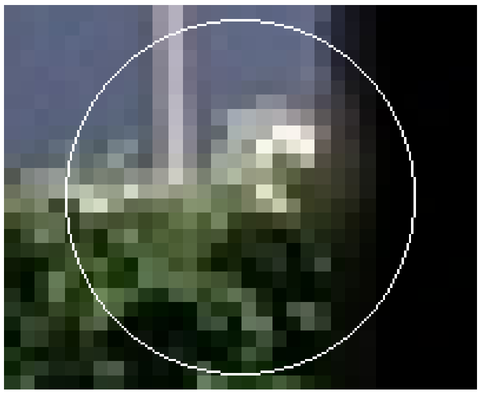


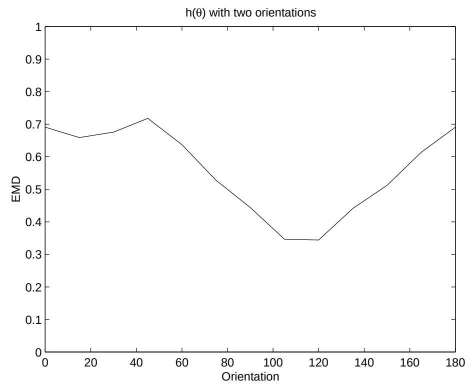
Figure 3.5: Applying the EMD to a circular neighborhood over a range of diameter orientations produces a function from which edge information is extracted.

values gives the true strength and orientation. Note that curve-fitting is necessary because we do not assume that the responses at $0°$ and $90°$ can be combined to form a gradient.

In addition, there are isolated instances where two separate edges can be made out, resulting in two separate maxima in $h(\theta)$. The algorithm can recognize this case and record two orientations, but the strength for both will be equal to the maximum of the two. An example is shown in Figure 3.6.

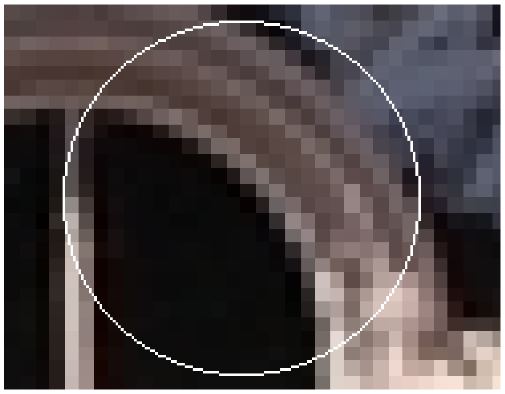


Two Step Edges

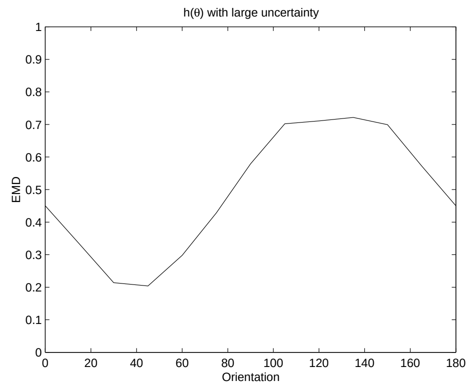


$h(\theta)$

Figure 3.6: In rare instances, $h(\theta)$ will have two peaks with the same strength, indicating two separate edges. The algorithm can record both orientations, in this case $0°$ and $42.7°$.



Curved Step Edge



$h(\theta)$

Figure 3.7: Curvature in an edge causes many orientations to have roughly the same EMD. The width of this interval is a measure of the uncertainty in the edge's orientation.

3.3.3.2 Uncertainty

The edge in the bottom half of Figure 3.5 is somewhat curved instead of straight, which is reflected in the fact that the two highest EMD values are almost equal. Figure 3.7 shows a more dramatic example where the curvature of the edge causes many orientations to have almost the same EMD values. Since parabolic interpolation would be unstable, we pursue a different approach.

The width of the continuous interval whose values are all close to the maximum EMD (meaning greater than 95%) is the *uncertainty* of the orientation. The strength is the maximum sampled EMD, and the orientation is the midpoint of the interval.

3.3.3.3 Abnormality

One of the main assumptions of gradient-based methods is that the derivative of the image intensity function in the direction of the edge is zero, or equivalently that the edge exists along an isocontour of intensity or color, which is not always true. For an ideal step edge (such as that in Figure 3.5) this is the case; at the orientation normal to the edge, the two color signatures are identical and the EMD is exactly zero.

In the real image examples we have shown so far, the minimum EMD over all orientations is never exactly zero due to inhomogeneities in the spatial distribution of colors. The closer the minimum is to zero, the more the image data matches the model of an ideal step edge regardless of the strength. For this reason, the minimum value of $h(\theta)$ is called the *abnormality*.

Although abnormality can provide a quantitative measure of confidence in our edge computation, it serves a much more interesting purpose. When abnormality is unusually high, it indicates a complete lack of symmetry in the image data that very often corresponds to a junction where three or more regions meet. The notion is very similar to the original idea of Moravec [63], who developed an "interest operator" by looking for neighborhoods where the sum of squared differences of adjacent pixel intensities is high in all directions. Abnormality naturally extends this idea to entire neighborhoods.

Figure 3.8, where three regions meet in a junction, illustrates this concept. Points near the edges have high strength, but only those near the junction have high abnormality as well. In addition, the abnormality of these points appears to be well modeled by a triangular pyramid. The sides of the base of the pyramid are as normal as possible to the directions of the edges that form the junction while still remaining a triangle.

These properties are especially attractive because they allow reconstruction of edges near junctions, which are among the most difficult places for any edge detector

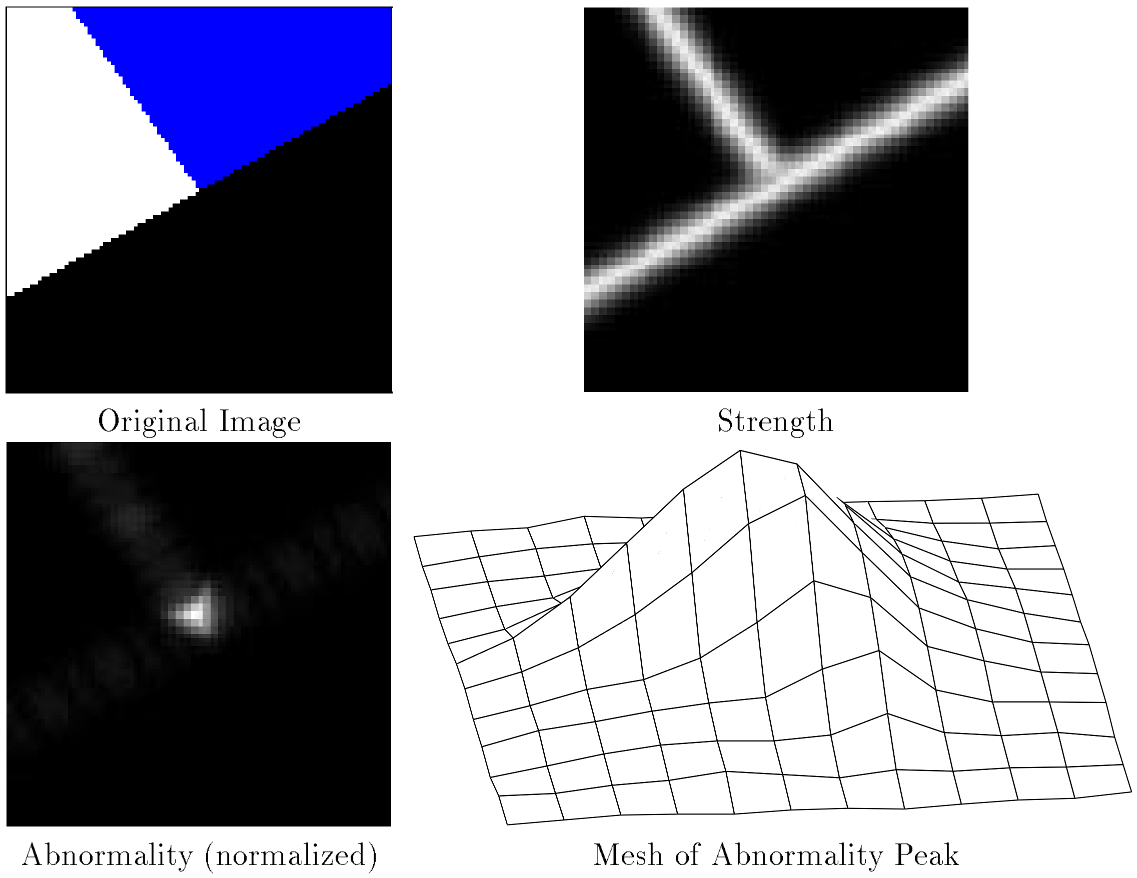


Figure 3.8: Whereas the strength at each point represents the maximum EMD value over all orientations, the abnormality represents the minimum, which is very high near junctions. The mesh shows a shape that resembles a triangular pyramid. Strength can often approach a value of 1, while abnormality is usually not higher than 0.67.

to function properly. The behavior of abnormality in natural images, however, is more complicated, and we have not fully investigated its properties.

3.4 Results

The practical advantage of the compass operator is that it provides superior strength and orientation estimates in the following situations:

- Large neighborhoods involving hundreds of pixels and many colors.

- Near junctions, especially when the edges have unequal contrast.

- Between textures, especially those with edges that might cause interference.

This section provides a variety of examples to illustrate these points by comparing the output of the compass operator to that of a multidimensional gradient method formed by combining Di Zenzo's gradient ideas in CIE-Lab with Canny's approach into a single operator. We have chosen this operator because the vision community treats Canny's detector as a de facto standard of edge detection due to its simplicity and ease of implementation. For both operators the actual edges are extracted by using non-maximal suppression and hysteresis thresholding. When the compass operator produces two or more responses at a point, only one needs to be a maximum.

Figure 3.9 is a closeup of the helmet and the occluded background from the statue image. This sub-image contains four regions, two light and two dark. The edge separating the two dark regions has relatively low contrast. One would expect that the low-contrast edge can be extracted by using a low threshold, but this turns out not to be the case.

At smaller scales ($\sigma = 1$ or 4), the gradient magnitude cannot detect the low-contrast edge at all. Neighborhoods centered on this edge include enough pixels from the bright region that the orientation of the gradient becomes consistent with the high-contrast edge, and these responses are suppressed because they are non-maximal. The use of a saturating distance measure would prevent the bright region from having such a marked effect on the computation.

If σ is increased past 8, the boundary of the helmet is found, but only at the cost of destroying the other edges. The four regions cannot be separated except by analyzing output at different scales, which is an open problem. The compass operator, on the other hand, finds all the relevant edges at one scale.

In addition, the abnormality located two junctions in the image, which would be useful for closing the gaps and labeling the helmet's boundary as an occluding edge. However, the peaks are not perfectly localized, likely due to the fact that these two junctions are close to each other relative to σ.

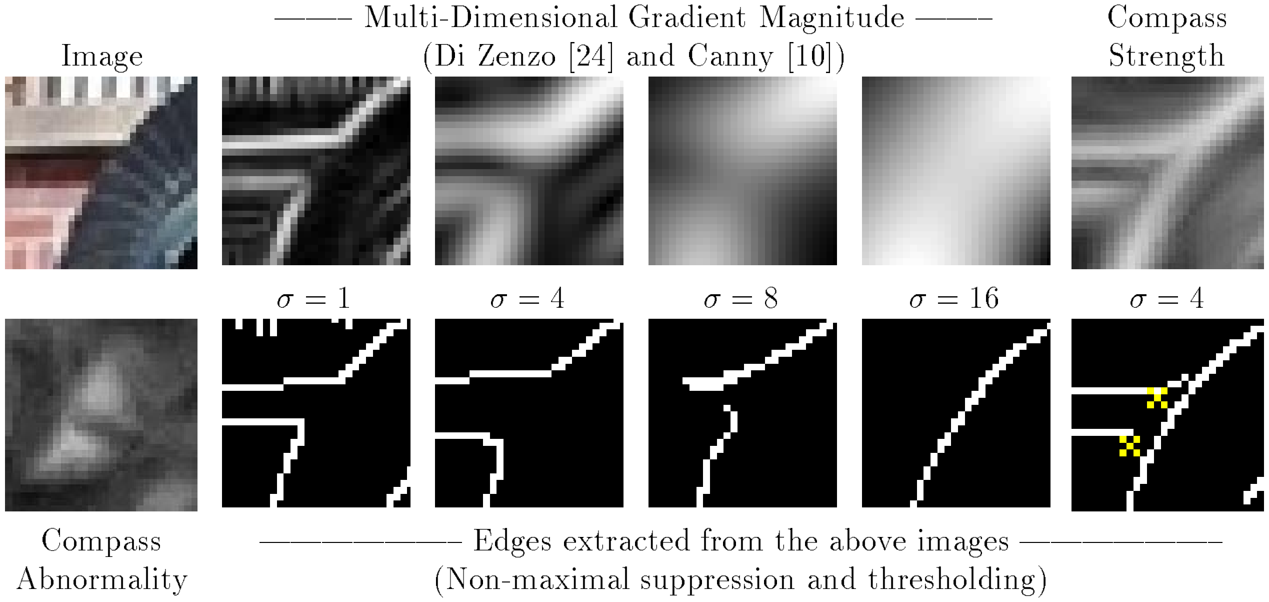


Image
——– Multi-Dimensional Gradient Magnitude ——–
(Di Zenzo [24] and Canny [10])
Compass Strength
$\sigma = 1$
$\sigma = 4$
$\sigma = 8$
$\sigma = 16$
$\sigma = 4$
Compass Abnormality
————— Edges extracted from the above images —————
(Non-maximal suppression and thresholding)

Figure 3.9: A region containing high- and low-contrast edges. The gradient magnitude is insufficient to find all of the edges at any one scale and would require further processing, while the compass operator does find all the edges at one scale. Two junctions are located in the abnormality image, but their peaks (yellow crosses) are not well localized.

The next example, Figure 3.10, shows two regions, one with strong edges and pixels that are the same color as the other region. The edge between them should be salient over a wide range of scales. However, the multidimensional version of Canny connects the true boundary to an intra-texture edge at $\sigma = 8$, while the compass operator does not. Increasing σ further eventually corrects the mistake, but note that the edge at $\sigma = 16$ is quite curved. These differences are due more to the use of distributions to model different colors accurately than to the distance measure used.

When the output on larger image regions is compared, the differences are often quite striking. Figure 3.11 shows the statue along with edges extracted by both operators at a medium scale ($\sigma = 4$). Smaller scales pick up too many edges in the ivy and bricks to be useful. The differences between the two are most noticeable in the helmet, shield, right arm, and legs of the statue. Admittedly, the compass operator's edges do not fully segment the statue from the background; however, it would almost certainly be easier for a segmentation algorithm to find the statue correctly using

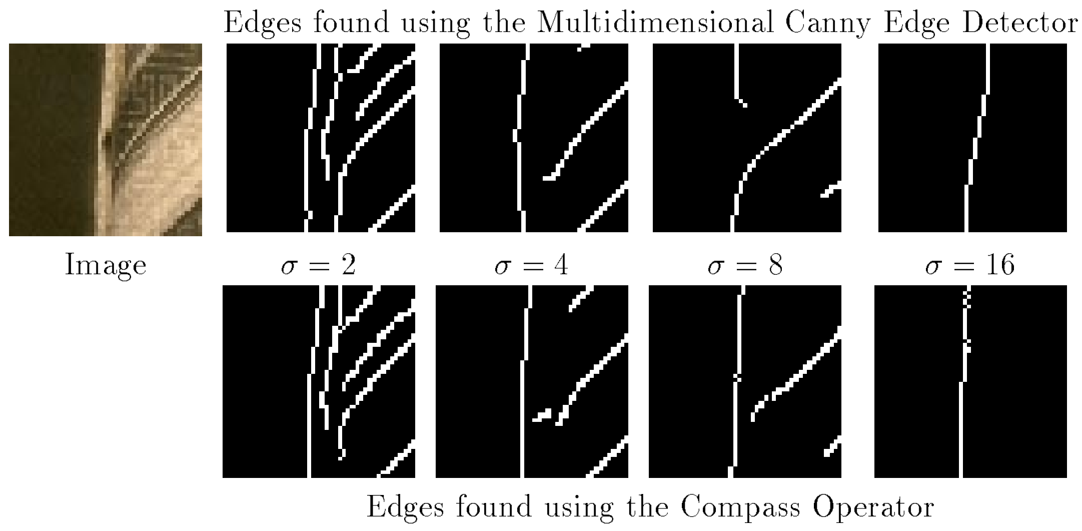

Edges found using the Multidimensional Canny Edge Detector
Image
$\sigma = 2$
$\sigma = 4$
$\sigma = 8$
$\sigma = 16$
Edges found using the Compass Operator

Figure 3.10: The main edge in this image is salient at a large range of scales even though one region contains strong edges. Canny's edge detector makes an error before $\sigma = 8$, and the lone edge at $\sigma = 16$ is badly localized. The compass operator creates more stable edges as scale changes.

those edges as opposed to Canny's.

Segmentation, as we have said earlier, is too ambitious a goal in general. A more practical goal is detecting and labeling occlusions, which is important for determining the relative depth of objects in the camera's field of view. In Figure 3.12, labeling occlusions properly is critical for future research that would be able to reconstruct the snake.

As in the statue image, a small σ results in too many details being detected to be of use. Figure 3.13 shows the edges detected by both operators at $\sigma = 4$. Finding the snake by using either of these images as a guide is bound to be an arduous task; nonetheless, the general impression is that the compass operator has done a better job of keeping the occluding edges intact. In these edge maps, thresholding has been omitted, and the darkness of each pixel is proportional to its strength. Canny's output is lighter on average because the gradient magnitude occupies a much wider range and a few points have a very large magnitude. Because the compass operator uses a saturating distance measure, more of the edges have large strengths relative to the range of values.

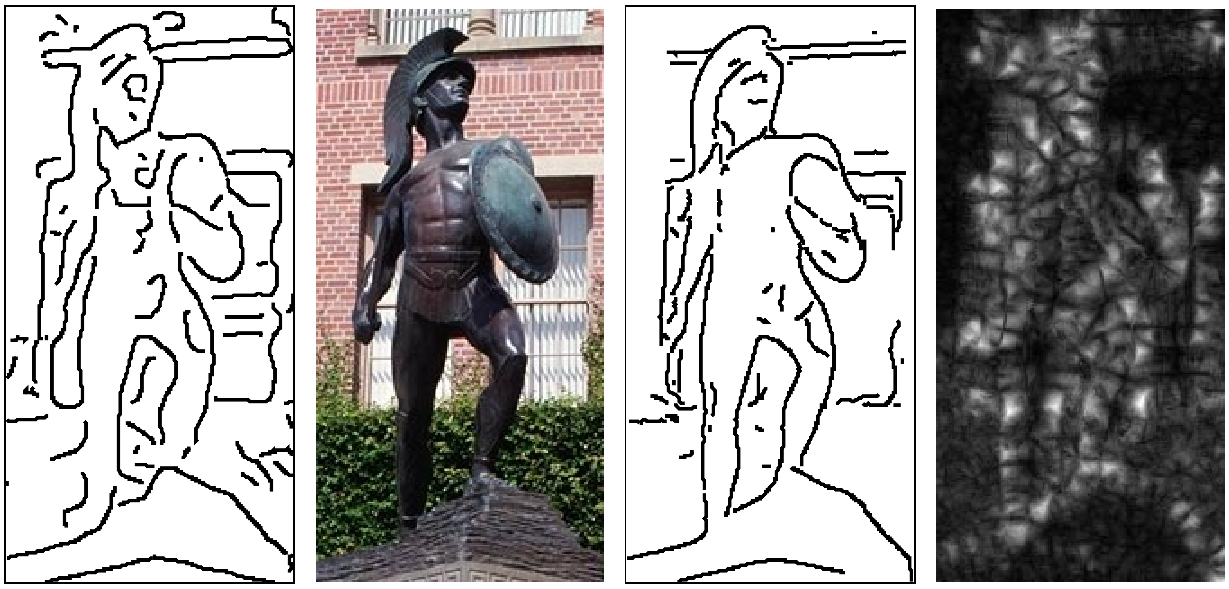


Figure 3.11: Comparison of edges on a 320×160 image. Note the differences in the helmet, shield, right arm, and legs. The peaks in the abnormality, normalized for display, indicate potential junctions. Both algorithms were run using $\sigma = 4$.

