Advances in Piecewise Smooth Image Reconstruction

by

Ralf Juengling

A dissertation submitted in partial fullfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

Portland State University
2014

Abstract

Advances and new insights into algorithms for piecewise smooth image reconstruction are presented. Such algorithms fit a piecewise smooth function to image data without prior knowledge of the number of regions or the location of region boundaries in the best fitting function. This is a difficult model selection problem since the number of parameters of possible solutions varies widely.

The approach followed in this work was proposed by Yvan Leclerc. It uses the Minimum Description Length principle to make the reconstruction problem well-posed—the best fitting function yields the shortest encoding of the image data. In order to derive a code length formula, the class of functions is restricted to piecewise polynomial. The resulting optimization problem may have many local minima, and a good initial approximation is required in order to find acceptable solutions. Good initial approximations may be generated at the cost of solving a sequence of related optimization problems, as prescribed by a continuation method.

Several problems with this approach are identified and addressed. First, success or failure of the continuation method is found to be sensitive to the choice of objective function parameters. Second, the optimization method used in prior work may fail to converge, and, third, it converges too slowly to be useful in many vision applications.

I address the first problem in three different ways. First, a revised continuation method is less sensitive to parameter choice. Second, I show how to move control over success or failure from the objective function parameters to the continuation method. Third, a new objective function is derived which includes one parameter instead of the two parameters used in prior work. Experimental results show that all measures improve robustness with respect to parameter choice.

In order to address the optimization-related problems I use a quasi-Newton line-search method. This method is guaranteed to converge and may converge at a faster rate than the relaxation method used in prior work. To realize a faster convergence rate, I introduce a new parameter whose role is to improve variable scaling and problem conditioning. Further runtime improvements result from using extrapolation in the continuation method. Experimental results show overall runtime improvements of an order of magnitude and more.

My reconstruction algorithm performs superior to the well-known Canny edge detector on the Berkeley boundary detection task. This is a novel result that demonstrates the merits of image reconstruction as a means for extracting information from an image.

Dedication

To Hans-Werner and Gisela.

Acknowledgments

I probably would not have spent quite as many years in school if I did not enjoy learning and exploring ideas so much. I thank my advisor Melanie Mitchell for letting me go wherever my curiosity led me.

What I will most likely remember in the long run from my time at Portland State University are the remarkable individuals I was fortunate to meet. “Grizzly” Martin, who once took me up to the top of Mount Hood. You left your mark on Portland State, my friend! James, who for many years organized the PSU ultimate frisbee pickup that kept me sane. Kathryn, who single-handedly can make a computer science crowd feel like a group of real human beings. And Rashawn, who works harder than anyone else and still finds time for a cup of coffee.

I am grateful to my managers at Synopsys, Tom and Pat, for granting me time off from work so I could finish this dissertation. Thank you to Bryant, Stefan, and Heidi for feedback on my writing, and to the boys in the lab, Mick, Will, and Max, for moral support in the home stretch. And thank you, Verena, for cheering me on from 5000 miles away.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

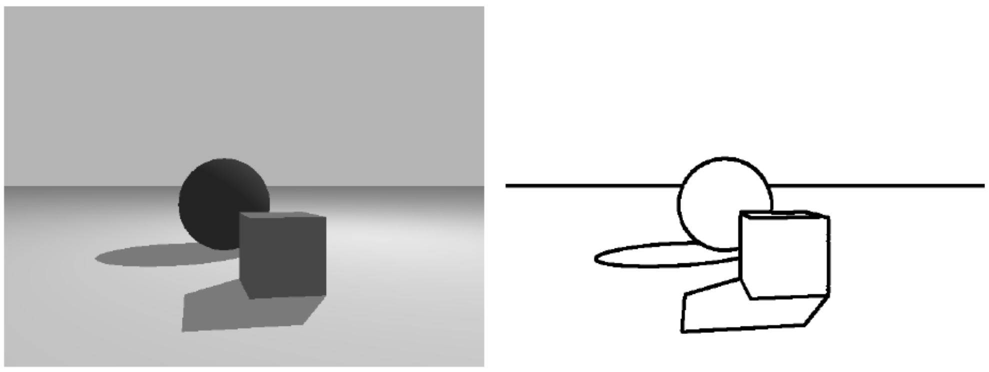


Figure 1.1. Simple, idealized scene of two objects illuminated by a distant point source and homogenous ambient light (left). Discontinuities in the intensity function (right).

People have eyes to see the world around them. "Seeing" means extracting relevant information from the light intensity pattern that is hitting the retina. What information is relevant depends on the situation; the information might be visual in nature, such as the subtle texture on a mushroom's skin that reveals the mushroom species, or it might be more abstract, such as the type of beverage consumed at a social gathering.

Computers may use a camera to record light intensity projecting from their surroundings. If we want a computer to carry out tasks such as identifying mushroom species, interpreting medical images, or driving a car, information relevant to the task at hand needs to be extracted from this intensity data.

Computer vision is concerned with developing methods for extracting information from visual data. While there has been continuous progress in the field for over fifty years, today there is yet no general-purpose computer vision system with the flexibility and analytical capabilities of a seeing human. Rather, a new system needs to be developed for each new task involving analysis of visual data. Some computational steps like segmenting images into regions, or detecting points of large change in visual data need to be carried out in many vision systems. It is economical to develop and study algorithms for such common computational steps in their own right.

This dissertation is concerned with algorithms for "reconstructing" intensity functions from recorded intensity data. Such algorithms may be used as a component in computer vision systems when contours need to be extracted from images, when noise in the recorded data needs to be removed, or when intensity or derivatives of the intensity need to be estimated accurately.

To understand the purpose of a reconstruction algorithm, consider Figure 1.1. The image on the left shows a simple scene where all visible surfaces are smooth. As a result, the intensity incident on the camera's image plane is smooth almost everywhere; points where this intensity function is not smooth are marked black in the image on the right. The recorded intensity data, on the other hand, is not a function but a finite set of gray values. The *reconstruction problem* is to estimate the intensity function, given the gray values. The requirement that the estimated function be smooth almost everywhere makes this task feasible.

In this dissertation I discuss a particular formulation of the reconstruction problem put forth by Leclerc in a seminal work [48], as well as algorithms derived from this formulation. My main goal is to develop a reconstruction algorithm that is more practical as a component of a computer vision system than Leclerc's original algorithm. My improvements concern robustness and runtime performance.

1.1 The Nature of Visual Data

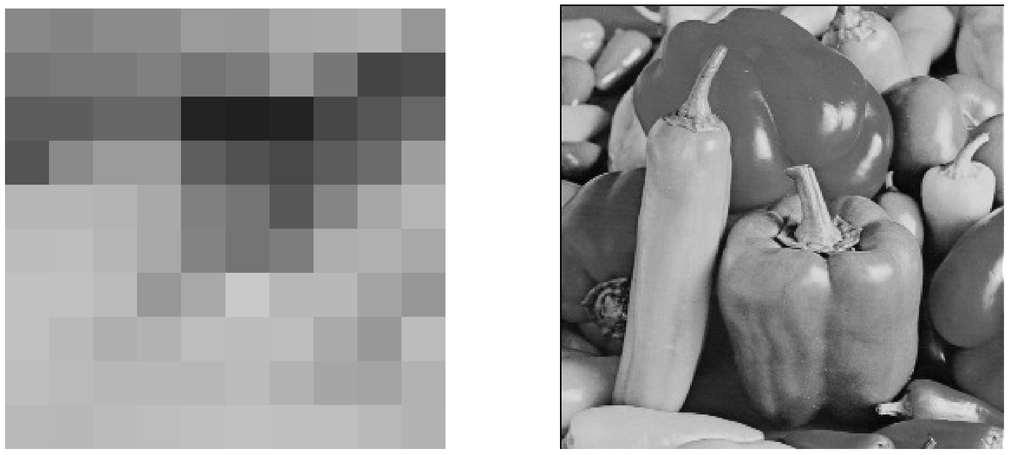


Figure 1.2. 10×10 gray squares of varying intensity (left) and 256×256 gray squares of varying intensity (right). The left image is a close-up of a small piece of the right image.

Teaching *machines* how to make sense of visual data is difficult. Almost equally hard is to convince *people* that this is so. To get a sense of the magnitude of the task consider Figure 1.2. It shows two regular arrangements of gray squares of varying intensity. Human subjects have no difficulty seeing the squares in the left image but have great difficulty seeing the squares in the right image. If instructed to do so, however, most subjects are able to give a verbal description of the right image in terms of visible surfaces, their shape and reflectance properties, and a rough estimate of the lighting that must have illuminated the scene when the image was taken.

The image on the right in Figure 1.2 really is an arrangement of gray squares. And although the light hitting the retina is a faithful replication of those squares on paper, human observers do not have access to the retina images at the *signal level*. It appears that the human visual system constantly summarizes its input and generates abstract descriptions of it. What reaches awareness usually is a much more compact description in terms of setting, objects, and interactions between objects. When consciously directing attention to it, human observers can notice aspects of *lower-level*, less abstract descriptions (surfaces, their interaction with illumination, imaging blur), but perception of the raw, uninterpreted, non-summarized visual input seems out of reach.

Machines acquire visual input by using imaging sensors which record intensity. This input is an arrangement of “gray squares” or *pixels* representing recorded intensity levels. In a general sense the task of machine vision is to generate increasingly abstract descriptions of this input. Borrowing from our intuition about human vision [60] it seems like a good strategy to do this in a hierarchical fashion and to organize descriptions in layers of abstraction. The bottom layer would contain some form of description of the pixel values, the top layer contain descriptions in terms of setting, objects, and so on. Descriptions from lower levels could be utilized in generating descriptions at higher levels.

This dissertation addresses a problem in machine vision. Its topic is the generation of particular kinds of descriptions for the lowest layer within a hierarchy of descriptions. In order to summarize the array of pixel values, regularities need to be identified and exploited, and relevant and irrelevant properties of the intensity data must be separated. Regarding the regularities to exploit and the particular form of the descriptions, I build on prior work [13, 48]. The contributions of this dissertation include both new insights as well as algorithmic improvements. I state them in Section 1.6 after presenting some preliminaries.

1.2 The Image Reconstruction Problem

In deciding what form a description of an array of pixel values should take we need to consider what regularities to expect in the visual signal, and which of its features ought to be made explicit in order to aid the generation of more abstract image descriptions. The light hitting an imaging sensor reaches it directly from a light source or from the surfaces of an object. Points in close proximity on an object's surface project to points in close proximity on the imaging sensor. We can imagine dividing up the area occupied by the imaging sensor into regions corresponding to the different surfaces in the scene that project onto those regions.

For the image on the left of Figure 1.1, for example, this division is shown in the image on the right. Intensity changes gradually within each region but may change in a "jump" when moving from one region to another. A more precise way to state this property is to say that the intensity function across an imaging sensor is *piecewise smooth*.

The assumption of a piecewise smooth intensity suggests that piecewise smooth functions are one suitable type of description at the lowest level of the image description hierarchy. This immediately motivates the problem of *image reconstruction* [13, 48, 58]. The problem is to identify which piecewise smooth function has *most likely* given rise to the recording of the given pixel values.

A sudden change or jump in a function is called a *discontinuity*. Discontinuities in the intensity function may be traced back to a number of causes, such as one object occluding another, sharp changes in the light hitting a surface (shadows), sharp changes in surface reflectance (markings), and so on. The causes of discontinuities are not explicit in a piecewise smooth reconstruction, but the precise shape of the intensity function around discontinuities contains clues [26, 36, 51] that might be exploited by other processes concerned with inferring those causes [5]. For example, the contours on the ground in Figure 1.1 may be identified as shadow contours by determining that the derivatives (of the logarithm) of the intensity on both sides of the discontinuity are of equal magnitude [51]. This information may be readily computed when the intensity is known as a *function*, but not directly from the raw intensity data. The piecewise smooth function description thus aims to make two kinds of information explicit—discontinuities and the shape of the intensity function in the neighborhood of discontinuities.

Note that the piecewise smoothness assumption is not appropriate for *all* images. Scenes of nature, for example, often contain highly textured regions to which the assumption does not apply. For scenes of urban or other man-made environments, on the other hand, that contain predominantly smooth surfaces, the assumption is appropriate [6]. Also, some applications deal with special kinds of images where piecewise smoothness or even stronger assumptions apply (see Chapter 3).

One question that makes the reconstruction problem interesting is whether or not it has a unique answer. Estimation problems without a unique answer are *ill-posed* [78, 68]. This question must be addressed before methods for solving the reconstruction problem can be entertained. Addressing it requires that the meaning of "most likely" in the informal problem statement above be made precise.

1.3 Steps to a Well-Posed Problem

Two aspects add difficulty to the image reconstruction problem. First, the discontinuities delineating the "pieces" of the piecewise smooth function are not known beforehand. Second, the idea of piecewise smoothness is an idealization, as the image acquisition process introduces distortions to the signal through blurring, sensor noise, and quantization. Some of these distortions are random.

Thus, even if the discontinuities were known beforehand, given the intensity levels it is not obvious what each smooth piece of the reconstruction should look like. This is the question of how to make the problem well-posed. Different approaches answer this question differently. This section and the next discuss one such answer, while other approaches are described in Chapter 3.

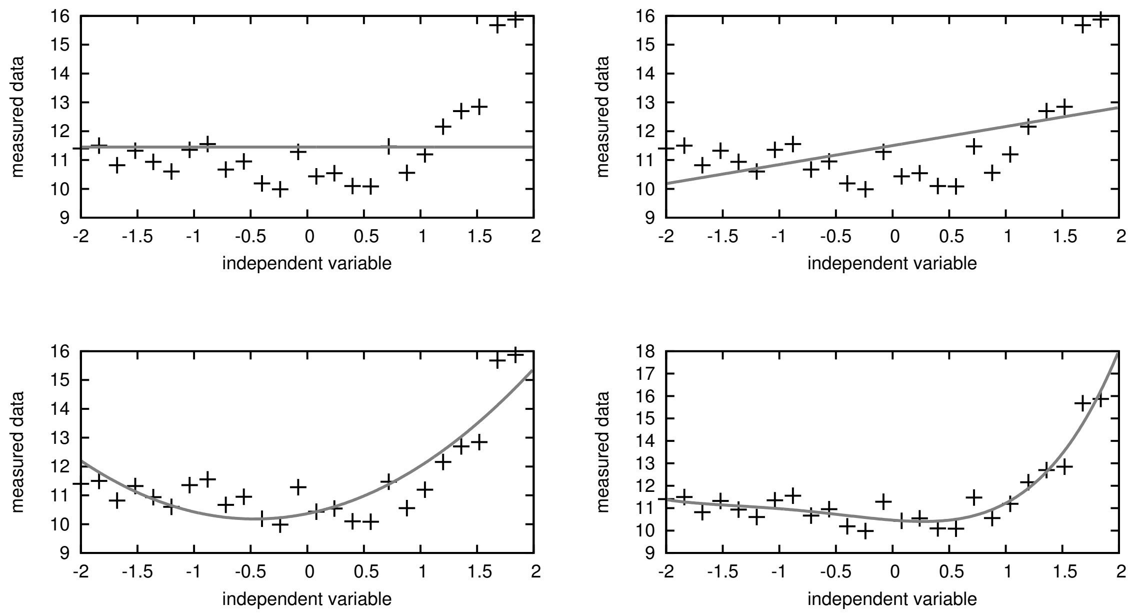


Figure 1.3. A set of data points (top left) and its average. The same points with best fitting linear (top right), quadratic (bottom left), and quartic polynomial (bottom right), respectively.

To simplify the discussion, consider the estimation problem in one dimension. Suppose the intensity levels in Figure 1.3 (top left) have been observed and we are to estimate a smooth function that gave rise to this observation. To solve this problem, we need both a set of candidate functions to choose from, and a criterion for making the choice.

Criteria for choosing a set of functions may include prior knowledge about the problem (e.g., knowledge about what "typical" smooth functions look like), and computational considerations, such as the potential for fast feature extraction from the function representation. Suppose we choose low-degree polynomials with some maximum degree $p_{\max}$, knowing that the unobserved functions are sometimes linear and generally non-oscillatory over the interval sizes considered. Absent any additional constraints or prior knowledge, we would, for each variety of polynomials (constant, linear, etc.), choose the polynomial which best fits the observations in the least-squares sense. For example, Figure 1.3 shows the constant, linear, quadratic and quartic (degree four) polynomial, respectively, that best fits the observations.

While we have the least-squares criterion for selecting a function *within* each variety of polynomials, we still need a criterion for selecting *among* the varieties. We know that the observations include two components—random distortions and the non-random signal. Ideally, we would like to separate the two. The minimum least-squares criterion fails to do this. Since the minimum least-squares criterion would always choose the polynomial of largest degree $p_{\max}$, the chosen polynomial would always interpolate the data when $p_{\max}$ is sufficiently large. Thus, the polynomial would depend on both components of the data.

Intuitively speaking, we are looking for a criterion that selects a function of "appropriate complexity". Ideally, it should both capture all of the qualitative trends in the observations, and be free of accidental detail. A suitable criterion is to select the polynomial that best serves to compactly encode the observations. This is explained in the following section.

1.4 The Minimum Description Length Principle

Rissanen formalized the intuitive notion of "appropriate complexity" as the Minimum Description Length (MDL) principle [71], for which he borrowed ideas from communication theory [75]. To develop the MDL principle using our example, suppose that observations like the ones in Figure 1.3 must be sent frequently from one scientist to another over a costly, digital channel. The scientists agree to a sufficient precision of the numerical data and set out to devise an efficient code for it.

For their first version of the code, the scientists use the fact that, averaged over long periods of time, the observations average to zero and have a mean-square of σ^2. They use Shannon's results to construct a code which minimizes the worst case code length [75]. This leads to an *entropy code* for a Gaussian process with mean zero and variance σ^2. On average, the code length for a vector of observations $\boldsymbol{z}$ is

$$L_0(\boldsymbol{z}) = -\log_2 \prod_{i=1}^{n} P(z_i) = a \sum_{i=1}^{n} \left(c + \frac{z_i^2}{\sigma^2} \right) \tag{1.1}$$

bits, where $a = 1/(2 \log 2)$ and $c = \log 2\pi + 2 \log \sigma$. This code is efficient when the values z_i in $\boldsymbol{z}$ are uncorrelated and when the assumption of zero-centered Gaussianity holds.

The mean and mean-square of each vector $\boldsymbol{z}$ are quite different from zero and σ^2, respectively. The scientists realize that a better code with shorter average code length can be constructed if short-term estimates of mean and variance, $\hat{z}$ and $\hat{\sigma}^2$, are included in the message and utilized in the code. The average code length of this encoding scheme is

$$L_1(\boldsymbol{z}) = L(\hat{z}) + L(\hat{\sigma}) + L_0(\boldsymbol{z} - \hat{z}) = \frac{2}{2} \log_2 n + a \sum_{i=1}^{n} \left(c + \frac{(z_i - \hat{z})^2}{\hat{\sigma}^2} \right) \tag{1.2}$$

bits, where the term $\log n$ on the r.h.s. of (1.2) accounts for the encoding of the two parameters $\hat{z}$ and $\hat{\sigma}$. This term depends on n because, in an optimal code, the more observations to be transmitted, the more accurately $\hat{z}$ and $\hat{\sigma}$ need to be encoded [71].

The second code is more efficient than the first when $L(\hat{z}) + L(\hat{\sigma}) < L_0(\boldsymbol{z}) - L_0(\boldsymbol{z} - \hat{z})$, that is, when the number of bits used to encode the parameters $\hat{z}$ and $\hat{\sigma}$ is smaller than the number of bits saved by entropy encoding the data $\boldsymbol{z} - \hat{z}$ instead of $\boldsymbol{z}$. This will in general only be the case when the observations contain a non-random component.

After a good night's sleep, the scientists realize that even more efficient codes might be constructable in which more data trends are included in the message. The third code they create includes a variable number of parameters in the message. The encoder for this version of the code has more work to do. It fits polynomials of increasing degree p to the observations and calculates the length of the code which would result if $\hat{\sigma}$ and the $p+1$ coefficients of each polynomial were included in the message,

$$L_p(\boldsymbol{z}) = \log_2(p) + \frac{p+2}{2}\log_2 n + a\sum_{i=1}^{n}\left(c + \frac{(z_i - u_p(i))^2}{\hat{\sigma}^2}\right). \tag{1.3}$$

The first term in (1.3) accounts for encoding the polynomial degree, and u_p denotes the polynomial of degree p chosen by the encoder. The encoder stops when $L_{p+1}(\boldsymbol{z}) > L_p(\boldsymbol{z})$ and uses the code with length $L_p(\boldsymbol{z})$. (No matter what code the encoder chooses, the message always begins with an encoding of p, so the decoder knows how to proceed after decoding p.)

The third encoding scheme uses the MDL principle. Simply put, the principle states that, when it comes to separating random and non-random data components, the best *model* (non-random part) of the data is the one which minimizes the length of an optimal code using that model[1.1]. Note that the principle is agnostic about model classes. Choosing good model classes usually requires insight into the processes generating the data at hand.

The MDL criterion for choosing the polynomial u_p^* that best represents the non-random component of the observations in Figure 1.3 is

$$u_p^* = \arg\min_{u_p}\{L_p(\boldsymbol{z}) \,|\, p \in [0, p_{\max}]\}, \tag{1.4}$$

where L_p is the code length formula (1.3). Note that the least-squares criterion for selecting a polynomial of any given degree is implied by (1.4) (if u_p were not the least-squares polynomial of degree p, then the contribution of the last term in (1.3) could be reduced).

1.1. MDL is a principle for *model selection* and belongs to statistics [33], and not to communication theory. A common misunderstanding for novices to the MDL idea is that applying MDL involves construction of encoders and decoders. This is not the case. Knowledge of *how* to construct efficient codes for certain data is usually not required. What needs to be known is existence of codes and ways to compute the length of encodings.

1.5 Using MDL for Image Reconstruction

Image reconstruction is a difficult model selection problem where the data consists of recorded intensity levels and the model class is the set of piecewise smooth functions. This class contains many functions, some simple with few regions of simple shape, others complex and consisting of many regions with complex shapes, and the number of model parameters varies widely for this class of models.

MDL can serve as the model selection criterion for this problem. This requires that code lengths for plausible encodings of the model functions are available. In order to develop code length formulas, the class of piecewise smooth functions needs to be restricted. Just as the scientists in Section 1.4 choose sufficient precision and parametric functions for capturing trends in their data, the class of piecewise smooth functions needs to be replaced with approximations which admit finite encodings. For example, the functions can be restricted to piecewise polynomial with regions consisting of connected sets of pixels. (This is, in fact, exactly the class of functions that is used in this work.)

For the image reconstruction problem the MDL criterion then gives the length of an encoding of the intensity data in terms of a model (a piecewise polynomial function) and *residuals* (the differences between the model and the data). As in the example of Section 1.4, the view is that model and residuals describe deterministic and stochastic components of the observations, respectively, and that the model minimizing the encoding length is the best representation of the deterministic component.

At this point the problem is reduced to an optimization problem—the search for the model that minimizes the encoding length. It turns out that compromises are necessary in order to arrive at a solution algorithm. Specifically, the encoding length for the model is approximated to fulfill the differentiability requirements of numerical optimization methods. This introduces unknown constants (parameters) into the MDL criterion (note that in (1.3) above all constants are known) for which values need to be determined experimentally (parameter tuning).

1.6 Dissertation Contributions

The program laid out above—posing the image reconstruction problem, formulating it as a model selection problem amenable to an MDL-based solution, and devising a suitable optimization method—was done for the first time in groundbreaking work by Leclerc [48]. His work inspired other work in computer vision at the time (cf. Section 3.3), but is cited much less today.

Reviewing the state of the art in image reconstruction today, however, one finds that currently used and recently proposed algorithms solve seemingly less difficult problems than the algorithm developed by Leclerc. Either simpler image models are being used (e.g., piecewise constant functions) or the problem solving techniques require additional user input for success (e.g., good initializations with level set methods).

Given the continuing interest in image reconstruction, why is it that Leclerc's algorithm (or a more powerful one that might have been developed since) is not widely used today? It turns out that reproducing Leclerc's work and implementing his method is a major undertaking. I did not find accounts of other researchers who have implemented it, except for one recent publication [41]. Most of what is known about the algorithm therefore is from Leclerc's own presentation.

I implemented Leclerc's algorithm and studied its behavior extensively. My findings disagree with Leclerc's account on several points. Most importantly, his algorithm generates poor results when implemented exactly as published. In this work I develop and evaluate a number of modifications and show that these greatly improve the algorithm's performance, both in terms of quality of results as well as runtime. There are four main contributions.

Characterization of Performance. I present quantitative evaluations of the algorithm's reconstruction performance. Since it is practically impossible to obtain ground truth for natural images, I use synthetic test images to probe algorithm behavior and to discover weaknesses. In a separate study using natural images I evaluate how my derivative reconstruction algorithm performs on the related task of detecting image contours.

Robustness Improvements. I find two separate issues concerning algorithm robustness. One issue is that Leclerc's minimization procedure may fail to converge. The second issue is that the choice of values for the MDL criterion's parameters may be crucial for success or failure of the method, and that there is no choice of values which guarantees success for all problems. I suggest changes to the optimization method that alleviate this problem: solutions are found for a larger set of problems. This is a precondition for fixing parameter values that work for many problems.

Improved MDL Criterion. Without parameters in the MDL criterion, finding good parameter values would not be a problem. The two parameters in Leclerc's MDL criterion are introduced in the course of two approximation steps, which in turn are necessitated by computational concerns. I derive another MDL criterion which caters better to the computational concerns and which requires only one approximation step. The new criterion is more compelling and has only one parameter. Experimental results suggest that it is less sensitive to parameter choice.

Optimization. Leclerc's optimization method may fail to converge; if not, its slow convergence is prohibitive for many applications. In order to guarantee convergence I choose a different optimization method (a quasi-Newton line-search method). Runtime is addressed by a combination of techniques that together accelerate convergence by one to two orders of magnitude. Perhaps the most significant contribution is a new parameter in the MDL criterion which does not affect the calculated code length, but which offers some control over the *scaling* and the *conditioning* of the optimization problem.

1.7 Overview

The text is structured as follows. Chapter 2 recapitulates the derivation of Leclerc's MDL criterion and discusses the resulting optimization problem. Chapter 3 discusses related work. Chapter 4 introduces test problems, illustrates the general behavior of Leclerc's algorithm, and demonstrates parameter sensitivity.

Chapter 5 describes those elements of my new reconstruction algorithm that are not directly related to runtime performance. These elements include a new MDL criterion, new terms for the evaluation of discontinuities, and an improved "embedding", which is a central part of the solution method. Chapter 6 presents experimental evaluations of the new algorithm with a focus on quality of results. The main "takeaways" are that good solutions are found more reliably with the new embedding, and that the new MDL criterion results in an algorithm that is less sensitive with respect to choice of parameters.

The focus of Chapter 7 is runtime performance. Different methods for solving the optimization problem are described, including the relaxation method used by Leclerc, and the quasi-Newton method which I use in my derivative algorithm. An experimental runtime evaluation shows improvements of an order of magnitude or more. Additional experiments provide the interesting insight that the quasi-Newton method may perform *worse* than the relaxation method if no additional measures are taken.

Chapter 8 summarizes the main points and suggests directions for future work.

Chapter 2

Leclerc's MDL-based Reconstruction Algorithm

This chapter summarizes the derivation of Leclerc's reconstruction algorithm [48]. The algorithm produces reconstructions of gray-scale images as piecewise polynomial functions. A function is piecewise polynomial when there is a partitioning of the function's domain into regions such that the function is polynomial over each region. The polynomials of different regions may be of different degree. In this work an additional requirement is that the polynomial degree is not larger than some upper bound $p_{\max}$.

Leclerc considers any $p_{\max}$, that is, he actually derives a *family* of algorithms. The $p_{\max} = 0$ version of the algorithm produces piecewise constant reconstructions and is also referred to as the *piecewise constant* algorithm. Figure 4.1 (top row) in Chapter 4 shows examples of piecewise polynomial functions we consider here. The upper left image in Figure 4.1 shows a piecewise constant function, the other images in the top row are not piecewise constant.

Considering that algorithms for $p_{\max} > 2$ become less practical as the number of independent variables multiplies, and that piecewise quadratic ($p_{\max} = 2$) image descriptions are "appropriate for a large class of real images" ([48], page 92), I simplify the presentation by deriving the $p_{\max}=2$ algorithm directly. The $p_{\max}=2$ version of the algorithm is also referred to as the *piecewise polynomial* algorithm in this work.

2.1 Representation of Images and Reconstructions

An image is a piecewise constant function z from a bounded two-dimensional domain $\Omega \subset \mathbb{R}^2$ (Figure 2.1a) to a bounded set $Z \subset \mathbb{R}$ of gray levels,

$$z\colon \Omega \to Z. \tag{2.1}$$

Ω is partitioned into *cells*[2.1] $\{\Omega_i\}_{i\in I}$ (I is a set of cell labels), $\Omega = \cup_{i\in I}\Omega_i$, which are the smallest regions over which z is constant (Figure 2.1b). The value of z in cell i, z_i, presumably corresponds more or less directly to a measurement of intensity at an imaging sensor [39]. (This is just a more precise way of saying that an image is a "tiling of gray squares".)

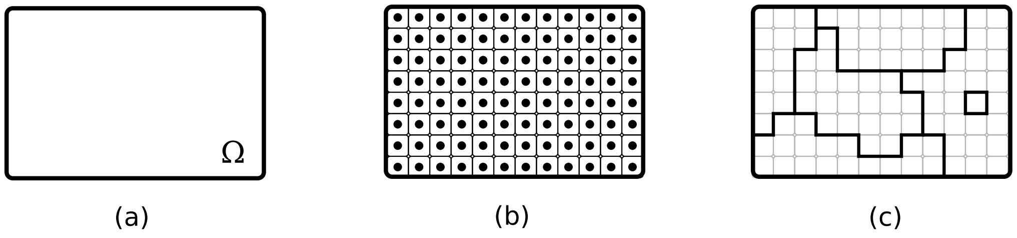


Figure 2.1. (a) Rectangular domain Ω. (b) The domain is partitioned into cells $\{\Omega_i\}$ with centers $\{\boldsymbol{x}_i\}$, $i \in I$. (c) Example partition of Ω into six regions. Note that region boundaries always go along cell boundaries.

Possible reconstructions u of z are defined over the same domain Ω, $u\colon \Omega \to \mathbb{R}$. Reconstructions are piecewise polynomial and the regions over which u may be smooth are unions of cells (Figure 2.1c). The polynomial degree of u may be different in different regions.

Since u is piecewise polynomial and the regions are unions of cells, we may completely describe any possible reconstruction u by tuples of polynomial coefficients $\{\boldsymbol{u}_i\}$, one tuple per cell. That is, with $\boldsymbol{u}_i = (u_{i,0}, u_{i,1}, \ldots, u_{i,5})$, u over cell i's domain Ω_i is

$$\begin{aligned} u|_{\Omega_i} \;=\;& u_{i,0} + u_{i,1}(x - x_i) + u_{i,2}(y - y_i) + \frac{u_{i,3}}{2}(x - x_i)^2 + u_{i,4}(x - x_i)(y - y_i) \\ &+ \frac{u_{i,5}}{2}(y - y_i)^2, \end{aligned} \tag{2.2}$$

2.1. I use the terms *cell* and *pixel* interchangeably.

where $\boldsymbol{x}_i = (x_i, y_i)$ are the coordinates of the center of cell i. When there are $|I|$ cells, u is represented by a coefficient vector $\boldsymbol{u}$ with $|I| \times 6$ entries[2.2].

Note that the coefficients of cell i are with respect to i's own coordinate system, which is centered on $\boldsymbol{x}_i$. Coefficient vectors $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ of two adjacent cells i and j, respectively, belong to the same smooth region when they share coefficients, that is, when they have the same coefficients in the same coordinate system. In regions in which u is of degree less than two, the higher-order coefficients are zero. For instance, for a cell i in a degree-1 region $u_{i,3} = u_{i,4} = u_{i,5} = 0$.

2.2 Simple Descriptions

The image z, as defined in (2.1), is a possible result of Leclerc's algorithm—it is piecewise polynomial with discontinuities consisting of cell boundary segments. The trivial "reconstruction" $u = z$ will in general not be the right answer, of course, and will fail to separate the random and non-random components of z. We need to state some additional properties that u must have in order to make the reconstruction problem well-posed [22, 68].

As is made clear in Chapter 1, u should be close to z with "appropriate complexity". We now develop an MDL criterion to select the best u. This requires that there is an encoding scheme suitable for encoding any u, and that the length of the encoding for any u can be computed. If we write $L(u)$ for the length (in bits) of some encoding of u and $L(z|u)$ for the length (in bits) of an encoding of z utilizing u, then the encoding length of z is

$$L(z) = L(u) + L(z|u). \tag{2.3}$$

2.2. This representation of u is akin to a finite element representation and I sometimes refer to the restriction $u|_{\Omega_i}$ of u to a single cell as a finite element.

The problem of finding the best reconstruction of z is then to find u^* that minimizes (2.3),

$$u^* = \operatorname*{argmin}_u L(z). \tag{2.4}$$

Note that the $|I| \times 6$ numbers in the vector $\boldsymbol{u}$ are an encoding of u, albeit not an efficient one (with this encoding $L(u)$ would be constant). To realize a more efficient encoding we exploit the piecewise smooth nature of u.

Let $\{R_j\}_{j\in J}$ denote the (maximal) regions over which u is smooth (J is a set of region labels), that is, $R_i \cap R_j = \varnothing$ when $i \neq j$, $\cup_{j\in J} R_j = \Omega$, and each R_j is a connected set of the form $R_j = \cup_{i\in I_j} \Omega_i$. Then $u|_{R_j}$ is a polynomial of degree zero, one, or two. We may fully describe u by specifying the regions $\{R_j\}$ plus a tuple of polynomial coefficients per region.

The next two sections develop formulas for computing the length of region-wise descriptions as a function of the coefficient vector $\boldsymbol{u}$. It is convenient to introduce another vector $\boldsymbol{r}$ with $|I|$ entries which holds all the residuals $\{r_i = z_i - u_{i,0}\}$. Our task then is to construct functions $L(\boldsymbol{u})$ and $L(\boldsymbol{r})$ that play the role of $L(u)$ and $L(z|u)$ in (2.3).

2.3 The Piecewise Constant Case

We begin with the simpler problem of piecewise *constant* reconstruction (cf. Section 5 in [48]). In this case $\boldsymbol{u}$ has just $|I|$ entries—the constant coefficients $\{u_{i,0}\}_{i\in I}$. From Section 1.4 we understand that if the residuals $\{r_i\}_{i\in I}$ have mean-square σ^2, the optimal encoding with minimal worst-case code length is

$$L(\boldsymbol{r}) = a \sum_{i\in I} \left[c + \left(\frac{r_i}{\sigma}\right)^2 \right]. \tag{2.5}$$

We want the other term, $L(\boldsymbol{u})$, to evaluate the code length of $\boldsymbol{u}$ in terms of regions and the polynomial coefficients per region. Let there be $|J|$ regions, $\{R_j\}_{j\in J}$. In the piecewise constant case there is always exactly one coefficient per region to encode. If we knew the regions, $L(\boldsymbol{u})$ could be calculated as[2.3]

$$L(\boldsymbol{u}) = L(|J|) + \sum_{j\in J} L(u|_{R_j}) + \sum_{j\in J} L(R_j), \tag{2.6}$$

where $L(u|_{R_j})$ is the length of an encoding of the coefficient describing u over R_j. One way to encode the regions R_j (Figure 2.1c) is to encode all the region boundaries using a chain code [27]. In a chain code encoding we specify one element of the boundary as the starting element and then give a sequence of successor elements by which to continue the curve. Because the number of possible successor elements at any point on the curve is small—three for interior cells—only slightly more than $\log_2 3$ bits per element are required. With this encoding

$$L(R_j) = L(\partial R_j) = l_s + l_e\,(|\partial R_j| - 1) \approx l_e\,|\partial R_j|, \tag{2.7}$$

where ∂R_j denotes the boundary of region R_j, and $|\partial R_j|$ the number of elements it consists of. The constant l_s is the description length required for specifying a starting element, and $l_e = \varepsilon + \log_2 3$ is the description length per boundary element (we add some small number $\varepsilon > 0$ since $\log_2 3$ is theoretically the shortest code length in the limit $|\partial R| \to \infty$).

