Machine Vision for Froth Flotation

J.J Francis

Thesis presented for the Degree
DOCTOR OF PHILOSOPHY
in the Department of Electrical Engineering
UNIVERSITY OF CAPE TOWN

16 June 2001

Acknowledgements

- Professor Gerhard de Jager for supervising.
- UCT Minerals Processing Research Unit, Dr. Dee Bradshaw, for keeping us on course.
- Craig Sweet, Ben Wright, Esther Ventura-Medina, Doug Hatfield and Bart Rapacz for making this project memorable.
- The Digital Image Processing Group for providing a stimulating environment. Fred Nicolls, in particular, for providing a sounding board. Brian Shand, for listening to my stranger ideas.
- AngloPlats for financial and material support.
- NRF for financial support.

Abstract

Machine Vision for Froth Flotation
by **Jerome Jonathan Francis**
16 June 2001

Denoising, motion estimation and segmentation is examined in the context of froth imaging. Froth floatation is an industrial process seeking to separate valuable minerals from mined ore. Size, shape and motion of bubbles within froth are useful process indicators and, in principle, may be extracted via a machine vision system. This thesis examines the issues involved in selecting suitable algorithms for a real (or near real) time machine vision system.

The segmentation algorithm examined was Watershed segmentation. Watershed segmentation is a form of non-linear Voronoi tessellation of the image plane. The process of selecting the markers for optimum segmentation was addressed. The process of selecting good markers is complementary to the denoising process, since good marker selection implies the rejection of spurious markers caused by noise. It was determined that using a rolling ball of radius one pixel as a structuring element for erosion, and greyscale reconstruction of the eroded image, led to the domes being extracted by subtracting the reconstructed image from the original. The domes selected are the regional maxima of the original image. The markers are post processed by discarding dark and small markers. Dark markers may be defined as any marker whose maximum intensity is less than 0.4 of the maximum intensity pixel in the image.

This emphasises that the intended markers are the highlights of the bubbles. Small point-like markers were seen to be potentially noise induced. A connected components area opening removed small point-like markers of five or fewer pixels. The remaining markers may then be dilated to merge close markers. The structuring element used was a disc of radius 2. The parameters used were selected by trial and error for the images examined.

A weakness of the watershed segmentation process is described. This involves the over-segmentation of small bubbles on large bubbles. The extent of the small bubble is over-estimated. This is a result of the size and relative position of the markers for large and small bubbles respectively.

An attempt to segment an unprocessed froth image using watershed segmentation leads to over-segmentation of the bubbles. Many of the regions found (with the exception of small bubbles) do not correspond to actual bubbles on the froth surface. This implies that a noise removal stage is required. Many denoising algorithms are described in the literature. The problem for froth imaging is to find an optimum (fast and accurate) denoising algorithm which will eliminate noise and at the same time preserve the bubble boundaries as far as possible (a contradictory requirement). This edge preservation requirement leads to the rejection of linear noise elimination processes i.e low-pass filters. The optimum denoising filter must be then non-linear.

The non-linear filters examined included anisotropic diffusion, median filters, morphological filters and Occam filters. All by virtue of being non-linear, are to some degree edge preserving. The algorithms were quantitatively compared. The simple median filter, wavelet based denoising and anisotropic diffusion algorithms were fast enough for a real, or near real, time system. In terms of performance, however, anisotropic diffusion is preferred to wavelet based denoising, which is preferred to the simple median filter. The qualitative test used involved finding the filter which produced the least over-segmentation. The over-segmentation was visually quantified.

Many different algorithms have been examined for motion estimation of flotation froths. The algorithms range from correlation based to gradient based optical flow algorithms to algorithms more resembling image registration.

In a sense the process of finding a motion field is an image registration process. This suggests the use of the energy in the difference between the inverse warped motion image (at time $t + \Delta t$) and the reference image (at time t) as a performance measure of various algorithms. In the absence of this performance measure, two other possible performance measure suggest themselves. Somehow find a pair of images with known motion and compare the estimated and actual motion. Alternatively, a froth image may be artifically warped and an algorithm ranked in terms of how well it estimates the warping field.

Examination of the motion field and registration error showed that detecting burst bubbles, merged bubbles or highlight changed events is feasible. The variance of the motion field in the vicinity of the events is high, and the registration error is significant.

Block matching, optical flow type, and cluster (region) based motion estimation algorithms were examined. It was demonstrated that embedding block matching algorithms in a hierarchical framework reduced the running time of the algorithms without impacting on the performance. A hierarchical framework improved the performance of the optical flow algorithm used for large displacements (motion). The cluster based algorithms, while estimating large displacements well and being fast, performed relatively poorly.

The algorithms were quantitatively examined. Two hierarchical frameworks were examined. It was shown that when robustness, speed, performance, especially for large uniform displacement is considered, then Minimum Absolute Difference (MAD) or Mean Square Error (MSE) based Successive Elimination Algorithm (SEA) embedded in Hierarchical Framework 1 is preferred. For small motion, Horn and Schunk's algorithm, followed by Minimum Absolute Difference (MAD) or Mean Square Error (MSE) based Successive Elimination Algorithm (SEA) embedded in Hierarchical Framework 2 is suggested.

In short, algorithms for fast and reliable denoising, segmentation and motion estimation exist. It is feasible to detect bubbles merging or bursting or highlights changing shape or aspect.

Contents

List of Figures

List of Tables

Nomenclature

$\mathbf{x}$	Denotes a vector.
$\langle f, g\rangle$	Inner product. $\int f(x)\overline{g(x)}\,dx$
$\| f \|$	Euclidean norm.
∇	Gradient Operator $(\frac{\partial}{\partial x} \frac{\partial}{\partial y} \frac{\partial}{\partial z})$
∇^2	Laplacian. $\nabla^t \cdot \nabla$
$\overline{f}$	Complex conjugate of f.
$E\{\cdot\}$	Expectation operator
$f() * g()$	Convolution.
H^*	Matrix conjugate transpose
H^T	Matrix transpose
H^{-1}	Matrix inverse

AngloPlats Anglo American Platinum Corporation

BME Block Motion Estimation

CDF Cumulative distribution function

ChaCo Characterization of Flotation Froth Structure and Color by Machine Vision

DCT Discrete Cosine Transform

FFT Fast Fourier Transform

HVS Human Visual System

JPEG Joint Photographic Experts Group

MAD	Minimum Absolute Difference
ME	Motion Estimation
MPEG	Motion Pictures Expert Group
MPRU	Minerals Processing Research Unit
MSE	Mean Square Error
NGLDM	Neighboring Grey Level Dependence Matrix
NMAD	Normalised Mean Absolute Difference
PAL	Phase Alternating Line
PBIL	Population Based Incremental Learning
PCT	Principle Components Transform
PDE	Partial Differential Equation
PSNR	Peak Signal to Noise Ratio
RMS	Root Mean Squared
SEA	Successive Elimination Algorithm
SPIHT	Set Partitioning in Hierarchical Trees
SGLDM	Spatial Grey Level Dependence Matrix
SNR	Signal to Noise Ratio
SSE	Sum Squared Error
UCT	University of Cape Town
UMIST	University of Manchester

Chapter 1

Introduction

Froth flotation is a minerals extraction process which seeks to extract a mineral of interest from its ore [66]. Presently the froth flotation process is largely controlled by human operators using the visual characteristics of the froth, especially, bubble size and shape and motion. Based on experience and training, the operator then adjusts the process, for example, by modifying the reagent addition rate [52, 66]. A complication is that different operators interpret the visual features in different ways. Some, or all, of these visual features can, in turn, be measured by a machine vision system optimised for analysing froth images. A machine vision system then has potential for improving the consistency of process operation across shifts, compiling a history of froth visual changes which can be compared to assay results, and as a tool for investigating the relative importance of the visual features used by operators.

Preliminary investigation of a froth image involved taking a cut across a region of the froth and examining the intensity profile. A typical froth image is shown in figure 1.1 and an intensity profile is taken at line 25. A subregion taken from figure 1.1 is shown in figure 1.2(a), the position of the cut in figure 1.2(b) and the intensity profile in figure 1.2(c). It can be seen that each bubble in the froth has a bright region (a peak in intensity), or highlight. Where bubbles make contact with each other a valley forms, which tends to be darker than the surrounding pixels.

One could expect to see an increase in the profile as one ascends the side of a bubble, peaking at the highlight, and then decreasing until the boundary between two bubbles is encountered. The observed intensity profile, however, is different, extra extrema are present. The artifacts observed cannot be explained by some features on the bubbles themselves, which only leaves noise as a possible explanation. The minima in the intensity profile refer to the edges between adjacent bubbles. Any method which seeks to remove noise must, as far as possible, maintain the position of the edges. Edge preservation is important to correctly localise the bubbles within the froth and to accurately measure the extent of the bubble. Consider the same intensity profile as shown in figure 1.2(c) filtered with a low-pass filter. The low pass filter is a circularly symmetric Gaussian with sigma $\sigma = 2$, and radius 6. The filtered intensity profile is shown in figure 1.3. As expected, the small extrema are severely attenuated or removed, however the positions of the strong extrema are now uncertain. (A low pass filter blurs images.) This uncertainty in the positions of stronger extrema leads to difficulties in the localisation of bubbles, especially since close examination of the intensity profile (figure 1.2(c)) shows that the boundaries, or edges, between bubbles correspond to the strong minima. This implies that a denoising method should be as edge preserving as possible. Standard linear denoising techniques (see [18, 24]) do not preserve edges at all. Non-linear denoising techniques, on the other hand, are designed with edge preservation in mind (see, for example, [101, 148]).

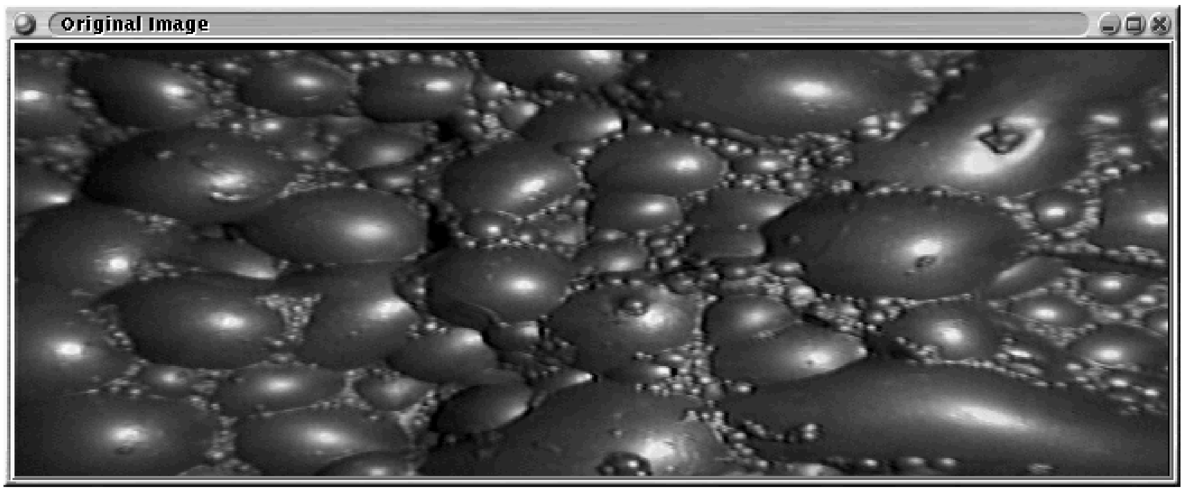
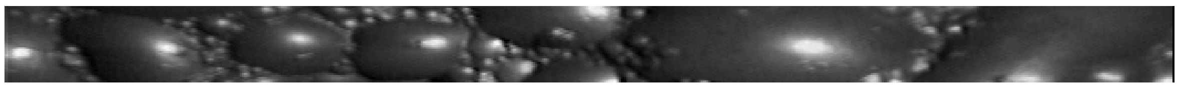


FIGURE 1.1 An example field of a froth image.

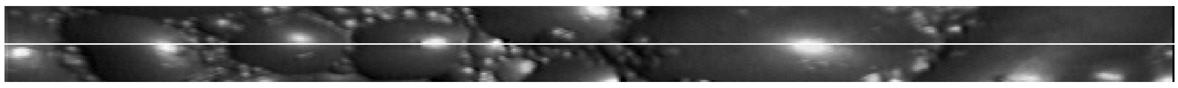
Developing a machine vision system for froth flotation demands that the features used by human operators be measured from froth images, these features include the size and shape of individual bubbles. This is an example of the general segmentation problem for which, at present, there is still no general theory. This can be seen from the many different approaches to segmentation ranging from texture, colour to statistical modelling based techniques (see [25, 57, 108, 128] for some very different approaches). Froth image segmentation is complicated by the absence of a background with different properties, usually segmentation is defined in terms of extracting an object with properties *different* to the objects surrounding it [48, 57, 105]. Here one cannot segment on shape (via connected components for example [17, 30]) since the bubble shape is highly variable (see figure 1.1 for an example froth image). Each bubble shares its properties with every other bubble making segmentation based on colour, motion or texture difficult.

While texture could be used to segment a froth image, only large bubbles could be distinguished from small bubbles lumped together. This is because the texture of a cluster of small bubbles together will be different to the texture of a large bubble.

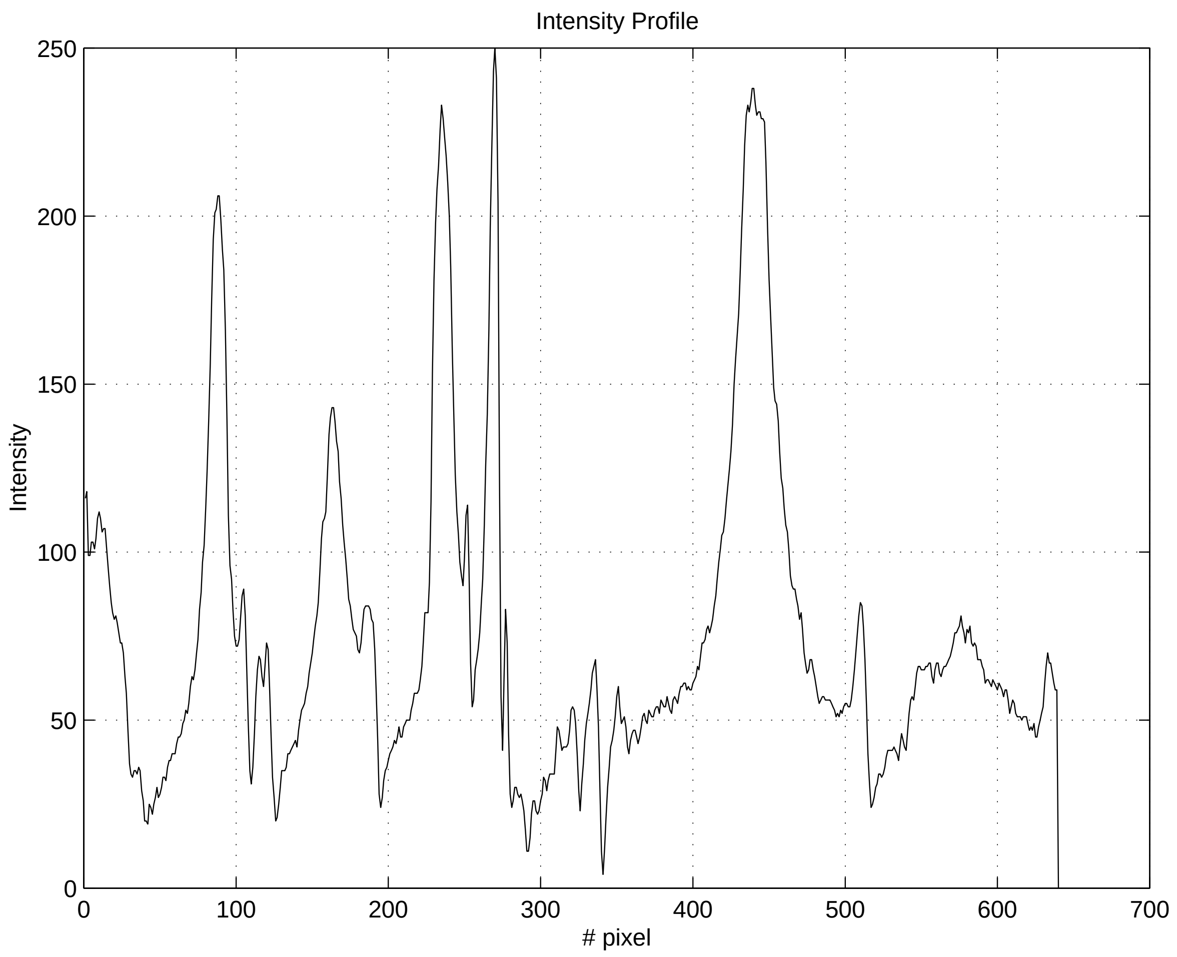
Thresholding could be used to find the bubble highlights. The difficulty is that the highlights have different intensities. This makes the selection of a single threshold capable of finding and correctly separating all highlights from each other, impossible. A bubble not well illuminated will have a relatively dim highlight. The dim highlight intensity will be similar to non-highlight pixels in the well illuminated regions of the image.



(a) Sub-region of froth image.



(b) Position of intensity cut.



(c) Intensity profile.

FIGURE 1.2 Extraction of an intensity profile.

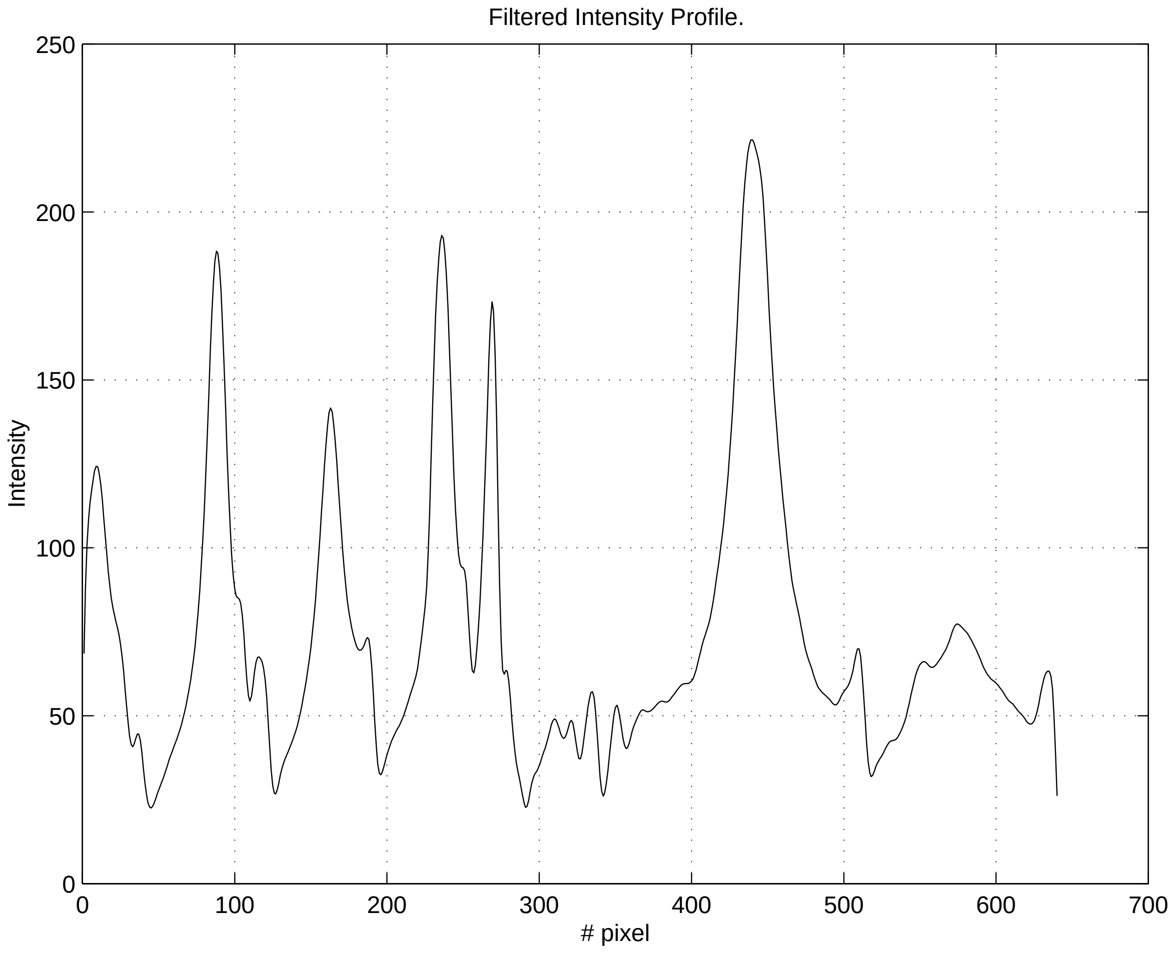
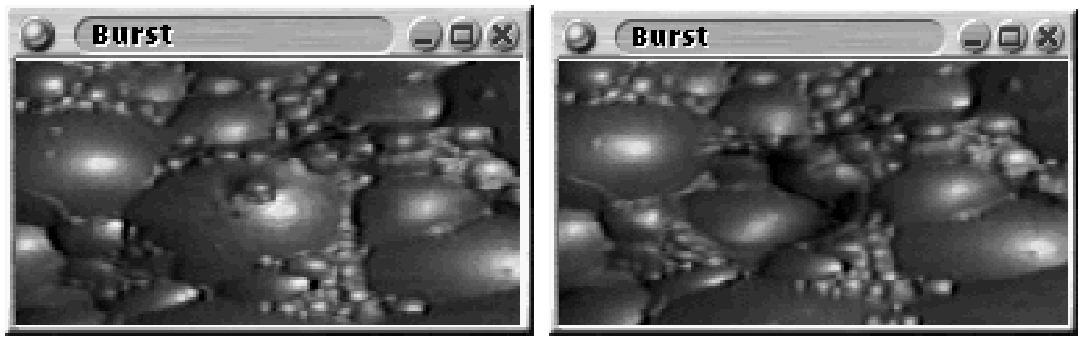


FIGURE 1.3 The intensity profile of a filtered image.

The difficulties involved in thresholding suggest a region growing segmentation algorithm [110]. Here one would find a point, or points, interior to each bubble, and grow the point(s) into a region, until the regions encounter each other. The region growing algorithm should, for accuracy and efficiency, take the intensity shape of the bubble into account. Some segmentation algorithms are examined, not only with regard to their suitability for the application at hand, but both qualitatively and quantitatively.

One of the features used by operators is the motion of the froth. Ideally, the froth contains a high proportion of the desired mineral and one wants this froth to overflow the flotation tank for later collection. The operator typically wants to know not only if this overflowing is occurring, but how fast it is happening [66]. This requirement raises the difficult problem of motion estimation (just how difficult can be seen in the volume of papers produced, see [158] for example).

Many motion estimation algorithms are described in the literature, these include block matching algorithms [9, 39], gradient based optical flow [2, 11, 138] and cluster and segmentation based algorithms [84, 128]. Implicit to these algorithms is the tracking of features from one image to the next. This is explicit in the correlation and cluster based algorithms [39, 128], and implicit in gradient based optical flow techniques where edges, and especially corners, are used to determine and propagate motion estimates [2, 68]. Motion estimation in froth imaging is complicated by bubble bursts, bubble merges and gross changes in the intensity profile of a bubble. The following figures 1.4, 1.5 and 1.6 illustrate bubble bursts, merges and gross changes in the intensity profile of a bubble.



(a) Even field of frame before burst event - frame 118 of cleaner data.

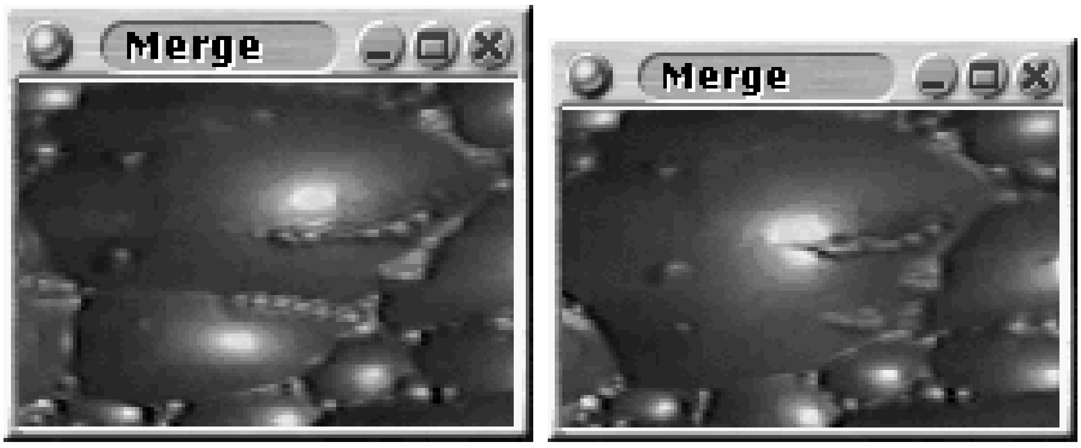
(b) Even field of frame for burst event – frame 119 of cleaner data.

FIGURE 1.4 A burst event.

The layout of this thesis is subsequently described.

1.1 Thesis layout

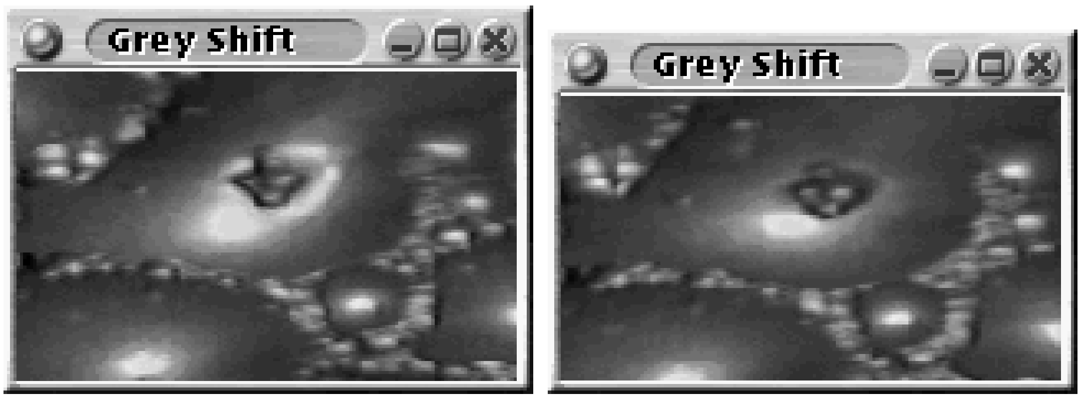
This thesis examines image denoising, segmentation and motion estimation in the context of developing a machine vision system for froth image, and froth image sequence analysis. The text is then divided into three parts, preceeded by a short description of froth flotation.



(a) Even field of frame before merge event - frame 22 of cleaner data.

(b) Even field of frame for merge event – frame 23 of cleaner data.

FIGURE 1.5 A merge event.



(a) Even field of frame before intensity change event - frame 111 of cleaner data.

(b) Even field of frame during intensity change event – frame 112 of cleaner data.

FIGURE 1.6 An example of a change in bubble intensity profile.

Part I examines segmenting froth images in some detail. Some segmentation algorithms are examined. A variety of methods for generating and subsequent processing of the marker images is examined. A marker serves to indicate the presence of a bubble. Once a bubble marker is found, region growing techniques makes finding the extent of the bubble possible (see [13, 65, 110] for background and examples of marker based segmentation). The denoising and pre-processing algorithms are examined in terms of their effect on the segmentation process, this leads to segmentation being examined first.

Part II examines the need for a preprocessing stage. This preprocessing stage is concerned with denoising and enhancing the froth image before further processing.

Part III examines the problem of estimating a dense motion field for a froth motion sequence. A variety of algorithms are examined and qualitatively and quantitatively compared. The problem of quantitatively comparing motion estimation algorithms is examined. The problem of detecting bubble bursts and merges is considered. (This will be seen in Chapter two to be relevant to a froth stability measure.)

Part IV concludes and makes some recommendations for a machine vision system for froth flotation.

The subsequent chapter introduces froth flotation from a non-specialist's perspective. It is in no way intended to be exhaustive and does not, for example, examine the chemistry or hydrodynamics involved in the process. The chapter also serves to describe previous attempts, with varying degrees of success, at a froth imaging system. Once the background has been described, the goals of this thesis are stated.

Chapter 2

Froth Flotation for Electrical Engineers

This chapter is divided into four sections. First, froth flotation is reviewed. Secondly, the potential benefits of applying machine vision to froth flotation are described. Thirdly, an overview of some machine vision systems applied to froth flotation, as described in the literature, is given. Finally, the problems examined by this thesis (and why) is stated.

2.1 What is froth flotation?

A problem in processing mined ore is that the proportion of desired material (for example, platinum) in the volume of mined material is small. Typical values include 2.6 grams of platinum per ton (Merensky) and between 3.5 and 29 g/t (UG2) [66]. A means of extracting the desired material (or in metallurgical parlance, separating or concentrating) is required. Of the variety of concentration methods available, froth flotation is an example. As the name implies, the process involves floating particles of the desired material and inhibiting the floatation of particles corresponding to the rest of the ore body. Two descriptors of this floated material include grade and recovery. In the context of platinum, grade refers to the proportion of platinum to other solid material floated. Recovery refers to the amount of platinum in the ore subsequently arriving in the floated product. The ideal of floatation, or any separation technique, involves maximising these two variables. Other separation methods include gravity, magnetic separation, electrostatic separation and various forms of floatation (see [126] for details). Froth flotation has been used for minerals concentration since 1887 [126].

The particular example used here describes flotation of platinum group metals. The description is fairly typical of flotation in general [126]. Platinum group metals (platinum, palladium, rhodium, iridium and ruthenium) have wide commercial use, from catalytic converters in motor vehicle exhausts to jewelry to fuel cells (platinum) [66].

The ore is first crushed to a fine gravel with a pebble diameter of several millimetres. The gravel is milled to a particular particle size, mixed to a slurry with water and passed through several stages of flotation to produce a concentrate which contains a higher percentage platinum than the ore. The concentrate is then smelted to produce platinum (or more correctly, a range of platinum group and some base metals, for example, nickel) in metallic form. A typical plant, as shown in figure 2.1, is subsequently described in detail.

Milling occurs in two stages. The ore is first milled and the result screened. Particles deemed too large are returned to the mill, particles small enough are shunted into the flotation rougher circuit. Water is added as part of the milling process, this helps to keep the milled product mobile.

Reagents are added to the pulp to promote flotation of the desired product. A conditioning stage, if needed, is used to provide enough time for reagents to finish reacting with the particles. Reagents are divided into activators, collectors, depressants and frothers. Activators prepare the particle surfaces to enable the collector molecules to attach. Activators also assist in collapsing the froth between flotation stages. Depressants inhibit the flotation of undesired product. Frothers stabilise the froth by decreasing surface tension. Collectors promote the flotation of the desired product. The exact mechanism by which the reagents work is poorly understood [52]. It is hypothesised that collectors make the desired particles (full or part) of desired product hydrophobic (and hence more likely to float), and depressants the particles of undesirable product hydrophilic (and hence less likely to float).

Since the interaction between reagents and froth is not clearly understood, it is possible that changing a reagent suite might change froth appearance. A reagent suite is the family of reagents used.

The conditioned pulp is first pumped into a series of interconnected flotation banks. Each flotation bank consists of a number of serially connected flotation cells. Within each cell the pulp is agitated by an impeller, and air is pumped into the cell. The impeller prevents sedimentation and serves to fragment the air entering the pulp. The contents of the cell then forms two layers; a pulp phase (liquid) and a froth phase (mostly air by volume). The froth then overflows the cell, if the pulp level is high enough, and is collected into launders to form the concentrate. If all goes well, the concentrate has a higher percentage of the desired product than the pulp entering the cell. The depleted pulp leaving the cell is referred to as the tailings of the cell. The tailings of one cell flows into the next cell of the current flotation bank. The tailings of a flotation bank is subsequently processed.

Flotation banks are differentiated by function. The rougher bank (one or more) receives the pulp immediately after conditioning. The rougher concentrate has a high concentration of the desired product and is fed into a cleaner bank. The cleaner concentrate is fed into a recleaner bank. The cleaner and recleaner improves the grade of the final product. The final concentrate is taken from the recleaner bank. The final concentrate is thickened, filtered, sometimes dried, and sent off to a smelter. The thickening process together with filtering removes excess water which is fed back into the process as a water conservation measure. The filter residue is sent to the smelter. The filtrate is fed back into the thickening process.

The rougher tailing is fed into a scavenger bank. The scavenger minimises the proportion of platinum in the final tail. The scavenger tail, or final tail, is then marked for disposal (thickening and a slimes dam). Again the excess water from the thickener is fed back into the mill to conserve water. The scavenger concentrate is fed back into the rougher bank. The cleaner tail is fed back into the rougher bank. The recleaner tail is fed back into the cleaner bank.

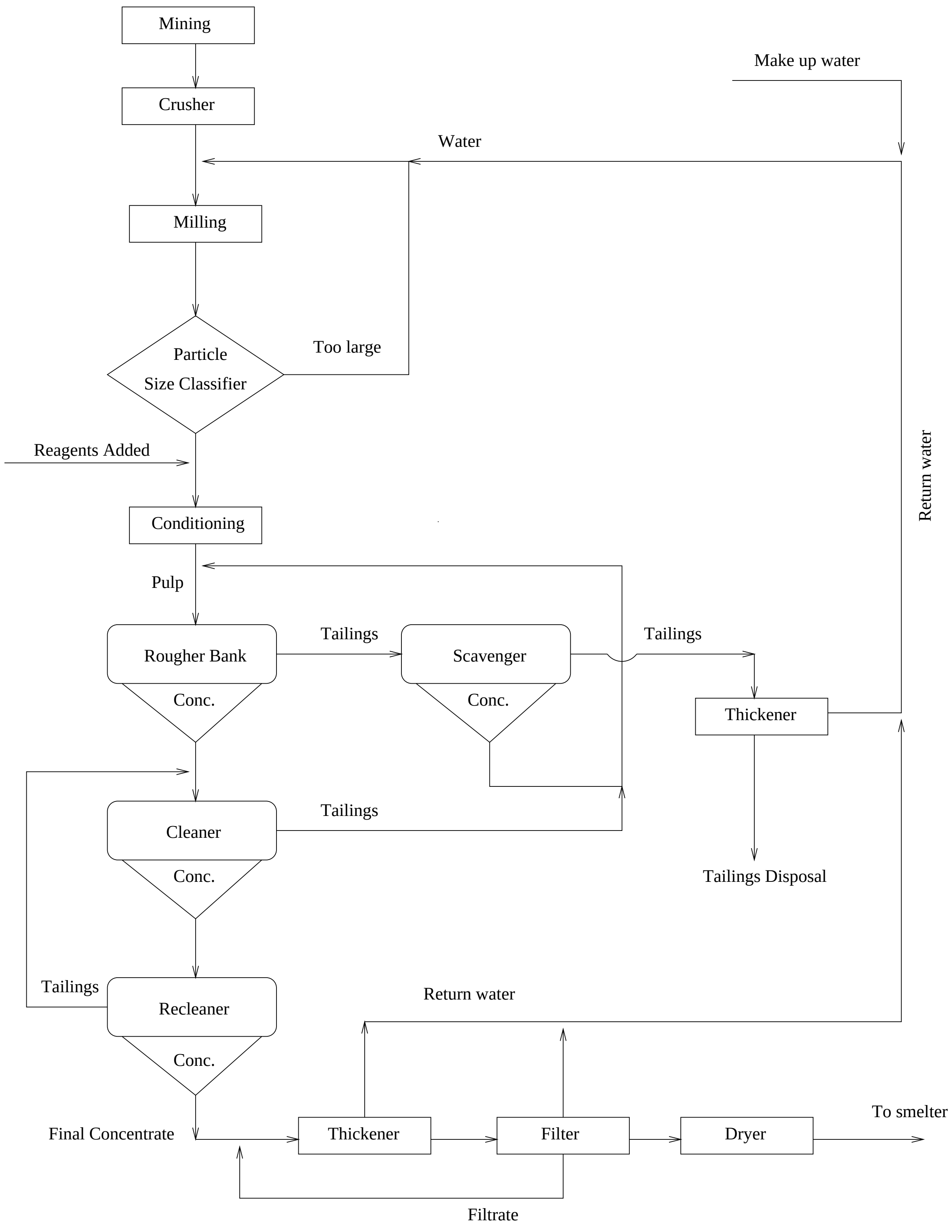


FIGURE 2.1 A flow-sheet for a typical concentrator.

2.2 Why apply machine vision to flotation?

Current operation of flotation banks involves an operator looking at the froth and then adjusting the process. The process variables include level, or height of the pulp (liquid), in the flotation bank, the rate of addition of chemical reagents, and adjusting the airflow (aeration) of the cell. It should be emphasised that *two* distinct processes are at work within the operator. The first process extracts the visual features of the froth. The visual features includes bubble size, colour, shape, movement and stability. The second process uses these features and infers the state of the flotation cell or bank. Based on prior experience and training, the operator will then adjust one or more of the process variables, or may well do nothing.

Since the control is based on the froth visual appearance, and image analysis has advanced enough to measure some, if not all, of these parameters, a machine vision system to analyse froth appearance is feasible. The advantages of a machine vision system include,

(i) Consistency across shifts. There is no guarantee that different individuals will quantify visual features in the same way. Machine vision promises the same visual features measured in the same way, always.

(ii) Compiling a history of changes in the froth visual appearance as the process variables change.

(iii) Potential as a training tool for operators.

(iv) Visual feature extraction for operator based control.

(v) In the long term, providing some of the the input variables for automatic control of the flotation process.

The development of the machine vision system was driven by, and provided data for, research linking froth appearance to the underlying metallurgy. This was the ambit of Craig Sweet and Doug Hatfield, undertaken at the Minerals Processing Research Unit (MPRU) at the University of Cape Town (UCT), and Ester Ventura-Medina's work, undertaken at the University of Manchester (UMIST) Froth and Foams Research Unit. Ben Wright implemented a prototype real-time machine vision system to measure some of the visual characteristics of froth [156, 157] (work undertaken at the Digital Image Processing unit, UCT). Bart Rapacz (in a masters project underway) is working on extending and refining the prototype real-time machine vision system.

Both the literature [53, 60, 96] and interaction with flotation bank operators indicate froth visual parameters (as described in table 2.1) useful for inferring the state of the process.

TABLE 2.1 Some useful froth visual parameters.

Parameter	Infers
Ellipticity of bubbles	Froth stickiness.
Orientation of bubbles with respect to the weir	Bubble loading. (Attachment of milled particles to the bubbles in the froth.)
Size distribution of bubbles	Froth stability, loading and yield of the process.
Motion estimation	Is the froth pulling, or flowing, over the weir.
Bubble bursts and coalescence	Froth stability.
Colour	Not an issue with platinum, except at the first one or two rougher cells and the recleaners. Colour is useful for other floats eg. copper and coal.