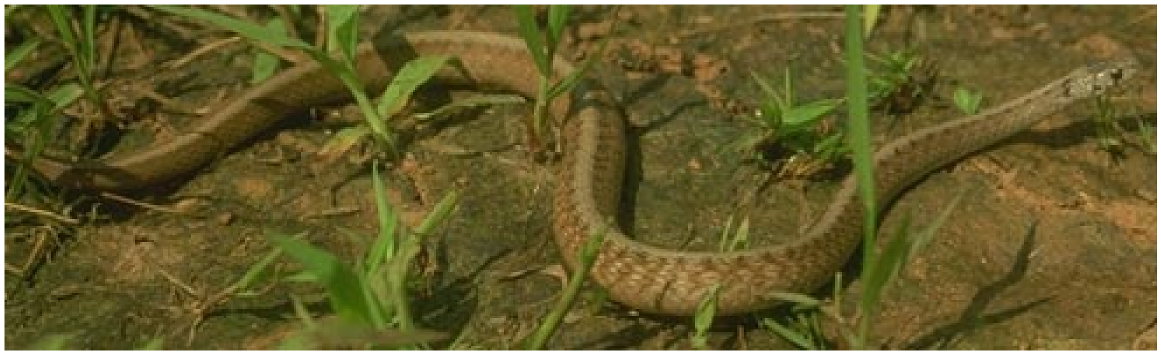
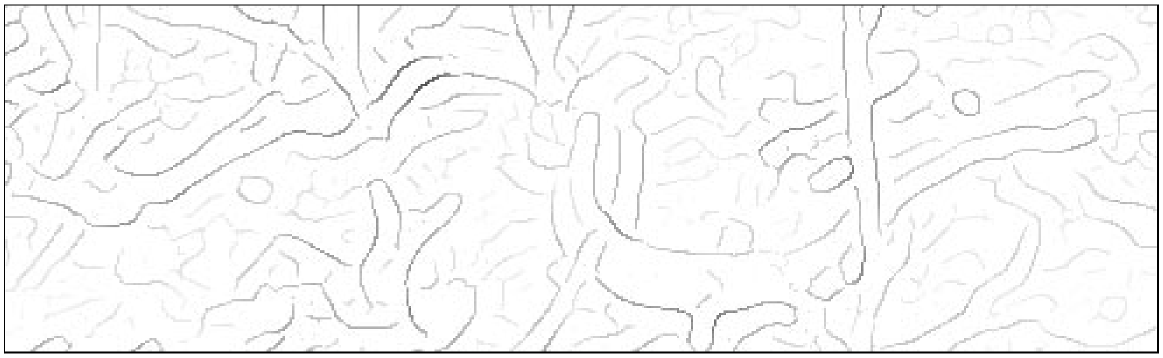
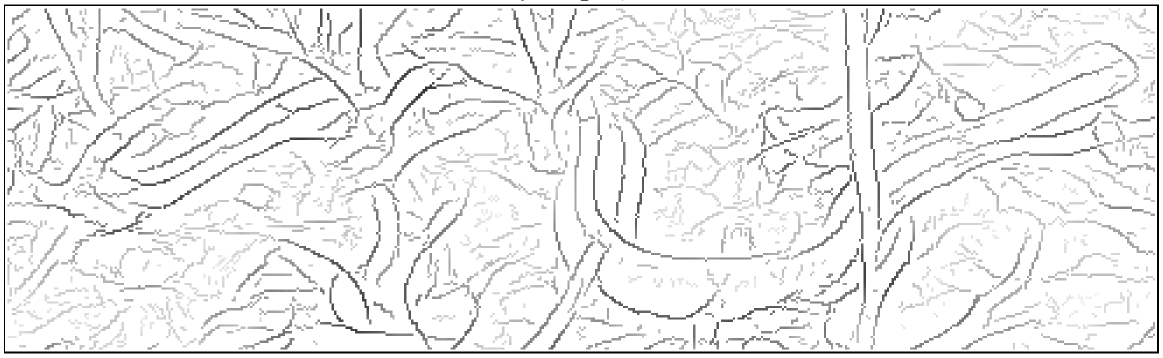


Figure 3.12: Distinguishing edges of the snake from edges of the grass at points where the two meet is crucial for future identification of the snake.



Canny edges, $\sigma = 4$



Compass edges, $\sigma = 4$

Figure 3.13: The compass operator better separates snake edges from grass edges. Darker edges have higher strengths relative to the maximum of each operator's range.

Closer examination of some of the occlusions in this image (see Figure 3.14) provides better insight into the comparative behavior of the two operators. In the first row, both operators find the three edges visible in the image, but the compass operator's edges are better localized and come closer to the edge of the blade of grass. In the second row, the edges found by Canny mix boundaries of the snake and grass together, while the compass operator is better able to resolve edges of the snake on both sides of the grass. The final row again shows Canny combining snake and grass edges together so that neither will be easy to identify; the compass operator gives us a much better chance of being able to distinguish occluding edges from occluded edges.

The one area where the compass operator fails to match up well against a multidimensional version of Canny's operator is in running time. Canny is separable (its 2-D

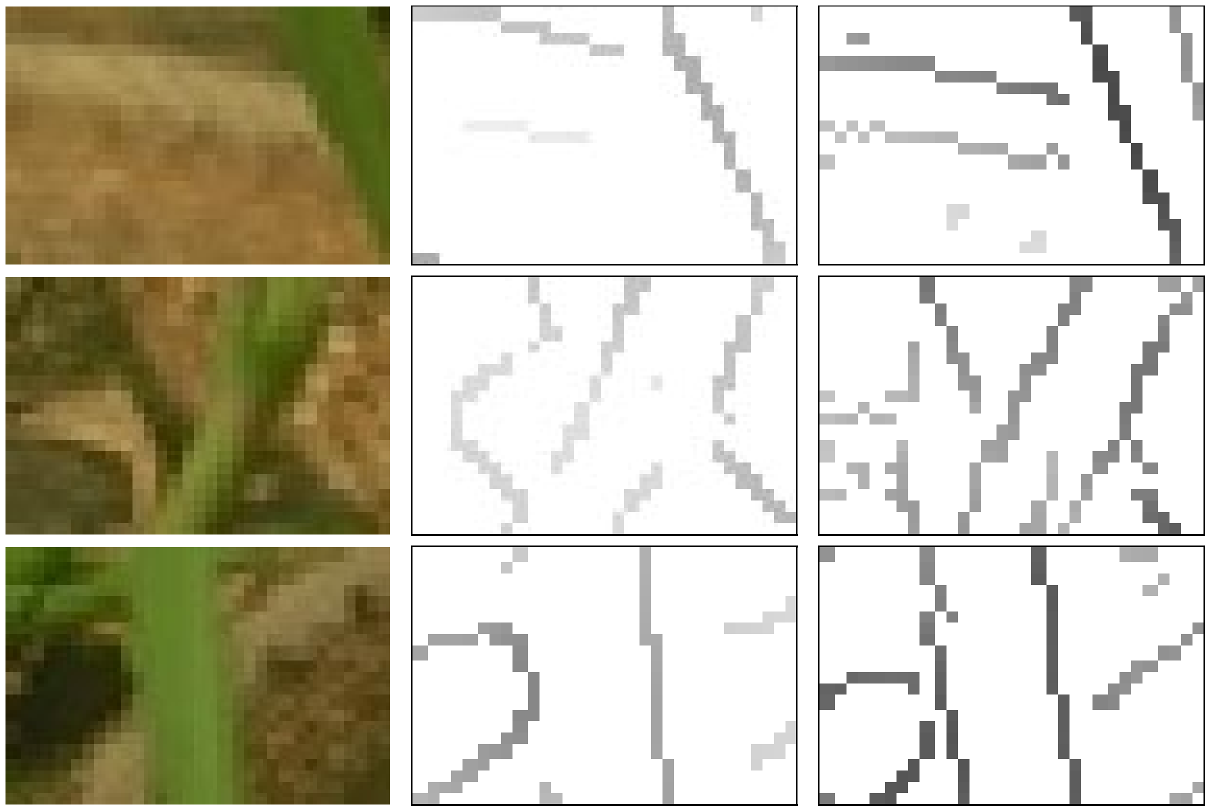


Figure 3.14: Detail of edge maps in Figure 3.13. Because the compass operator accurately models neighborhoods with 3 or more colors, occlusions will be easier to reconstruct.

convolution mask can be split into two 1-D masks), computes weighted averages, and finds the Euclidean distance between these averages at two orientations. By contrast, the compass operator is non-separable, performs vector quantization, and finds the EMD between color signatures at a number of orientations. As a result, Canny's operator can be run on the statue image (768×512) at practically any reasonable scale in a fraction of a second on an SGI Octane, while the compass operator requires almost 3-1/2 minutes at $\sigma = 1$, 14 minutes at $\sigma = 4$, and 33 minutes at $\sigma = 8$.

Obviously, the compass operator is too slow to use except in situations where the images are very colorful and textured or where obtaining better results than Canny can offer is crucial. The inherent computational limitations provide little incentive to optimize the current implementation. Schemes for reducing the overall running time that have been considered but not implemented include orientation prediction, using histograms instead of signatures, alternatives or approximations to the EMD, and pre-quantizing pixel colors.

One key issue in improving the operator's future performance is stability, both of pixels and of neighborhoods. Pixels in the interior of the image are part of approximately $9\pi\sigma^2$ different circular neighborhoods. If a pixel is stable, each of its cluster representatives is close to the pixel and has little variance as a group. Unfortunately, pixels along edges usually have uncommon colors, and so they can be mapped to very different cluster representatives. The result is instability and edges that appear noisy compared to Canny's.

Another type of stability occurs when comparing neighborhoods. If two neighborhoods share 95% of their pixels, for example, we would expect the EMD between them to be no higher than 0.05. Because the binary split algorithm is run on each separately, however, this is not guaranteed. More or fewer clusters may be created, or clusters may be perturbed. Ensuring stability between neighborhoods can potentially reduce both artifacts and computation time.

3.5 Greyscale Edge Detection

The compass operator can be run on binary, greyscale, color, or hyperspectral images with hardly any modification. All that is necessary is that a meaningful distance measure between any two points in the chosen image range be defined. However, the fact that greyscale images have only one value per pixel allows for significant speedup of the algorithm. Since this is a fault with the compass operator in general, and since most edge detectors are designed for greyscale images, it is useful to understand the differences involved in greyscale images.

The two biggest differences are the representation of the pixel values and the way in which signatures are created. Since we represent color in the CIE-Lab color space, and since the intensity range is a subset of this space, it makes sense to continue to transform pixels into this space and use only the L values.

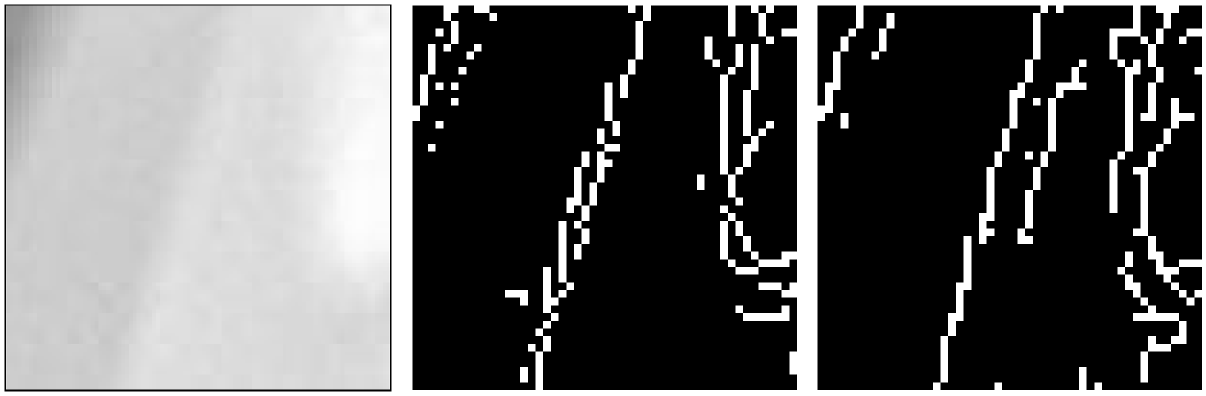


Image with Blurry Edges   CIE-Lab Lightness Edges   RGB Intensity Edges

Figure 3.15: The non-linear effects of transforming intensities into lightness values in CIE-Lab often increases the noise and results in poor edge localization.

However, the transformation to CIE-Lab is non-linear; it therefore has non-uniform effects on the noise (noticed previously in [103]) that can lead to poor edge localization. The example in Figure 3.15 shows how dramatic this difference can be.

The other main difference is how the signatures are computed. We partition the range of intensities into intervals and take weighted averages of the pixels whose intensities fall into each interval. If the weights were merely summed, we would have histograms instead. The number of clusters is now fixed at a larger number (many clusters may have zero mass), but the one-dimensional EMD, the topic of the next section, is still much faster to compute.

3.5.1 A Faster Earth Mover's Distance

The match distance used by Shen and Wong finds the difference between two one-dimensional cumulative distribution functions. This metric was described earlier by Vallender [112], who showed it as a restricted case of the Wasserstein distance on R^n using the L_1 distance. Under these conditions the EMD reduces to this distance for one-dimensional signatures.

We would still like to use the perceptual distance metric, however, so the match distance will not suffice. In addition, we cannot use the EMD in its original formulation. Such results are likely to be counterintuitive because the perceptual distance

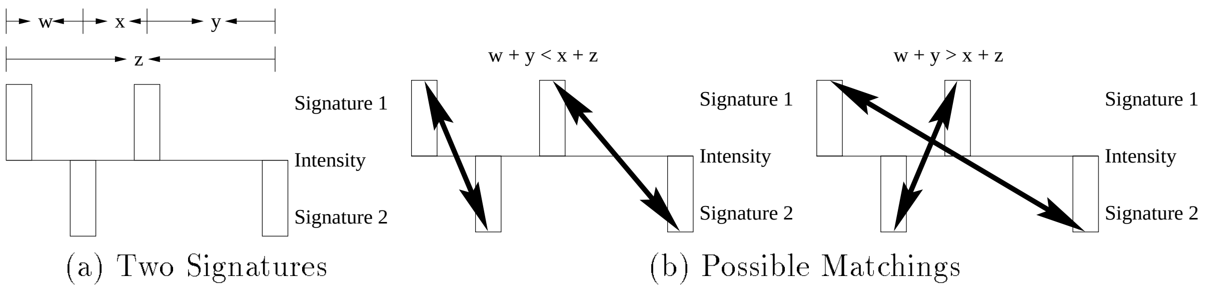


Figure 3.16: The distances between grey levels in the two signatures of (a) need not satisfy the triangle inequality. Therefore, either of the two matchings in (b) is possible. The first is monotonic, the second is not because the arrows intersect.

metric is not additive. Consider the example in Figure 3.16(a), which shows two signatures, each of which has two clusters that have 50% of the mass of each signature. Since we cannot guarantee that the distances w, x, and y add to z, two possibilities for the match can occur as shown in Figure 3.16(b).

We believe the first alternative should always be chosen, regardless of the values of w, x, y, and z, because the matching is *monotonic*. To be more precise, if the EMD decides that mass at a value of $\mathbf{u}$ in one signature is matched against mass at a value of $\mathbf{v}$ in the other, then all mass at values less than or equal to $\mathbf{u}$ must be matched with masses at values less than or equal to $\mathbf{v}$, and similarly for masses at values greater than or equal to $\mathbf{u}$.

The algorithm for computing this variation of the EMD is shown in Table 3.1. Intuitively, the algorithm enforces monotonic matching while using our perceptual ground distance to compute the work. Like the match distance, it can be computed with one pass over the data.

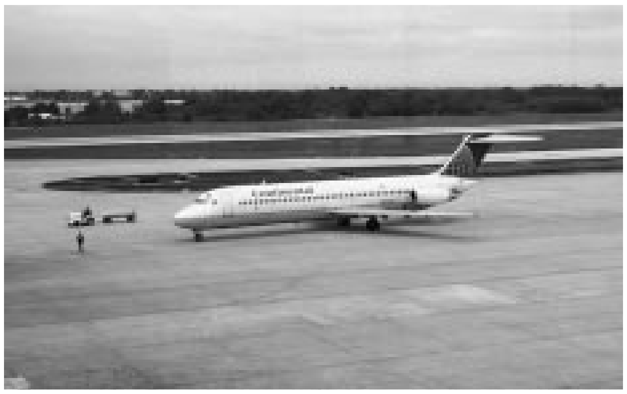
3.5.2 Results

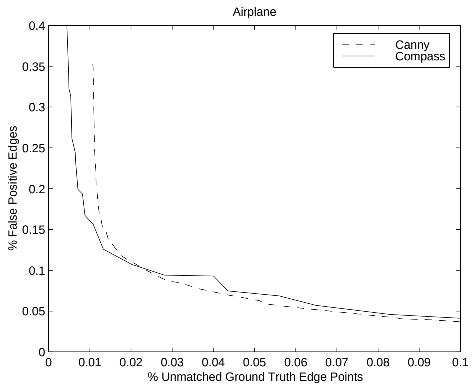
The comparison of greyscale edge detectors is a topic that has received more interest as the number of proposed edge detectors has increased [1, 27, 98]. The authors of [9] agreed to test the compass operator using the framework outlined in their paper. Comparisons are made with the Canny operator, as it was deemed to be one of the two best algorithms (the other being the Heitger operator [37]).

```
float GreyEMD(Signature $\{(u_i, p_i)\}$, Signature $\{(v_j, q_j)\}$)
/* Each point mass is located at $u_i$ ($v_j$) with an amount $p_i$ ($q_j$) */
$i$ = -1
$j$ = -1
$leftover_u$ = 0.0
$leftover_v$ = 0.0
$work$ = 0.0
do (forever)
   if ($leftover_u == 0.0$)
      advance $i$ to next non-empty interval
      if (no more intervals)
         return($work$)
      $u_{amt}$ = $p_i$
   else
      $u_{amt}$ = $leftover_u$

   {similar computation for $\{(v_j, q_j)\}$ and $j$}

   $mass$ = $\min(u_{amt}, v_{amt})$
   $work$ += $mass$ * $(1 - \exp(-|u_i - v_j|/\gamma))$
   $leftover_u$ = $u_{amt}$ - $mass$
   $leftover_v$ = $v_{amt}$ - $mass$
end
```

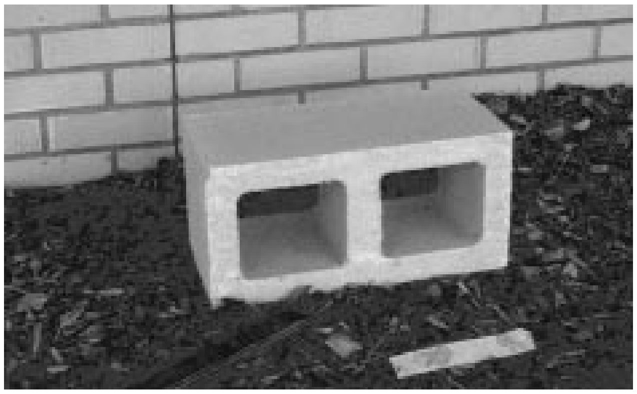
Table 3.1: The algorithm for finding the EMD between two greyscale signatures with a perceptual ground distance.

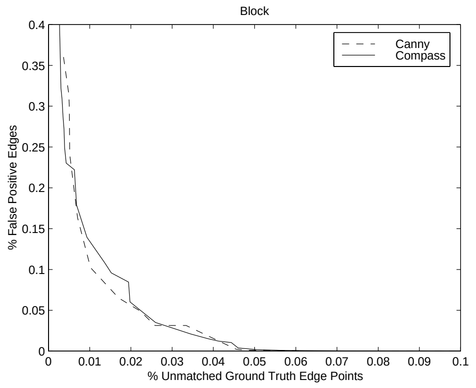
Examples from their test set of 10 images are shown in the left column of Figure 3.17. Note that the images have considerably less complexity than the examples shown for the color version: objects are untextured and most of the edges are in focus. Each edge detector is run over a range of values for the different parameters (for both the Canny operator and the compass operator, the parameters were σ and the two values for hysteresis thresholding). Each set of parameters generates an edge map that is compared to a ground truth specification created by a human. The edge map becomes a point on a plane whose coordinates measure the percentage of ground truth points not identified and the amount of false positives.