With the approximation $L(R_j) \approx l_e |R_j|$ we can compute $\sum_{j\in J} L(R_j)$ in (2.6) by simply counting all cell boundary elements across which u is discontinuous. That is,

$$\sum_{j\in J} L(\partial R_j) = \sum_{j\in J} l_e\,|\partial R_j| = \frac{l_e}{2} \sum_{i\in I} \sum_{j\in N_i} \tilde{\delta}(u_{i,0} - u_{j,0}), \tag{2.8}$$

where $\tilde{\delta}\colon \mathbb{R} \to \{0,1\}$ is a “counting function” defined as

$$\tilde{\delta}(x) = \begin{cases} 0 \text{ if } x = 0 \\ 1 \text{ if } x \neq 0 \end{cases},$$

and N_i denotes the cells adjacent to cell i.

2.3. Note that our notation deviates here from Leclerc’s [48]. Leclerc writes $L(\boldsymbol{u})$ to denote the length for an encoding of the image z when $\boldsymbol{u}$ is used as the model.

Finally, Leclerc drops the first two terms in (2.6) and approximates $L(\boldsymbol{u})$ as

$$\begin{aligned} L(\boldsymbol{u}) &= L(|J|) + \sum_{j\in J} L(u|_{R_j}) + \frac{l_e}{2}\sum_{i\in I}\sum_{j\in N_i}\tilde{\delta}(u_{i,0}-u_{j,0}) \\ &\approx \frac{b}{2}\sum_{i\in I}\sum_{j\in N_i}\tilde{\delta}(u_{i,0}-u_{j,0}), \end{aligned} \tag{2.9}$$

and suggests that we can account for the dropped terms by defining the parameter b appropriately. He writes

> "... b is the sum of (1) the number of bits required to encode each element in the chain code and (2) the number of bits required to encode the constant intensity and starting element, divided by the average region-boundary length." ([48], page 80)
>
> "..., for 4-connected elements, b should be at least as large as $\log_2 3$, but not much more than two." (ibid, page 81)

In summary, we have derived the following approximate code length for z using a piecewise constant reconstruction $\boldsymbol{u}$,

$$L(z) = \min_{\boldsymbol{u}} \sum_{i\in I}\left\{a\left(\frac{z_i-u_{i,0}}{\sigma}\right)^2 + a\,c + \frac{b}{2}\sum_{j\in N_i}\tilde{\delta}(u_{i,0}-u_{j,0})\right\}, \tag{2.10}$$

where c represents the constant terms in (2.5) and may be dropped from the expression when only the minimizer $\boldsymbol{u}^*$ of (2.10) is of interest.

2.4 The Piecewise Polynomial Case

We now add extra degrees of freedom to the reconstruction problem and allow u to be piecewise polynomial ($p_{\max} = 2$). Remember that $\boldsymbol{u}_i$ in this case represents a set of six polynomial coefficients, $\boldsymbol{u}_i = (u_{i,0}, u_{i,1}, u_{i,2}, u_{i,3}, u_{i,4}, u_{i,5})$. The calculation of the code length $L(u)$ in (2.3) must be modified accordingly: the number of coefficients that must be encoded for region j—one, three, or six—depends on the polynomial degree of $u|_{R_j}$. Leclerc assesses a fixed encoding cost l_c per coefficient, so the term

$L(u|_{R_j})$ in (2.6) becomes proportional to the number of coefficients (this neglects the small cost for encoding the number of coefficients itself). Let $\boldsymbol{u}_{i_j}$ be the coefficient vector of some cell Ω_{i_j} that belongs to region R_j, then

$$\begin{aligned} L(\boldsymbol{u}) &= L(|J|) + \sum_{j\in J} l_c \left[1 + 2 \max_{k\in\{1,\ldots,5\}} \tilde{\delta}(u_{i_j,k}) + 3 \max_{k\in\{3,4,5\}} \tilde{\delta}(u_{i_j,k}) \right] \\ &\quad + \sum_{j\in J} L(R_j) \qquad (2.11) \\ &\approx \sum_{i\in I} d \left[1 + 2 \max_{k\in\{1,\ldots,5\}} \tilde{\delta}(u_{i,k}) + 3 \max_{k\in\{3,4,5\}} \tilde{\delta}(u_{i,k}) \right] + \sum_{j\in J} L(R_j) \qquad (2.12) \end{aligned}$$

The term $\max_{k\in\{1,\ldots,5\}}\tilde{\delta}(u_{i_j,k})$ in (2.11) determines whether the polynomial order in region j is at least linear, in which case two linear coefficients must be encoded. The second max-term determines if $u|_{R_j}$ is quadratic, in which case three more coefficients need to be encoded. Note that in (2.11) the first sum is over all regions, whereas in (2.12) it is over all cells. Leclerc says the new parameter "d is the number of bits required to encode a nonzero coefficient, divided by the average region size" ([48], page 88). This approximation brings the length formula into an efficiently computable form. It is also necessitated by the optimization method (this becomes clear in the following section).

The term for $L(R)$ in (2.6) must be modified as well for the piecewise polynomial case. It is no longer sufficient to compare the zero-order coefficients of adjacent cells to determine a discontinuity. We now must also consider discontinuities in the derivatives of u. Only when u and all its derivatives are continuous across the boundary of two adjacent cells i and j do i and j belong to the same region.

Some new notation helps to make this more precise. Let $\Delta_{i,j,k}(\boldsymbol{u})$, or simply $\Delta_{i,j,k}$, denote the difference between the k-th derivatives of $u|_{\Omega_i}$ and $u|_{\Omega_j}$ at the point $\overline{\boldsymbol{x}_{i,j}} = \frac{1}{2}(\boldsymbol{x}_i + \boldsymbol{x}_j)$ on the boundary between Ω_i and Ω_j, :

$$\begin{aligned} \Delta_{i,j,0} &= u|_{\Omega_i}(\overline{\boldsymbol{x}_{i,j}}) - u|_{\Omega_j}(\overline{\boldsymbol{x}_{i,j}}) \\ \Delta_{i,j,1} &= \frac{\partial u|_{\Omega_i}}{\partial x}(\overline{\boldsymbol{x}_{i,j}}) - \frac{\partial u|_{\Omega_j}}{\partial x}(\overline{\boldsymbol{x}_{i,j}}) \\ \Delta_{i,j,2} &= \frac{\partial u|_{\Omega_i}}{\partial y}(\overline{\boldsymbol{x}_{i,j}}) - \frac{\partial u|_{\Omega_j}}{\partial y}(\overline{\boldsymbol{x}_{i,j}}) \end{aligned}$$

$$\begin{aligned}
\Delta_{i,j,3} &= \frac{\partial^2 u|_{\Omega_i}}{\partial x^2}(\overline{\boldsymbol{x}_{i,j}}) - \frac{\partial^2 u|_{\Omega_j}}{\partial x^2}(\overline{\boldsymbol{x}_{i,j}}) \\
\Delta_{i,j,4} &= \frac{\partial^2 u|_{\Omega_i}}{\partial x \partial y}(\overline{\boldsymbol{x}_{i,j}}) - \frac{\partial^2 u|_{\Omega_j}}{\partial x \partial y}(\overline{\boldsymbol{x}_{i,j}}) \\
\Delta_{i,j,5} &= \frac{\partial^2 u|_{\Omega_i}}{\partial y^2}(\overline{\boldsymbol{x}_{i,j}}) - \frac{\partial^2 u|_{\Omega_j}}{\partial y^2}(\overline{\boldsymbol{x}_{i,j}}).
\end{aligned}$$

With these terms a cell boundary element $\partial\Omega_i \cap \partial\Omega_j$ is a region boundary element when $\Delta_{i,j,k} \neq 0$ for any $k = 0, \ldots, 5$, or, equivalently, when $\max_k \tilde{\delta}(\Delta_{i,j,k}) = 1$. Hence, the last sum in (2.11) becomes

$$\sum_{j \in J} L(R_j) = \frac{l_e}{2} \sum_{i \in I} \sum_{j \in N_i} \max_{k \in \{0, \ldots, 5\}} \tilde{\delta}(\Delta_{i,j,k}). \tag{2.13}$$

Curiously, instead of (2.13), Leclerc writes down a different sum (expression (12) in [48], page 88). Adapted to the notation used here and for the case $p_{\max} = 2$ Leclerc's sum is

$$\sum_{j \in J} L(R_j) \approx \frac{b}{2} \sum_{i \in I} \sum_{j \in N_i} \left[\tilde{\delta}(\Delta_{i,j,0}) + 2 \max_{k \in \{0,1,2\}} \tilde{\delta}(\Delta_{i,j,k}) + 3 \max_{k \in \{0, \ldots, 5\}} \tilde{\delta}(\Delta_{i,j,k}) \right]. \tag{2.14}$$

Leclerc says the following about (2.14):

> ”The second term in the description length is proportional to the number of discontinuities in the function and its derivatives (up to order $p_{\max}$) between adjacent elements.” ([48], page 88)

Note that (2.13) is the direct generalization of (2.8) to the piecewise polynomial case. That is, it counts the number of pairs of adjacent cells that belong to different regions, and the encoding length of the region boundaries is proportional to the total boundary length (chain code encoding), just like in the piecewise constant case. Leclerc's sum (2.14), on the other, is not proportional to total boundary length, and Leclerc does not explain how to interpret it as the length of an encoding of the region boundaries.

With (2.12) and (2.14) $L(z)$ for the piecewise polynomial case then becomes

$$\begin{aligned} L(z) &= \min_{\boldsymbol{u}} \sum_{i\in I} \Bigg\{ \frac{1}{2\log 2}\left(\frac{z_i - u_{i,0}}{\sigma}\right)^2 \\ &\quad + d\left[1 + 2\max_{k\in\{1,\dots,5\}} \tilde{\delta}(u_{i,k}) + 3\max_{k\in\{3,4,5\}} \tilde{\delta}(u_{i,k})\right] \\ &\quad + \frac{b}{2}\sum_{j\in N_i}\left[\tilde{\delta}(\Delta_{i,j,0}) + 2\max_{k\in\{0,1,2\}} \tilde{\delta}(\Delta_{i,j,k}) + 3\max_{k\in\{0,\dots,5\}} \tilde{\delta}(\Delta_{i,j,k})\right]\Bigg\} \qquad (2.15) \\ &= \min_{\boldsymbol{u}} F_0(\boldsymbol{u}). \end{aligned}$$

2.5 Solving the Optimization Problem

A value $\boldsymbol{u}^*$ is called a *global minimizer* of F_0 if $F_0(\boldsymbol{u}^*) \leq F_0(\boldsymbol{u})$ for all $\boldsymbol{u}$. In the derivation leading up to expression (2.15) we have reduced the piecewise polynomial reconstruction problem to the problem of finding the global minimizer of F_0. Finding the global optimizer of F_0 is difficult because F_0 has none of the properties such as convexity or separability that standard methods for solving such problems require. The only sure method of finding a global minimizer—exhaustive search—is not feasible.

2.5.1 Continuation Method

As a way out of this quandary, instead of trying to solve $\min_{\boldsymbol{u}} F_0(\boldsymbol{u})$ directly, Leclerc uses what is called a *continuation method* [21, 48]. In a continuation method the problem of interest is replaced by a family of related problems, which are easier to solve individually. In the context of solving $\min_{\boldsymbol{u}} F_0(\boldsymbol{u})$, the objective function $F_0(\boldsymbol{u})$ is replaced by a family of objective functions $F_0(\boldsymbol{u}, s)$, $s > 0$, which must meet the following requirements:

1. $F_0(\boldsymbol{u}, s)$ is smooth,
2. for $s \geq s_{\text{large}} \gg 1$ $F_0(\boldsymbol{u}, s)$ is convex with a unique (hence global) minimum, and

3. $F_0(\boldsymbol{u}) = \lim_{s\to 0} F_0(\boldsymbol{u}, s)$.

The first requirement comes from the optimization methods we want to use for solving $\min_{\boldsymbol{u}}\ F_0(\boldsymbol{u},\ s)$. The second requirement will become clear momentarily, and the third requirement means that $F_0(\boldsymbol{u},\ s)$ resembles $F_0(\boldsymbol{u})$ more and more closely as the *continuation parameter* s approaches zero. It is therefore said that the objective function $F_0(\boldsymbol{u})$ is *embedded* in a family of functions $F_0(\boldsymbol{u}, s)$, and the particular choice of $F_0(\boldsymbol{u}, s)$ is called the *embedding*.

The continuation method then consists of solving a sequence of minimization problems $\min_{\boldsymbol{u}} F_0(\boldsymbol{u}, s)$ for a decreasing sequence of continuation parameter values s_0, s_1, s_2, ..., s_f. Solving the initial problem with $s_0 \geq s_{\text{large}}$ yields a minimizer $\boldsymbol{u}^{*,0}$ that is a global minimizer because of the second requirement above. For each following problem $\min_{\boldsymbol{u}} F_0(\boldsymbol{u}, s_k)$, $k > 0$, the solution $\boldsymbol{u}^{*,k-1}$ from the previous problem is used as the starting point. The corresponding sequence of minimizers $\boldsymbol{u}^{*,0}$, $\boldsymbol{u}^{*,1}$, $\boldsymbol{u}^{*,2}$, ..., $\boldsymbol{u}^{*,f}$ converges to a minimizer $\boldsymbol{u}^*$ of $F_0(\boldsymbol{u})$.

As the continuation parameter decreases, $F_0(\boldsymbol{u}, s)$ develops more and more local minima and the problem of finding the global minimizer becomes increasingly difficult. The idea is that by starting with a problem $F_0(\boldsymbol{u}, s_0)$ that is easy enough so that a global minimizer may be found, and by using the minimizer $\boldsymbol{u}^{*,k-1}$ as the starting point for the next minimization, the search is always started from a point close to the global minimum of $F_0(\boldsymbol{u},\ s_k)$ and will converge to it. The continuation process therefore "traces a path" through the search space from $\boldsymbol{u}^{*,0}$ to the minimizer $\boldsymbol{u}^{*,f}$ of the final problem $\min_{\boldsymbol{u}} F(\boldsymbol{u}, s_f)$. Leclerc writes[2.4] "For an ideal embedding, there will be no bifurcations along this path, and the value of $\boldsymbol{u}^{*,k}$ for a sufficiently large k (and hence a sufficiently small s_k) will be close or equal to the global minimum of $F_0(\boldsymbol{u})$." ([48], page 82). Below I occasionally refer to this path as "the continuation path" to a solution.

2.4. The mathematical symbols in this quote are translated to match the notation used in this work.

2.5.2 Constructing an Embedding for F_0

To meet the first requirement for the embedding, two non-smooth functions occurring in F_0 (see (2.15)), max and δ, need to be replaced by smooth functions. To that end we first rewrite F_0 as

$$F_0(\boldsymbol{u}) = \sum_{i\in I}\left\{\frac{1}{2\log 2}\left(\frac{u_{i,0}-z_i}{\sigma}\right)^2 + d\left[1+2\left(1-\prod_{k=1}^{5}\delta(u_{i,k})\right)+3\left(1-\prod_{k=3}^{5}\delta(u_{i,k})\right)\right]\right.$$
$$\left.+\frac{b}{2}\sum_{j\in N_i}\left[1-\delta(\Delta_{i,j,0})+2\left(1-\prod_{k=0}^{2}\delta(\Delta_{i,j,k})\right)+3\left(1-\prod_{k=0}^{5}\delta(\Delta_{i,j,k})\right)\right]\right\} \quad (2.16)$$

with $\delta(x) = 1 - \tilde{\delta}(x)$, where we made use of the relationships

$$\max_k \tilde{\delta}(x_k) = 1 - \min_k \delta(x_k) \quad (2.17)$$

and $\min_k \delta(x_k) = \prod_k \delta(x_k)$. This removes the max operation. Next the δ-functions are replaced by smooth functions $e(x, s)$ with the property $\lim_{s\to 0} e(x, s) = \delta(x)$ for the third embedding requirement. Leclerc uses exponentials and makes the substitutions

$$\delta(u_{i,k}) \;\rightarrow\; \exp\left(-u_{i,k}^2/(f\,s\,\sigma)^2\right) \quad (2.18)$$
$$\delta(\Delta_{i,j,k}) \;\rightarrow\; \exp\left(-\Delta_{i,j,k}^2/(s\,\sigma)^2\right). \quad (2.19)$$

The new parameter f goes unmentioned ([48], page 88). Detailed expressions for the objective function resulting from this embedding is given in the Appendix, Section A.1.

To see that the second embedding requirement is met, consider the exponential terms (2.18) and (2.19) as a function of s. As s tends to infinity, the value of the exponentials tend to one, and we see (replace the δ-terms in (2.16) by ones) that

$$F_0(\boldsymbol{u}, \infty) = \lim_{s\to\infty} F_0(\boldsymbol{u}, s) = \frac{1}{2\log 2}\sum_{i\in I}\left(\frac{u_{i,0}-z_i}{\sigma}\right)^2,$$

which is convex. We also see that the minimizer of $F_0(\boldsymbol{u}, \infty)$ is $\boldsymbol{u}_i = (z_i, 0, 0, 0, 0, 0)$, $i \in I$, and this value should be used as the starting point for the initial problem of the continuation method.

2.5.3 Structure of the Optimization Algorithm

Chapter 7 discusses methods for solving the problem $\min_{\boldsymbol{u}} F_0(\boldsymbol{u}, s)$. These are iterative methods that take an initial estimate of a minimizer—a point $\boldsymbol{u}^0$ in the search space—and generate a sequence of points $\boldsymbol{u}^1$, $\boldsymbol{u}^2$, ... that converge to a minimizer $\boldsymbol{u}^*$. The optimization scheme therefore consists of two nested loops. The outer loop implements the continuation method and in each iteration s gets reduced by a constant factor r, $0 < r < 1$,

$$s_{i+1} = r\, s_i. \tag{2.20}$$

The inner loop solves $\min_{\boldsymbol{u}} F_0(\boldsymbol{u}, s_i)$ for fixed s_i.

2.6 When $\boldsymbol{\sigma}$ is Unknown

In Section 2.3 above I introduced the term $a \sum_{i \in I} \left(\frac{z_i - u_{i,0}}{\sigma}\right)^2$ as the code length for the residuals. This term is justified when two assumptions hold, namely, when the variance σ^2 of the residuals is known, and when it is constant across the image.

For scenes satisfying the piecewise smoothness assumption (see Section 1.2), σ quantifies noise introduced by the imaging process and the assumption of constant variance is appropriate. When σ is constant but unknown, it may be inferred by running the algorithm several times with different values for σ, until a shortest code length $L(z)$ for the image is obtained (bisection search suggests itself).

With scenes violating the piecewise smoothness assumption, constant σ may no longer be assumed. In this situation, the residuals are dominated by texture and will have different variance in different parts of the image. This suggests treating σ as a *piecewise-wise constant* unknown ([48], Section 6.3), which requires $|R|$ numbers to describe (σ_r for each region $r \in R$).

The description length formula then needs to account for the encoding of the σ_r-values, that is, the values for parameters b and d in (2.15) need to be adjusted accordingly. Appendix A.3 gives details on an algorithm for the case of unknown, non-constant variance.

2.7 Some Remarks on Lecler's Work

Before discussing related work I pause to highlight some aspects of Leclerc's work that I will return to in the following chapters.

2.7.1 A Parameter-Free Problem Should Have a Parameter-Free Solution

The question "Which piecewise polynomial function u minimizes the code length $L(z)$ in (2.3)?" does not involve any parameters except for $p_{\max}$. Its answer should therefore not depend on any parameters other than $p_{\max}$.

However, Leclerc's objective function F_0 in (2.15) *does* include two parameters, b and d, annihilating one advantage of the MDL-approach. Those parameters are introduced in the course of approximations to more accurate code-length formulas. Note how the optimization strategy discussed in Section 2.5 necessitates some approximations: in the context of the continuation method, $L(\boldsymbol{u})$, as in (2.11), cannot be used since a separation of u into regions is not well defined for most values s_0, s_1, $s_2, \ldots$. With the approximation (11), on the other hand, $F_0(\boldsymbol{u})$ generalizes to $F_0(\boldsymbol{u}, s)$ seamlessly.

In Chapter 5 I derive a new MDL criterion for this problem, where the underlying encoding is a run-length encoding of $\boldsymbol{u}$. Fewer approximations are required to derive an objective function amenable to the optimization strategy described in Section 2.5, and my objective function includes only one parameter instead of two.

2.7.2 Finding Good Solutions

While we wish to find a global minimum of the objective function F_0, the continuation method described in Section 2.5.1 is *not* guaranteed to find one. Empirically it often finds *good* solutions, however, meaning that the solution found is qualitatively very similar to a globally optimal solution. As I show in Chapters 4 and 6, the quality of solutions found depends the values of b and d in F_0, the accuracy to which the intermediate solutions $\boldsymbol{u}^{*,0}$, $\boldsymbol{u}^{*,1}$, ... are computed, and on details of the embedding (e.g., (2.18) and (2.19)). Much of my work is aimed at improving the quality of solutions by making changes in these areas.

2.7.3 The Objective Function with Correct Boundary Term

In Section 2.4 I argued that (2.13) and not (2.14) is the correct estimate for the code-length of a chain-code encoding of the region boundaries. The objective function F_2 that would result with (2.13) reads

$$\begin{aligned} F_2(\boldsymbol{u}) \;=\; \sum_{i\in I} \Bigg\{ & \frac{1}{2\log 2}\left(\frac{z_i - u_{i,0}}{\sigma}\right)^2 + d\left[1 + 2\max_{k\in\{1,\ldots,5\}} \tilde{\delta}(u_{i,k}) + 3\max_{k\in\{3,4,5\}} \tilde{\delta}(u_{i,k})\right] \\ & + \frac{b}{2}\sum_{j\in N_i} \max_{k\in\{0,\ldots,5\}} \tilde{\delta}(\Delta_{i,j,k}) \Bigg\}. \end{aligned} \tag{2.21}$$

Note that (2.21) accounts for the encoding of the coefficients in the d-term. In Leclerc's criterion (2.15), on the other hand, it is accounted for in the d-term *and* the b-term.

2.7.4 Stability of Region Boundaries

An interesting feature of Leclerc's algorithm is that it produces stability scores for all boundary elements in the reconstruction. Some boundaries or parts of a boundary are more prone to change with small changes in the input or in the objective function parameters than other boundaries. The stability score is an indicator of this sensitivity and it correlates with the contrast-to-noise ratio of a boundary element ([48], pages 83 ff). The stability scores are used in Section 6.5.

Chapter 3

Related Work

3.1 Related Problems

A solution to the image reconstruction problem includes the solutions to a few closely related problems such as image segmentation [34], boundary detection [15], and image denoising [72]. Considering these problems in their own right may lead to algorithms which perform better in some sense, or algorithms that are applicable in situations where the piecewise smoothness assumption does not hold.

3.1.1 Image Segmentation

The problem of image segmentation is to partition an image into regions. The reconstruction $\boldsymbol{u}$ that Leclerc's algorithm generates includes a segmentation of the image. Some authors consider segmentation the main purpose of image reconstruction [12, 46]. Applications may require other criteria than smoothness for determining regions [70], or they may allow for utilization of prior knowledge about expected region shape [19].

There are a few general algorithmic approaches to segmentation. The most prominent are region growing [46, 83], the evolution of curves represented as level sets [17, 61], and graph partitioning methods [74].

3.1.2 Detection of Contours

Contours in images are important features for recognition tasks [24, 65]. Algorithms for detecting contours are therefore of interest in their own right [56].

Capturing the intuitive notion of *contour* in precise terms is quite difficult. A conceptually simple substitute with a long tradition in computer vision is *discontinuity*—discontinuity in intensity or some other visual quantity.

There are two general kinds of algorithms for detecting discontinuities aside from reconstruction algorithms. One approach is to first smooth an image by convolving it with a linear filter, then to look for large gradients in the result [15]. (Algorithms in this category are commonly referred to as "edge detectors"). The smoothing step reduces noise and is to avoid detection of "false discontinuities". One difficulty in this approach is deciding what level of smoothing is appropriate, so that "true discontinuities" are not missed or distorted.

Algorithms following the second approach try to locally fit "templates" of step edges to the image data (i.e., to cutouts like Figure 1.2, left) [38]. Locations of good-enough fit are labeled as discontinuities.

More recent approaches try to detect changes in local image statistics rather than just in intensity [69]. They use machine learning ideas and try to *learn* to detect contours from training data. More specifically, a classifier is trained to distinguish contour and non-contour pixels based on local image statistics [56].

3.1.3 Image Denoising

Algorithms for denoising an image take an image z and produce a new image u which contains fewer noise artifacts than z. This is not the same problem as the image reconstruction problem since discontinuities and differential structure need not be made explicit.

A popular approach to noise removal results from a problem formulation that is very similar to the Mumford-Shah formulation of the reconstruction problem (Section 3.2.1). In this *total variation* based approach Rudin, Osher and Fatemi pose the problem of minimizing

$$\int_\Omega |\nabla u|\, d\omega \quad \text{subject to} \int_\Omega (u-z)^2\, d\omega = \sigma^2,$$

where σ^2 is the known noise variance. The name of the method derives from the term $\int_\Omega |\nabla u|\, d\omega$, which measures the total variation of u. In other words, the problem is to find the function of lowest total variation which is compatible with the noise level. The problem may be shown to be equivalent to minimizing

$$\int_\Omega (|\nabla u| + \lambda\, (u-z)^2)\, d\omega,$$

when λ is chosen appropriately [16].

Another important denoising approach uses a wavelet decomposition of the image [54], and reduces the wavelet coefficients in the detail sub-bands by some rule [23, 18]. The inverse wavelet transform then gives the noise-reduced image.

3.1.4 The Piecewise Smoothness Constraint

Finding a piecewise smooth approximation of an intensity image is just one application of the piecewise-smoothness constraint. In fact, it applies to other areas of computer vision such as reconstruction of visible surfaces from sparse data [32, 76], the estimation of motion/optical flow [7, 80], or the decomposition of an object's silhouette contour [4, 10, 43].

For the problem of estimating optical flow, for example, the intensity data z becomes time-dependent. The *brightness conservation assumption*

$$z(x,y,t) = z(x + u\,\delta t, y + v\,\delta t, t + \delta t)$$

expresses how intensities in consecutive frames are related (δt is a small time increment). The object of interest here is the *flow field* with components u and v. With independently moving objects in the scene, or with a moving camera and objects at varying distances to the camera, u and v are piecewise smooth functions.

3.2 Other Approaches to Reconstruction

3.2.1 Mumford-Shah and Blake-Zisserman

Around the same time in the mid 1980s, both Mumford and Shah [9, 58] as well as Blake and Zisserman [9, 13] propose the piecewise smooth reconstruction problem in the following way. Let u denote a possible solution defined over Ω and that is differentiable over $\Omega \setminus \Gamma_u$. Consider the following functional,

$$E_{\lambda,\alpha}(u) = \int_{\Omega} (u-z)^2 \, d\omega + \int_{\Omega \setminus \Gamma_u} \lambda^2 \, (\nabla u)^2 \, d\omega + \alpha \int_{\Gamma_u} d\gamma \, . \tag{3.1}$$

The solution to the reconstruction problem is the function u^* which minimizes $E_{\lambda,\alpha}$.

The functional (3.1) is called the *energy* of u and plays the same role as the MDL criterion in the MDL-based approach. Note that $\Gamma_u \subset \Omega$ depends on u: Γ_u are the points at which u is discontinuous or not differentiable. The first term in (3.1) ensures closeness to the data z. The term $\lambda^2(\nabla u)^2$ penalizes "roughness", for solutions are to be piecewise *smooth*. The last term is proportional to the length of Γ_u when Γ_u consists of a set of curves. Its purpose is to penalize long and "rough" boundaries in u. The parameters λ and α weight the penalty terms and offer some control over the degree of smoothness (λ), the sensitivity to noise (α), and the contrast-to-noise ratio threshold at which discontinuities are detected (α and λ) [13].

Blake and Zisserman motivate the use of (3.1) for several computer vision problems [13]. In their work the continuous formulation (3.1) is used to study the behavior of solutions with respect to the parameters λ and α. Discrete versions of (3.1) are used to develop algorithms. One discretization leads to an algorithm simulating a "weak membrane", which may be considered Blake and Zisserman's solution to the reconstruction problem.

Mumford and Shah, on the other hand, concern themselves with mathematical questions such as the appropriate space of possible solutions, how, precisely, to define the last term in (3.1), the existence and uniqueness of minimizers, and so on [59]. Their work generated interest in the mathematics community and gave rise to a number of theoretical works [1, 2, 62]. But finding solutions to the minimization problem (3.1) turned out to be difficult, and a striking gap between theoretical and algorithmic progress developed [12]. Most published algorithms treat the special case $\lambda \to \infty$ [17, 47, 66], which Mumford and Shah call the *cartoon limit*: in this case solutions are piecewise constant. More recent works expressly treat the piecewise *smooth* case $\lambda < \infty$ [77, 12, 31, 67].

3.2.2 Reconstruction as Bayesian Inference

Other authors cast the reconstruction problem in probabilistic terms. In this view not only the observed image z is a realization of a stochastic process: the unobserved image u is as well. The problem then becomes that of constructing a probability distribution over the space of possible reconstructions u, and of finding the most probable u. Well-posedness in this framework means that there is a unique most probable u.

Geman and Geman established these ideas in a seminal work [30]. In their formulation the unobserved variable (reconstruction) consists of both an *intensity process* and a *line process*. A realization of the intensity process gives the unobserved intensities for each cell, the line process gives the discontinuities between cells (cf. Figure 2.1). Using piecewise continuity and other assumptions, they construct a prior distribution for the unobserved variable, and using a simple model of the imaging process they derive a probability distribution for the observed image. Applying Bayes rule then gives the posterior distribution of the unobserved image given the observed image.

The distributions used in their work have the structure of Markov Random Fields (MRF).

To find solutions Geman and Geman propose a Markov chain Monte Carlo algorithm for sampling the posterior distribution [30]. Such randomized algorithms perform poorly compared to deterministic minimization algorithms [8], and this aspect of their work is of lesser interest today. The recent development of fast algorithms for approximate energy minimization [11] has rekindled interest in MRF models for vision and image processing [14], however.

3.2.3 Discussion

It is reasonable to ask how the Mumford-Shah formulation (3.1) and Leclerc's formulation differ in capturing the piecewise smoothness constraint. Leclerc's formulation restricts the concept to piecewise polynomial. What biases are implicit in the energy-based formulation?

Blake and Zisserman discuss discrete versions of the energy-based formulation and admit some shortcomings [13]. The first order discretization leading to their weak membrane algorithm, for instance, is biased towards flat reconstructions. As a consequence there is an upper bound on gradient magnitude of a solution u. It is not clear whether Blake and Zisserman considered the piecewise polynomial model, which does not have this problem, when they conclude "that a spline under weak continuity constraints (or the equivalent MRF) is not quite the right model. But it is the best that is available at the moment." ([13], page 15).

The MDL approach can be shown to be equivalent to a Bayesian inference approach [48]. Communication theory tells how to construct short codes given probabilities of messages. The other way—constructing probabilities given (lengths of) codes—is also possible: by the Kraft inequality [20]

$$\sum_i 2^{-L(u^i)} \leq 1,$$

message u^i may be assigned the probability $p(u^i) = 2^{-L(u^i)}/\sum_i 2^{-L(u^i)}$. From the probabilistic point of view $L(\boldsymbol{u})$ in (2.6) implicitly defines a prior on possible reconstructions, whereas $L(z)$ in (2.15), the encoding length of the image given the model $\boldsymbol{u}$, implicitly defines the posterior. Leclerc comments on the equivalence of MDL and probabilistic approaches. In his view the MDL approach holds greater appeal because intuition about processes underlying observations are more easily expressed by *describing* models and conceiving encodings for their descriptions, than by assigning probabilities to models.

3.3 MDL-based Approaches to Problems in Computer Vision

Leclerc was among the first to suggest the MDL principle as a guiding principle for solving inference problems in computer vision [48, 49]. I review some of the works where his influence is most apparent. The algorithms have in common that a solution is found by minimizing an MDL criterion.

Kanungo, Dom, Niblack and Steele develop an algorithm that also produces piecewise polynomial image descriptions [46]. Their solutions may include only jump discontinuities, however. The MDL-criterion is developed along the same lines as Leclerc's but approximation steps and the introduction of parameters into the criterion are avoided. A greedy region-merging algorithm is used for generating reconstructions. The algorithm starts with a gross over-segmentation of the image, with regions of one to three pixels in size. The algorithm proceeds by iteratively merging the respective pair of adjacent regions whose merging reduces the description length the most. In the first merging phase only constant regions are created ($p = 0$). In each subsequent phase p is increased and merging steps may create regions of maximum degree p. The algorithm terminates after $p_{\max}+1$ phases when the description length may not be reduced any further.

It is the region-merging strategy that allows for the use of a more accurate MDL criterion. Since every step of the merging process yields an explicit segmentation, region sizes and boundary lengths are always known. On the other hand, there is little reason to expect the region-merging will yield solutions that are close to optimal. Errors committed early in the process (i.e., merging of regions that should not be merged) may not be corrected later.

Pan uses a variant of Kanungo et al's algorithm for segmenting remote sensing data [64], and Lee for segmenting microscopic images of aluminum grains as well as remote sensing data [52]. Ivanovska uses a variant of Leclerc's algorithm for analyzing histological images [40]. All three authors consider only piecewise constant reconstructions.

Zhu and Yuille draw inspiration from Leclerc's and other works for devising their "region competition" segmentation algorithm [83]. The algorithm starts with growing regions from many small "seed regions". It then alternates between moving region boundaries and merging adjacent regions. All the steps of the algorithm are guided by an MDL criterion and the algorithm terminates at a local minimum. The objective function is a simplification of Leclerc's and contains one parameter.

Avoiding free parameters in their algorithm is the main motivation for Galland et al. to consider an MDL approach for segmenting Synthetic Aperture Radar images [28]. Their algorithm is different in that regions are described by polygons and the length of region descriptions may be computed accurately from the number of polygon vertices and segments. The algorithm starts like Kanungo et al.'s with an over-segmentation into many small regions. To modify a segmentation the algorithm uses merging of regions, shifting of polygon vertices, and removal of polygon vertices. Again, the algorithm terminates when neither of these operations would bring about a further reduction in description length. The reconstructions are piecewise constant with region-specific noise parameters [28].

Chapter 4

How Leclerc's Algorithm Behaves

In this chapter I first describe the general behavior of Leclerc's algorithm in Section 4.2. In the following two sections I describe two problems I encountered while working with the algorithm. First, Section 4.3 demonstrates that different images require different parameter choices for the algorithm to produce satisfactory solutions. This is at odds with Leclerc's own characterization of his algorithm [48].

A more subtle algorithmic aspect affecting the quality of results is to what accuracy intermediate solutions are computed in the continuation process. In his dissertation Leclerc mentions the continuation step size (r) as an important factor [50]. In Section 4.4 I add the relative accuracy to which coefficients are computed as another factor.

The synthetic test images I use might seem simplistic at first, but they were chosen carefully and each one poses a different challenge to a reconstruction algorithm.