2.3 A review of some existing machine vision systems for flotation

This section reviews some previous and concurrent[1] attempts at at constructing a machine vision system for froth images and image sequences. It is difficult to create a time-line showing the development of the field but as far as possible the dates of submission of papers and the like for publication will be used to approximately date work.

2.3.1 JKFrothCam

JKFrothCam is a commercial instrument produced by JKTech. The core algorithms are pixel tracing (a motion estimation algorithm described, in detail, in section 14.3) and texture measurement via the texture spectrum (described subsequently). The motion estimation algorithm used is pixel tracing developed by Thornton and Nguyen [107]. The algorithm is essentially a correlation based algorithm with the space of motion vectors sub-sampled. A single block at the centre of the image is correlated with the corresponding block, offset, in the next frame. The motion vector finally selected is the one with the greatest correlation.

The texture measurement used is the texture spectrum [106] as developed by He and Wang [62]. The texture spectrum from a froth image is related to some set of predetermined froth types. Together with froth speed, an expert system evaluates the current froth type and suggests (or even controls process) to some pre-determined ideal state. Bubble size is related to the middle peak of the texture spectrum. The relationship is described as non-linear.

[1]At least, concurrent to this thesis.

Discussion

The pixel tracing algorithm does not give a motion estimate for all possible directions of motion. Pixel tracing is essentially block motion estimation (see [159] for some examples) with a radially subsampled search space and a *single* search block. This allows for a fast algorithm.

The texture spectrum is used to identify froth types (the control strategy is to push the flotation cell(s) to some predetermined froth type).

2.3.2 University of Manchester (UMIST)

Early work by Woodburn in 1994. Work has been done at the froth and foams research unit, under the direction of Jan Cilliers. The following papers are described,

(i) WOODBURN, AUSTIN AND STOCKTON (1994) [155] describe a bubble segmentation algorithm. The bubbles were extracted using a simple intensity threshold, and bubble area, perimeter and variance of area measured.

(ii) SADR-KAZEMI AND CILLIERS (1997) [120] describe a marker based watershed method for extracting the bubble boundaries. The images are pre-processed by applying histogram equalisation. The domes of the image are than found by first subtracting from the image a constant grey-level. The subtracted image is then grey-scale reconstructed. The difference between the original and reconstructed image yields then the marker image.

Discussion

(a) In (i) above, WOODBURN, AUSTIN AND STOCKTON do not describe the exact techniques used in the image processing. The paper implies that the features measured were measured from the thresholded bubble highlights.

2.3.3 University of Nottingham

This was work essentially on coal and tin flotation.

(i) HARGRAVE, MILES AND HALL (1996) [60] demonstrated a correlation between grey level (pixel intensity) and flotation process variables. The average intensity of pixels brighter than 75 and darker than 175 taken over three images, correlated with the ash content of the concentrate as well as the mass flow rate.

(ii) HARGRAVE AND HALL (1997) [59] investigated colour and texture in tin flotation. The bubbles were segmented via watershed segmentation. A colour profile was generated and shown to predict grade of the concentrate. The fractal dimension of the bubble size distribution was calculated. This showed two different dimensions and a breakpoint. This data, with colour information, was shown to be good predictors of flow rate via neural network modelling. Hargrave and

Hall postulate that the first fractal dimension yields information about froth instability (bubble coalescence and bursts), the second the small bubbles or basic froth structure, and the breakpoint the relative tradeoff between these two processes. A distortion profile of the bubbles is used as a descriptor of the speed of the motion of the froth. The distortion profile catalogues the direction of the semi-major axes of the bubbles. In fast moving froth the bubbles are stretched parallel to the direction of motion. In slow moving froth the bubbles are stretched perpendicularly to the direction of motion.

(iii) HARGRAVE, BROWN AND HALL (1997) investigate fractal dimensions as a descriptor of froth structure [58]. The Sierpinski Carpet is used as a model for froth images. The fractal dimension of the froth images were calculated from the bubble size distribution,

$$\beta = C\alpha^{2-D}$$

Where D is the fractal dimension, α the bubble area to remove, β the residual area. The bubble area removed and residual area are normalised and plotted on log axes. The slope of the line is the fractal dimension. It was determined that the froth images had two fractal dimensions (two lines on the log-log plot), separated by a breakpoint (or transition region). The two fractal dimensions identified two different structures in the froth, a basic structure of smaller bubbles and an unstable structure of larger bubbles.

(iv) HARGRAVE, SHIRAZI AND HALL (1997) [61] This is an attempt to correlate colour and surface texture with process performance characteristics (eg. grade, solids and water recoveries etc.). A feed forward neural network with one hidden layer was used to relate colour and texture with the process characteristics. Fractals are suggested as a texture measure.

Discussion

(a) HARGRAVE, MILES AND HALL The grey level range determination (75 to 175) is somewhat *ad hoc*.

(b) HARGRAVE AND HALL investigate colour and texture – The images are pre-processed by a “sharpening” filter before the watershed is applied. The sharpening filter is not described. For that matter the watershed algorithm is not described either – several exist in the literature [13, 31, 71, 141]. The neural network topology is not described. The fractal dimension is essentially a texture estimation process.

(c) HARGRAVE, BROWN AND HALL The segmentation algorithm used to segment the bubbles prior to generating the bubble size distribution is not described.

(d) HARGRAVE, SHIRAZI AND HALL Again the nature of the image processing algorithms (with the exception of the colour analysis) are not described in any detail.

2.3.4 The Catholic University of Chile (ACEFLOT)

(i) GUARINI, CIPRIANO, SOTO AND GUESALAGA (1995) [53] describe an image processing system for measuring froth parameters. Careful lighting ensures that the highlights of the froth are the bubble centres. The image is transformed into the HSI colour space and the intensity band further processed. The image is smoothed using a low-pass filter of extent 7×7, then stretched to fully occupy the grey level values from 0 to 255.

The highlights (brightest points on the froth) are found by edge detection (using a 3×3 domain) and comparing the intensity of the edge to a threshold. If *less* than the threshold the point is regarded as a candidate highlight. This prevents busy regions from being selected as highlights. The candidate highlights are finally processed by comparing each candidate highlight pixel to its 24 neighbours (in a 5×5 domain) and only selecting a pixel if the pixel is brighter than its neighbours. Minima are prevented from being closer than 5 pixels from the centre of the bubble.

The bubble boundaries are selected by scanning radially in 12 directions 30^o apart and finding the minima. The minima are joined (bubble edges are interpolated) using pieces of elliptical curves.

The measured bubble parameters include: mean diameter, major and minor axes (a shape descriptor), colour and area.

(ii) CIPRIANO, GUARINI, SOTO, BRICENO AND MERY (1997) [26] describe the operation and some of the algorithmic internals of the ACEFLOT system. The machine vision component of the ACEFLOT system is very similar to that described in the previous paper by Guarini et al. [53].

The bubbles are segmented by finding the highlights and moving away in twelve radial directions and finding minima along the scan direction. The boundary is then on the minima. Motion and instability is measured. Bubble parameters measured include: area, perimeter, diameter and circularity. Froth parameters measured include: number of bubbles, density and colour. An expert system evaluates the bubble and froth parameters, diagnoses a froth condition and suggests corrective action if any.

(iii) CIPRIAN, GUARINI, SOTO, SEPULVEDA AND BRISENO (1998) [27] describe ACEFLOT. The details provided resemble those provided in the previous papers.

Discussion

These papers document an attempt at directly measuring relevant froth image structures and relating these to the underlying process variables.

(a) GUARINI, CIPRIANO, SOTO AND GUESALAGA The low-pass filter is not described. The context in which this filter is described in the paper, implies a “pill-box” filter. The authors write,

> First a preprocessing step is performed to smooth the image in order to filter out noise. This is achieved through convolution with a 7×7 low-pass mask.

(b) CIPRIANO, GUARINI, SOTO, BRICENO AND MERY The motion estimation algorithm is not described. It is *implied* that it is fast [27]. The nature of the instability measurement algorithm is not described. The authors write,

> The foam velocity and stability is evaluated through the processing of consecutive images, at a rate of 20 frames per second. The speed is computed determining the movement of bubble centres from one frame to the next, averaging the results of 5 consecutive frames each time. The stability is a measure of the rate of bubble explosions. It is estimated comparing two consecutive image frames, and evaluating a measure of the rate of change in appearance. Again the results of 5 consecutive measures are averaged to present the result.

2.3.5 University of Amsterdam

(i) DE SWART, VAN VLIET AND KRISHNA (1996) [133] describe a machine vision system for segmenting bubbles in the liquid phase (pulp phase) of a bubble reactor. The images were converted to greyscale and thresholded. Since the images show a gas and liquid phase, the histogram of the image is bimodal, allowing for the selection of an optimal threshold. The thresholded regions correspond to the bubbles. The area and diameter of the bubbles are extracted.

2.3.6 University of Cape Town

Work here involved image analysis of graphite and platinum floats.

(i) PAUL SYMONDS (1992) [135, 136] First masters project at UCT. This attempted to segment the bubbles using a morphological edge detection method. The edges were detected using a spherical structuring element and subtracting the image from the morphologically closed image. The edge image is then thinned, and where possible, edge gaps are filled.

(ii) JIA LIU (1995) [89] Second masters project at UCT. This applied watershed segmentation to froth images. The segmentation was improved by linear low-pass filtering. The images were smoothed by convolving with a Gaussian kernel. In an effort to reduce over-segmentation, watersheds at multiple image scales are computed. Comparing the resulting watersheds allows for over-segmented regions to be detected and, to some degree, compensated for. The segmentations were compared to hand segmented images. It is demonstrated, via Chi-Squared goodness of fit tests, that the hand and automatic segmentations compare well. An older form of watershed using gradient descent is used.

(iii) MOOLMAN, ALDRITCH, VAN DEVENTER AND BRADSHAW (1995) [96]. This work proceeded independently of the work in the Digital Image Processing group. This paper is described in the subsequent subsection (subsection 2.3.7).

(iv) MPRU, ANGLOPLATS, UMIST PROJECT Collaborative project, funded by Anglo American Platinum Corporation (AngloPlats), between MPRU, UMIST and the Digital Image Processing Group at UCT. MPRU and UMIST relates features as measured by the machine vision system to the underlying process parameters. Digital Image Processing develops a real time machine vision system to measure froth parameters for froth image sequences.

A prototype real time machine vision system was developed [157], using a modern watershed algorithm (Vincent and Soille's algorithm [141]).

2.3.7 University of Stellenbosch/Crusader

(i) MOOLMAN, ALDRITCH, VAN DEVENTER AND BRADSHAW (1995) [96] describe a joint effort between the University of Stellenbosch and UCT. The approach is texture based using Sun and Wee's Neighboring Grey Level Dependence Matrix (NGLDM). The textural features extracted are then related, via linear regression, to various froth measurements, eg. motion, average bubble size and stability. Grade and intensity (for copper floats) were found to be correlated. A back propagation neural net was used to determine a relationship between grade, recovery and the parameters measured by the NGLDM.

(ii) MOOLMAN, ALDRITCH, VAN DEVENTER AND STANGE (1995) [97] use features derived from Spatial Grey Level Dependence Matrix (SGLDM) and NGLDM as inputs to a Learning Vector Quantisation neural network. Moolman et al. report better classification of froth types (five distinct types) using the Learning Vector Quantisation neural network as opposed to the back propagation network. It was determined that a feature set combining features from both NGLDM and SGLDM performed significantly better than either alone.

(iii) MOOLMAN, EKSTEEN, ALDRITCH AND VAN DEVENTER (1996) [98] motivate the use of a machine vision system for floatation froth analysis. The effect of changes in the froth appearance is related to the state of the flotation process. The paper provides a succinct overview of the effect the process variables have on froth appearance.

(iv) VAN DEVENTER, MOOLMAN AND ALDRITCH (1996) [29] apply the parameters as extracted by the machine vision system to a self organising map. This is an attempt to relate disturbances in the process to the froth appearance.

(v) BEZUIDENHOUT, VAN DEVENTER AND MOOLMAN(1997) [14] identify perturbations in the flotation process based on machine vision analysis on the froth appearance. It was shown that problems in the grinding circuit were consistently reflected in the parameters as measured by a machine vision system.

(vi) BOTHA, WEBER, VAN OLST AND MOOLMAN(1999) [16] describe an industrial platform for processing froth images on-line and in real time. Here the watershed algorithm is used for segmenting the froth image from a marker image. The marker image is extracted (via some undescribed process), thresholded, and used to modify the homotopy of the original image. This allows Vincent's[2] fast watershed algorithm to be used for segmentation. Several parameters are extracted from the segmented regions (bubbles) such as area, euclidian perimeter, ellipticity/circularity, orientation of the major axis and colour. Motion is estimated from the markers. The markers of the even and then the odd fields are extracted from the digitised interlaced image and the movement of the corresponding markers from even to odd field is measured to give an average motion of the bubble.

The Crusader system is commercial, and as such, little information is available. The system started out as an outgrowth of research done at the University of Stellenbosch. It is likely that most of the internals of the system are described in the papers above.

Discussion

(a) As MOOLMAN, ALDRITCH, VAN DEVENTER AND BRADSHAW admit in [96],

> *Although it is difficult to attach physical meaning to the parameters in this method*[3], the parameter SNE (small number emphasis of the feature matrix) can be seen as an indication of the fine texture of the image, whereas LNE (large number emphasis) is a measure of the coarseness of the texture. SM (second moment of the feature matrix) is a measure of the homogeneity of the image, and the NNU (non-number uniformity) and the ϵ_N (entropy) parameters are related to the coarseness of the image, *but it is difficult to explain which textural characteristics are explained by them.*[3]

The motion estimation algorithm described, averages over a number of images by temporal rather than spatial smoothing using a feature of the camera used. This is sensitive to the underlying incident light distribution on the froth. The smoothing algorithm is not described. It is likely that this was achieved by reducing the shutter speed. In fact if the smoothing window is too long (the shutter speed too low) the lighting distribution results. This is similar to the process of baseline subtraction subsequently described in section 11.3.

(b) BEZUIDENHOUT, VAN DEVENTER AND MOOLMAN, again make the following admission,

> *Unfortunately, it was not always possible to attribute a physical meaning such as average bubble size to individual image features*[3]

[2]Vincent's fast watershed algorithm is described subsequently in Part I of this thesis.
[3]My emphasis.

(c) BOTHA, WEBER, VAN OLST AND MOOLMAN describe their system in outline, but no details are discussed. For example, the highlight detection algorithm is described as follows,

> A robust morphological filtering technique is used to segment out these markers from an acquired froth image. This marker image is thresholded (in this case a trivial process due to the nature of the marker extraction algorithm) and used to modify the original froth image for watershedding.

The phrase “Robust morphological filtering technique” covers a great deal of ground (see [65] for the range of morphological techniques available).

The actual froth image features are not measured. It is difficult to relate the textural measurements to the bubble parameters such as size and shape and so on. The NGLDM and SGLDM provide features which are used to infer process state via some classification algorithm such as Neural networks, Learning Vector Quantisation or Kohononen maps. The Botha et al. paper is a considerable departure from the previous work, and is an implicit acknowledgement of the limitations of a texture based system for estimating bubble sizes.

2.3.8 University of Mining and Metallurgy, Cracow

(i) KORDEK AND KULIG (1997) [83] related the diffraction patterns of froth images to the “content of useful component.” Optical apparatus is used to convert the image into the Fourier domain. A 64 piece photodiode detector then digitises the Fourier image. The photodiode is configured as a RING-WEDGE detector. The detector plane is divided in two, one half is composed of wedges originating at the centre of the plane, and the other half is divided in to concentric half rings also centred on the centre of the plane. The wedges provide some directional information on orientation of structures in the image. Kordek and Kulig use the ring segment only, citing the lack of directional information in the froth images.

Kordek and Kulig transform the digitised images into the Fourier domain, filter using two band-pass filters, and inverse transform. The effect of these two filters are applied to adjust for grey-level differences between images and to allow for grey levels to be reduced. An inverse correlation between intensity and bubble mineralisation (or loading of the desired component) is demonstrated.

Discussion

The optical Fourier Transform is really unnecessary given the existence of the Fast Fourier Transform (an order $nlog(n)$ algorithm) [50, 73] and image acquisition hardware. All computations could have been performed on a digital computer with considerably better resolution in the frequency domain and without resorting to unusual hardware. The effect of the RING-WEDGE detectors may be simulated as well.

The precise nature and spectral characteristics of the filters applied are not described, nor is the reasoning behind the choice of these specific filters. It is well known that the use of sharp cut-off filters (as used here) in the Fourier domain leads to “ringing’ in the spatial domain (see [50]).

2.3.9 The ChaCo project

The Characterization of Flotation Froth Structure and Color by Machine Vision (ChaCo) project is an European Union funded project. The aim is to develop an on-line optical system for monitoring mineral froth variation and making optimisation for mineral processing production [147].

Division of Engineering Geology, Royal Institute of Technology, Stockholm

(i) WANG, STEPHANSSON, WANG (2000) [147] describe a system for measuring global froth parameters, and parameters for individual bubbles within the froth. The systems begins by examining the quality of individual images and rejects images of low quality. Global properties for the froth image as a whole are then extracted. Prior to segmentation, the image is enhanced to minimise the effect of noise. Individual bubbles are then segmented via valley edge finding. This attempts to detect the boundaries between bubbles. Edge tracing then fills any gaps in the valley edges bound.

The bubble highlights are extracted via using a dynamic thresholding method (described as optimal). The authors claim that the bubble highlights[4] vary with the size of the bubble. The size distribution of the highlights then is used to estimate the bubble size distribution.

Discussion The application and broad outline of the system is described. The details of every algorithm within the system are not supplied. The segmentation algorithm described (valley edge finding followed by edge tracing) is superficially similar to the bubble segmentation algorithm used within ACEFLOT (see subsection 2.3.4), and similar to work done by Paul Symmonds at UCT (see 2.3.6).

2.4 Thesis goals

The previous section details some machine vision systems for froth flotation. Implemented machine vision systems usually select image analysis algorithms based on a tradeoff between speed and accuracy. Examination of some of previous systems shows this dynamic in action. For example, the Pixel Tracing method is likely fast, but is not accurate since it does not examine the entire motion search space.

This thesis attempts direct measurements of the image, or image sequence, features of interest. The algorithms found, or developed, are ranked in terms of accuracy and speed. Engineering judgement then selects appropriate algorithms for inclusion into a machine vision system.

[4]The bright spot on each bubble.

This thesis examines the following,

(i) Marker based segmentation and how markers may be processed to improve segmentation is considered. Marker processing is expected to improve noise immunity. Paul Symonds and the ACEFLOT system both employ edge detection to delineate bubbles. Both report gaps in the delineations and present methods for filling in these gaps. This problem is considered. Since the UCT segmentation is largely Watershed based, it is necessary to find the limits of this technique.

(ii) It is shown (in Part I) how noise affects the watershed segmentation. It then follows that denoising techniques are needed which will not significantly affect the segmentation of bubbles. Noise removal is also expected to improve the accuracy of the motion estimation process. The problem of quantitatively comparing denoising algorithms is also considered. Denoising, while used, has not featured extensively in the UCT work on froth image analysis.

(iii) Image enhancement algorithms which improve contrast and segmentation are considered.

(iv) The motion estimation techniques, as detailed in the previous section, are either limited, as in JKFrothCam or the Crusader system, or inferred from other measures (ACEFLOT). These systems do not provide motion estimates at each pixel. Motion estimation of froth image sequences then needs to addressed in a systematic manner. Part of this involves the question of quantitative comparison of motion estimation algorithms. Motion, as well, has not featured extensively in the UCT work.

(v) A direct means of bubble burst and merge detection is considered. This is part of measuring the stability of a froth. This is new to the UCT work. Previous systems used indirect measures to assess stability (Crusader).

Part I

Segmentation of Froth Images

Chapter 3

Segmentation: An Introduction

Bubble size distributions are shown in Chapter 2 to be used by operators to control froth. An estimate of the size distribution may be generated by segmenting individual bubbles and then collating the bubble areas.

The issue of segmenting froth images will be examined here. Image segmentation is usually defined in terms of differences. The features of the segmented regions of an image differ from each other in some sense. Features used include intensity, colour, texture and shape (see [73, 95, 105, 146]). A froth image (see figure 3.1(a)) can be seen to consist of bubbles which have similar features. The shape, intensity gradients, texture and the like are essentially the same. This does imply that conventional segmentation strategies are of limited usefulness. This prompts the investigation of region growing segmentation schemes.

Region growing schemes involve finding a "seed" region (or marker) for each region to be segmented. This marker is the group of pixels which are the nucleus from which the region growing progresses. The problem of finding markers is not trivial (this should be at least as complex as the general segmentation problem). The stages followed include some denoising and some form of marker extraction, marker processing and finally using the markers to segment the froth image.

The domes of an image can be defined as the regional maxima of an image. Consider the domes of an unprocessed froth image (see figure 3.1)(These domes were extracted by subtracting five, for example, from each pixel in the original image, greyscale reconstructing by dilation and then subtracting the reconstructed from the original image. The constant, five, is arbitrary and is used for illustrative purposes. Greyscale reconstruction is subsequently described in chapter 4. To aid visualisation, each dome is set to the corresponding intensity on the original image.) Notice that there are a number of domes which do not correspond to highlights but are a result of noise. The domes are thresholded to form the markers. (It should be noted that for printing purposes this is an inverted image – the markers correspond to the dark regions.) There are two obvious problems with the unprocessed marker image: noise adds spurious domes and fragments larger domes, and not all regional extrema are in fact valid domes. This suggests denoising the image and rejecting all domes darker than some threshold. Merging close markers is another possibility.

This part (Chapters 3 and 4) examines the issues behind marker processing, the algorithms needed for marker based segmentation, and finally, connected components techniques.

This chapter subsequently examines the different watersheds possible (section 3.1), as well as the notion of connected components and its possible application to froth images (section 3.2). Chapter 4 examines the issues involved in extracting and processing the markers to reduce over-segmentation (chapter 4).

3.1 Watershed

Watershed segmentation classifies pixels based on distance from a centre, or seed, region. The algorithm is based on finding the watershed basins of an image, if one treats the greyscale image as a topographical height map. The notion of watershed segmentation has been in existence for some time(see [12] for some early work on watershed segmentation by Beucher and Lantejoul, circa 1979). Until Vincent and Soille's paper [141] a fast method for computing watersheds did not exist. The running time of the fast watershed is proportional to the number of pixels in the image to be processed [141]. Prior to the fast algorithm, the computation time was prohibitive, limiting applications of the technique.

3.1.1 Vincent and Soille

Vincent and Soille proposed a fast watershed segmentation algorithm [141]. The image is, conceptually, divided into a series of flat regions, where each flat region is of the same intensity. Note that some of these flat regions may only be one pixel in extent. Starting with the flat regions of lowest intensity, progressing to the highest intensity, if a flat region touches an already labelled region, the flat region is added to the labelled region. If a flat region touches a number of labelled regions, the flat region is partitioned. The pixels within the flat region are assigned to the labelled region they are closest to. The distance is a geodesic distance, in other words, the distance is the distance of a path completely contained in the flat region. If a flat region does not touch a labelled region, then it is assigned a new label, and becomes a new labelled region.

3.1.2 Marker based watershed – Beucher and Meyer

The algorithm is proposed by Beucher and Meyer [13]. Only certain minima of the image are allowed to be seeds for the growth of watershed regions. These preselected regions (or points) are referred to as markers. The markers are computed by some outside process. Each marker is assigned a unique label. The pixels corresponding to the marker positions are assigned the lowest pixel intensity in the image.

The watershed process proceeds as before with the water level, or flood level, increasing gradually. Regional minima which before would have become new watershed regions are now simply filled in and partitioned by their neighbours. The region growing, or flooding process, assigns unlabelled pixels the label of the first labelled pixel in their neighbourhood.

Unlike Vincent's algorithm, distances are not explicitly propagated. This, potentially, may lead to some difficulties, which are considered in the next subsection.

3.1.3 Some difficulties with conventional watershed algorithms

The watershed algorithm as developed by Vincent and Soille [141] and extended by Beucher and Meyer [13] for marker based segmentation, exhibits some difficulties.

Vincent's watershed algorithm is optimised for a 4-connected grid, or for the hexagonal grid (this optimisation is not made explicit [141]). For these grids the distance between neighbouring pixels is the same (unity). The 8-connected sampling grid, used in this thesis, is different, the pixels in directions NW, NE, SE and SW are further away from the centre pixel ($\sqrt{2}$ pixels) than the other pixels in the neighbourhood (1 pixel). This modification is simple to implement.

Vincent's algorithm is not marker based, effectively, every minimum in the image is a marker which becomes a watershed region [141]. If one wished to remove a minimum, for any reason, a homotopy modification is required [141, 144]. Homotopy modification removes a minimum by first setting the pixel intensities within the minimum to the maximum intensity on the image (or larger) and then performing a greyscale reconstruction by erosion of the modified image over the original. Greyscale reconstruction by erosion is the dual of greyscale reconstruction by dilation, in other words, greyscale reconstruction by dilation on an image is equivalent to greyscale reconstruction by erosion on the *inverted* image [144]. The greyscale reconstruction fills in the entire minimum which has been masked and removes any effect it might have had on the watershed. Use of a marker based watershed removes the need for homotopy modification. This suggests using a marker based watershed such as Beucher and Meyer's algorithm.

Beucher and Meyer's algorithm does exhibit some difficulties. Firstly, distances are not explicitly propagated. In the instance of froth image analysis, few flat regions are present and this does not affect the segmentation significantly. However, consider the segmentation of a plane with a few isolated points (pixels) used as markers. Since the labelling of pixels at the boundaries of the watershed regions is affected by the order in which the neighbourhood is scanned and which pixel is labelled first, the segmentation could change if these were different. It is reasonable to expect the watershed segmentation of an image to be unchanged by minor details such as the order in which the pixels in a neighbourhood are scanned.

This thesis then uses a modified form of Beucher and Meyer's algorithm in which distances are explicitly propagated as Vincent suggests. The NE, NW, SE and SW pixels, in a 3×3 neighbourhood, are $\sqrt{2}$ pixels further from the centre pixel than the N, S, E and W pixels. Each labelled pixel (child) has the distance from the pixel labelling it (the parent pixel), noted. If another parent pixel is closer, then the child pixel is allowed to change labels (or distances, if the second parent has the same label).

Difficulties common to all watersheds for froth segmentation include "comet-tailing" which occurs when a small bubble lies on a larger bubble. The ideal segmentation delineates the smaller bubble only. The traditional watershed on the other hand segments the bubble but exaggerates the smaller bubble to include part of the larger bubble. This is due to the shape and location of the markers. See figure 3.2 for an example of comet-tailing. This is likely to happen in all watershed algorithms if a small marker is close to a larger one, or a small relatively unflooded minimum is close to a large flooded minimum.

3.1.4 Watershed on the ridge image

This is an attempt to circumvent the problems as described before. A marker based watershed is used for the segmentation. The markers are extracted by an outside process (marker extraction is discussed in detail in chapter 4 following). The following ridge extraction schemes are considered: morphological reconstruction methods, a composite watershed and morphological filter schemes.

Prior to segmentation the image is first de-interlaced, then denoised via Perona and Malik's algorithm using 4 iterations, a time step of $\gamma = 0.2$ and $g(\cdot) = e^{(-(\|\nabla I\|/K)^2)}$. the parameter K is set automatically via Canny's noise estimator (see section 7.1 for a detailed description of Perona and Malik's algorithm).

3.1.5 Ridges from greyscale reconstruction

It is possible to modify greyscale reconstruction algorithms to apply to a single line of an image instead of the image itself. This modification is simple to implement and produces a simple greyscale reconstruction algorithm. This can be achieved by, for example, modifying Vincent's sequential greyscale reconstruction algorithm to only consider pixels on the same scan line as the pixel to be reconstructed [144]. The difference between the greyscale reconstructed image and the original image yields the ridges of the image. The ridges of an image will be defined as the local maxima separating local minima from each other. This is the same as the topographical use of the term (treating images as a topographical height map). The process described will extract isolated regional maxima (peaks) as well as ridges. The dual greyscale reconstruction by erosion may be used to find the valleys of an image by subtracting the original image from the dual greyscale reconstructed image.

The process is as follows,

(i) Extract the ridges of the inverted froth image.

(ii) Compute the markers of the denoised froth image.

(iii) Perform a marker based watershed, using the markers computed above, on the ridge image.

Figure 3.3(a) shows an example of ridges for a froth image. Note that isolated maxima are present. Figure 3.3(b) shows the markers used for a segmentation. The markers are extracted by finding the domes of the image. The domes are formed by subtracting a greyscale reconstructed image from original. The greyscale reconstructed image is formed by greyscale reconstruction by dilation of an eroded image under the original. The eroded image is formed by a rolling ball erosion of radius one on the original image. The issues involved in marker extraction are subsequently explored in chapter 4. Figure 3.3(c) shows the segmentation overlaid on the original image.

3.1.6 Watershed on the distance function of the ridge image

Examination of the ridge image shows that the bubbles are not completely delineated by the ridge lines. The question of how to close the gaps in the ridges can be posed. The distance function of the ridges to the background pixels is computed. (The distance from each background pixel to the nearest ridge pixel is computed.) The watershed is computed using the denoised image markers on this distance image. Figure 3.4 shows the ridges, distance function and final watershed on the distance function using the given markers. The markers used are identical to those used in the previous subsection.

3.1.7 Discussion

It is easy to see that the watershed algorithm in particular, and region growing algorithms in general, are specific examples of Voronoi tessellations of the image in question (see [110] for a introduction to Voronoi tessellations). Close examination of Vincent and Soille's [38, 141] fast watershed algorithm show that, at a particular flood level the unlabelled pixels are labelled according to the closest (in geodesic distance) labelled region (or pixel). This implies (for a computationally naive algorithm) that all paths from labelled pixels to the unlabelled pixels need to be examined.

Given a path γ, where γ_k indexes a particular pixel along the path (as an $(x_k,\ y_k,\ z_k)$ triple), and γ_0 refers to the first pixel and γ_N refers to the last pixel.

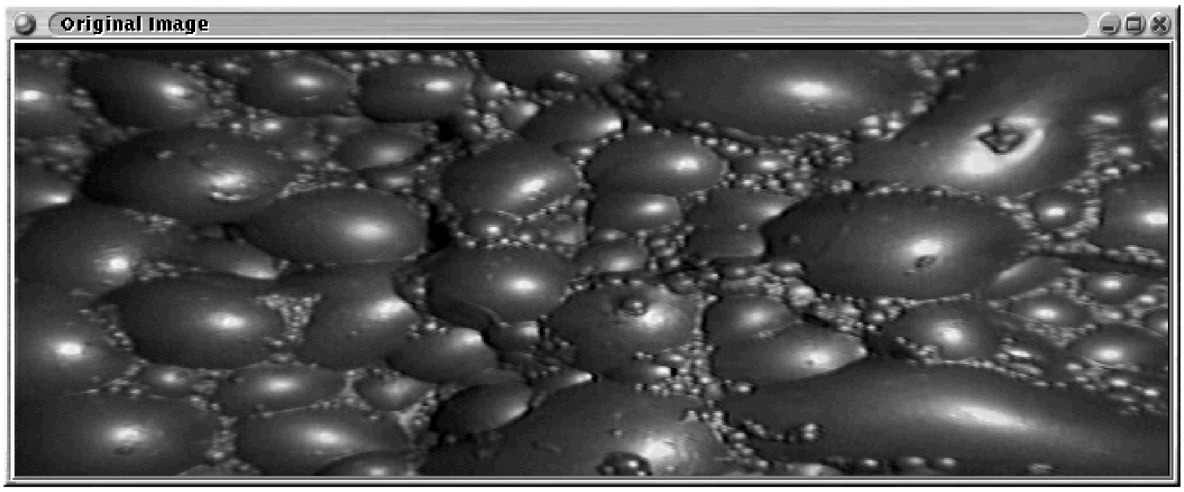
$$\text{Path length}(\gamma) = \left(\sum_{k=1}^{N} \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} \right) + \alpha \left| \max(z_k : \forall \gamma_k) - z_0 \right| \tag{3.1}$$

The value of α is large ($\gg$ diameter of the image) for a typical watershed segmentation algorithm. Given an image then, and a set of markers, the distances from any pixel to the markers may be computed. The distance is then,

$$\Gamma(p_0, p_1) = \text{argmin}_\gamma \{\text{Path length}(\gamma) : \gamma \text{ a possible path between } p_0 \text{ and } p_1\} \tag{3.2}$$

This then makes explicit the relationship between watershed segmentations and Voronoi tessellations of the plane. Specifically, watershed segmentation can be seen to be a form of multiplicative weighted Voronoi tessellation (see [38, 110] for an overview of Voronoi tessellations).

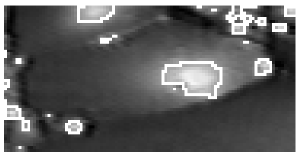
The watershed on the ridge image generates a good segmentation, but is computationally intensive.

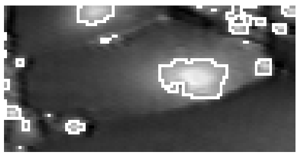


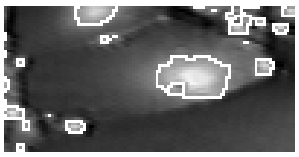
(a) A typical froth image.

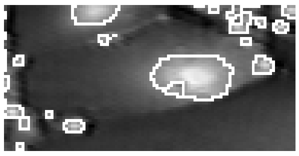
(b) The unprocessed domes of a typical froth image.

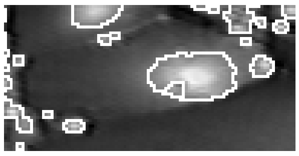
FIGURE 3.1 A typical froth image and associated domes.

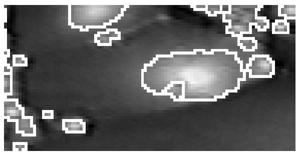

(a) Intensity level 90.

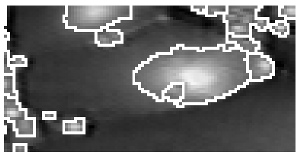

(b) Intensity level 100.

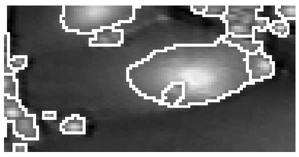

(c) Intensity level 110.

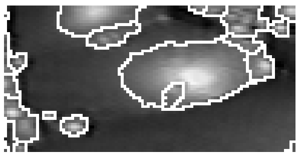

(d) Intensity level 120.

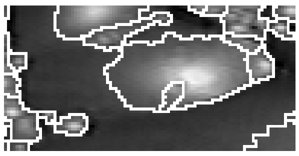

(e) Intensity level 130.

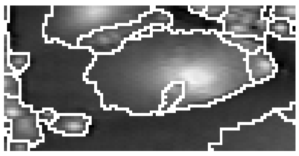

(f) Intensity level 140.

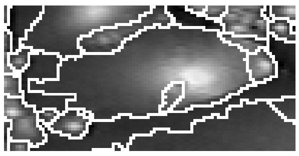

(g) Intensity level 150.

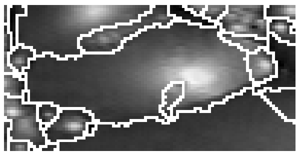

(h) Intensity level 160.

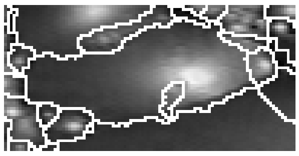

(i) Intensity level 170.


(j) Intensity level 180.


(k) Intensity level 190.


(l) Intensity level 200.


(m) Intensity level 210.


(n) Intensity level 220.

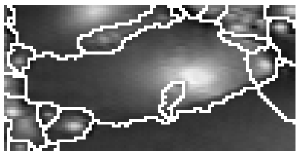
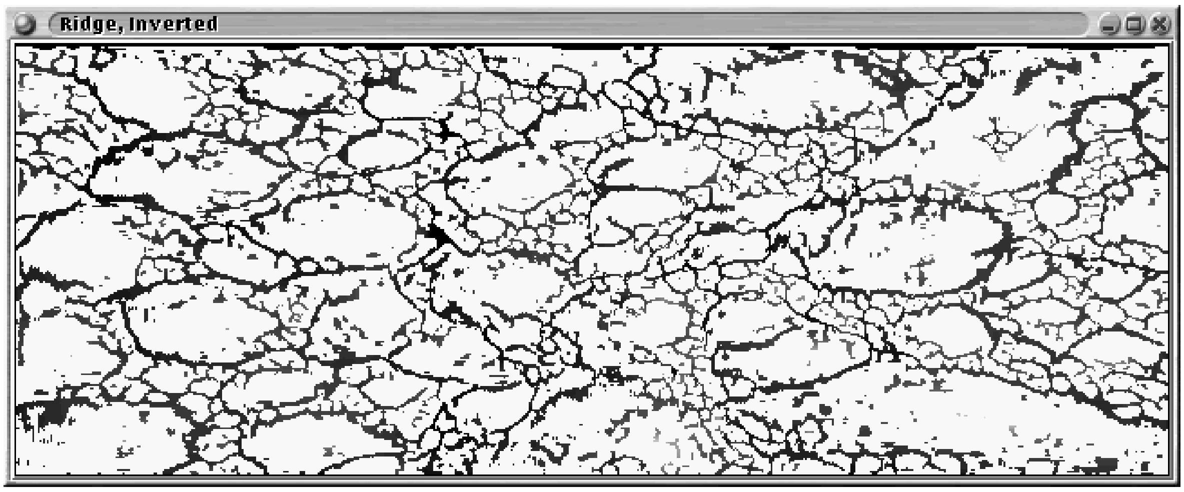
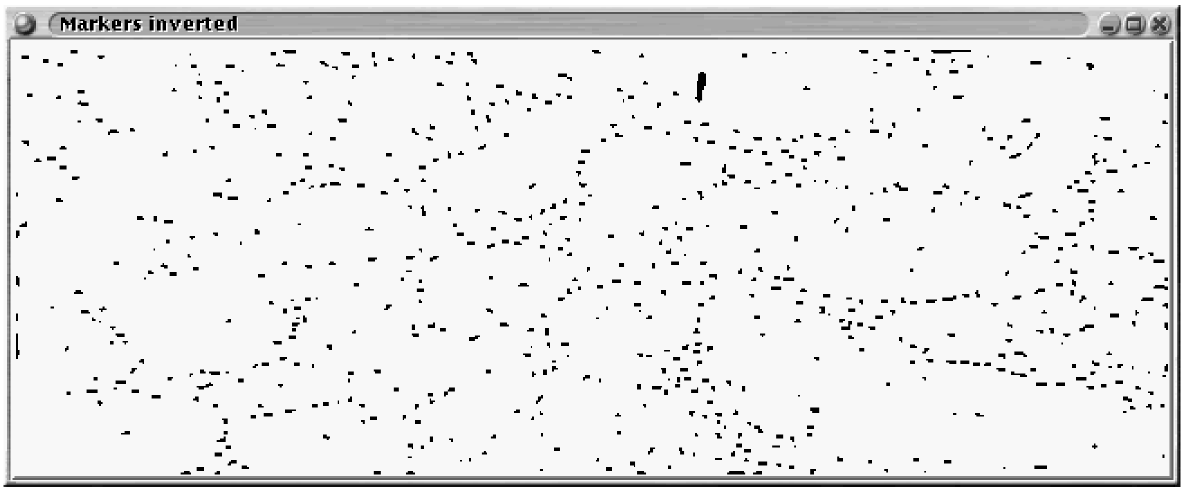

(o) Intensity level 230.