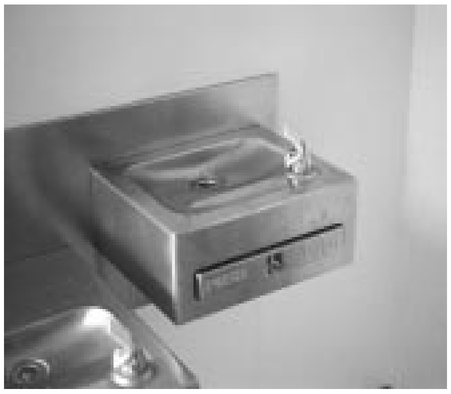


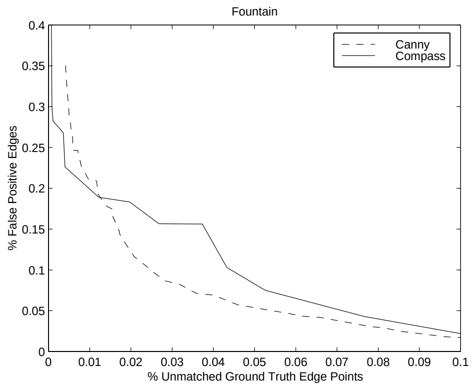
Best Comparative Result





Typical Comparative Result





Worst Comparative Result

Figure 3.17: Three images from the test set, and the Receiver Operating Characteristic (ROC) curves for Canny (dotted line) and the greyscale compass operator (solid line). (Data Credit: Christine Kranenburg and Kevin Bowyer)

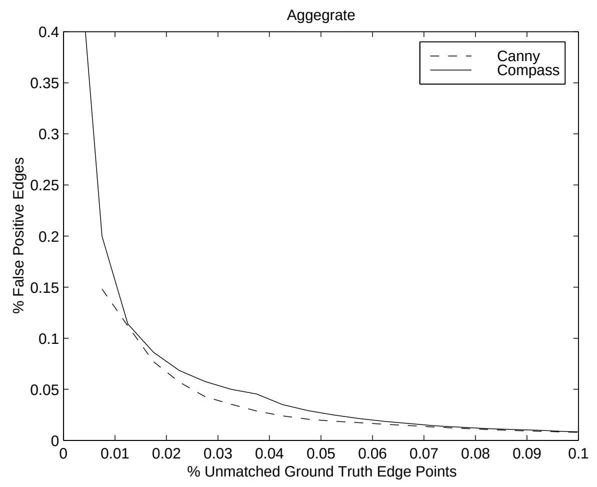


Figure 3.18: Aggregate ROC curves averaged over all 10 images. (Data Credit: Christine Kranenburg and Kevin Bowyer)

From these points a Receiver Operating Characteristic (ROC) curve is generated. The curve is generated by the subset of points that do not have other points to their lower-left; in other words, no point on the curve can have more false positives and more unmatched ground truth than another point in the set. The smaller the area under this curve, the better the operator's performance. Refer to [9] for details.

ROC curves for Canny and the compass operator for some of the images are shown in the right column of Figure 3.17. In all 10 images, two trends are apparent. The first is that Canny's "elbow" is always at least a little closer to the origin than that of the compass operator. The second is that the compass operator is always able to find at least as much of the ground truth as Canny is, and sometimes much more. The displayed images were chosen to show where the compass operator compares most and least favorably, as well as an image from the middle in terms of performance. The overall performance, shown by the aggregate ROC curve for all 10 images in Figure 3.18, has been deemed to be of roughly the same performance as Canny [45].

It appears that there is a tradeoff between the power of distributions and the EMD versus the non-optimal shape of the compass weighting function. For these images, in which most edges can be found at small scales (the largest value of σ used was 3.0) and in which most regions are untextured, the weighting function appears to be the key, while the signatures are modeling only characteristics of the noise. In more challenging images, the benefits of distributions make the relative weighting of pixels less important.

The running time for the greyscale compass operator is orders of magnitude faster than the color version but still slower than Canny. For the Fountain image (580×502), the running time is 1-1/2 minutes for $\sigma = 1$, 2-1/2 minutes for $\sigma = 4$, and 4-1/2 minutes for $\sigma = 8$.

3.6 Conclusions

The strength of the compass operator lies in its ability to find edges in difficult situations, which is mostly due to the fact that its representation of a neighborhood, a distribution of colors, is more general than previous methods. The saturating color distance function coupled with the EMD provides an effective method for measuring the perceptual distance between two color signatures. As a result, edges separating textured regions are more likely to be detected, and we have better guarantees that occluding edges will be recovered properly for labeling.

The compass operator in its current implementation will not supplant other edge detectors in the near future, however. The computational cost associated with forming and comparing color signatures would seem to outweigh its benefits for most of today's applications. Furthermore, it does not outperform Canny's operator on images where Canny's assumptions largely hold. The promise of being able to detect more of what a human perceives to be ground truth than Canny is able to does give incentive for future research, though.

Such research should focus on the issues of stability and speed. Pixels near an edge are likely to have unique colors that will not be given their own cluster. Perturbations in the clusters that are formed may cause these pixels to be represented by very different colors at different times. Computation time can be lowered by investigating approximations to the EMD as well as faster, possibly global, clustering schemes.

In the next chapter we discuss a generalization of the compass operator that allows us to find corners in addition to edges.

Chapter 4

Color Corner Detection

A *corner* is defined as a point where converging lines, edges, or sides meet[1]. Corners are therefore heavily dependent on edges, a fact we will exploit to detect them. From the standpoint of early vision, corners have two important properties that distinguish them from edge points. The first is that they can summarize a set of edges. Just as a 2-D region can be succinctly described by a set of edges, a small set of corner points is sufficient to describe a set of 1-D straight edges [123]. The second property is that corners provide reliable measurements about the shape and location of objects. When tracking an edge over a sequence of images, for example, only the component of translation normal to an edge can be recovered, a difficulty known as the *aperture problem* [65].

However, corners have the disadvantage of being harder to detect than edges. In the first place, an edge is a set of connected points, while a corner is only one point. If one edge point is lost, it can often be recovered, whereas a missing corner cannot. Also, the dictionary definition of a corner implicitly assumes that the sides of a corner are straight. However, edges often curve, and deciding that a corner exists is essentially done by thresholding the change in orientation along a boundary. Scale also plays a role, as the transition from no curvature to high curvature must happen over a small part of the image relative to the chosen scale. Corners also have an

[1]Other researchers use this definition for "L-corners" or "L-junctions." Other letters used include T, Y, X, and K. We refer to these four as junctions.

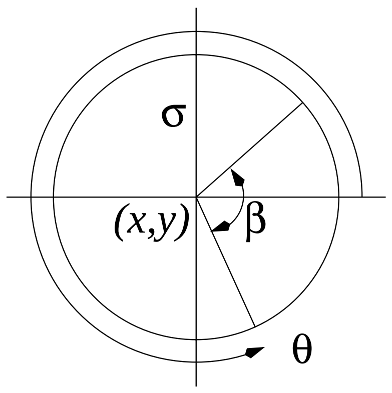


Figure 4.1: The parameters of the corner detector. (x, y) represents the center of the circle, and σ is the scale parameter (the radius is still 3σ). The orientation of the clockwise side of the corner is θ, and β is the angle subtended by the corner.

additional semantic burden: we may find corners in a tree or in a rock formation, but we tend not to think of natural objects as having corners.

In this chapter[2] we present a corner detector that uses the principles of Chapter 2 instantiated in the same way as in the compass operator. The only difference in the model is that we no longer have half the wedges of the circle (recall Figure 3.3) on each side. An extra parameter β is introduced to measure the angle subtended by the corner. The range of θ must now be $[0, 360)$ since the two radii that form the sides of the corner no longer have odd symmetry. We refer to the two sides as "clockwise" and "counterclockwise," and θ refers to the orientation of the clockwise side. Figure 4.1 illustrates the parameters.

Otherwise, the algorithm is similar to that for edge detection. A circular neighborhood is clustered and two color signatures are formed. The EMD measures the dissimilarity between signatures. The decision process for corners is more complicated because it must refer to edge information at nearby points. The resulting algorithm finds corners in situations as difficult as those handled by the compass operator. We believe it to be the first detector that can handle either color or most arbitrary textures.

This chapter details the differences necessary to detect corners instead of edges.

[2]Much of the work in this chapter has been published in [90].

After surveying the literature, we consider two specific problems: the EMD when two signatures have unequal amounts of mass and how corners are detected from the operator's output.

4.1 A Review of Corner Detectors

We start by reviewing many of the models that others have used for their corner detectors. Thetwo main approaches are indirect methods, in which corners are found from the output of an edge detector, and direct methods, where corners are found from the image data itself. We exclude methods in which the input is already a single, closed, parameterized curve.

The earliest indirect method comes from Perkins and Binford [75] and is also the first published corner detection paper. It used straight line segments found by the Hueckel operator and extended them to meet if model criteria were met. Another early entry by Sebok *et al.* [94] looked for overlapping connected components of edge points in adjacent rows of an image to find corners.

A unique approach to the problem was formed by Han *et al.* [35], who measured the distance from edge points to a reference line segment and looked for extrema. If the set of edge points considered contained more than two line segments, the process was repeated recursively.

Other models in this category include Medioni and Yasumoto's use of cubic splines [57], Bell and Pau's use of logic programming [7], Matas and Kittler's use of probabilistic relaxation [56], a wavelet approach by Lee *et al.* [48], simulated annealing by Xie *et al.* [120], and least squares fitting by Ji and Haralick [42].

One of the most recent proposals in the literature is notable because it tracks corners through different scales. Mokhtarian and Suomela [62] find initial corner points at orientation discontinuities in a Canny edge map. They find corners by measuring curvature at a large scale and tracking the corners through small scales to localize them.

Although these methods have achieved some success, there are serious arguments

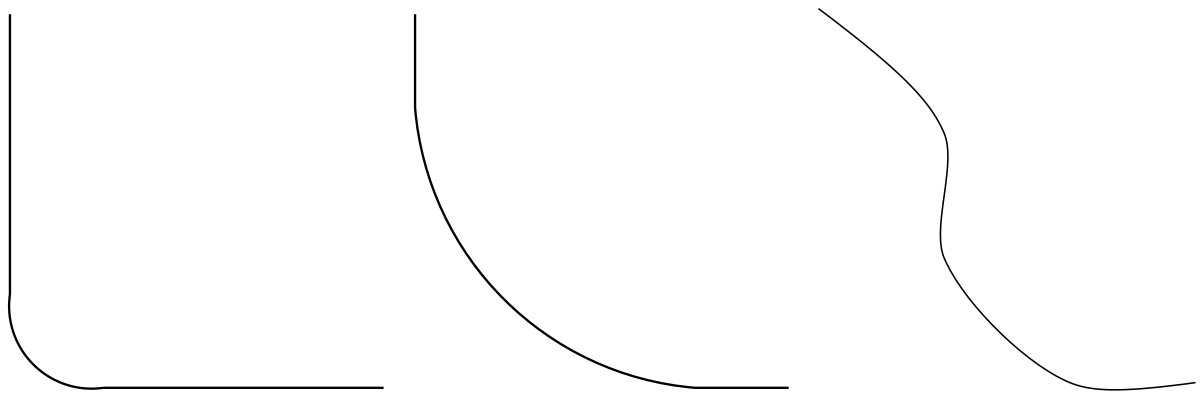


Figure 4.2: Indirect methods rely on edge detection, but edges are "rounded off" near corners. The third drawing implies that we need straight edges to perceive a corner.

against using them. Foremost is the fact that edge detectors are generally not designed to find corners, and whatever biases the detector comes with will affect later processing. A more detailed argument is given by Deriche and Giraudon [23], who point out that first-derivative-based edge detectors (a group that includes the compass operator) have a "rounding off" effect at corners, and the severity of the effect is dependent on scale (see Figure 4.2). The implication is that indirect methods can work only when the scale of the edge detector is very small.

Their answer was to combine indirect methods with direct methods. The Marr-Hildreth edge detector, which is based on second derivatives, does not suffer from rounding off. They also used Beaudet's measure [6], the determinant of the Hessian ($I_{xy}^2 - I_{xx}I_{yy}$), at two different scales. Since the peaks in this measure fall on the bisector line of the corner, the corner's location is easily revealed.

Beaudet's measure was one of the first purely direct methods, but many more have been tried. Paler *et al.* [73] used median filtering and sorted the intensities in a neighborhood to find corners. Rangarajan *et al.* [80] developed an optimal corner detector in the spirit of Canny that required 12 convolution masks. Wu and Rosenfeld [118] used discontinuities in the marginal distributions of grey levels in an image to find initial corner candidates. Checking for locally uniform intensity distributions weeded out false positives.

Mehrotra *et al.* [58] were one of a few to look for “half-edges,” found by placing the center of a filter at a corner point and using only two of the four quadrants to look for edges, resulting in corner responses that were as strong as edges. Rohr [84] used parametric models to fit edges and intensity regions together to find corners. Cooper *et al.* [18] noted that translating a window along an edge produces little change in the image data, but the change is high in all directions at a corner.

Computation time has recently become important for real-time tracking applications. Seeger and Seeger [95], Wang and Brady [114], and Trajkovic and Hedley [109] have all modified other methods to get robust corner detection without much computation.

Other direct methods have included using wavelets (Chen *et al.* [16]), Bayesian approaches and fuzzy logic (Lee and Bien [49]), energy minimization (Parida *et al.* [74]), gradient vector fields (Luo *et al.* [51]), morphological operators (Laganiere [46]), and a measure of “unidirectionality” (Chabat *et al.* [13]).

The work of Alvarez and Morales [5] measured curvature and tracked it through an affine morphological scale space. It is noteworthy because it models neighborhoods as level sets, making it the only corner detector in this survey that does not invoke the constancy assumption. The level set assumption restricts all intensities in one neighborhood to be darker or brighter than adjacent ones. This model is not as general as distributions, however, and it is not clear how level sets would apply to color, which has no similar total ordering of values.

Out of the many corner detectors that have been proposed, a few have become standards because they are conceptually interesting and easy to implement. The second one (after Beaudet) was that of Kitchen and Rosenfeld [44]. They used second derivatives of the image to calculate the rate of change of the gradient direction along an edge. Second derivatives can be unstable, however, so Singh and Shneier [100] combined it with another gradient-based method to achieve robustness.

More popular has been the Plessey operator, apparently published first by Harris and Stephens [36]. A 2×2 autocorrelation matrix is formed from the outer product of the gradient, and corners are detected where both eigenvalues are high. Noble [71] analyzed it thoroughly and found it worked well only at L-corners; he proposed a method that was able to find junctions as well. Tomasi and Kanade [106] re-derived the original formulation from the optical flow equation and used those corners as feature points for tracking. Zheng *et al.* [125] developed a faster version.

Two different issues come into play in the analysis of corner detectors. The first is whether the corner model is applied to the image data itself or to the gradient; it is similar to the dichotomy between indirect and direct methods. Therefore, a similar argument can be made that, since the gradient finds the single direction in which intensity is changing fastest, it is not as applicable to corners, which require information in at least two directions. Gradient methods are usually much faster than so-called "template methods," and this is why they remain popular.

The second issue is whether a detector finds only L-corners or whether it finds junctions as well. Many of the corner detectors just mentioned are really junction detectors, with L-corners forming a restricted case. A junction is a place where multiple corners meet, and it is advantageous to recognize this fact, as we discussed in the last chapter.

Given all the detectors that have been published, why should we propose another? The answers are exactly those used for the compass operator: large scales, color, and texture. None of the methods published considers color explicitly, though indirect methods could use the edge map produced by a color edge detector as their input. Even multidimensional gradient methods could be applied to Kitchen and Rosenfeld's measure. Template methods would have a problem fusing possibly different answers about corners in different color components together to form a final answer. In any case, we believe that models for corner detection should be as applicable to color images as they are to greyscale.

The second reason is that most models invoke the constancy assumption; there have been no analogues to the Wang-Binford operator, for instance, to detect corners in the presence of shading. We want to find corners in images where neighborhoods contain correlated variations due to shading or texture rather than uncorrelated noise variations.

Compared to other direct methods, our corner detector is an interesting combination of template and gradient methods. All hypothesized corners in an image are tested, but the difference between neighborhoods is computed by using the EMD to find the orientation that maximizes the distance between neighborhoods, and we have already seen that this creates a pair of numbers similar in function to a gradient.

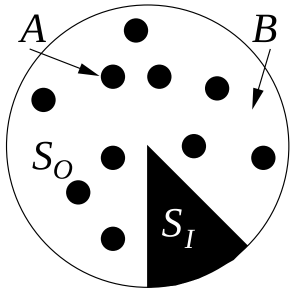
The one drawback of our detector compared with many others is that it does not detect junctions. The EMD does not directly lend itself to measuring distances between three or more neighborhoods. Other recent junction detectors also use wedges [74, 122], basing their computation on differences between wedges, but since we do not assume that wedges are constant or that neighborhoods are homogeneous, taking the EMD between adjacent wedges would produce too many false positives. Instead, we believe that the concept of abnormality is better suited for finding junctions than our corner detector.

4.2 Partial vs. Normalized EMD

Like the compass operator, the corner detector finds the EMD between the color signatures of two adjacent neighborhoods inside a circle. The biggest difference is that S_O, the signature for the neighborhood outside the corner, has more mass than S_I, the signature for the neighborhood inside the corner. We must choose between allowing unequal-mass distributions in the EMD or deciding that two neighborhoods will always have the same mass no matter how few pixels are in S_I. The first leads to the "partial" EMD, where correspondences are found between S_I and the subset of S_O that results in the minimum amount of work[3]. The second leads to the "normalized" EMD where we set the total mass of S_I and of S_O equal to each other. In either case, we set the mass of S_I to be 1 so that the EMD continues to lie in the range $[0, 1]$.

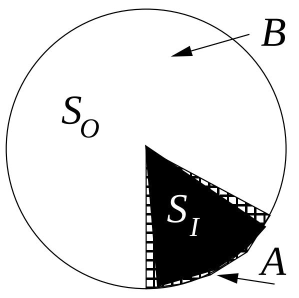
Each type of EMD has advantages over the other in certain situations. In Figure 4.3 we have one color inside a 45° corner and a polka-dotted texture outside it. Even though the polka dots take up only a fraction of S_O, the total mass of color B is equal to that in S_I, resulting in no corner. The normalized EMD, however, detects

[3]This is accomplished by adding a dummy point mass to S_I equal to the difference in mass between S_I and S_O. The unit transportation cost from this point mass to all point masses in S_O is zero.



	partial			normalized	
	A	B		A	B
S_I	1	0	S_I	1	0
S_O	1	6	S_O	1/7	6/7
EMD: 0			EMD: 6/7		

Figure 4.3: The normalized EMD can find corners that the partial EMD cannot.



30°	partial			normalized	
	A	B		A	B
S_I	1	0	S_I	1	0
S_O	0.8	10.2	S_O	0.07	0.93
EMD: 0.2			**EMD: 0.93**		

60°	partial			normalized	
	A	B		A	B
S_I	0.9	0.1	S_I	0.9	0.1
S_O	0	5	S_O	0	1
EMD: 0.9			EMD: 0.9		

Figure 4.4: The partial EMD can more accurately describe a corner. The partial EMD has a stronger response at 60°, while the normalized EMD prefers a 30° corner.

this corner easily.

Figure 4.4 shows a situation where the partial EMD describes a corner more accurately than the normalized EMD. A 60° corner consists entirely of color A except for the two edges, which contain some pixels of color B. If 10% of the pixels inside the corner have color B, then the values of the EMD are those shown in the accompanying table. Both types of EMD detect a corner, but the normalized EMD estimates β to be 30°, while the partial EMD correctly estimates β to be 60°. Figure 4.5 illustrates this difference on real image data. The corner found by the partial EMD runs along the edges, while the other does not. The reason for this behavior is that each pixel in S_I has much more relative mass than it would if it were in S_O. The effect of edge pixels on the normalized EMD is minimized if they are excluded from the inside signature.