4.1 Test Images

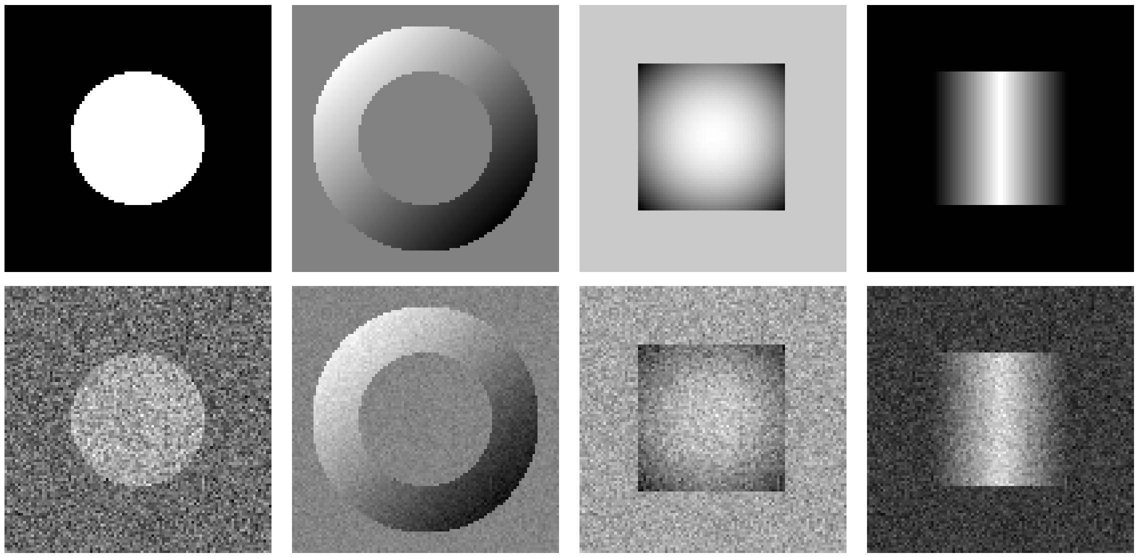


Figure 4.1. Test cases 1 to 4 (bottom row) and the corresponding generating images used to generate the test images (top row).

In order to study the behavior of Leclerc's algorithm, I use synthetic test images which are generated in a way that exactly matches the algorithm's encoding scheme (cf. Figure 4.1). That is, zero-centered Gaussian i.i.d. noise is added to the samples of piecewise polynomial functions. This way we have good known solutions, namely the respective functions we sample (Figure 4.1, top row). I use $\boldsymbol{u}^g$ to denote the coefficient vectors of these *generating images*.

Note that the generating image is the *expected* optimal solution, which is close to but in general not identical to the optimal solution for any realization of the noise process. The polynomial coefficients of the optimal solution will differ slightly from $\boldsymbol{u}^g$ in a way that depends on the realization of the noise process. This makes $\boldsymbol{u}^g$ a good solution to compare against. If the objective function value for $\boldsymbol{u}^g$ is lower than the value for the algorithm's solution then we know that the algorithm did not find the optimal solution.

The four synthetic test images in Figure 4.1 pose different challenges to a reconstruction algorithm. The first image in the top row is similar to a test image used by

Leclerc (see Fig. 3 in [48]); the generating image is piecewise constant. The challenging aspect of this test image is that the ratio of the difference between foreground and background to the noise level (σ)—called the *contrast-to-noise ratio* (CNR) in this work—is low for all points on the boundary.

The CNR distribution is one indicator of the difficulty of a test image. Figure 4.2 shows the CNR distribution for all four test images, that is, the CNRs of all boundary elements arranged in order of increasing CNR. In my experience, boundary elements with CNR above two are typically recovered with negligible location error. Boundaries with CNR between two and one become increasingly challenging to recover and localize, and boundaries with CNR one or below are usually not recovered by the algorithm[4.1].

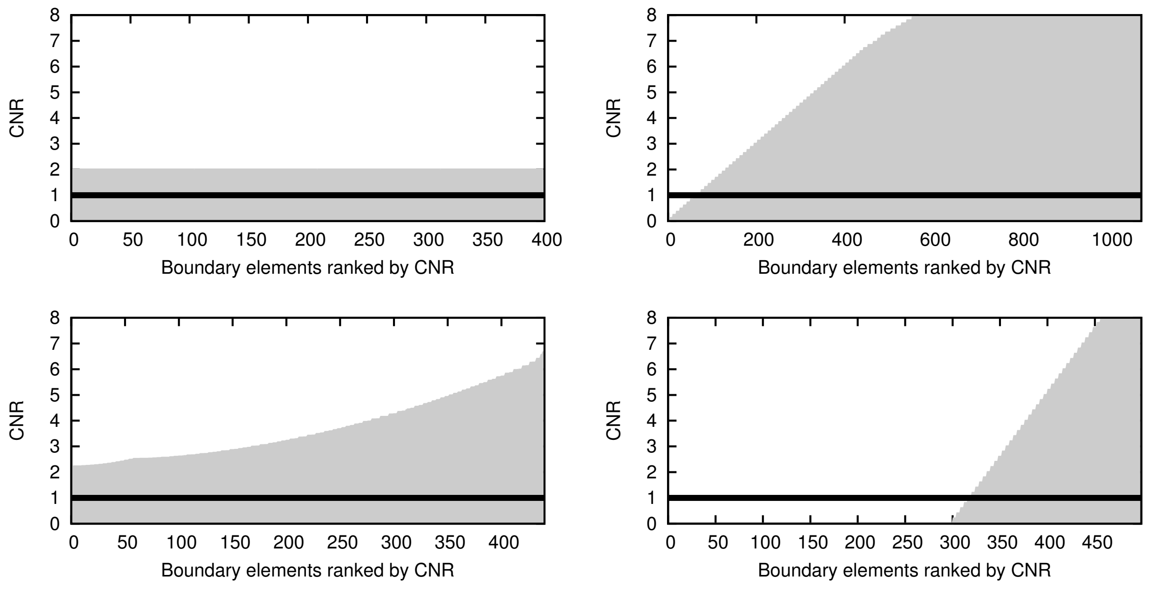


Figure 4.2. CNR over boundary elements for the test images 1 (upper left) to 4 (lower right) shown in Figure 4.1. CNR values above 8 are not shown.

4.1. Leclerc reports his algorithm recovering boundaries with a CNR of 0.5 (Fig. 3f in [48]). Note that this result was obtained with $p_{\max} = 0$, that is, with a piecewise constant image model. The results I discuss here are all obtained with $p_{\max} = 2$.

For the second test image the CNR is not constant but varies along the boundaries. The CNR is large for the most part but drops below one and even approaches zero (cf. Figure 4.2, top right). The challenge of this test case is to recover these extremely low CNR boundary segments. In this case we have reason to expect a recovery because we know the high-CNR parts of the boundary will be recovered reliably and any alternative closures of the boundaries are likely to result in a reconstruction u requiring longer descriptions.

The generating image $\boldsymbol{u}^g$ of test image 3 consists of two regions, a quadratic and a constant region. The challenge posed by this test case is to correctly recover the quadratic region.

The boundaries in test images 1, 2 and 3 are C^0-discontinuities. The generating image for test image four also includes C^1-discontinuities, where all vertical boundaries are C^1, all horizontal boundaries are C^0. Figure 4.3 shows a horizontal cross section through the center of the test image and a cross section of $\boldsymbol{u}^g$. The challenge of this test image is to recover the C^1 discontinuities. (To clarify, the lower right chart in Figure 4.2 includes CNR values for both C^0 and C^1 boundary elements.)

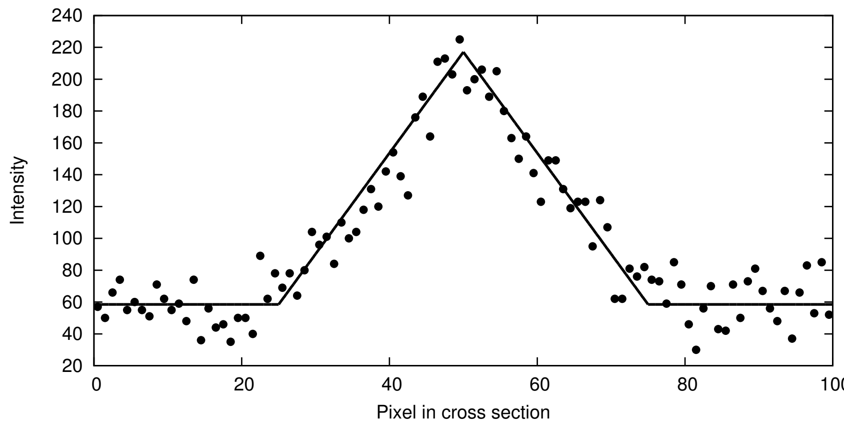


Figure 4.3. Horizontal cross section through the center of test image four. Dots are gray values, the solid line is a cross section of the graph of the corresponding generating image $\boldsymbol{u}^g$, featuring three C^1 discontinuities.

4.2 Evolution of a Solution

Before proceeding to specifics, I use test case 3 as an example to describe the algorithm's general behavior. In this example F_0 is minimized with parameter values $b = 0.41$, $d = 0.01$, and $f = 0.5$. The continuation method is started at $s_0 = 10$, stopped at $s_f = 0.001$, and the intermediate steps are calculated with $r = 0.96$ (cf. (2.20)). Figure 4.4 shows the progression of value $F_0(\boldsymbol{u}_i^*, s_f)$ in the iteration, where $\boldsymbol{u}^{*,0}, \boldsymbol{u}^{*,1}$, $\boldsymbol{u}^{*,2}, \ldots \boldsymbol{u}^{*,f}$ is the sequence of minimizers produced by the outer loop.

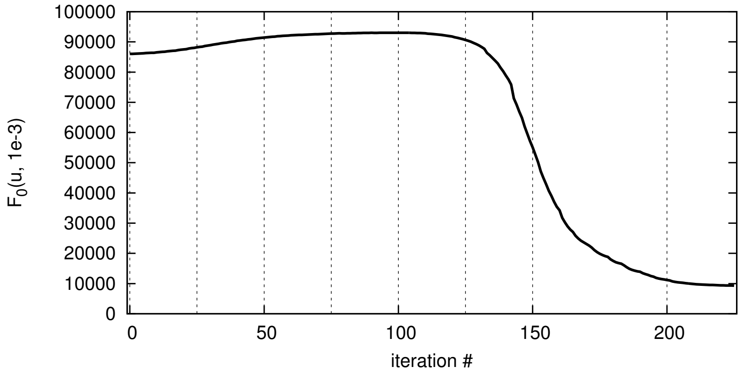


Figure 4.4. The value $F_0(\boldsymbol{u}^{*,i}, 0.001)$ over iterations i of the outer loop for test image 3. In the continuation method we minimize with respect to a different objective function $F_0(., s_i)$ in each iteration, whereas the plot above shows the value of the same objective function $F_0(., 0.001)$ at all minimizers $\{\boldsymbol{u}_i^*\}$. This explains why the value initially increases.

Figure 4.5 shows aspects of selected minimizers $\boldsymbol{u}^{*,0}, \boldsymbol{u}^{*,25}, \boldsymbol{u}^{*,50}, \ldots$ corresponding to the iterations marked by vertical dashed lines in Figure 4.4. The left column shows per-pixel samples of the minimizers, respectively, the right column shows horizontal cross sections through the center.

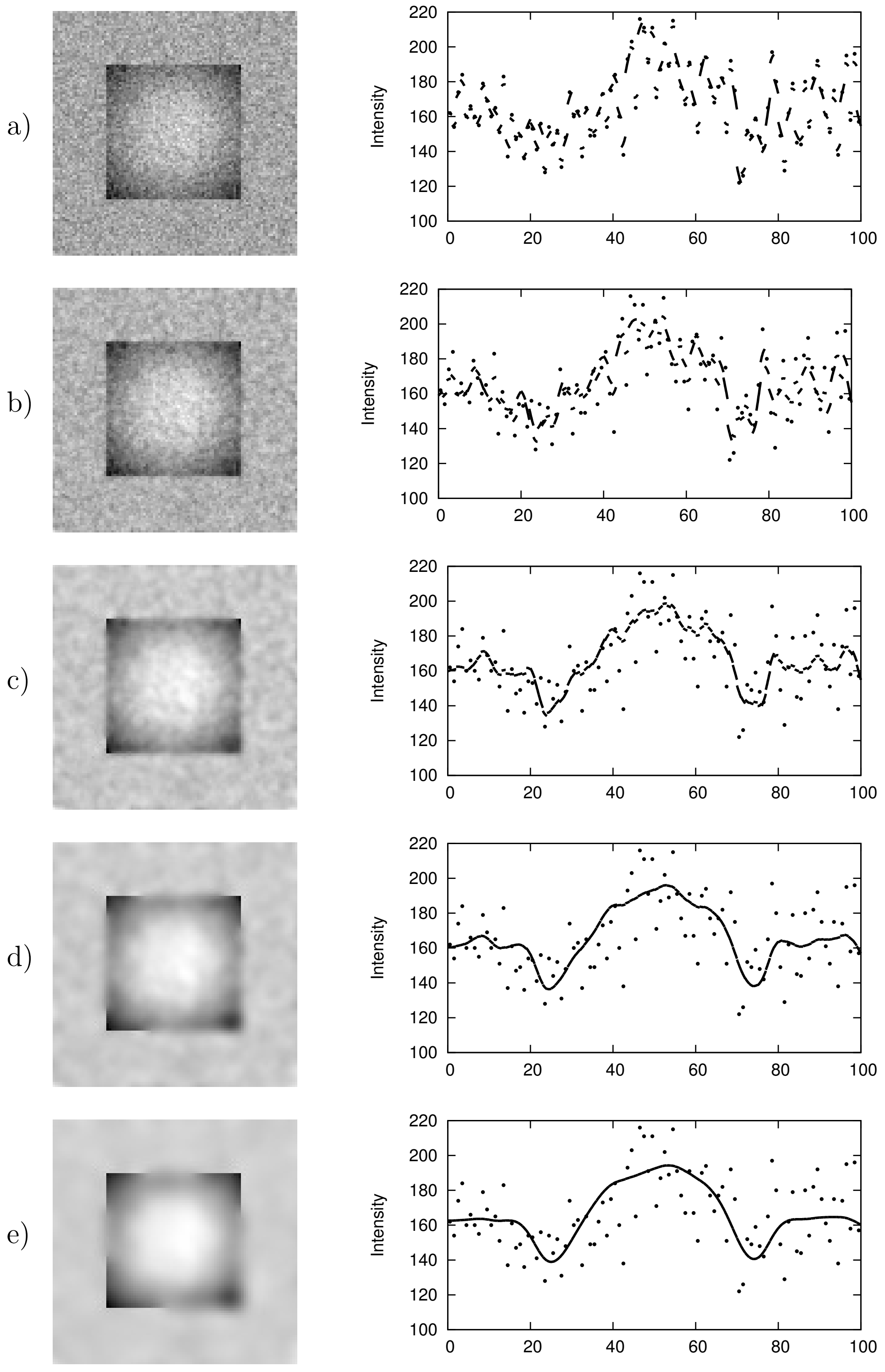


Figure 4.5.

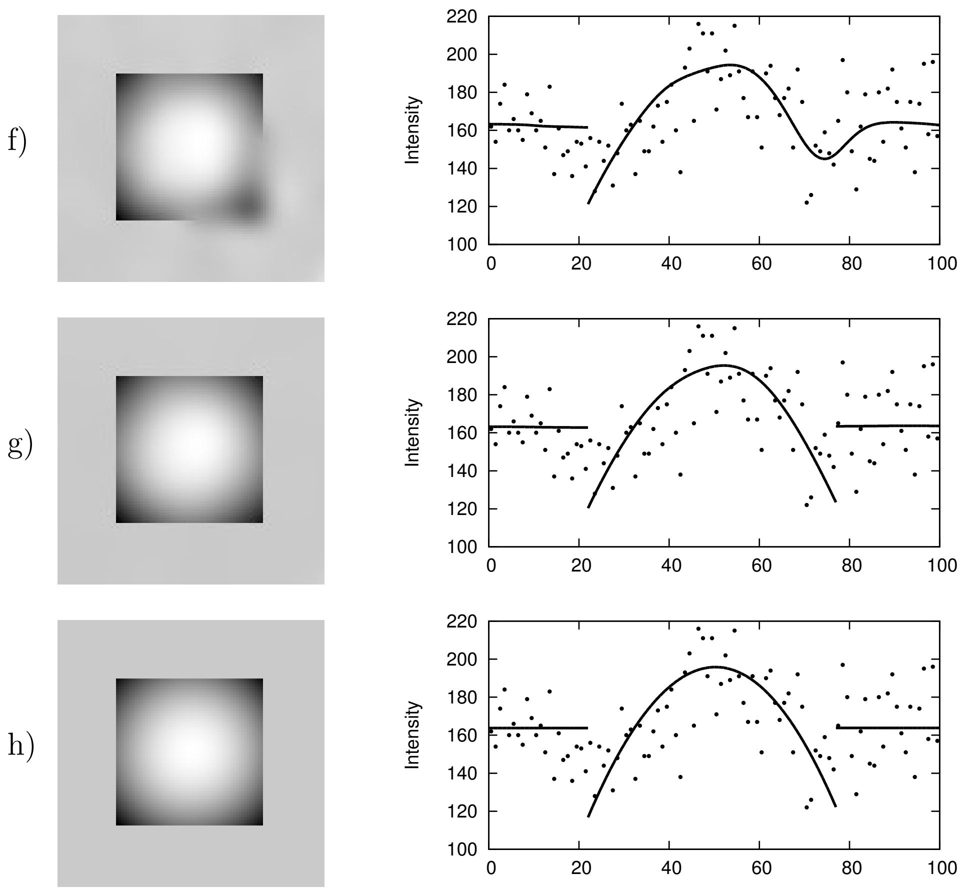


Figure 4.5. The constant coefficients of the minimizers $\{\boldsymbol{u}^{*,i}\}$ (left column), and horizontal cross-sections through the minimizers and $\boldsymbol{z}$ (right column). From top to bottom row the displayed data corresponds to the iterations marked by vertical dashed lines from left to right in Figure 4.4. The value of the continuation parameter s corresponding to each row is 10, 3.6, 1.3, 0.47, 0.17, 0.061, 0.022, and 0.0028, respectively.

It is most instructive to compare the cross section plots in the right column of Figure 4.5. At the beginning of the continuation when the continuation parameter s is larger than one, mismatches at cell boundaries ($\Delta_{i,j,k} \neq 0$) incur a negligible penalty in the objective function and the first term in (2.16)—the data term—dominates the character of the minimizers. These minimizers are discontinuous everywhere (cf. Figure 4.5 rows a) to c) with $s = 10$, 3.6 and 1.30, respectively).

As the continuation method progresses, the width of the embedding functions

(2.18) and (2.19) decreases and the influence of the other terms in (2.16) increases. In response to increasing penalties for nonzero $\Delta_{i,j,k}$ the finite elements move away from the gray level values $\boldsymbol{z}$, line up, and form a smooth graph almost everywhere (cf. Figure 4.5 rows d) and e) with $s = 0.47$ and 0.17, respectively). Discontinuities remain, however, where their penalties are offset by a decrease in the data term for cells near the discontinuities. This happens earlier in the continuation for discontinuities with larger contrast, as may be seen in the first column of Figure 4.5, rows d) and e), at the corners of the square region.

As the continuation parameter decreases further, the penalties for nonzero polynomial coefficients gain influence, making the finite elements increasingly stiff (cf. Figure 4.5, rows f) to h) with $s = 0.061$, 0.022, and 0.0028, respectively). This may cause discontinuities to be introduced or to spread relatively late in the process, as is evident around the lower right corner of the square region when comparing rows f) and g) in Figure 4.5.

To summarize, throughout the continuation process, the intermediate solutions $\boldsymbol{u}^{*,k}$ gradually morph into a smoothed version of the input image where the smoothing is adaptive and stops at discontinuities found in the process. As the continuation parameter decreases further, the spatial scale over which the piecewise polynomial form is enforced increases, and $\boldsymbol{u}^{*,k}$ evolves from a locally polynomial to a globally piecewise polynomial function.

4.3 Sensitivity to Cost Function Parameters

I now turn attention to the quality of the results produced by the reconstruction algorithm. The quality depends crucially on the choice of values for the parameters b and d (cf. Section 2.5.1). We will see in this section that excellent results may be achieved for each test case if the parameters are tuned specifically for the test case, but not when the same parameters are used for all test cases.

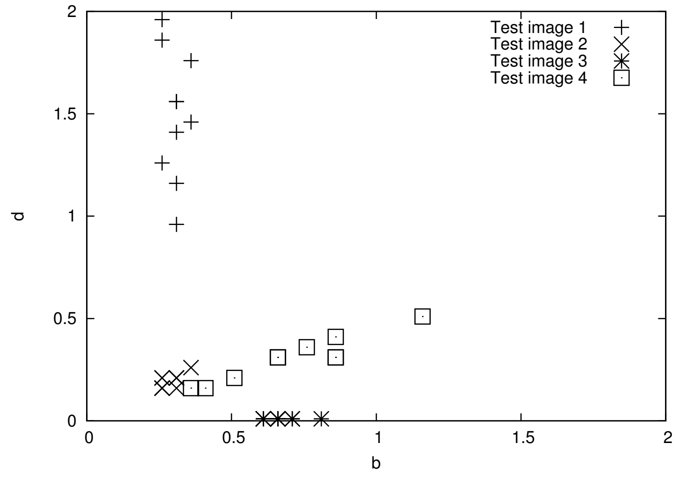


Figure 4.6. Best values in the interval $(0, 2)$ for parameters b and d for the different test images. From each generating image in Figure 4.1, top row, ten test images are created by adding different realizations of the same Gaussian noise process. For each test image the algorithm was run with a large number of combinations of values for b and d and the parameter value combination from the run producing the largest noise reduction (see text) was recorded in the above plot. Preliminary tests suggested 0.5 as a good value for the embedding parameter f (cf Section 2.5.1), and $f = 0.5$ was used for all results discussed in this section.

Since results obtained with different objective functions (i.e., F_0 with different values for b and d) are to be compared, the objective function value itself may not be used as a measure of quality of results. Instead we make use of each test case's good known solution $\boldsymbol{u}^g$ (cf. Section 4.1) by pretending it is the optimal solution. We may then compute the mean squared error,

$$\mathrm{mse}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{|I|} \sum_{i \in I} (u_{i,0} - v_{i,0})^2, \tag{4.1}$$

between $\boldsymbol{u}^g$ and the solution $\boldsymbol{u}^{*,f}$ produced by the algorithm, and compare it to the mean squared error of the input. I call the ratio of the two (in decibel) the noise reduction (NRED),

$$\mathrm{nred}(\boldsymbol{z}) = 10\log_{10}\left(\frac{\mathrm{mse}(\boldsymbol{u}^{g}, \boldsymbol{z})}{\mathrm{mse}(\boldsymbol{u}^{g}, \boldsymbol{u}^{*,f})}\right), \tag{4.2}$$

achieved for a test case.

Running the reconstruction algorithm for each test image with many different parameter value combinations and identifying the maximum-NRED result, respectively, one finds that the parameter value combination yielding the best solution is different from test case to test case (cf. Figure 4.6).

Figure 4.7 shows the maximum-NRED result for each test case. Aside from a few small artifacts in a) and b) we note that rows a) to c) exhibit the correct image descriptions. That is, all regions are found with excellent boundary localization and the polynomial degree of each region is correctly recovered. The reconstruction in Figure 4.7 d) is almost as expected—two out of three C^1-contours are reconstructed. The C^1-contour forming the crest of the rooftop-shaped generating image (cf. Figure 4.3), on the other hand, was instead described as a narrow, quadratic region that smoothly connects the two faces of the rooftop (see the cross section in Figure 4.7 d)).

That different data require different objective function parameter values for best reconstruction is not a remarkable observation. The more important question is whether there are parameter value combinations that give reasonably good results for all test cases. If this is the case then we should expect a point that is near the points in Figure 4.6 to be a good parameter value choice.

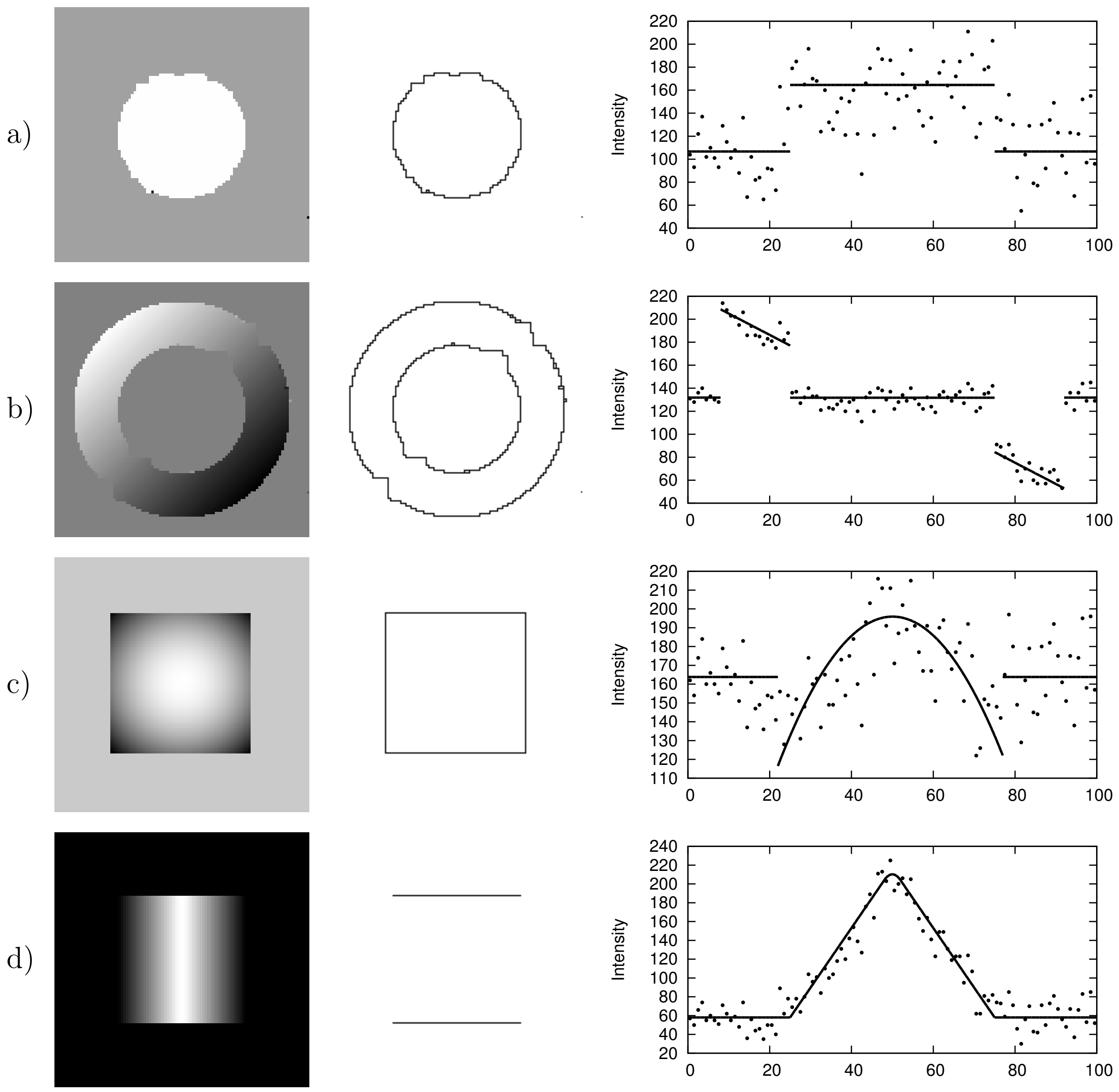


Figure 4.7. Results with test-case specific parameter values: a) $b = 0.31$, $d = 0.96$, b) $b = 0.26$, $d = 0.16$, c) $b = 0.41$, $d = 0.01$, d) $b = 0.66$, $d = 0.31$. The display above is similar to Figure 4.5, but the center column shows the C^0-discontinuities in the reconstruction.

The average of all the points in Figure 4.6 gives $b = 0.4938$ and $d = 0.5013$. Figure 4.8 shows the results for the four test images with these parameters (all other parameters are the same as in the runs producing Figure 4.7).

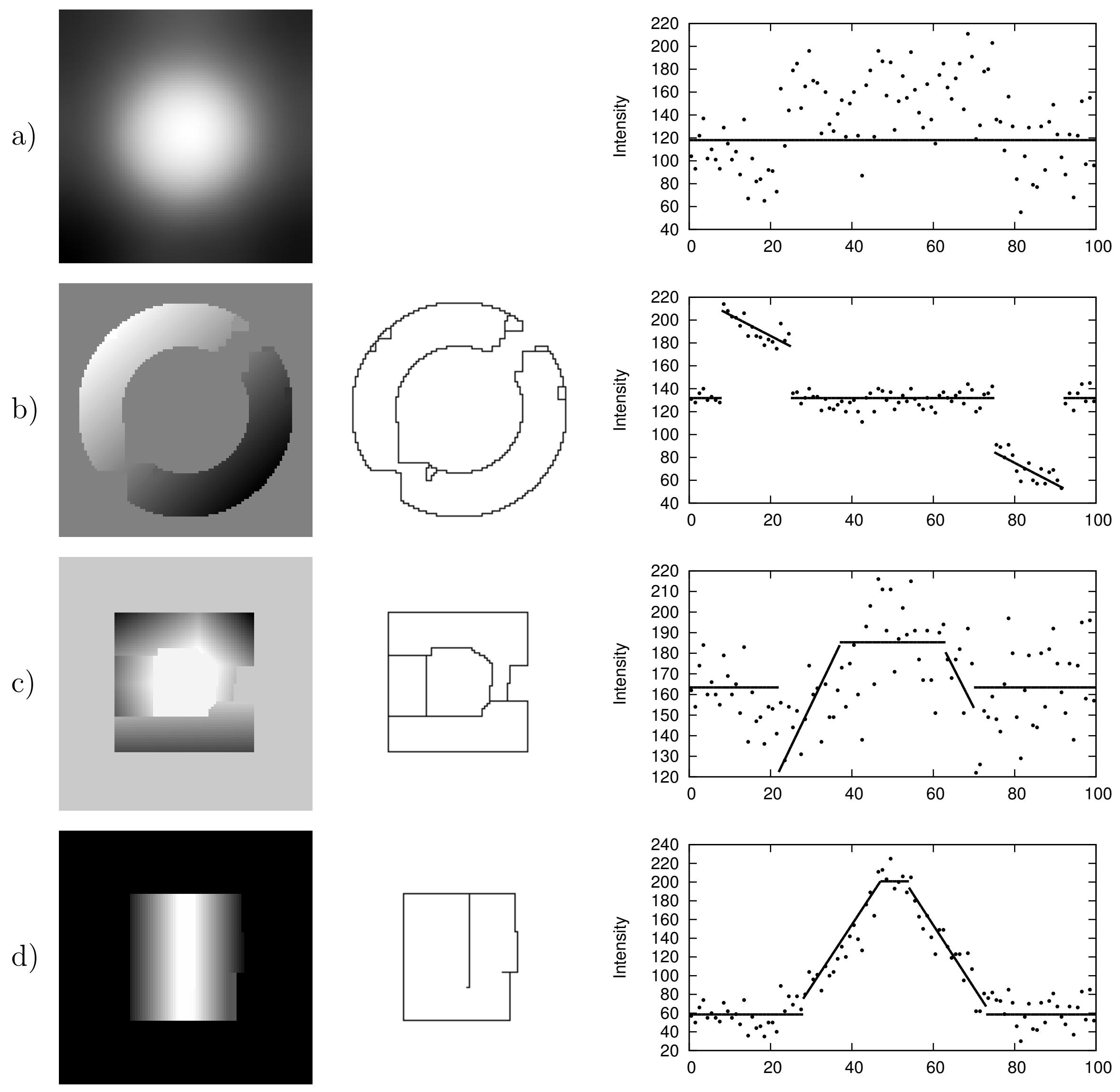


Figure 4.8. Results similar to Figure 4.7 but with test-case independent parameter values $b = 0.42938$ and $d = 0.5013$. The reconstruction of test case 1 shown in a) is essentially flat. The intensity values of the pictures in the left column are scaled to utilize the full gray-value range, which causes the appearance of a bell-shaped graph in the left picture a).

The results shown in Figure 4.8 are not all satisfactory. The reconstructions are particularly bad for test cases 1 and 3. For test case 1 the algorithm failed to separate the two regions (Figure 4.8 a). And for test case 3 the inner region with quadratic variation was not reconstructed as such but is instead reconstructed as piecewise linear involving multiple regions (Figure 4.8 c). This result seems at odds with Leclerc's suggestion that the algorithm does not require parameter tuning[4.2].

4.3.1 Results for Objective Function F_2

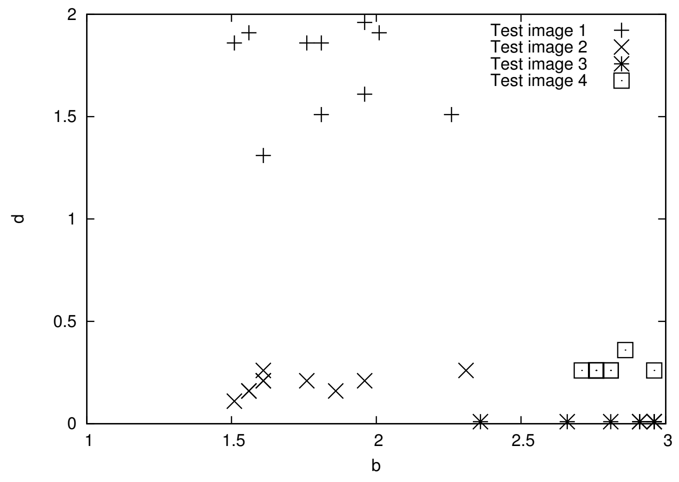


Figure 4.9. Best parameter values for objective function F_2, see Figure 4.6 for more information.

In order to gauge its viability I include objective function F_2, (2.21), in this parameter-sensitivity study. The scatter plot of best results (Figure 4.9) shows some similarity to the plot for F_0 (Figure 4.6), the main differences being a shift to larger b-values and a wider spread of b-values. The quality of results with F_2 turns out to be as good as with F_0 for test cases 1, 2 and 3. For test case 4 the results are qualitatively different in that the F_2-results contain C^0-discontinuities where F_0-results have C^1- or C^2-discontinuities. Figure 4.10 shows the best F_2-result for test case 4.

4.2. Leclerc emphasizes the fact that the same parameter values were used in producing all his results in Section 7 of [48] (page 92). I note that except for his first experiment (Fig. 3 in [48]), the noise levels are lower in his test images than in the ones used here, which makes for less challenging test cases. Also, Leclerc does not present a quantitative evaluation of results.

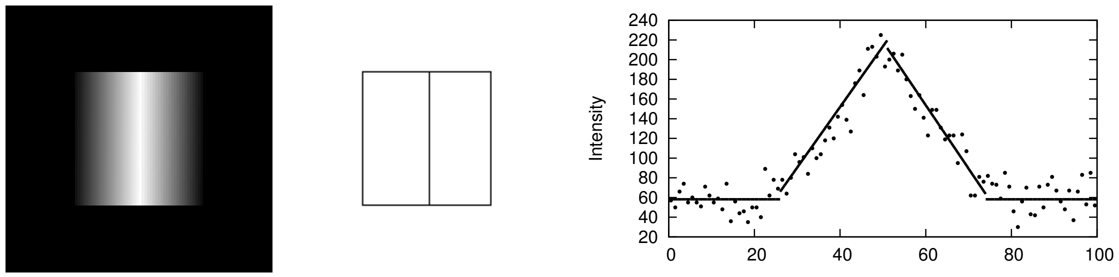


Figure 4.10. Result similar to Figure 4.7 for test case 4 with objective function F_2, $b = 2.82$, $d = 0.36$. Note that the discontinuities in the reconstruction are C^0, not C^1.

4.4 Solution Accuracy Affects the Continuation Path

Different images are more or less difficult to reconstruct. More difficult cases require a higher accuracy of the solution produced by the inner loop (Section 2.5.1), and a smaller step size for the continuation parameter s (i.e., a value for r closer to 1). Thus more difficult reconstructions require more computation time.

4.4.1 Cases with Low CNR

Leclerc explains this phenomenon for cases that are difficult due to the presence of low-CNR contours and concludes that the lower the CNR of contours one expects to reconstruct, the smaller the step size needs to be in the continuation loop ([50], page 156).