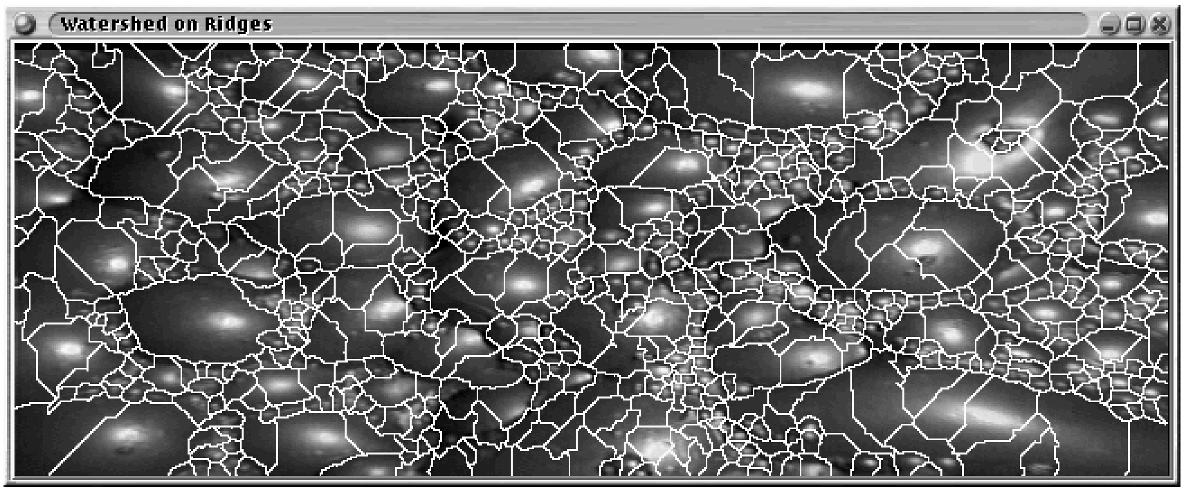
FIGURE 3.2 An example of comet-tailing. The white lines are watershed lines at different stages of the watershed segmentation.



(a) Ridges.

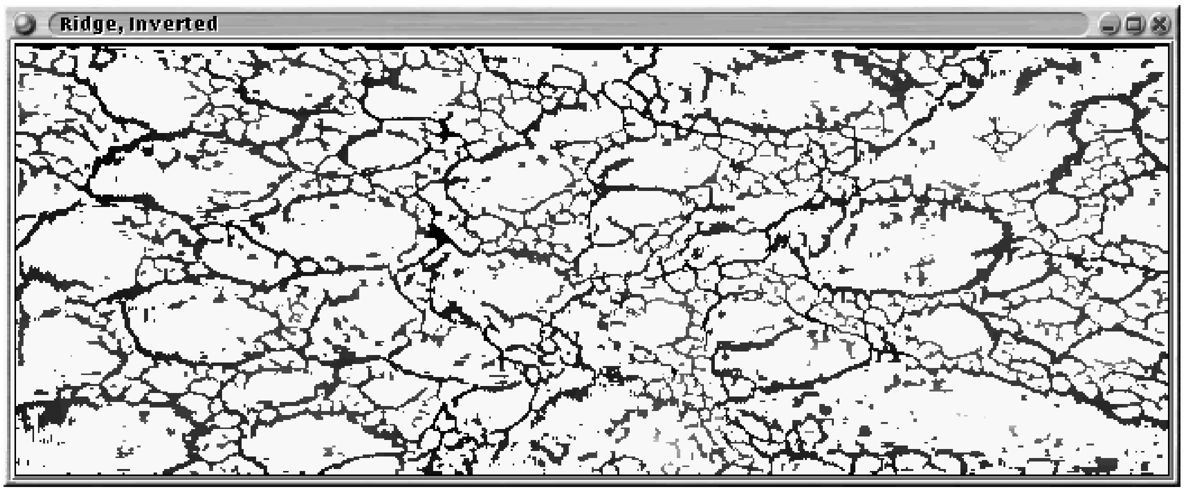


(b) Markers.

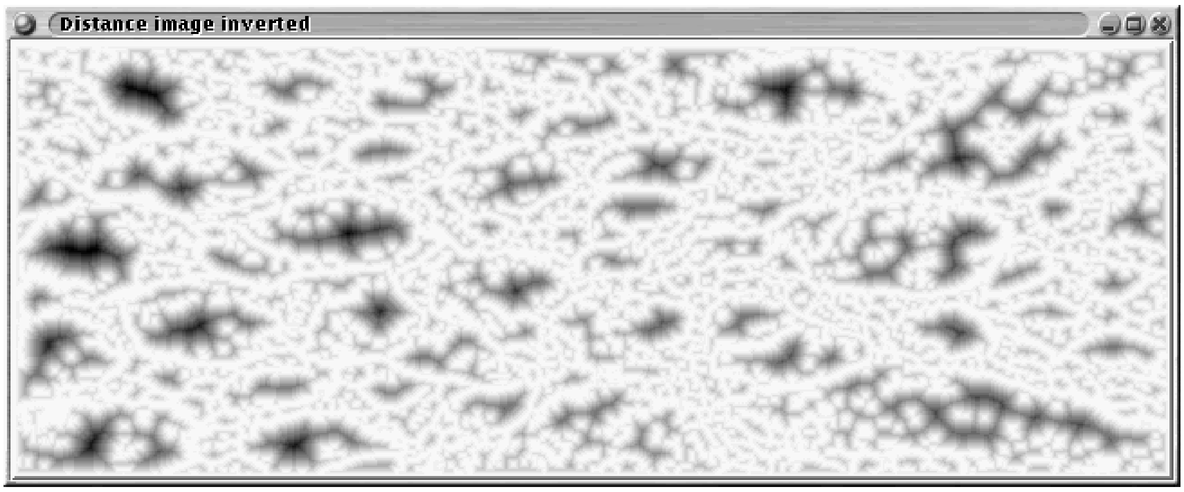


(c) Watershed.

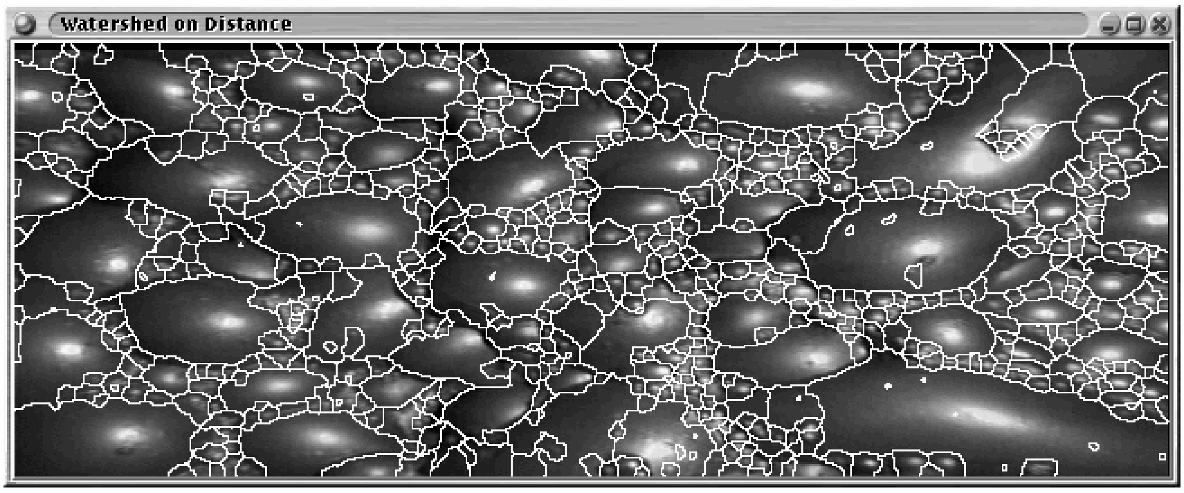
FIGURE 3.3 Watershed on the ridge image.



(a) Ridges.



(b) Distance function, inverted.



(c) Watershed.

FIGURE 3.4 Watershed on the distance function of the ridge image.

3.2 Connected Components

Connected components can be used to not only find a better set of markers for froth image segmentation but also to partially segment an image. Examination of a typical froth image (see figure 3.1(a)) shows, for a singe dominant illumination source, one highlight per bubble. This implies that extracting the domes of an image (see section 4.1) and using these as markers for segmentation is not sufficient — not every “dome” found will correspond to the highlight. For example, a bright dome is more likely to correspond to a bubble than a dark one.

Simply put, connected components filtering tags every region in an image with a number of numerical descriptors, for example, area, average pixel intensity, ratio of major to minor axes; and then removes a region if its tag does not fit some criterion. This can be regarded as a generalisation of a morphological thinning process [65]. Traditionally connected components are generated by iteratively thresholding an image, and extracting the binary connected components. Each binary connected component is associated with the threshold used to generate it [77].

This section examines some of the algorithms in the literature (some of which are optimised for speed), and then examines the application of connected components type algorithms to froth image segmentation.

3.2.1 Algorithms in literature

It should be said that if it is known in advance which properties are required for preserving regions, then as the thresholding progresses the regions can be processed, and the output image incrementally built up. If, on the other hand, an interactive algorithm is required with the precise attributes of each region unknown then a component trees approach is required [77].

Connected components filters can be defined as either morphological openings or thinnings [17]. A morphological filter is an opening if it is anti-extensive, idempotent and increasing. A morphological filter is a thinning if it is idempotent and anti-extensive. An idempotent filter applied many times generates the same output as a single application. An anti-extensive filtered image has a domain (region of support) smaller than, or equal to, the unfiltered image. An increasing morphological filter is one in which the order of regions is preserved, in other words, if A is a subset of B then the filter applied to A is a subset of the filter applied to B ($\Phi(A) \subseteq \Phi(B)$) [63, 64, 65].

Once all the connected components in an image have been extracted and filtered, the surviving connected components must be combined to generate the output image. Traditionally, with the connected components from level sets (iterated thresholding) approach, the output image is simply the point-wise maximum of all the pixels which overlap in the different connected components [17, 30, 140]. This is sometimes called the stacking procedure [17].

The work of Vincent, Breen and Jones, and Jones in processing connected components is examined subsequently.

3.2.2 Area openings - Vincent

Vincent [142, 143, 140] describes an area opening in order to remove connected components of an image. This is a specific case of the attribute openings defined by Breen and Jones [17]. All connected components of area less than some threshold are suppressed. The process is idempotent because once regions with area less than (or equal to) some threshold are removed, iterating the filter has no further effect. The process is anti-extensive because the region of support of the filtered regions is smaller (or the same as) the region of support of the unfiltered regions. The process is increasing (the properties of morphological filters are described in Appendix B) because:

(i) $A \subseteq B$, and both survive the area opening, then $\Phi(A) \subseteq \Phi(B)$ implies $A \subseteq B$.

(ii) $A \subseteq B$, and both are eliminated, then $\Phi(A) \subseteq \Phi(B) \implies \oslash \subseteq \oslash$, which holds. $\oslash$ denotes the empty set. Set theory holds that the empty (null) set is a proper subset of every set [75].

(iii) $A \subseteq B$, and B survives but A does not, then $\Phi(A) \subseteq \Phi(B) \implies \oslash \subseteq B$, which holds.

The case A survives but B does not, cannot happen.

Attribute openings, thinnings and granulometries - Breen and Jones

Breen and Jones [17] generalise the area opening of Vincent (as described in the previous subsection 3.2.2). Breen and Jones describe an attribute based opening on connected components and then for non-increasing criteria define a thinning (by definition a thinning has the same properties as an opening except that it is non-increasing).

Component Trees – Jones

Component trees were proposed by Jones [77] to overcome the limitations of flat filter based connected component approaches. In essence every connected component in an image at every possible thresholding is identified. The root of the tree is the connected component with as many pixels as the image — the connected component formed by including every pixel larger than or equal to zero, for normal greyscale images. The first layer of the tree contains connected components with pixel intensities larger than or equal to one and so on. A particular node has as its children all connected components (at higher thresholds) contained within it. Each node in the tree is labelled with (one or more) attribute signatures which correspond to certain features of the connected components, for example, area (number of pixels), ellipticity, average grey-level and so on. Any node could be marked inactive if its attribute signatures are undesirable. An output image can be generated from the component tree by reversing the threshold decomposition. If each connected component marked as active is given the grey-level corresponding to its level (or threshold used to generate the component) then taking the maximum at each pixel of the output image for every active connected component containing the pixel results in the output image. Jones describes an efficient implementation [76].

3.2.3 Discussion

Connected component filtering for greyscale images can be viewed as applying an operator to an image, finding the regions of support of the non-zero pixels (a thresholding), iterating the operator, and then stacking the regions of support. It should be noted that the intensity of the regions at different iterations are different (for example, if candidate regions are extracted by extracting the level sets of an image). The regions are then combined to yield the output image. This stacking procedure (the usual definition) computes the pointwise maximum of regions generated by successive iterations. This essentially reproduces the pixel values of the pixels which are not filtered out by the connected components operators.

If one can define an index (k) on the operator ($A(I)$, I an image) then as the index increases, $A_k(I) \subseteq A_{k+1}(I)$. For example, if $A_k(I)$ denotes the level set of an image (at level n), then $A_{k+1}(I)$ denotes the level set of an image at a lower level (m, $m \leq n$). Clearly then, the level set at m contains the level set at n. In order to allow for extreme cases, for example to create the root node of a connected component tree, $A_{k_{max}}(I) \subseteq \text{Domain}(I)$. The operator required must then be an anti-extensive operator, by definition, since $A_k(I) \subseteq A_{k+1}(I)$ (see Appendix B).

Several operators are possible given the above. For example, eroding an image using rolling balls, cones and inverted parabola. Each of these structuring elements have a parameter which sets size (or scale) for example, radius, height and radius, and height respectively. The eroded image is then greyscale reconstructed by dilation under the original image. The domes of the image are then found by subtracting the reconstructed form the original image. The domes at a particular scale are then the result or the operator A.

Consider the following example:

(i) An operator is chosen, for example, the rolling ball erosion, greyscale reconstruction and dome extraction process described above.

(ii) The operator is iterated (for example, for increasing diameters of the rolling ball).

(iii) After the erosion is applied the image is thresholded to extract the position of non-zero pixels. This establishes the regions of support of the non-zero pixels.

(iv) Similarly to the connected component tree, a tree can be constructed as follows:

  (a) The root node corresponds to pixels where the erosion has been applied with a zero diameter ball. This ensures the existence of a root node.

  (b) Each successive node corresponds to a connected component in the thresholded image.

  (c) Each level of the tree corresponds to a particular iteration of the operator.

  (d) The parent-leaf relationship of two nodes is set as follows: if a connected component shares pixels at one iteration with a connected component at the previous iteration then a leaf-parent relationship exists between these two nodes.

(v) All the usual processing can then be applied to this tree as in the connected components tree of Jones described above.

In principle, connected components operators may be used to segment and find markers for images. In practice, the computational cost is prohibitive for real and near real-time applications[1] The computational burden lies in the repeated thresholdings and then examining each connected component.

Connected components are not suitable for froth segmentation. At best, a careful selection of features may enable the bubbles to be almost completely extracted. The pixels between the bubbles will not be part of any segmented bubble. In other words, connected component segmentation as described in the literature cannot assign every pixel in an image to a particular bubble. At best, this allows for marker extraction, with a marker based watershed completing the segmentation. Marker extraction and processing is subsequently described in the following chapter (Chapter 4).

[1]Caveat, using current PC's that is.

Chapter 4

Markers

Markers are “seed” regions, or nuclei, for region growing segmentation algorithms. Region growing algorithms first assign a unique label to markers before attempting to label unlabelled pixels in the image. The pixels neighbouring the markers, or already labelled pixels, are assigned a label according to some criterion (see [13, 38, 141] for examples of some region growing algorithms). Marker processing is related in some sense to image denoising. Finding an optimum set of markers from a noisy image can lead to good segmentation. Traditionally, watershed segmentation has been used for delineating the bubbles within a froth image [45, 156, 157].

Marker based segmentation is only as good as the set of markers used. The problem of segmenting an image can then be changed into the problem of finding good markers. This is complementary to a denoising process since a good marker extraction process should reject any marker induced by noise. This chapter proceeds by first examining different methods of extracting markers from an image, and then shows how those markers may be processed to reduce over-segmentation.

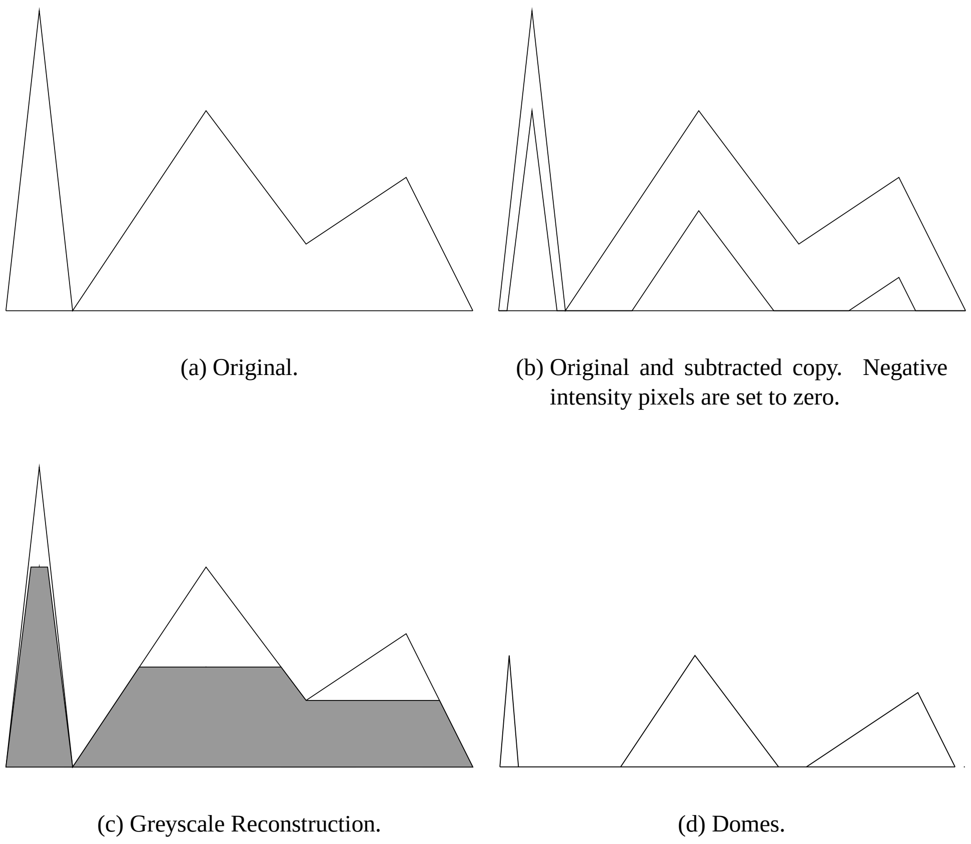
4.1 Extracting the markers

The markers of an image may be informally defined as regions which indicate centres of areas of interest within the image. The problem of finding markers is at least as difficult as general segmentation, since finding all the markers of an image makes segmentation using a region growing scheme trivial.

Under certain lighting conditions, namely, a single powerful lightsource close to the camera which dominates all other light sources, the highlights — or regions of maximum intensity — are indicators (markers) of bubbles. There is only one highlight per bubble (if the lighting has been carefully arranged). Figure 3.1(a), which represents a typical froth image, illustrates this. The highlights can then be used as markers for bubbles. A simple thresholding is not sufficient to extract the markers since the highlight intensity varies from bubble to bubble. The method used subsequently is morphological dome extraction.

Dome extraction proceeds as follows: the original image is modified in some way (for example, shifted down in intensity). (Possible modifications are examined subsequently.) The modified image is grey-level reconstructed by dilation. The grey-level reconstructed image is subtracted from the original to form the domes [144]. Greyscale reconstruction by dilation is essentially a form of conditional dilation generalised to greyscale images. By an accident of terminology in the greyscale reconstruction process, a marker image is introduced. This should not to be confused with the bubble markers used for segmentation.

The original image is referred to as the mask image. A marker image (constrained to lie on or below the mask image) is continually dilated. After each dilation, the marker is modified as follows: if the intensity of a marker pixel exceeds that of the mask pixel at the same position, the marker pixel is set to the mask pixel intensity. The modified marker image becomes the marker image for the next dilation. The process continues until the marker image no longer changes. For visualisation purposes the domes' intensities are rescaled to their original values. To simplify the resultant image, negative pixels are set to zero. The process is illustrated in figure 4.1.


(a) Original.

(b) Original and subtracted copy. Negative intensity pixels are set to zero.

(c) Greyscale Reconstruction.

(d) Domes.

FIGURE 4.1 Dome extraction.

The process of extracting domes may be regarded as extracting the regional maxima of an image. The markers are then used in a marker based watershed to extract the watershed regions of the inverted image. Usually though, further processing of the markers is required.

Watershed segmentation floods an image from the minima upwards in intensity. This requires the image to be inverted prior to segmentation to ensure that flooding proceeds from the highlights. The regional maxima found by the dome extraction process then become regional minima for the watershed. Only these regional minima are flooded from (see [12, 13, 141] and section 3.1 for detailed descriptions of watershed segmentation).

It should be pointed out that the same domes can be extracted using reconstruction by erosion on the inverted image. Here a mask image, constrained to be larger or equal to the marker image, is conditionally eroded. Conditional erosion ensures the eroded mask is everywhere larger than or equal to the marker, or original image. Conditional erosion tests each pixel on the mask image and ensures that if the mask pixel is less than the marker pixel in the same position, the mask pixel intensity is set to that of the marker pixel. The mask image is then set to the modified mask. The process continues until the mask image stabilises. The inverted domes (regional minima) are found by subtracting the original image from the conditionally eroded mask.

Noise leads to a number of spurious minima forming, and to larger valid markers being broken up. Minimising the effects of over-segmentation is examined in section 4.4.

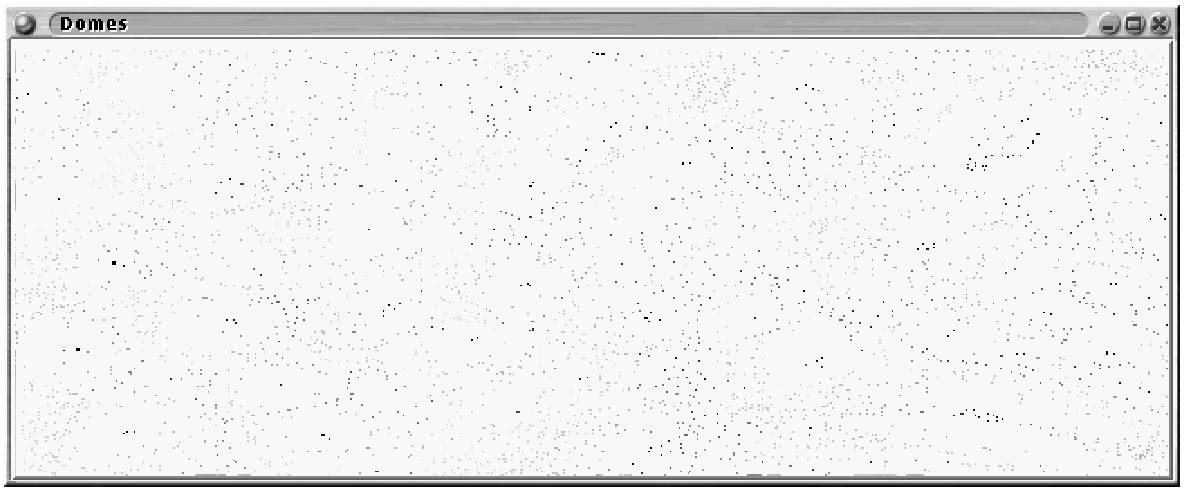
4.2 Markers by subtraction

Here the modified image referred to above is created by subtracting a constant from the original image. As an optimisation, pixels less than the minimum pixel intensity of the original image are set to the minimum intensity.

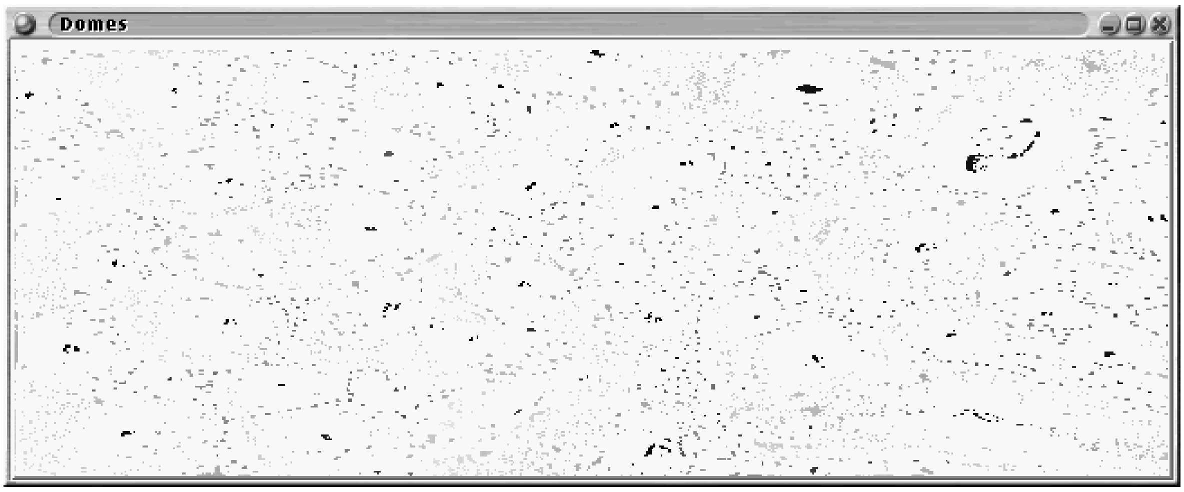
A difficulty with the dome approach to marker generation is that the shift level decides the extent of the markers. The larger the shift value the more the domes (and hence markers) tend to be merged. This is is illustrated in figure 4.2 following. A further difficulty is that the shape of the highlights is not exploited for their extraction. The large shift required for reasonable dome formation is not related to any physical parameter within the froth (for example, size of the small bubbles in pixels). This is especially the case if the image has been enhanced in any way (for example histogram equalisation). It should be emphasised that the dome images (for printing purposes) are inverted images. Table 4.1 shows the decrease in the number of watershed regions as the shift increases.

TABLE 4.1 Decrease in the number of watershed regions as the shift value increases.

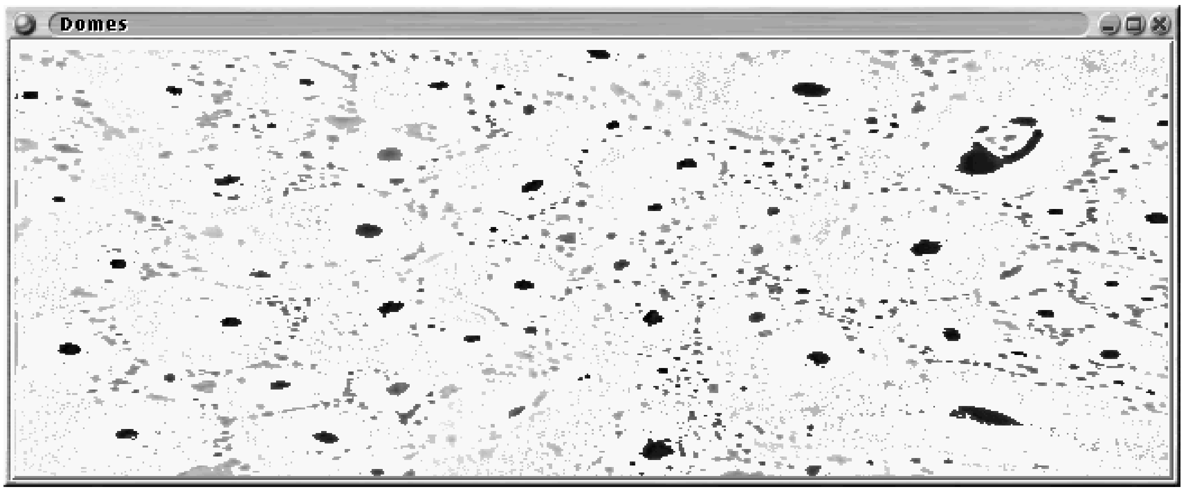
Shift	# Watershed Regions
1	3505
10	3345
20	3244
50	3070
100	2808
150	2483
200	1241



(a) Domes for shift of 1.



(b) Domes for shift of 10.



(c) Domes for shift of 50.

FIGURE 4.2 Domes merging as shift level increases.

4.3 Other methods of extracting the markers

Anti-extensive morphological filters could be used to find the marker image for the dome extraction process (i.e. greyscale reconstruction by dilation). Extensive morphological filters could be used to find regional minima via greyscale reconstruction by erosion (see Appendix B for a brief introduction to morphology). Application of an anti-extensive filter to an image results in an image which is everywhere less than or equal to the original [124, 125].

Examples of anti-extensive morphological filters include flat erosions, openings and clos-openings[1]. Examples of extensive morphological filters include flat dilations, closings and open-closings[2].

Only erosions will be considered for greyscale reconstruction by dilation. Openings will tend to merge close markers (a result of the dilation of the markers). This is especially undesirable for large structuring elements, markers relatively distant may be merged. Clearly cascaded filters will A similar argument may be constructed for closings and open-closings for greyscale reconstruction by erosion.

4.3.1 Using morphological filters to extract Domes

The Dome extraction process relies on greyscale reconstruction by dilation. An image was reconstructed by dilation from the marker, and subtracted from the original to form the domes. This proceeds as previously described, except that instead of creating the marker by shifting the intensities of the image downwards, the image is morphologically eroded. Any morphological filter could be used, provided the resultant image is everywhere less than or equal to the original. In other words, the morphological filter must be anti-extensive [65]. This is a consequence of the way in which the greyscale reconstruction by dilation proceeds, the reconstructed image must be less than or equal to another image which constrains the reconstruction (see [144] and section 4.1).

Two morphological filters are considered: flat erosion with a disc structuring element, and a rolling ball erosion. These filters are applied to the original image to form the greyscale reconstruction by dilation marker image. The segmentation process then proceeds normally.

Flat erosion

Table 4.2 shows the change in the number of watershed regions as the radius of the disc used for flat erosion of the original image increases. Figure 4.3 shows the changes in the domes and hence watershed of the original image, again as the radius of the disc for flat erosion increases. Note that as the radius increases markers are merged. A large radius then risks merging the markers corresponding to close bubbles. Radius 2 shows the best tradeoff between finding markers and merging close markers. Notice the number of small domes one or two pixels across, these are likely to be noise

[1]A closing followed by an opening.

[2]An opening followed by a closing.

induced because inspection of the image does not show any bubble with highlights this small in the corresponding positions.

TABLE 4.2 Decrease in the number of watershed regions as the radius of the structuring element for flat erosion increases.

Radius	# Watershed Regions
1	3285
2	2991
3	2818
4	2728
5	2650

Rolling ball erosion

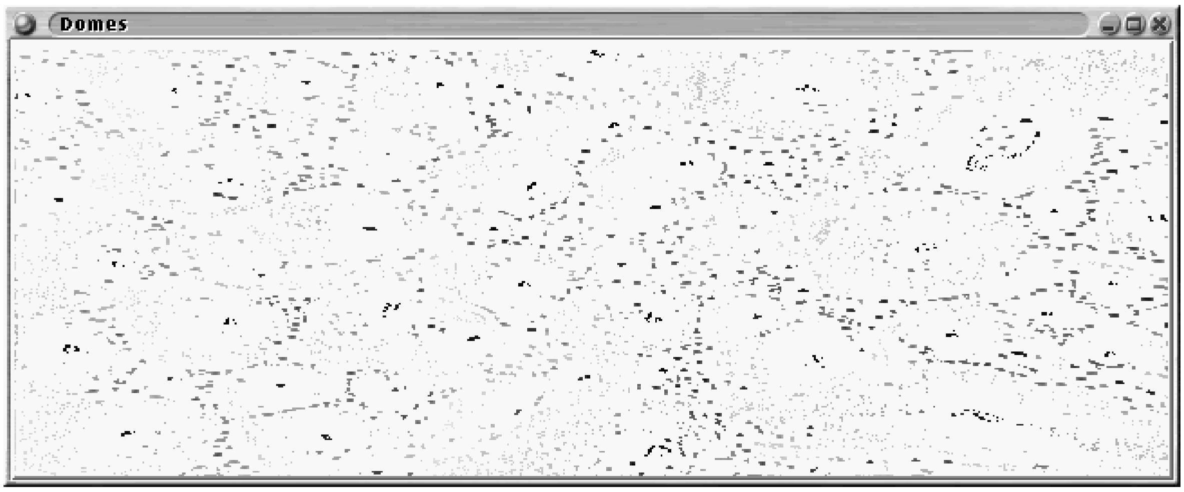
An alternative is using a rolling ball as the structuring element. The advantage is that the ball smoothes the image. Rolling balls explicitly set the size (in pixels) of the regional maxima removed by the erosion. This implies that the larger highlights appear as larger domes.

Table 4.3 shows the decrease in the number of watershed regions as the radius of the rolling ball increases. Figure 4.4 shows the changes in the domes, and hence the watershed, as the radius of the rolling ball increases. Note that the domes are larger than for the subtracted image case. Note that as the radius of the rolling ball increases, close markers are merged. Also note that the large markers correspond to large highlights. A large radius then risks merging the markers corresponding to close bubbles. Radius 2 shows the best tradeoff between finding markers and merging close markers.

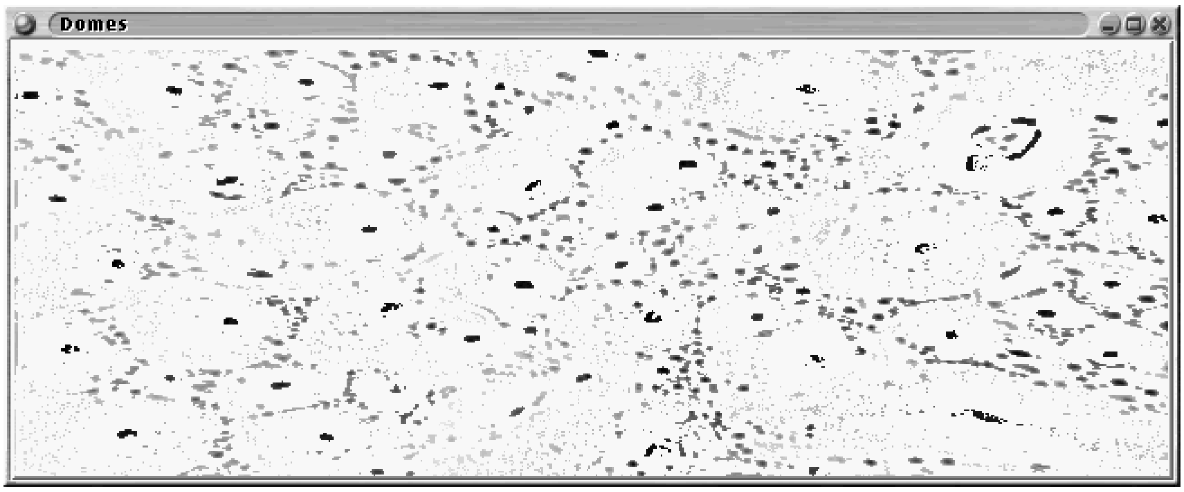
Note that in all the dome images there are a number of small domes one or two pixels across. These are likely to be noise induced domes and it is quite obvious that they will needlessly complicate the segmentation. Section 4.4.3 describes how the small marker problem may be alleviated. Larger domes also exist which are bright (in the original image these will be dark maxima). These are then unlikely to be bubble highlights and contribute to the over-segmentation of large bubbles. Section 4.4.1 describes how the dark marker (in the original image) problem may be alleviated.

TABLE 4.3 Decrease in the number of watershed regions as the radius of the rolling ball for marker extraction increases.

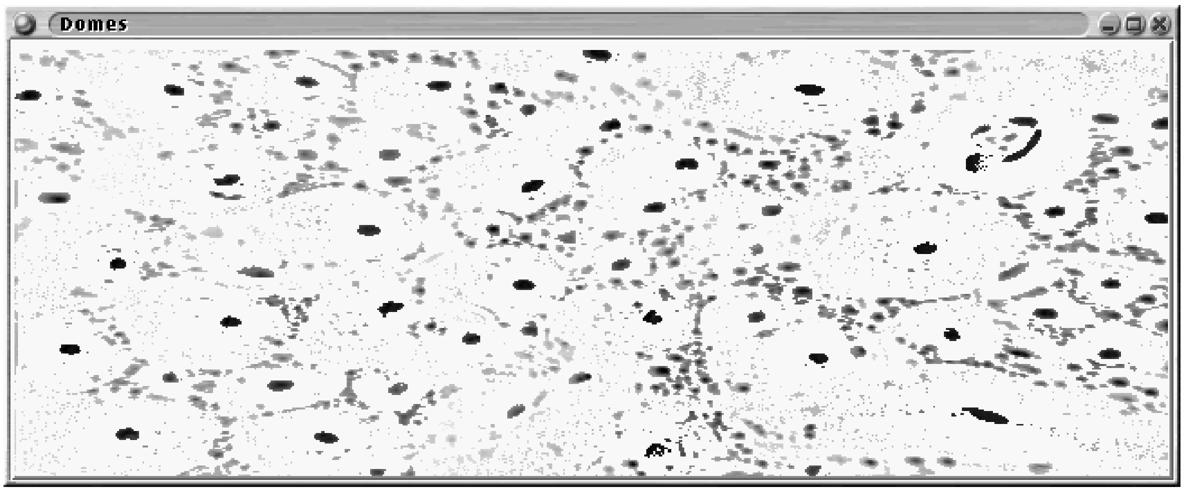
Radius	# Watershed Regions
1	3165
2	2864
3	2737
4	2657
5	2569



(a) Domes – flat erosion, radius 1.