We have chosen the partial EMD for our experiments. We may suffer some false

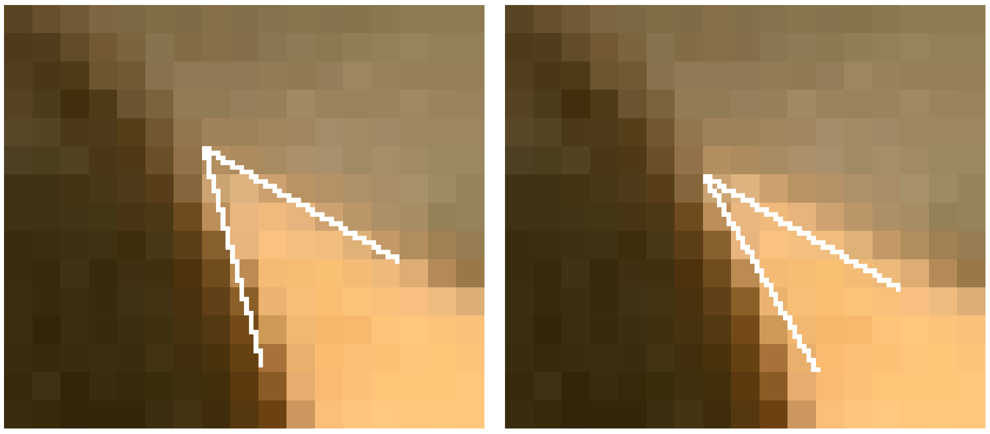


Figure 4.5: Both the partial EMD and the normalized EMD detect the corner, but the vertex is in different places and the normalized EMD underestimates the size. The sides of the partial EMD corner run more along the edges in the image, and this advantage is why we have chosen the partial EMD for the corner detector.

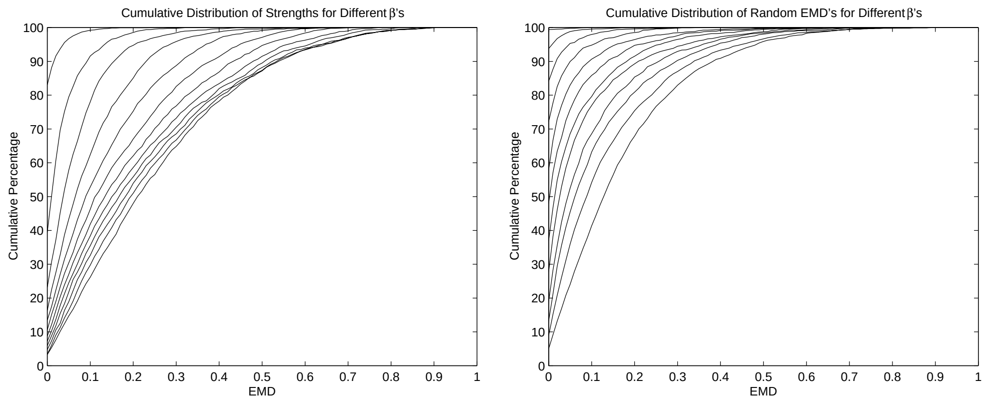
negative corners when β is very small, but this is not likely to happen often, and the benefit of describing corners more accurately outweighs this disadvantage. One other important side effect of the partial EMD is that we cannot create a faster version of the corner detector for greyscale images as we did for the compass operator. The GreyEMD routine (recall Table 3.1) works only on signatures of equal mass.

4.3 The Edge Model

The result of applying the corner detector to an image is essentially a four-dimensional (x, y, θ, β) array of EMD values, and corners are relative maxima above a minimum strength in this array. However, there are some complicating factors due to the fact that the conditions for corners are more restrictive than for edges.

In the first place, a corner is a response to a phenomena that takes place over a relatively large portion of the image, so checking only the nearest neighbors in the array will produce too many corners. It is quite possible to get strong maxima all responding to the same corner phenomenon but with significantly different parameters.

In the second place, we cannot directly compare EMD responses from parameter



(a) Maximum EMD (Strength) (b) Random EMD

Figure 4.6: Cumulative distributions for the response of the corner detector. Each curve represents a different value of β, with $\beta = 15°$ closest to the upper left and $\beta = 180°$ closest to the lower right. (a) Distribution of maximum EMDs (strengths) at each point. (b) Distribution of randomly chosen EMDs at each point. The significant differences in the curves precludes us from comparing corner responses that differ in β.

values that differ in β. Changing the size of the corner also changes the statistics of the EMDs that are generated. Figure 4.6 shows the result of an experiment that evaluated the corner detector for different values of β at 2,000 randomly selected image points over a database of 20,000 images. Figure 4.6(a) shows the cumulative distribution of the strengths at each point, and Figure 4.6(b) shows the cumulative distribution of EMDs from randomly chosen orientations at each point.

As we vary β in 15° increments from 15° to 180°, the curves move from the upper left corner towards the lower right. Note that the changes in the distribution of random EMD's is fairly even as β varies, but the amount of change in the distribution of strengths is inversely proportional to the value of β. As a result, comparing responses with different corner sizes is not advisable. Even finding the initial corner candidates requires varying the threshold linearly with β so that small corners can be found without introducing too many false responses for larger corners.

Therefore, we must use a more complicated procedure to winnow the set of maxima over x, y, and θ for every value of β to the actual corners. We do this by comparing the parameters of each corner with edge information provided by the compass operator.

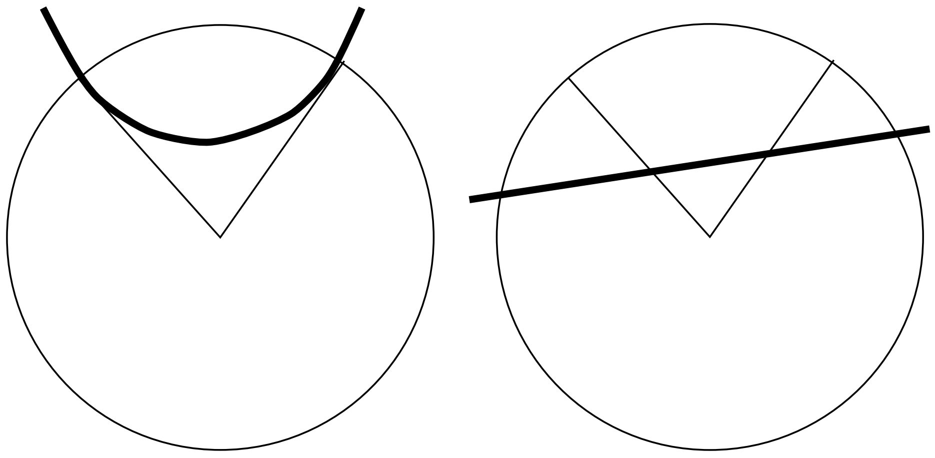


Figure 4.7: True corners are aligned with edges at the endpoints. The edge is rounded off in the middle, where the response weakens. False positives occur due to inhomogeneities on either side of an edge.

If multiple responses to a corner remain, we select one according to a heuristic. These steps are covered in the next two sections.

4.3.1 Testing Corner Candidates

Figure 4.7 shows how false positives are distinguished from true corners. A true corner exists because two edges meet at a point, in which case we expect that the orientations of the edges at the endpoints of the corner match those predicted by the corner detector. Furthermore, the rounding off noticed by Deriche and Giraudon means that there should be a weak edge response between the two sides of the corner.

On the other hand, a false positive is sometimes generated when small inhomogeneities on either side of an edge cause a spurious response. In this case, there will be no supporting edge information in the three places that we have predicted.

For each corner we compute θ_C and θ_{CC}, the difference between the orientation of the clockwise and counterclockwise sides of the corner, respectively, and the edge orientation at the endpoint of each side. The fit to the model is expressed as

$$P = \cos\theta_C + \cos\theta_{CC},$$

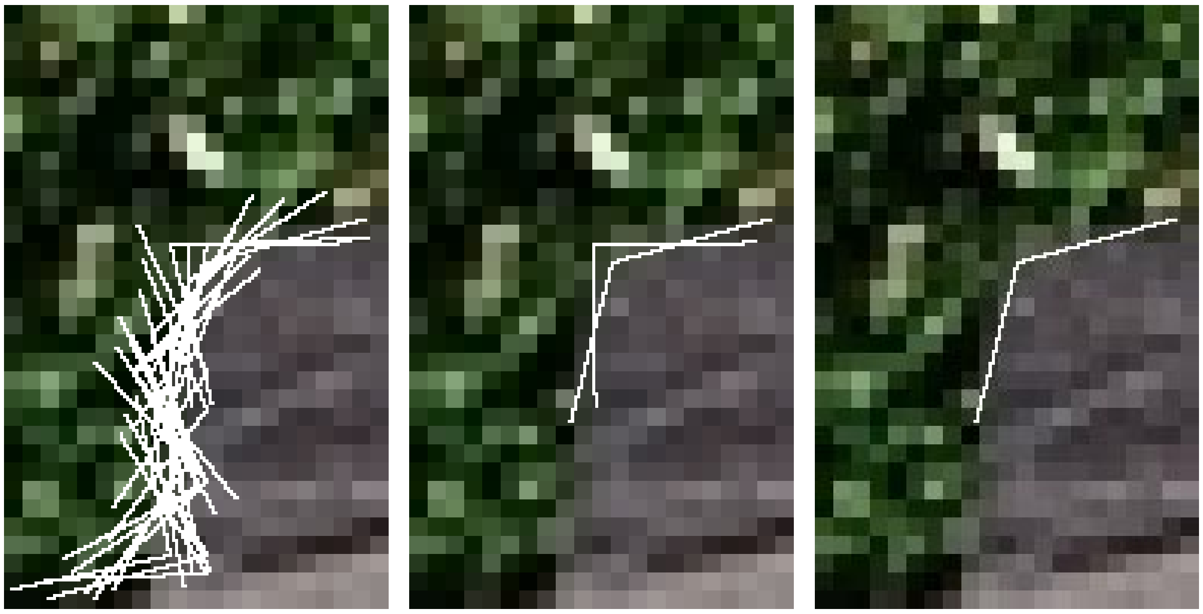


Figure 4.8: Detecting corners from an initial set of candidates involves applying an edge model followed by pruning multiple responses (if they exist).

where P lies between 0 and 2. We threshold P at 1.97.

Also, where the edge crosses the corner's axis of symmetry, the projection of the edge response vector onto the line normal to this axis must be weaker than the corner response. We check responses on a small interval along this axis centered at a point $3\sigma(\sec\frac{\beta}{2} - \tan\frac{\beta}{2})$ pixels away from the corner[4]. This quantity is the distance from the corner point to the circumference of an imaginary circle tangent to the sides of the corner at its endpoints.

Figure 4.8 shows the initial corner candidates found by thresholding the relative maxima. Most of these responses are false positives and applying the edge model greatly reduces their number.

[4]Recall that the radius of the operator is 3σ.

4.3.2 Pruning Multiple Responses

Ideally, only one response to each corner phenomenon will remain after this step, but Figure 4.8 shows that this is not always the case. We offer a heuristic to choose one when this happens, though perhaps multiple responses could be combined in some way.

First, we must decide when two corner candidates are responding to the same actual corner. In Figure 4.8 this is trivial, but the general question admits no equally general solution. We define two corners as being "close enough" if the corner points are within $9\sigma/4$ pixels of each other and one of two conditions is true: (1) either the two clockwise or the two counterclockwise orientations differ by no more than $10°$, or (2) the sum of these differences is no more than $40°$. These conditions group "nested" corners while preserving multiple corners near junctions.

An ambiguity arises when corner X is close to corners Y and Z, but Y and Z are not close to each other. If our notion of "closeness" is global, then the order in which we examine corners affects the final output. Since this is unacceptable, we compute the transitive closure of "closeness," that is, X, Y, and Z will all become part of the same set. It is theoretically possible that corners in distant parts of the image could become part of the same set; in practice, however, the application of the edge model removes enough candidates to prevent this.

Once we have computed the transitive closure, we select the member of each set that maximizes the expression $2C + P + E$, where C is the corner strength, P is the degree of orientation match described earlier, and E is the sum of the edge strengths at the endpoints of the two sides of the corner. C is doubled so that each term contributes equally. The final corner of our example is shown in the third column of Figure 4.8.

4.4 Results

In this section we present results on a variety of image patches in order to convey the versatility of the operator. All these results were computed with the partial EMD

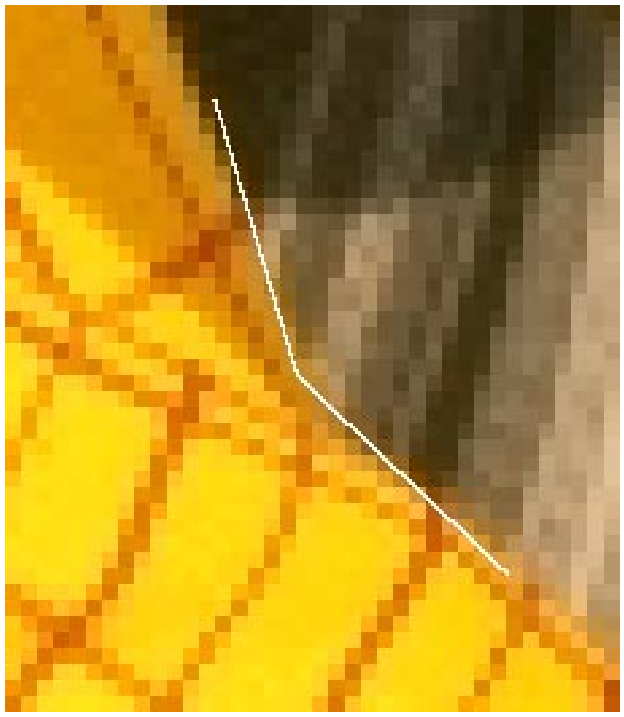


Figure 4.9: Corner detector applied to two fabrics. Note the heterogeneity of each texture as well as the existence of shadows.

and the same set of thresholds. The lengths of the sides of the corners drawn in the images are equal to the radius of the operator. Also, we have restricted β to the range $[30, 150]$ because $15°$ and $165°$ corners are unreliable; the small corners do not contain enough pixels to get an accurate representation, and the large corners are nearly impossible to distinguish from edges. However, this does appear to be wider than the range of most other detectors.

Figure 4.9 shows one fabric occluding another. Although each contains texture that varies greatly in color and has regions in partial shadow, the corner is correctly detected. Figure 4.10 is more complicated because three textures are involved: trees, illuminated rock, and rock in deep shadow. Five corners separate the regions and serendipitously form most of the boundary of the illuminated rock.

Turning our attention to junctions, we reiterate that the corner detector is not able to model them explicitly. Figure 4.11 shows three regions, one significantly textured, coming together at a point. Corners at three different scales are shown. At the two

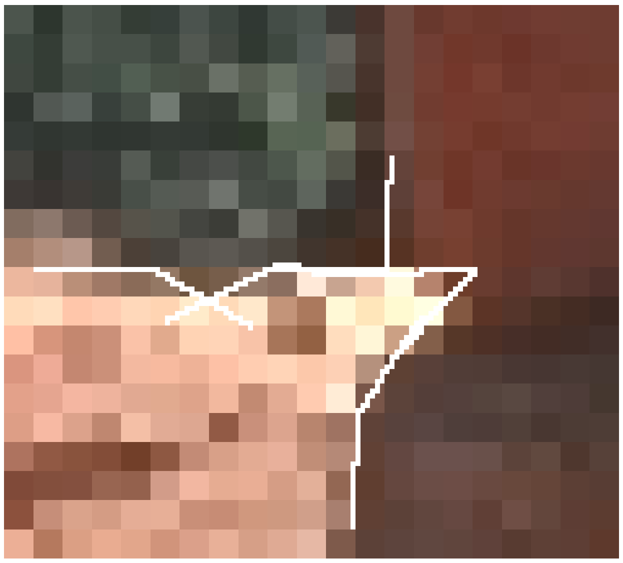


Figure 4.10: Corner detector applied to three regions of natural texture.

smaller scales it is clear that even 150° corners are difficult to distinguish from edges. The fact that these false positives disappear at the largest scale provides evidence that scale selection is no less important for corner detection than for edge detection.

Even if we ignore these corners we do not quite capture the information at the junction correctly. The edges which form this junction are curved, and though two corners in the junction are found at all scales (the third is bigger than 150°), their vertices do not coincide, and the other corner detected in that area is arguably a curved edge. This junction could be detected using a procedure similar to the one that determined if multiple corners are responding to the same part of the image.

4.5 More Results: Corners and Edges

Figure 4.12 illustrates a more conceptual difficulty with the corner detector. The left image shows corners, including those as large as 165°, and the other shows edges extracted using the compass operator. There is a high degree of overlap between the features as they are drawn on this image since most of the corners have high values of β. It begs the question of whether corners are relevant in images that do not consist

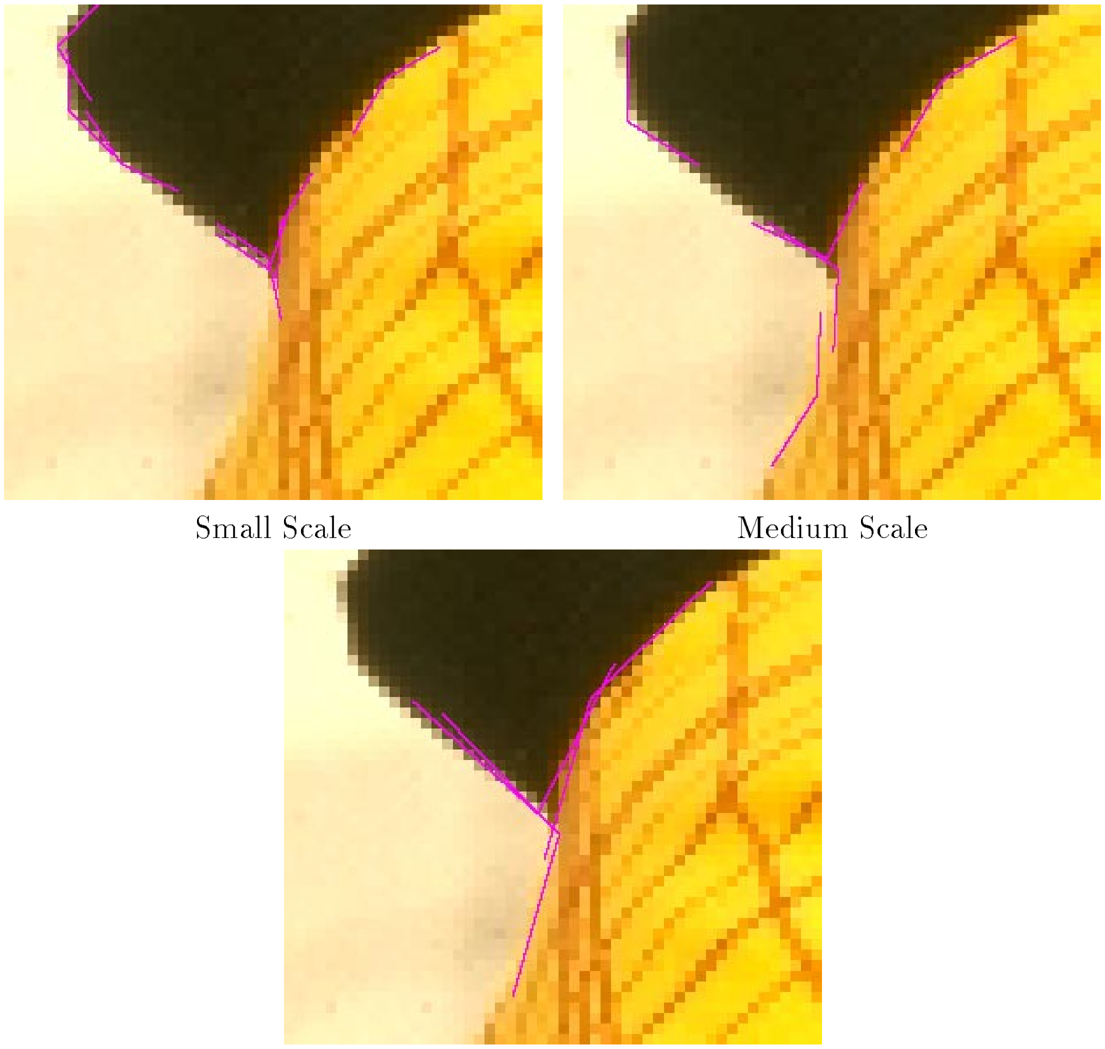


Figure 4.11: Corner detector applied to a junction at three scales. The difficulties encountered at curved edges is apparent. One corner of the junction is greater than 150° and is unrecoverable.