Intuitively, the decision as to whether a discontinuity ought to be introduced into the solution becomes harder as CNR decreases. In terms of the optimization problem, two minima corresponding to two qualitatively different solutions—with or without a discontinuity—both make themselves felt at some point in the continuation process. In the worst case a *bifurcation* may occur, that is, a point with multiple descent directions leading to qualitatively different solutions. It is plausible that close to such a bifurcation point the branch completing the continuation path may be decided quasi-randomly because the continuation path is computed approximately.

4.4.2 Cases with Non-Constant Regions

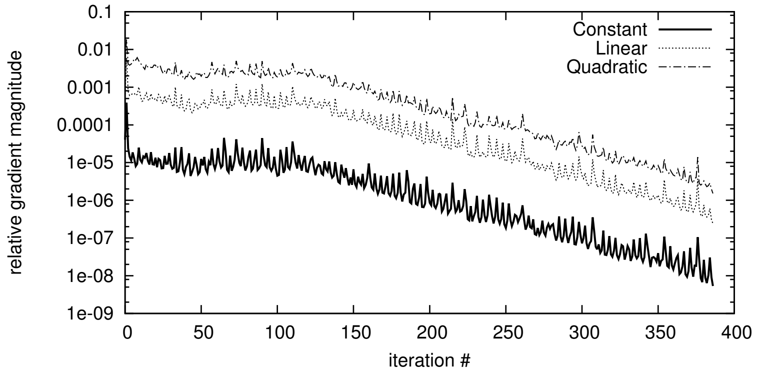


Figure 4.11. Development of the quantities $\|(\nabla F_0(\boldsymbol{u}, s))_{-,k}\|_2 / \max(1, \|\boldsymbol{u}_{-,k}\|_2)$ for constant, linear, and quadratic coefficients, respectively, over iterations of the inner-loop in the minimization of objective function F_0 with test case 3. The notation $\boldsymbol{u}_{-,k}$ denotes the vector consisting of the k-th coefficients in vector $\boldsymbol{u}$.

Other kinds of decisions may manifest themselves in bifurcations as well—for instance the decision as to whether part of an image ought to be described with quadratic or linear regions (cf. Figure 4.12). A subtlety of the finite element representation might aggravate the problem for these kinds of bifurcations.

The range of values of the linear coefficients is typically an order of magnitude smaller than the range of values of the constant coefficients. The same applies to the quadratic coefficients compared to the linear coefficients. This means that the termination criteria used in the optimization effectively apply to the constant coefficients only (i.e., (41) and (43) in Chapter 7). The relative errors in the linear and quadratic coefficients, on the other hand, are much larger than these criteria would suggest (Figure 4.11). Reliable reconstruction of cases including quadratic regions then generally require lower tolerances in the termination criteria than cases including linear or constant regions. Figure 4.12 b) shows an example of this phenomenon. In Section 5.3.2 I present a way to address this issue.

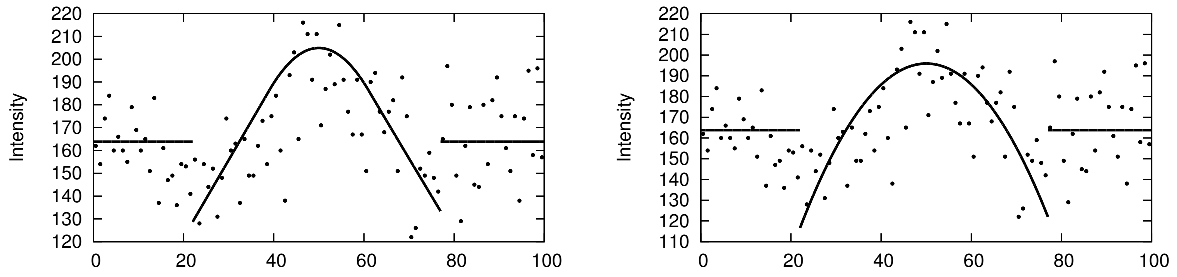


Figure 4.12. Example showing different results for test case 3 obtained with different accuracy of the intermediate solutions computed in the inner-loop. For the result on the left gtol $= 10^{-4}$ was used, for the result on the right gtol $= 10^{-6}$ was used in (7.4). The values for parameters b and d where chosen differently than in Figure 4.7 and Figure 4.8 to demonstrate this phenomenon.

4.5 Discussion

4.5.1 Recoverability of Low-CNR Contours

Using a test case similar to 1 in Figure 4.1 Leclerc observes that the $p_{\max} = 0$ version of the algorithm can reconstruct contours of CNR as low as 1/2 [48]. He further notes that contours of CNR below some threshold are not recovered—as one would expect. In my experiments with the $p_{\max} = 2$ version of the algorithm I have not been able to reproduce Leclerc's result. Contours with CNR below one appear to be non-recoverable.

I believe that the CNR threshold for recoverable contours of the $p_{\max} = 0$ algorithm is different from and lower than the CNR threshold of the $p_{\max} = 2$ algorithm. Intuitively, the new degrees of freedom that the piecewise polynomial image model introduces can be used to achieve smaller $\Delta_{i,j,0}$-values early in the continuation process (see Figure 4.5 a) and b), right column). Hence, differences in gray-values that would cause discontinuities to develop with the $p_{\max} = 0$ model may instead develop as a locally smooth reconstruction of nonzero slope with the $p_{\max} = 2$ model.

4.5.2 F_0 vs F_2 and the Meaning of the b-Term

In Section 4.3.1 we see that, except for cases with higher-order discontinuities, objective function F_2 may be used in place of F_0 and achieve the same quality of results. We also see in Figure 4.9 that good values for parameter b with F_2 lie around 2. This better fits Leclerc's prescription from Section 2.3 that "b should be at least as large as $\log_2 3$, but not much more than two", than the significantly smaller values for b in Figure 4.6. It lends support to my claim that the b-term in F_2 has the same meaning as the b-term in (2.10), and reinforces the question as to how to interpret Leclerc's b-term (2.14).

Chapter 5

Improving Algorithm Robustness

Lack of plausible reasoning for some steps in the derivation recounted in Section 2.4 and the issues discussed in Chapter 4 led me to explore modifications of Leclerc's algorithm. In this chapter I present alternative algorithmic choices that are connected by a common theme—they all lead to a different algorithm than Leclerc's for the case $p_{\max} > 0$.

In Section 5.1 I derive a different MDL criterion for the reconstruction problem. Its derivation uses approximations that are tighter than Leclerc's and it does not require use of unknown quantities such as an average region size. Because fewer approximations are required, my MDL criterion contains only one parameter instead of two. A second improvement, presented in Section 5.2, is a redefinition of the difference terms $\Delta_{i,j,k}$ (Section 2.4). It helps to avoid misclassification of cell boundary elements that may occur in cases of vanishing CNR (e.g., for test case 2).

In Section 5.3 I argue that Leclerc's embedding for the piecewise polynomial case does not account for the scale of the coefficients, and I derive a different embedding which does account for scale. In this section I also suggest a way to address the problem that higher-degree coefficients are computed to lower accuracy than lower-degree coefficients (Section 4.4.2); in Section 5.4 I suggest a second way to address this problem.

In Section 5.5 I discuss the relationship between Leclerc's MDL criterion and mine, and I contemplate the roles the MDL parameters and the embedding play, respectively, in finding global optima.

5.1 MDL Criterion

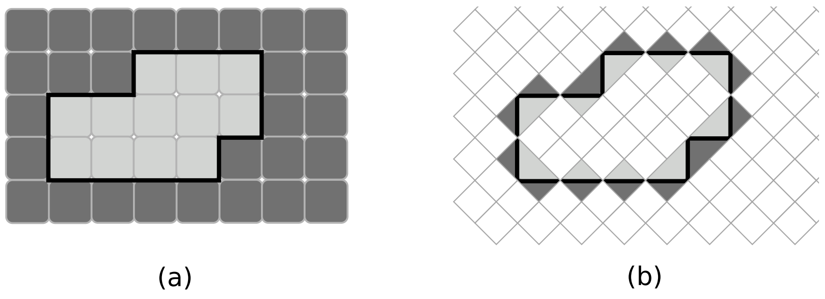


Figure 5.1. (a) Two regions. (b) Description length L_0 may be reduced when adjacent cells share coefficients. Boundary elements between coefficient-sharing cells are covered with a white diamond. When the worst case description length is corrected for each white diamond, the resulting description length is proportional to the number of coefficients in each region, as well as proportional to the boundary length.

In deriving an expression for the description length of the image z the first step is the same as in Chapter 2, and we encode z in terms of a reconstruction and residuals as in (2.3). We use the same estimate for the code length of the residuals (2.5) but derive an alternative to (2.6) for an estimate of the length of u.

Consider the worst case where every cell has different coefficients from its neighboring cells. The code length for $\boldsymbol{u}$ in this case is

$$L_0(\boldsymbol{u}) = l_c \sum_{i \in I} \left[1 + 2 \max_{k \in \{1,\ldots,5\}} \tilde{\delta}(u_{i,k}) + 3 \max_{k \in \{3,4,5\}} \tilde{\delta}(u_{i,k}) \right].$$

For cases other than the worst case there are regions consisting of more than one cell and this code length may be improved upon. To realize a more efficient code we need to determine the number of coefficients that are shared by any pair of adjacent cells and subtract from L_0 a number of bits proportional to the number of shared coefficients.

For example, if cells i and j, $j \in N_i$, have a common (equivalent) constant coefficient, we subtract $l_c\,(1/|N_i| + 1/|N_j|)$. If, in addition, i and j share the two first-order coefficients we subtract $2\,l_c\,(1/|N_i| + 1/|N_j|)$ more bits. Finally, if i and j also share the second-order coefficients, we subtract $3\,l_c\,(1/|N_i| + 1/|N_j|)$ more bits. To evaluate what coefficients are shared we may use the $\Delta_{i,j,k}$-terms defined in Section 2.4. Thus, the code-length "correction" for cell pair (i,j) is

$$\left(\frac{l_c}{|N_i|}+\frac{l_c}{|N_j|}\right)\left\{\delta(\Delta_{i,j,0})+2\max_{k\in\{1,\ldots,5\}}\tilde{\delta}(u_{i,k})\times\min_{k\in\{0,1,2\}}\delta(\Delta_{i,j,k})\right.$$
$$\left.+3\max_{k\in\{3,4,5\}}\tilde{\delta}(u_{i,k})\times\min_{k\in\{0,\ldots,5\}}\delta(\Delta_{i,j,k})\right\}$$

with $\delta(x)=1-\tilde{\delta}(x)$. Subtracting this term from L_0 for all cell boundary elements we get

$$\begin{aligned} L(\boldsymbol{u}) \;=\; l_c\sum_{i\in I}\Bigg\{ & 1-\frac{1}{|N_i|}\sum_{j\in N_i}\delta(\Delta_{i,j,0}) \\ & +2\max_{k\in\{1,\ldots 5\}}\tilde{\delta}(u_{i,k})\times\left(1-\frac{1}{|N_i|}\sum_{j\in N_i}\min_{k\in\{0,1,2\}}\delta(\Delta_{i,j,k})\right) \\ & +3\max_{k\in\{3,4,5\}}\tilde{\delta}(u_{i,k})\times\left(1-\frac{1}{|N_i|}\sum_{j\in N_i}\min_{k\in\{0,\ldots,5\}}\delta(\Delta_{i,j,k})\right)\Bigg\} \end{aligned} \tag{5.1}$$

Formula (5.1) approximates the length of a *run-length encoding* [44] of $\boldsymbol{u}$. In a run-length encoding data on a grid is encoded on a row-by-row or column-by-column basis. It exploits *runs* (homogeneous sub-strings) in a row by replacing each run by the respective value and its repetition-count. It is an effective encoding when typical row data consists of a small number of runs per row. In a run-length encoding of $\boldsymbol{u}$ a run is a sequence of equivalent or "same-region" cells.

Unlike an actual run-length encoding, however, (5.1) is invariant with respect to 90-degrees rotation of the image. It achieves this by effectively averaging the run-length encodings with respect to columns and with respect to rows. To see this, consider that the correction terms are nonzero for all cell-boundary elements inside a region but are zero at region boundaries (Figure 5.1). The effect is that only cells along a boundary contribute to the description length, and (5.1) scales with boundary length and number of coefficients,

$$L(\boldsymbol{u})\approx\sum_{j\in J}|\partial R_j|\times L(u|_{R_j}). \tag{5.2}$$

The encoding of run-length is not accounted for in (5.1). To account for it we replace l_c by a new parameter g, which may be interpreted as the sum of l_c and a fixed number of bits for encoding the run-lengths. With this result the new MDL criterion for the piecewise polynomial case reads

$$\begin{aligned} L(z) &= \min_{\boldsymbol{u}} \sum_{i\in I} \Bigg\{ \frac{1}{2\log 2}\Big(\frac{z_i - u_{i,0}}{\sigma}\Big)^2 + g\Bigg(1 - \frac{1}{|N_i|}\sum_{j\in N_i} \delta(\Delta_{i,j,0})\Bigg) \\ &\qquad\qquad + 2\,g \max_{k\in\{1,\ldots 5\}} \tilde{\delta}(u_{i,k}) \times \Bigg(1 - \frac{1}{|N_i|}\sum_{j\in N_i} \min_{k\in\{0,1,2\}} \delta(\Delta_{i,j,k})\Bigg) \\ &\qquad\qquad + 3\,g \max_{k\in\{3,4,5\}} \tilde{\delta}(u_{i,k}) \times \Bigg(1 - \frac{1}{|N_i|}\sum_{j\in N_i} \min_{k\in\{0,\ldots,5\}} \delta(\Delta_{i,j,k})\Bigg)\Bigg\} \\ &= \min_{\boldsymbol{u}} F_1(\boldsymbol{u}). \end{aligned} \tag{5.3}$$

Note that unlike (2.11), which requires knowledge of the regions, (5.1) uses only local information to estimate the code length $L(\boldsymbol{u})$. The approximation we make in (5.3) to account for run-length with the parameter g is a much tighter approximation than Leclerc's, which introduces an average region size for parameter d and an average boundary length for parameter b in (2.15).

5.2 Difference Terms

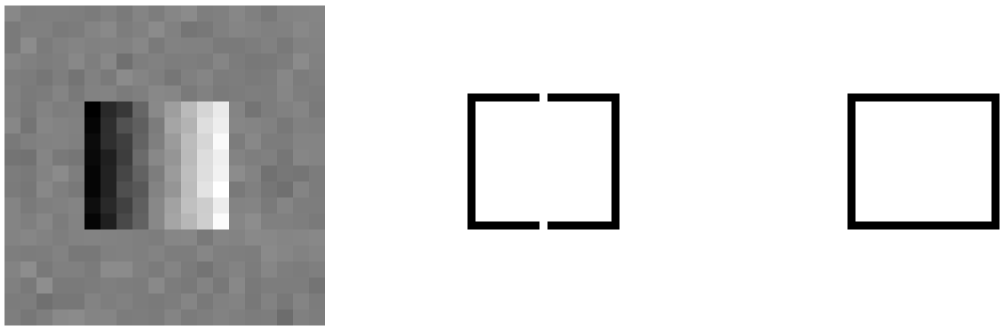


Figure 5.2. Ramp image from Leclerc's paper [48] (left), C^0-discontinuities in reconstruction with difference terms from Section 2.4 (middle), and C^0-discontinuities with revised difference terms from Section 5.2 (right).

Leclerc uses the ramp image Figure 5.2, left, to demonstrate that his reconstruction algorithm is capable of detecting contours in situations where CNR vanishes. I use the same example to illustrate a problem with the difference terms $\{\Delta_{i,j,k}\}$ as defined in Section 2.4.

The C^0-discontinuities that result with the definitions from Section 2.4 are shown in Figure 5.2, middle. The gaps in the boundary occur where the value of u in the ramp region coincides with the value of u in the background region. Each gap is exactly one cell wide and the two cells adjacent to a gap have the same value exactly at the center of the cell boundary that is the gap[5.1]. In other words, the two adjacent cells are discontinuous across the cell boundary except for a single point. This cell boundary element therefore ought to be classified as a C^0-discontinuity. (Another way to make this argument is to note that the relative size of the gap goes to zero as the resolution of the image increases.)

The problem occurs because the difference terms as defined in Section 2.4 evaluate u (or a derivative of u) at a single point on the cell boundary. There is no problem when $p_{\max} = 0$, and one might call this an instance of inappropriate generalization from the piecewise constant to the piecewise polynomial case.

To fix the problem I redefine the difference term $\Delta_{i,j,0}$ such it evaluates u at *all* cell boundary points,

$$\Delta^2_{i,j,0} = \frac{1}{h} \int_{\partial\Omega_i \cap \partial\Omega_j} (u|_{\Omega_i}(x,y) - u|_{\Omega_j}(x,y))^2 \, d\tau, \tag{5.4}$$

where h is the cell size (i.e., the length of the line segment $\partial\Omega_i \cap \partial\Omega_j$). The other terms $\Delta_{i,j,k}$, $k > 1$, are defined in a similar way. Expressions for evaluating the terms $\Delta^2_{i,j,k}$ are given in the appendix, Section B. Unless indicated otherwise, the difference terms defined by (5.4) etc. are used in this work.

5.1. Leclerc makes the same observation about this example, see page 94 and Figures 4b and 4c in [48].

5.3 Embedding

We now consider the embedding discussed in Section 2.5.1, specifically the use of (2.18) and (2.19). I will argue that the noise level σ is not a good choice of scale factor except for the C^0-discontinuity terms and will suggest different scale factors and other modifications to the embedding.

Leclerc first develops an objective function for the piecewise constant case, $p_{\max} = 0$, then generalizes to the general case $p_{\max} > 0$. This includes the embedding. In the piecewise constant case the embedding concerns only one type of quantity—the potential C^0-discontinuities $\{\Delta_{i,j,0}\}$. Leclerc suggests the embedding

$$\delta(\Delta_{i,j,0}) \to \exp\left(-\Delta_{i,j,0}^2/(s\,\sigma)^2\right) \tag{5.5}$$

([48], page 82). Here s is the embedding parameter (cf. Section 2.5.1) and σ serves as a scale for $\Delta_{i,j,0}$: when $s=1$ and $\Delta_{i,j,0} \gg \sigma$, then u has a true discontinuity across the cell boundary between i and j; when $\Delta_{i,j,0} \sim \sigma$ then the mismatch between $u_{i,0}$ and $u_{j,0}$ is more likely due to noise.

The use of σ as a scale for $\{\Delta_{i,j,0}\}$ makes sense. In the piecewise polynomial case $p_{\max} > 0$ more δ-terms in the objective function, involving other quantities, require a similar embedding treatment. Leclerc uses the same scale factor σ in those embeddings—cf. (2.18) and (2.19)—but the reasons are unclear. We next consider the quantities in the δ-terms of (2.16) and derive appropriate scale factors for them.

5.3.1 Coefficient Scaling

We begin with the polynomial coefficients $\{u_{i,k}\}$. Their typical scale depends on the image data, where the important data-specific quantities are the range of valid gray values G and the typical region diameter R. For example, if gray values span the range $[0, 255]$ we have $G = 255$. We now consider a quadratic polynomial with domain $[-R/2, R/2] \times [-R/2, R/2]$ and range $[0, G]$,

$$u(x,y) = u_0 + u_1\,x + u_2\,y + u_3\,x^2 + u_4\,x\,y + u_5\,y^2,$$

and ask what is the typical magnitude of the coefficients $\{u_k\}$? Since a polynomial is linear in its coefficients, we have $u_k \sim G$ for all k. The dependency of the u_k on R derives from the constraint $0 \le u \le G$: u_0 is independent of R, whereas $u_1, u_2 \sim \frac{1}{R}$, and $u_3, u_4, u_5 \sim \frac{1}{R^2}$. The polynomial coefficients therefore scale as

$$u_0 \sim G,\, u_1, u_2 \sim \frac{G}{R},\, u_3, u_4, u_5 \sim \frac{G}{R^2}. \tag{5.6}$$

An embedding that accounts for this scaling behavior is

$$\begin{aligned} \delta(u_{i,k}) &\;\rightarrow\; \exp\left(-u_{i,k}^2/(\min(1,s)\,\rho_k)^2\right), \text{ with} && (5.7)\\ \rho_k &= \begin{cases} \frac{G}{2R}\;, & \text{if } k=1,2 \\ \frac{G}{2\,R^2}, & \text{if } k=3,4,5 \end{cases}. && (5.8) \end{aligned}$$

The value for G is determined by the image data format. The value for R is taken as a fraction of the image diagonal D so that R scales with image resolution,

$$R = f_R\, D, \tag{5.9}$$

where f_R values in the range $1/10 - 1/5$ should work well for most images ($f_R = 1/6$ is used in this work). The min operation in (5.7) is to prevent unrealistically large coefficients early in the continuation process.

5.3.2 Scaling the Coefficients for Improved Accuracy and Runtime

With an understanding of the coefficient's scaling behavior we are in a position to improve the accuracy of the higher-degree coefficients (cf. Section 4.4.2). If we arrange for the cell size h to be variable instead of fixed (cf. Section 5.2), then (5.6) becomes

$$u_0 \sim G,\, u_1, u_2 \sim \frac{G}{R\,h},\, u_3, u_4, u_5 \sim \frac{G}{(R\,h)^2}, \tag{5.10}$$

and we see that all coefficients will have similar scale when $h = 1/R$. The reconstruction algorithm can set h internally and rescale the coefficients according to $h = 1$ before returning its solution.

Note that Leclerc does not have parameter h in his formulation [48]. Using h in the way described here is also important for achieving good runtime performance. This is discussed in detail in Chapter 7.

5.3.3 Discontinuity Scaling

While σ makes sense as a scale factor for $\Delta_{i,j,0}$ in (5.5), σ is not directly related to differences in derivatives of u across cell boundaries, $\{\Delta_{i,j,k}\}$, $k > 1$. Those differences become zero when the reconstruction is smooth across a cell boundary and nonzero when there is a change in slope or a change in curvature. In general, the scale for differences in slope is the same as the scale for slope, and likewise for curvature. In other words, the same scale factors (5.8) apply to the quantities $\{\Delta_{i,j,k}\}$, $k > 1$, as well.

5.3.4 Embedding for the Piecewise Polynomial Case

To summarize, I suggest the following embedding for the $p_{\max} = 2$ algorithm:

$$\delta(\Delta_{i,j,k}) \rightarrow \exp\left(-\Delta_{i,j,k}^2/(s\,\rho_k)^2\right),\ k > 1 \tag{5.11}$$

$$\delta(u_{i,k}) \rightarrow \exp\left(-u_{i,k}^2/(\min(1, 2\,s)\,\rho_k)^2\right) \tag{5.12}$$

$$\rho_k = \begin{cases} \sigma, & k = 0 \\ G/2, & k > 0 \end{cases} \tag{5.13}$$

$$h = 1/R. \tag{5.14}$$

The factor 2 included in (5.12) effects a delay of the tightening of the coefficients' embedding functions relative to the discontinuities' embedding functions. The motivation for this may be understood from rows a) to d) of Figure 4.5: there need to be continuity between cells before there can be a meaningful reading of the coefficient values.

5.4 Ensuring Coefficient Accuracy

As I discuss in Section 4.4, care must be taken to ensure that all coefficients are computed to comparable accuracy in all intermediate problems of the continuation method. A measure complementary to the choice of cell size (5.14) is to perform optimizations with subsets of coefficients.

Suppose we separate the independent variables $\boldsymbol{u}$ and arrange them into three vectors $\boldsymbol{q}$, $\boldsymbol{v}$, and $\boldsymbol{w}$, such that $\boldsymbol{q}$ contains all constant coefficients, $\boldsymbol{v}$ all linear, and $\boldsymbol{w}$ all quadratic coefficients. Writing objective function F as a function of these three vectors, $F(\boldsymbol{q},\ \boldsymbol{v},\ \boldsymbol{w},\ s)$, an effective way of improving the relative accuracy of the independent variables $\boldsymbol{v}$ and $\boldsymbol{w}$ is to solve the problems

$$\min_{(\boldsymbol{v},\boldsymbol{w})} F(\boldsymbol{q},\boldsymbol{v},\boldsymbol{w},s) \tag{5.15}$$

and

$$\min_{\boldsymbol{w}} F(\boldsymbol{q},\boldsymbol{v},\boldsymbol{w},s), \tag{5.16}$$

respectively.

In practical terms, the continuation method would consist of solving three problems in each iteration instead of one—minimization of $F(\boldsymbol{u}, s)$ w.r.t. $\boldsymbol{u}$, followed by minimization w.r.t. $(\boldsymbol{v}, \boldsymbol{w})$, followed by minimization w.r.t. $\boldsymbol{w}$.

5.5 Discussion

5.5.1 Relation Between F_0 and F_1

The objective functions F_0 and F_1 appear to have not much in common besides the data term. But it can be shown that F_0 is included in F_1 in the sense that

$$\begin{aligned} F_1(\boldsymbol{u}) &= F_0(\boldsymbol{u}) + M(\boldsymbol{u}) \text{ when } d = g \text{ and } b = g/2, \text{ where} \\ M(\boldsymbol{u}) &= \frac{g}{4}\sum_{i\in I}\sum_{j\in N_i}\Big[2\min_{k\in\{1,\dots,5\}}\delta(u_{i,k})\times\min_{k\in\{0,1,2\}}\delta(\Delta_{i,j,k}) \\ &\qquad +3\min_{k\in\{3,4,5\}}\delta(u_{i,k})\times\min_{k\in\{0,\dots,5\}}\delta(\Delta_{i,j,k})-6\Big] \end{aligned} \tag{5.17}$$

(cf. Appendix C).

The result is surprising and it poses the question whether F_0 could be regarded as an approximation to F_1. This would correspond to the following, alternative way of deriving Leclerc's objective function:

1. start with the MDL criterion based on a run-length encoding of u,

2. separate terms depending on $u_{i,k}$ and terms depending on $\Delta_{i,j,k}$
3. drop the mixed terms M, and
4. allow for independent weighting of $u_{i,k}$-terms and $\Delta_{i,j,k}$-terms (i.e., introduce a new parameter).

The experiments of Section 4.3 do not seem to corroborate this view of F_0,—the relation $d = 2\ b$ between the two parameters b and d cannot not be seen in the parameter choices required for F_0 (cf. Figure 4.6).

Even so, relation (5.17) may hold new insights into F_0 by pointing away from a chain-code encoding interpretation. The b-term in F_0, for instance, scales similar to (5.2). It is proportional to boundary length *and* the encoding length for the coefficients of the two neighboring regions *of higher degree*. For the underlying image of test case 3, for example, the b-term is proportional to the encoding length for six coefficients since the central region is quadratic. That means F_0 without the d-term could be interpreted as an imperfect length estimate of a run-length encoding of u, the imperfection being that the encoding length of the lower-degree region's coefficients is overestimated. The d-term could serve to correct for this by penalizing the use of higher-degree coefficients.

5.5.2 Solver Bias and Generalized Embeddings

Returning to the idea that F_0 could be an approximation of F_1, there is a possible explanation why $d = 2\,b$ is not born out in experiments. Changing b or d in F_0 means changing the graph of F_0, including the number and/or location of local minima of F_0. While it is the global minimum that is of chief interest, other minima (of $F_0(\boldsymbol{u}, s)$) close to the continuation path may influence the result of the algorithm by changing the continuation path. Because the synthetic test images used in this work (cf. Section 4.1) are constructed according to the algorithms' image model, we may expect the global optimum to be quite stable with respect to perturbations in b and

d. A compelling explanation for the parameter sensitivity documented in Section 4.3 then is that—in the ranges considered—perturbations to b and d do not cause the global optimum to change but cause the continuation path to change.

This situation is undesirable. We wish the objective function to determine the global optimum and the embedding to determine the continuation path. When this separation of concerns breaks down, engineering a robust algorithm becomes very difficult. In order to achieve this separation the embedding needs to be generalized.

There are many ways to generalize the embedding in Section 5.3.4. One way that is considered in the next chapter is to multiply the exponentials with an s-dependent weighting function

$$\delta(\Delta_{i,j,k}) \rightarrow w_\Delta(s) \times \exp\left(-\Delta_{i,j,k}^2/(s\,\rho_k)^2\right) \tag{5.18}$$

$$\delta(u_{i,k}) \rightarrow w_u(s) \times \exp\left(-u_{i,k}^2/(\min(1, 2\,s)\,\rho_k)^2\right) \tag{5.19}$$

with ρ_k and h as in Section 5.3.4.

Chapter 6

Experiments and Results

In the first two sections of this chapter I present experimental results for synthetic images that show the merits of my new MDL criterion (F_1) and the new embedding developed in Section 5.3. The evaluation is similar to Section 4.3. That is, for each algorithm variant one parameter is varied and the noise reduction (4.2) is measured for each parameter value. Larger noise reduction (NRED) means more complete separation of the stochastic and deterministic components of a test image. In case of F_0 parameter b is varied and d is kept fixed at $d = 0.01$.

Section 6.3 discusses *solver bias*, the phenomenon that an optimization algorithm may perform well on some problem instances and poorly on others. More specifically, I demonstrate that with Leclerc's algorithm there is a trade-off between sensitivity for low contrast-to-noise (CNR) contours on the one hand and the ability to reconstruct regions with curvature on the other hand. This trade-off may be mitigated with a more general form of embedding.

Sections 6.4 and 6.5 discuss reconstruction results for natural images (photographs). I find that the class of reconstruction algorithms studied in this work significantly outperforms a state-of-the-art edge detector such as the Canny detector [15] on the Berkeley boundary detection benchmark. This is a new result.

6.1 Algorithms Using Old and New Embedding

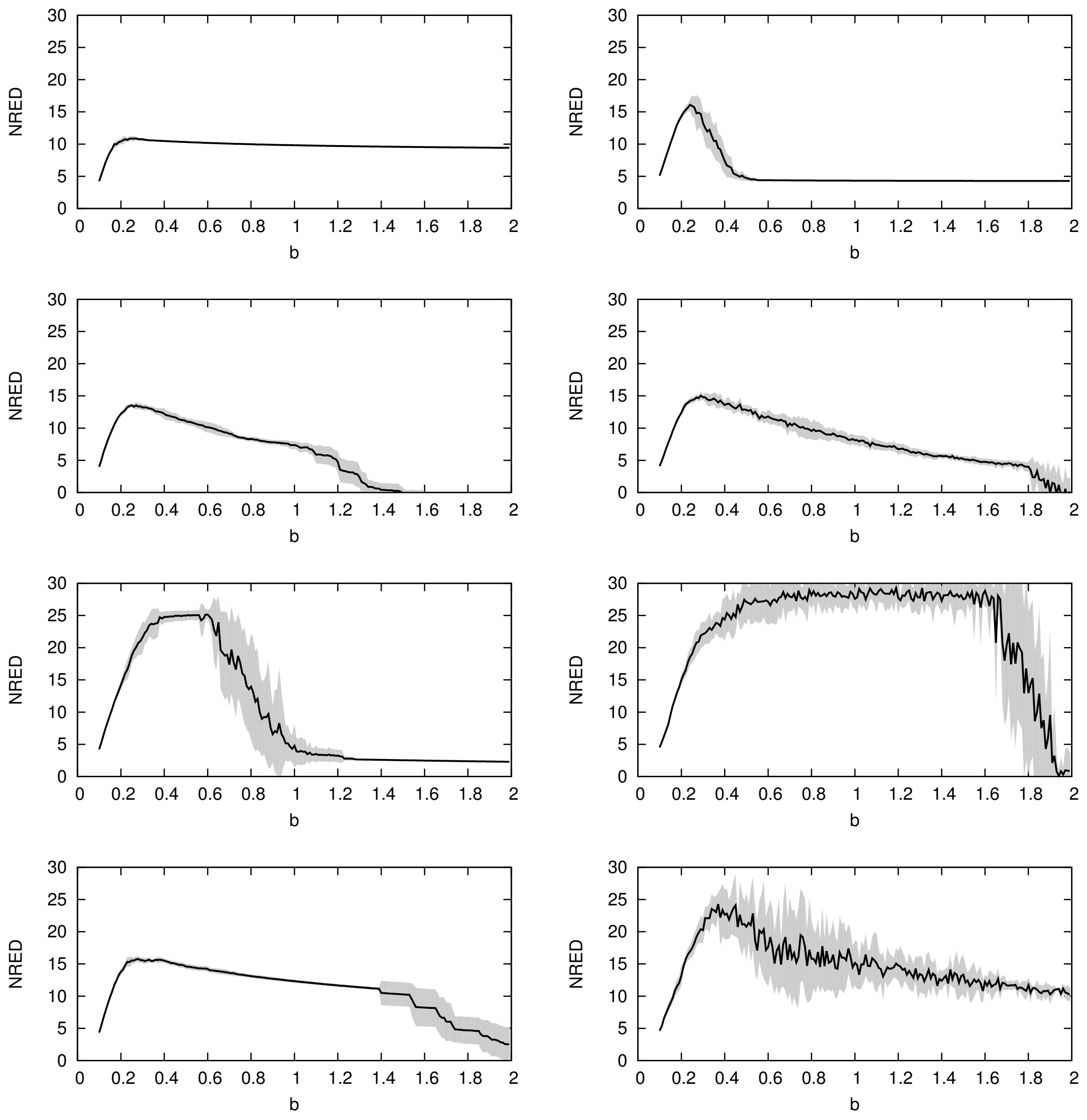


Figure 6.1. Average noise reduction vs. parameter value for an algorithm based on F_0 using Leclerc's embedding described in Section 2.5.1 (left column), and using my new embedding from Section 5.3.4 (right column), respectively. Rows a) to d) correspond to test cases 1 to 4. Ten different noise realizations were added to each test case's generating image and the resulting NRED values averaged. The black line shows the average NRED value, the gray band its standard deviation, as a function of parameter value.

In the first experiment algorithms based on F_0 are compared. One of the algorithms uses Leclerc's embedding described in Section 2.5.1 (Figure 6.1, left column); the other algorithm uses my embedding, described in Section 5.3.4 (Figure 6.1, right column).

The first thing to note from Figure 6.1 is that the best results achievable are all of higher quality with the new embedding (Figure 6.1, right column) than with the embedding proposed by Leclerc. This shows that the details of the embedding are as crucial for the success of the algorithm as is the choice of parameter values.

As in Section 4.3 above I consider a reconstruction result *good* when it agrees *qualitatively* with the good known solution, that is, when it has the expected number of regions and the expected polynomial degree in each region. Inspecting individual reconstruction results from the experiments summarized in Figure 6.1 I find that results with an NRED value of about 15 or larger are good. Hence, we can tell from the NRED curves whether an reconstruction algorithm is successful for a given parameter value and test image. From the NRED curves in the first row of Figure 6.1, for example, we see that with Leclerc's embedding the algorithm fails for test image 1 for all choices of parameter b.

Note that with the new embedding a qualitatively correct reconstruction of test case 1 (Figure 4.7 a) can be achieved and that, in turn, we have found a choice of parameter values which yields good results for all four test cases (i.e., $b \approx 0.3$, $d = 0.01$). Also interesting is that with the new embedding the quality of results for test case 3 is much less sensitive to the choice of parameters (cf. Figure 6.1, third row).