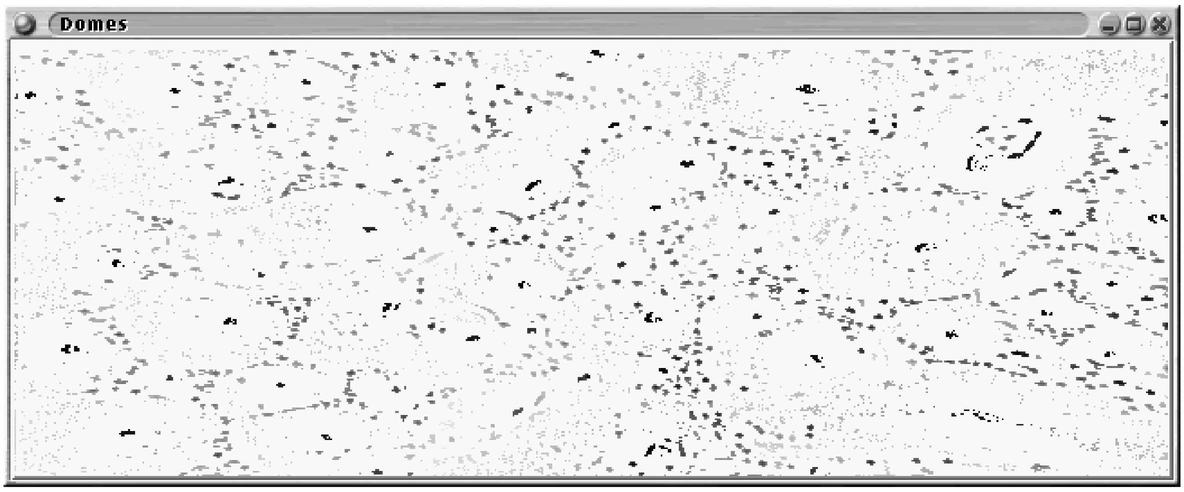


(b) Domes – flat erosion, radius 2.

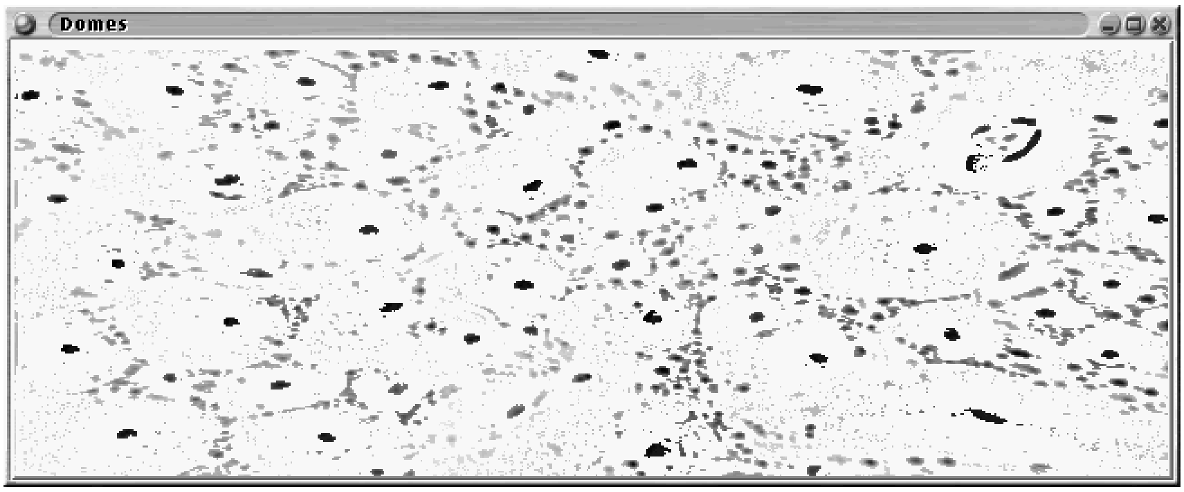


(c) Domes – flat erosion, radius 3.

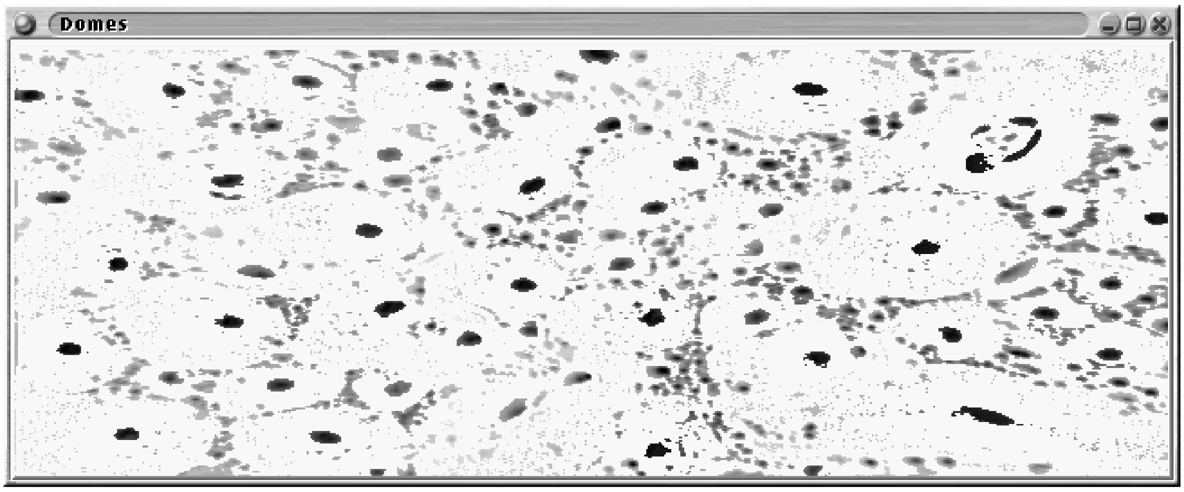
FIGURE 4.3 Changes in domes and watershed as flat erosion progresses.



(a) Domes – rolling ball erosion, radius 1.



(b) Domes – rolling ball erosion, radius 2.



(c) Domes – rolling ball erosion, radius 3.

FIGURE 4.4 Changes in the domes and watershed as the radius of the rolling ball increases.

4.3.2 Connected component marker extraction

It has been established that the markers correspond to the highlights of the image. Examination of these highlights show them to be usually circular or elliptical, and not of large extent. The attributes used to then segment the highlights from the standard froth image will be ellipticity and area (in pixels). A simple circularity measure will be used [119]:

$$\text{Circularity} = \frac{(\text{Perimeter})^2}{4\pi\,\text{Area}} \tag{4.1}$$

The connected components are extracted by successive thresholdings, that is, the level sets of the image are extracted. The markers are selected by eliminating the connected components of the image which have circularity larger than 2. Note that small clusters of pixels have high perimeter to area ratio (circularity). This implies that rejecting all connected components where the circularity measure is high. Alternatively, thin regions have a low circularity measure. This implies rejecting regions with circularity measure somewhat less than 1, as a threshold 0.5 was chosen here. Elliptical regions are retained by allowing for a circularity measure larger than 1.

As an optimisation, only the level sets of the image larger than 100 are considered. The level sets of an image are the formed by thresholding an image with successively larger thresholds [17]. A simplification sets the intensity of pixels larger than, or equal to, the threshold to the threshold. Taking only the level sets larger than 100 biases the process against large and dark connected components. A further optimisation eliminates all connected components larger than 2000 pixels. This attempts to ensure that small markers are not merged, and biases the process against overly large markers.

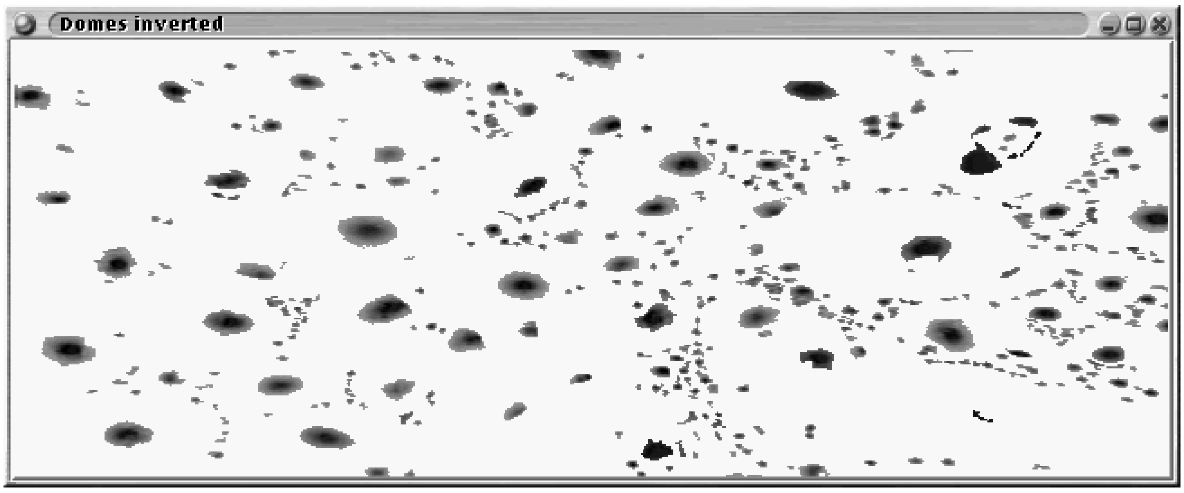
Figure 4.5 shows the domes, (once thresholded these then become the markers for segmentation) markers and the final watershed segmentation based on these markers. Note that the segmentation is complicated by the presence of small spurious domes. These could be eliminated using the area opening described subsequently.

4.4 Post-processing the markers

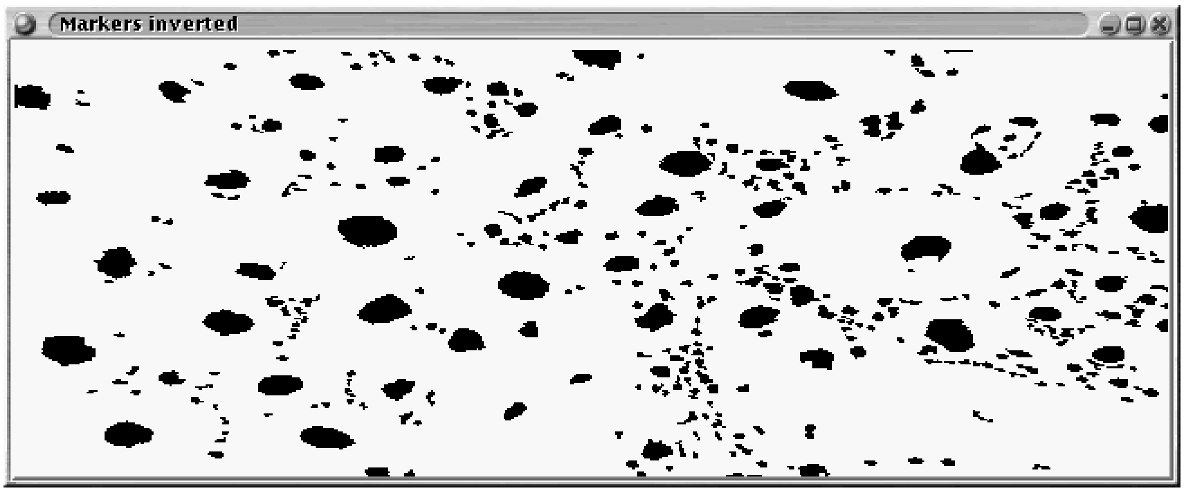
Coupled with a denoising technique, post-processing of the markers limits over-segmentation. The following post-processing techniques will be examined: removal of dark markers (markers associated with dark domes), and merging of close markers (this defragments fragmented markers). The process of examining markers and then removing those whose attributes (area and average intensity for example) fall short is referred to as connected component filtering. The markers are generated by rolling ball erosion of radius 2. These markers are then further processed.

4.4.1 Removing dark markers

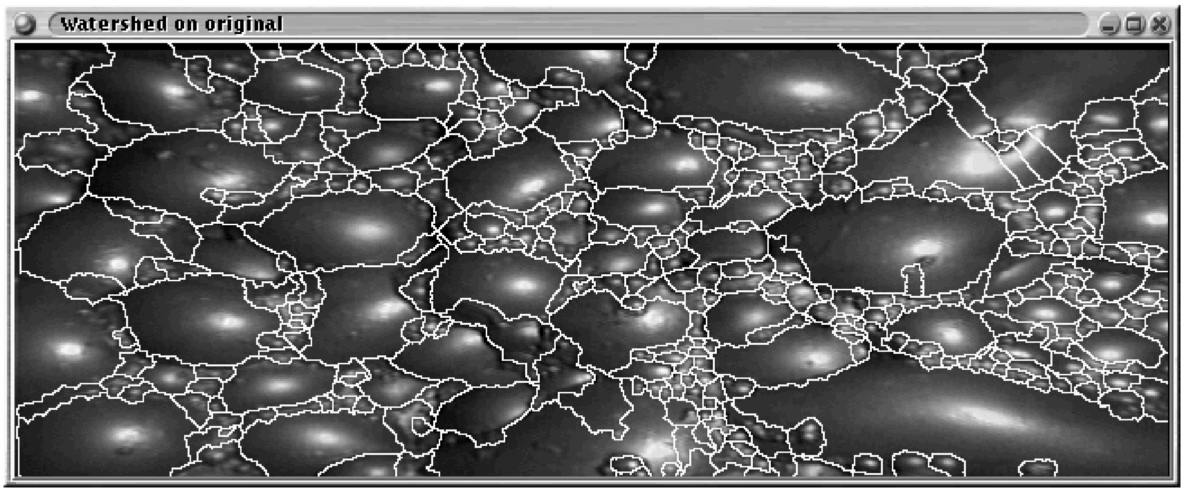
For froth images, highlights are used as markers (the regional maxima of the image (the domes) are extracted and thresholded to form the markers). This suggests discarding any dome whose average



(a) Domes extracted.



(b) Markers.



(c) Domes extracted.

FIGURE 4.5 Connected component marker finding

(or maximum) grey level is low.

The net effect of removing dark markers is a reduction in over-segmentation. A difficulty is the selection of the threshold. Figure 4.6 shows the variation in the number of markers and segmentation as the threshold varies. The best tradeoff between preserving small bubble markers and removing spurious markers appears at 0.4 $\times$ *Maximum Grey Level*. Table 4.4 shows the decrease in the number of markers, and hence, watershed regions, as the threshold increases.

TABLE 4.4 Decrease in the number of watershed regions as the remove dark marker threshold increases.

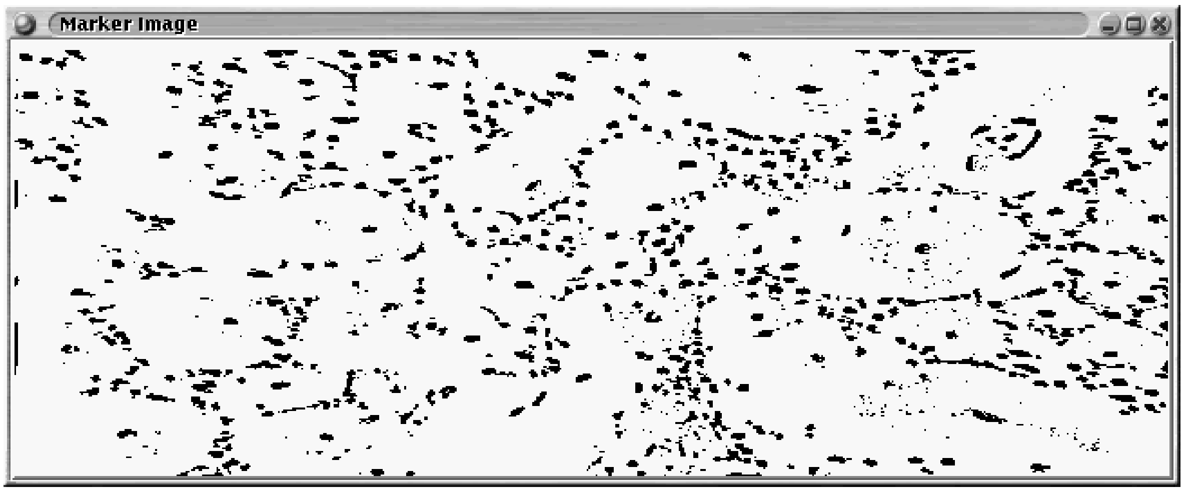
Multiplier	# Watershed Regions
0.1	2945
0.2	2160
0.3	942
0.4	587
0.5	397
0.6	270
0.7	170
0.8	111

4.4.2 Merging close markers

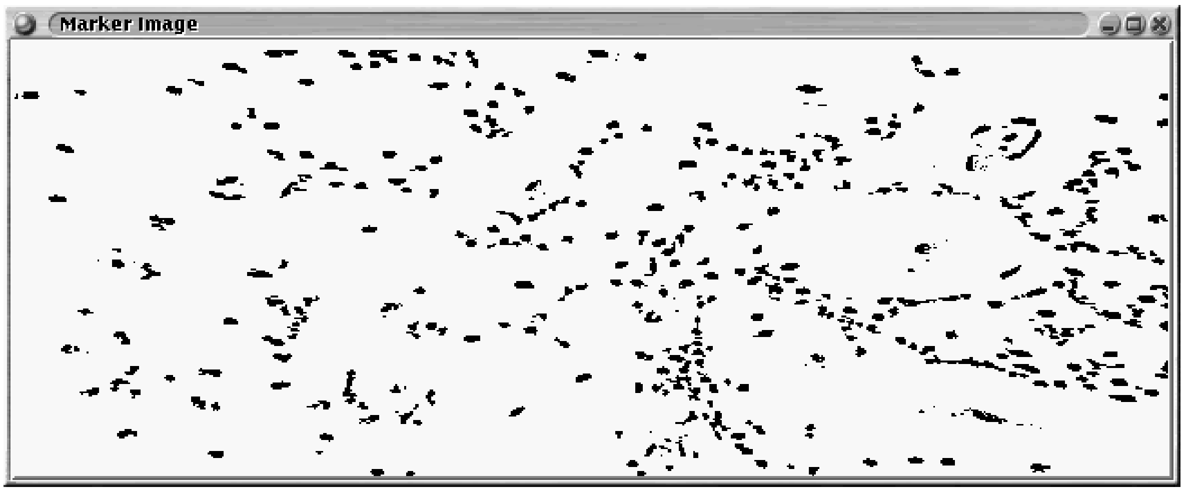
The next problem is that of a large marker being fragmented – a possible remedy is morphological dilation or closing of the markers. In order to minimise the effect of merging dark markers into markers more likely to be correct, these dark markers are removed. All dark markers less than 0.4 $\times$ *Maximum Grey Level* are removed prior to the merging stage. This threshold is suggested in the previous subsection (subsection 4.4.1).

A difficulty lies in the selection of the shape and width of the structuring element. The distribution of markers across an image cannot be predicted, which implies that the distance or direction from a marker to its closest neighbour cannot be predicted. This implies the use of a circular structuring element (as would be demanded by isotropy in any case). The width of the structuring element is constrained by the tendency to merge small bubbles and the reduction in the over-segmentation. Careful consideration chooses morphological closing as the merging operation. Closing minimises the potential situation for small bubbles and large radii structuring elements that the marker may be larger than the actual bubble itself.

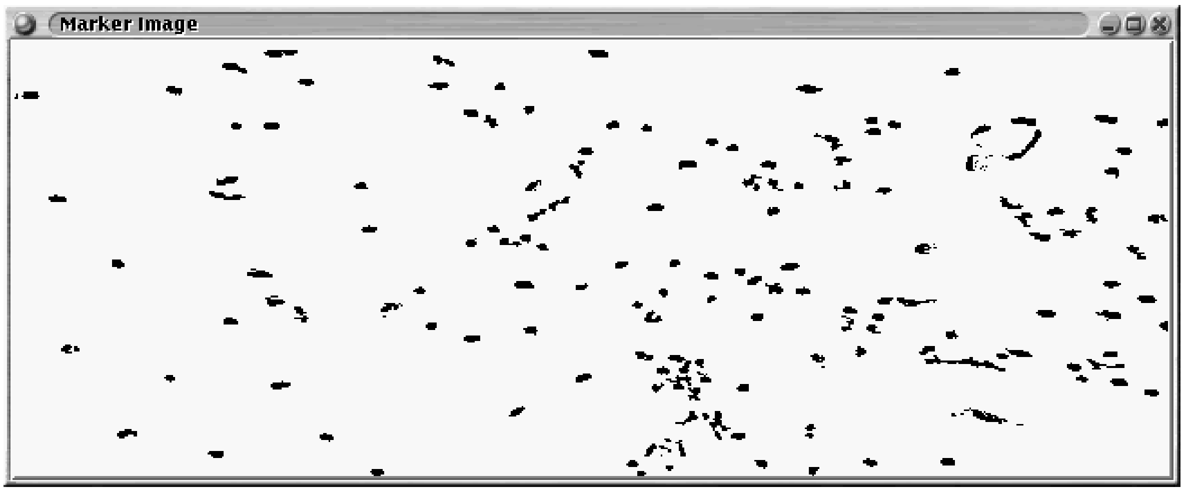
Table 4.5 shows the decrease in the number or watershed regions as the width of the structuring element increases. Figures 4.7 shows the change in the markers and the concomitant change in the watershed as the radius of the closing structuring element increases. Note the tradeoff between the correct merging of fragmented markers and the undesirable merging of small bubble markers.



(a) Multiplier 0.3 – Markers.

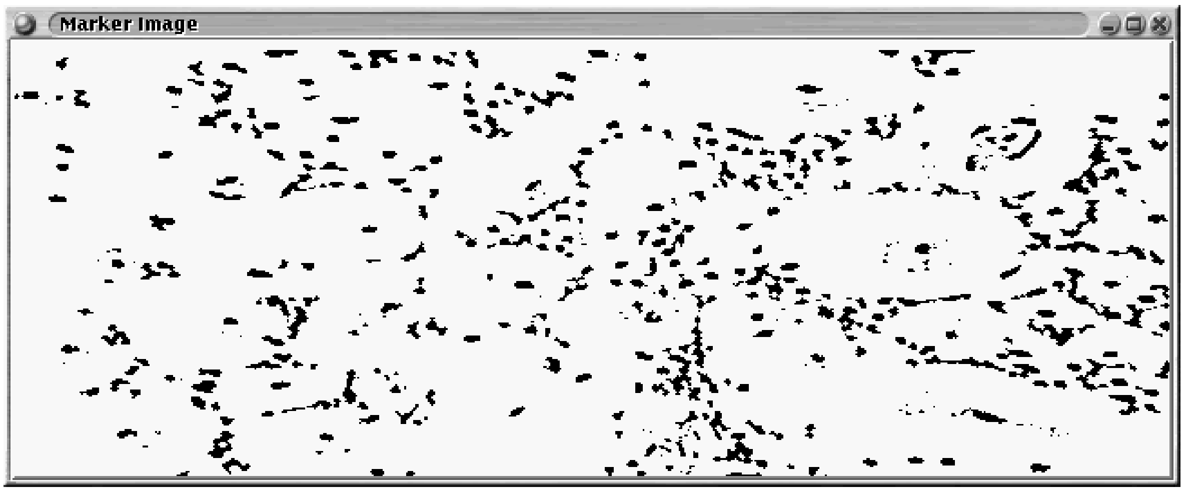


(b) Multiplier 0.5 – Markers.

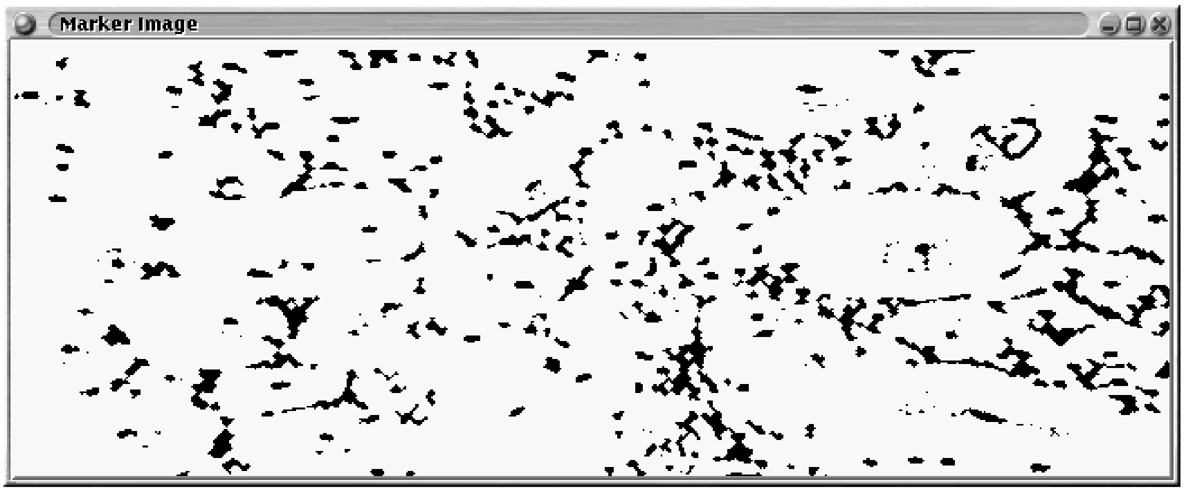


(c) Multiplier 0.7 – Markers.

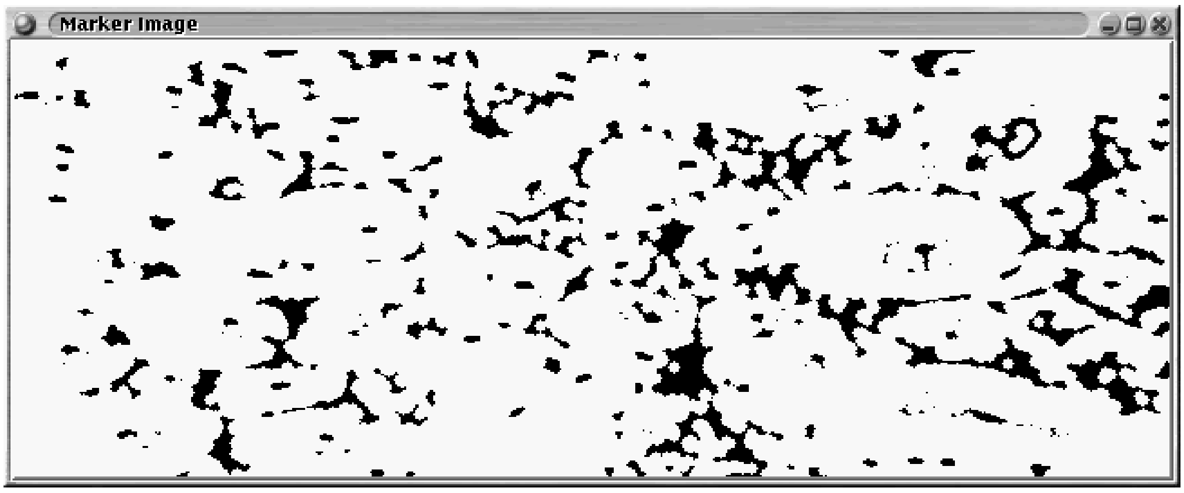
FIGURE 4.6 Variation in the markers as the remove dark marker threshold increases.



(a) Marker image – closing, disc radius 1.



(b) Marker image – closing, disc radius 2.



(c) Marker image – closing, disc structuring element radius 3.

FIGURE 4.7 Change in segmentation and markers as markers closed by increasing radii structuring elements.

TABLE 4.5 Decrease in the number of watershed regions as the radius of structuring element increases for marker closing.

Radius	# Watershed Regions
1	460
2	333
3	244
4	187
5	150

4.4.3 Removing small markers

Examination of the marker images show a number of small markers (a few pixels in extent). Since noise markers are likely to be small and of small extent (noise samples have low correlation) any small marker is likely to be noise induced. This situation regarding small markers is especially visible in the rolling ball extraction of markers. Morphological opening of the marker image provides the best tradeoff between preserving the extent of large markers and removing small markers. An alternative to morphological filters is that of connected components opening.

In theory, any anti-extensive morphological operator may be used to remove small markers. A second consideration is whether this operator is expected to, as far as possible, preserve the extent of the remaining markers. An anti-extensive operator is likely to fragment markers. This could be beneficial to separate small bubble markers which have been merged by the marker extraction process. However, since the extent of the small bubble markers are set by the marker extraction process, it is necessary to prevent fragmentation of markers. This suggests a conditional dilation after markers are removed.

Connected components opening for removing small markers, for the purposes of the current discussion, may be defined as follows. The number of pixels in each marker is counted and compared to a threshold. If a marker has fewer pixels than allowed by the threshold the marker is removed, otherwise the marker is preserved. This process allows for perfect maintenance of the preserved markers' extent. This does not allow, however, for the fragmentation of merged bubbles as allowed by the morphological techniques.

For illustrative purposes the removal of small markers is performed on an markers extracted via rolling ball erosion of radius 3. Although some of the small bubbles are merged in the segmentation (see figure 4.4(c)), this was chosen to explore the marker separation properties as well as the reduction in over-segmentation due to small marker removal via anti-extensive morphological filtering. The process can be quite easily repeated for greyscale reconstruction by dilation marker formation using subtraction or flat erosion of the original image.

Erosion of markers

Table 4.6 shows the decrease in the number of watershed regions as the radius of of the structuring element increases. Figure 4.8 show the cleaned up marker images for various disc radii used. It is clear that markers are potentially fragmented by the erosion. This is especially visible for radii one and two.

TABLE 4.6 Decrease in the number of watershed regions as the radius of structuring element increases for marker erosion.

Radius	# Watershed Regions
1	578
2	318
3	162
4	21

Erosion of markers followed by conditional dilation

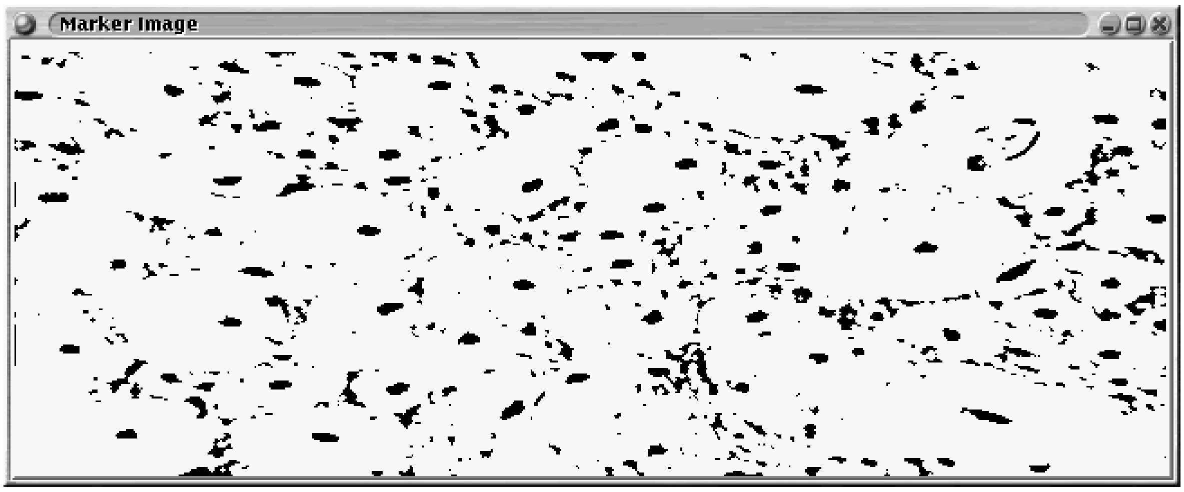
This process erodes the markers as before, except that fragmentation of the markers is then prevented by conditional dilation. Table 4.7 shows the decrease in the number of watershed regions as the width of the disc used as structuring element increases. Figure 4.9 shows the markers for discs of several radii. It is clear here that surviving markers do so on the basis of shape and size.

TABLE 4.7 Decrease in the number of watershed regions as the radius of structuring element increases for marker erosion.

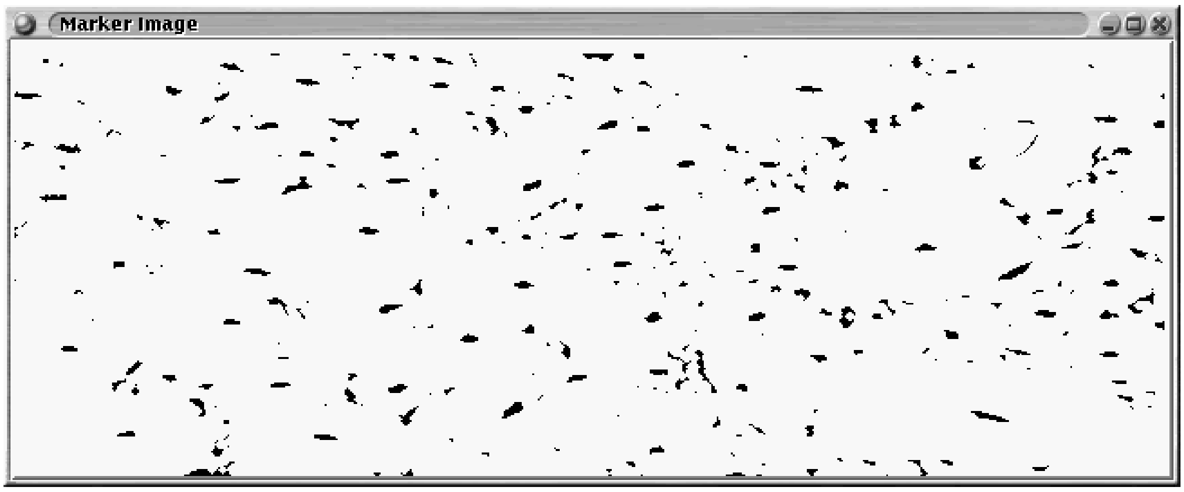
Radius	# Watershed Regions
1	443
2	234
3	124
4	17

Opening of markers

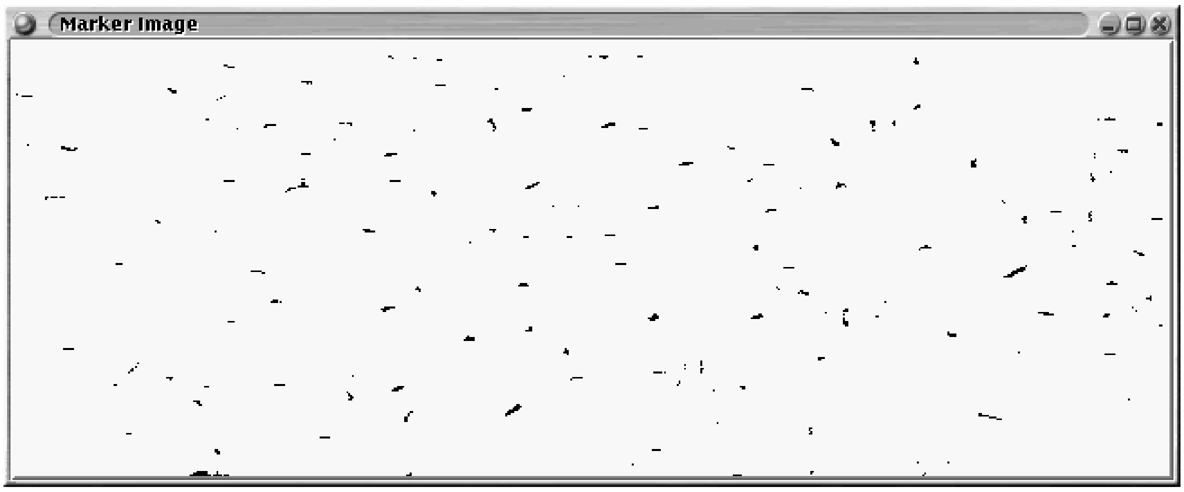
Table 4.8 shows the decrease in the number of watershed regions as the radius of the structuring element increases for opening of the markers. Figure 4.10 show the opening in the marker images for increasing radii of disc used as structuring element. A difficulty is that this process changes the shape of the markers.



(a) Markers – eroding radius 1.

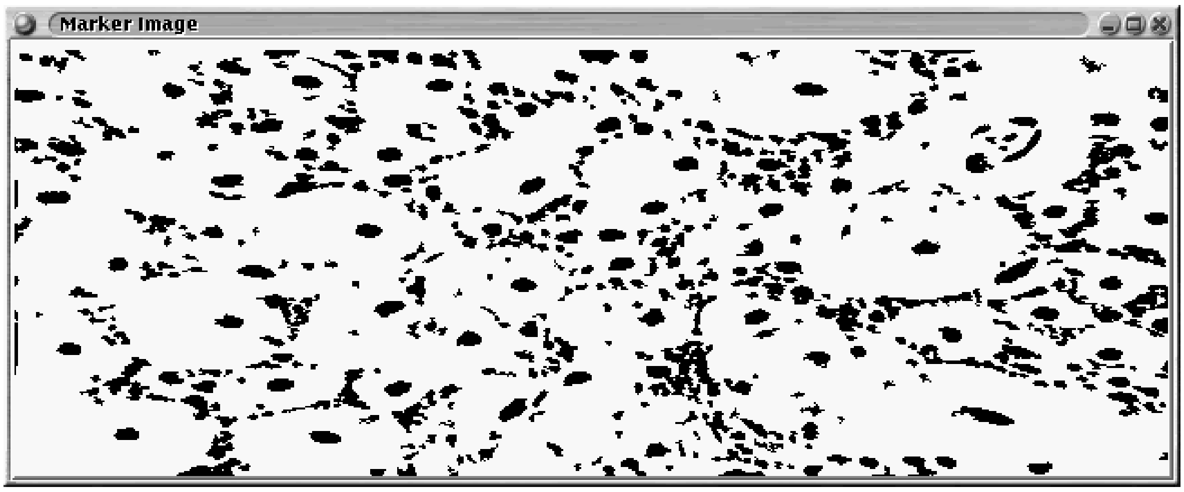


(b) Markers – eroding radius 2.

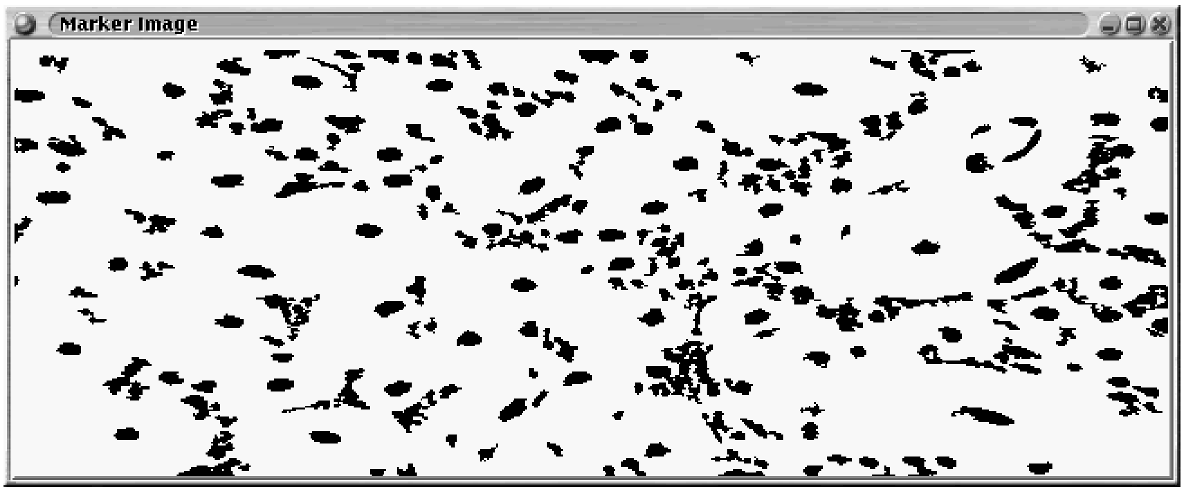


(c) Markers – eroding radius 3.