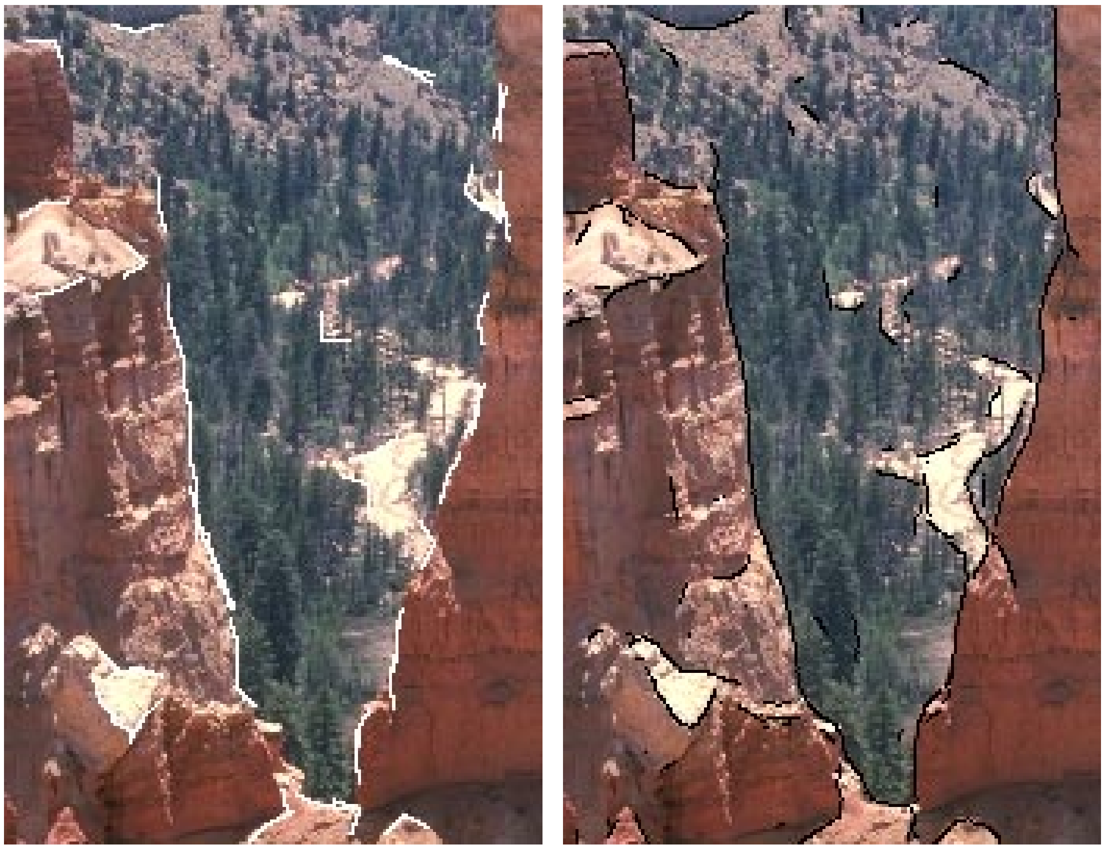


Figure 4.12: Corner detector and edge detector applied to an image of a canyon. Note the high amount of overlap between the two sets of results. 165° corners have been included.

mostly of man-made objects.

Even more disconcerting is the fact that we have developed two separate algorithms for detecting features from what is arguably the same data. Edge information is simply that obtained by setting β to 180°, while corners cannot be reliably detected without edge information. The two features are complementary and both are needed for accurate boundary representation.

Figure 4.13 illustrates this last point. The edges found by the compass operator outline the boundary of this rock, but because the rock is convex in shape, the edges are sometimes a few pixels inside the rock. The two corners, however, are well localized, and combining this information with the edges would produce a more

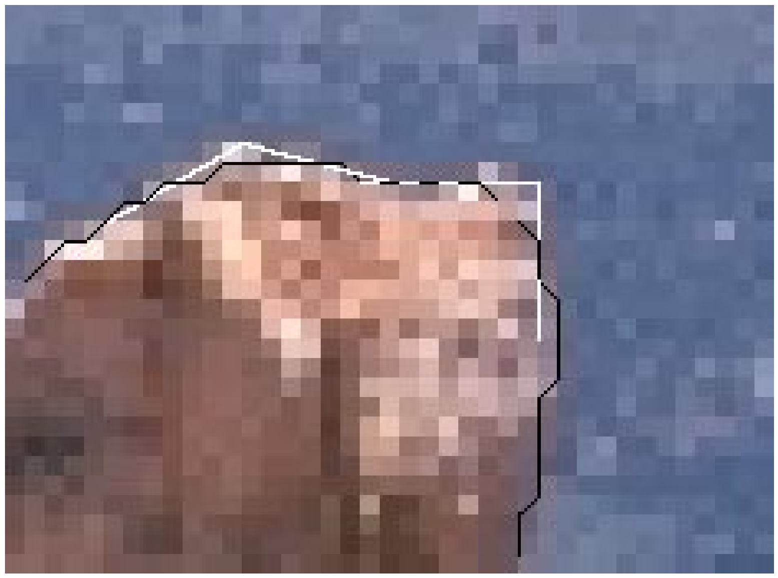


Figure 4.13: Corner and edge information is complementary. Is it possible to extract a boundary that contains both at the same time?

accurate boundary.

One way this could be done is by an energy minimization method. The corners can be thought of as basins of attraction that perturb the edges. A better method would be to use the entire 4-D array of data to extract a more accurate boundary directly. Following an edge would require decreasing β from $180°$ to the value at the endpoint of the corner and then changing direction and increasing β back to $180°$.

4.6 Conclusions

The models used to find corners have generally lagged behind models for edges. None of the corner detectors surveyed dealt explicitly with shading, unless they are indirect methods that can be modified to use, for instance, the Wang-Binford edge detector. The corner detector we described here brings the models for corner and edge detection to the same point.

Extracting corners is more difficult than extracting edges. The fewer pixels inside a corner, the more susceptible the computation is to noise and sampling effects. The problems of stability mentioned in the previous chapter are even more important here. Combining corners and edges in complex regions containing textures remains a long-term goal. However, the framework here is an important step because it extends the set of images on which reasonable results can be obtained.

Chapter 5

Alpha Estimation

The advances in edge and corner detection discussed in the previous two chapters do not place all image boundaries within the realm of detection. We saw in Chapter 1 that some boundaries are not well represented by edges and corners. In particular, natural objects such as trees, hair, smoke, and water often fail to follow even the compass operator's general model of an edge between two objects.

This chapter[1] examines a model in which a pixel is allowed to belong to two regions whenever its color appears to have been formed by light reflecting off of two separate objects and reaching the same patch on the camera's sensor. The colors and the relative amounts of each can be determined by examining the color distributions of nearby pixels that receive light from only one object. The percentage that a region contributes to the color of a pixel is referred to as its *alpha value*.

Of course, we have little hope of finding out in general whether a pixel is gathering light from one object or two. Since traditional edge models break down, we require additional input in the form of a segmentation of an image into regions that are definitely an object versus regions that contain a boundary. These boundary regions need not conform exactly to a boundary because extraneous pixels can be reclassified as belonging to an object.

The following sections explain the details surrounding this procedure: the history

[1]Much of this work was published in [91].

and rationale of previous approaches, the constraints on specifying boundary and object regions, and the estimation of alpha values and the "unmixed" colors that formed the color of a boundary pixel. Results demonstrate the algorithm's effectiveness on boundaries that cannot otherwise be adequately captured.

5.1 The History of Alpha

The concept of alpha was invented separately by three different communities: computer graphics, film and television, and remote sensing. The assumptions, methods, and applications of the three groups are quite different, however, so it is worthwhile to examine them separately.

The original application of alpha in computer graphics was for *soft filling*, or changing the color of an antialiased region such as an edge[2]. Fishkin and Barsky [31] published the most comprehensive technique for soft filling when alpha values were unknown but the original foreground and background colors were known exactly. The color of an edge pixel was presumed to fall into the vector subspace in color space spanned by the original colors. Alpha could be computed and used to recolor the pixel with new colors. Due to round-off error, a pixel's color may not lie inside the subspace, but the error is expected to be negligible. This technique works for up to four colors between the two objects [32].

At the same time, Porter and Duff [78] developed an algebra for compositing images containing objects that had been rendered separately and were now to be overlapped. Given the geometry of the boundaries involved and an alpha value for each pixel, they could compute the colors of pixels where both objects were visible.

The passage of time has brought many in the graphics community to the realization that building 3-D models of most natural objects is not feasible. Consequently, image-based rendering, where models are sampled from images and rendered under other conditions such as viewpoint or illumination changes, has become an important research topic. Estimating alpha from nothing more than an array of pixel colors is

[2]Translucent objects are modeled using *opacity*, which is different from alpha in that alpha represents the percentage of a pixel covered by an object.

a pressing need for accurate modeling of object boundaries.

For a long time now the film and television industries were already using *blue screen matting* to perform image-based rendering. An actor could be filmed against a blue (or other color) screen, after which a matte could be extracted and the blue screen replaced. Smith and Blinn [101] give an excellent overview of previous work on this *matting problem*, crediting Petro Vlahos for his many patents. His inventions allowed an operator to adjust parameters until the optimal matte was extracted while also solving problems of interobject reflection and shadows.

Smith and Blinn analyzed the matting problem in great detail, noting that a unique solution exists only for the easiest cases. They showed that a unique solution can be found in the general case if the foreground object is filmed against two backgrounds that differ in every pixel. However, this approach is useful only in studios where non-moving objects can be photographed twice. They provided bounds on alpha for the general case.

Mitsunaga *et al.* [59] developed a system for estimating alpha that assumed that the gradient of alpha across a boundary is proportional to the multidimensional gradient magnitude. Projecting image gradient vectors onto a reference vector connecting the average color of the foreground and background increased the signal-to-noise ratio. Alpha values for hair and water, however, do not follow this assumption.

Finally, the remote sensing community has long been interested in unmixing pixels because each pixel from a satellite image covers many square meters and therefore many different substances. A simple yet effective implementation of this idea came from Adams *et al.* [3], who deduced the composition of rocks and soil in an image of the Martian surface and estimated the amount of each substance everywhere in the image while also accounting for illumination effects.

Such techniques usually involve much information not present in the image, however, such as laboratory reference spectra of materials, heuristics for ranking candidates, and other analyses of the data. The problem we consider uses only the image data. Furthermore, the material classes typically have unimodal distributions, while color distributions for an object are less constrained, and the two unmixed colors at each pixel need not even be modal values.

The closest related problem in computer vision is blur estimation, in which alpha values are implicitly assumed to be constrained roughly to a 2-D sigmoid across the edge. Elder and Zucker [30] use this model to perform local scale control in detecting edges. Assuming that a boundary between two regions consists only of an edge corrupted by blurring is applicable to a wide range of images and imposes constraints on the alpha values depending on their position in the image. Often, though, the overlap of natural objects does not follow such constraints.

5.2 Specifying Object and Boundary Regions

Since the algorithm we present here is more a tool than a system, the user must specify more than the input image. We restrict our attention mainly to images in which there are only two regions, foreground and background. Such an image must be partitioned into not two but three regions, the third being the boundary region. This section details two alternatives for specifying these regions by using the tree example in Figure 5.1.

A chain of pixels separating two objects can be dilated to form a boundary region. This chain can be constructed from the edges found by an edge detector, the boundary found by a region segmentation algorithm, a hand-drawn boundary using a paint program, or a boundary-finding tool such as Intelligent Scissors [64]. In our example, the boundary was specified using the compass operator at a very coarse scale ($\sigma = 16$) and dilated $\lceil \sigma \rceil$ times. The compass operator produced a few gaps and other anomalies, but the dilation compensated for them.

Also, a paint program can be used to specify parts of the image as "pure," or consisting only of pixels belonging to one of the two objects. In our example magenta pixels mark the sky and yellow pixels delineate the tree (the two colors should not exist in the image already, of course).

Most image boundaries are more effectively captured by one method than the other. The object specification method is superior in this example because many pixels well inside the outer boundary of the tree contain blue. The boundary specification method is advantageous in images where the color distribution in one or

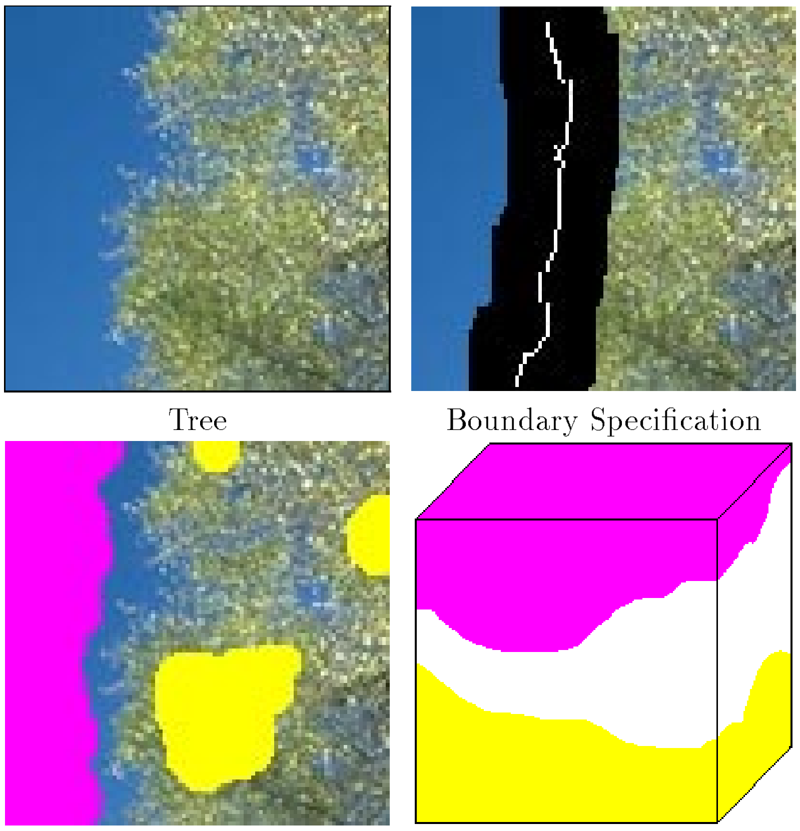


Figure 5.1: Computing alpha values along a boundary requires a specification of the object regions or of the boundary region. Object specification is more flexible and preferable for this example, but boundary specification is useful when the color distribution of an object changes spatially.

both object regions changes as we move along the boundary, however. If we have a boundary, we can form local correspondences; a pixel in the boundary region can be judged as a combination only of nearby colors in the image, which will improve the alpha estimate. Otherwise, we can use only global distributions for entire objects, and they may not be separable in color space.

If an image contained multiple objects, all of which we would like to segment, the only complication would be labeling each pixel in the boundary region with the two objects responsible for its color. Here, the boundary specification method would make this task simpler since the pixel chains can be segregated by the junctions. Because the object specification method has no restrictions on its topology, neither color similarity nor spatial proximity would provide adequate pixel labels in all situations. It is perhaps easier to perform the algorithm multiple times, once for each object.

The purpose of specifying object and boundary regions is twofold: it labels each pixel for proper use in the computation and it partitions color space. The final illustration of Figure 5.1 is a conceptualization of the colors of the tree and sky mapped into color space. Pixels from the boundary region lying in these two regions of color space will be assigned alpha values of 0 or 1, while pixels whose colors values are in between will be assigned fractional alpha values. Thus, the specification of the boundary region need not be precise so long as it actually contains the boundary, and so long as it leaves enough of a color in an object region that it can be represented by one or more clusters.

5.3 Estimating Alpha

Our algorithm for alpha estimation for the most basic cases is similar to the algorithms of Fishkin and Barsky. The complexities come from the fact that we have noise and other sources of variance in the data, potentially many more than four colors, and unmixed colors that need not correspond to modes of the color distribution. This section describes the mechanisms for dealing with this complexity.

5.3.1 Building a Manifold in Color Space

Alpha values are measured along a manifold connecting the "frontiers" of each object's color distribution. Once again each distribution is represented as a color signature found through vector quantization in CIE-Lab. We denote the two distributions as $X = \{(x_j, \mathbf{u}_j, \sigma^2_{\mathbf{u}_j})\}$ $(j = 1, \ldots, M)$ and $Y = \{(y_k, \mathbf{v}_k, \sigma^2_{\mathbf{v}_k})\}$ $(k = 1, \ldots, N)$, where the x_j's and y_k's are percentages of each color specified by $\mathbf{u}_j$ and $\mathbf{v}_k$ respectively. We also add the variance of each cluster to our representation.

One manifold is constructed for each pair of distributions. If the boundary region was initially specified, we must decide which pixels to use for each pair of distributions.

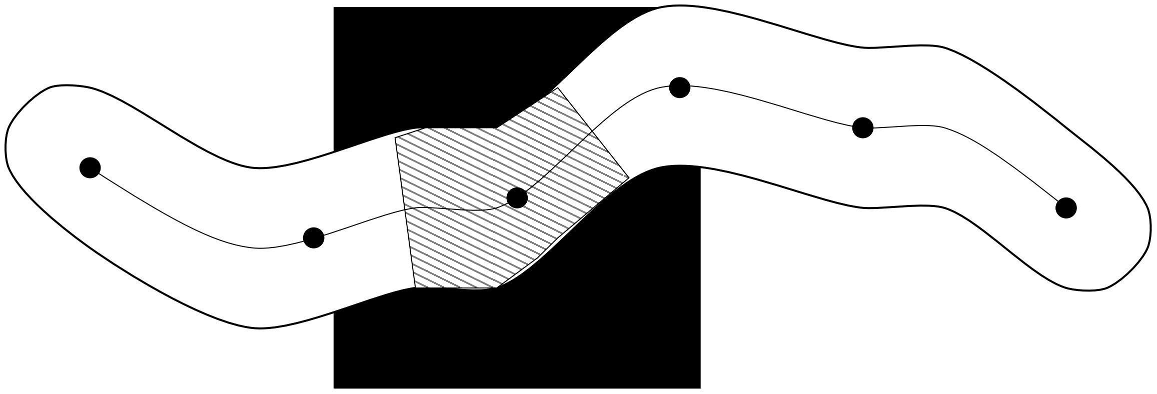


Figure 5.2: When a chain of pixels is specified and dilated to form the boundary region, it is divided into intervals separated by anchor points. The square centered at one of the anchor points shows which pixels from each object region will be clustered to form two local color distributions. All boundary pixels in the shaded region will have their alpha values computed using these two distributions.

We divide the chain of pixels into non-overlapping intervals and define the endpoints of the intervals as *anchor points*. The length of each interval is equal to three times the amount of dilation performed. Since we allow different parts of the chain to be dilated by different amounts to capture the nature of the boundary accurately, the length of the intervals can also differ. The anchor points serve two purposes: (1) each becomes the center of a window defining the pixels in each object region that form the local color distributions, and (2) they divide the boundary region into pieces, where each piece consists of all pixels with a common nearest anchor point. Each piece of the boundary region will use the color distributions specified by the corresponding anchor point for computing alpha. Figure 5.2 illustrates these two functions.

The set of line segments in color space connecting one point mass from each signature can be represented as the Cartesian product $\{1,\ldots,M\} \times \{1,\ldots,N\}$. The manifold is constructed using a subset of this product to construct a flow between the signatures. Rather than a minimum-cost flow constraint, which gives rise to the EMD, we maximize the number of line segments on which a nonzero amount of mass is transported. Doing so assures us that as many boundary region pixels as possible are near a line segment, providing accuracy to the computed alpha values. It is important that the segments do not conflict, however, or ambiguity will enter the computation.

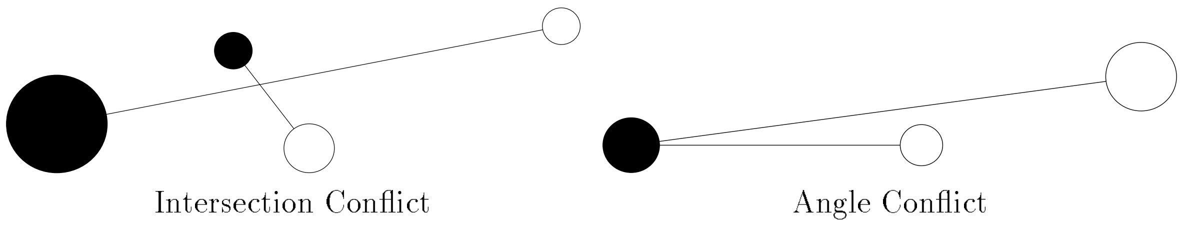


Figure 5.3: Two segments that share no endpoints may have an intersection conflict in three dimensions if they are too close to each other. Two segments that do share an endpoint may have an angle conflict if they are almost collinear. In both cases the true unmixed colors that produced the color of a boundary pixel may be unrecoverable; we reject the longer segments in each case as being unlikely.