6.2 Algorithms Using Old and New MDL Criterion

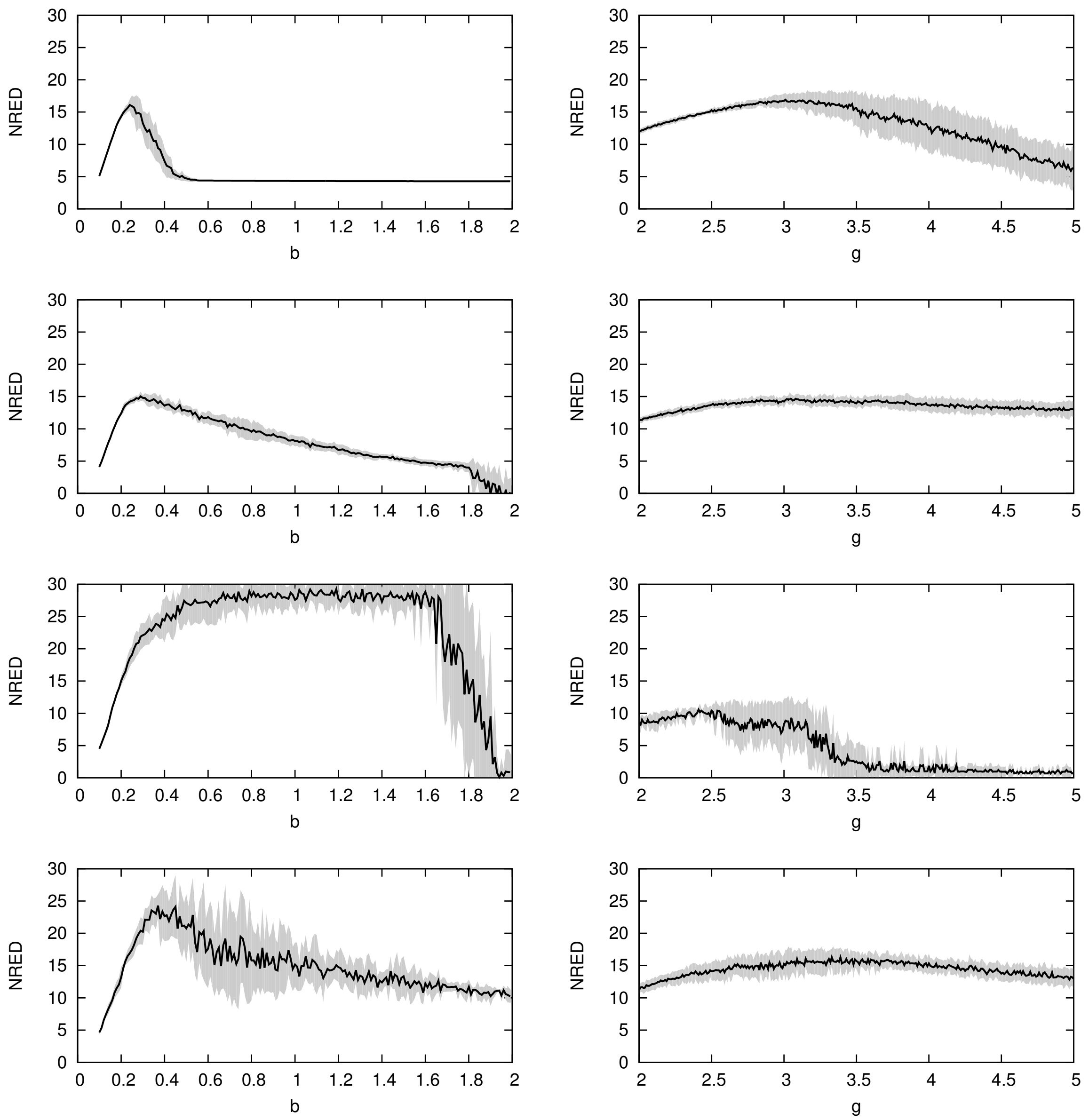


Figure 6.2. Average noise reduction vs. parameter value for algorithms based on Leclerc's objective function F_0 (left column) and my objective function F_1 (right column). The left column contains the same graphs as the right column of Figure 6.1.

In the next experiment I test the viability of the new MDL criterion (F_1). Two algorithms that result with the two objective functions F_0 (Leclerc) and F_1

(Juengling), respectively, are compared. Both algorithms use my new embedding.

Figure 6.2 juxtaposes NRED curves for the two algorithms for test cases 1 to 4. First note that the MDL criterion F_1 "works", that is, it may be used as a basis for the reconstruction algorithm. Examining the curves in Figure 6.2 one finds that the F_1-based algorithm achieves good results for three out of the four test cases. For test case 3 the F_1-based algorithm fails to reconstruct the quadratic region for all parameter choices (NRED clearly below 15).

Figure 6.2 also shows that the F_1-based algorithm's quality of result is much less sensitive to the choice of parameter value than the F_0-based algorithm. While the F_0-based algorithm requires a value b close to 0.3 for good results, the F_1-based algorithm produces good results for values g in the range 2.5 to 3.5, except for test image 3 (see the corresponding results in the third row of Figure 6.2).

6.3 Solver Bias

An optimization algorithm performing better on certain problems than on others is exhibiting *bias* towards the problems it solves well [79]. In Section 5.5.2 I conjectured that the parameter sensitivity of Leclerc's algorithm is a manifestation of solver bias, and not of unstable optimal solutions. In this section I revisit this hypothesis, provide experimental evidence for it, and use the insights gained from these experiments to engineer a generalized embedding with which an algorithm using the F_1 objective function succeeds on test case 3.

6.3.1 Low-CNR vs. Curvature

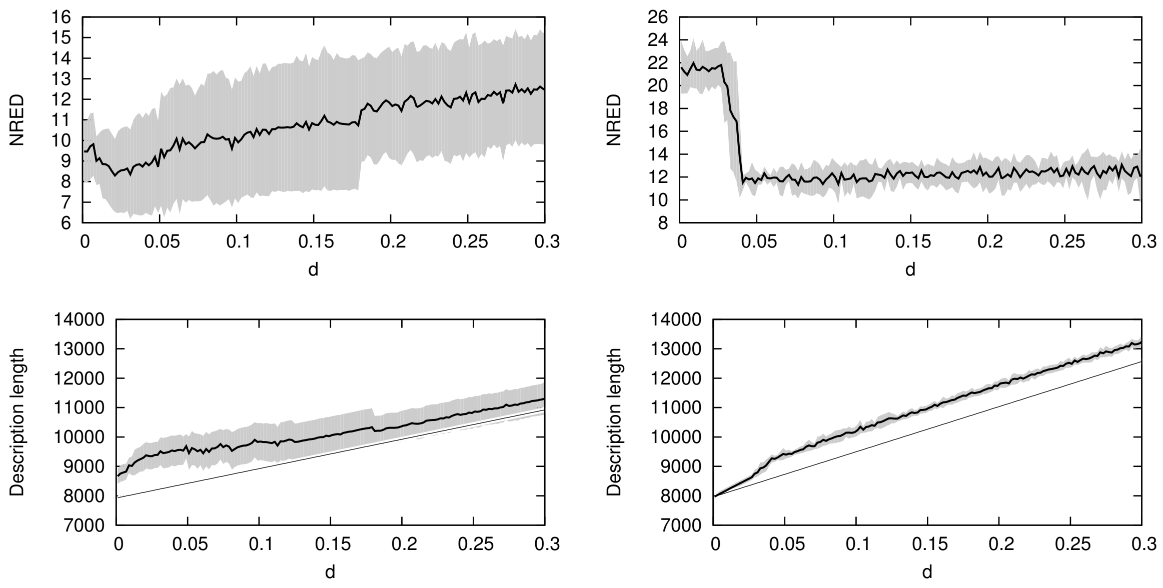


Figure 6.3. Noise reduction vs. parameter d value (top row) for an algorithm using F_0 and my new embedding. The value for parameter b fixed at 0.3. The curve on the left corresponds to different noise-variants of test case 1, the curve on the right to variants of test case 3. The curves in the bottom row show desription length vs. parameter d value for the same experimental data (i.e., value of $F_0(\boldsymbol{u}^*, s_f)$ with $s_f = 0.01$). The thinner plot line shows $F_0(\boldsymbol{u}^g, s_f)$, the description length of the good known solution.

Test cases 1 and 3 may serve as examples to demonstrate solver bias in the continuation method. Figure 4.6 in Section 4.3 already contains clues that these two problems may require opposing algorithm tunings. In this experiment I evaluate the quality of results for these two test cases as it depends on parameter d (the CNR-ratio of test case 1 is decreased to about 1.43 to exhibit the solver bias phenomenon more clearly).

Figure 6.3 shows that the quality of results for test case 1 improves when d is increased. For test case 3, on the other hand, we observe the opposite behavior: the continuation method fails to find good solutions when d is increased beyond a certain threshold.

The optimal solutions to the two problems are essentially constant for the d-values considered in this experiment (cf. the bottom plots in Figure 6.3 comparing optimization results against the good known solution). This means that changing d is affecting the continuation path and not the location of the best solution. Higher d values bias the algorithm towards recovering lower CNR contours (the characteristic feature of test case 1), whereas lower d values bias the algorithm towards recovering regions with curvature (the characteristic feature of test case 3).

That the d-parameter might have such an effect is in line with my claim (Section 4.5.1) that the piecewise polynomial algorithm has a higher CNR threshold for detecting discontinuities than the piecewise constant algorithm. Since the d-parameter weights the terms which penalize the use of coefficients $u_{i,k}$, $k > 0$, increasing d gradually restricts the degrees of freedom offered by the higher-degree coefficients (using a physical metaphor we might say that increasing d makes the finite elements $\boldsymbol{u}|_{\Omega_i}$ more *stiff*). This causes the values of the difference terms $\Delta_{i,j,0}(\boldsymbol{u}^{*,k})$ of intermediate solutions $\boldsymbol{u}^{*,k}$ to be larger, and thereby effects a reduction in the CNR threshold. To complete the argument, consider that for large values of d the piecewise polynomial objective function has the same optima as the piecewise constant objective function.

6.3.2 Controlling Solver Bias with a Generalized Embedding

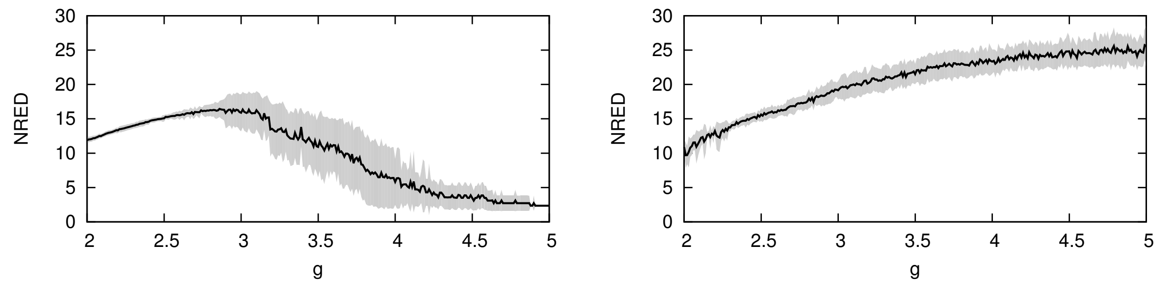


Figure 6.4. Noise reduction vs. parameter value with an algorithm using F_1 and the embedding depicted in Figure 6.5. The curve on the left corresponds to Figure 6.2 a), the curve on the right to Figure 6.2 c) (right column). The solutions for test case 3 are now excellent, while the solutions for test case 1 degrade and test case 4 (not shown) degrade slightly.

The reconstruction algorithms have difficulty with test case 3: using F_1 the algorithm fails to find a good solution for any value of parameter g, and using F_0 the parameter values need to be finely tuned. As I discuss above in Section 5.5.2 and in the previous section, we need to modify the embedding to change the reconstruction algorithm's biases.

In the previous section we see that the continuation method succeeds for test case 3 when the coefficient-term in F_0 is weighted lightly. I now use this insight to construct a generalized embedding which biases the algorithm towards finding solutions for test case 3. With a generalized embedding that leaves the weighting of the $\Delta_{i,j,k}$-terms alone (i.e., $w_\Delta(s) = 1$), and a function w_u which transitionally decreases the weighting of the $u_{i,k}$-embedding (Figure 6.5), the F_1-based algorithm finds good solutions for test case 3 as well (Figure 6.4).

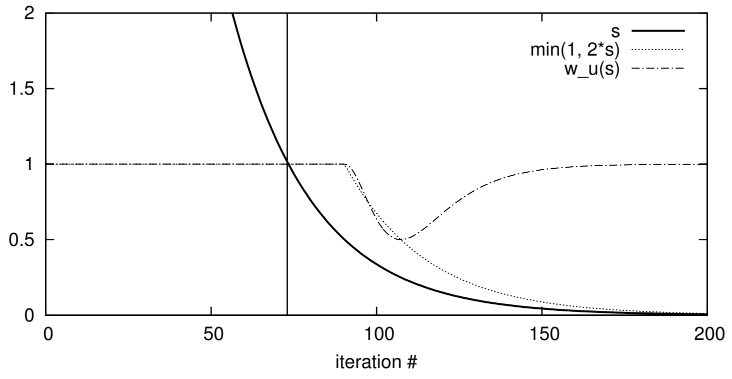


Figure 6.5. The graph of $w_u(s) = \begin{cases} 1, & 1/2 \leq s \\ \frac{1}{2} - \frac{1}{4}\cos\left(\frac{2\,\pi}{1/2 - s_f}\,2\,s\right), & s_f < s < 1/2 \end{cases}$ over iterations is shown as a dashed line. For context the other two curves show $1/s$ and $1/\min(1, 2s)$, respectively (cf. the embedding in Section 5.5.2). The point $s = 1$ is marked by a vertical line.

6.4 Results for Natural Images

This section considers the quality of results of the F_1-based reconstruction algorithm when applied to real-world images. Since good known solutions are not available for these images, the discussion focuses on qualitative aspects of reconstruction results. The test images[6.1] used as examples contain mostly smooth surfaces and little texture in order to be in accordance with the smooth-surfaces assumption set forth in Section 1.2.

6.4.1 Boundary Detection

As discussed in Section 3.1.2, discontinuities are used as features in recognition tasks, where it is assumed that the silhouette contours of objects are included in the discontinuity set. Figure 6.6 shows the discontinuities in the reconstruction of the "peppers" image from Chapter 1, and the discontinuities for the image of a house, respectively. The results in the second row in Figure 6.6 are obtained with my reconstruction algorithm. The results in the third row are obtained with my implementation of Kanungo et al.'s reconstruction algorithm [46].

As expected, both algorithms generate discontinuities that correspond to silhouette contours, as these are points of large intensity contrast. Low-contrast parts of silhouette contours are found when a contour also includes high-contrast parts. See, for example, the contour in the lower left of the "peppers" image which separates a partially visible light pepper from an elongated, upright standing light pepper.

6.1. The two images are from `http://decsai.ugr.es/~javier/denoise/test_images/index.htm`.

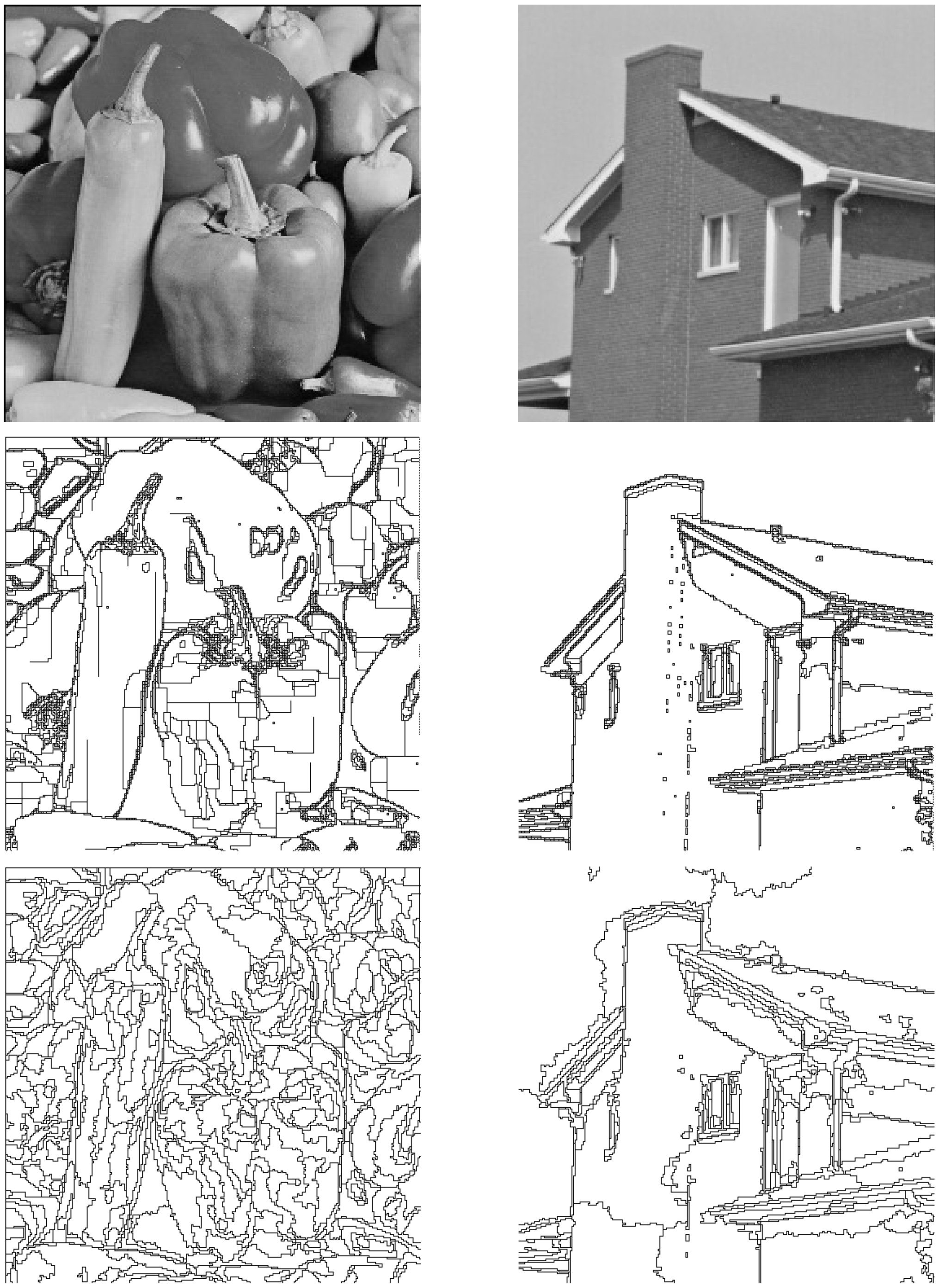


Figure 6.6. Peppers image and C^0-discontinuities in its reconstruction (left column). The same for house the image (right column). The discontinuities in the middle row were obtained with my reconstruction algorithm; the continuation method was terminated at $s_f = 0.1$. The results in the bottom row were obtained with the reconstruction algorithm of Kanungo et al. [46], which is briefly discussed in Section 3.3.

Both algorithms also produce some discontinuities at points of low contrast, and which do not correspond to an object's silhouette (see, for example, contours in the sky region in the lower right image of Figure 6.6). We might call such contours "spurious", for they do not correspond to a true discontinuity in the unobserved generating image. Spurious contours may occur when there is no low-order polynomial to accurately and completely describe the image of a smooth surface.

The center image on the left in Figure 6.6 exhibits a number of predominantly vertical or horizontal contours. This phenomenon reflects a bias of the reconstruction algorithm towards straight contours[6.2], which may affect the shape of contours in low-contrast regions (that is, of spurious contours).

Another undesirable phenomenon on display in in Figure 6.6 are double discontinuities at silhouette contours (for example, the contour separating sky and brick wall in the house image). This phenomenon may be understood from the fact that pixels at object boundaries may not exclusively belong to one object's surface or another. The intensity of such pixels falls in between the two local intensities corresponding to the two sides of the boundary. When such "transition pixels" show a large intensity difference to pixels on either side of the object boundary, the reconstruction algorithms introduce discontinuities to their neighbors on either side.

Transition pixels are a consequence of the imaging process and are absent in the synthetic images of Section 4.1. One way to avoid double discontinuities is to include the blurring by the imaging process into the algorithm's image model[6.3] [48], a feature not included in my implementation.

6.2. Leclerc explains that this bias may be reduced by using an eight-connected pixel neighborhood instead of a four-connected neighborhood in the derivation of the objective function [48].

6.3. In order to model imaging blur, the observed image intensity $\boldsymbol{z}$ is assumed to be the sum of the convolution of the reconstruction $\boldsymbol{u}$ with some blurring kernel K and the residuals $\boldsymbol{r}$, $r_i = z_i - \sum_{j-i \in S} K_{j-i} u_{j,0}$, where S is the support of K. Including imaging blur in the algorithm's image model means using this definition of residual in the data term (2.5).

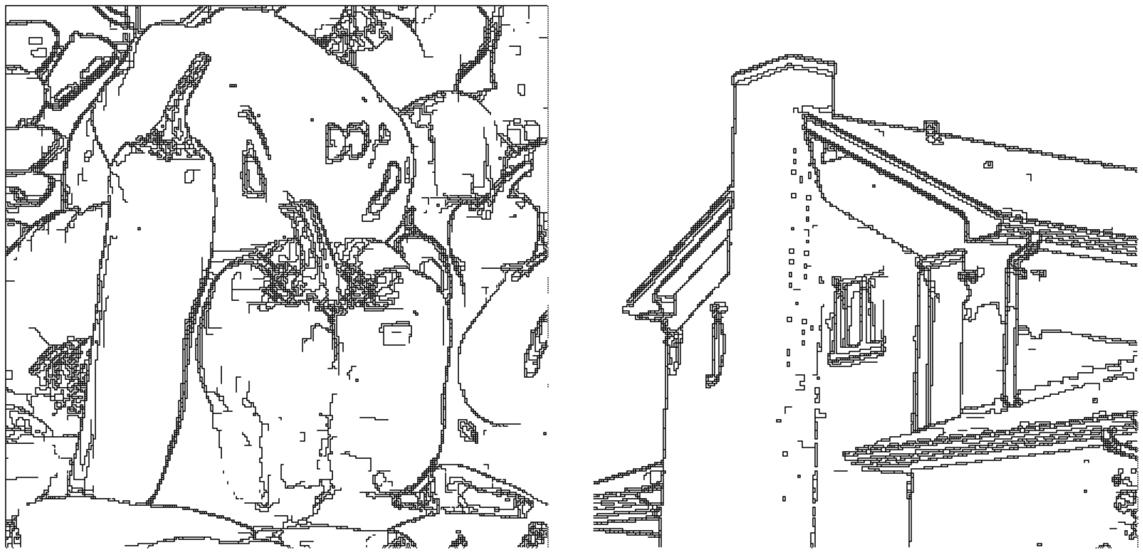


Figure 6.7. Discontinuities obtained similarly as in Figure 6.6, middle row, but the continuation method was stopped at $s_f = 0.5$. While the discontinuities for the house image are almost the same as in Figure 6.6, a number of spurious boundaries are absent in the result for the peppers image.

A useful feature of the continuation method is that discontinuities are introduced, roughly speaking, in the order of decreasing local image contrast. That is, higher-contrast discontinuities are being introduced early (cf. Figure 4.5). It is therefore possible to reduce the number of spurious boundaries by terminating the continuation method earlier. Figure 6.7 shows the results of my algorithm when the continuation method is stopped at $s_f = 0.5$. Note that there are fewer spurious contours in Figure 6.7 than in the results of Kanungo et al.'s algorithm in Figure 6.6, especially for the peppers image.

6.4.2 Image Denoising

To demonstrate that the reconstruction algorithm may be used to remove noise in images, I created test images by adding Gaussian noise to the peppers and house image, respectively (Figure 6.8, top row). The middle row of Figure 6.8 shows the denoised images produced by my reconstruction algorithm, and the bottom row are the results produced by a state-of-the-art wavelet-based denoising method [18].

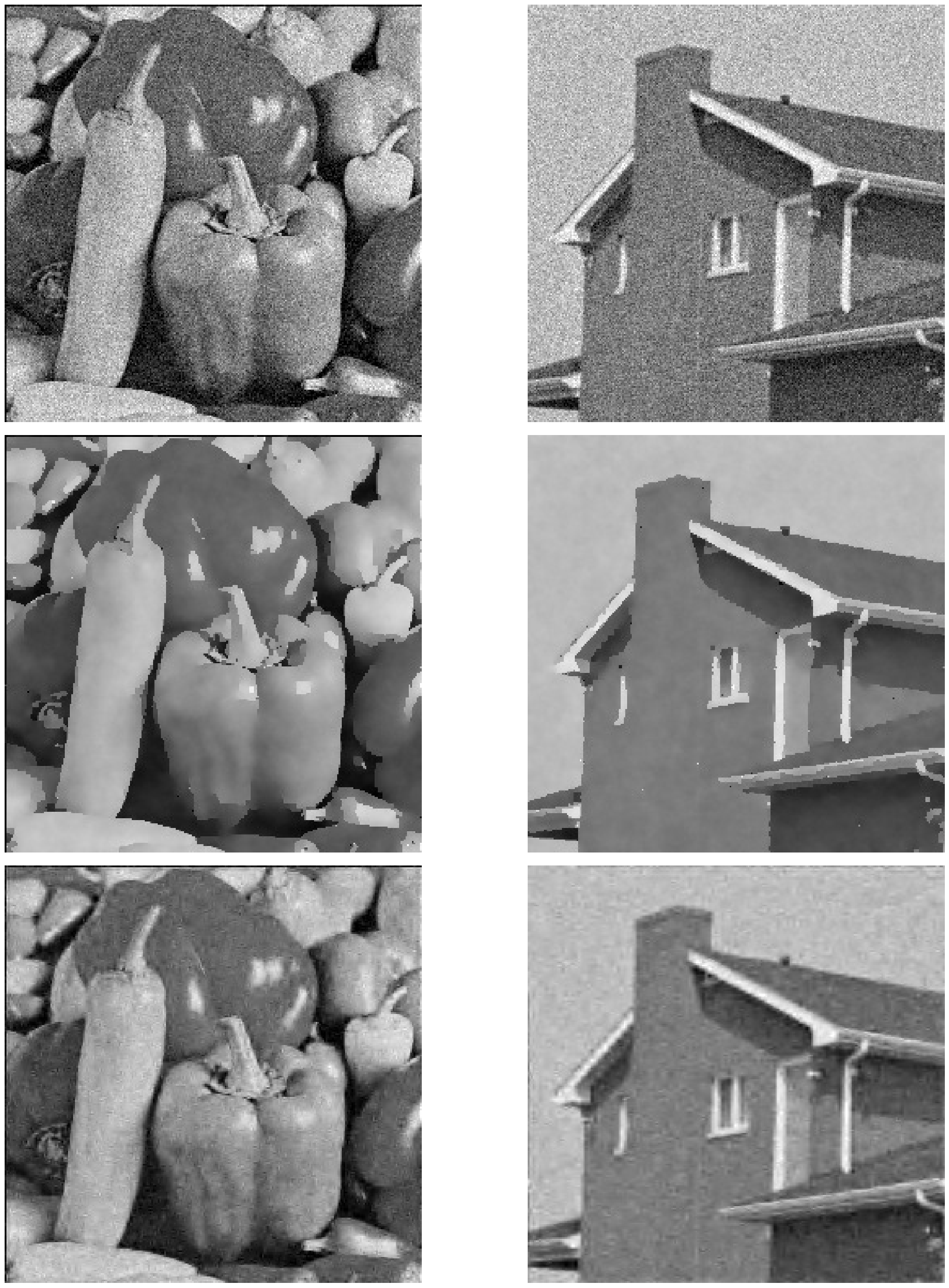


Figure 6.8. Peppers image with Gaussian noise added ($\sigma = 20$), its reconstruction, and denoised image (left column). Same for the house image (right column). The reconstructions in the middle row are obtained with my algorithm, where the continuation method was terminated at $s_f = 0.5$. The results in the bottom row are obtained with Chang et al.'s denoising algorithm [18].

The smooth regions in the results of my algorithm appear smoother than in the results of the wavelet-based method and, one might argue, resemble the noise-free images more closely in this regard (Figure 6.6, top row). Of course, smoothness is precisely the property my algorithm is designed to exploit in reconstructing the signal, whereas the wavelet-based method does not incorporate this assumption. On images which include more texture, a denoising algorithm may be expected to do better than the reconstruction algorithm.

6.5 Evaluation on a Boundary Detection Benchmark

The quality measure used in the foregoing sections, the noise reduction (4.2), is tailor-made for evaluating reconstruction algorithms (i.e., algorithms separating signal and noise). I use it in this work to gauge the benefits of different algorithms devised within Leclerc's methodical framework. In order to compare my algorithm to algorithms devised along other approaches, I shift the focus in this section to the related *boundary detection* task (Section 3.1.2). A widely used benchmark and dataset for this task is the Berkeley boundary detection benchmark [56].

In the following Section 6.5.1 I briefly describe the Berkeley boundary detection benchmark. The benchmark dataset consists of photographs spanning a wide variety of scenes, and the code that performs the benchmarking expects detection results in a particular data format. In Section 6.5.2 I describe what technical modifications I implemented to cater to these two aspects. Section 6.5.3 presents and discusses results for the benchmark.

6.5.1 The Berkeley Boundary Detection Benchmark

The Berkeley boundary detection benchmark consists of the dataset “BSDS300”, and of computer codes for automatic evaluation of boundary detection results for the images in the dataset. Both may be downloaded from a UC Berkeley server[6.4].

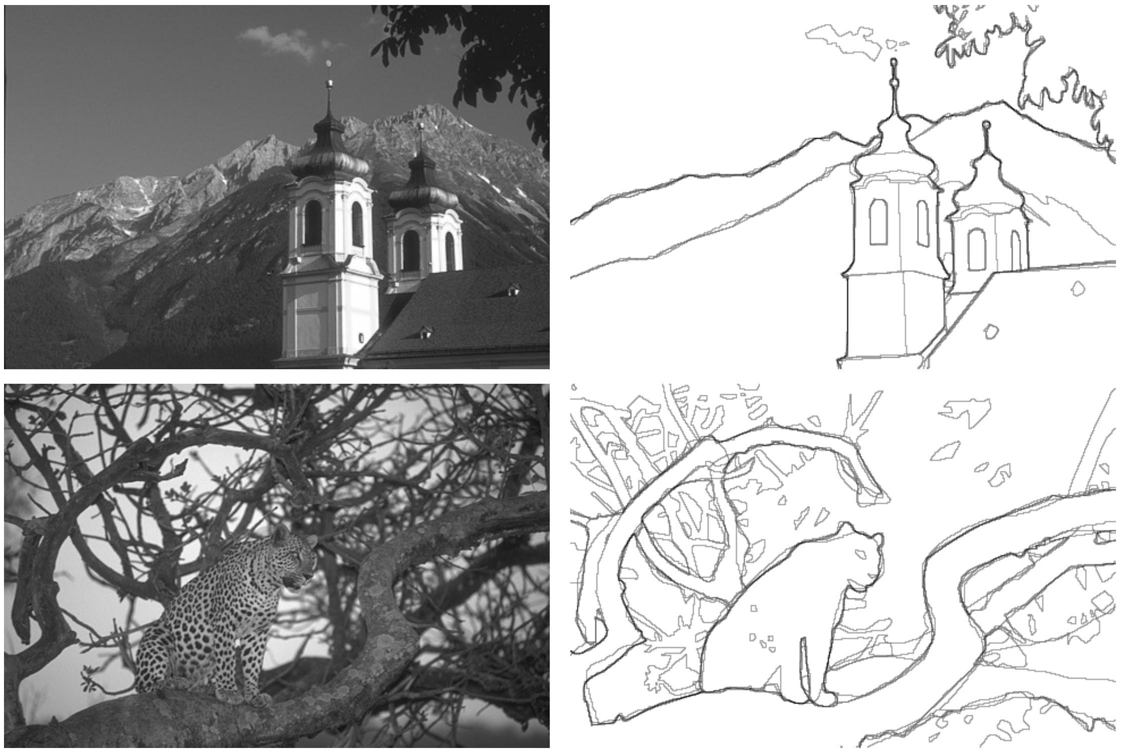


Figure 6.9. Example images and overlaid contours from human-generated segmentations from the BSDS300 dataset. Darker contours have been selected by more human subjects.

The images, 481x321 pixels in size, are a subset from the "Corel image database" (see Figure 6.9 for two examples). Martin et al. say they "chose images of natural scenes that contain at least one discernible object." ([55], page 418).

Human subjects were instructed to segment these images assisted by an image segmentation tool. For each image, five to ten human segmentations were collected. The segmentations define contours which are shown overlaid on the right in Figure 6.9. As this figure makes clear, segmentations from different subjects may very well be

6.4. `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench` different. There are two kinds of differences: small differences in the exact localization of a contour, and differences in the number of contours. With regards to differences in the number of contours the authors remark that “Even if two observers have exactly the same perceptual organization of an image, they may choose to segment at varying levels of granularity.” ([55], page 417).

6.5.1.1 Evaluation Methodology

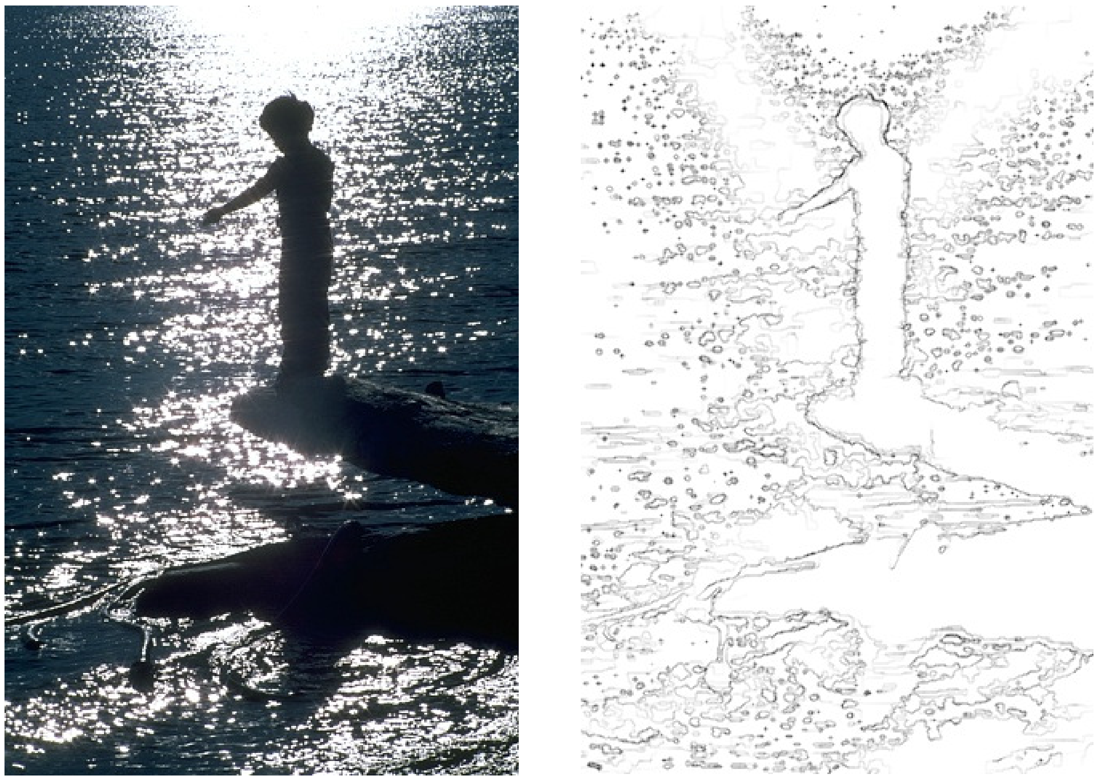


Figure 6.10. Image from the BSDS300 dataset (left) and a soft boundary map (right).

In a later publication, Martin et al. propose a performance measure which applies to algorithms for image segmentation as well as to algorithms for boundary detection. It is based on a concept called a *boundary map*. A boundary map for an image is a labeling of an image’s pixels as being either part of a contour or not part of a contour. Assume there is ground truth which tells for each pixel whether it truly is part of a contour or not. The quality of a *detected* boundary map may then be expressed by two numbers, *precision* (P), the fraction of all detected boundary pixels that are true, and *recall* (R), the fraction of all true boundary pixels that were detected. Together precision and recall mark a point (P,R) in the interval $[0,1]\times[0,1]$. Higher precision and recall values mean better results. A detection result which reproduces the ground truth perfectly corresponds to the point (1,1).