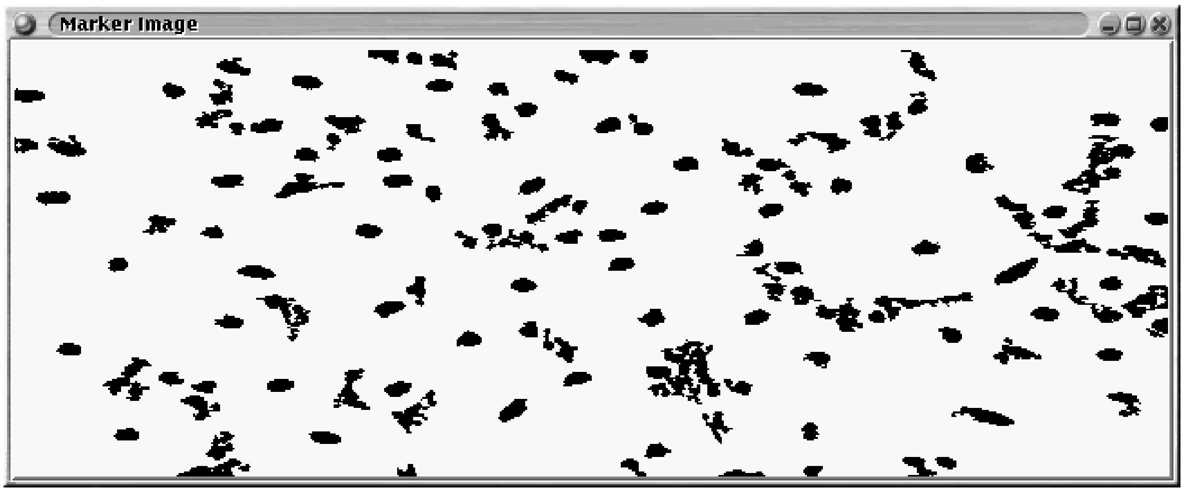
FIGURE 4.8 The domes and watershed segmentation after eroding the markers.



(a) Markers – erosion, radius 1, followed by conditional dilation.

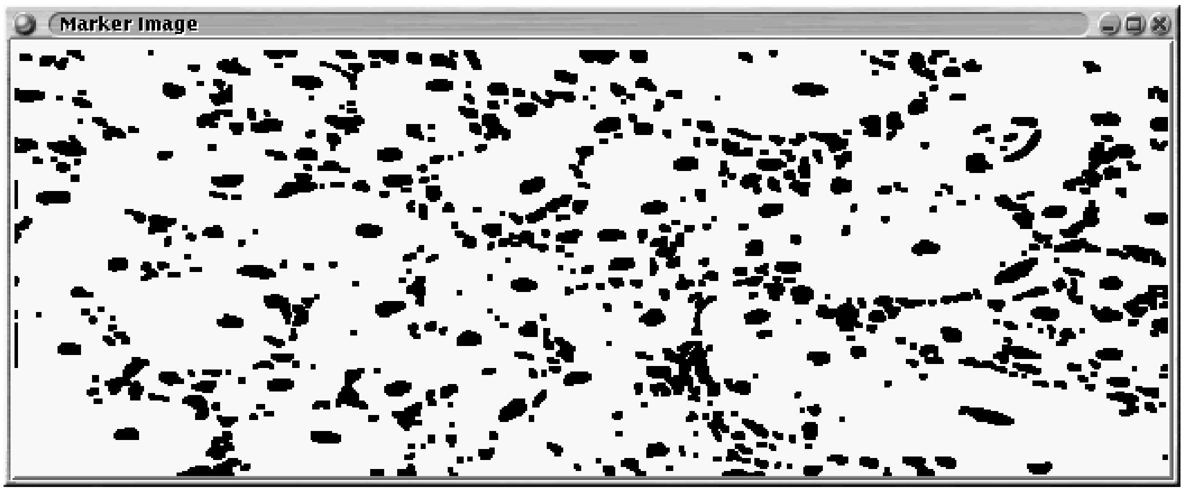


(b) Markers – erosion, radius 2, followed by conditional dilation.

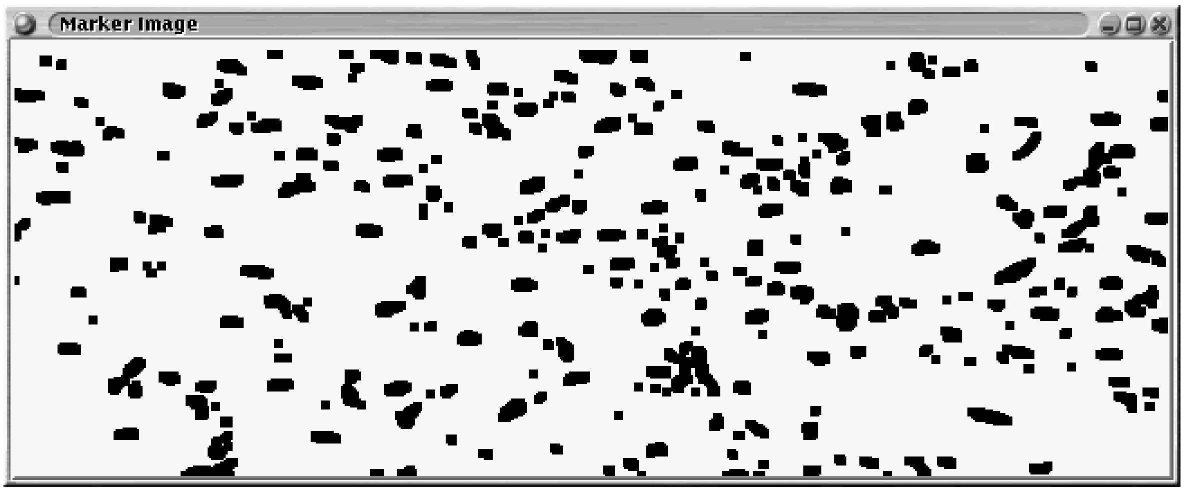


(c) Markers – erosion, radius 3, followed by conditional dilation.

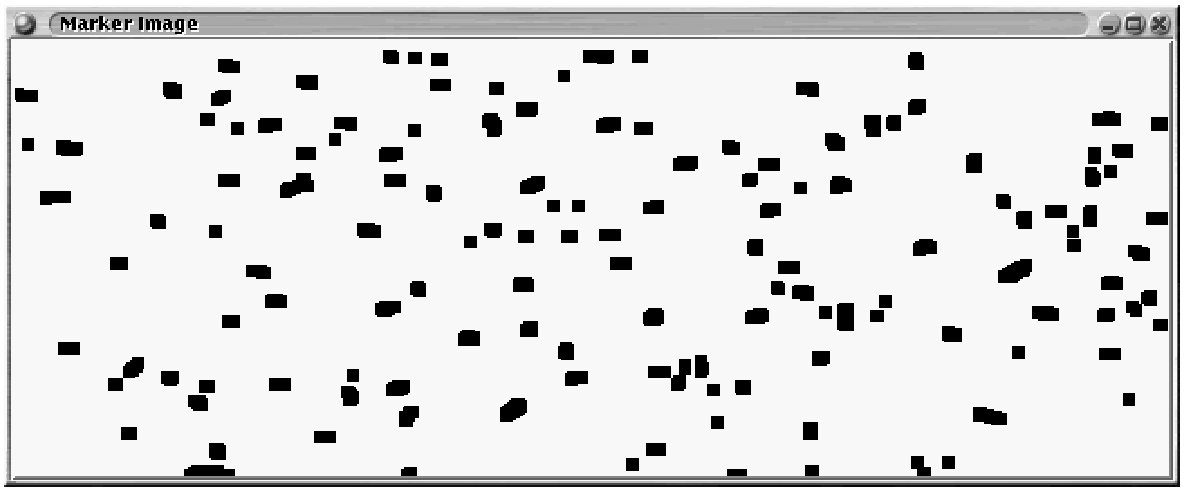
FIGURE 4.9 The domes and watershed segmentation after eroding the markers.



(a) Markers – marker opening, radius 1.



(b) Markers – marker opening, radius 2.



(c) Markers – marker opening, radius 3.

FIGURE 4.10 The domes and watershed segmentation after opening the markers.

TABLE 4.8 Decrease in the number of watershed regions as the radius of structuring element increases for marker opening.

Radius	# Watershed Regions
1	455
2	250
3	134
4	17

Connected Components area opening of markers

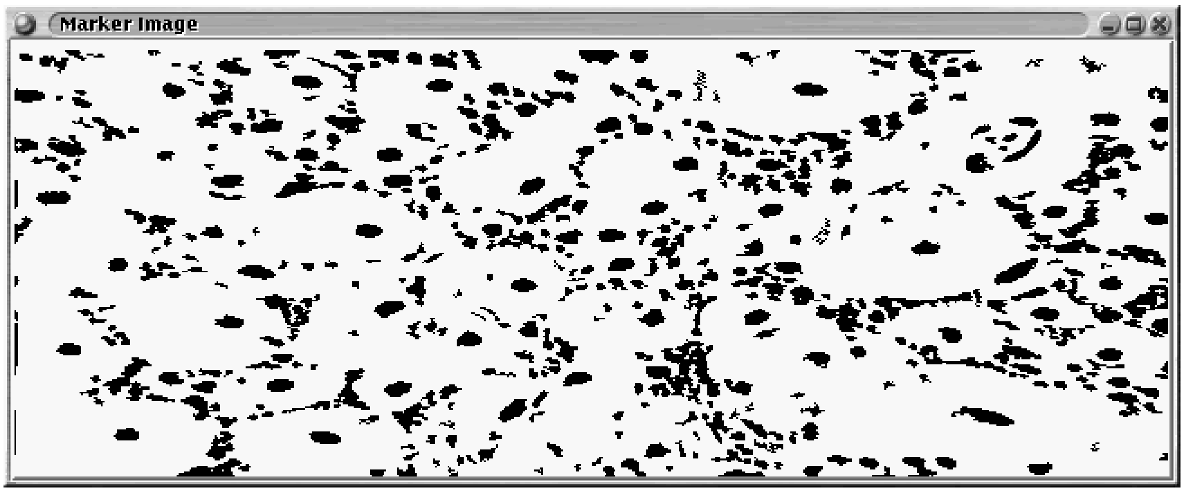
Essentially the area (in pixels) of each marker is compared to some threshold, if lower then the marker is suppressed, otherwise the marker is preserved. Table 4.9 records the decrease in the number of watershed regions (markers) as the threshold increases. Figure 4.11 records the domes and associated watershed regions as the threshold increases. The threshold is selected by computing the area (rounded up) of discs of increasing radii from one onwards.

TABLE 4.9 Decrease in the number of watershed regions as the threshold increases for connected component area opening.

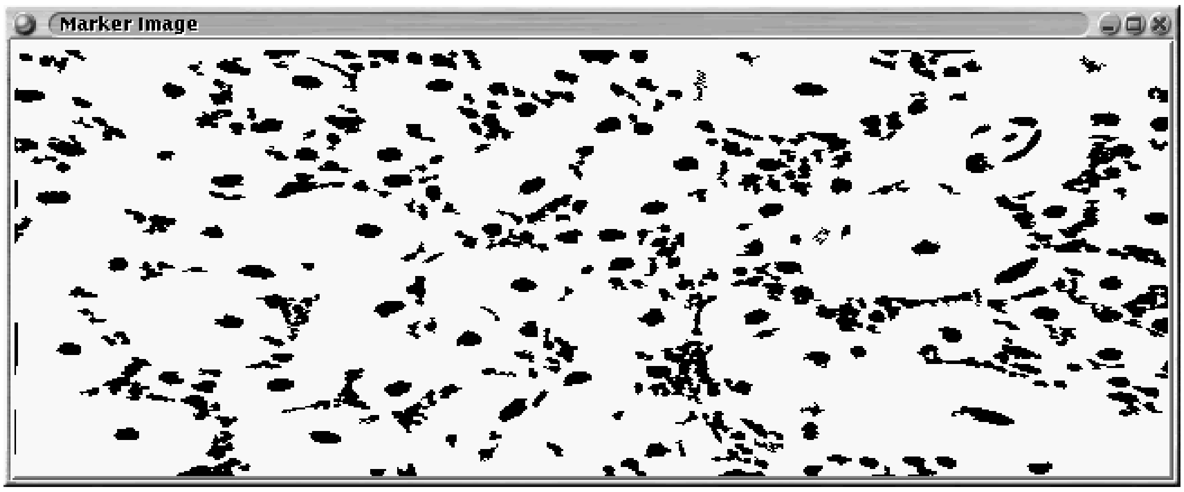
Disc Radius	Threshold (Area)	# Watershed Regions
1.00	9	438
1.87	11	403
2.00	13	377
2.59	21	297
3.00	29	249
4.00	50	166

4.5 Discussion

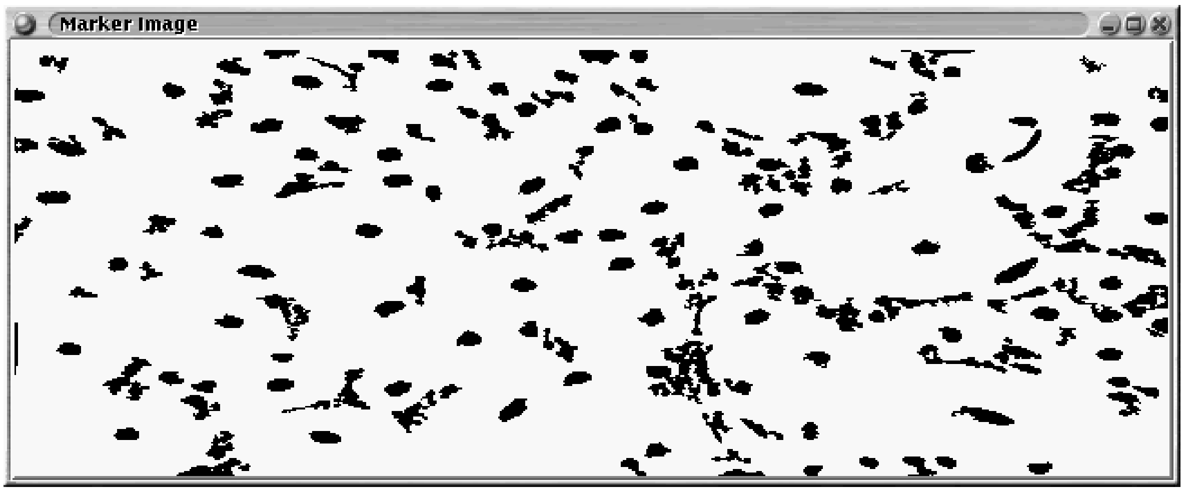
Care in extracting and processing the markers of an image can reduce over-segmentation. Examination of some marker extraction techniques showed that eroding the image before reconstruction, is better than subtracting before reconstruction. While the connected component approach produces good markers it is computationally intensive. There is an explicit relationship between the scale, or width, of the domes extracted by the erosion and greyscale reconstruction process, and the radius of the rolling ball used to erode the image (see sub-subsection 4.3.1). A relationship between scale and shift parameter, on the other hand, does exist, but is not obvious and is image dependent (see section 4.2). Marker extraction should be followed by removal of dark markers (dark in the sense that the maximum grey-level of the marker when taking corresponding pixels on the original, or denoised, image is low). The threshold for removal of dark markers was found to be 0.4 of the maximum image



(a) Markers – area opening, threshold 9.



(b) Markers – area opening, threshold 21.



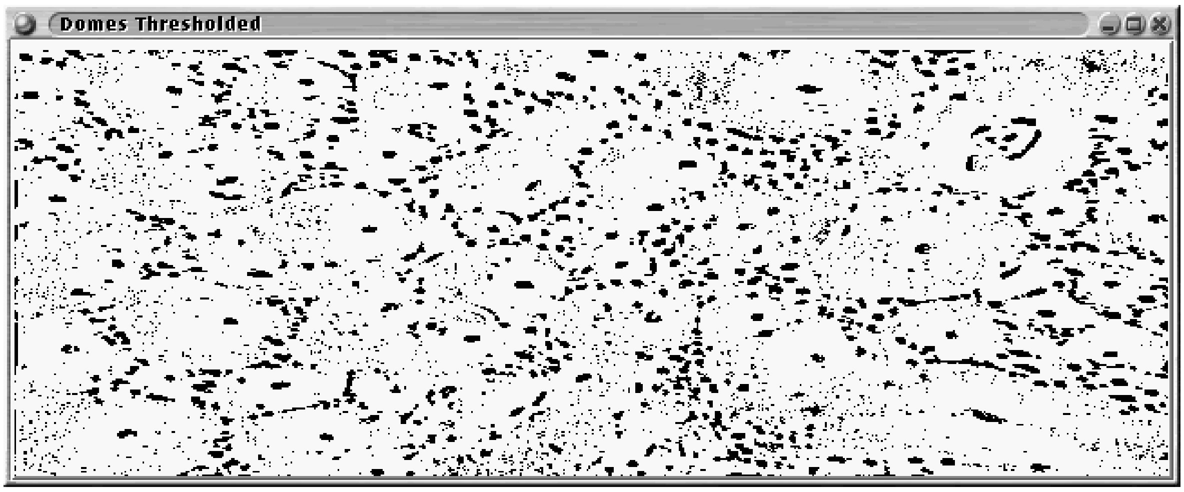
(c) Markers – area opening, threshold 50.

FIGURE 4.11 The domes and watershed segmentation after area opening of the markers.

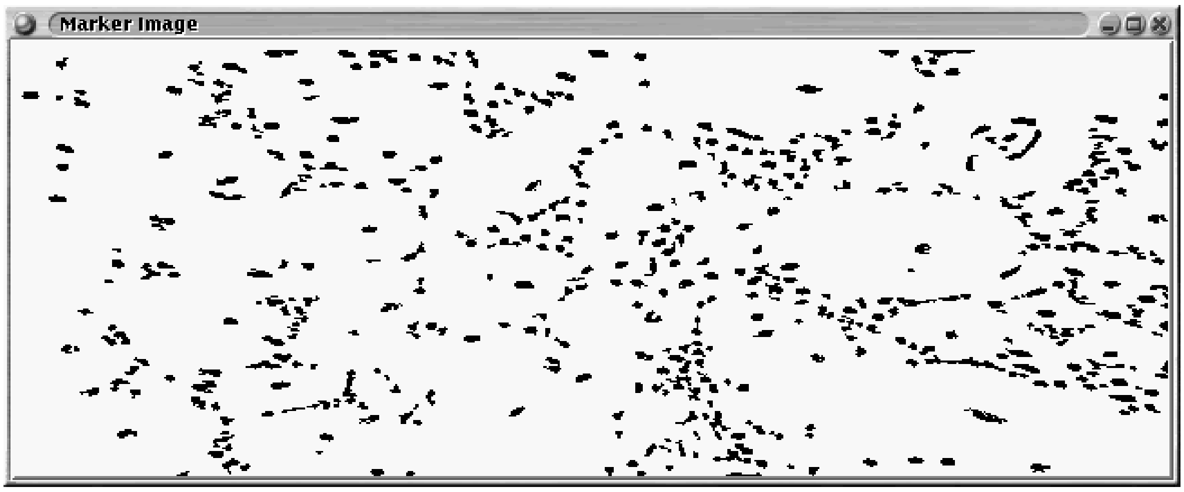
intensity (subsection 4.4.2). Small markers, irrespective of intensity, are likely noise induced or due to fragmented markers. These may be removed by a connected components area opening, or, an erosion followed by conditional dilation on the binary marker image. Finally, fragmented markers should be merged by closing the binary marker image with a circular (for isotropy) structuring element. The optimal radius was found to be 1. The precise parameters of the marker extraction and processing algorithm would need to be experimentally verified for different types of froth (for example UG2 vs. Merensky, as well as the types of bank under observation (cleaner, rougher, etc.)) and would depend on the tradeoff between over-segmentation of large bubbles and merging small bubbles

As an example of what can be done using this approach consider figure 4.12. The data is not denoised in any way and the segmentation is achieved using the marker based techniques described in this chapter. Note that the image is not denoised or preprocessed in any way.

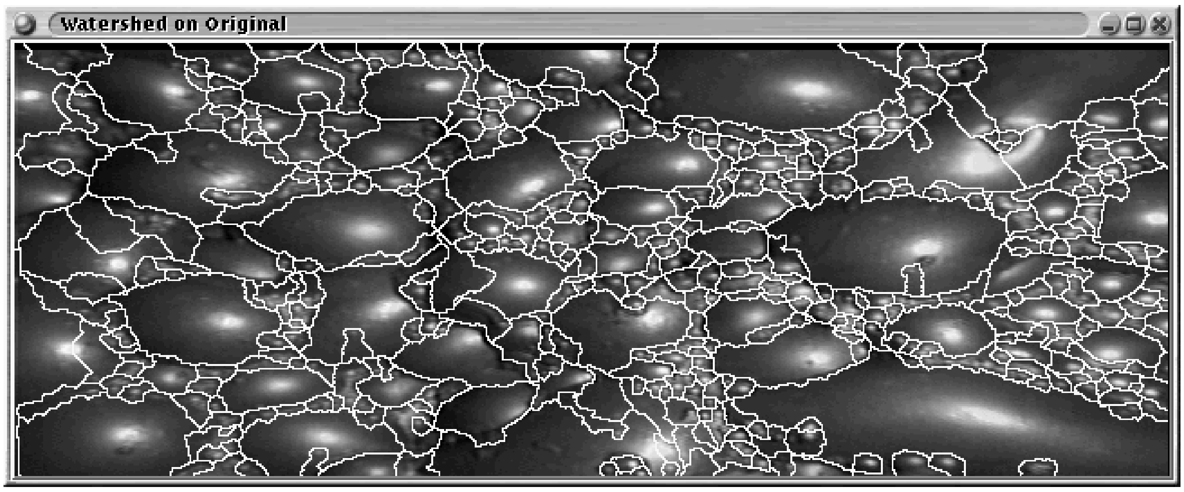
The domes are extracted using a rolling ball eroded image. The radius of the rolling ball is 2. All domes less than $0.4 \times$ *Maximum Grey Level* are removed. The threshold for the area opening is 5.



(a) Domes thresholded.



(b) Markers.



(c) Watershed.

FIGURE 4.12 Demonstration of marker processing techniques on segmentation.

Part II

Denoising of Froth Images

Chapter 5

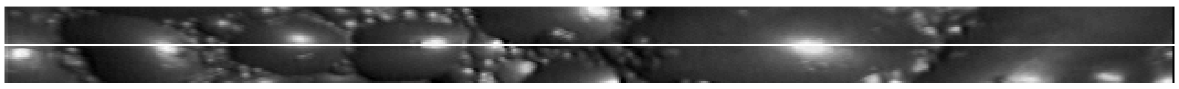
Denoising: An Introduction

Denoising could be described as a process removing an interfering component from a signal of interest. This broad definition covers the situation where the interfering signal is uncorrelated to the signal of interest, an image with Gaussian noise, for example, as well as the case where one wishes to remove structures on the signal. In the context of froth image analysis, for example, noise and tiny bubbles on large bubbles, complicate the segmentation and should be removed.

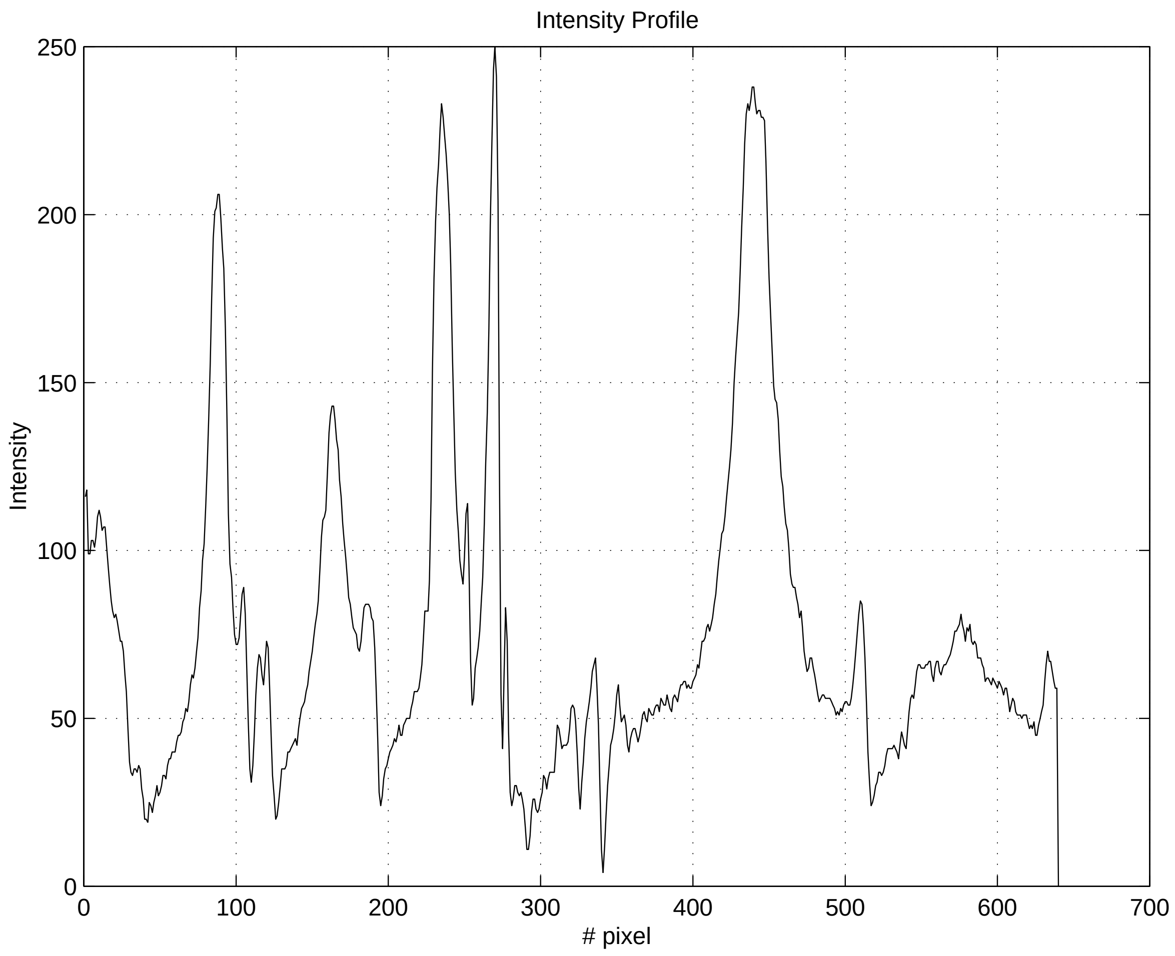
This part (part II, consisting of chapters 5 through 12 inclusive) examines denoising of images, as applied to froth images. Several denoising, image enhancement and contrast enhancement algorithms will be examined in subsequent chapters. *This* chapter will introduce the problem of denoising froth images and attempt to provide some justification for the practice. Peculiarities of froth images affecting denoising will be examined. The question of how denoising algorithms will be ranked in terms of performance will be addressed.

5.1 Why denoise at all?

If one considers the profile of a cut taken across a froth image (as shown in figure 5.1), more maxima are present than can be accounted for as bubble highlights (or regions of largest intensity). Some of these maxima are due to noise and others due to peculiarities of the froth image (the peculiarities will be examined later). It should be noted that the utility of a preprocessing (here denoising) algorithm can only be measured by examining the effect of the preprocessing algorithm on the *process* using the preprocessed data. This leads to the examination of a watershed segmented unprocessed froth image (figure 5.4(c)). Clearly, the bubbles making up the froth are over-segmented. The nature of the watershed segmentation used is that each local maxima acts as the nucleus of a watershed region [13, 141] (see chapter 3 for a discussion of watershed algorithms). Any unwanted maxima (due to noise for example) simply adds to the number of regions and complicates the segmentation. The froth segmentation problem requires then not only that small maxima and minima due to noise are removed, but also that small non-noise maxima (due to tiny bubbles, for example) are also removed.



(a) Position of intensity profile on froth subimage.



(b) Intensity profile.

FIGURE 5.1 Extraction of an intensity profile on an unprocessed froth image.

5.2 Some peculiarities of froth images

A number of artifacts are present on froth images. These complicate the denoising process, in that they must be handled in a reasonable way. Given that any maximum leads to a segmented region, the following peculiarities can be seen to produce watershed regions which do not correspond to an actual bubble:

(i) *Small bubbles on larger bubbles* (see figure 5.2(a)). The watershed segmentation process is marker driven. The highlights (or regions of largest intensity) are used as the markers. A large bubble with some small bubbles on the surface, then appears to have more than one highlight and is over-segmented. At best the small bubbles are delineated within the larger bubble.

(ii) *Reflections of small bubbles off of large bubbles* (see figure 5.2(b)). These complicate matters as explained above.

(iii) *Fragmentation of highlights* (see figure 5.2(c)). These complicate matters as explained above.

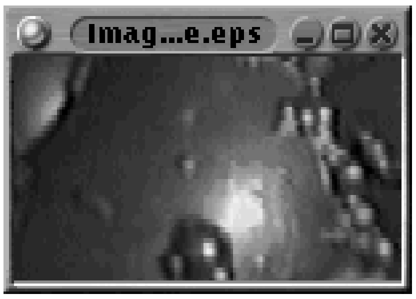
(iv) *Transparent bubbles* (see figure 5.2(d)). The transparent bubble introduces a highlight over a partially shaded region which may (or may not) contain other bubbles. The bubbles visible underneath will be segmented rather than the transparent bubble itself.

The above considerations prompt the choice of the figure 5.4(a) as a test image for the algorithms to be *qualitatively* evaluated. Figure 5.4(a) shows examples of the above pathologies but also contains bubbles of various shapes and sizes.

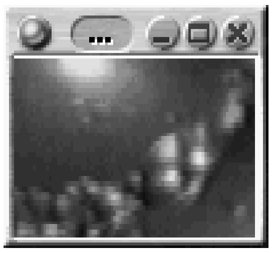
5.3 Contrast and image enhancement

Contrast enhancement, not only for improving the visibility of froth structures, but also as an adjunct to the denoising process, needs to be examined. Close examination of the original image, figure 5.3(a), and the average over the sequence (see figure 5.3(b)) shows variation in the lighting across the surface of the froth from right to left. The sequence used was 224 images long and taken from the Merensky cleaner bank. Only the even fields were used. Figure 5.3(c) shows the difference between the original image and the average image. The difference image is then stretched in intensity to occupy the intensity range of 0 to 255 [50, 73]. Note also that the bubbles on the left of the corrected image are brighter than before. The well illuminated right half of the image shows little contrast change. The lesser illuminated half of the image is “fuzzier”, or shows a random texture. This is likely due to noise, which is enhanced by histogram stretching, especially for low intensity regions.

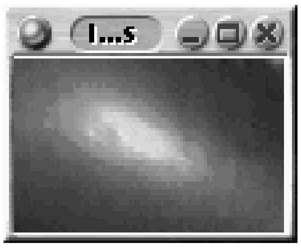
This lighting variation is an inevitable consequence of the difficulties of robust industrial lighting coupled with the need for a *single* light source. A single light source ensures that each bubble will have at most only one region of maximum intensity, or highlight. On the other hand, a single light source may not uniformly illuminate the froth surface. These highlights can potentially be used as nuclei



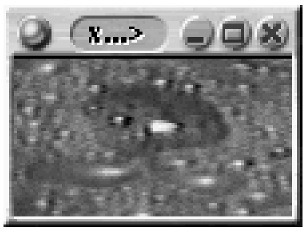
(a) Small bubbles on a large bubble.



(b) Bubble inter-reflections – upper left hand corner.



(c) Highlight fragmentation.



(d) Transparent bubbles.

FIGURE 5.2 Some froth image pathologies.

for a region growing segmentation process. A tool is then needed for removing lighting variations and, if possible, enhancing the bubble boundaries. The primary difficulty, is that the enhanced image is susceptible to noise. Removing low frequency information, by removing slow intensity changes (the lighting variation), emphasises the high frequency content of the image, or the region of the spectrum where the Signal to Noise Ratio (SNR) is low (realizable signals are band-limited [50, 73]). The other constraint of emphasising bubble boundaries, implies that the image gradient close to, and at, the boundary be emphasised. It is well known that gradient operations in images is noise prone [50, 73]. This indicates that care is needed to avoid enhancing noise at the expense of the image features desired.

5.4 Comparing denoising algorithms

A large number of denoising algorithms are described in the literature (see [56, 113, 124, 125] for some examples). Selecting an algorithm for a particular application requires some way of ranking the algorithms. Some means of comparison suggest themselves, and tend to be used in the literature:

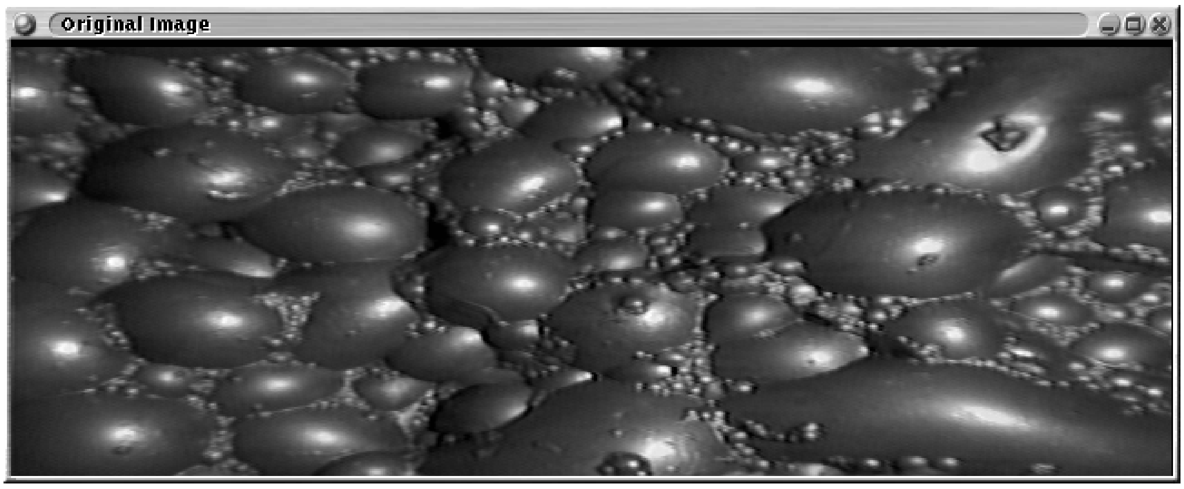
(i) Since the denoised and/or enhanced images will be segmented (see table 2.1 for some useful parameters measurable from froth images or image sequences) it seems reasonable to rank the denoising algorithms in terms of the effect on the segmentation. The primary disadvantage is that no means of making the comparisons appear to exist except for hand segmentations of froth images, or computer generated images where ground truth is available.

(ii) Adding noise to images, denoising and comparing the denoised and original images. The primary objection to this approach is that there is no way of assessing the suitability of the algorithm for the segmentation process. The advantage is that some ground truth is available which will allow for *quantitative* comparison of the algorithms. A difficulty is that noise already present in these images complicates the assessment process.

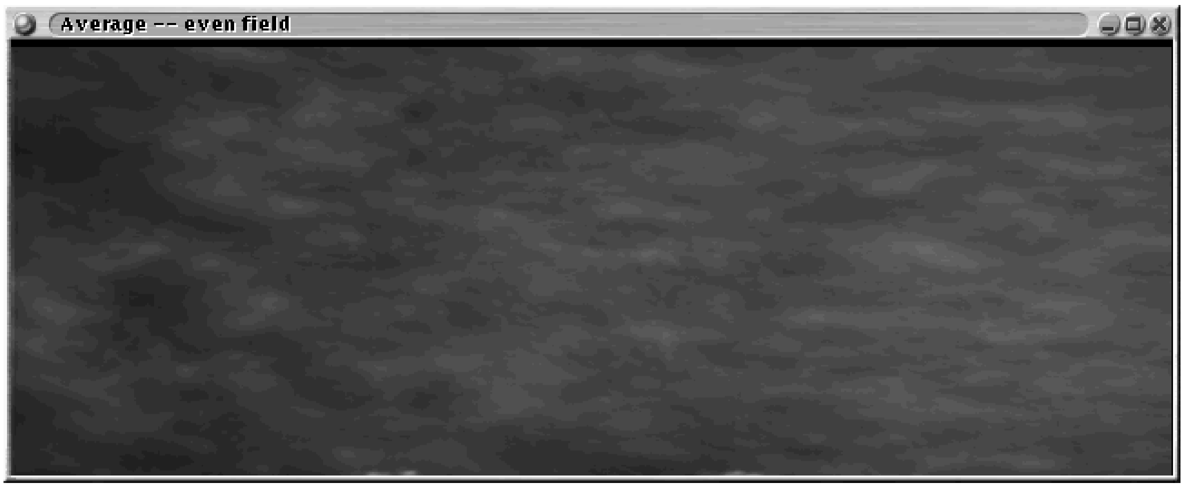
(iii) *Qualitative* comparisons of the algorithms. In other words comparing the results of the segmentation and denoised images by eye. This approach has the disadvantage of being observer dependent. A second disadvantage is that the data set used for comparison is limited, there are only so many images a human can reasonably review. This appears to be the primary means of comparison of algorithms proposed in the literature (see [113], for some examples).

The issues involved in, and results of, the comparison process are summarised in chapter 12.

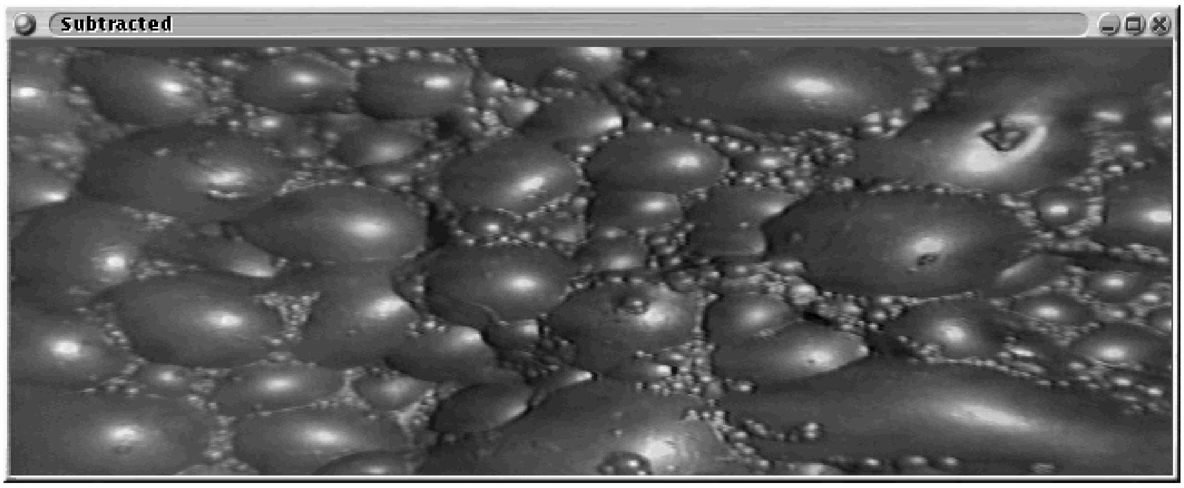
As an example of the qualitative results possible, examine the segmentation of figure 5.4(a), an unprocessed image, as shown in figure 5.4(c). The effect of a denoising algorithm on the segmentation can be used to gauge the performance of the denoising algorithm. The markers used for segmentation (figure 5.4(b)) are generated by morphological opening with a rolling ball of radius one and greyscale



(a) Original image.