Figure 5.3 illustrates two types of conflicts, "intersection" and "angle." A pixel near the intersection of two line segments would be more likely to be a combination of the pair of colors at the endpoints of the shorter segment; the longer one should be rejected as less likely (though not impossible). In three dimensions, two line segments never intersect, so this definition must be amended by declaring an intersection whenever the minimum distance between any pair of points from each segment is smaller than a threshold (set at 5 CIE-Lab units). Intersection conflicts can occur only when the two segments do not share an endpoint.

Angle conflicts, on the other hand, can occur only when the two segments do share an endpoint. When the angle between the segments is small (less than $10°$), the three clusters are almost collinear. This is another source of ambiguity, noticed previously by Smith and Blinn, because we cannot tell which pair of colors actually produced a color in the middle. Again, we choose the smaller segment as being more likely.

The algorithm for computing the set of non-conflicting line segments is shown in Table 5.1, and Figure 5.4 shows an example of its output. It is a greedy algorithm that adds segments to the set if they do not conflict with segments that have already been accepted. Note that it is possible that one or more clusters will not be an endpoint of any accepted line segment; any such clusters are excluded from the rest of the computation and the x_j's or y_k's are renormalized. The n line segments chosen help define the manifold. Each segment is represented as an ordered pair (j, k), and we define two functions, $J(i) = j$ and $K(i) = k$, that return the index into X and Y,

```
Segment-Set BuildManifold(Point-Set $\{u_i\}$ $(1 \leq i \leq M)$, Point-Set $\{v_j\}$ $(1 \leq j \leq N)$)

$Accepted$ = $\{\}$
for $i$ = 1 to $M$
    for $j$ = 1 to $N$
        $d_{ij}$ = distance($u_i, v_j$)

$\{l_1, \ldots, l_{MN}\}$ = line segments sorted by increasing $d_{ij}$

for $i$ = 1 to $MN$
    for (each segment $s_j$ in $Accepted$)
        if ($J(l_i)$==$J(s_j)$) /* $J()$ returns index into $X$ */
            $\theta$ = $\angle K(l_i) - J(l_i) - K(s_j)$
            if ($\theta < T_1$) /* $T_1 = 10°$ */
                REJECT; break
        elseif ($K(l_i)$==$K(s_j)$) /* $K()$ returns index into $Y$ */
            $\theta$ = $\angle J(l_i) - K(l_i) - J(s_j)$
            if ($\theta < T_1$)
                REJECT; break
        else
            $D$ = minimum distance between $l_i$ and $s_j$
            if ($D < T_2$) /* $T_2 = 5$ units */
                REJECT; break

    if (segment was not rejected)
        $Accepted$ = $Accepted \cup \{l_i\}$

return($Accepted$)
end
```

Table 5.1: The algorithm for choosing a subset of the line segments that connect clusters from two distributions to define a manifold.

respectively, of the clusters marking the endpoints of the segment.

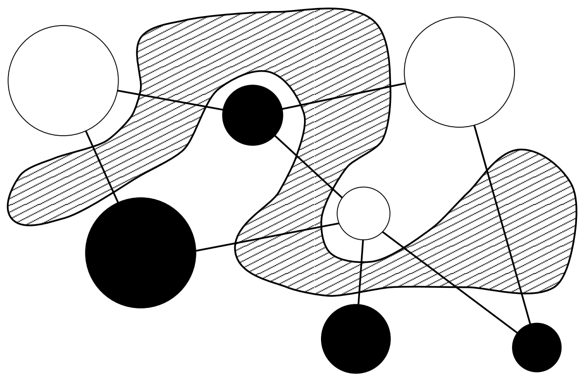


Figure 5.4: An example of a manifold. The line segments are those accepted by the algorithm, and the shaded region indicates where we expect colors of pixels receiving light from two objects to lie in color space.

5.3.2 Computing Alpha and Unmixed Colors

The two signatures X and Y serve as discrete representations of the colors from each object region. We must now form a relationship between these two distributions and an arbitrary pixel Q in color space. This task can be accomplished more naturally if we convert the color signatures to continuous probability distributions. We use a mixture of isotropic Gaussians placed at the points $\mathbf{u}_j$ in X and $\mathbf{v}_k$ in Y, ensuring a simple formulation and nonzero probabilities at all points in color space. The ratio of these two distributions at Q would be one way to estimate α_Q, the alpha value of Q.

However, this simple solution fails for three reasons. The first is that such a ratio yields a non-linear function as a pixel's color moves linearly between the distributions. Since alpha values imply linear combinations, we must have a linear estimation method. The second reason is that such a method says nothing about the unmixed colors; there is no way to use the alpha value to map a color back to specific colors from the objects. The final reason is that numerical measurements at Q when Q is not close to a mode of either distribution are so small as to be numerically meaningless.

We argue that Q is actually drawn from a probability distribution formed as the colors of X are "morphing" into the colors of Y across the boundary. This morphing can be modeled as a linear interpolation between the two probability distributions.

We treat alpha estimation as a maximum likelihood estimation problem: find the interpolated probability density that maximizes the value at Q.

We start by defining a function $f(t)$ that produces a probability distribution for every value of t:

$$f(t) = p_t(\mathbf{c}),\ 0 \leq t \leq 1 \ ,$$

where $\mathbf{c}$ is an arbitrary color vector. The values at $f(0)$ and $f(1)$ are the probability densities corresponding to X and Y, respectively. We model these distributions as mixtures of n isotropic Gaussians (denoted $G_i(\mathbf{c}; \mu_i, \sigma_i^2)$), each being interpolated along one of the n line segments defining the manifold. The distributions for $t = 0, 1$ can be written as:

$$\begin{aligned} p_0(\mathbf{c}) &= \sum_{i=1}^{n} a_i G_i(\mathbf{c}; \mathbf{u}_{J(i)}, \sigma^2_{\mathbf{u}_{J(i)}}) \ , \\ p_1(\mathbf{c}) &= \sum_{i=1}^{n} a_i G_i(\mathbf{c}; \mathbf{v}_{K(i)}, \sigma^2_{\mathbf{v}_{K(i)}}) \ , \end{aligned}$$

where a_i is the amplitude of each Gaussian. The amplitude is proportional to the sizes of the clusters at either endpoint:

$$\begin{aligned} a_i &\propto x_{J(i)} \cdot y_{K(i)} \ , \\ \sum_{i=1}^{n} a_i &= 1 \ . \end{aligned}$$

For any intermediate value of t, the distribution produced by $f(t)$ interpolates the mean and variance of the Gaussians:

$$\begin{aligned} p_t(\mathbf{c}) &= \sum_{i=1}^{n} a_i G_i(\mathbf{c}; \mu_i(t), \sigma_i^2(t)) \ , \text{ where} \\ \mu_i(t) &= (1-t)\mathbf{u}_{J(i)} + t\mathbf{v}_{K(i)} \text{ and} \\ \sigma_i^2(t) &= (1-t)\sigma^2_{\mathbf{u}_{J(i)}} + t\sigma^2_{\mathbf{v}_{K(i)}} \ . \end{aligned}$$

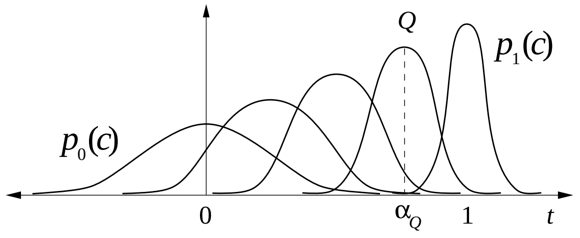


Figure 5.5: An example of interpolation. $p_0(\mathbf{c})$ and $p_1(\mathbf{c})$ are unimodal Gaussian distributions. As t varies, the mean and variance of the interpolated Gaussian also varies. The value of t producing the Gaussian that maximizes the value at Q is the alpha value α_Q.

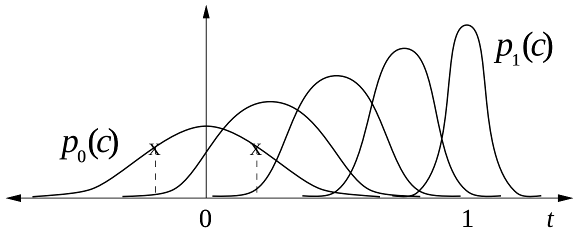


Figure 5.6: The two colors marked by Xs are equally likely to be drawn from $p_0(\mathbf{c})$, but they will have different alpha values if we do not give special treatment to the values 0 and 1.

The expression for computing alpha values then becomes straightforward:

$$\alpha_Q = \arg\max_t f(t)|_Q \quad .$$

In practice, we discretize t at a resolution of 0.01 and evaluate Q for each set of Gaussians produced. Figure 5.5 shows a one-dimensional example of two unimodal distributions and the interpolated Gaussians.

The sole exception to this rule is when Q appears to have been drawn from $p_0(\mathbf{c})$ or $p_1(\mathbf{c})$; Figure 5.6 illustrates this case. Both pixels marked with Xs are equally likely to have been drawn from $p_0(\mathbf{c})$, but the direction of the noise is different, resulting in two different alpha values. If $f(0)|_Q$ or $f(1)|_Q$ is above a threshold, the alpha value of Q is set to 0 or 1 respectively.

Of course, we must also find the unmixed color from each distribution. These colors are not independent because Q must lie on the line segment connecting them.

Our approach is to use the weights provided by each Gaussian component of the distribution to estimate the unmixed color of each side, followed by a small perturbation so that Q divides this new line segment in the proper proportion.

Figure 5.7(a) shows a typical function consisting of the values at Q as t varies. This function is a sum of many smooth functions, each corresponding to one Gaussian as shown in Figure 5.7(b). The component functions are not Gaussians because the variance is a function of t. At the maximum, each of the n Gaussians contributes a weight w_i to $f(\alpha_Q)|_Q$. By mapping each weight back to the endpoints of the segment it came from, we form a weighted average of the colors of each signature to estimate the unmixed colors:

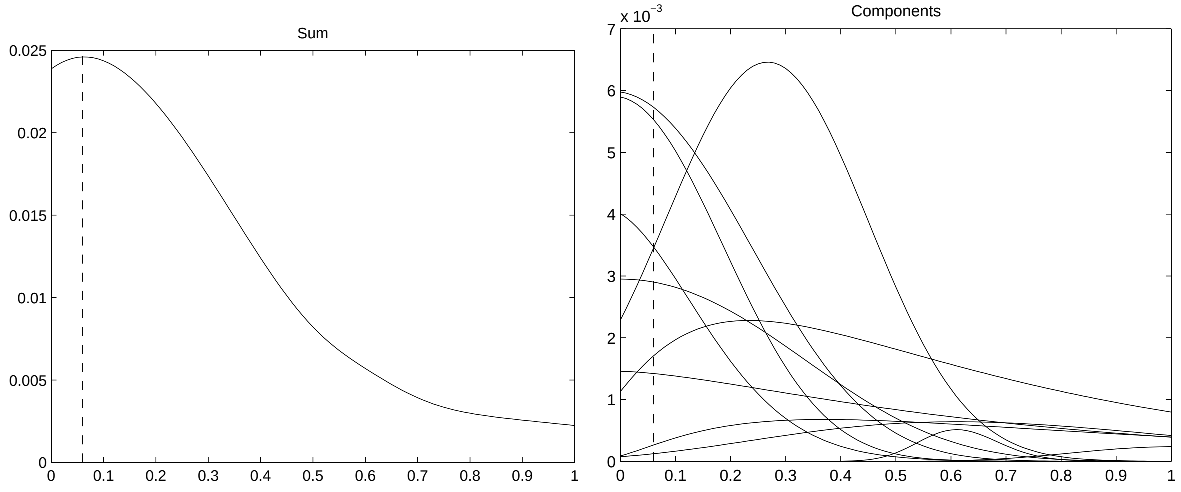
$$\begin{aligned}\hat{\mathbf{u}}_Q &= \frac{\sum_{i=1}^{n} w_i \mathbf{u}_i}{\sum_{i=1}^{n} w_i} \ , \\ \hat{\mathbf{v}}_Q &= \frac{\sum_{i=1}^{n} w_i \mathbf{v}_i}{\sum_{i=1}^{n} w_i} \ .\end{aligned}$$

Normally, no more than a few nearby Gaussians significantly influence the unmixed color of a pixel.

If we denote the point on the line formed by $\hat{\mathbf{u}}_Q$ and $\hat{\mathbf{v}}_Q$ that divides it in the ratio $\alpha : 1 - \alpha$ as Q', we can compute the final colors $\mathbf{u}_Q$ and $\mathbf{v}_Q$ by perturbing $\hat{\mathbf{u}}_Q$ and $\hat{\mathbf{v}}_Q$ by the vector $\overrightarrow{Q'Q}$:

$$\begin{aligned}\mathbf{u}_Q &= \hat{\mathbf{u}}_Q + \overrightarrow{Q'Q} \ , \\ \mathbf{v}_Q &= \hat{\mathbf{v}}_Q + \overrightarrow{Q'Q} \ .\end{aligned}$$

In this way we are assured that combining the two colors at each pixel with the estimated alpha value recreates the original image.



(a) $f(t)|_Q$ (b) Components of $f(t)|_Q$

Figure 5.7: (a) The alpha value α_Q is found by maximizing $f(t)|_Q$ as t varies from 0 to 1. (b) This function is the sum of many smooth functions, each corresponding to a Gaussian interpolated along a line segment and evaluated at Q. Once α_Q is found (0.06 in this case), the component weights at that value are used to compute a weighted average to find the unmixed color from each object.

5.4 Results

This section shows a set of examples on a wide variety of natural objects. Using both methods of specifying object and boundary regions, we create local or global color distributions of object regions to estimate the alpha values of pixels in the boundary region. The signatures we create have no more than 5 clusters each. An object is moved to a destination image by combining unmixed colors with the corresponding pixel colors from the new background according to the computed alpha values.

We follow up Figure 5.1 by placing the tree on a red background using both object specification and boundary specification (see Figure 5.8). This example is essentially blue screen matting. As predicted, the boundary specification method performs poorly because blue is present in both objects. The result using the object specification method is much improved.

Of course, the algorithm is designed to handle more interesting backgrounds than sky. Figure 5.9 also shows a tree branch, but the background consists of trees. A

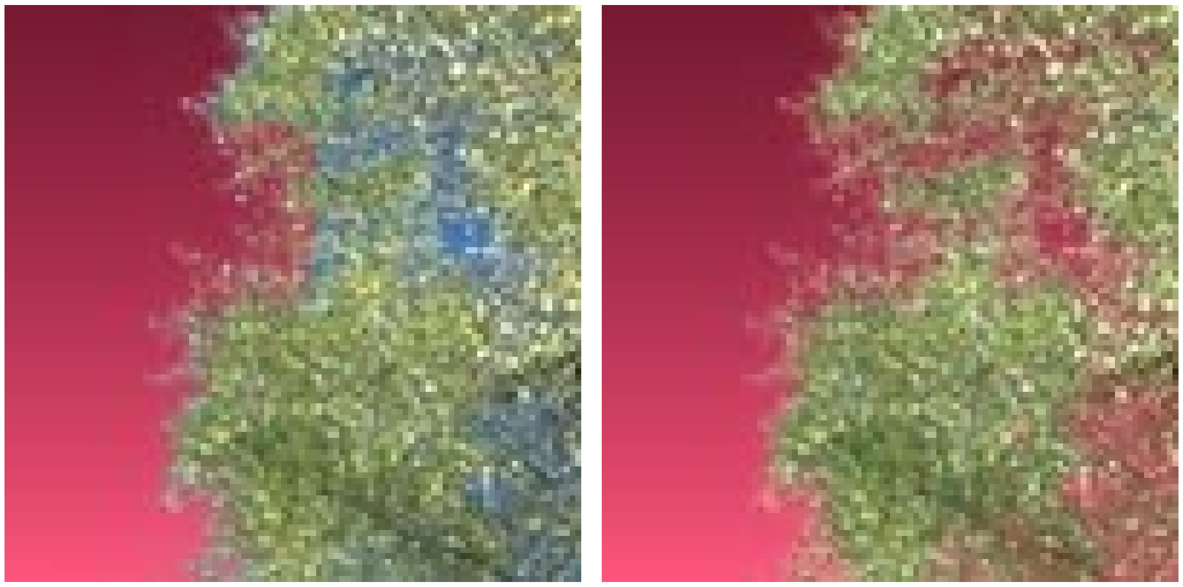


Figure 5.8: The results of the algorithm when the boundary region is specified are poor because blue appears in both object regions. Specifying the regions that are purely tree or sky leads to a more satisfying result.

conservative specification of the two object regions is enough to recover most of the branch. The twigs connecting the leaves are dark, and so they are lost, and some of the highlights from the background are brought into the foreground, but overall the new rendering retains photorealism.

Figure 5.10 displays a plume of smoke extracted using the boundary specification method. Since another plume is in this image, specifying object regions would produce poor results. The boundary is extracted using Intelligent Scissors and dilated by different amounts to capture, among other things, the hole in the plume near one endpoint. The other plume does not have deleterious effects on the result.

Figure 5.11 shows how a waterfall image whose regions have irregular topology can still be extracted. The interior region between the two halves of the waterfall is successfully removed. The colors are numerous and well localized spatially, but the algorithm creates global color distributions that are unable to take advantage of this fact.

The final example, Figure 5.12, is of a woman whose hair is being blown about

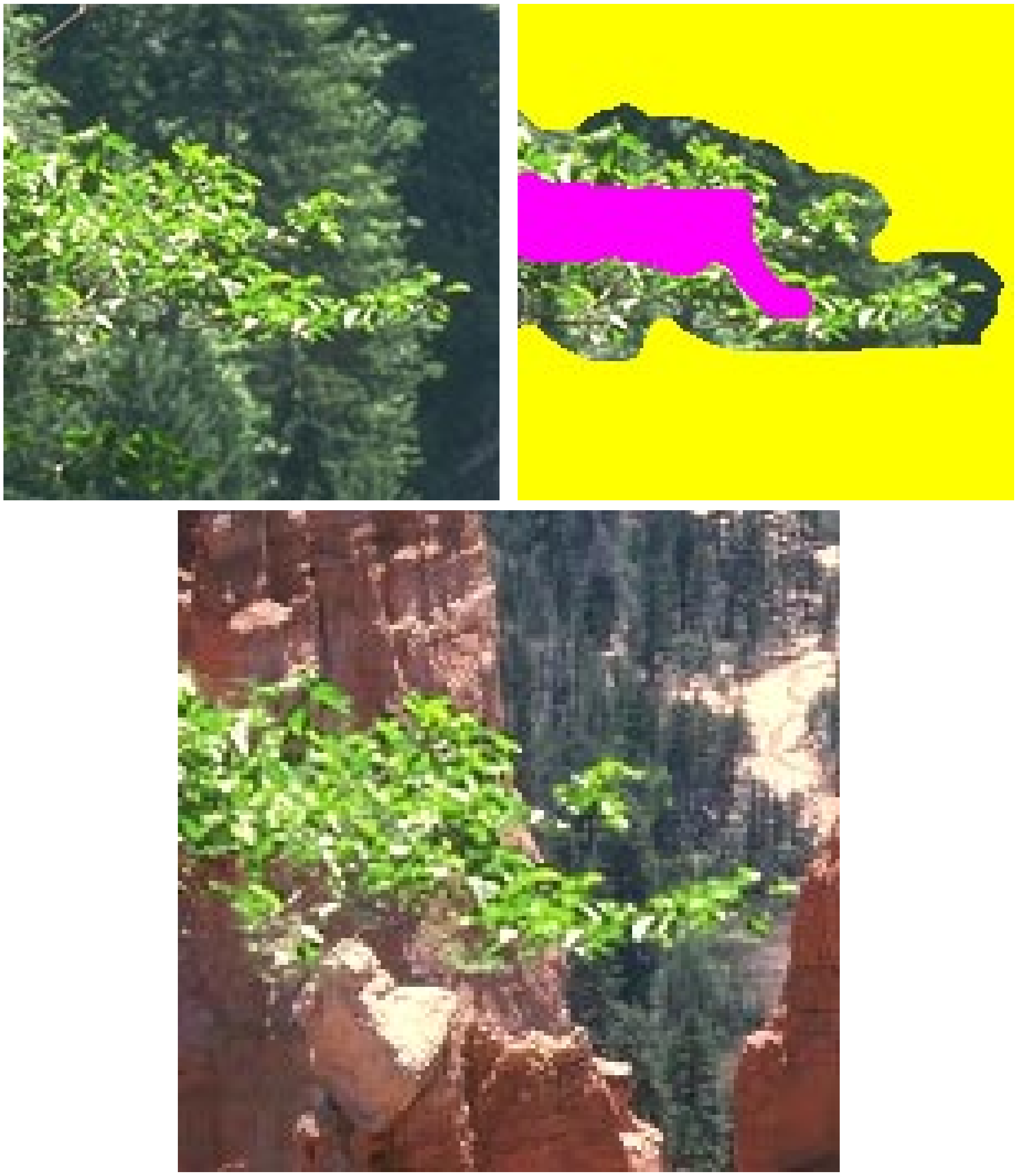


Figure 5.9: A tree branch is extracted from a tree background.