Instead of simply labeling pixels, many algorithms may quantify (on some continuous scale) a degree of certainty that a given pixel is part of a contour or not. If this scale goes from zero to one then these quantities may be interpreted as pixel-wise probabilities for the presence of a contour. Martin et al. call such a detection result a *soft boundary map* (see Figure 6.10 for an example). Note that one may obtain a boundary map from a soft boundary map by *thresholding*, that is, by labeling all pixels with probability larger than some threshold t, $t\in[0,1]$ as contour pixels. Thus, for each threshold t we get a point (P_t, R_t) in the precision-recall interval, and, taken together, these points make up what is called a *precision-recall curve*. Martin et al. write

> "The precision and recall measures are particularly meaningful in the context of boundary detection when we consider applications that make use of boundary maps, such as stereo or object recognition. If it is reasonable to characterize higher level processing in terms of how much true signal is required to succeed R (recall), and how much noise can be tolerated P (precision). A particular application can define a relative cost α between these quantities, which focuses attention at a specific point on the precision-recall curve. The *F-measure*, defined as
>
> $$F = PR/(\alpha R + (1-\alpha)P) \tag{6.1}$$
>
> captures this trade off as weighted harmonic mean of P and R. The

> location of the maximum F-measure along the curve provides the optimal detector threshold for the application given α, which we set to 0.5 in our experiments.” ([56], page 536).

To compute precision and recall, boundary maps derived from the human-generated segmentations serve as ground truth. As we observed above, this ground truth data is not accurate in terms of boundary-localization, and detected boundary pixels need to be brought into correspondence with ground-truth boundary pixels in some way that is tolerant of these location errors (see Martin et al. [56] for how this problem is solved in the Berkeley benchmark).

For each image in the dataset, there are five to ten ground-truth boundary maps from the human segmentations. To generate a single precision-recall curve from these multiple ground truths, Martin et al. correspond detected boundary pixels to each ground-truth boundary map in turn. Only those detected boundary pixels not matching any ground-truth boundary pixel are counted as false positives. The recall values are averaged over the different ground-truths, “so that to achieve perfect recall the machine boundary map must explain all of the human data.” ([56], page 536)

Finally one precision-recall curve for the whole dataset is produced, which expresses precision and recall for the entirety of the data.

6.5.1.2 Training Set vs. Test Set

The BSDS300 dataset in fact consists of two datasets. The purpose of the first set, consisting of 200 images plus ground-truth data, is to *train* or tune a contour detection algorithm. That is, benchmark results obtained with this *training set* may suggest modifications to the algorithm which could improve results.

The second dataset, the *test set*, is to be used for the actual evaluation and to be reported in publications. It consists of 100 images plus ground-truth data.

6.5.2 My Algorithm for the Boundary Detection Task

In Sections 6.1 to 6.4 of this chapter I discuss results of reconstruction algorithms where σ is assumed to be constant and known. To the images of the BSDS300 datasets these assumptions do not apply. For the boundary detection task posed by the Berkeley benchmark I therefore use the more complicated algorithm outlined in Section 2.6, which performs region-wise estimation of the variance as part of the reconstruction process.

Note that the “discontinuity maps” I present in Section 6.4.1 above (cf. Figure 6.6) are not like the soft boundary maps expected by the benchmarking code. The discontinuity elements subject to labeling are not pixels, but pixel boundaries, and, secondly, the labels are binary instead of “soft” probabilities of contours.

To obtain “soft” contour indicators I record the *stability* of the discontinuity elements, which is the value of the continuation parameter s at which a discontinuity first occurs (see also Figure 6.7 and the discussion at the end of Section 6.4.1). The stability is zero when no discontinuity was detected. From this “stability map” I obtain a soft boundary map in two steps. First, each pixel in the map is assigned the average of the pixel boundaries’ stability values. Next the logarithm of the averages is taken (since the continuation parameter values are equally spaced on a log-scale), and the result is rescaled so the map values span the range $[0, 1]$. Figure 6.10 above shows an example of a soft boundary map obtained this way.

Finally, to increase precision I apply a standard image processing technique and remove small connected components from the stability map (see Figure 6.11 for an example). This a reasonable measure since the images in the dataset “contain at least one discernible object” [55], which means that the boundaries of objects of interest are generally much longer than boundaries attributable to distant background objects, clutter, or small-scale artifacts.

Table 6.1 lists the parameter values I used to obtain the results reported in the following section. My experiments also included a version of the algorithm using Leclerc's objective function F_0, and I found it to perform equally well with certain parameter combinations (but not better).

Objective function	g	o	connected-component size
F_1	4	100	250

Table 6.1. Parameter values used in the evaluation runs on the Berkeley boundary detection benchmark.

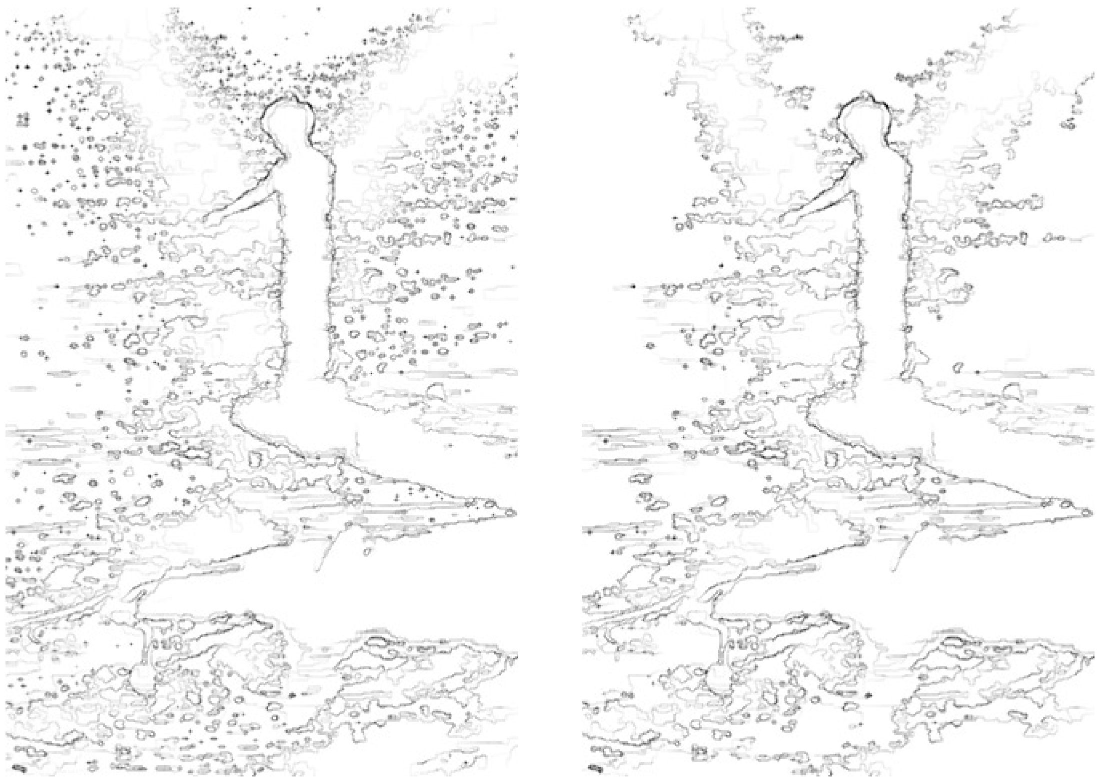
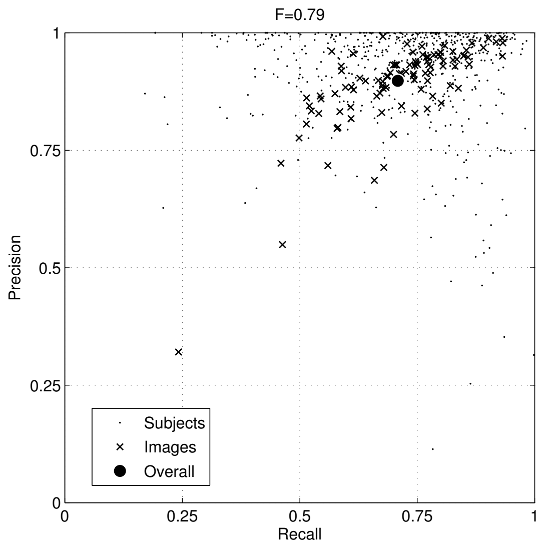


Figure 6.11. Soft boundary map before removal of small connected components (left), and after removal (right).

6.5.3 Results on the Berkeley Benchmark



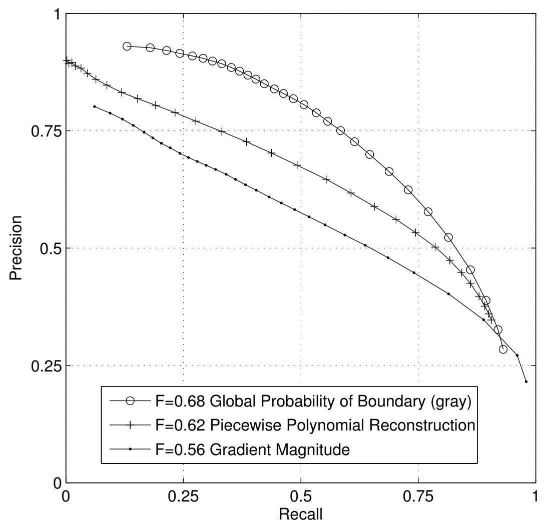


Figure 6.12. Precision-recall points for human segmentations (left), and precision-recall curves for three different algorithms (right).

The precision-recall curve of a detector with perfect precision would consist of the points on the upper boundary of the precision-recall interval $\{(x,1)|x\in[0,1]\}$. Coming close to this result is unrealistic for the Berkeley boundary detection benchmark, however, considering that matching the different human ground-truth data for a given image do typically not result in such points (see Figure 6.12, left). In other words, humans do not achieve perfect scores on this benchmark, and it seems implausible that machines could do any better.

To rank different algorithms the authors of the Berkeley benchmark use the F-measure (6.1), which is a single number derived from a precision-recall curve. It turns out that the practical range of F-measure values is $[0.41, 0.79]$, where the upper bound 0.79 is the score achieved by human subjects, and the lower bound 0.41 is the score achieved by an algorithm producing random soft boundary maps.

Figure 6.12 (right) shows the precision-recall curve "Piecewise Polynomial Reconstruction" that I obtain with my reconstruction algorithm described in the previous section. To put my result into perspective, the diagram also shows a result demonstrating current state-of-the-art performance for boundary detection on grayscale images ("Global Probability of Boundary" [3]), and a result obtained with a standard edge detector similar to Canny's [15] ("Gradient Magnitude")[6.5].

The figure shows that the reconstruction algorithm performs significantly better than a standard edge detector, but it does not achieve state-of-the-art performance. Example results are shown in Figure 6.13 below and in Appendix D.

6.5.3.1 What Makes a Good Boundary Detector?

To gain some understanding as to why the reconstruction algorithm performs as it does on the Berkeley benchmark, consider Figure 6.13. It shows some results for an image from the BSDS300 test dataset. The first thing to note is that the ground-truth contains fewer boundaries than are present in the machine's soft boundary maps, and that most if not all ground-truth contour pixels are contained in a machine's boundary map for some threshold. This situation is typical and what it means is that it is easy (for most images) to achieve high recall, but difficult to achieve high precision. Algorithms incorporating strategies designed to boost precision may therefore expected to be superior. For example, the contour marking the shoreline in the marina picture of Figure 6.13 is assigned higher probability by the "Global Probability of Boundary" algorithm than by the other two algorithms. This puts it ahead of the other two algorithms because the shoreline contour is included in the ground truth.

There are two general strategies for improving precision over a standard edge detector. The first strategy is to search for better local criteria for determining presence of a boundary [69]. The second strategy is to identify *salient* contours in the soft boundary map and to assign higher probability to them [73].

6.5. The detector uses Gaussian derivatives and non-maxima suppression just like the Canny edge detector. The three parameters of this detector—the scale of the Gaussian, and two parameters in the logistic function mapping detector response to a probability—were determined by optimizing detector performance on the training dataset.

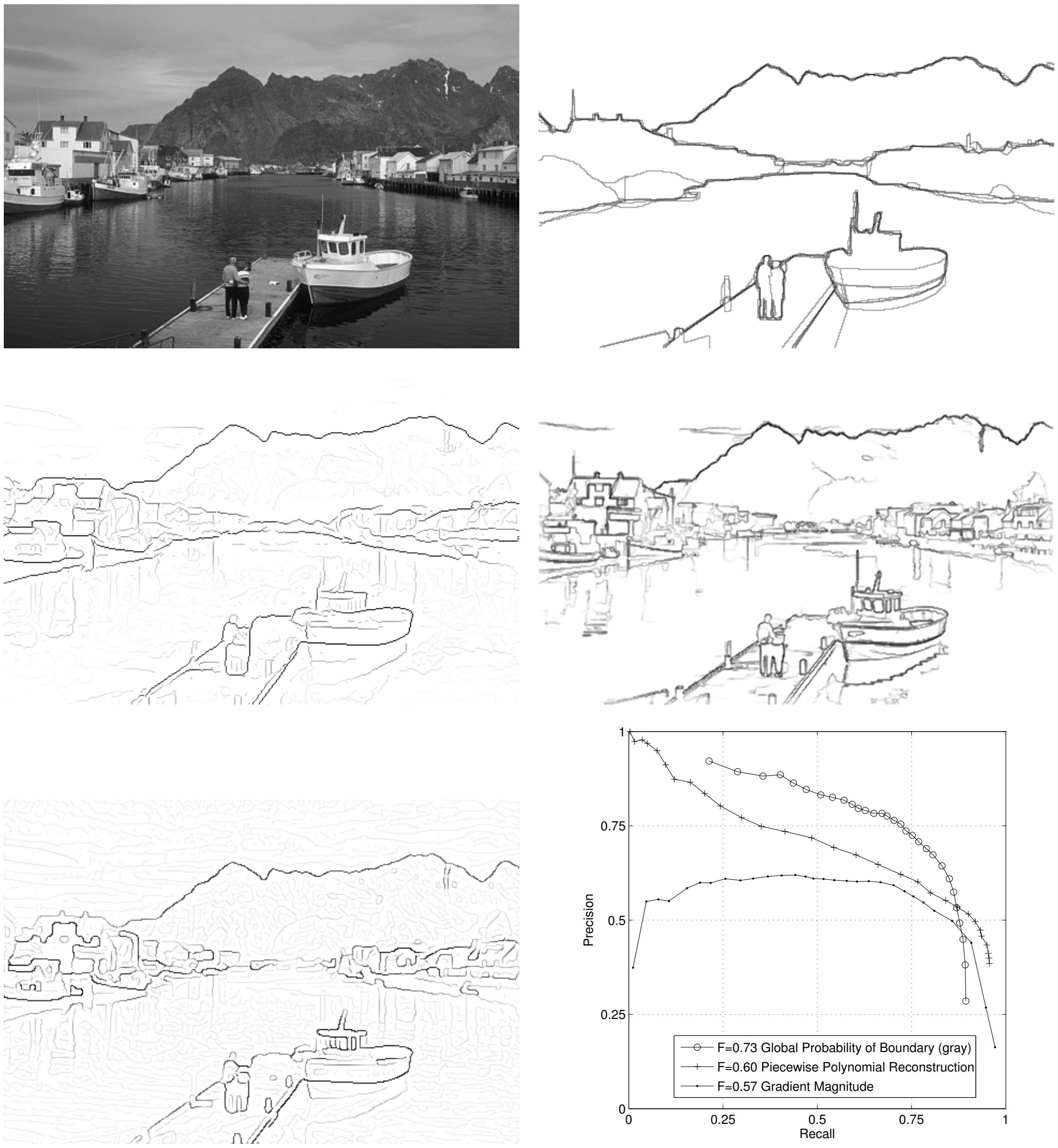


Figure 6.13. Results for a particular image from the BSDS300 test dataset. Top row: grayscale image (left) and ground-truth boundaries from human segmentations overlaid (right). Middle row: soft boundary maps by "Global Probability of Boundary" [3] (left), and by my "Piecewise Polynomial Reconstruction" algorithm (right). Bottom row: soft boundary map by "Gradient Magnitude" [56] (left), and precision-recall curves for the three named algorithms. Darker tone in the soft boundary maps mean higher probability of boundary.

The first strategy is pursued systematically by Martin et al. [56], who use a combination of histogram-based change detectors sensitive to brightness, color, and texture changes, respectively, and who cleverly combine them into a single boundary detector. More recent improvements in state-of-the-art performance on the Berkeley benchmark are achieved by the second strategy (boosting salient contours), also called a "globalization" step by some authors [3].

The reconstruction algorithm *is* a globalization strategy by design, that is, the problem of finding contours is posed in the form of a global optimization problem. Simply put, my reconstruction algorithm ("Piecewise Polynomial Reconstruction" in Figure 6.13) incorporates the second strategy but not the first. This explains why it performs better than a standard edge detector, which uses neither strategy, and worse than a state-of-the-art contour detector, which uses both.

6.6 Summary and Discussion

With my new embedding, the continuation method appears to be working more reliably than with the embedding used by Leclerc; good solutions are found for more problems and over a wider range of parameter choices (Section 6.1). Combining this embedding with my new MDL criterion (F_1) produces an even more reliable reconstruction algorithm in terms of parameter sensitivity, with the exception of test case 3 (Section 6.2).

An investigation into the failure on test case 3 points to solver bias. Solver bias may be observed with either MDL criterion and is a plausible explanation of the parameter sensitivity observed in Section 4.3. I argued that it is the embedding and not the objective function that is the appropriate place in which to address problems with solver bias. Section 6.3 gives an example: insight into how the weighting of the d-term in F_0 influences the results for test case 3 is used to construct a generalized embedding which lets an algorithm using F_1 succeed on all four test cases.

Experimental results not discussed above show that the staggered optimization scheme for improved coefficient accuracy (Section 5.4) does not significantly improve results when the new embedding is used.

Application to natural images that meet the smooth surfaces assumption shows that my reconstruction algorithm produces useful results. When applied to denoising problems it excels at restoring the piecewise smooth appearance of an image. When used for detecting contours the algorithm makes silhouette contours explicit as discontinuities. It tends to generate double discontinuities when imaging blur is not taken into account in the image model, however.

An evaluation of the reconstruction algorithm on the Berkeley contour detection benchmark clearly shows that it is superior to standard edge detectors on natural images. This belatedly demonstrates the merit of global image analysis approaches pioneered in the mid-to-late 1980s by Leclerc and others [13, 30, 48].

Chapter 7

Solving the Optimization Problem

7.1 Methods for Solving Unconstrained Problems

Section 2.5.1 presents

$$\min_{\boldsymbol{u}} F(\boldsymbol{u}, s) \tag{7.1}$$

as the optimization problem to be solved repeatedly by the continuation method (F may be F_0 or F_1). Since there are no constraints on the variables $\boldsymbol{u} \in \mathbb{R}^{|I|\times 6}$ and the range of F is $\mathbb{R}$, problem (7.1) is an *unconstrained optimization problem* [45, 63]. General methods for solving unconstrained problems find *local minimizers*, that is, points $\boldsymbol{u}^*$ for which $F(\boldsymbol{u}^*, s) \leq F(\boldsymbol{u}, s)$ for all $\boldsymbol{u}$ in a small neighborhood of $\boldsymbol{u}^*$. Although these methods may only find local minimizers, in the context of a continuation method with a perfect embedding and a sufficiently small step size for the continuation method, those local minimizers are also global minimizers (cf. Section 2.5.1).

This section introduces general ideas behind the methods studied in this work. Section 7.2 describes those methods and presents numerical results demonstrating a significant runtime improvement of the overall algorithm developed in this work compared to that of Leclerc [48]. Section 7.3 gives more results detailing the individual runtime improvements of different techniques employed in this work.

To keep the notation simple in this chapter $F(\boldsymbol{u}, s)$ is abbreviated to $F(\boldsymbol{u})$ when the continuation parameter is not relevant to the discussion.

7.1.1 Solving a Nonlinear System

A necessary condition for a minimizer $\boldsymbol{u}^*$ of F is

$$\nabla F(\boldsymbol{u}^*) = 0. \tag{7.2}$$

One strategy for solving problem (7.1) may therefore be to search for solutions of the nonlinear system (7.2). However, since maximizers and saddle points satisfy system (7.2) as well, additional strategies or arguments are required to guarantee that a solution of (7.2) is also a solution of (7.1).

7.1.2 Line-Search Methods

A broad class of algorithms for solving problem (7.1) are iterative. They start with a given starting point $\boldsymbol{u}^0$ and generate a sequence of points $\{\boldsymbol{u}^k\}_{k>0}$ that converges to some local minimizer $\boldsymbol{u}^*$ of F, and for which $F(\boldsymbol{u}^k) < F(\boldsymbol{u}^{k-1})$ for all $k > 0$. In practice only a finite number of points of this sequence are computed and a method uses some heuristic *termination criterion* to decide when to stop. This means that minimizers are determined only approximately by the final point $\boldsymbol{u}^f$ in the computed sequence.

One way to compute the next point in the sequence is to generate a *descent direction* $\boldsymbol{d}$, for which

$$F(\boldsymbol{u}^k + \varepsilon\, \boldsymbol{d}) < F(\boldsymbol{u}^k) \;\text{ for all } \varepsilon \in (0, \delta]$$

is true for some $\delta > 0$, and then to take a step in that direction. Probably the best known descent direction is the steepest descent direction $-\nabla\, F(\boldsymbol{u})$ but other directions can be more effective. Methods generating and taking steps along descent directions are called *line search methods*.

The theoretically best step in a chosen descent direction $\boldsymbol{d}$ corresponds to the value $\alpha > 0$ minimizing $F(\boldsymbol{u}^k + \alpha\, \boldsymbol{d})$. Finding the best value α constitutes a new minimization problem. Trying to solve this minimization problem would be simple but inefficient and practical methods for solving (7.1) use other criteria for determining α.

Such criteria are designed to keep the number of evaluations of F small while at the same time guaranteeing[7.1] convergence of the sequence $\{\boldsymbol{u}^k\}_{k>0}$ to a local minimizer [57].

7.1.3 Quasi-Newton Methods

When the Hessian of F at $\boldsymbol{u}^k$ is nonsingular, then

$$\nabla^2 F(\boldsymbol{u}^k)\, \boldsymbol{d} = -\nabla F(\boldsymbol{u}^k)$$

may be solved for $\boldsymbol{d}$. This vector, called the *Newton direction*, is a descent direction whenever $\nabla^2 F(\boldsymbol{u})$ is positive definite. Methods using the Newton direction when applicable are called *Newton methods*. When close enough to a minimizer they converge quickly, reducing the distance to the minimizer quadratically per step.

Quasi-Newton methods use directions defined in a similar way,

$$B_k\, \boldsymbol{d} = -\nabla F(\boldsymbol{u}^k).$$

The matrices $\{B_k\}_{k>0}$ are generated by the method and are positive definite by construction so $\boldsymbol{d}$ is always a descent direction. These methods are called quasi-Newton because the matrices B_k are designed to approximate the Hessian $\nabla^2 F(\boldsymbol{u})$ in some way to achieve good local convergence behavior. Quasi-Newton methods may be used instead of a Newton method when either the Hessian is unavailable, costly to evaluate, or when solving for the Newton direction would be too costly.

7.1.4 BFGS Method

With the difference between $\boldsymbol{u}^{k+1}$ and $\boldsymbol{u}^k$ small enough, the Hessian predicts the change in gradient as

$$\nabla^2 F(\boldsymbol{u}^k)\, (\boldsymbol{u}^{k+1} - \boldsymbol{u}^k) \approx \nabla F(\boldsymbol{u}^{k+1}) - \nabla F(\boldsymbol{u}^k).$$

7.1. The search directions $\boldsymbol{d}$ also need to meet a requirement for this guarantee.

This observation motivates imposing the constraint

$$B_{k+1}(\boldsymbol{u}^{k+1} - \boldsymbol{u}^k) = \nabla F(\boldsymbol{u}^{k+1}) - \nabla F(\boldsymbol{u}^k),$$

or, with new notation $\boldsymbol{s}_k = \boldsymbol{u}^{k+1} - \boldsymbol{u}^k$, $\boldsymbol{y}_k = \nabla F(\boldsymbol{u}^{k+1}) - \nabla F(\boldsymbol{u}^k)$,

$$B_{k+1}\,\boldsymbol{s}_k = \boldsymbol{y}_k$$

on the matrices $\{B_k\}$.

With additional constraints like symmetry and positive definiteness update formulas—computing B_{k+1} from B_k, $\boldsymbol{s}_k$, and $\boldsymbol{y}_k$—may be derived. As B_k is positive definite $H_k = B_k^{-1}$ exists, and similar formulas may be derived for H_k as well. The so called BFGS method uses the update formula

$$H_{k+1} = \left(I - \frac{\boldsymbol{s}_k\,\boldsymbol{y}_k^T}{\boldsymbol{y}_k^T\,\boldsymbol{s}_k}\right) H_k \left(I - \frac{\boldsymbol{y}_k\,\boldsymbol{s}_k^T}{\boldsymbol{y}_k^T\,\boldsymbol{s}_k}\right) + \frac{\boldsymbol{s}_k\,\boldsymbol{s}_k^T}{\boldsymbol{y}_k^T\,\boldsymbol{s}_k}, \text{ for } k > 0, \tag{7.3}$$

and computes search directions $\boldsymbol{d} = -H_k\,\nabla F(\boldsymbol{u}^k)$. The matrix H_0 is user-supplied or a simple default like I is used. The derivation of (7.3) and analysis of the convergence behavior of the BFGS method may be found in textbooks on optimization [45, 63].

7.2 Comparing Runtime Performance

The number n of independent variables in $\boldsymbol{u}$ is a multiple of the number of image cells and can be very large. The size of matrices like B_k or H_k in Section 7.1.4 are quadratic in this number, making Newton and quasi-Newton methods that generate such matrices impractical. The next section explains how the quasi-Newton method used in this work avoids forming the matrices $\{H_k\}$.

Relaxation methods is another class of methods applicable when n is very large and the matrices involved are sparse. Essentially, in every iteration a relaxation method solves an equation for each independent variable. This requires little work when each variable depends only on a few other independent variables (sparsity). Section 7.2.2 describes the relaxation method used by Leclerc. The remaining sections cover numerical performance experiments and results.

7.2.1 Limited-Memory BFGS

The limited-memory BFGS method (L-BFGS) operates with a matrix $\tilde{H}_k$ similar to H_k as defined in (7.3). The difference is that $\tilde{H}_k$ does not depend on all previous iterations but on the vectors $\boldsymbol{s}_i$ and $\boldsymbol{y}_i$ obtained over the last m iterations only, and on a diagonal matrix $\tilde{H}_{0,k}$, which may vary from iteration to iteration.

The second difference to the BFGS method is that the matrix $\tilde{H}_k$ is never explicitly formed. Note that the BFGS method uses H_k to compute the product $H_k\,\nabla F$. Because $\tilde{H}_k$ is a low-rank modification of the diagonal matrix $\tilde{H}_{0,k}$, the product $\tilde{H}_k\,\nabla F$ can be computed per a series of dot products, scalar-vector, and vector-vector operations involving the vectors $\boldsymbol{s}_i$, $\boldsymbol{y}_i$, $i=k-m,\ldots,k-1$, and $\tilde{H}_{0,k}$. For example, if $m=2$ and $k>2$ the update formula (7.3) applied twice yields $\tilde{H}_k$ as

$$\tilde{H}_k=\left(I-\frac{\boldsymbol{s}_{k-1}\,\boldsymbol{y}_{k-1}^T}{\boldsymbol{y}_{k-1}^T\,\boldsymbol{s}_{k-1}}\right)\left(I-\frac{\boldsymbol{s}_{k-2}\,\boldsymbol{y}_{k-2}^T}{\boldsymbol{y}_{k-2}^T\,\boldsymbol{s}_{k-2}}\right)\tilde{H}_{0,k}\left(I-\frac{\boldsymbol{y}_{k-2}\,\boldsymbol{s}_{k-2}^T}{\boldsymbol{y}_{k-2}^T\,\boldsymbol{s}_{k-2}}\right)\left(I-\frac{\boldsymbol{y}_{k-1}\,\boldsymbol{s}_{k-1}^T}{\boldsymbol{y}_{k-1}^T\,\boldsymbol{s}_{k-1}}\right)$$
$$+\left(I-\frac{\boldsymbol{s}_{k-1}\,\boldsymbol{y}_{k-1}^T}{\boldsymbol{y}_{k-1}^T\,\boldsymbol{s}_{k-1}}\right)\left(\frac{\boldsymbol{s}_{k-2}\,\boldsymbol{s}_{k-2}^T}{\boldsymbol{y}_{k-2}^T\,\boldsymbol{s}_{k-2}}\right)\left(I-\frac{\boldsymbol{y}_{k-1}\,\boldsymbol{s}_{k-1}^T}{\boldsymbol{y}_{k-1}^T\,\boldsymbol{s}_{k-1}}\right)+\frac{\boldsymbol{s}_{k-1}\,\boldsymbol{s}_{k-1}^T}{\boldsymbol{y}_{k-1}^T\,\boldsymbol{s}_{k-1}}.$$

The L-BFGS method keeps the vectors $\boldsymbol{s}_i$, $\boldsymbol{y}_i$, and the dot products $\boldsymbol{y}_i^T\cdot\boldsymbol{s}_i$, $i=k-m,\ldots.k-1$, in memory and uses them to compute $\tilde{H}_k\,\nabla F$. If arranged carefully, this computation requires just $4\,m\,n+n$ multiplications [53, 63].

The termination criterion I use with this method is

$$\|\nabla_{\boldsymbol{u}}F(\boldsymbol{u})\|_2<\mathrm{gtol}\max\left(1,\|\boldsymbol{u}\|_2\right), \tag{7.4}$$

with $\mathrm{gtol}=10^{-5}$.

7.2.2 Nonlinear Jacobi Method

Leclerc derives a method for solving the system (7.2). Linearizing the gradient of F, (7.2) takes the form

$$\nabla F(\boldsymbol{u})=\boldsymbol{A}(\boldsymbol{u})\,\boldsymbol{u}+\boldsymbol{f}=0,$$

where $\boldsymbol{A}(\boldsymbol{u})$ is sparse. One can take advantage of the sparsity of $\boldsymbol{A}(\boldsymbol{u})$ and solve the equations for each cell's coefficient vector $\boldsymbol{u}_i$ individually. That is, with

$$\nabla_{\boldsymbol{u}_i} F(\boldsymbol{u}) = \boldsymbol{A}_{i,i}(\boldsymbol{u})\,\boldsymbol{u}_i + \left(\sum_{j \in N_i} \boldsymbol{A}_{i,j}(\boldsymbol{u})\,\boldsymbol{u}_j - \boldsymbol{f}_i\right) = 0,$$

where $\boldsymbol{A}_{i,j}(\boldsymbol{u})$ are 6×6 matrices, the method consists of solving

$$\boldsymbol{A}_{i,i}(\boldsymbol{u}^k)\,\boldsymbol{u}_i^{k+1} = \boldsymbol{f}_i - \sum_{j \in N_i} \boldsymbol{A}_{i,j}(\boldsymbol{u}^k)\,\boldsymbol{u}_j^k \tag{7.5}$$

for $\boldsymbol{u}_i^{k+1}$ for all i.

This is a variant of the Jacobi method normally used for solving linear systems [82]. When $\boldsymbol{A}$ is diagonally dominant this scheme (using (7.5) in a fixed-point iteration) is known to converge. While there is no proof for the diagonal dominance of $\boldsymbol{A}$, empirically this scheme converges with $F = F_0$ (see also [50], appendix C).

Leclerc combines this method with a termination criterion based on step size. That is, the iteration is stopped when

$$\|\boldsymbol{u}^{k+1} - \boldsymbol{u}^k\|_\infty < \mathrm{utol}\, s\, \sigma, \tag{7.6}$$

where Leclerc recommends the value 0.1 for parameter utol [48].

7.2.3 The Algorithms Compared

The runtime performance of two algorithms for a $p_{\max} = 2$ piecewise polynomial reconstruction is compared. The first algorithm, A, is Leclerc's algorithm [48] with the following modifications:

1. the embedding based on (5.7) and (5.8) is used (cell size $h = 1$),
2. the termination criterion (7.4) is used.

The embedding is modified as it affects the continuation paths (cf. Chapter 6), therefore both algorithms ought to use the same embedding. The termination criterion is modified for the same reason.

The second algorithm, B, uses the same objective function and representation of u as algorithm A but differs with respect to the optimization. Algorithm B utilizes the L-BFGS method for the minimization problem (7.1), for instance. Other features unique to algorithm B are described in Section 7.3.

The comparison is in terms of total number of function evaluations. To be precise, for the first algorithm the Jacobi iterations (7.5) are counted, for the second algorithm gradient evaluations are counted. This turns out to be a fair comparison. With the implementations considered here the runtime of one Jacobi iteration is about 1.75 times the runtime of one gradient and one function evaluation together. When including the time for the L-BFGS logic—averaged over number of gradient evaluations—the factor drops to about one.

7.2.4 Runtime Comparison

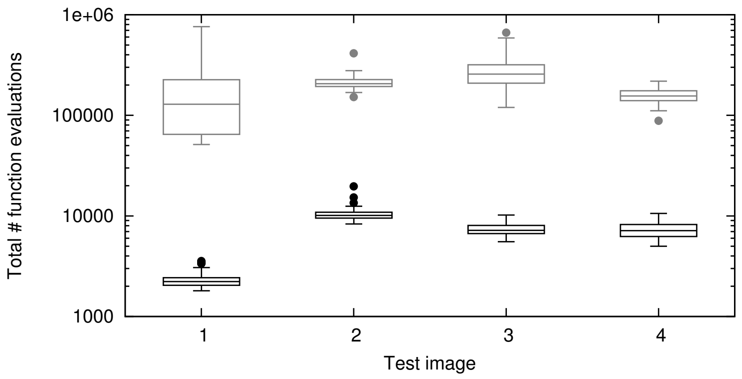


Figure 7.1. Box plots indicating the distribution of runtimes for algorithm A (gray) and B (black), respectively, with objective function parameters $b = 0.3$ and $d = b/10$. From left to right the box plots correspond to test cases 1, 2, 3 and 4. Each box stretches from the first to the third quartile of the group of measurements, respectively. The second quartile—also known as median—is shown as a horizontal line inside each box (see Table 7.1 for the median values). The whiskers above and below extend out up to 1.5 times the box's height. Measurements outside this range are considered outliers and are plotted as individual dots.

For the comparison 50 instances of each of the four test cases from Chapter 4 are generated (that is, 50 variants obtained with 50 different realizations of the noise component) and both algorithms are run on all 4×50 problems. With parameters $s_0 = 10$, $s_f = 0.01$, and $r = 0.96$ for the continuation method the outer loop generates 170 iterations.

Figure 7.1 shows nonparametric statistics of the total function evaluation counts after grouping by test case. Notice that expected runtime does not only depend on problem size.

	Test case 1	Test case 2	Test case 3	Test case 4
Algorithm A	128848	206643	256783	156457
Algorithm B	2237	10125	7193	7160
ratio	57.6	20.4	35.7	21.9

Table 7.1. Median runtimes (in # function evaluation) for algorithms A and B, respectively.

As Figure 7.1 makes clear algorithm B greatly outperforms algorithm A. This improvement is one of the major contributions of this dissertation. The speedup is 20-fold to 57-fold for the cases considered (cf. Table 7.1). A closer investigation in Section 7.3 shows that this speedup is not simply owed to the use of the L-BFGS method but to a combination of factors.

7.2.5 Comments

- The Jacobi iteration (7.5) was found not to converge with $F = F_1$. Even algorithm A ($F = F_0$) occasionally fails to converge (either algorithm is stopped when the termination criterion is not satisfied within 10^6 iterations).

- When termination criterion (7.6) with utol $= 0.1$ is used, algorithm A gives poor results for the test cases used in this study. Results improve and runtime increases when parameter utol is lowered, but so does the algorithm's tendency of failing to terminate. Overall, criterion (7.4) appears to be more appropriate for this problem than criterion (7.6).