(b) The time average over a sequence of images.



(c) Original image corrected for intensity gradient.

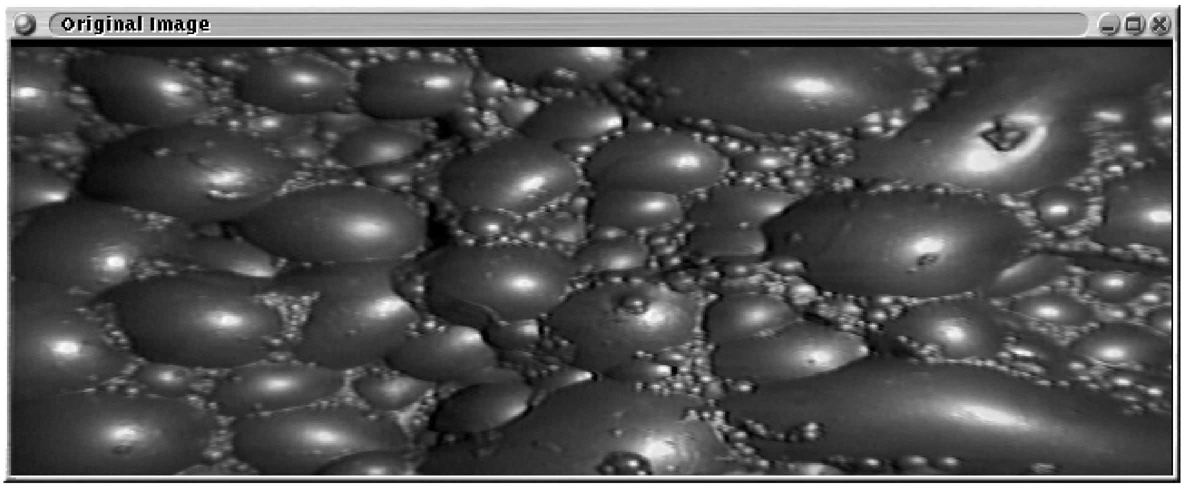
FIGURE 5.3 An example of baseline subtraction.

reconstructing by dilation (see chapter 4 for details). The watershed used is a marker based watershed using the markers generated by the previous stage (see chapter 3 for a discussion of watershed algorithms).

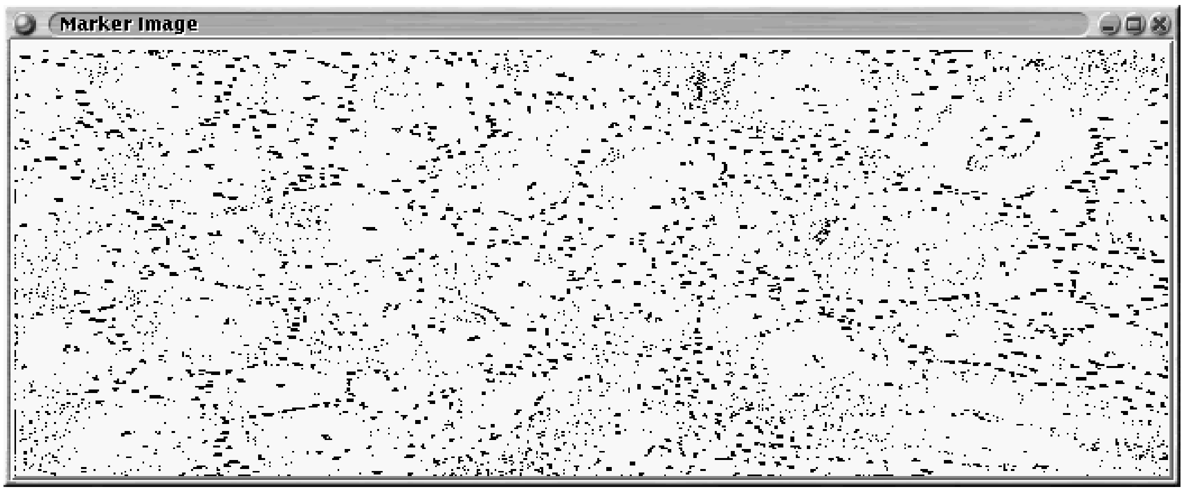
Figure 5.4 illustrates the effect of noise and froth image pathologies on the watershed segmentation. There are a number of minima (figure 5.4(b)) which when compared to the corresponding positions on the froth image (figure 5.4(a)) which either do not correspond to highlights of the froth image, or correspond to cases of the froth image pathologies previously described. The effect of these extra, or spurious, markers on the segmentation (figure 5.4(c)) is to reduce the extent of regions corresponding to valid markers. This leads to large bubbles being broken up, or over-segmented. On the other hand, it is possible for a preprocessing algorithm to *remove* a marker corresponding to a valid highlight (a valid marker in other words). This leads to a bubble being merged with another watershed region. This is the under-segmentation problem, or, in other words, not all bubbles are segmented.

5.5 Subsequent chapters

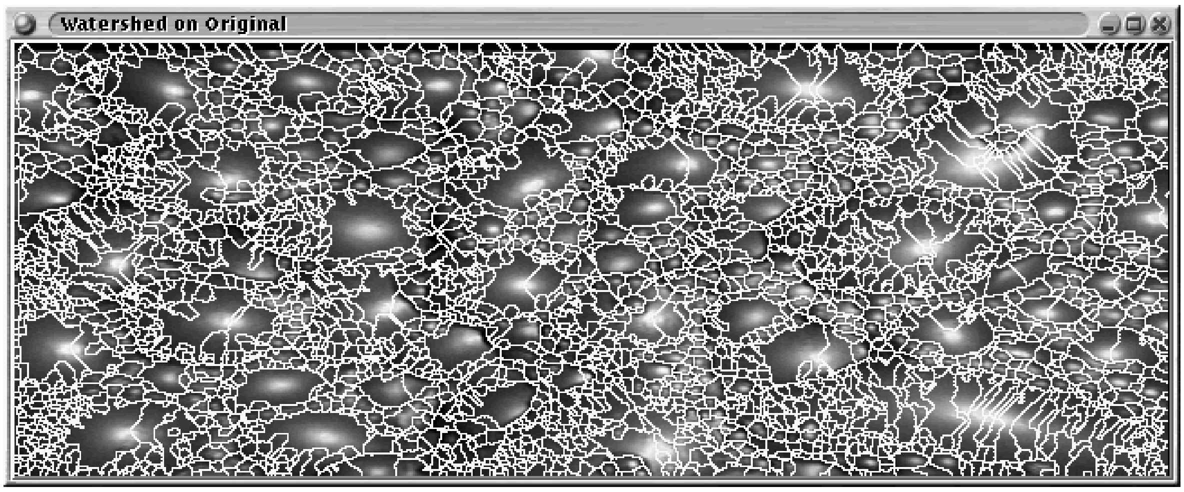
The subsequent chapters examine various denoising algorithms in some detail, namely, linear (mean) filters, anisotropic diffusion, median filters, morphological filters, finally, Occam filters and wavelet based denoising. The enhancement chapters present a brief overview of some contrast and image enhancement algorithms. Finally the various algorithms are compared and some recommendations for useful froth image denoising, enhancement and contrast modification algorithms are given.



(a) An unprocessed froth image.



(b) Markers for unprocessed froth image.



(c) Watershed segmented unprocessed froth image.

FIGURE 5.4 Unprocessed froth image, markers and associated watershed.

Chapter 6

Moving average filtering

Linear average (or low-pass) filters are traditional for separating a signal from additive Gaussian noise [50, 73]. realisable signals have finite band-width (no sensor can measure infinite bandwidth, and no realisable process can generate infinite band-width signals). Low-pass filters remove high frequency content of signals, preserving, as far as possible, low frequency content. It is well known that images have significantly more energy (information) in low frequencies than at high frequencies, for example, the Joint Photographic Experts Group (JPEG) lossy image compression standard preserves low-frequency information at the expense of high frequency data [145].

6.1 Introduction

It should be noted that convolution of an image with a Gaussian is equivalent to applying the heat diffusion equation [113]:

$$\begin{aligned}\frac{\partial I}{\partial t} &= \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \qquad (6.1)\\ &= I(x, y, 0) * g(x, y, t) \qquad (6.2)\end{aligned}$$

where,

$I(x, y, 0)$	is the initial image at time $t = 0$, and,
$g(x, y, t)$	is a Gaussian with $\sigma = t$, and,
$I(x, y, 0) * g(x, y, t)$	denotes a convolution.

In other words as the diffusion progresses, the process is equivalent to convolving with progressively wider Gaussians. Formally, a Gaussian is the Green's function of the Partial Differential Equation (PDE) given in equation (6.1) [40]. A maximum principle is associated with the heat diffusion equation [113] – this ensures that as the the diffusion progresses, no spurious extrema are created. A spurious maximum is one that has no analogue in the original unfiltered image. It is true, and perhaps desirable, that neighbouring maxima may merge to form a single new maximum. This maximum, however, has identifiable parents in the original image, a spurious maximum will not (by definition). The maximum principle cannot be guaranteed to hold for other linear low-pass filters. In the case of froth flotation images this guarantees that the watershed regions merge as one increases the standard deviation (or width) of the Gaussians used. The Gaussian convolution kernel used is defined is the product of 1D Gaussians in the x and y directions [73, 149]:

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{6.3}$$

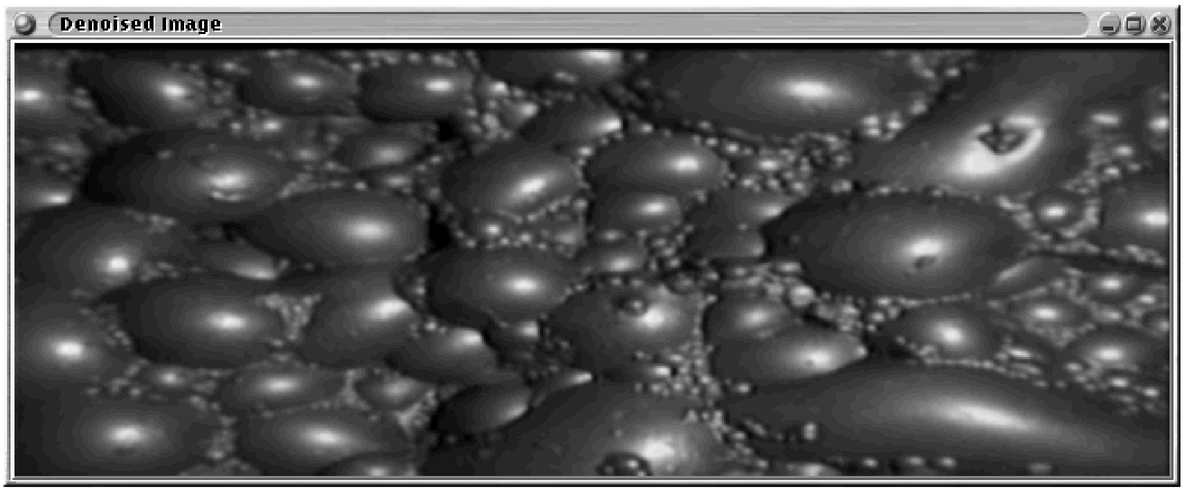
6.2 A qualitative comparison

Here an attempt is made to find the sigma best suited for denoising froth images. Figures 6.1 through 6.4 show the effect of mean filtering using a Gaussian kernel. The effects on the image, marker and watershed lines are shown. The markers are generated by opening with a rolling ball of radius one and greyscale reconstruction by dilation (see chapter 4 for details). The watershed used is a marker based watershed using the markers generated by the previous stage (see chapter 3 for a discussion of watershed algorithms).

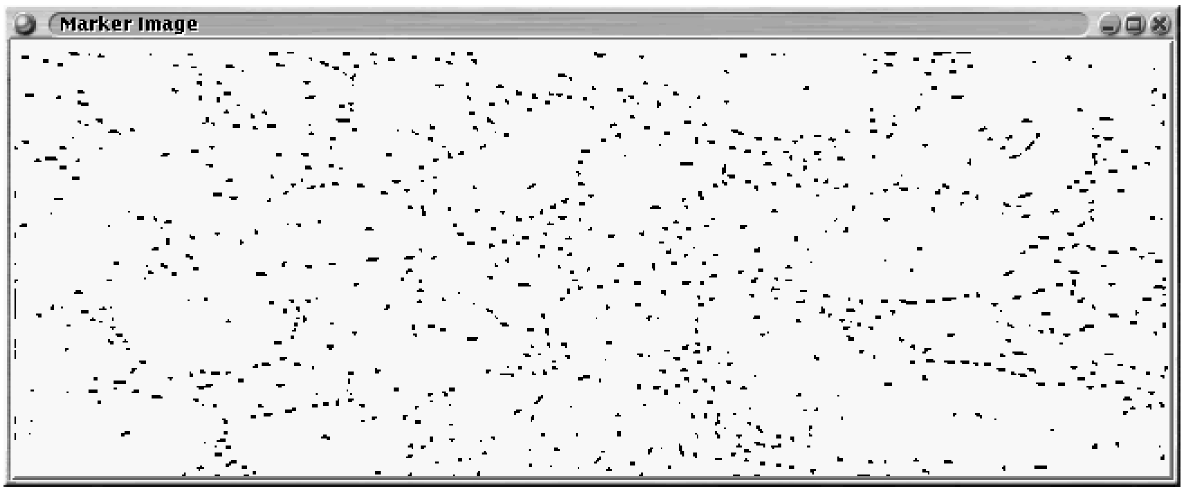
The region of support of the filter is truncated to a square. The size of the footprint is chosen to be $(8\sigma + 1) \times (8\sigma + 1)$ pixels rounded up, this accommodates a circle of radius 4σ. This allows for finite support of the filter (a Gaussian is otherwise of infinite extent). It can be shown that the pixel intensities on the radius selected, are weighted by the Gaussian to be less than $\frac{1}{2\pi\sigma^2} e^{-\frac{(4\sigma)^2}{2\sigma^2}}$. For integer valued σ this is less than 0.54×10^{-6}, when $\sigma = 1$, and hence pixels further than 4σ from the centre of the Gaussian contribute negligibly to the convolution sum.

Table 6.1 shows the number of watershed regions after Gaussian filtering with various values of σ and associated footprint sizes. Figure 6.1 shows the application of a Gaussian filter with $\sigma = 1$. Note the over-segmentation of the large bubbles and the merging of the small bubbles. Figure 6.2 shows the application of a Gaussian filter with $\sigma = 2$. The over-segmentation is reduced. Figure 6.3 shows the application of a Gaussian filter with $\sigma = 3$. Note that the over-segmentation is reduced but that small bubbles are merged. Figure 6.4 shows the application of a Gaussian filter with $\sigma = 4$. The results are much the same as for figure 6.3.

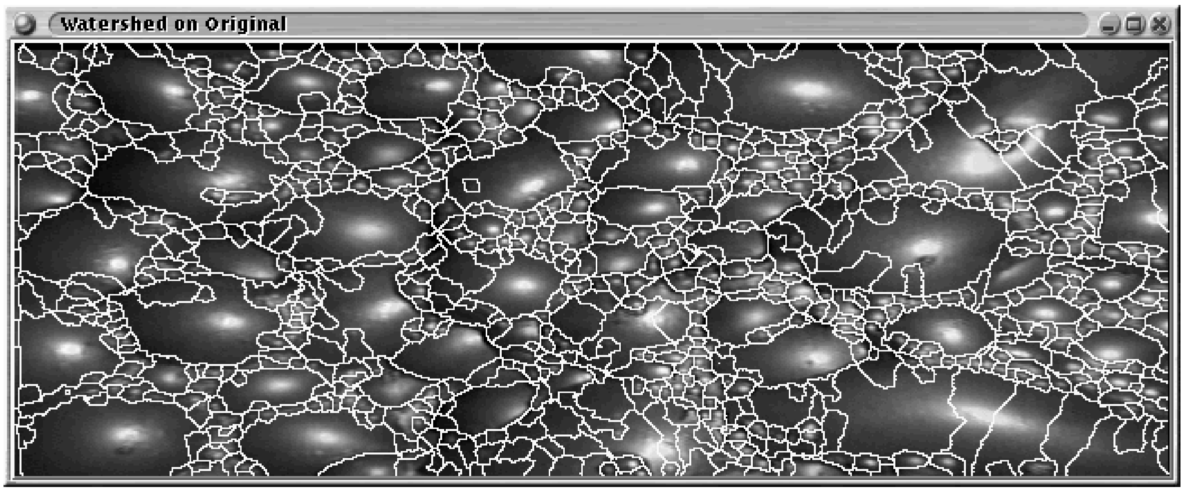
The effect of the denoising on the markers (figures 6.1(b) through 6.4(b)) is of particular interest. As the denoising increases, the markers of small extent are smoothed away. Shallow markers, or markers whose intensities are close to the mean, tend to be smoothed away sooner. Note especially that as the σ increases the denoised images become more blurred, this makes the positions of the boundaries between the bubbles uncertain. This is the primary objection to average filters in general (see [23, 56, 113] for more details).



(a) Denoised image.

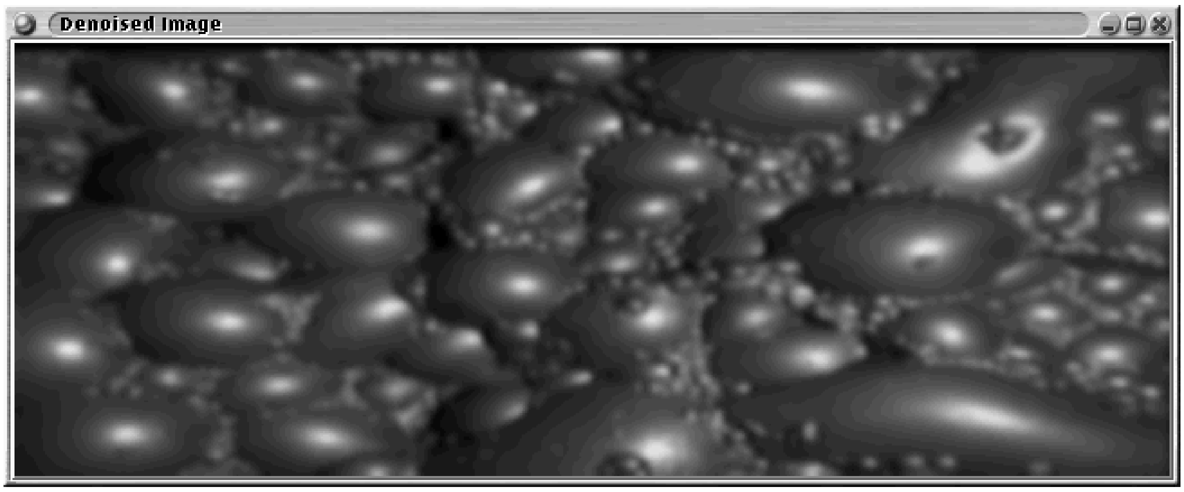


(b) Marker image.

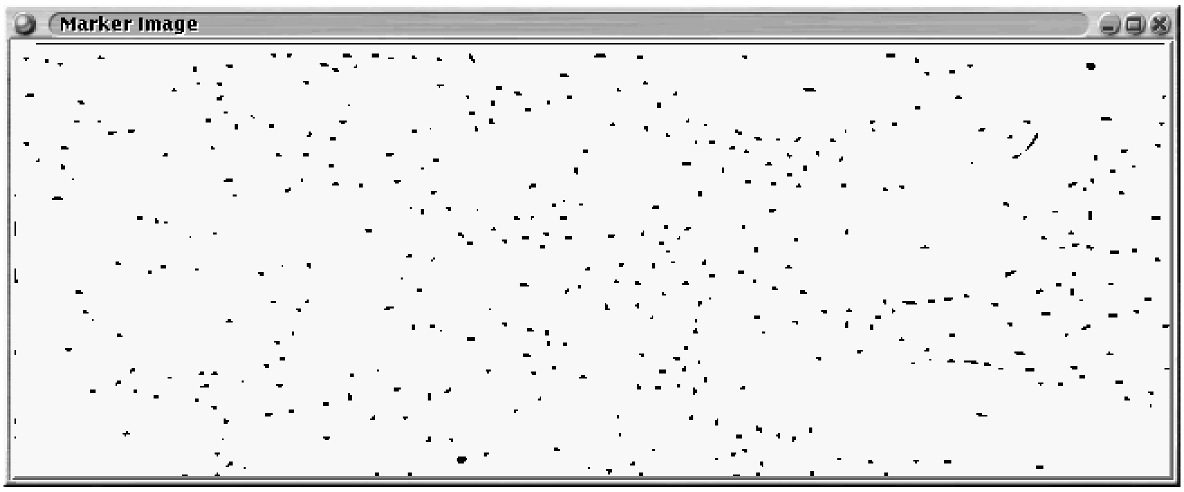


(c) Watershed segmented image.

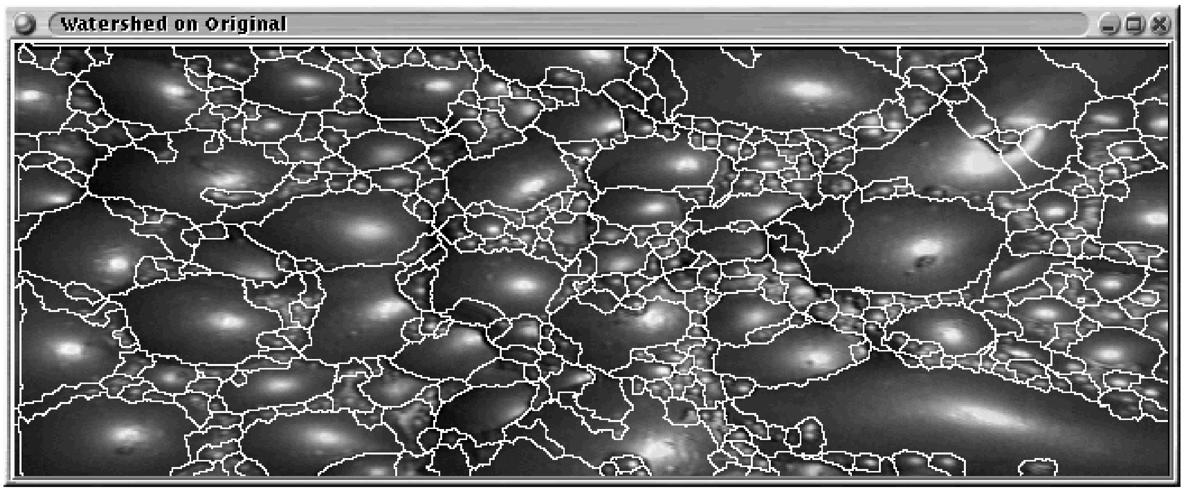
FIGURE 6.1 Result of mean filtering ($\sigma = 1$) on image, marker and watershed.



(a) Denoised image.

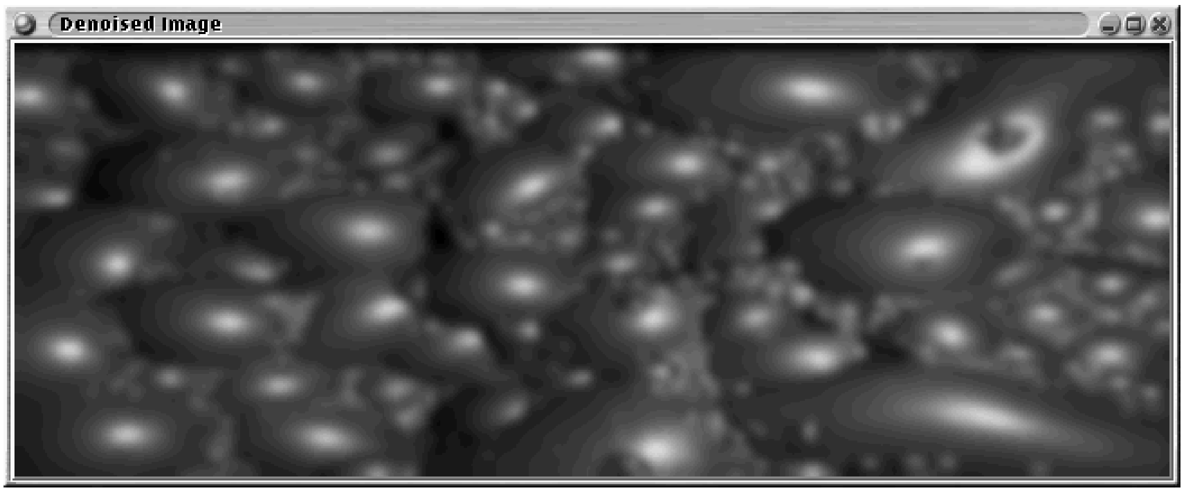


(b) Marker image.

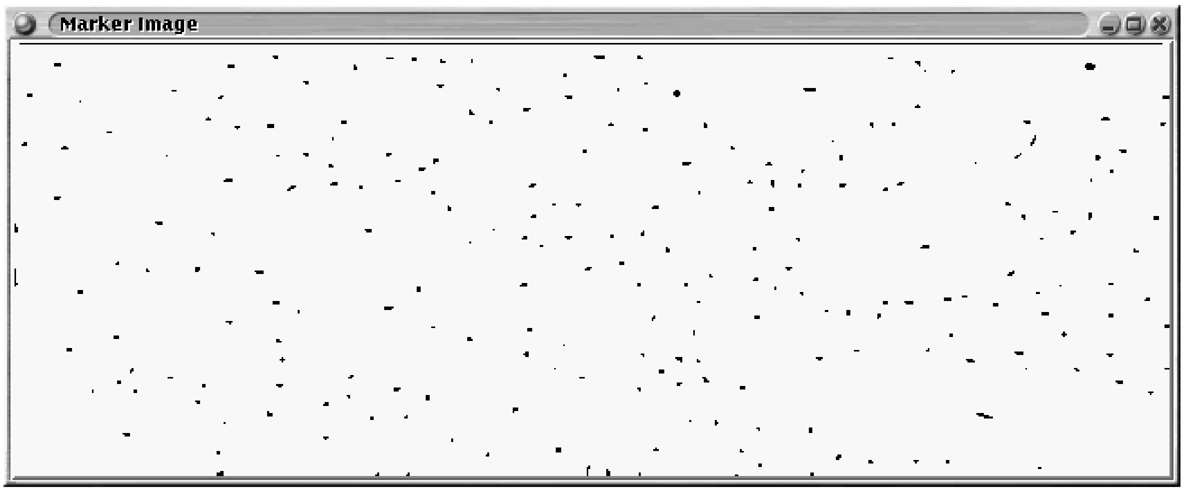


(c) Watershed segmented image.

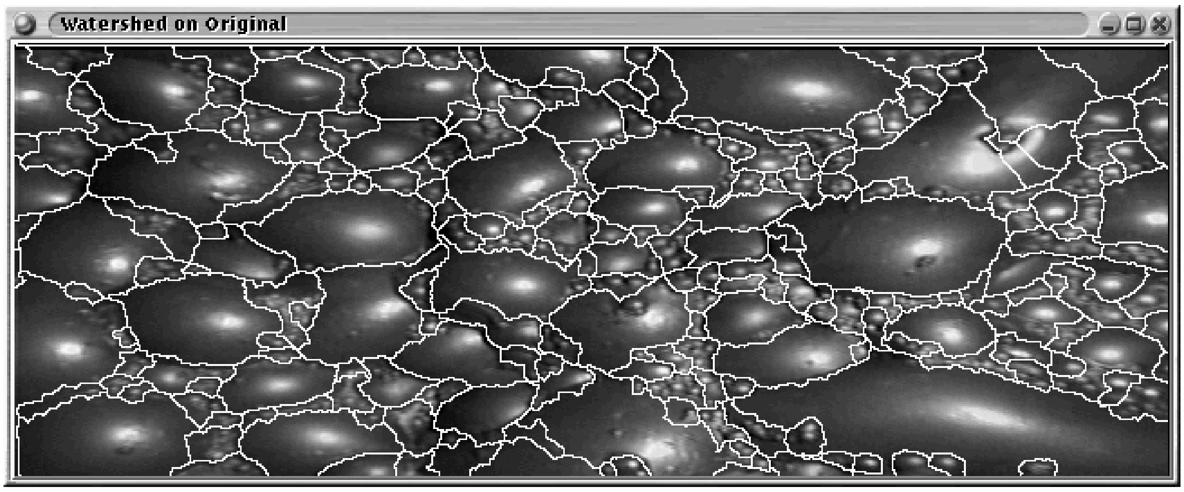
FIGURE 6.2 Result of mean filtering ($\sigma = 2$) on image, marker and watershed.



(a) Denoised image.

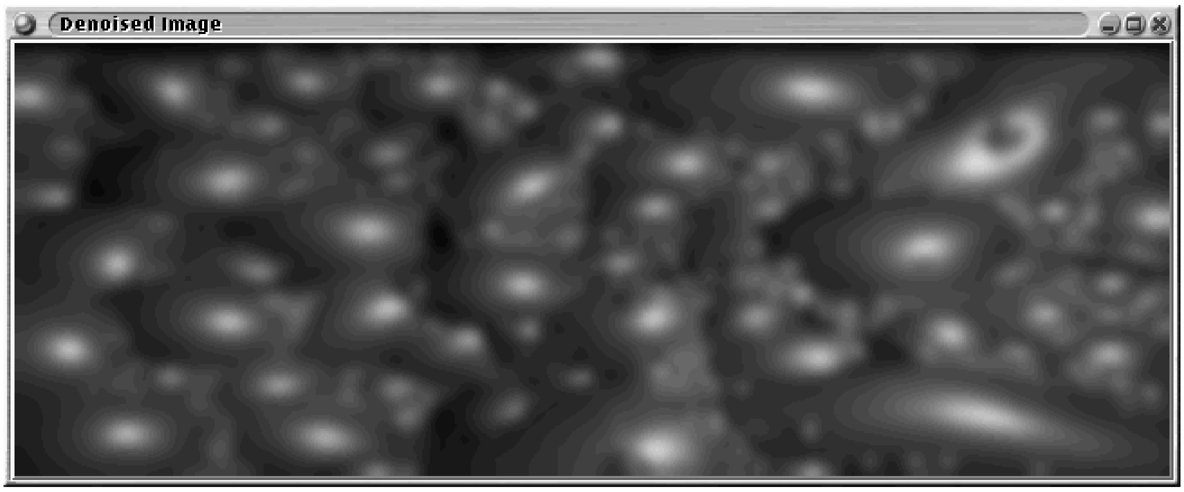


(b) Marker image.

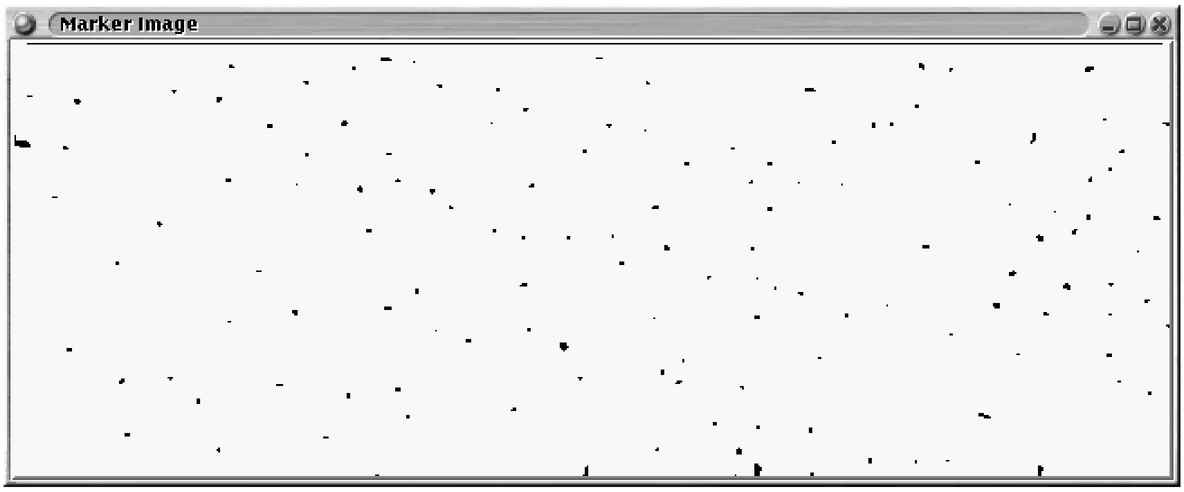


(c) Watershed segmented image.

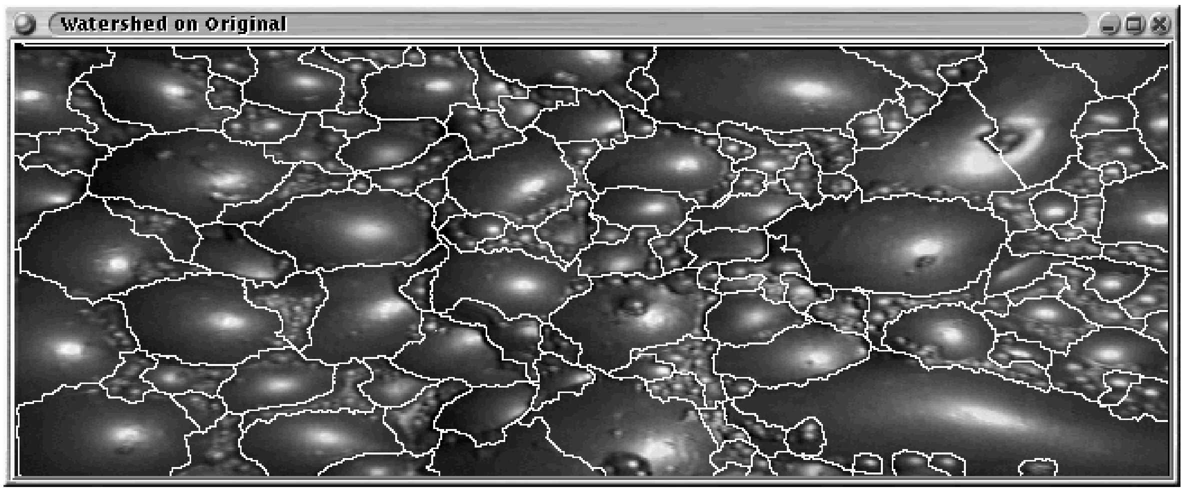
FIGURE 6.3 Result of mean filtering ($\sigma = 3$) on image, marker and watershed.



(a) Denoised image.



(b) Marker image.



(c) Watershed segmented image.

FIGURE 6.4 Result of mean filtering ($\sigma = 4$) on image, marker and watershed.

Clearly then a larger sigma will not improve the segmentation. Small bubbles are merged if their separation is less than the radius of the filter, or 3σ. Of the four σ's used, the best performance can be seen to be $\sigma = 2$.

6.3 Discussion

This form of denoising is biased against small bubbles, but is relatively simple to implement. A difficulty is that large sigma values require large filter domains (or footprints), this extends the computation time. A natural optimisation takes the separability of the Gaussian bump into account and first applies the filter in the x and then in the y direction.

Iterating a Gaussian filter can be shown to have the following effect:

$$g_1(t) * g_2(t) * \cdots * g_N(t) = \frac{1}{\sqrt{2\pi \sum_{k=1}^{N} \sigma_k^2}} e^{-\frac{t^2}{2\sum_{k=1}^{N}\sigma_k^2}}, \tag{6.4}$$

where $g_k(t)$ is a Gaussian kernel of unit area. N is the number of Gaussian kernels convolved. This can be easily demonstrated using the Fourier transform of a Gaussian kernel [130][pg. 95]:

$$\mathcal{F}\left\{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}}\right\} = e^{-\frac{\omega^2}{4\frac{1}{2\sigma^2}}} \tag{6.5}$$

This shows a narrow Gaussian kernel (small σ) in the time domain transforms to a wide Gaussian in the frequency domain and *vice versa*. If the widths of the kernels are all the same (constant σ), then:

$$g_1(t) * g_2(t) * \cdots * g_N(t) = \frac{1}{\sqrt{2\pi N\sigma^2}} e^{-\frac{t^2}{2N\sigma^2}} \tag{6.6}$$

The effective sigma is then: $\sqrt{N}\sigma$. This allows for the effect of large σ Gaussian filters to be rapidly computed by iterating smaller σ filters.

TABLE 6.1 Number of watershed regions against filter width.

σ	Footprint	# Watershed Regions
1	9×9	823
2	17×17	401
3	25×25	234
4	33×33	138

Chapter 7

Denoising via Anisotropic Diffusion

Several types of Anisotropic Diffusion exist, the simplest types generalise the heat diffusion equation in a non-linear way. All types attempt to model the non-linear smoothing of an image via some sort of Partial Differential Equation (PDE). Catte, Lions and Morel [19] extend the original formulation of Perona and Malik [113] and add robustness. Alvarez, Lions and Morel [4] examine a different class of PDE's. The front propagation techniques of Malladi and Sethian [92] are yet another PDE based approach for denoising and shape extraction. An advantage of a PDE recapitulation of Image Processing is that a series of PDE's can be cascaded in simple fashion and solved numerically. As an example, consider cascading contrast enhancement (through adaptive histogram equalisation) and denoising via Anisotropic Diffusion [122].

The following algorithms from the Anisotropic Diffusion family are considered: the classical Perona and Malik algorithm [113]; the reformulation of classical diffusion by Nitzberg and Shiota [109]; the Green's function reformulation of Fischl and Schwartz [41, 42]; the modification proposed by Catte, Lions and Morel [19]; and the new PDE proposed by Alvarez, Lions and Morel [4].

7.1 Classical Anisotropic Diffusion via Perona and Malik

Anisotropic diffusion is a generalisation of the heat diffusion equation $I_t = \nabla^2 I$ [113]. The modification proposed by Perona and Malik was to make the heat diffusion equation spatially variant. In order to preserve edges, the diffusion process is limited at the vicinity of strong edges. The algorithm then blurs significantly less across strong edges than it does across weak edges. Additionally, strong edges are enhanced. Perona and Malik prove a maximum principle which ensures that at each iteration the new pixel value lies between the maximum and minimum values in the neighbourhood of the pixel.