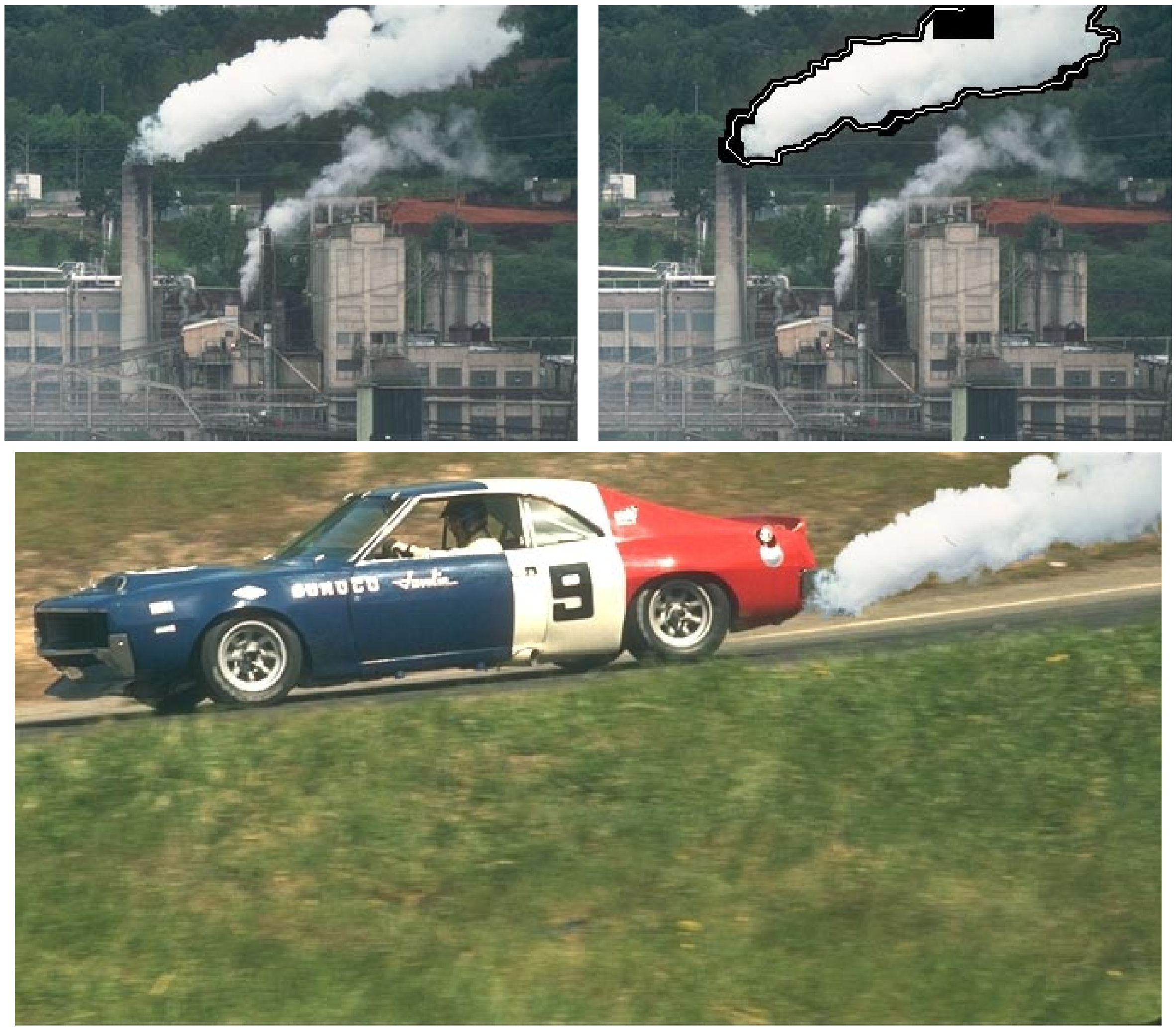


Figure 5.10: A plume of smoke causes a racing car to fail its emissions test.

by the wind. The boundary region has been specified using Intelligent Scissors and dilated by different amounts. The riverbank in the background has similar colors to the hair, so the boundary must be narrow in those areas to minimize artifacts. Mao's picture shows through her hair in the final image, but not every strand of hair is recovered.

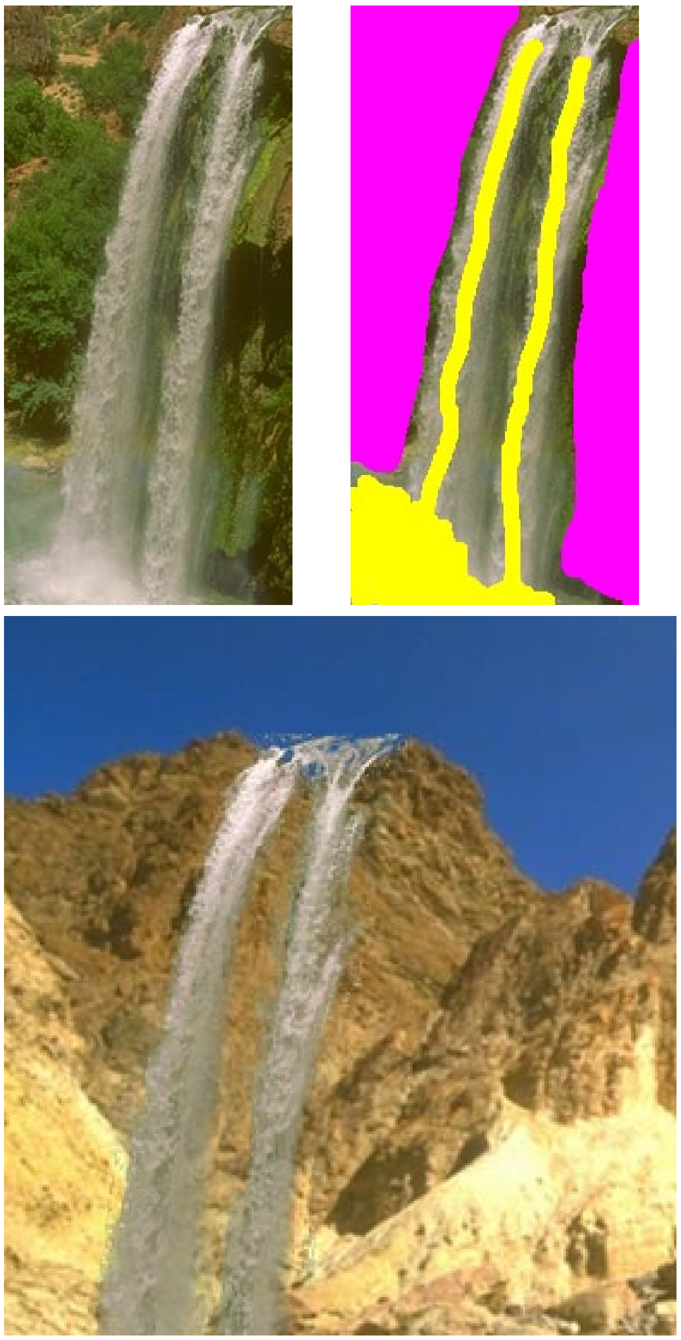


Figure 5.11: A waterfall is transported to arid Death Valley.

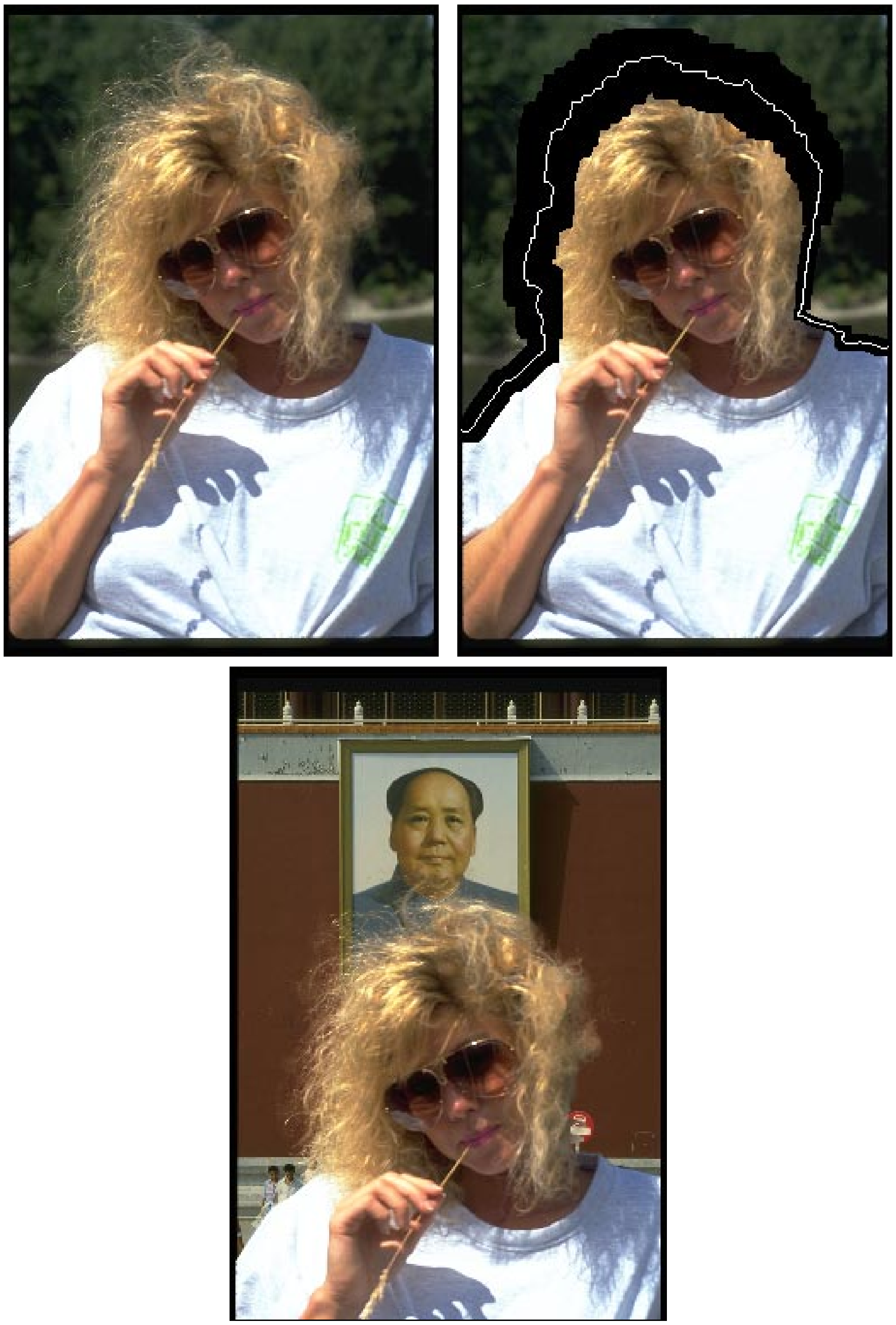


Figure 5.12: A woman takes an instantaneous vacation to Beijing.

5.5 Conclusions

The problem of extracting an object from its background has no general solution. The film and video industries use equipment that provides manual control over the extraction process, but only when the background is a constant color. In contrast, we have presented a tool for extracting image regions from almost arbitrary backgrounds. It requires enough knowledge of a boundary's location to estimate the color distributions of the two image regions accurately, and the computation does not allow for human intervention. If a result is unsatisfactory, the user must change the input specification or alter the alpha values manually.

The results show that foreground objects can be moved to new images without appearing counterfeit (with the exception of changes in illumination between the two images). A close examination will show some defects because the colors are not well separated, or because the color representation is not accurate enough. The second can be solved by increasing the size of the color signatures, but the first is still unapproachable. Translucent objects cannot be handled by this algorithm because no "pure" object pixels exist.

When the boundary specification method can be used, the algorithm produces more accurate results and ensures that any errors are local. For many boundary regions, however, the topology makes this impossible. Specifying object regions directly solves this problem, but only when the color distributions are spatially uniform over the image. Preventing distributions from sharing colors is more difficult in this case, and a method of providing local correspondence for arbitrary region topologies would be helpful.

The fact that we are excluding colors in objects that are not close to the boundary of the manifold in color space has the practical effect of producing biases in the alpha values. Angle conflicts in particular force us to overestimate values with respect to the signature with two of the three collinear clusters. Using information about the spatial distribution and relative amounts of colors might allow us to choose one color or the other at random.

Finally, we reiterate that this algorithm is indeed a tool, not a system. Without reliable user input as to the location of the boundary, the algorithm cannot succeed unless the boundary can be extracted through edge detection or segmentation algorithms, an unlikely prospect for the types of boundaries that benefit most from this algorithm. Nevertheless, it expands the power of image extraction techniques.

Chapter 6

Conclusions

Distributions form more accurate representations of image neighborhoods than can be achieved by using one value or by assuming that a neighborhood can be approximated by a polynomial surface. The results of the preceding three chapters have shown that this increase in accuracy does indeed have advantages when performing early vision tasks. We now summarize the overall contributions and point out areas for future research.

6.1 Contributions of the Thesis

The variety of natural and man-made objects in the world leads to more possibilities for a neighborhood of pixel values than we can account for using established models. Early edge detectors modeled neighborhoods as constant in intensity or color. Later, neighborhoods were modeled as quasi-constant surfaces such as hyperbolic tangents and then planes that were allowed to tilt. Such efforts did extend the set of detectable edges.

However, even polynomial surface models do not adequately represent waterfalls, ivy, or bricks except at the smallest scales where the assumptions made by previous models are valid. It has long been known that feature detection at large scales is more likely to yield robust, salient features, but the problem of how to get meaningful output at large scales has not been adequately addressed. Distributions are ideal for this case, as the color signatures we use to represent them can have arbitrary size. The results have repeatedly shown a marked increase in quality, though we have made computational tradeoffs that affect performance in images where simpler models are valid. The most striking qualitative difference is that the compass operator is able to preserve information about occlusions compared to other operators; junctions are the least likely places where edge models are valid.

All the arguments for using distributions for edge detection are even more true of corner detection. The detector we propose is a natural extension of our edge detector, whereas most edge detectors do not share this property. The difficulty of combining outputs for different color components is more pronounced than it is for edge detection because corners are point features as opposed to ridge features and because corners need more parameters to describe them. To our knowledge, the corner detector presented here is the only one that works on color images, and it is certainly the only one that can successfully deal with large, textured neighborhoods.

Estimating alpha is unlike the other two problems we have considered because it requires substantial user input and does not provide much information about objects in the world. Its relevance stems from the fact that it applies distributions to a class of boundaries that cannot be found using standard notions about edges and corners. Blue screen matting techniques are too restrictive if the user has only a single image to work with. If the local distributions near a potential boundary pixel can be estimated and are well separated in color space, we can find values of alpha that are stable with respect to perturbations of the pixel's color. The results show that these alpha values are also perceptually meaningful.

Overall, the conceptual advantage of distributions is the framework they present for color or multi-spectral imagery. The models developed for greyscale feature detection are often adapted for color images, but most of the time the algorithms require separate computation on each component. The methods presented in this work treat the image values as vectors, resulting in one answer per pixel instead of many.

6.2 What Still Lies Ahead

Despite the successes described here, we cannot claim to have had the last word on detecting and representing boundaries. In this final section we discuss areas of future research. We first consider problems within the scope of this work, followed by other areas to which the principles discussed here might be applied and hopefully extended.

6.2.1 Potential Improvements

The biggest issue in implementing color distributions as a data structure is stability as an operator translates through the image. The EMD was chosen for distance computations in part because of its robustness. However, feature detection requires finding maxima, and EMD values are compared to those in adjacent neighborhoods. Even small changes induced by those few pixels not in the overlap between two adjacent neighborhoods can have noticeable effects on the locations of edge and corner points. Similar problems affect alpha estimation even though it does not use the EMD.

Unfortunately, these problems are most noticeable near boundaries. Because the colors of pixels near boundaries have sparse density in the distribution, these colors are usually not represented as accurately. The result is that translating the window often varies a pixel's representative greatly; it may be grouped with pixels from both sides of an edge at different times. This effect degrades localization and even causes artifacts in alpha values at places where the local color distribution changes.

Another source of instability arises from our choice of the CIE-Lab color space. Although the combination of CIE-Lab with a saturating distance measure has proven effective, the colors produced through vector quantization do not always match human expectations. CIE-Lab was designed to measure distances between large, uniform color patches, not individual pixels. Other representations derived from CIE-Lab [124] may better take these effects into account. A separate difficulty is that the transformation from RGB to CIE-Lab or other perceptual color spaces has non-uniform effects on noise. Our experiments showed that the greyscale compass operator works much better in RGB than in CIE-Lab. A recent alternative proposed in [103] clustered in RGB but computed distances using CIE-Lab.

Again, these problems are exacerbated when finding corners. Because corners are point features, errors in localization are more noticeable. Also, the neighborhood inside the corner may have many fewer pixels, so noise problems are even more likely to produce inaccurate color signatures.

A separate area in which this work could be strengthened is the connection between the different parts. We have already alluded to the fact that we were unable to find a way to incorporate edge and corner information together. The corner detector computes all possible 276 EMDs (using default parameter values) between two complementary neighborhoods at a point. The resulting data structure must contain enough information to describe precisely the structure of the image in that neighborhood, yet we have not found a way to extract this information.

In addition, the edges produced by the compass operator are, in most cases, insufficient to use as the input to the alpha estimation algorithm. The fact that the boundary region often changes width implies that edges from different scales would have to be combined to produce usable input, and we have already placed this problem outside the scope of this work. Even so, edge detectors do not always form the boundary we are interested in, either because the local nature of the algorithm is insufficient to detect a boundary that is global in nature, or because they lack semantic information that is of primary interest to the user.

6.2.2 Future Applications

We believe that using distributions and recasting the resulting comparison problems in terms of flow is one that can be applied to more than just the three applications we have shown here. We end by describing some future possibilities and by considering the next step in representing image neighborhoods.

The Marr-Hildreth edge detector is isotropic and relies on detecting zero crossings in order to find edges. The sign of the operator states whether the center of a window is brighter or darker than its surround. Though the drawbacks of isotropic edge detectors have been made clear by many others, the operator is still popular because it is guaranteed to form closed contours.

What would a color version of this operator look like? One could add up the response over the different components (used in [64]), but this is not optimal. RGB values are correlated with intensity, so there is little difference between color and greyscale under this scheme. CIE-Lab components are uncorrelated, meaning that the signs of the values at each pixel are also uncorrelated, leading to poor results.

It would seem that color signatures and the EMD would provide a good answer to this dilemma. Unfortunately, the EMD, being a distance measure, yields only positive values; zero crossings do not exist, so detecting edges becomes much more difficult. Polarity is a sensible concept only in one dimension and it is undefined in the three dimensions of color. Is there an alternate formulation that makes more sense, or is this another reason why the Laplacian of the Gaussian should not be used?

The compass operator finds only step edges, but lines, or delta edges, are also important in specific applications such as finding roads in satellite images. A delta edge detector is definitely anisotropic, so its implementation is necessarily slower than that of the compass operator. The conceptual difficulty is that it would also respond to step edges, albeit not as strongly. Distinguishing a strong response to a step edge from a weaker response to a delta edge is difficult under this formulation.