7.3 Runtime Improvements in Detail

7.3.1 Jacobi Method vs L-BFGS

	Test case 1	Test case 2	Test case 3	Test case 4
Jacobi method, $h = 1$ (alg. A)	128848	206643	256783	156457
L-BFGS method, $h = 1$	814078	375421	368274	598173
ratio	0.158	0.550	0.697	0.262
Jacobi method, $h = 1/R$	44276	176381	47635	78343
L-BFGS method, $h = 1/R$	3060	18667	8915	9725
ratio	14.5	9.45	5.34	8.06

Table 7.2. Median runtime numbers with Jacobi method and L-BFGS method, respectively.

Changing the optimization method from Jacobi (Section 7.1.2) to L-BFGS alone does not improve runtime. To the contrary, runtime *degrades* if algorithm A is modified in this way (cf. Table 7.2). The picture reverses, however, if parameter h is chosen as described in Section 5.3.2 ($h = 1/R$).

7.3.2 Extrapolating the Continuation Path

Another way to reduce runtime does not concern the method used for solving problem (7.1) but rather the initial guesses from which a method starts its search. The prescription in Section 2.5.1 is to start the search in iteration k of the continuation method from the minimizer $\boldsymbol{u}^{*,k-1}$ obtained in the previous iteration.

However, a better starting point $\boldsymbol{v}^k$ can be constructed from the *two* previous iterations by linearly extrapolating the continuation path,

$$\boldsymbol{v}^k = \boldsymbol{u}^{*,k-1} + \omega \, \frac{s_k - s_{k-1}}{s_{k-1} - s_{k-2}} \, (\boldsymbol{u}^{*,k-1} - \boldsymbol{u}^{*,k-2}), \tag{7.7}$$

where a good value for factor ω can be found by experiment ($\omega = 0.7$ is used in this work).

Comparing runtimes with and without extrapolation shows that a speedup factor of 1.3 or better may be expected from this technique (cf. Table 7.3).

	Test case 1	Test case 2	Test case 3	Test case 4
Without extrapolation	3030	18254	9100	9360
With extrapolation (algorithm B)	2237	10125	7193	7160
ratio	1.35	1.80	1.27	1.31

Table 7.3. Median runtime numbers of variations of algorithm B without and with extrapolation of the continuation path, respectively.

7.4 Discussion

The study discussed in this chapter brings about a few insights and recommendations. The Jacobi method of Section 7.2.2 is unsafe; it may fail to converge. Generalizations such as the weighted Jacobi method or related relaxation methods such as the method of successive over-relaxation (SOR) [82] could be used to overcome this. However, alternatives to relaxation methods like the conjugate gradient method or limited-memory quasi-Newton methods are nowadays recommended for large-scale problems [63], and the prospect of developing a superior relaxation method seems remote.

Aside from the added safety of a globally convergent line-search method, the quasi-Newton method chosen in this work can greatly outperform the Jacobi method developed by Leclerc. Introducing the cell size parameter h is a key requisite to its success, however.

Setting h as described in Section 5.3.2 benefits the optimization in two ways. First, proper scaling of the independent variables ensures that the termination criterion (7.4) is appropriate (this is discussed in Section 4.4.2). Second, the *conditioning* of the problem improves, that is, the spread of eigenvalues of the Hessian matrix $\nabla^2 F_0(\boldsymbol{u})$ decreases. It is this second effect that brings about the dramatic improvement of the L-BFGS method seen in Table 7.2.

Chapter 8

Summary and Conclusions

I studied a particular approach to the problem of decomposing an image into two signal components, which are thought to represent different influences on the image formation process. The first, deterministic, component is modeled as a piecewise smooth function and represents the idealized projection of surfaces in the scene onto an image plane. The second, stochastic, component models noise in the image acquisition system as well as small-scale texturing of objects. The image features made explicit by the first component—discontinuities and differential structure—are of interest in computer vision applications.

This decomposition problem is ill-posed, and a simplicity argument is used to make it well-posed. The argument is formalized in an MDL criterion. The MDL criterion selects the decomposition which gives the shortest description of the image in terms of its two component signals. In its final form the problem is posed as a minimization problem.

This formulation of image reconstruction as a minimization problem, as well as an algorithm to solve it, was developed by Leclerc [48]. I studied Leclerc's work, reimplemented and evaluated his method, and developed solutions for problems I encountered. My contributions are quite technical in nature and I summarize them below. But first a few words to put Leclerc's work into broader perspective.

8.1 Leclerc's Work in Computer Vision

The piecewise smoothness assumption for visual data is a premise common to a number of formulations of the image reconstruction and the image segmentation problem [13, 30, 46, 48, 58]. These formulations are the basis of algorithms used in a variety of applications such as segmentation of medical images [17, 77, 61], analysis of remote-sensing data [28, 29, 64], image database retrieval [25, 37], interpretation of range data (robotics) [13, 32, 76], and estimation of optical flow (video compression, robotics) [7, 80].

The formulation generating the most follow-up research turned out to be the formulation of Mumford and Shah (Section 3.2.1). The work of Leclerc, on the other hand, has inspired much less follow-up work and is rarely cited in recent related publications. This might seem curious, considering that Leclerc's MDL formulation appears less ad hoc than Mumford and Shah's. The simplicity principle driving his approach has also been suggested as a basis for understanding human visual perception [35]. More importantly, perhaps, it provides a framework in which more sophisticated image analysis problems may be formulated [48].

I suggest two reasons why Leclerc's work may not have had a wider impact. The first reason is that workers in computer vision cannot easily evaluate Leclerc's reconstruction method because no implementation of his algorithm is readily available. The algorithm is fairly complex and reimplementing it requires a major effort.

The second reason is more mundane; I think Leclerc's work has been mostly forgotten. In a white paper, the initiators of the 2011 "Frontiers in Computer Vision" workshop are taking stock of recent research trends and identify areas for development in computer vision. On the state of the discipline's scholarship the authors comment:

> The current lack of scholarship not only results in the frequent reinvention of classic research but, perhaps more seriously, in good papers and grants being rejected ... There is a culture which evaluates researchers based on the number of papers they produce without taking their quality into account. In addition, older work, beyond a ten year time-span, seems often forgotten and is frequently being re-invented. ([81], page 6)

8.2 Contributions

8.2.1 Performance Characterization of Leclerc's Algorithm

In a recent article [3] published by the group at UC Berkeley who developed the Berkeley contour detection benchmark (Section 6.5), the authors present a chart juxtaposing benchmark results for a range of contour detection algorithms that have been published over the years (cf. Table 8.1). This chart shows how the field has gradually progressed over decades. But the selection of algorithms suffers from "availability bias" and one notices an awkward gap of almost 20 years—from the seminal work of Canny [15] to the work of Martin et al. [56]. Other interesting algorithms have been published during this time, of course, but for lack of readily available implementations, those other works are not mentioned.

F-measure	Algorithm	Year
0.70	gPb - Arbelaez, Maire, Fowlkes, Malik [3]	2011
0.68	Multiscale - Ren	2008
0.66	BEL - Dollar, Tu, Belongie	2006
0.66	Mairal, Leordeanu, Bach, Herbert, Ponce	2008
0.65	Min Cover - Felzenswalb, McAllester	2006
0.65	Pb - Martin, Fowlkes, Malik [56]	2004
0.64	Untangling Cycles - Zhu, Song, Shi	2007
0.64	CRF -Ren, Fowlkes, Malik	2005
0.58	Canny [15]	1986
0.56	Perona, Malik	1990
0.50	Hildreth, Marr	1980
0.48	Prewitt	1970
0.48	Sobel	1968
0.47	Roberts	1965

Table 8.1. Algorithms ranked by F-measure achieved on the Berkeley boundary detection benchmark (taken from Figure 1 in [3]).

One of my contributions consists of adding another data point to this ranking. If we attribute the new entry with F-measure value 0.62 (Section 6.5.3) completely to Leclerc, then this result asserts a significant improvement in contour detection performance over the Canny detector just three years after Canny's publication[8.1].

I also studied the behavior of Leclerc's algorithm on synthetic images and discovered a number of previously undocumented weaknesses, such as parameter sensitivity, solver bias, and a generally higher CNR threshold than was previously asserted.

8.2.2 Implementation

My source code is available to the research community so others may build on this work. The code represents a family of algorithms ranging from Leclerc's original algorithm to my derivative algorithm, which uses a newly derived objective function and includes a number of technical improvements boosting robustness and runtime. In the next section I briefly recount what these improvements are.

8.2.3 Robustness Improvements

A quantitative evaluation of the algorithm's performance on synthetic images revealed strong parameter sensitivity—a result at odds with Leclerc's claims. An investigation into the causes showed that the continuation method used for solving the optimization problem may fail to find good solutions, and that the choice of parameters sometimes determines success or failure. The continuation method's failure is an instance of solver bias, however, and should not force changes to an objective function. I presented two solutions.

8.2.3.1 More Appropriate Embedding Functions

First, I mitigated the problem with a new and more appropriate embedding which makes the continuation method work more reliably. Second, I showed that the parameter sensitivity is an expression of solver bias, and that solver bias may be controlled in the solver itself by use of more general forms of embeddings.

8.1. This would make Leclerc's entry the best performing method among those that are older than the Berkeley benchmark. It should also be noted that all the algorithms listed above Canny's are using color information, whereas Canny's and all entries below it do not. The F-measure value achieved by a state-of-the-art method using no color information is 0.68 (cf. Section 6.5).

8.2.3.2 An MDL Criterion with Fewer Parameters

MDL criteria are inherently parameter-free. In Leclerc's solution parameters are introduced when approximations are being made. To approach the parameter sensitivity problem from a different angle, I reconsidered the MDL criterion. The approximations in question are necessitated by the constraint that evaluation of the encoding length may only use information from pairs of connected cells. A run-length encoding caters better to this constraint and allows a tighter approximation than a chain-code encoding. This leads to the derivation of a new MDL criterion based on a run-length encoding. The new criterion has one free parameter instead of two, and experiments suggest the resulting algorithm is more robust with respect to parameter choice.

8.2.3.3 Ensuring Convergence using a Line-Search

The method for solving the optimization problem proposed by Leclerc is another source of difficulties. The termination criterion has proven inappropriate for achieving high-quality results, and the Jacobi relaxation method may fail to converge. Using a termination criterion based on gradient magnitude and employing a line-search method solves the robustness problems.

In recent PhD work Ivanovska extends Leclerc's piecewise constant algorithm so it may work directly on color images (RGB) [41, 40]. She also finds that the relaxation method produces poor results on synthetic test images and reports that minimizing by steepest descent instead "gives high quality results and reconstructs the initial images" ([40], page 80). (It is not clear from her description whether she used the same termination criterion when comparing both methods or not.)

8.2.4 Runtime Improvements

The steepest descent method chosen by Ivanovska performs poorly in terms of runtime, however. She reports that "the computational costs are high, for example, for a 100×100 color image the costs are in the range of several hours" ([40], page 80).

My algorithm, using a quasi-Newton method and including the runtime improvements reported in Section 7.3, typically converges in less than a minute for images of this size. The main runtime improvement is achieved through a rescaling of the independent variables, which accelerates convergence of the quasi-Newton method by one to two orders of magnitude.

8.3 Future Research

Future work may be concerned with further runtime and robustness improvements or with more extensive reconstruction problems.

8.3.1 Improving Runtime

Preliminary results that I did not report in this work indicate that "diagonal preconditioning" improves the runtime of the L-BFGS method for this problem.

Use of higher-level termination criteria and of approximate solutions may be promising opportunities for improving runtime. The number of steps the continuation method needs to take until all regions are identified depends on the image. A higher-level termination criterion could terminate the continuation method when no more progress in terms of discontinuity detection is to be expected.

If parts of the reconstruction could be approximated quickly, for instance by standard smoothing techniques, this information might be used to derive cell-specific bounds on the values of coefficients or other kinds of constraints on the solution. It might be possible to incorporate such constraints into the embedding in order to accelerate convergence.

Finally, one might try to apply entirely different optimization approaches such as genetic algorithms, dynamic programming or search heuristics.

8.3.2 Extending the Reconstruction Problem

As already mentioned above, the reconstruction problem itself may be extended in several ways. Perhaps most beneficial would be the incorporation of texture models to generalize the piecewise smoothness constraint to "piecewise smoothness or continuity in texture". Generalization to color images is another compelling next step.

There are different causes for discontinuities in intensity (occlusion, shadows, etc.). One might try to make more information related to those causes explicit, for example in form of "intrinsic images" [5]. Intrinsic images contain information about scene illumination, surface geometry, and albedo. A simple rendering equation expresses how these quantities interact to cause the intensity recorded by an imaging sensor. For a suitable class of images only might pose the problem of inverting the rendering equation and to reconstruct a scene's intrinsic images [42]. Piecewise smoothness constraints imposed on the intrinsic images and simplicity arguments would be used in order to make the problem well-posed.

8.4 Conclusion

Computer vision—enabling machines to "see"—is an incredibly challenging inference problem. Some hold that Bayesian statistics provides a framework for developing solutions in a rigorous way. Leclerc argued that MDL provides another suitable framework. He considered reconstruction of image intensity as a first step on a long, uncharted path towards solving the computer vision problem. I hope that this work will rekindle interest in the MDL approach, and that it will help others in taking the next step.

Bibliography

[1] Giovanni Alberti, Guy Bouchitté and Gianni Dal Maso. The calibration method for the mumford-shah functional and free-discontinuity problems. *Calculus of Variations and Partial Differential Equations*, 16(3):299–333, 2003.

[2] Luigi Ambrosio and Vincenzo Maria Tortorelli. Approximation of functional depending on jumps by elliptic functional via t-convergence. *Communications on Pure and Applied Mathematics*, 43(8):999–1036, 1990.

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes and Jitendra Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):898–916, 2011.

[4] H. Asada and M. Brady. The curvature primal sketch. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(1):2–14, jan 1986.

[5] H. G. Barrow and J. M. Tenenbaum. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, pages 3–26. Academic Press, 1978.

[6] S. M. Bileschi. *StreetScenes: Towards Scene Understanding in Still Images*. PhD thesis, 2006.

[7] Michael J Black and Paul Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer vision and image understanding*, 63(1):75–104, 1996.

[8] Andrew Blake. Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(1):2–12, 1989.

[9] Andrew Blake and A Zisserman. Using weak continuity constraints. *University of Edinburgh Department of Computer Science Report*, CSR-186-85, 1985.

[10] Andrew Blake, Andrew Zisserman and AV Papoulias. Weak continuity constraints generate uniform scale-space descriptions of plane curves. *Proc. EC AI, Brighton, England*, , 1986.

[11] Yuri Boykov, Olga Veksler and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[12] Thomas Brox and Daniel Cremers. On local region models and a statistical interpretation of the piecewise smooth mumford-shah functional. *International journal of computer vision*, 84(2):184–193, 2009.

[13] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, MA, 1987.

[14] Andrew Blake, Pushmeet Kohli and Carsten Rother. *Markov random fields for vision and image processing*. MIT Press, 2011.

[15] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, nov 1986.

[16] Antonin Chambolle and Pierre-Louis Lions. Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76(2):167–188, 1997.

[17] Tony F Chan and Luminita A Vese. Active contours without edges. *Image Processing, IEEE Transactions on*, 10(2):266–277, 2001.

[18] S Grace Chang, Bin Yu and Martin Vetterli. Adaptive wavelet thresholding for image denoising and compression. *Image Processing, IEEE Transactions on*, 9(9):1532–1546, 2000.

[19] Daniel Cremers, Florian Tischhäuser, Joachim Weickert and Christoph Schnörr. Diffusion snakes: introducing statistical shape knowledge into the mumford-shah functional. *International journal of computer vision*, 50(3):295–313, 2002.

[20] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley series in telecommunications. John Wiley & Sons, New York, USA, 1991.

[21] G Dahlquist and A Björck. Numerical methods. , :573, 1974.

[22] B. Dom. The minimum description length principle and ill-posed problems in computer vision. In *Proceedings of the International Workshop on Computer Vision and Applied Geometry*. Nordfjordeid, Norway, August 1995. (Also available as IBM Research Report RJ 10116 (91932) 4/13/98).

[23] David L Donoho and Jain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[24] P. F. Felzenszwalb. Representation and detection of deformable shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(2):208–220, feb 2005.

[25] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic et al. Query by image and video content: the qbic system. *Computer*, 28(9):23–32, 1995.

[26] D. Forsyth and A. Zisserman. Reflections on shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(7):671–679, jul 1991.

[27] H. Freeman. Computer processing of line-drawing images. *Computing Surveys*, 6:57–97, mar 1974.

[28] F. Galland, N. Bertaux and P. Refregier. Minimum description length synthetic aperture radar image segmentation. *IEEE Trans. Image Processing*, 12(9):995–1006, sep 2003.

[29] Frédéric Galland, Nicolas Bertaux and Philippe Réfrégier. Multi-component image segmentation in homogeneous regions based on description length minimization: application to speckle, poisson and bernoulli noise. *Pattern recognition*, 38(11):1926–1936, 2005.

[30] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, nov 1984.

[31] Leo Grady and Christopher V Alvino. The piecewise smooth mumford–shah functional on an arbitrary graph. *Image Processing, IEEE Transactions on*, 18(11):2547–2561, 2009.

[32] William Eric Leifur Grimson and WEL Grimson. *From images to surfaces: A computational study of the human early visual system*, volume 4. MIT press Cambridge, MA, 1981.

[33] Mark H Hansen and Bin Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

[34] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.

[35] Gary Hatfield and William Epstein. The status of the minimum principle in the theoretical analysis of visual perception. *Psychological Bulletin*, 97(2):155, 1985.

[36] B. K. P. Horn. Understanding image intensities. *Artificial Intelligence*, 8(2):201–231, apr 1977.

[37] Qian Huang, Byron Dom, N Megiddo and W Niblack. Segmenting and representing background in color images. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 13–17. IEEE, 1996.

[38] M. Hueckel. A local visual operator which recognizes edges and lines. *Journal of the ACM*, 20(4):634–647, oct 1973.

[39] B. K. P. Horn. *Robot Vision*. MIT Press, Cambridge, MA, 1986.

[40] Tetyana Ivanovska. *Efficient multichannel image partitioning: theory and application*. PhD thesis, Jacobs University Bremen, 2009.

[41] Tetyana Ivanovska, Horst K Hahn and Lars Linsen. On global mdl-based multichannel image restoration and partitioning. In *20th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG).* 2012.

[42] R. Juengling. Meaningful contours from intrinsic images. Computer Science, Portland State University, June 2008.

[43] R. Juengling and L. Prasad. Parsing silhouettes without boundary curvature. In Rita Cucchiara, editor, *ICIAP*, pages 665–670. IEEE Computer Society, 2007.

[44] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[45] J. E. Dennis Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

[46] T. Kanungo, B. Dom, W. Niblack and D. Steele. A fast algorithm for MDL-Based multi-band image segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 609–616. Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[47] Georges Koepfler, Christian Lopez and Jean-Michel Morel. A multiscale algorithm for image segmentation by variational method. *SIAM journal on numerical analysis*, 31(1):282–299, 1994.

[48] Y. G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3:73–102, may 1989.

[49] Y. G. Leclerc. Simplicity of description as the basis for visual interpretation. In *AAAI Symposium on The Theory and Application of Minimal Length Encoding*, pages 100–104. Mar 1990.

[50] Yvan G Leclerc. *The local structure of image intensity discontinuities*. PhD thesis, 1989.

[51] Yvan G Leclerc and Steven W Zucker. The local structure of image discontinuities in one dimension. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (3):341–355, 1987.

[52] T. C. M. Lee. A minimum description length–based image segmentation procedure, and its comparison with a cross-validation–based segmentation procedure. *Journal of the American Statistical Association*, 95(449):259–270, 2000.

[53] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[54] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(7):674–693, 1989.

[55] D. Martin, C. Fowlkes, D. Tal and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423. July 2001.

[56] D. R. Martin, C. C. Fowlkes and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(5):530–549, may 2004.

[57] Jorge J Moré and David J Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):286–307, 1994.

[58] D. Mumford and J. Shah. Boundary detection by minimizing functionals. In *IEEE Computer Vision and Pattern Recognition or CVPR*, pages 22–26. 1985.

[59] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, 42(5):577–685, 1989.

[60] D. Marr. *Vision*. W. H. Freeman & Co., San Francisco, 1982.

[61] Amar Mitiche and Ismail Ben Ayed. *Variational and level set methods in image segmentation*, volume 5. Springer, 2010.

[62] Jean Michel Morel and Sergio Solimini. *Variational methods in image segmentation*. Birkhauser Boston Inc., 1995.

[63] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[64] H. P. Pan. Two-level global optimization for image segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 49(2):21–32, 1994.

[65] Nadia Payet and Sinisa Todorovic. From contours to 3d object detection and pose estimation. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 983–990. IEEE, 2011.

[66] Thomas Pock, Antonin Chambolle, Daniel Cremers and Horst Bischof. A convex relaxation approach for computing minimal partitions. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 810–817. IEEE, 2009.

[67] Thomas Pock, Daniel Cremers, Horst Bischof and Antonin Chambolle. An algorithm for minimizing the mumford-shah functional. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1133–1140. IEEE, 2009.

[68] T. Poggio, V. Torre and C. Koch. Computational vision and regularization theory. *Nature*, 317:314–319, 1985.

[69] Jan Puzicha, Joachim M Buhmann, Yossi Rubner and Carlo Tomasi. Empirical evaluation of dissimilarity measures for color and texture. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1165–1172. IEEE, 1999.

[70] Todd R Reed and JM Hans Dubuf. A review of recent texture segmentation and feature extraction techniques. *CVGIP: Image understanding*, 57(3):359–372, 1993.

[71] J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, IT-30(4):629–636, 1984.

[72] Leonid I Rudin, Stanley Osher and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[73] A. Shashua and S. Ullman. Structural saliency: the detection of globally salient structures using a locally connected network. In *International Conference on Computer Vision*, pages 321–327. 1988.

[74] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[75] Claude Elwood Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, 1949.

[76] D. Terzopoulos. The computation of visible-surface representations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(4):417–438, jul 1988.

[77] Andy Tsai, Anthony Yezzi Jr and Alan S Willsky. Curve evolution implementation of the mumford-shah functional for image segmentation, denoising, interpolation, and magnification. *Image Processing, IEEE Transactions on*, 10(8):1169–1186, 2001.

[78] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed Problems*. Winston and Sons, Washington, DC, 1977.

[79] David H Wolpert and William G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

[80] Ming Ye, Robert M Haralick and Linda G Shapiro. Estimating piecewise-smooth optical flow with global matching and graduated optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(12):1625–1630, 2003.

[81] A. Yuille and A. Oliva. Frontiers in computer vision: nfs white paper. Technical Report, NSF, November 2010.

[82] D. Young. *Iterative Solution of Large Linear Systems*. Academic Press, New York, 1971.

[83] S. C. Zhu and A. Yuille. Region competition: unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(9):884–900, sep 1996.

Appendix A

Function and Gradient Expressions

A.1 Expressions for Objective Function $\boldsymbol{F_0}$

With an embedding as described in Section 2.5.1 F_0 turns into a function with two arguments,

$$F_0(\boldsymbol{u},s) = \sum_{i\in I}\Bigg\{\frac{1}{2\log 2}\Big(\frac{z_i-u_{i,0}}{\sigma}\Big)^2 + d\left[1+2\left(1-\prod_{k=1}^{5} e(u_{i,k},s)\right)+3\left(1-\prod_{k=3}^{5} e(u_{i,k},s)\right)\right] + \frac{b}{2}\sum_{j\in N_i}\left[1-e(\Delta_{i,j,0},s)+2\left(1-\prod_{k=0}^{2} e(\Delta_{i,j,k},s)\right)+3\left(1-\prod_{k=0}^{5} e(\Delta_{i,j,k},s)\right)\right]\Bigg\}$$

It is helpful to introduce names for the new product terms to,

$$C_1(\boldsymbol{u}_i) = \prod_{k=1}^{5} e(u_{i,k},s), \tag{A.1}$$

$$C_2(\boldsymbol{u}_i) = \prod_{k=3}^{5} e(u_{i,k},s), \tag{A.2}$$

$$D_0(\boldsymbol{u}_i,\boldsymbol{u}_j) = e(\Delta_{i,j,k},s), \tag{A.3}$$

$$D_1(\boldsymbol{u}_i,\boldsymbol{u}_j) = \prod_{k=0}^{2} e(\Delta_{i,j,k},s), \tag{A.4}$$

$$D_2(\boldsymbol{u}_i,\boldsymbol{u}_j) = \prod_{k=0}^{5} e(\Delta_{i,j,k},s). \tag{A.5}$$

With these definitions $F_0(\boldsymbol{u},s)$ reads

$$F_0(\boldsymbol{u},s) = \sum_{i\in I}\Bigg\{\frac{1}{2\log 2}\Big(\frac{z_i-u_{i,0}}{\sigma}\Big)^2 + d\,(1+2\,[1-C_1(\boldsymbol{u}_i)]+3\,[1-C_2(\boldsymbol{u}_i)]) + \frac{b}{2}\sum_{j\in N_i}(1-D_0(\boldsymbol{u}_i,\boldsymbol{u}_j)+2\,[1-D_1(\boldsymbol{u}_i,\boldsymbol{u}_j)]+3\,[1-D_2(\boldsymbol{u}_i,\boldsymbol{u}_j)])\Bigg\} \tag{A.6}$$

The gradient of (A.6) with respect to $\boldsymbol{u}_i$ is

$$\begin{aligned}\nabla_{\boldsymbol{u}_i} F_0(\boldsymbol{u}, s) = \frac{1}{\log 2} \frac{u_{i,0} - z_i}{\sigma^2} \\ -d\,(2\,\nabla_{\boldsymbol{u}_i} C_1(\boldsymbol{u}_i) + 3\,\nabla_{\boldsymbol{u}_i} C_2(\boldsymbol{u}_i)) \\ -b \sum_{j\in N_i} \{\nabla_{\boldsymbol{u}_i} D_0(\boldsymbol{u}_i, \boldsymbol{u}_j) + 2\,\nabla_{\boldsymbol{u}_i} D_1(\boldsymbol{u}_i, \boldsymbol{u}_j) + 3\,\nabla_{\boldsymbol{u}_i} D_2(\boldsymbol{u}_i, \boldsymbol{u}_j)\} \end{aligned} \tag{A.7}$$

where we made use of the fact that $\nabla_{\boldsymbol{u}_i} D_l(\boldsymbol{u}_i, \boldsymbol{u}_j) = \nabla_{\boldsymbol{u}_i} D_l(\boldsymbol{u}_j, \boldsymbol{u}_i)$. The gradients of the product terms (A.1) to (A.5) read

$$\nabla_{\boldsymbol{u}_i} C_1(\boldsymbol{u}_i) = -2\,C_1(\boldsymbol{u}_i) \begin{pmatrix} 0 & \frac{u_{i,1}}{s^2\sigma^2} & \frac{u_{i,2}}{s^2\sigma^2} & \frac{u_{i,3}}{s^2\sigma^2} & \frac{u_{i,4}}{s^2\sigma^2} & \frac{u_{i,5}}{s^2\sigma^2} \end{pmatrix}^T, \tag{A.8}$$

$$\nabla_{\boldsymbol{u}_i} C_2(\boldsymbol{u}_i) = -2\,C_2(\boldsymbol{u}_i) \begin{pmatrix} 0 & 0 & 0 & \frac{u_{i,3}}{s^2\sigma^2} & \frac{u_{i,4}}{s^2\sigma^2} & \frac{u_{i,5}}{s^2\sigma^2} \end{pmatrix}^T, \tag{A.9}$$

$$\nabla_{\boldsymbol{u}_i} D_0(\boldsymbol{u}_i, \boldsymbol{u}_j) = -D_0(\boldsymbol{u}_i, \boldsymbol{u}_j)\,\frac{1}{s^2\sigma^2}\,\nabla_{\boldsymbol{u}_i} \Delta^2_{i,j,0}(\boldsymbol{u})\,, \tag{A.10}$$

$$\nabla_{\boldsymbol{u}_i} D_1(\boldsymbol{u}_i, \boldsymbol{u}_j) = -D_1(\boldsymbol{u}_i, \boldsymbol{u}_j) \times \left(\sum_{k=0}^{2} \frac{1}{s^2\sigma^2} \nabla_{\boldsymbol{u}_i} \Delta^2_{i,j,k}(\boldsymbol{u}) \right), \text{ and} \tag{A.11}$$

$$\nabla_{\boldsymbol{u}_i} D_2(\boldsymbol{u}_i, \boldsymbol{u}_j) = -D_2(\boldsymbol{u}_i, \boldsymbol{u}_j) \times \left(\sum_{k=0}^{5} \frac{1}{s^2\sigma^2} \nabla_{\boldsymbol{u}_i} \Delta^2_{i,j,k}(\boldsymbol{u}) \right). \tag{A.12}$$

The gradient expressions (A.8) to (A.12) assume the embedding (2.18) and (2.19); with the embedding in Section 5.3.4 similar expressions result. The gradients $\nabla_{\boldsymbol{u}_i}\Delta^2_{i,j,k}(\boldsymbol{u})$ finally are of the form

$$\nabla_{\boldsymbol{u}_i}\Delta^2_{i,j,k}(\boldsymbol{u}) = 2\,E_{NN,k}\,\boldsymbol{u}_i - 2\,E_{NS,k}\,\boldsymbol{u}_j,$$

(cf. expression (B.1)) with matrices $E_{NN,k}$ etc. as given in Section B.

A.2 Expressions for Objective Function F_1

Using relation (2.17) and $\min_k \delta(x_k) = \prod_k \delta(x_k)$ we can bring F_1 in (5.3) into a form similar to (2.16), then apply the embedding as in Section A.1. The embedded function F_1 reads

$$\begin{aligned} F_1(\boldsymbol{u}, s) = \sum_{i\in I} \Bigg\{ \frac{1}{2\log 2}\Big(\frac{z_i - u_{i,0}}{\sigma}\Big)^2 + g\left(1 - \frac{1}{|N_i|}\sum_{j\in N_i} e(\Delta_{i,j,0}, s)\right) \\ +2\,g\left(1 - \prod_{k=1}^{5} e(u_{i,k}, s)\right) \times \left(1 - \frac{1}{|N_i|}\sum_{j\in N_i}\prod_{k=0}^{2} e(\Delta_{i,j,k}, s)\right) \end{aligned} \tag{A.13}$$

$$+3\,g\left(1-\prod_{k=3}^{5}e(u_{i,k},s)\right)\times\left(1-\frac{1}{|N_i|}\sum_{j\in N_i}\prod_{k=0}^{5}e(\Delta_{i,j,k},s)\right)\Bigg\},$$

or, using definitions (A.1) to (A.5),

$$\begin{aligned}F_1(\boldsymbol{u},s) \;=\; \sum_{i\in I}\Bigg\{&\frac{1}{2\log 2}\Big(\frac{z_i-u_{i,0}}{\sigma}\Big)^2+g\,(2\,[1-C_1(\boldsymbol{u}_i)]+3\,[1-C_2(\boldsymbol{u}_i)])\\ &-g\,\frac{1}{|N_i|}\sum_{j\in N_i}[D_0(\boldsymbol{u}_i,\boldsymbol{u}_j)+2\,[1-C_1(\boldsymbol{u}_i)]\times D_1(\boldsymbol{u}_i,\boldsymbol{u}_j)\\ &\qquad+3\,[1-C_2(\boldsymbol{u}_i)]\times D_2(\boldsymbol{u}_i,\boldsymbol{u}_j)]\Bigg\}\end{aligned} \tag{A.14}$$

After fixing $|N_i|=4$ for all i the gradient of (A.14) with respect to $\boldsymbol{u}_i$ reads

$$\begin{aligned}\nabla_{\boldsymbol{u}_i}F_1(\boldsymbol{u},s)=&\frac{1}{\log 2}\frac{u_{i,0}-z_i}{\sigma^2}-2\,g\left(1-\frac{1}{4}\sum_{j\in N_i}D_1(\boldsymbol{u}_i,\boldsymbol{u}_j)\right)\nabla_{\boldsymbol{u}_i}C_1(\boldsymbol{u}_i)\\ &-3\,g\left(1-\frac{1}{4}\sum_{j\in N_i}D_2(\boldsymbol{u}_i,\boldsymbol{u}_j)\right)\nabla_{\boldsymbol{u}_i}C_2(\boldsymbol{u}_i)\\ &-\frac{g}{4}\sum_{j\in N_i}\{2\,\nabla_{\boldsymbol{u}_i}D_0(\boldsymbol{u}_i,\boldsymbol{u}_j)+2\,(2-C_1(\boldsymbol{u}_i)-C_1(\boldsymbol{u}_j))\,\nabla_{\boldsymbol{u}_i}D_1(\boldsymbol{u}_i,\boldsymbol{u}_j)\\ &\qquad+3\,(2-C_2(\boldsymbol{u}_i)-C_2(\boldsymbol{u}_j))\,\nabla_{\boldsymbol{u}_i}D_2(\boldsymbol{u}_i,\boldsymbol{u}_j)\}\end{aligned}$$

with $\nabla_{\boldsymbol{u}_i}C_1(\boldsymbol{u}_i)$ etc. as given in Section A.1.

A.3 The Optimization Problem with Piecewise-Constant Variance

When σ is unknown there is an additional unknown σ_i for each pixel to infer, and we replace objective function F_0 in (2.15) by

$$\tilde{F}_0(\boldsymbol{u},\boldsymbol{\sigma})=\sum_{i\in I}\left\{\frac{1}{2\log 2}\left(\frac{z_i-u_{i,0}}{\sigma_i}\right)^2+\ldots\right\}, \tag{A.15}$$

where the remaining terms '....' are the same as in F_0. We assume piecewise constant variance where the boundaries are aligned with the intensity boundaries. This means the solution $\boldsymbol{\sigma}^*$ must satisfy the constraint

$$\sigma_i^*=\sigma_j^* \text{ when } \delta(\Delta_{i,j,0})=1. \tag{A.16}$$

Leclerc suggests the following approach to solve this problem [48]. Instead of trying to minimize (the embedded version of) $\tilde{F}_0$ with respect to both $\boldsymbol{u}$ and $\boldsymbol{\sigma}$ directly, in each iteration of the continuation method one uses an intermediate solution $\boldsymbol{\sigma}^{*,t}$ for $\boldsymbol{\sigma}$ and minimizes with respect to $\boldsymbol{u}$ only. The intermediate solution $\boldsymbol{\sigma}^{*,t}$ is obtained from solving another minimization problem, namely minimizing E in (A.18) below. The algorithm then consists of solving two optimization problems in each iteration of the continuation method, and the embedding consists of the substitutions

$$\begin{aligned}\delta(u_{i,k}) &\rightarrow \exp\left(\frac{-u_{i,k}^2}{f\,s\,\sigma_i^{*,t}}\right)\\ \delta(\Delta_{i,j,k}) &\rightarrow \exp\left(\frac{-\Delta_{i,j,k}^2}{s\left[\sigma_i^{*,t}+\sigma_j^{*,t}\right]/2}\right)\end{aligned}$$

instead of (2.18) and (2.19).