The generalisation is:

$$I_t = \nabla \cdot (g(\|\nabla I\|)\nabla I) \tag{7.1}$$

The diffusion algorithm, as given, has three weaknesses:

(i) Noisy edges are not smoothed.

(ii) Some care is needed in choosing $g()$.

(iii) The number of iterations of the algorithm needs to be decided beforehand.

Perona and Malik suggest the following for $g()$:

$$g(\|\nabla I\|) = e^{(-(\|\nabla I\|/K)^2)} \qquad \text{or,} \tag{7.2}$$

$$g(\|\nabla I\|) = \frac{1}{1 + (\frac{\|\nabla I\|}{K})^2} \tag{7.3}$$

These are sometimes referred to in the literature as GaussianG and QuadG respectively [15]. For the numerical implementation of the anisotropic diffusion equation (equation 7.1), Perona and Malik show that $0 < \Delta t \leq 0.25$. Canny's noise estimator estimates K by setting K to be the 90% value of the cumulative distribution function of the histogram of the magnitude of the gradient of the image [18]. Between iterations K is updated on the denoised image generated.

7.1.1 Discussion

There is an explicit relationship between the number of iterations and the scale of the structures surviving in the denoised image. Baubaud et al. [7] demonstrated a relationship between scale and time as $\sigma = \sqrt{2t}$ for Gaussian smoothing (the heat diffusion equation). This, especially for smooth or flat regions within an image, relates scale and time for anisotropic diffusion.

The are some problems with definition as it stands: the gradient is computed in each direction via simple differences which are known to be noisy [50, 73], and edge enhancement occurs via the backward heat diffusion equation which is known to be unstable [19, 113, 152].

Canny's noise estimator [18, 113], as recommended by Perona and Malik, is used to estimate K (and reduce the number of free parameters in the algorithm). The values of the time step γ and the number of iterations are set by hand.

An alternative definition of diffusion could be:

$$I_t = g(\|\nabla I\|)\nabla \cdot \nabla I \tag{7.4}$$

This allows for the inhibiting of diffusion at high edge strengths (using the same definitions of $g(\cdot)$ as before), but no edge enhancement. This is less general than Perona and Malik's formulation since:

$$\begin{aligned} I_t &= \nabla \cdot (g(\|\nabla I\|)\nabla I) \\ &= \big(\nabla \cdot (g(\|\nabla I\|)\big)\nabla I + g(\|\nabla I\|)\big(\nabla \cdot \nabla I\big) \end{aligned}$$

The term introducing the edge enhancement, and hence potential difficulties since this implies the backwards heat diffusion – well known to be ill-posed [19, 113, 152], is then $(\nabla \cdot (g(\|\nabla I\|))$.

Table 7.1 records the decrease in the number of watershed regions as the number of iterations increases. Figures 7.1 through 7.4 shows the effect of $g()$ and iterations on the denoised image, markers and watershed segmentation. The markers are generated by opening with a rolling ball of radius one and greyscale reconstruction by dilation (see chapter 4 for details). The watershed used is a marker based watershed using the markers generated by the previous stage (see chapter 3 for a discussion of watershed algorithms).

Note that at large iteration values the large bubbles show stair-casing. Note that strong edges remain sharp. Stair-casing is essentially a smooth curve becoming a series of grey-level steps. This can be seen by examining the denoised image after 8 iterations (see figures 7.2 and 7.4). The gamma value, or time step, used for generating the images is 0.2, and K is generated, as specified, using Canny's noise estimator.

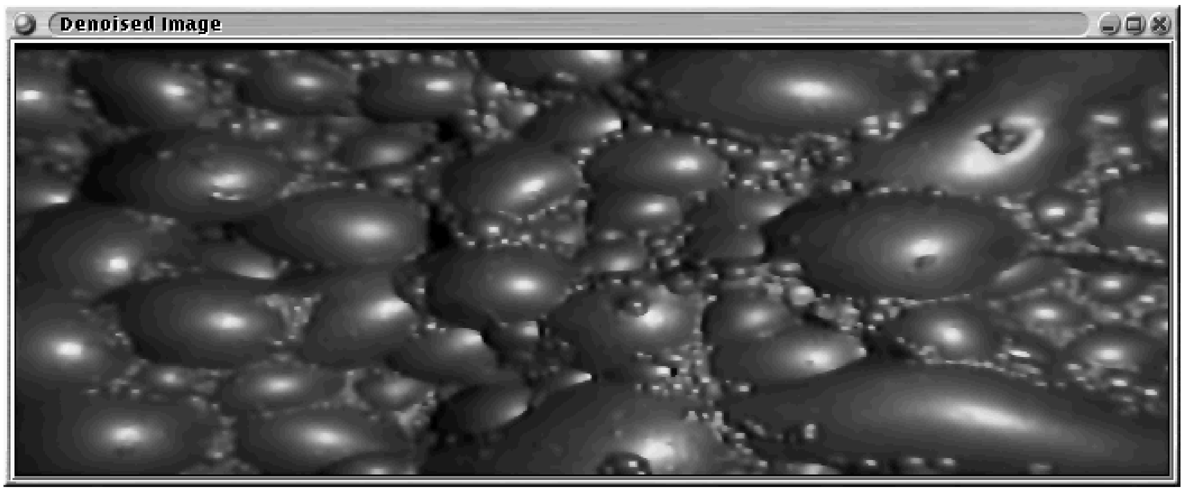
TABLE 7.1 Decrease in the number of watershed regions as the number of iterations increases.

Iterations	# Watershed Regions $g(x) = e^{-\|\nabla I\|/K^2}$	$g(x) = \frac{1}{1+\left(\frac{\|\nabla I\|}{K}\right)^2}$
1	1272	1259
2	923	911
3	783	767
4	681	670
5	624	607
6	576	558
7	544	519
8	501	476
10	454	424
12	415	379
14	384	355
16	360	325
18	343	301
20	323	280
24	300	249
32	267	196
40	230	157

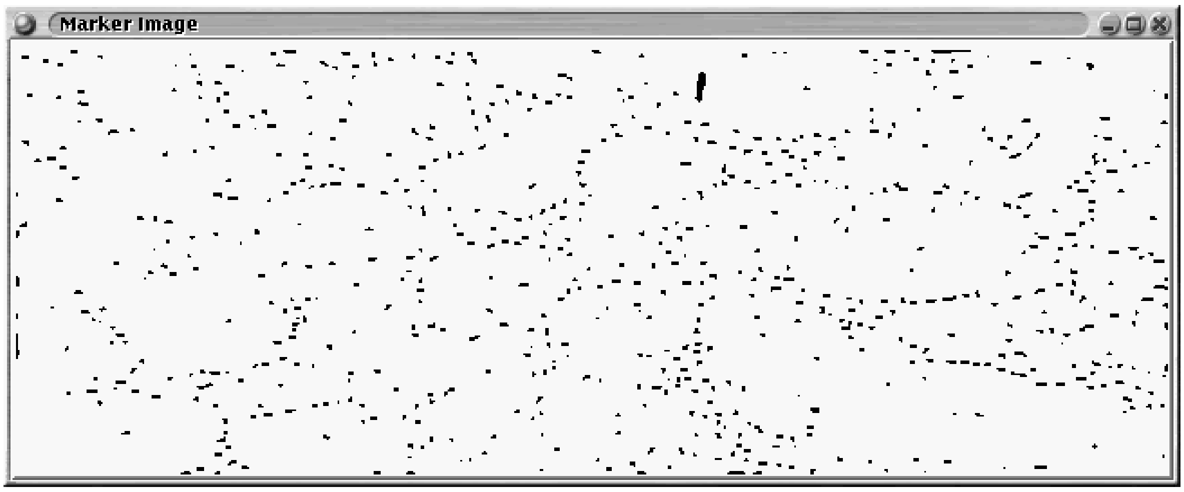
7.2 The modification of Catte, Lions and Morel

The observation that, in general, the inverse heat diffusion equation is ill-posed, raised certain problems with the formulation of diffusion by Perona and Malik [113]. The process of edge enhancement during Anisotropic Diffusion relies on the inverse heat equation. This implies that two similar starting images may diverge considerably as the diffusion process continues.

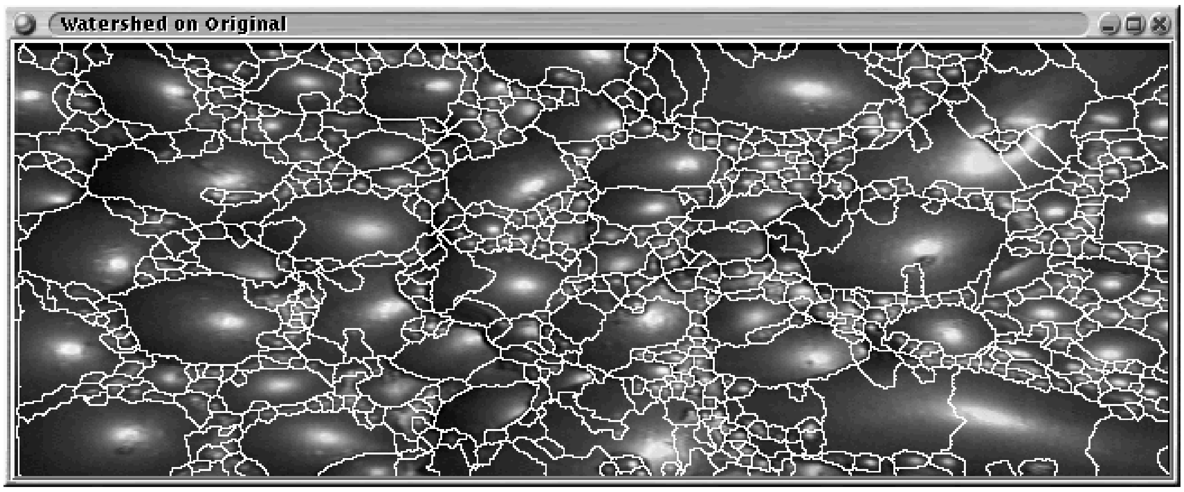
In the final modification proposed by Catte et al., [19] $g()$ operates on $\nabla Gaussian * I$, instead



(a) Denoised image – iteration 4, Perona Malik.

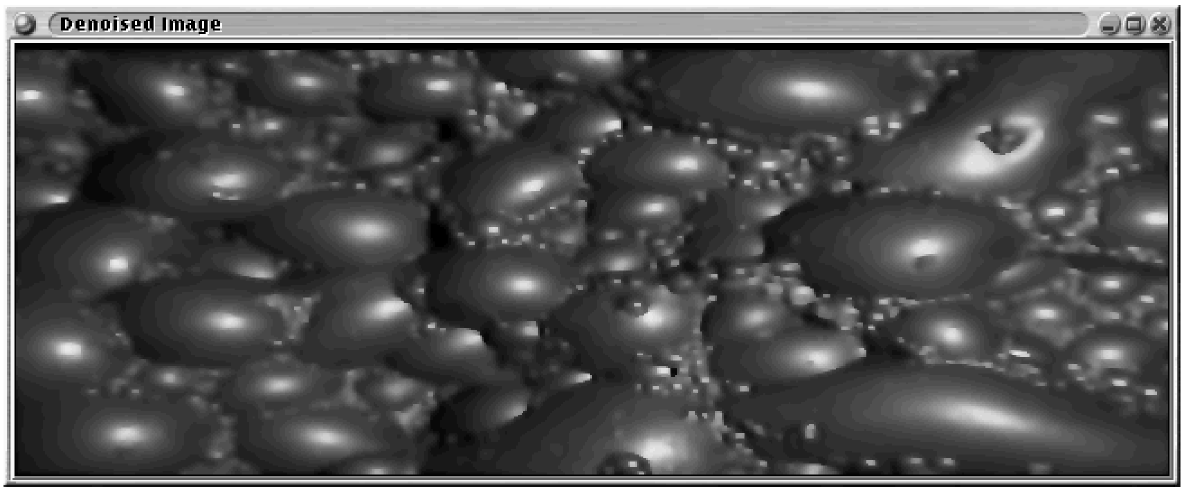


(b) Marker image – iteration 4, Perona Malik.

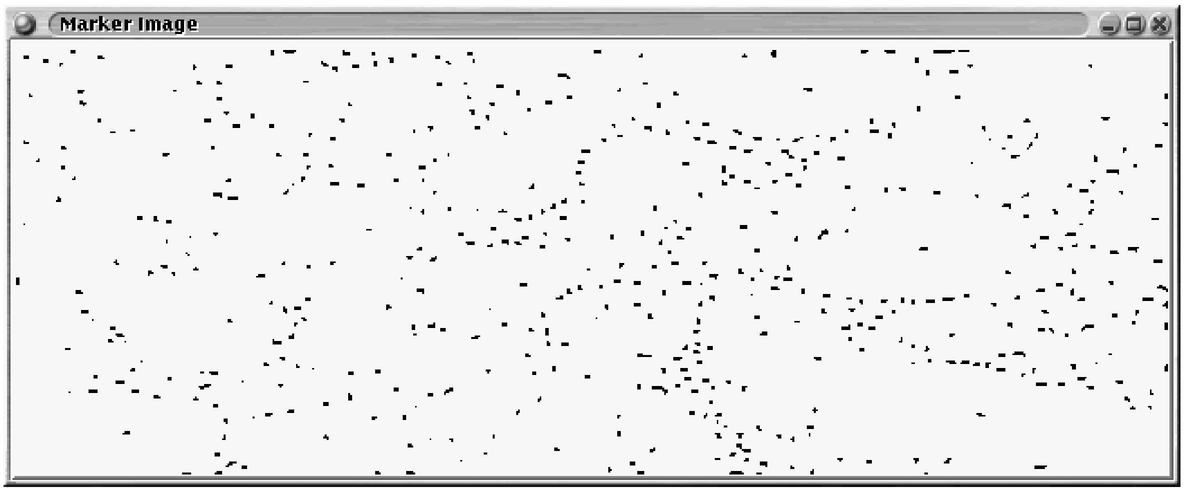


(c) Watershed image – iteration 4, Perona Malik.

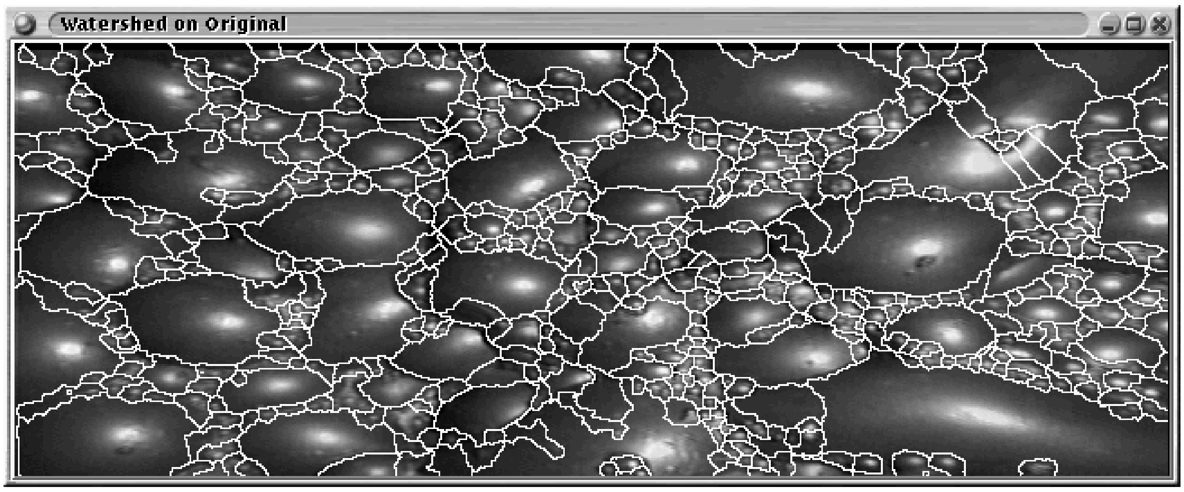
FIGURE 7.1 Denoising via Perona and Malik, $g()$ used is (7.2).



(a) Denoised image – iteration 8, Perona Malik.

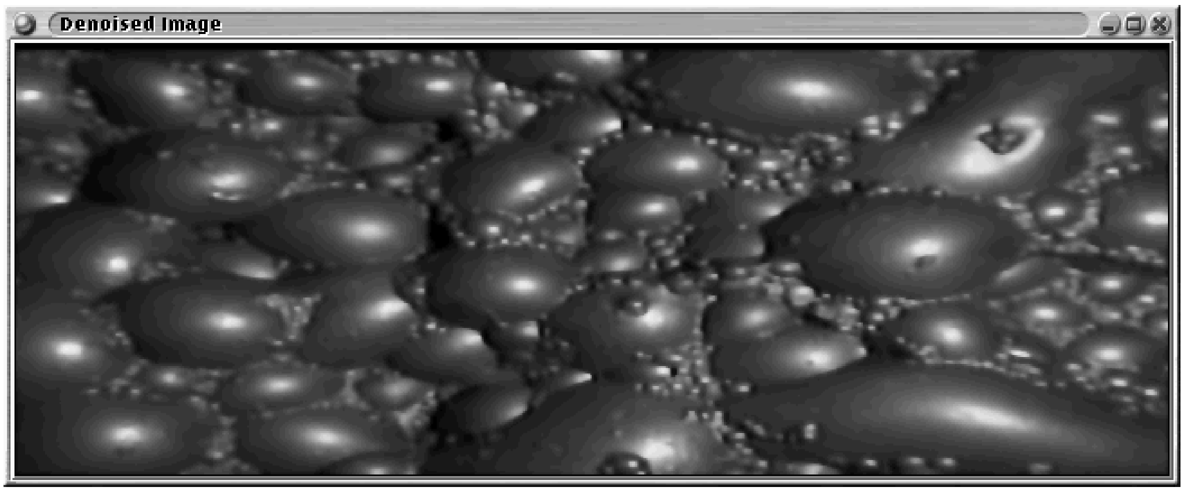


(b) Marker image – iteration 8, Perona Malik.

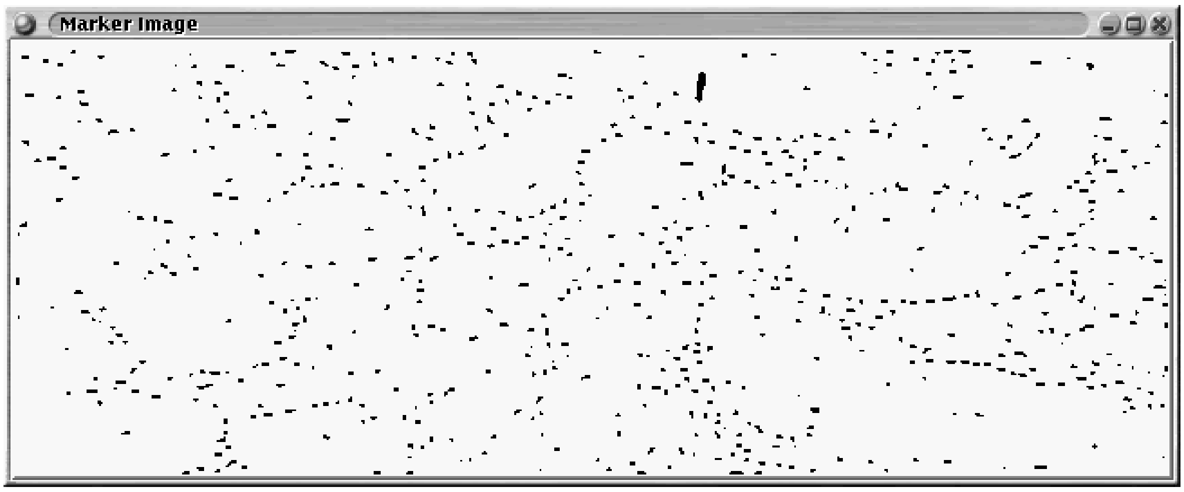


(c) Watershed image – iteration 8, Perona Malik.

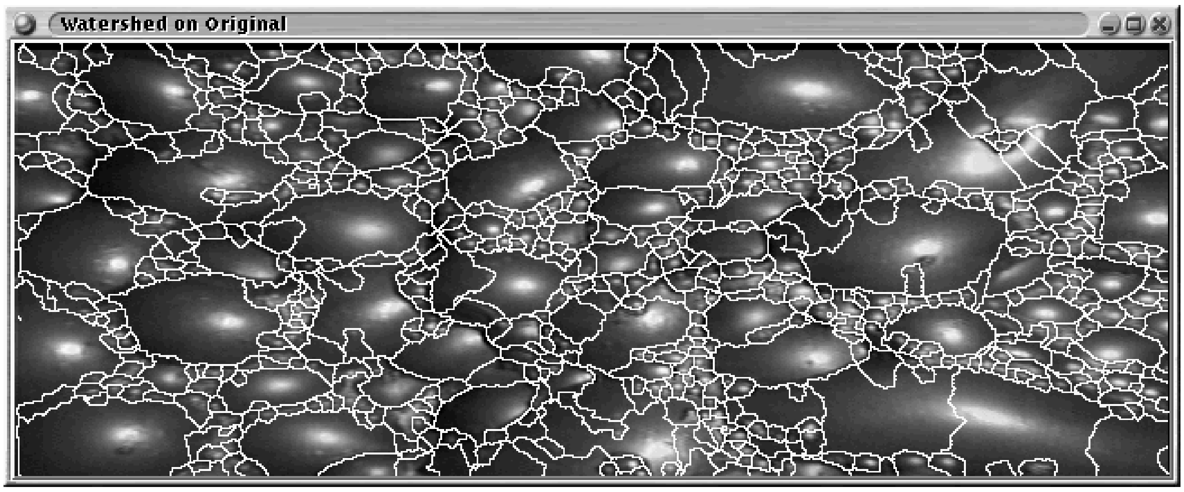
FIGURE 7.2 Denoising via Perona and Malik, $g()$ used is (7.2).



(a) Denoised image – iteration 4, Perona Malik.

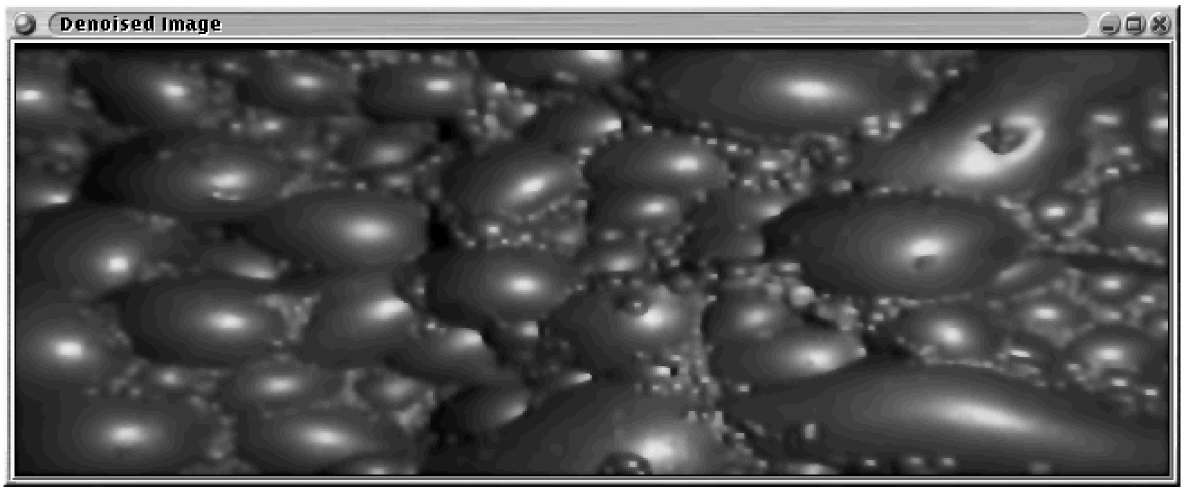


(b) Marker image – iteration 4, Perona Malik.

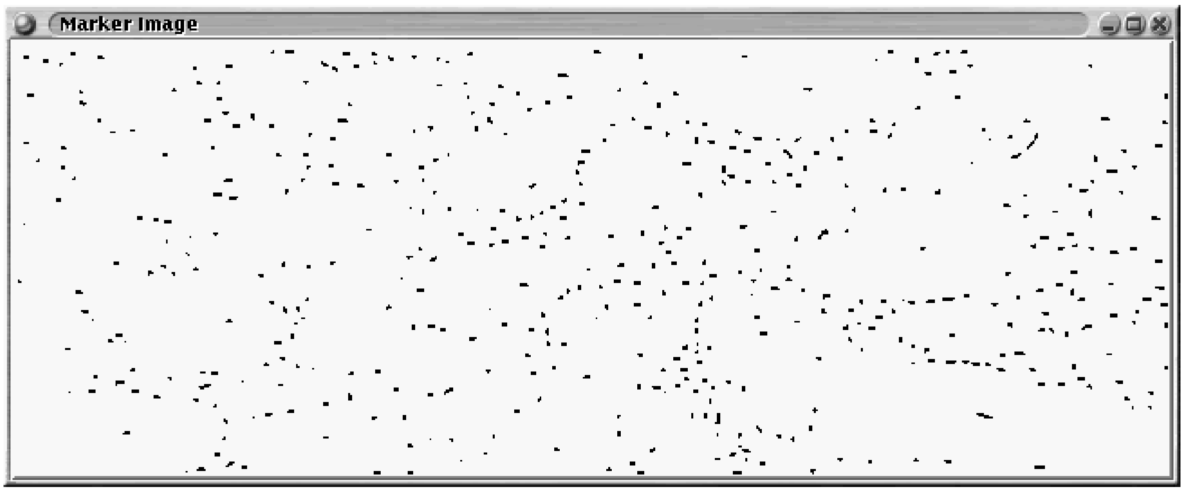


(c) Watershed image – iteration 4, Perona Malik.

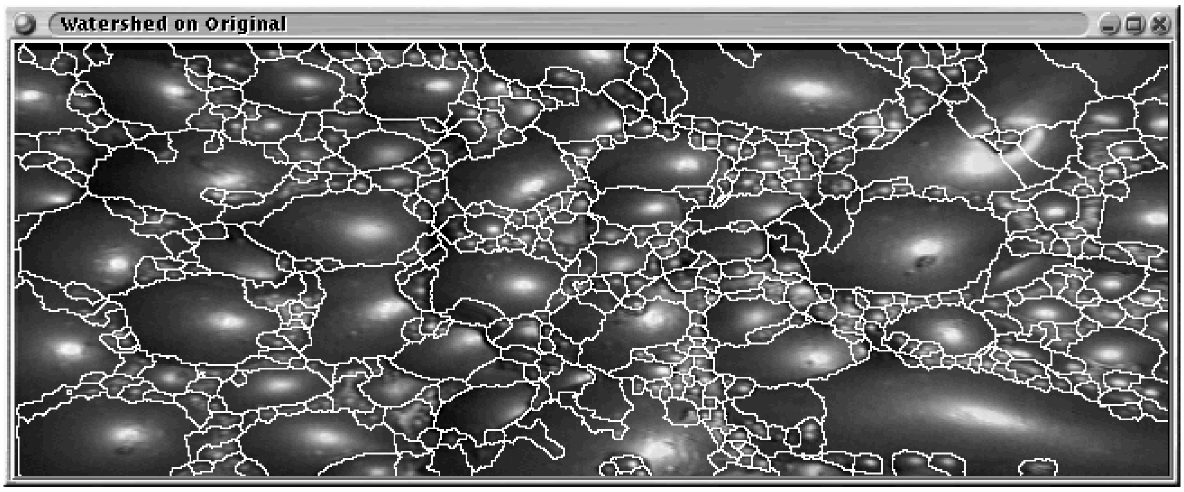
FIGURE 7.3 Denoising via Perona and Malik, $g()$ used is (7.3).



(a) Denoised image – iteration 8, Perona Malik.



(b) Marker image – iteration 8, Perona Malik.



(c) Watershed image – iteration 8, Perona Malik.

FIGURE 7.4 Denoising via Perona and Malik, $g()$ used is (7.3).

of ∇I. This modification removes the following difficulties with the implementation by Perona and Malik (indeed these difficulties were documented in Perona and Malik's original paper [113]):

(i) Introduced noise implies a noisy estimate of the gradient position and magnitude. Perona and Malik recommend an initial smoothing to reduce the noise, but this has the undesired effect of blurring edges.

(ii) The second difficulty involves $g(\nabla I)\nabla I$. In one dimension the anisotropic diffusion equation reduces to:

$$\begin{aligned} I_t &= \frac{d}{dx} g(\|I_x\|) I_x \\ &= g(\|I_x\|) I_{xx} + g_x(\|I_x\|) I_{xx} I_x \\ &= I_{xx} \left[g(\|I_x\|) + g_x(\|I_x\|) I_x \right] \end{aligned}$$

If $g_x(\|I_x\|)I_x$ is negative and larger than $g(\|I_x\|)$ then the heat equation runs backwards. This is known to be an ill-posed problem. Catte et al's modification changes the above to,

$$I_t = \frac{d}{dx} g(\|I_x * Gaussian(\sigma, x)\|) I_x \tag{7.5}$$

$$= g(\|I_x * Gaussian(\sigma, x)\|) I_{xx} + g_x(\|I_x * Gaussian(\sigma, x)\|) \frac{d}{dx} (I_x * Gaussian(\sigma, x)) I_x \tag{7.6}$$

$$= I_{xx} \left[g(\|I_x * Gaussian(\sigma, x)\|) + g_x(\|I_x * Gaussian(\sigma, x)\|)(I_x * Gaussian(\sigma, x)) I_x \right] \tag{7.7}$$

Nitzberg and Shiota [109] demonstrate the sensitivity of Perona and Malik's algorithm to grid size and algorithm specifics (for example, interpolation algorithms used). While Perona and Malik do prove a maximum principle which ensures that as one diffuses and edges are enhanced, that new minima are not introduced, this does not translate well into the discrete domain. Further any "bumps" along a gradient (due to noise for example) may well increase in height to become hills. A blurred edge sharpens to become an irregular staircase, instead of a single step.

Catte et al. use an implicit discretisation scheme to solve the above PDE. (An implicit scheme at some time step expresses the solution in terms of pixels at the new time step.) An implicit scheme requires matrix methods to solve. Explicit schemes, on the other hand, express the new intensity at a pixel using pixel intensities at the previous time step exclusively. The Perona and Malik numerical scheme is an example of an explicit scheme.

In the interests of simplicity, Catte et al's proposed numerical scheme is not used. Instead, a Gaussian smoothed version of the image at a time step is used for edge estimates for computing $g()$ only, the other edge estimates are computed using the unsmoothed image. This allows Perona and Malik's discretisation scheme to be used, and the PDE's to be compared directly. The differences in the discretisation as proposed by Catte et al., as compared to that of Perona and Malik, are then removed.

7.2.1 Discussion

Table 7.2 shows the reduction in the number of watershed regions (and hence number of markers) as the number of iterations of the diffusion process increases. Figures 7.5 through 7.8 show the effect of increasing the number of iterations on the denoised, marker and segmented froth image.

TABLE 7.2 Decease in the number of watershed regions as the number of iterations increases.

Iterations	# Watershed Regions	
	$g(x) = e^{-\|\nabla I\|/K^2}$	$g(x) = \frac{1}{1+\left(\frac{\|\nabla I\|}{K}\right)^2}$
1	1691	1674
2	1081	1061
3	879	855
4	746	727
8	568	515
16	476	347

7.3 The new PDE of Alvarez, Lions and Morel

This is an attempt to circumvent the original problems with the classical Anisotropic Diffusion algorithm.

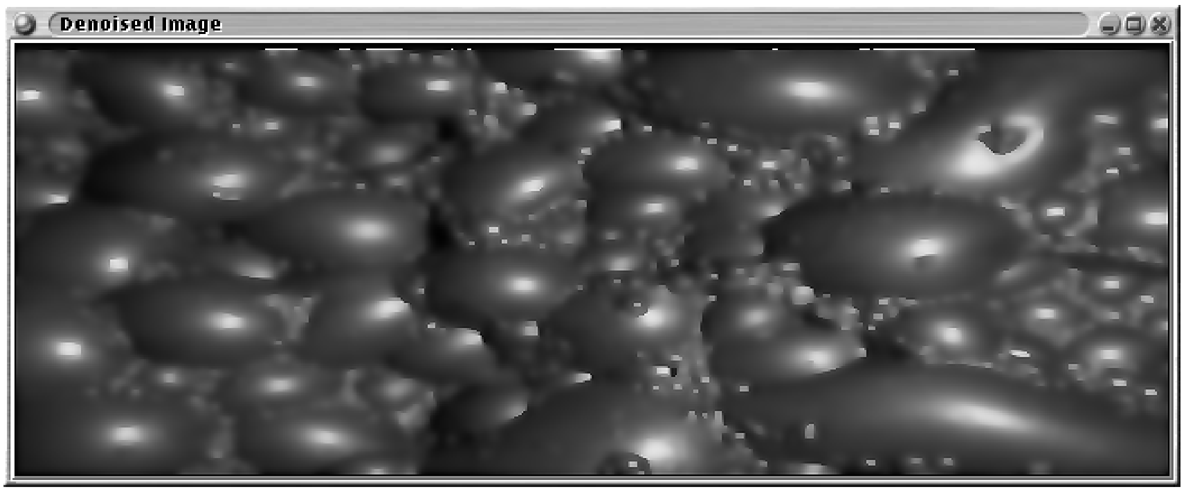
The proposed PDE is [4]:

$$I_t = g(\|G * \nabla I\|)\|\nabla I\|\nabla \cdot \frac{\nabla I}{\|\nabla I\|} \tag{7.8}$$

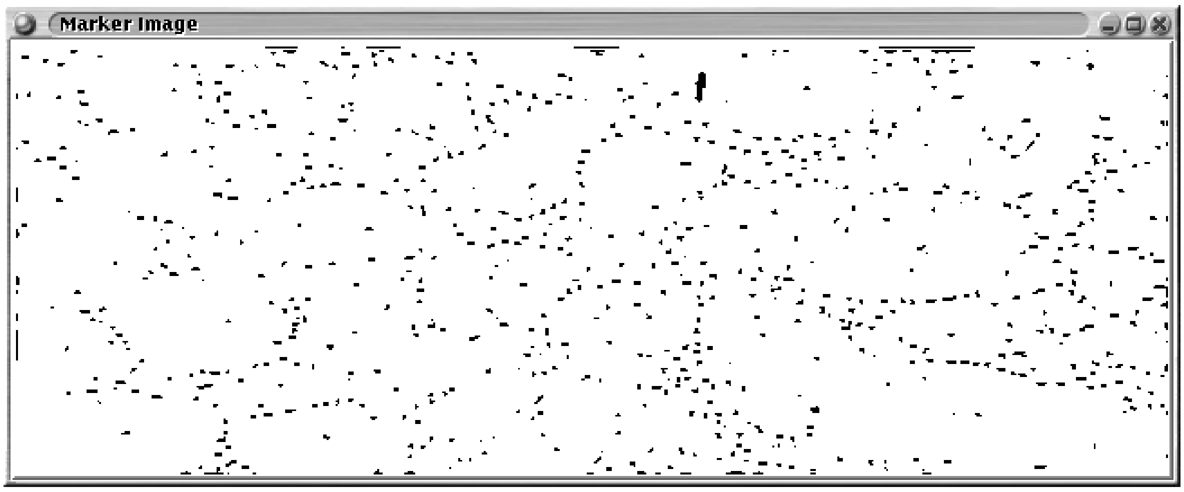
This can be regarded as a variation of curvature based diffusion. The term $\nabla \cdot \frac{\nabla I}{\|\nabla I\|}$ is the usual definition for curvature [81, 92].

7.4 Nitzberg and Shiota's reformulation of classical diffusion

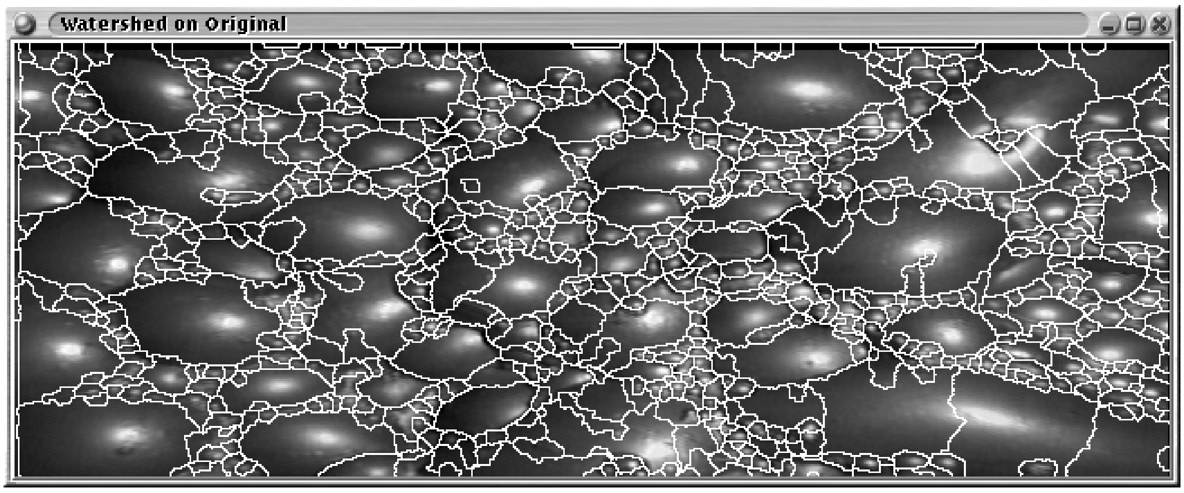
Nitzberg and Shiota express Perona and Malik's diffusion process as an adaptive filtering process [109]. They do not avoid the problems associated with classical diffusion but since the spatial extent of the adaptive filter is larger the dependency on the underlying sampling grid is reduced. Classical diffusion tends to favour edges aligned with the principle axes of the sampling grid. Alvarez et al. adopt a numerical scheme which limits the number of directions in which diffusion can occur [4].



(a) Denoised image – iteration 4.

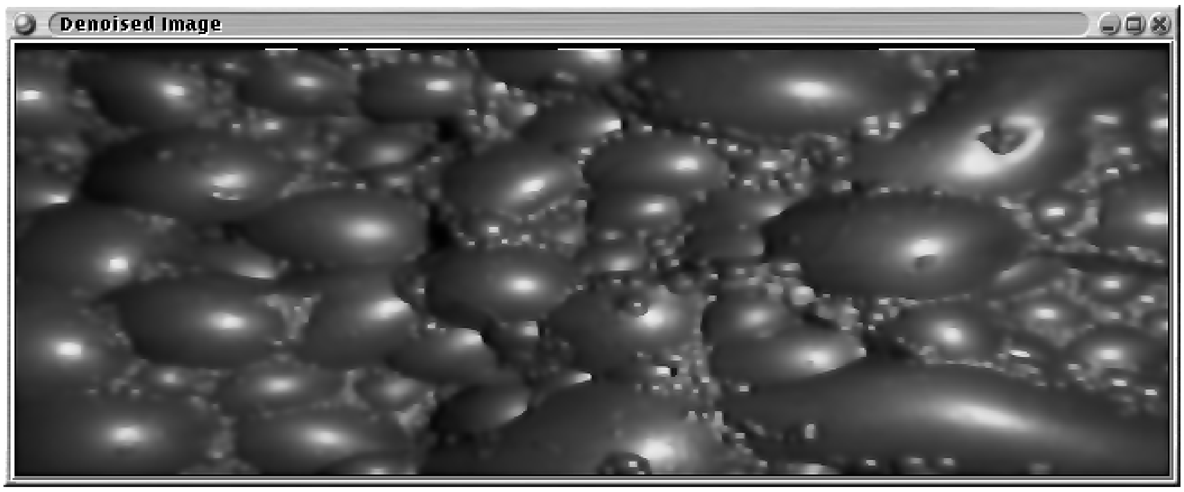


(b) Marker image – iteration 4.