There is room for these principles to be applied to texture in two different ways: finding edges in textured regions and implementing texture filters. A family of filters applied to the same point in an image results in a vector, often quite long. These vectors could be used as input to a multidimensional version of the compass operator. The main issue here is creating a perceptual distance function. Since the filters have different sizes, their values change at different rates as we move across an image, and the distance function must take this into account. Distributions have already been used in this way for simple texture mosaics [87].

Another way to use distributions for texture is in implementing a filter. For instance, Gabor filters are zero-mean, so they can be implemented for color images in the same way as the compass operator. Doing so would reduce the length of the vector by two-thirds since there would no longer be a need to filter each image neighborhood once for each color component. It would also eliminate the need to denote some numbers in a vector as being more strongly related because the only difference between the filters was the component to which each was applied. However, it is not clear that the resulting vectors would still contain the same amount of information for performing tasks such as classification. Also, the polarity of the result may be as important for Gabor filters as for the Marr-Hildreth operator.

The most intriguing part of this work that remains unexplored is abnormality. Lack of symmetry appears to be an important cue for junctions, and we have shown that edges found by the compass operator reconstruct junctions more accurately than previous methods. Discovering the relationship between the two, especially as scale changes, would hopefully make it feasible to begin constructing relative depth maps from single images based on occlusions.

We end by asking the question, what next? Distributions are the most complete representation of a neighborhood that uses only pixel values. It may turn out that other data structures or vector quantization algorithms may be better than what we have proposed, but such improvements would only concur with the principles established here. The most obvious generalization would include spatial information with each cluster. In Chapter 5 we added the variance of each cluster to our representation, but its usage is not generalizable beyond our algorithm. Another quantity that could be added is the spatial center of mass of a cluster. If a set of pixels that map to a particular color is scattered evenly throughout a neighborhood, for example, it may be useful to exclude that color from the EMD. The SUSAN operator [102] implemented spatial ideas for finding corners and edges simultaneously, but its model is too idealized for the images used as examples here.

A more unified method would extend a color vector to include its x- and y-coordinates. This mixture of different modalities immediately leads to the problem of how color and spatial information should be weighed against each other, or even whether the weighting should be constant for all tasks, which it is probably not. Regardless, we predict that the most fruitful source of progress will come from merging information about distributions of pixel values with their coordinates into a single, unified model. Neither color nor texture alone will find all the features we would like to detect, but the combination of the two has the potential to take early vision even farther.

Bibliography

[1] I. Abdou and W. Pratt. Qualitative design and evaluation of enhancement/thresholding edge detector. *Proceedings of the IEEE*, 67(5):753–763, May 1979.

[2] J. Abramatic. Why the simplest "Hueckel" edge detector is a Roberts operator. *Computer Graphics and Image Processing*, 17:79–83, 1981.

[3] J. Adams, M. Smith, and P. Johnson. Spectral mixture modeling: A new analysis of rock and soil types at the Viking 1 lander site. *Journal of Geophysical Research*, 91(B8):8098–8112, July 10, 1986.

[4] R. Alberto Salinas, C. Richardson, M. Abidi, and R. Gonzalez. Data fusion: Color edge detection and surface reconstruction through regularization. *IEEE Transactions on Industrial Electronics*, 43(3):355–363, June 1996.

[5] L. Alvarez and F. Morales. Affine morphological multiscale analysis of corners and junctions. *International Journal of Computer Vision*, 25(2):95–107, November 1997.

[6] P. Beaudet. Rotationally invariant image operators. In *Proceedings of the International Joint Conference on Pattern Recognition*, pages 579–583, 1978.

[7] B. Bell and L. Pau. Contour tracking and corner detection in a logic programming environment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):913–917, September 1990.

[8] T. Binford and P.-C. Chiang. Generic, model-based edge estimation in the image surface. In *Proceedings of the Image Understanding Workshop*, volume II, pages 1237–1246, May 1997.

[9] K. Bowyer, C. Kranenburg, and S. Dougherty. Edge detector evaluation using empirical ROC curves. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 354–359, 1999.

[10] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[11] T. Carron and P. Lambert. Color edge detector using jointly hue, saturation, and intensity. In *IEEE International Conference on Image Processing*, volume 3, pages 977–981, November 1994.

[12] T. Carron and P. Lambert. Fuzzy color edge extraction by inference rules: Quantitative study and evaluation of performances. In *IEEE International Conference on Image Processing*, volume 2, pages 181–184, October 1995.

[13] F. Chabat, G. Yang, and D. Hansell. A corner orientation detector. *Image and Vision Computing*, 17(10):761–769, August 1999.

[14] M. Chapron. A new chromatic edge detector used for color image segmentation. In *Proceedings—International Conference on Pattern Recognition*, volume III, pages 311–314, August 1992.

[15] M. Chapron. A chromatic contour detector based on abrupt change techniques. In *IEEE International Conference on Image Processing*, volume III, pages 18–21, October 1997.

[16] C. Chen, J. Lee, and Y. Sun. Wavelet transformation for gray-level corner detection. *Pattern Recognition*, 28(6):853–861, June 1995.

[17] S. Cohen. *Finding Color and Shape Patterns in Images.* PhD thesis, Computer Science Department., Stanford University, Stanford, CA, May 1999.

[18] J. Cooper, S. Venkatesh, and L. Kitchen. Early jump-out corner detectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):823–828, August 1993.

[19] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*, chapter 27. McGraw-Hill, 1990.

[20] A. Cumani. Edge detection in multispectral images. *CVGIP: Graphical Models and Image Processing*, 53(1):40–51, January 1991.

[21] A. Cumani, P. Grattoni, and A. Guiducci. An edge-based description of color images. *CVGIP: Graphical Models and Image Processing*, 53(4):313–323, July 1991.

[22] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2):167–187, 1987.

[23] R. Deriche and G. Giraudon. A computational approach for corner and vertex detection. *International Journal of Computer Vision*, 10(2):101–124, April 1993.

[24] S. Di Zenzo. A note on the gradient of a multi-image. *Computer Vision, Graphics, and Image Processing*, 33(1):116–125, January 1986.

[25] P. Djuric and J. Fwu. On the detection of edges in vector images. *IEEE Transactions on Image Processing*, 6(11):1595–1601, November 1997.

[26] R. Dobrushin. Prescribing a system of random variables by conditional distributions. *Theory of Probability and its Applications*, 15(3):458–486, September 1970.

[27] S. Dougherty and K. Bowyer. Objective evaluation of edge detectors using a formally defined framework. In K. Bowyer and P. Phillips, editors, *Empirical Evaluation Techniques in Computer Vision*, pages 211–234. IEEE Computer Society Press, 1998.

[28] C. Drewniok. Multispectral edge-detection—some experiments on data from Landsat-TM. *International Journal of Remote Sensing*, 15(18):3743–3766, December 1994.

[29] R. Dudley. Distances of probability measures and random variables. *Annals of Mathematical Statistics*, 39(5):1563–1572, 1968.

[30] J. Elder and S. Zucker. Local scale control for edge detection and blur estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):699–716, July 1998.

[31] K. Fishkin and B. Barsky. A family of new algorithms for soft filling. *Computer Graphics*, 18(3):235–244, July 1984.

[32] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, Reading, MA, 1990.

[33] A. Gersho and R. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, Norwell, MA, 1992.

[34] R. Gray, D. Neuhoff, and P. Shields. A generalization of Ornstein's $\bar{d}$ distance with applications to information theory. *Annals of Probability*, 3(2):315–328, April 1975.

[35] M. Han, D. Jang, and J. Foster. Identification of corner points of two-dimensional images using a line search method. *Pattern Recognition*, 22(1):13–20, February 1989.

[36] C. Harris and M. Stephens. A combined corner and edge detector. In *Fourth Alvey Conference*, pages 147–152, 1988.

[37] F. Heitger. Feature detection using suppression and enhancement. Technical Report 163, Swiss Federal Institute of Technology ETH, Zurich, 1995.

[38] M. Hueckel. An operator which locates edges in digitized pictures. *Journal of the ACM*, 18(1):113–125, January 1971.

[39] T. Huntsberger and M. Descalzi. Color edge detection. *Pattern Recognition Letters*, 3:205–209, 1985.

[40] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[41] M. Isard and A. Blake. CONDENSATION—conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[42] Q. Ji and R. Haralick. Breakpoint detection using covariance propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):845–851, August 1998.

[43] M. Kelly. Edge detection by computer using planning. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume VI, pages 397–409. Edinburgh University Press, 1971.

[44] L. Kitchen and A. Rosenfeld. Grey level corner detection. *Pattern Recognition Letters*, 1(2):95–102, December 1982.

[45] C. Kranenburg. Personal communication, February 2000.

[46] R. Laganiere. A morphological operator for corner detection. *Pattern Recognition*, 31(11):1643–1652, November 1998.

[47] Y. Leclerc and S. Zucker. The local structure of image discontinuities in one dimension. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3):341–355, May 1987.

[48] J. Lee, Y. Sun, C. Chen, and C. Tsai. Wavelet based corner detection. *Pattern Recognition*, 26(6):853–865, June 1993.

[49] K. Lee and Z. Bien. A gray-level corner detector using fuzzy-logic. *Pattern Recognition Letters*, 17(9):939–950, August 1996.

[50] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Mathematics—Doklady*, 10:707–710, 1966.

[51] B. Luo, A. Cross, and E. Hancock. Corner detection via topographic analysis of vector potential. *Pattern Recognition Letters*, 20(6):635–650, June 1999.

[52] L. Macaire, V. Ultre, and J. Postaire. Determination of compatibility coefficients for colour edge detection by relaxation. In *IEEE International Conference on Image Processing*, volume III, pages 1045–1048, September 1996.

[53] R. Machuca and K. Phillips. Applications of vector fields to image processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(3):316–329, May 1983.

[54] M. Malowany and A. Malowany. Color-edge detectors for a VLSI convolver. In *Proceedings of the SPIE*, volume 1199, pages 1116–1126, 1989.

[55] D. Marr and E. Hildreth. Theory of edge-detection. *Proceedings of the Royal Society of London*, B-207(1167):187–217, 1980.

[56] J. Matas and J. Kittler. Junction detection using probabilistic relaxation. *Image and Vision Computing*, 11(4):197–202, May 1993.

[57] G. Medioni and Y. Yasumoto. Corner detection and curve representation using cubic B-splines. *Computer Vision, Graphics, and Image Processing*, 39(3):267–278, September 1987.

[58] R. Mehrotra, S. Nichani, and N. Ranganathan. Corner detection. *Pattern Recognition*, 23(11):1223–1233, November 1990.

[59] T. Mitsunaga, T. Yokoyama, and T. Totsuka. AutoKey: Human assisted key extraction. In *Computer Graphics Proceedings (SIGGRAPH)*, pages 265–272, August 1995.

[60] A. Moghaddamzadeh and N. Bourbakis. A fuzzy approach for smoothing and edge detection in color images. In *Proceedings of the SPIE*, volume 2421, pages 90–102, February 1995.

[61] A. Moghaddamzadeh, D. Goldman, and N. Bourbakis. Fuzzy-like approach for smoothing and edge detection in color images. *International Journal of Pattern Recognition and Artificial Intelligence*, 12(6):801, September 1998.

[62] F. Mokhtarian and R. Suomela. Robust image corner detection through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1376–1381, December 1998.

[63] H. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the International Joint Conference on Artificial Intelligence*, page 584, 1977.

[64] E. Mortensen and W. Barrett. Interactive segmentation with intelligent scissors. *CVGIP: Graphical Models and Image Processing*, 60:349–384, September 1998.

[65] V. Nalwa. *A Guided Tour of Computer Vision*. Addison-Wesley, Reading, MA, 1993.

[66] V. Nalwa and T. Binford. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):699–714, November 1986.

[67] A. Nering and A. Tucker. *Linear Programs and Related Problems*, chapter 9. Academic Press, New York, 1993.

[68] R. Nevatia. A color edge detector. In *Proceedings—International Conference on Pattern Recognition*, pages 829–832, November 1976.

[69] R. Nevatia. A color edge detector and its use in scene segmentation. *IEEE Transactions on Systems, Man and Cybernetics*, 7(11):820–826, November 1977.

[70] W. Niblack, R. Barber, W. Equitz, M. D. Flickner, E. H. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. Querying images by content, using color, texture, and shape. In *Proceedings of the SPIE*, volume 1908, pages 173–187, April 1993.

[71] J. Noble. Finding corners. *Image and Vision Computing*, 6(2):121–128, May 1988.

[72] M. Orchard and C. Bouman. Color quantization of images. *IEEE Transactions on Signal Processing*, 39(12):2677–2690, December 1991.

[73] K. Paler, J. Foglein, J. Illingworth, and J. Kittler. Local ordered gray levels as an aid to corner detection. *Pattern Recognition*, 17(5):535–543, September 1984.

[74] L. Parida, D. Geiger, and R. Hummel. Junctions: Detection, classification, and reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):687–698, July 1998.

[75] W. Perkins and T. Binford. A corner finder for visual feedback. *Computer Graphics and Image Processing*, 2(3/4):355–376, December 1973.

[76] M. Pietikainen and D. Harwood. Edge information in color images based on histograms of differences. In *Proceedings—International Conference on Pattern Recognition*, pages 594–596, 1986.

[77] K. Pingle. Visual perception by a computer. In A. Grasselli, editor, *Automatic Interpretation and Classification of Images*, pages 277–284. Academic Press, New York, 1969.

[78] T. Porter and T. Duff. Compositing digital images. *Computer Graphics*, 18(3):253–259, July 1984.

[79] J. Prewitt. Object enhancement and extraction. In B. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics*, pages 75–149. Academic Press, New York, 1970.

[80] K. Rangarajan, M. Shah, and D. Van Brackle. Optimal corner detector. *Computer Vision, Graphics, and Image Processing*, 48(2):230–245, November 1989.

[81] L. Roberts. Machine perception of three-dimensional solids. In J. Tippet, D. Berkowitz, L. Clapp, C. Koester, and A. Vanderburgh, Fr., editors, *Optical and Electro-Optical Information Processing*, pages 159–197. MIT Press, Cambridge, MA, 1965.

[82] G. Robinson. Color edge detection. *Proceedings of the SPIE*, 87:126–133, August 1976.

[83] G. Robinson. Color edge detection. *Optical Engineering*, 16(5):479–484, September 1977.

[84] K. Rohr. Recognizing corners by fitting parametric models. *International Journal of Computer Vision*, 9(3):213–230, 1992.

[85] A. Rosenfeld. The max Roberts operator is a Hueckel-type edge detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(1):101–103, January 1981.

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

[87] Y. Rubner. *Perceptual Metrics for Image Database Navigation.* PhD thesis, Computer Science Department, Stanford University, Stanford, CA, May 1999.

[88] Y. Rubner, C. Tomasi, and L. Guibas. A metric for distributions with applications to image databases. In *IEEE International Conference on Computer Vision*, pages 59–66, January 1998.

[89] M. Ruzon and C. Tomasi. Color edge detection with the compass operator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 160–166, June 1999.

[90] M. Ruzon and C. Tomasi. Corner detection in textured color images. In *IEEE International Conference on Computer Vision*, volume II, pages 1039–1045, September 1999.

[91] M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2000. to appear.

[92] E. Saber, A. Tekalp, and G. Bozdagi. Fusion of color and edge information for improved segmentation and edge linking. *Image and Vision Computing*, 15(10):769–780, October 1997.

[93] J. Scharcanski and A. Venetsanopoulos. Edge detection of color images using directional operators. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(2):396–401, April 1997.

[94] T. Sebok, L. Roemer, and G. Malindzak, Jr. An algorithm for line intersection identification. *Pattern Recognition*, 13(2):159–166, April 1981.

[95] U. Seeger and R. Seeger. Fast corner detection in grey-level images. *Pattern Recognition Letters*, 15(7):669–675, July 1994.

[96] H. Shen and A. Wong. Generalized texture representation and metric. *Computer Vision, Graphics, and Image Processing*, 23(2):187–206, August 1983.

[97] R. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–23, 11 September 1987.

[98] M. Shin, D. Goldgof, and K. Bowyer. An objective comparison methodology of edge detection algorithms using a structure from motion task. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 190–195, 1998.

[99] A. Shiozaki. Edge extraction using entropy operator. *Computer Vision, Graphics, and Image Processing*, 36(1):1–9, October 1986.

[100] A. Singh and M. Shneier. Grey level corner detection: A generalization and a robust real time implementation. *Computer Vision, Graphics, and Image Processing*, 51(1):54–69, July 1990.

[101] A. Smith and J. Blinn. Blue screen matting. In *Computer Graphics Proceedings (SIGGRAPH)*, pages 259–268, August 1996.

[102] S. Smith and J. Brady. SUSAN—a new approach to low-level image-processing. *International Journal of Computer Vision*, 23(1):45–78, May 1997.

[103] K. Song, J. Kittler, and M. Petrou. Defect detection in random color textures. *Image and Vision Computing*, 14(9):667–683, October 1996.

[104] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, November 1991.

[105] H. Tao and T. Huang. Color image edge detection using cluster analysis. In *IEEE International Conference on Image Processing*, volume I, pages 834–837, 1997.

[106] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

[107] P. Trahanias and A. Venetsanopoulos. Color edge detection using vector order statistics. *IEEE Transactions on Image Processing*, 2(2):259–264, April 1993.

[108] P. Trahanias and A. Venetsanopoulos. Vector order-statistics operators as color edge detectors. *IEEE Transactions on Systems, Man and Cybernetics*, B-26(1):135–143, February 1996.

[109] M. Trajkovic and M. Hedley. Fast corner detection. *Image and Vision Computing*, 16(2):75–87, February 1998.

[110] P. Tsang and W. Tsang. Edge detection on object color. In *IEEE International Conference on Image Processing*, volume 3, pages 1049–1052, 1996.

[111] W. Tsang and P. Tsang. Suppression of false edge-detection due to specular reflection in color images. *Pattern Recognition Letters*, 18(2):165–171, February 1997.

[112] S. Vallender. Calculation of the Wasserstein distance between probability distributions on the line. *Theory of Probability and its Applications*, 18(4):784–786, December 1973.

[113] H. von Helmholtz. *Handbuch der Physiologischen Optik*, volume 1. The Optical Society of America, 1909. Translated by J.P.C. Southall, 1924.

[114] H. Wang and M. Brady. Real-time corner detection algorithm for motion estimation. *Image and Vision Computing*, 13(9):695–703, November 1995.

[115] S.-J. Wang and T. Binford. Generic, model-based estimation and detection of discontinuities in image surfaces. In *Proceedings of the Image Understanding Workshop*, volume II, pages 113–116, November 1994.

[116] A. Weeks and H. Myler. Edge detection of color images using the HSL color space. In *Proceedings of the SPIE*, volume 2424, pages 291–301, February 1995.

[117] M. Werman, S. Peleg, and A. Rosenfeld. A distance metric for multidimensional histograms. *Computer Vision, Graphics, and Image Processing*, 32(3):328–336, December 1985.

[118] Z. Wu and A. Rosenfeld. Filtered projections as an aid in corner detection. *Pattern Recognition*, 16(1):31–38, January 1983.

[119] G. Wyszecki and W. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley and Sons, New York, NY, 1982.

[120] X. Xie, R. Sudhakar, and H. Zhuang. Corner detection by a cost minimization approach. *Pattern Recognition*, 26(8):1235–1243, August 1993.

[121] C. Yang and W. Tsai. Reduction of color space dimensionality by moment-preserving thresholding and its application for edge-detection in color images. *Pattern Recognition Letters*, 17(5):481–490, May 1996.

[122] W. Yu, K. Daniilidis, and G. Sommer. Rotated wedge averaging method for junction classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 390–395, 1998.

[123] C. Zetsche and G. Krieger. *Natural-Image Statistics and the Exploitation of Local Intrinsic Dimensionality*. Akademischer Verlag, Munich, 1997.

[124] X. Zhang and B. Wandell. A spatial extension of CIELAB for digital color image reproduction. In *Proceedings of the Society for Information Display*, pages 731–734, San Diego, CA, May 1996.

[125] Z. Zheng, H. Wang, and E. Teoh. Analysis of gray level corner detection. *Pattern Recognition Letters*, 20(2):149–162, February 1999.