This approach requires that the objective function E “selects” the same solution $\boldsymbol{\sigma}^*$, given $\boldsymbol{u}^*$, as would be obtained by direct minimization of $\tilde{F}_0$ with respect to $\boldsymbol{u}$ and $\boldsymbol{\sigma}$, such that (A.16) is satisfied. It is easily shown that the values σ_r^* minimizing the description length with region-specific variances $\{\sigma_r\}$ equal the variance estimates [48], that is,

$$(\sigma_r^*)^2 = \frac{1}{|R_r|}\sum_{i\in R_r}(u_r^*-z_i)^2. \tag{A.17}$$

This variance estimate may also be written as an average of *local variance estimates* $\hat{\sigma_i}$,

$$(\sigma_r^*)^2 = \frac{1}{|R_r|}\sum_{i\in R_r}\hat{\sigma}_i^2 = \frac{1}{|R_r|}\sum_{i\in R_r}\frac{\sum_{i\in\bar{N}_j\cap R_r}\delta(\Delta_{i,j,0})\,(u_r^*-z_j)^2}{\sum_{j\in\bar{N}_i\cap R_r}1},$$

where $\bar{N}_i$ includes cell i, $\bar{N}_i = N_i\cup\{i\}$. With this idea we can write an objective function for $\boldsymbol{\sigma}$, where a solution gradually develops from local to global, per-region variance estimates as the continuation parameter decreases (recall that the regions R are not known up front and are only defined in the limit $s\rightarrow 0$),

$$\begin{aligned}E(\boldsymbol{\sigma},s) &= a\sum_{i\in I}\left(\frac{\sigma_i-\hat{\sigma}_i}{\sigma_i^*}\right)^2\\ &\quad+\frac{o}{2}\sum_{i\in I}\sum_{j\in N_i}e(\Delta_{i,j,0}(\boldsymbol{u}^*))\times\left[1-\exp\left(-\frac{(\sigma_i-\sigma_j)^2}{[s\,(\sigma_i^*+\sigma_j^*)/2]^2}\right)\right].\end{aligned} \tag{A.18}$$

Here o is a new parameter discussed momentarily, σ_i^* denotes the minimum for σ_i obtained in the previous step of the continuation method. The local estimate $\hat{\sigma}_i$ is defined as

$$\hat{\sigma}_i = \max\left\{1, \sqrt{\frac{\sum_{j\in\bar{N}_i} e(\Delta_{i,j,0}(\boldsymbol{u}^*))\,(z_j - u_i^*(\boldsymbol{x}_j))^2}{\sum_{j\in\bar{N}_i} e(\Delta_{i,j,0}(\boldsymbol{u}^*))}}\right\}, \tag{A.19}$$

to account for quantization effects, where $u_i^*(\boldsymbol{x}_j)$ denotes the polynomial with coefficients $\boldsymbol{u}_i^*$ evaluated at the point $\boldsymbol{x}_j$. Note that these estimates are not functions of the coefficients $\boldsymbol{u}$ that are to be determined, but are functions of the known coefficients $\boldsymbol{u}^*$, that are the result of the most recent minimization with respect to $\boldsymbol{u}$.

The first term in (A.18) prefers values for σ_i that are close to the local estimates $\hat{\sigma}_i$, and the second term forces σ to be a piecewise-constant function for $s \to 0$. The parameter o has no direct interpretation in terms of number of bits per per coefficient, but needs to be chosen empirically (e.g., by using training data for parameter tuning as in Section 6.5.1).

The gradient of E is

$$\frac{\partial E}{\partial \sigma_i} = 2\,a\,\frac{\sigma_i - \hat{\sigma}_i}{\sigma_i^{*2}} + 2\,o \sum_{j\in N_i} e^*_{i,j,0} \times \exp\left(-\frac{(\sigma_i - \sigma_j)^2}{[s\,(\sigma_i^* + \sigma_j^*)/2]^2}\right) \times \frac{(\sigma_i - \sigma_j)}{[s\,(\sigma_i^* + \sigma_j^*)/2]^2},$$

where $e^*_{i,j,0} = e(\Delta_{i,j,0}(\boldsymbol{u}^*))$, and use is made use of the fact that $e^*_{i,j,0} = e^*_{j,i,0}$.

Appendix B

Difference Terms in Detail

To evaluate integrals like (5.4) we first define some vector-valued functions $\boldsymbol{e}_{N,k}(x)$, $\boldsymbol{e}_{E,k}(y)$, $\boldsymbol{e}_{S,k}(x)$, $\boldsymbol{e}_{W,k}(y)$,

$$
\begin{aligned}
\boldsymbol{e}^T_{N,0}(x) &= \begin{pmatrix} 1 & x & -\frac{h}{2} & \frac{1}{2}x^2 & -\frac{h}{2}x & \frac{h^2}{8} \end{pmatrix} \\
\boldsymbol{e}^T_{E,0}(y) &= \begin{pmatrix} 1 & \frac{h}{2} & y & \frac{h^2}{8} & \frac{h}{2}y & \frac{1}{2}y^2 \end{pmatrix} \\
\boldsymbol{e}^T_{S,0}(x) &= \begin{pmatrix} 1 & x & \frac{h}{2} & \frac{1}{2}x^2 & \frac{h}{2}x & \frac{h^2}{8} \end{pmatrix} \\
\boldsymbol{e}^T_{W,0}(y) &= \begin{pmatrix} 1 & -\frac{h}{2} & y & \frac{h^2}{8} & -\frac{h}{2}y & \frac{1}{2}y^2 \end{pmatrix} \\
\boldsymbol{e}^T_{N,1}(x) &= \begin{pmatrix} 0 & 1 & 0 & x & -\frac{h}{2} & 0 \end{pmatrix} \\
\boldsymbol{e}^T_{E,1}(y) &= \begin{pmatrix} 0 & 1 & 0 & \frac{h}{2} & y & 0 \end{pmatrix} \\
\boldsymbol{e}^T_{S,1}(x) &= \begin{pmatrix} 0 & 1 & 0 & x & \frac{h}{2} & 0 \end{pmatrix} \\
\boldsymbol{e}^T_{W,1}(y) &= \begin{pmatrix} 0 & 1 & 0 & -\frac{h}{2} & y & 0 \end{pmatrix} \\
\boldsymbol{e}^T_{N,2}(x) &= \begin{pmatrix} 0 & 0 & 1 & 0 & x & -\frac{h}{2} \end{pmatrix} \\
\boldsymbol{e}^T_{E,2}(y) &= \begin{pmatrix} 0 & 0 & 1 & 0 & \frac{h}{2} & y \end{pmatrix} \\
\boldsymbol{e}^T_{S,2}(x) &= \begin{pmatrix} 0 & 0 & 1 & 0 & x & \frac{h}{2} \end{pmatrix} \\
\boldsymbol{e}^T_{W,2}(y) &= \begin{pmatrix} 0 & 0 & 1 & 0 & -\frac{h}{2} & y \end{pmatrix},
\end{aligned}
$$

and

$$
\begin{aligned}
\boldsymbol{e}^T_{N,3}(x) = \boldsymbol{e}^T_{E,3}(y) = \boldsymbol{e}^T_{S,3}(x) = \boldsymbol{e}^T_{W,3}(y) &= (\ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ ) \\
\boldsymbol{e}^T_{N,4}(x) = \boldsymbol{e}^T_{E,4}(y) = \boldsymbol{e}^T_{S,4}(x) = \boldsymbol{e}^T_{W,4}(y) &= (\ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ ) \\
\boldsymbol{e}^T_{N,5}(x) = \boldsymbol{e}^T_{E,5}(y) = \boldsymbol{e}^T_{S,5}(x) = \boldsymbol{e}^T_{W,5}(y) &= (\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ ).
\end{aligned}
$$

The expression $\boldsymbol{e}^T_{N,0}(x)\,\boldsymbol{u}_i$ evaluates $u|_{\Omega_i}$ on the northern cell boundary at point $(x, -h/2)$ in the coordinate system of cell i. Analogously, inner products with the other functions evaluate $u|_{\Omega_i}$ or a derivative of u on one of the boundaries of cell i. We now use these functions to derive expressions for integrals like (5.4). For instance, if cell j is to the north of i then

$$\begin{aligned}
\Delta^2_{i,j,k}(\boldsymbol{u}) &= \frac{1}{h}\int_{-h/2}^{h/2} (\boldsymbol{e}^T_{N,k}(x)\,\boldsymbol{u}_i - \boldsymbol{e}^T_{S,k}(x)\,\boldsymbol{u}_j)^2\,\mathrm{dx} \\
&= \frac{1}{h}\int_{-h/2}^{h/2} (\boldsymbol{e}^T_{N,k}(x)\,\boldsymbol{u}_i)^2 - 2\,(\boldsymbol{e}^T_{N,k}(x)\,\boldsymbol{u}_i)(\boldsymbol{e}^T_{S,k}(x)\,\boldsymbol{u}_j) + (\boldsymbol{e}^T_{S,k}(x)\,\boldsymbol{u}_j)^2\,\mathrm{dx} \\
&= \boldsymbol{u}_i^T\left[\frac{1}{h}\int_{-h/2}^{h/2} \boldsymbol{e}_{N,k}(x)\,\boldsymbol{e}^T_{N,k}(x)\,\mathrm{dx}\right]\boldsymbol{u}_i - \\
&\quad \boldsymbol{u}_i^T\left[\frac{1}{h}\int_{-h/2}^{h/2} \boldsymbol{e}_{N,k}(x)\,\boldsymbol{e}^T_{S,k}(x)\,\mathrm{dx}\right]\boldsymbol{u}_j + \\
&\quad \boldsymbol{u}_j^T\left[\frac{1}{h}\int_{-h/2}^{h/2} \boldsymbol{e}_{S,k}(x)\,\boldsymbol{e}^T_{S,k}(x)\,\mathrm{dx}\right]\boldsymbol{u}_j \\
&= \boldsymbol{u}_i^T E_{NN,k}\,\boldsymbol{u}_i - 2\,\boldsymbol{u}_i^T E_{NS,k}\,\boldsymbol{u}_j + \boldsymbol{u}_j^T E_{SS,k}\,\boldsymbol{u}_j
\end{aligned} \tag{B.1}$$

This representation of $\Delta^2_{i,j,k}$ is convenient when it comes to writing out the gradient of F_0 or F_1, but we can exploit the symmetry of the matrices in (B.1) to compute $\Delta^2_{i,j,k}$ more efficiently[B.1]. The matrices $E_{NN,0}$ and so on are obtained by writing out the outer product and integrating element-wise. For instance,

$$E_{NN,0} = \frac{1}{h}\int_{-h/2}^{h/2} \boldsymbol{e}_{N,0}(x)\,\boldsymbol{e}^T_{N,0}(x)\,\mathrm{dx}$$

B.1. For instance, if j is the northern neighbor cell of i, and writing $\delta_N\boldsymbol{u}$ for

$$\delta_N\boldsymbol{u} = \begin{pmatrix} u_{i,0} - u_{j,0} \\ u_{i,1} - u_{j,1} \\ u_{i,2} + u_{j,2} \\ u_{i,3} - u_{j,3} \\ u_{i,4} + u_{j,4} \\ u_{i,5} - u_{j,5} \end{pmatrix} \text{ and } \delta'_N\boldsymbol{u} = \begin{pmatrix} u_{i,0} - u_{j,0} \\ u_{i,1} - u_{j,1} \\ u_{i,2} - u_{j,2} \\ u_{i,3} - u_{j,3} \\ u_{i,4} - u_{j,4} \\ u_{i,5} + u_{j,5} \end{pmatrix}$$

then $\Delta^2_{i,j,0}(\boldsymbol{u})$ is equal to

$$\begin{aligned}
\Delta^2_{i,j,0}(\boldsymbol{u}) &= \frac{1}{h}\int_{-\frac{h}{2}}^{\frac{h}{2}} \left(\delta_N u_0 + \delta_N u_1\,x - \delta_N u_2\,\frac{h}{2} + \delta_N u_3\,\frac{x^2}{2} - \delta_N u_4\,\frac{h}{2}\,x + \delta_N u_5\,\frac{h^2}{8}\right)^2 \mathrm{dx} \\
&= \delta_N\boldsymbol{u}^T E_{NN,0}\,\delta_N\boldsymbol{u} \\
\Delta^2_{i,j,1}(\boldsymbol{u}) &= \frac{1}{h}\int_{-\frac{h}{2}}^{\frac{h}{2}} \left(\delta_N u_1 + \delta_N u_3\,x - \delta_N u_4\,\frac{h}{2}\right)^2 \mathrm{dx} = \delta_N\boldsymbol{u}^T E_{NN,1}\,\delta_N\boldsymbol{u} \\
\Delta^2_{i,j,2}(\boldsymbol{u}) &= \frac{1}{h}\int_{-\frac{h}{2}}^{\frac{h}{2}} \left(\delta'_N u_2 + \delta'_N u_4\,x - \delta'_N u_5\,\frac{h}{2}\right)^2 \mathrm{dx} = \delta'_N\boldsymbol{u}^T E_{NN,2}\,\delta'_N\boldsymbol{u} \quad \text{etc.}
\end{aligned}$$

$$= \frac{1}{h}\int_{-h/2}^{h/2} \begin{pmatrix} 1 & x & -\frac{h}{2} & \frac{x^2}{2} & -\frac{h\,x}{2} & \frac{h^2}{8} \\ x & x^2 & -\frac{h\,x}{2} & \frac{x^3}{2} & -\frac{h\,x^2}{2} & \frac{h^2\,x}{8} \\ -\frac{h}{2} & -\frac{h\,x}{2} & \frac{h^2}{4} & -\frac{h\,x^2}{4} & \frac{h^2\,x}{4} & -\frac{h^3}{16} \\ \frac{x^2}{2} & \frac{x^3}{2} & -\frac{h\,x^2}{4} & \frac{x^4}{4} & -\frac{h\,x^3}{4} & \frac{h^2\,x^2}{16} \\ -\frac{h\,x}{2} & -\frac{h\,x^2}{2} & \frac{h^2\,x}{4} & -\frac{h\,x^3}{4} & \frac{h^2\,x^2}{4} & -\frac{h^3\,x}{16} \\ \frac{h^2}{8} & \frac{h^2\,x}{8} & -\frac{h^3}{16} & \frac{h^2\,x^2}{16} & -\frac{h^3\,x}{16} & \frac{h^4}{64} \end{pmatrix} \mathrm{dx}$$

$$= \begin{pmatrix} 1 & 0 & -\frac{h}{2} & \frac{h^2}{24} & 0 & \frac{h^2}{8} \\ 0 & \frac{h^2}{12} & 0 & 0 & -\frac{h^3}{24} & 0 \\ -\frac{h}{2} & 0 & \frac{h^2}{4} & -\frac{h^3}{48} & 0 & -\frac{h^3}{16} \\ \frac{h^2}{24} & 0 & -\frac{h^3}{48} & \frac{h^4}{320} & 0 & \frac{h^4}{192} \\ 0 & -\frac{h^3}{24} & 0 & 0 & \frac{h^4}{48} & 0 \\ \frac{h^2}{8} & 0 & -\frac{h^3}{16} & \frac{h^4}{192} & 0 & \frac{h^4}{64} \end{pmatrix}.$$

The other matrices turn out to be

$$E_{SS,0}=\begin{pmatrix} 1 & 0 & \frac{h}{2} & \frac{h^2}{24} & 0 & \frac{h^2}{8} \\ 0 & \frac{h^2}{12} & 0 & 0 & \frac{h^3}{24} & 0 \\ \frac{h}{2} & 0 & \frac{h^2}{4} & \frac{h^3}{48} & 0 & \frac{h^3}{16} \\ \frac{h^2}{24} & 0 & \frac{h^3}{48} & \frac{h^4}{320} & 0 & \frac{h^4}{192} \\ 0 & \frac{h^3}{24} & 0 & 0 & \frac{h^4}{48} & 0 \\ \frac{h^2}{8} & 0 & \frac{h^3}{16} & \frac{h^4}{192} & 0 & \frac{h^4}{64} \end{pmatrix}, \qquad E_{NS,0}=\begin{pmatrix} 1 & 0 & \frac{h}{2} & \frac{h^2}{24} & 0 & \frac{h^2}{8} \\ 0 & \frac{h^2}{12} & 0 & 0 & \frac{h^3}{24} & 0 \\ -\frac{h}{2} & 0 & -\frac{h^2}{4} & -\frac{h^3}{48} & 0 & -\frac{h^3}{16} \\ \frac{h^2}{24} & 0 & \frac{h^3}{48} & \frac{h^4}{320} & 0 & \frac{h^4}{192} \\ 0 & -\frac{h^3}{24} & 0 & 0 & -\frac{h^4}{48} & 0 \\ \frac{h^2}{8} & 0 & \frac{h^3}{16} & \frac{h^4}{192} & 0 & \frac{h^4}{64} \end{pmatrix},$$

$$E_{EE,0}=\begin{pmatrix} 1 & \frac{h}{2} & 0 & \frac{h^2}{8} & 0 & \frac{h^2}{24} \\ \frac{h}{2} & \frac{h^2}{4} & 0 & \frac{h^3}{16} & 0 & \frac{h^3}{48} \\ 0 & 0 & \frac{h^2}{12} & 0 & \frac{h^3}{24} & 0 \\ \frac{h^2}{8} & \frac{h^3}{16} & 0 & \frac{h^4}{64} & 0 & \frac{h^4}{192} \\ 0 & 0 & \frac{h^3}{24} & 0 & \frac{h^4}{48} & 0 \\ \frac{h^2}{24} & \frac{h^3}{48} & 0 & \frac{h^4}{192} & 0 & \frac{h^4}{320} \end{pmatrix}, \quad E_{WW,0}=\begin{pmatrix} 1 & -\frac{h}{2} & 0 & \frac{h^2}{8} & 0 & \frac{h^2}{24} \\ -\frac{h}{2} & \frac{h^2}{4} & 0 & -\frac{h^3}{16} & 0 & -\frac{h^3}{48} \\ 0 & 0 & \frac{h^2}{12} & 0 & -\frac{h^3}{24} & 0 \\ \frac{h^2}{8} & -\frac{h^3}{16} & 0 & \frac{h^4}{64} & 0 & \frac{h^4}{192} \\ 0 & 0 & -\frac{h^3}{24} & 0 & \frac{h^4}{48} & 0 \\ \frac{h^2}{24} & -\frac{h^3}{48} & 0 & \frac{h^4}{192} & 0 & \frac{h^4}{320} \end{pmatrix},$$

$$E_{WE,0}=\begin{pmatrix} 1 & \frac{h}{2} & 0 & \frac{h^2}{8} & 0 & \frac{h^2}{24} \\ -\frac{h}{2} & -\frac{h^2}{4} & 0 & -\frac{h^3}{16} & 0 & -\frac{h^3}{48} \\ 0 & 0 & \frac{h^2}{12} & 0 & \frac{h^3}{24} & 0 \\ \frac{h^2}{8} & \frac{h^3}{16} & 0 & \frac{h^4}{64} & 0 & \frac{h^4}{192} \\ 0 & 0 & -\frac{h^3}{24} & 0 & -\frac{h^4}{48} & 0 \\ \frac{h^2}{24} & \frac{h^3}{48} & 0 & \frac{h^4}{192} & 0 & \frac{h^4}{320} \end{pmatrix}$$

$$E_{NN,1}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \frac{-h}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{h^2}{12} & 0 & 0 \\ 0 & \frac{-h}{2} & 0 & 0 & \frac{h^2}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \qquad E_{SS,1}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \frac{h}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{h^2}{12} & 0 & 0 \\ 0 & \frac{h}{2} & 0 & 0 & \frac{h^2}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$E_{EE,1}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \frac{h}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{h}{2} & 0 & \frac{h^2}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{h^2}{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \qquad E_{WW,1}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \frac{-h}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-h}{2} & 0 & \frac{h^2}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{h^2}{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$E_{NS,1}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \frac{h}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{h^2}{12} & 0 & 0 \\ 0 & \frac{-h}{2} & 0 & 0 & \frac{-h^2}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \qquad E_{WE,1}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \frac{h}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-h}{2} & 0 & \frac{-h^2}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{h^2}{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$E_{NN,2}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \frac{-h}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{h^2}{12} & 0 \\ 0 & 0 & \frac{-h}{2} & 0 & 0 & \frac{h^2}{4} \end{pmatrix}, \qquad E_{SS,2}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \frac{h}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{h^2}{12} & 0 \\ 0 & 0 & \frac{h}{2} & 0 & 0 & \frac{h^2}{4} \end{pmatrix},$$

$$E_{EE,2}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \frac{h}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{h}{2} & 0 & \frac{h^2}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{h^2}{12} \end{pmatrix}, \qquad E_{WW,2}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \frac{-h}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-h}{2} & 0 & \frac{h^2}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{h^2}{12} \end{pmatrix},$$

$$E_{NS,2}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \frac{h}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{h^2}{12} & 0 \\ 0 & 0 & \frac{-h}{2} & 0 & 0 & \frac{-h^2}{4} \end{pmatrix} \qquad E_{WE,2}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \frac{h}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-h}{2} & 0 & \frac{-h^2}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{h^2}{12} \end{pmatrix}$$

$$E_{\cdot\cdot,3}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \qquad E_{\cdot\cdot,4}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \qquad \text{and} \quad E_{\cdot\cdot,5}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The difference terms as defined in Section 2.4 may be written in the same form, the matrices $E_{NN,k}$ etc. will then have fewer nonzero entries.

Appendix C

Relationship Between $\boldsymbol{F_0}$ and $\boldsymbol{F_1}$

Making repeated use of relation (2.17) we can rewrite F_1 in (5.3) as

$$\begin{aligned}
F_1(\boldsymbol{u}) &= \sum_{i\in I}\Bigg\{\frac{1}{2\log 2}\Big(\frac{z_i-u_{i,0}}{\sigma}\Big)^2+c \\
&\quad +g\Big[1+2\max_{k\in\{1,\ldots,5\}}\tilde{\delta}(u_{i,k})+3\max_{k\in\{3,4,5\}}\tilde{\delta}(u_{i,k})\Big] \\
&\quad -\frac{g}{4}\sum_{j\in N_i}\Big(\delta(\Delta_{i,j,0})+2\max_{k\in\{1,\ldots,5\}}\tilde{\delta}(u_{i,k})\times\min_{k\in\{0,1,2\}}\delta(\Delta_{i,j,k}) \\
&\qquad +3\max_{k\in\{3,4,5\}}\tilde{\delta}(u_{i,k})\times\min_{k\in\{0,\ldots,5\}}\delta(\Delta_{i,j,k})\Big)\Bigg\} \\
&= \sum_{i\in I}\Bigg\{\frac{1}{2\log 2}\Big(\frac{z_i-u_{i,0}}{\sigma}\Big)^2+c \\
&\quad +g\Big[1+2\max_{k\in\{1,\ldots,5\}}\tilde{\delta}(u_{i,k})+3\max_{k\in\{3,4,5\}}\tilde{\delta}(u_{i,k})\Big] \\
&\quad +\frac{g}{4}\sum_{j\in N_i}\Big[\tilde{\delta}(\Delta_{i,j,0})-1 \\
&\qquad +2\Big(1-\min_{k\in\{1,\ldots,5\}}\delta(u_{i,k})\Big)\times\Big(\max_{k\in\{0,1,2\}}\tilde{\delta}(\Delta_{i,j,k})-1\Big) \\
&\qquad +3\Big(1-\min_{k\in\{3,4,5\}}\delta(u_{i,k})\Big)\times\Big(\max_{k\in\{0,\ldots,5\}}\tilde{\delta}(\Delta_{i,j,k})-1\Big)\Big]\Bigg\} \\
&= \sum_{i\in I}\Bigg\{\frac{1}{2\log 2}\Big(\frac{z_i-u_{i,0}}{\sigma}\Big)^2+c \\
&\quad +g\Big[1+2\max_{k\in\{1,\ldots,5\}}\tilde{\delta}(u_{i,k})+3\max_{k\in\{3,4,5\}}\tilde{\delta}(u_{i,k})\Big] \\
&\quad +\frac{g}{4}\sum_{j\in N_i}\Big[\tilde{\delta}(\Delta_{i,j,0})+2\max_{k\in\{0,1,2\}}\tilde{\delta}(\Delta_{i,j,k})+3\max_{k\in\{0,\ldots,5\}}\tilde{\delta}(\Delta_{i,j,k})\Big]\Bigg\} \\
&\quad +M(\boldsymbol{u}),
\end{aligned}$$

with $M(\boldsymbol{u})$ denoting a sum of mixed min-terms and constants,

$$\begin{aligned} M(\boldsymbol{u}) &= \frac{g}{4}\sum_{i\in I}\sum_{j\in N_i}\Big[2\min_{k\in\{1,\ldots,5\}}\delta(u_{i,k})\times\min_{k\in\{0,1,2\}}\delta(\Delta_{i,j,k}) \\ &\qquad +3\min_{k\in\{3,4,5\}}\delta(u_{i,k})\times\min_{k\in\{0,\ldots,5\}}\delta(\Delta_{i,j,k})-6\Big]. \end{aligned} \tag{C.1}$$

Thus $F_1(\boldsymbol{u})=F_0(\boldsymbol{u})+M(\boldsymbol{u})$ when $d=g$ and $b=g/2$.

Appendix D

More Example Results for the Berkeley Boundary Detection Benchmark

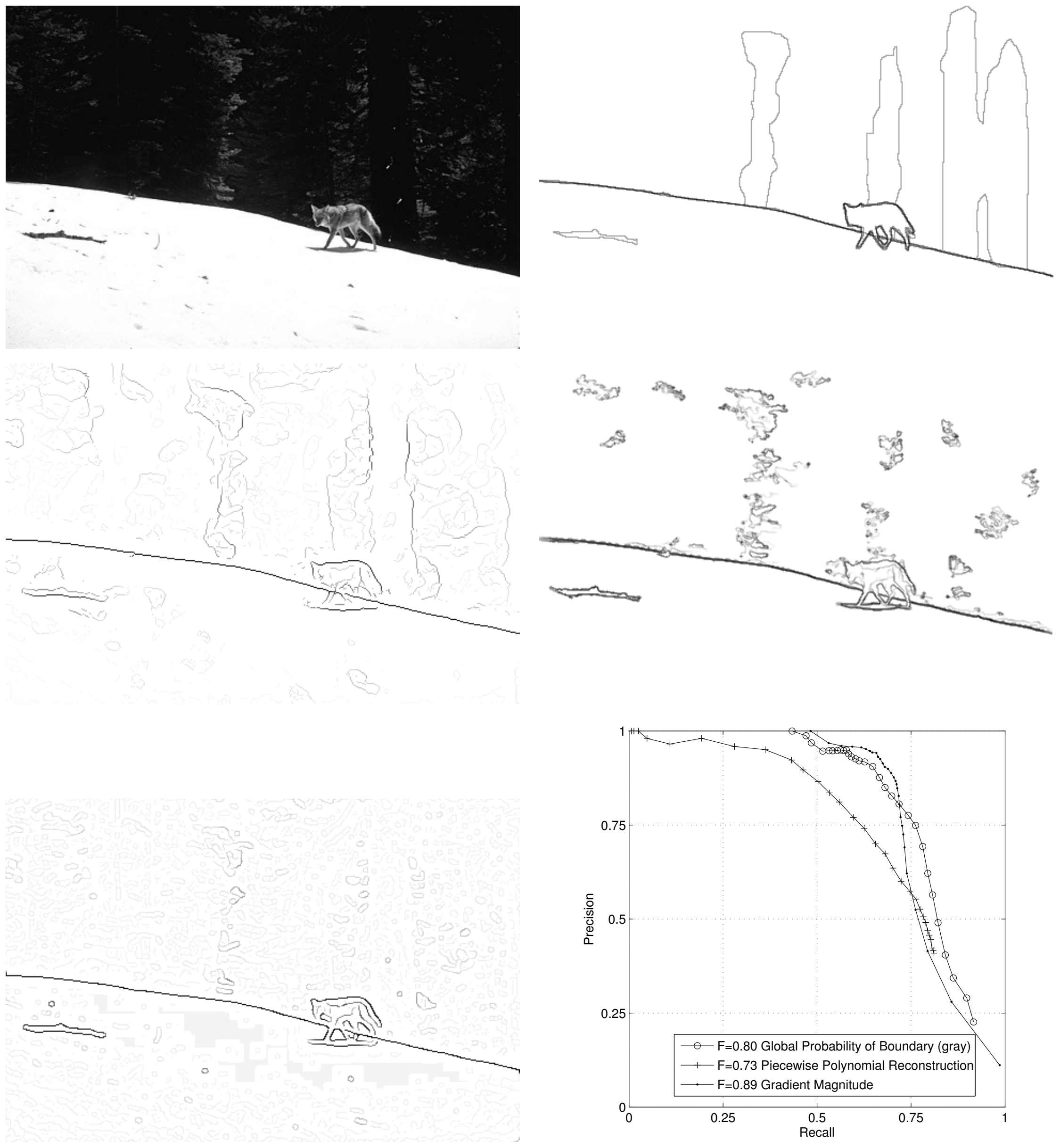


Figure D.1. Results for the Berkeley benchmark similar to Figure 6.13. For this image, algorithm "Gradient Magnitude" produces the best result.

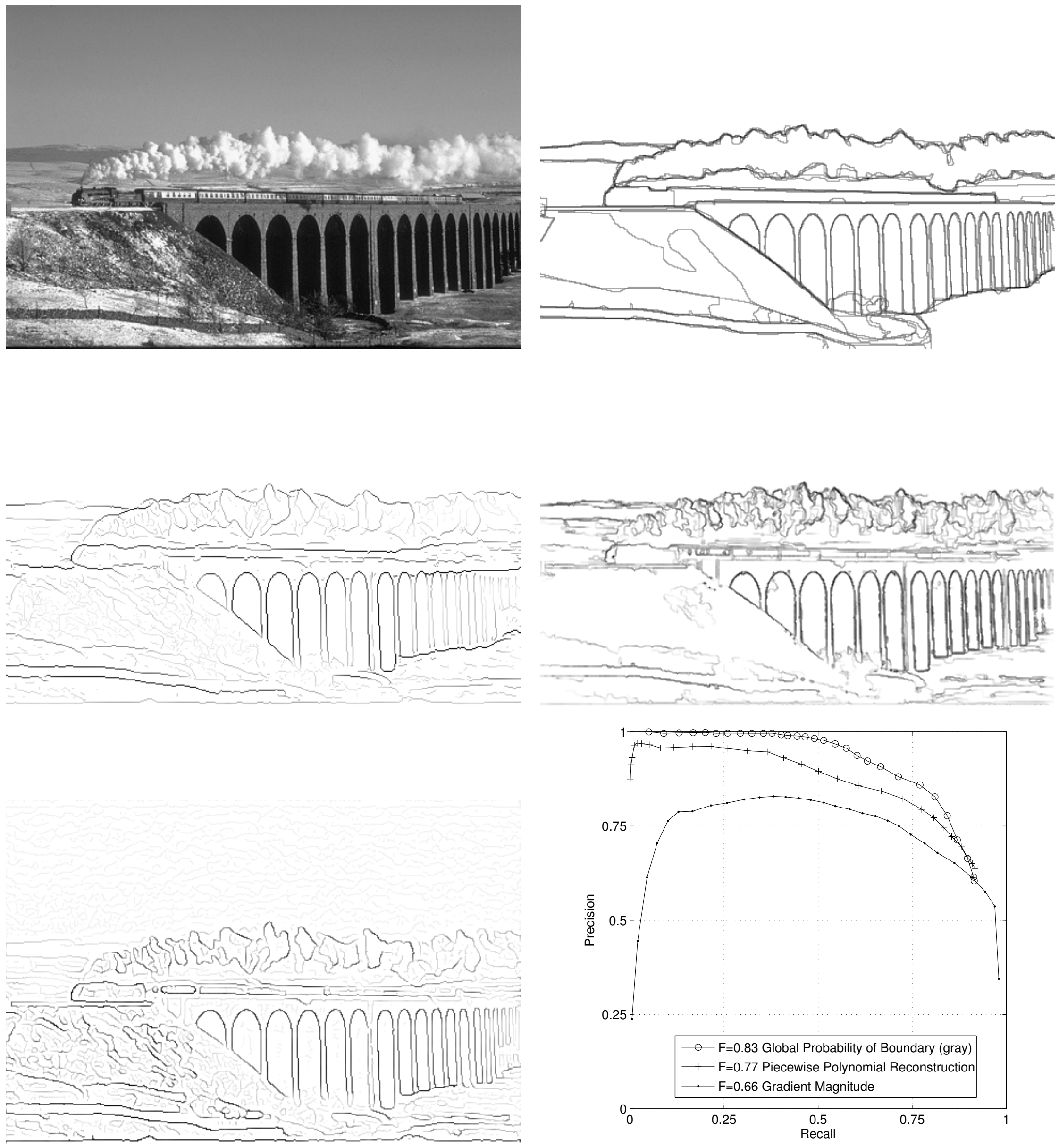


Figure D.2. Results for the Berkeley benchmark similar to Figure 6.13. For this image, algorithm "Global Probability of Boundary" produces the best result.

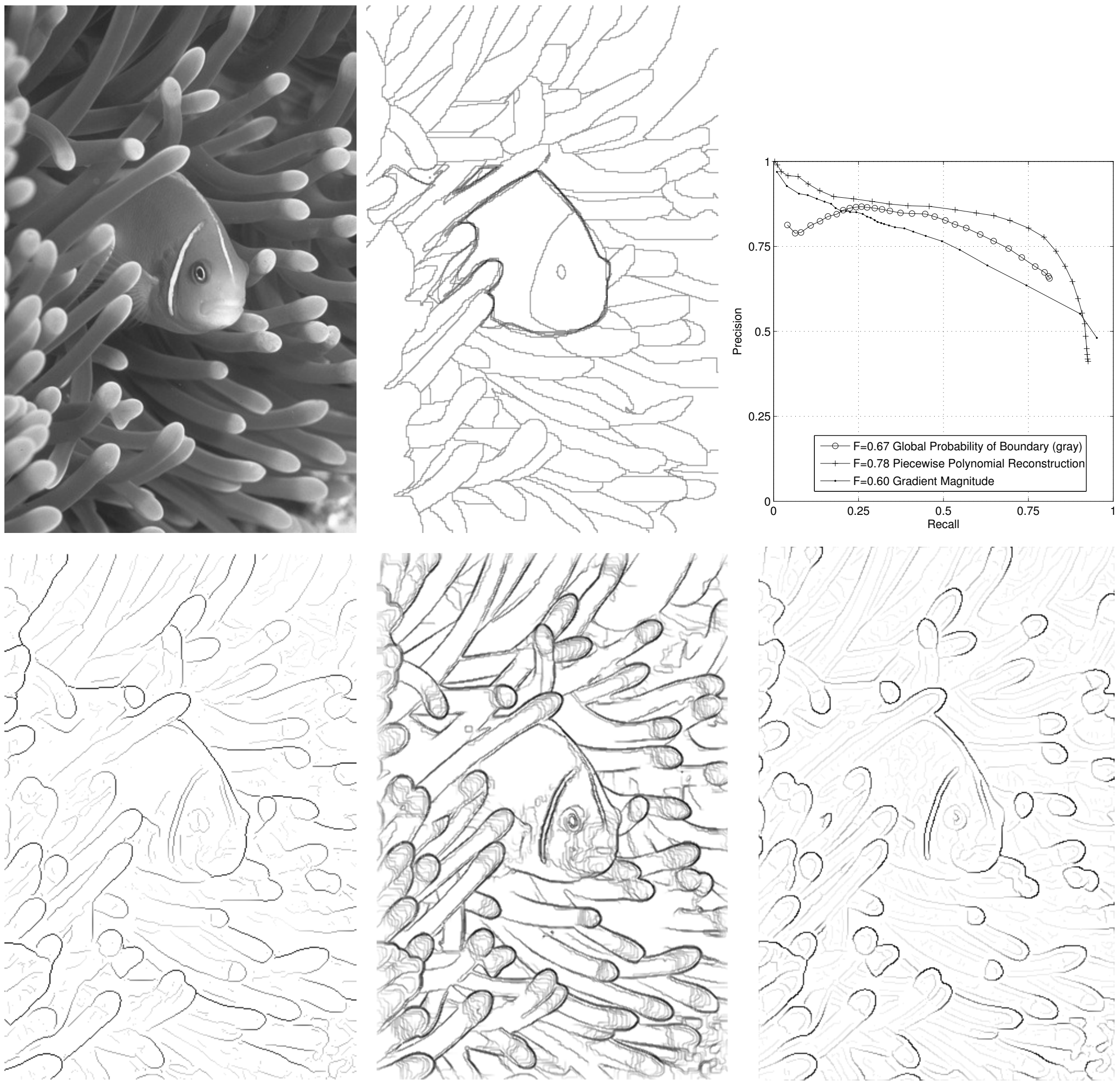


Figure D.3. Results for the Berkeley benchmark similar to Figure 6.13. Top row: image (left), ground-truth segments from human segmentations (middle), precision-recall curves (right). Bottom row: soft boundary maps produced by "Global Probability of Boundary" (left), "Piecewise Polynomial Reconstruction" (middle), and "Gradient Magnitude" (right), respectively. For this image, my algorithm ("Piecewise Polynomial Reconstruction") produces the best result.