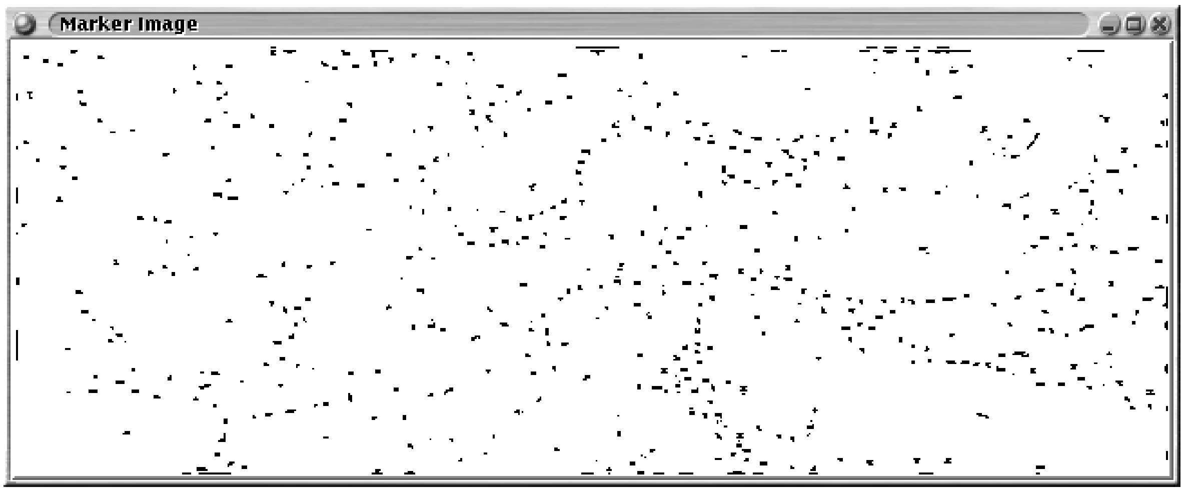


(c) Watershed image – iteration 4.

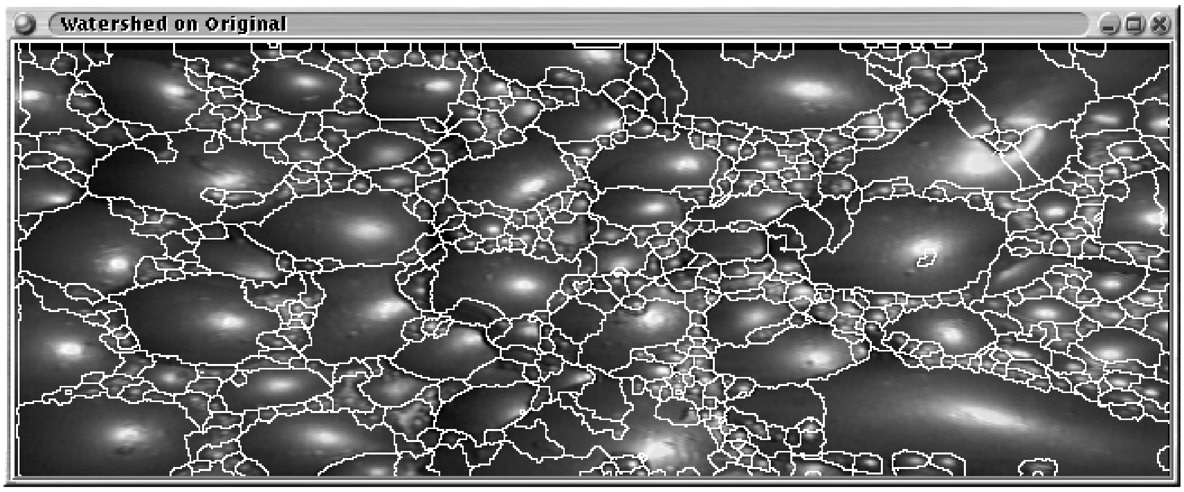
FIGURE 7.5 Denoising via Catte, Lions and Morel, $g()$ used is (7.2).



(a) Denoised image – iteration 8.

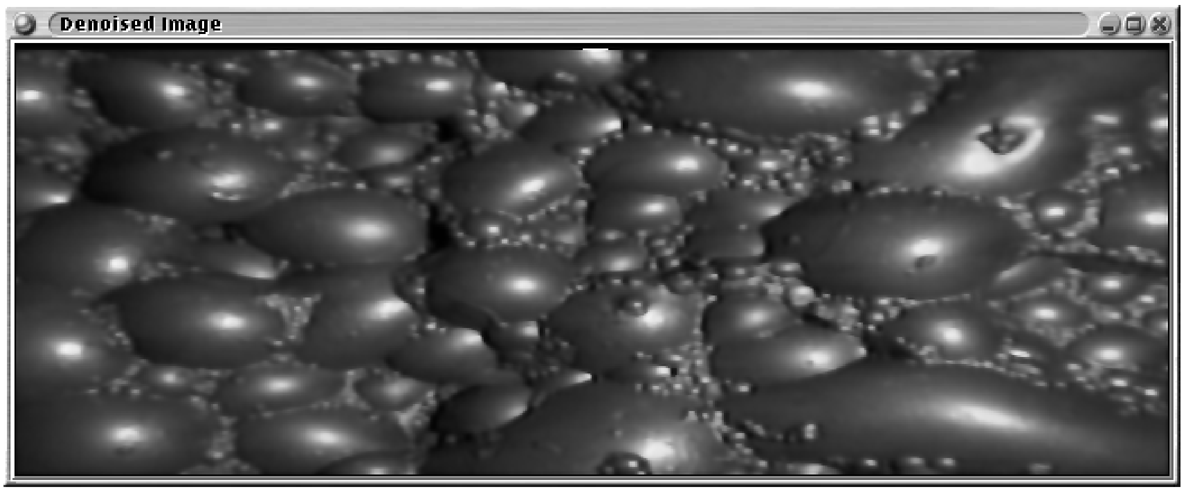


(b) Marker image – iteration 8.

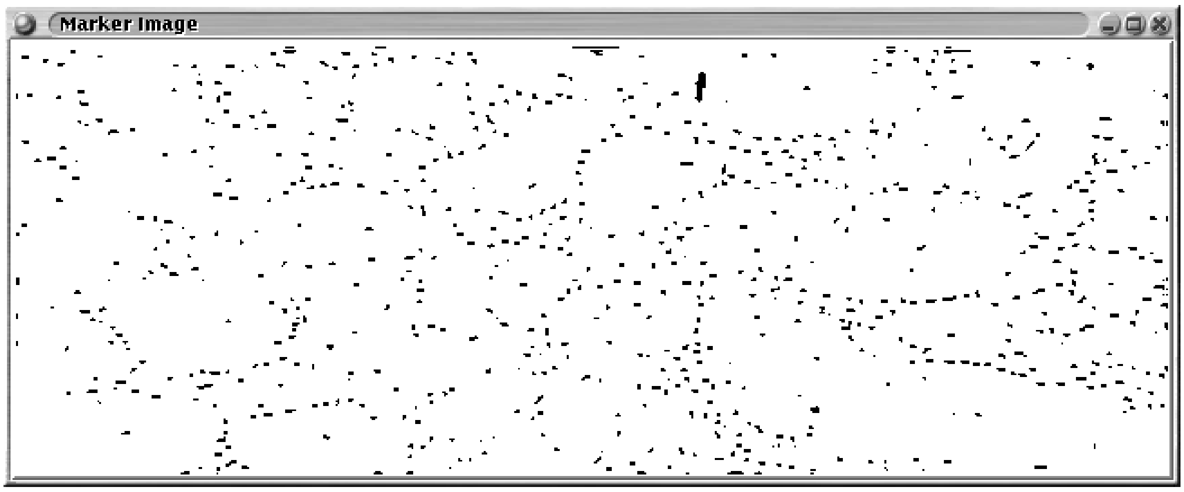


(c) Watershed image – iteration 8.

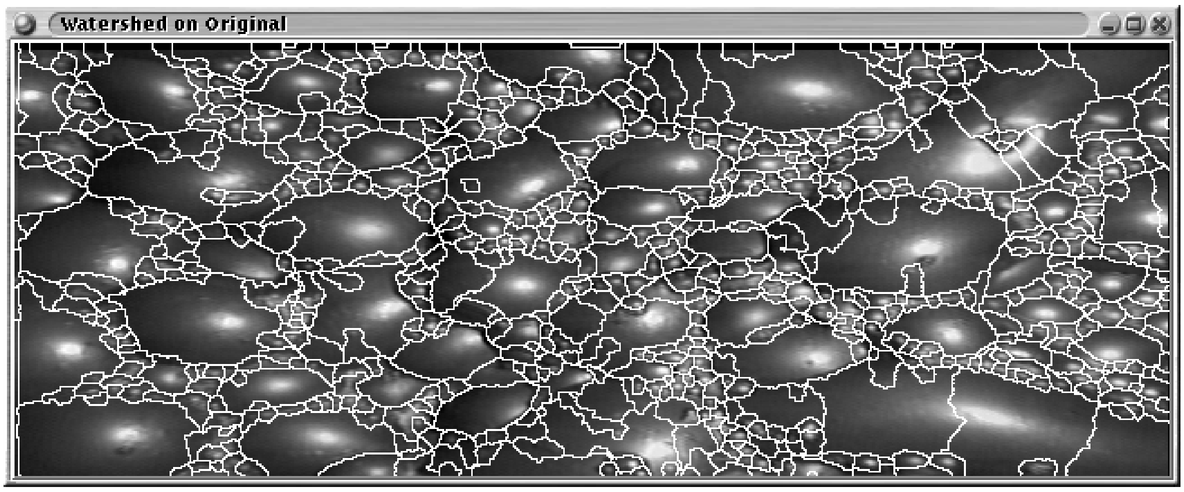
FIGURE 7.6 Denoising via Catte, Lions and Morel, $g()$ used is (7.2).



(a) Denoised image – iteration 4.

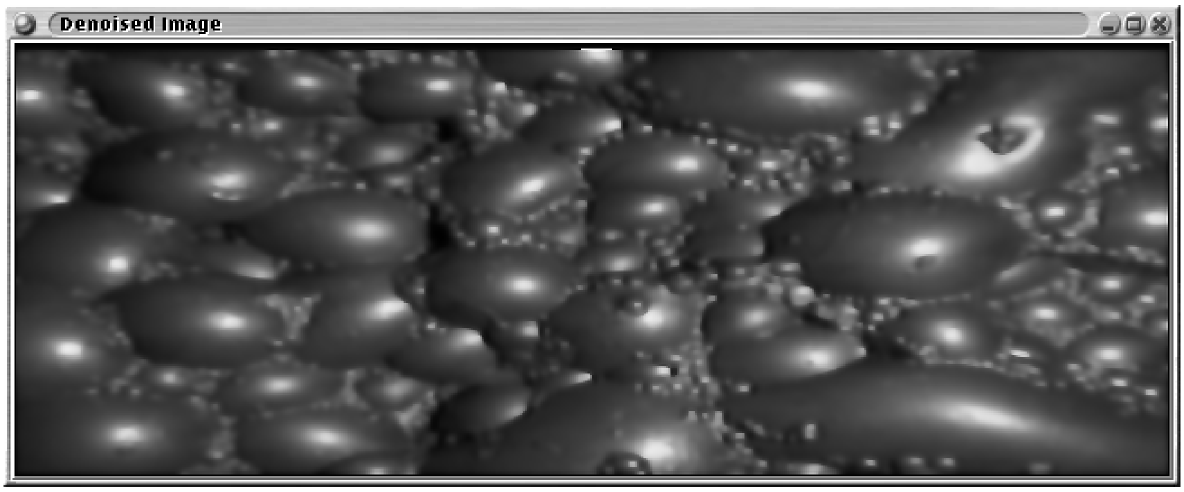


(b) Marker image – iteration 4.

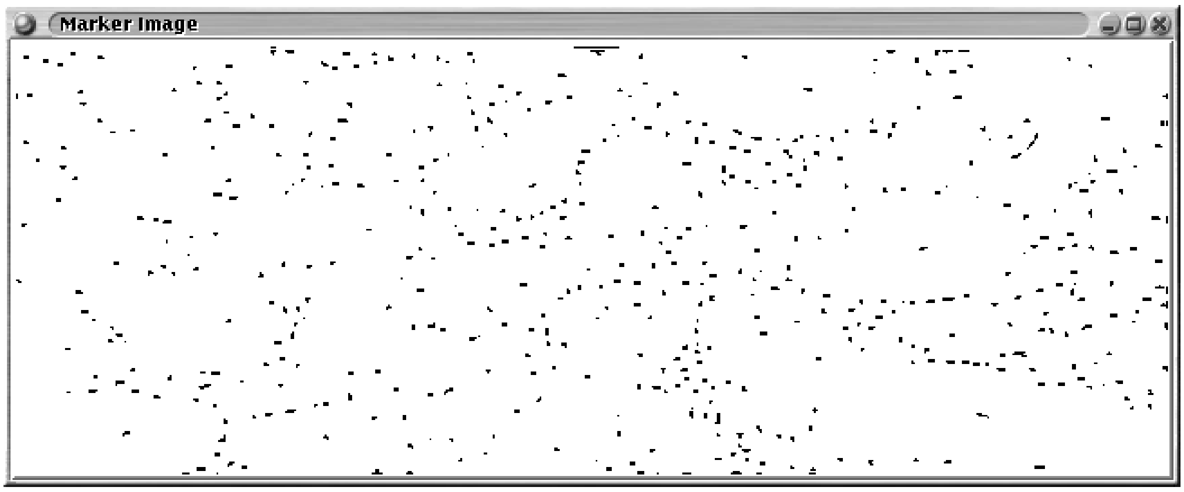


(c) Watershed image – iteration 4.

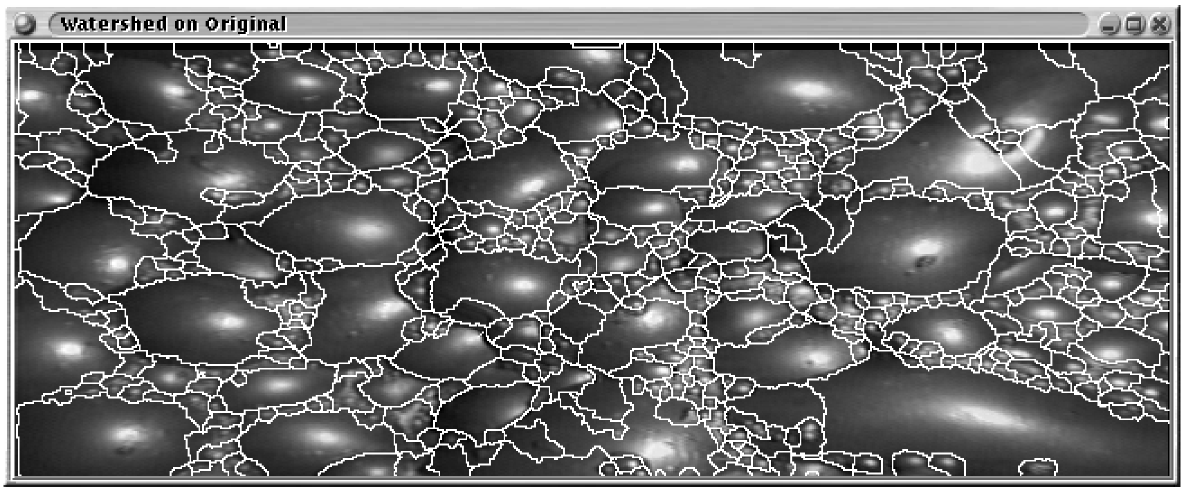
FIGURE 7.7 Denoising via Catte, Lions and Morel, $g()$ used is (7.3).



(a) Denoised image – iteration 8.



(b) Marker image – iteration 8.



(c) Watershed image – iteration 8.

FIGURE 7.8 Denoising via Catte, Lions and Morel, $g()$ used is (7.3).

Limiting the number of directions in which diffusion can occur could potentially lead to a smooth curve being replaced by polygonal sections. For smooth surfaces this is sub-optimal.

The algorithm as describes by Nitzberg and Shiota proceeds as follows,

(i) Decide on the scale of features to be maintained.

(ii) The filtered image is:

$$I_{\mathbf{x_0}} = \frac{1}{Z(\mathbf{x_0})} \int_D k(\mathbf{x_0}, \mathbf{x}) I(\mathbf{x}) d\mathbf{x} \tag{7.9}$$

(iii) $Z(\mathbf{x_0})$ is a normalising factor, $\int_D k(\mathbf{x_0}, \mathbf{x}) d\mathbf{x}$. Normalising ensures that the a pixel's value after filtering is:

$$minI \leq min_D I \leq I_{out}(\mathbf{x_0}) \leq max_D I \leq maxI \tag{7.10}$$

This ensures that the maximum principle holds.

(iv) The edge enhancement effect is created by displacing the kernel away from the edge. This reduces the blurring across the edge. The kernel displacement is designed to enhance corners and T junctions (triple points).

$$\alpha(\mathbf{x_0}) = \phi(V) \tag{7.11}$$

$$\phi(V) = \frac{c_1 V}{\sqrt{\mu^2 + |V|^2}} \tag{7.12}$$

$$V = \int_U \frac{1}{|\mathbf{y}|} \left(\nabla I(\mathbf{x}_0 + \mathbf{y}) \cdot \mathbf{y} \right) \nabla I(\mathbf{x_0} + \mathbf{y}) \rho_3(\mathbf{y}) d\mathbf{y} \tag{7.13}$$

The following is suggested for c_1,

$$c_1 \approx \frac{1}{4} \cdot \text{kernel size} \tag{7.14}$$

ρ_3 is a cut-off positive symmetric function. It is normalised to unit volume, alternatively, it can be normalised to make:

$$c = \frac{1}{d} \int |\mathbf{y}| \rho_3(\mathbf{y}) d\mathbf{y} = 1 \tag{7.15}$$

where d is the dimension of $\mathbf{y}$.

(v) The kernel shape is controlled as follows:

$$\frac{1}{Z(\mathbf{x_0})} k(\mathbf{x_0}, \mathbf{x}) = \frac{1}{Z(\mathbf{x_0})} \rho_1(\mathbf{x} - \mathbf{x_0} + \alpha(\mathbf{x_0})) e^{-Q\mathbf{x_0}(\mathbf{x} - \mathbf{x_0} + \alpha(\mathbf{x_0}))} \tag{7.16}$$

where, ρ_1 is chosen to be rotationally symmetric and have maximum amplitude (1) at $\mathbf{0}$.

$$Q\mathbf{x_0} = \frac{1}{\sigma^2}\begin{pmatrix} E & F \\ F & G \end{pmatrix}\mathbf{x_0} \tag{7.17}$$

where:

$$E = \int_U I_x(\mathbf{x})^2 \rho_2(\mathbf{x} - \mathbf{x_0}) d\mathbf{x} \tag{7.18}$$

$$F = \int_U I_x(\mathbf{x}) I_y(\mathbf{x}) \rho_2(\mathbf{x} - \mathbf{x_0}) d\mathbf{x} \tag{7.19}$$

$$G = \int_U I_y(\mathbf{x})^2 \rho_2(\mathbf{x} - \mathbf{x_0}) d\mathbf{x} \tag{7.20}$$

Again, ρ_2 is a smooth nonnegative function of unit volume which gives a higher weight to gradient estimates close to the centre of the kernel ($\mathbf{x_0}$). ρ_2 sets the effective window size of the filtering process by controlling the smoothing of the gradient estimates.

7.5 Fischl and Schwartz – Optimisations

Fischl and Schwartz propose several optimisations to the anisotropic diffusion process [42, 41]. The following sub-sections examine the optimisations in detail.

7.5.1 Anisotropic diffusion in the complex log domain

Fischl and Schwartz propose diffusion in the complex log domain to improve the speed of the diffusion process [42]. The complex log mapping domain is specified as follows:

$$z = re^{\jmath\theta} = x + \jmath y \tag{7.21}$$

$$r = \sqrt{x^2 + y^2} \tag{7.22}$$

$$\theta = \tan^{-1}\left(\frac{y}{x}\right) \tag{7.23}$$

the mapping used is:

$$w = \begin{cases} \log(z + a) & Re(z) \geq 0, a\epsilon\mathbb{R} \\ 2\log(a) - \log(-z + a) & Re(z) < 0 \end{cases} \tag{7.24}$$

$$w = \rho(z) + j\phi(z) \tag{7.25}$$

The constraint for stability is as follows:

$$\Delta t \leq \frac{(\Delta x)^2}{4c}, \qquad \text{Cartesian co-ordinates [113].} \qquad (7.26)$$

$$\Delta t \leq \frac{e^{2\rho}}{4c}, \qquad \text{Complex log co-ordinates [42].} \qquad (7.27)$$

This implies that at the periphery of the complex log image (large ρ) large time steps are possible. This implies that a scheme is possible to diffuse using large time steps at the periphery of the complex log image and then reduce the time steps as one approaches the origin of the mapping. Fischl and Schwartz propose a scheme [42] where the diffusion converges rapidly at the periphery. The unequal rates of convergence across the image implies that for each iteration, fewer pixels need to be considered. This leads to a reduction in processing time.

7.5.2 Approximating the Green's function

This is a modification of Nitzberg and Shiota's formulation to improve computational performance. Fischl and Schwartz [41] propose a separation of the offset vector field calculations and the filtering process.

The Green's function (the convolution kernel of a filter is an example of a Green's function [75]) is approximated for the image and Perona and Malik's anisotropic diffusion. The algorithm attempts to estimate $G()$ in the following integral:

$$I_t = \iint du\, dv\, G_t(x, u, y, v) I(x, y, 0) \qquad (7.28)$$

Different kernels, G, exist at different places and times in the image.

The algorithm proceeds as follows:

(i) A position-time modified kernel is estimated for every point in the image.

(ii) The diffused resultant image is derived by convolution of the position-time modified kernel as shown in equation (7.28).

The authors propose an iterative update equation [41][pg 345,346]. The actual construction of the kernels is performed by a modified back-propagation neural net. The neural net then finds the kernels which when applied to a training image yields a result equivalent to applying Perona and Malik's [113] algorithm for a specified number of iterations, time step and so on.

Offset vector filtering

Here an offset vector field is generated. An arbitrary filter is applied to the image. A pixel in the result image is replaced by the filtered image pixel to which the offset vector points. Fishl and Schwartz propose this variation of offset vector filtering to overcome certain difficulties of Nitzberg and Shiota's (see [109]) formulation.

The process occurs as follows:

(i) The orientation is computed [40][pg 10]. The image is first smoothed using a Gaussian kernel with $\sigma = 2$. The gradients are computed using Sobel filters for the x and y directions.

$$O(z) = \int_W s\left(\nabla I(z+z')\right)\nabla I(z+z')dz', \tag{7.29}$$

$$s(z) = \begin{cases} -1, & x < 0 \\ 1, & x \leq 0 \end{cases} \tag{7.30}$$

where,

W a window, may be 3×3.

z' is the vector from z to a point in W.

$s(z)$ limits the orientation of O(z). This allows "uphill' and "downhill" orientation along the gradient only.

(ii) The sign,

$$d(z) = \operatorname{sgn}\left(\int_W |O(z)\cdot\nabla I(z+z')|(O(z)\cdot z')dz'\right) \tag{7.31}$$

(iii) The magnitude,

$$v_i(z) = d(z)O(z) \tag{7.32}$$

$$m(z) = \min_\alpha\{(v_i(z+\alpha v_i(z)))\cdot v_i(z) \leq 0\} \tag{7.33}$$

finally,

$$v(z) = m(z)\frac{v_i(z)}{|v_i(z)|} \tag{7.34}$$

The nature of the process estimating the magnitude of the final offset vector preserves small scale structures. Fischl and Schwartz demonstrate [40][pg. 11, 12] that the formulation of Nitzberg and Shiota (see [109] and section 7.4) does not, under certain conditions, preserve small scale structures.

7.5.3 Discussion

The approach of building kernels to replicate the diffusion of an existing diffusion algorithm leads to the flaws of the diffusion algorithm to be replicated as well. For example, it is known that Perona and Malik's algorithm preferentially preserves edges aligned with the sampling grid axes [109][fig. 1, pg 826]. If the parameters change, for example, number of iterations, then the neural net needs to be retrained.

There are some objections to the offset vector filtering approach as described above:

(i) Scale. The Gaussian smoothing sets the scale of the smallest structures in the image. How should the width of the Gaussian be chosen for different types of images?

(ii) Edge estimates are known to be sensitive to noise [19][pg 183]. The offset vector field will then be sensitive to noise.

(iii) Smooth slowly varying regions will *not* be preserved. Consider the following degenerate case of an image consisting of piecewise regions of constant intensity gradient. The gradients are allowed to vary in orientation and magnitude. Consider one such region. Consider a point away from the borders of this region. The orientation is constant everywhere except at the borders. The sign is constant everywhere except at the borders. When the magnitude of the offset vector at an interior point is computed, the only point at which sign and orientation is different will be at the border or within a neighbouring region (looking along a line from an interior point along the orientation vector at this point). After filtering then each interior point's intensity is replaced by the intensity of the border point. The constant slope regions are replaced by piecewise constant regions *irrespective* of the filter used. This is itself visible in Fischl and Schwartz's paper [40][fig 6.4, pg 12]. This makes the utility of offset vector filtering for anisotropic diffusion suspect. It may, however, be useful for shock filters (see the following for an introduction to shock filters [5, 112]).

(iv) Consider an image consisting of an intensity ramp from zero to some positive intensity, a constant region and another ramp sloping back to zero intensity. Following to the end of the image is a zero intensity region. Without loss of generality, let the gradient of the first ramp be $(1, 0)$, and than that of the second be $(-1, 0)$.

The offset vector field is computed as specified by Fischl and Schwartz [40] above. The initial orientation, $O(z)$, of the first ramp is $(1, 0)$. The sign, $d(z)$, 1. The initial offset field, $v_i(z)$, is $(1, 0)$. The length of each vector is simply the distance from a point on the ramp to the first pixel in direction $(1, 0)$ on the constant intensity region. In other words, after filtering, every point on the ramp has the same value as the first pixel on the filtered constant intensity region. The ramp then disappears.

In the same way, the second ramp disappears. The ramp is replaced by the filtered value of the first pixel on the zero intensity region. Note that $S(z)$, reverses the direction of $O(z)$ from $(-1, 0)$ to $(1, 0)$.

The edge positions are changed. This illustrates a drawback of offset vector filtering.

7.6 Robust anisotropic diffusion

Black, Sapiro, Marimont and Heeger show anisotropic diffusion can be described in terms of robust estimation [15]. Specifically, Black et al. show a relationship between robust error norms and the gradient weighting functions $g()$ proposed by Perona and Malik. Black et al. propose the following:

(i) A robust method for computing the magnitude of significant edges to be preserved.

(ii) A new class of functions for $g()$.

Perona and Malik use Canny's noise estimator to gauge the magnitude of significant edges in an image [113]. Black et al. propose using the median absolute deviation, defined as follows:

$$\sigma_e = 1.4826 \, \text{median}_I \left(\|\nabla I - \text{median}_I(\|\nabla I\|)\| \right) \tag{7.35}$$

The functions for $g()$ proposed by Perona and Malik [113] are non-zero even for large edge magnitudes. This implies that even very strong edges are diffused and will eventually decay, this may take hundreds of iterations however. Black et al. propose functions which are zero above certain gradient magnitudes, this implies that strong edges are preserved always and that the diffusion process converges to a steady state faster. Black et al. propose Tukey's biweight function (although others are possible):

$$g(x,\sigma) = \begin{cases} \frac{1}{2}\left(1-\left(\frac{x}{\sigma}\right)^2\right) & ,|x| \le \sigma \\ 0 & \text{,otherwise} \end{cases} \tag{7.36}$$

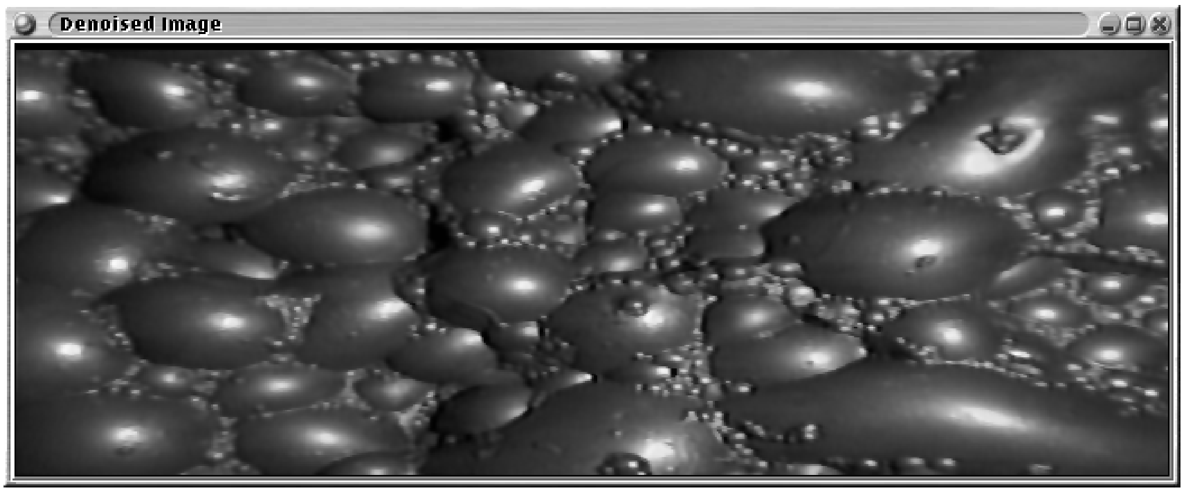
σ may be computed using Median Absolute Difference (or in the Perona and Malik case, Canny's noise estimator).

7.6.1 Discussion

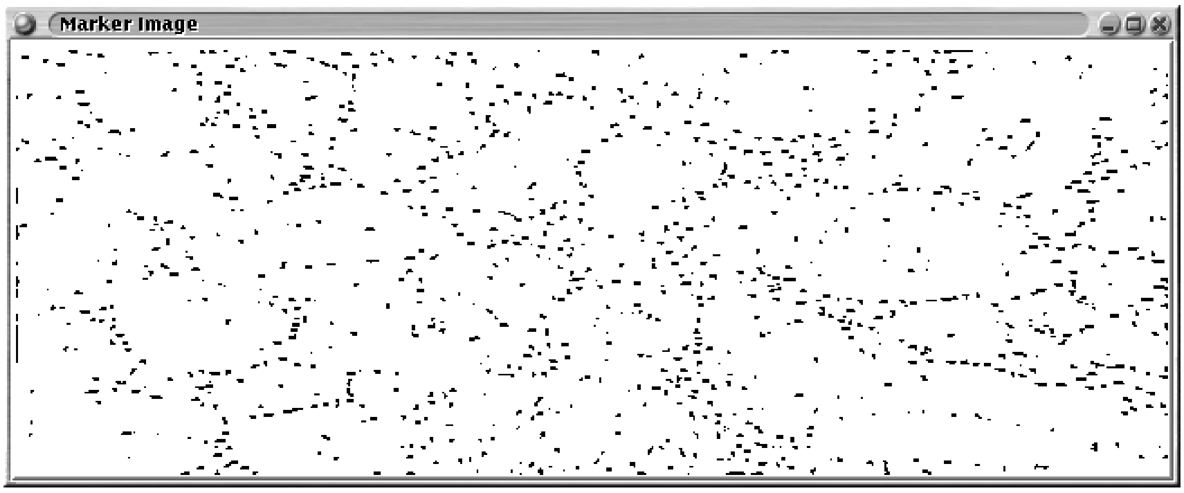
Table 7.3 shows the decrease in the number of watershed regions and hence minima as the number of iterations increases. Tukey's biweight function is used as $g()$ and the time step used is $\gamma = 0.2$. Both Perona and Malik's, as well as Catte et als. formulation of diffusion is used. The discretisation used for both is Perona and Malik's discretisation. This allows the results to be directly compared without needing to compensate for differences in discretisation. The value σ is computed using the Median of Absolute Differences.

Figures 7.9 through 7.12 show the effects of Tukey's BiWeight function applied to Perona and Malik's, and Catte et als. framework four and eight times respectively.

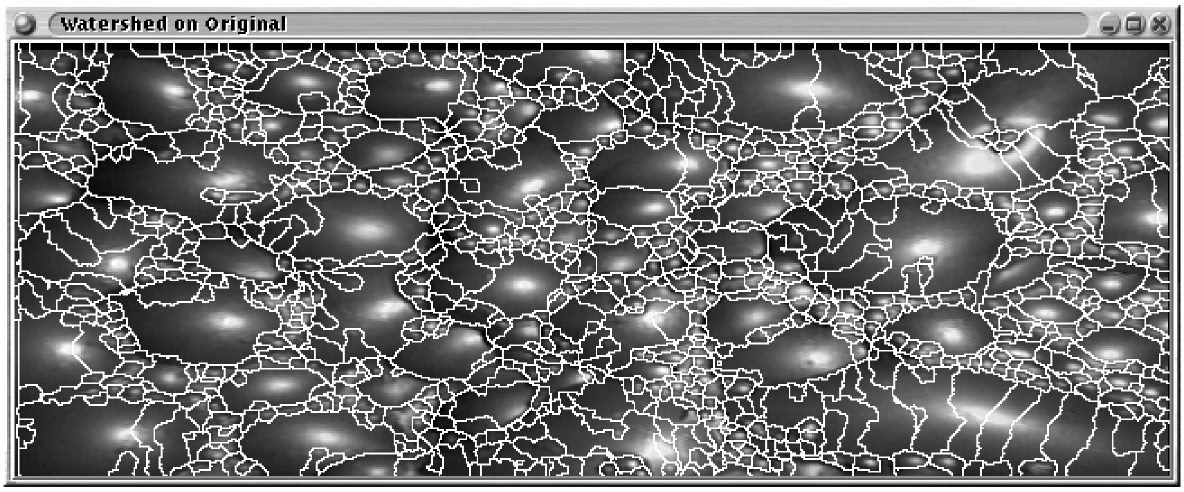
Noisy edges, where the edge strength is high, are not smoothed. The edge preservation property Tukey's BiWeight function implies more iterations are needed to achieve the same level of denoising as the other $g()$ functions proposed by Perona and Malik. Note that small bubbles are well preserved, even at high iterations when over-segmentation is reduced.



(a) Denoised image – iteration 4.

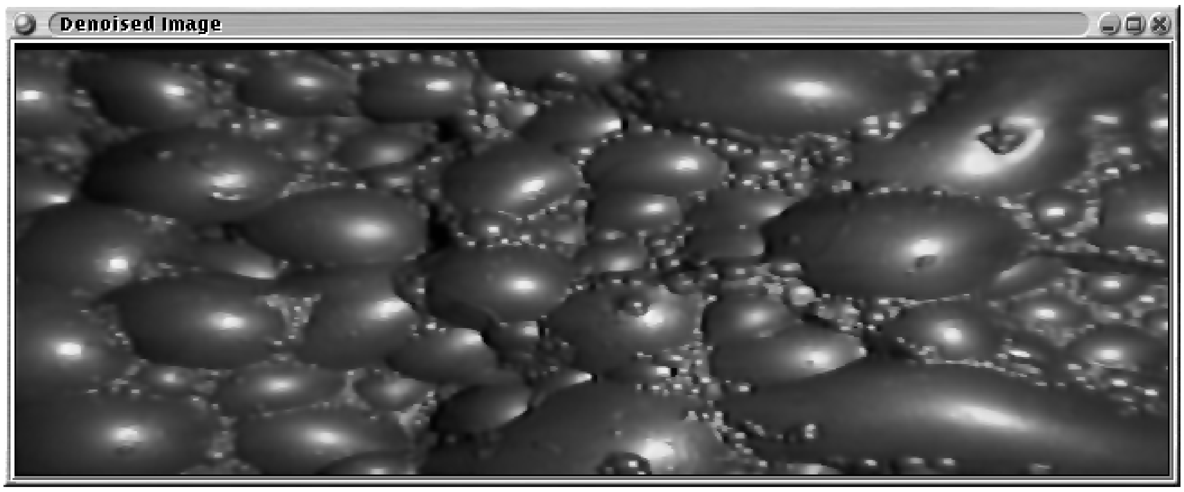


(b) Marker image – iteration 4.

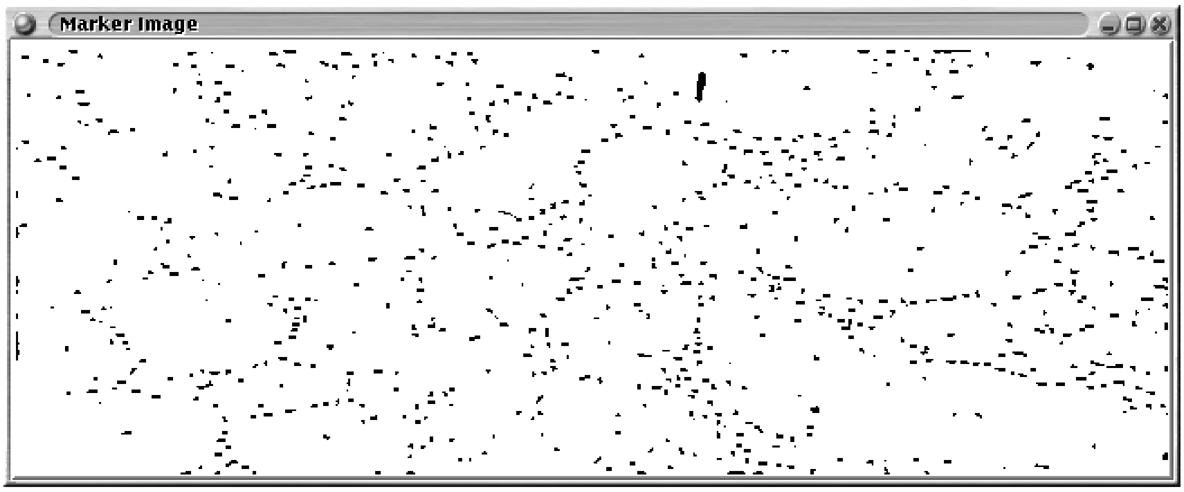


(c) Watershed image – iteration 4.

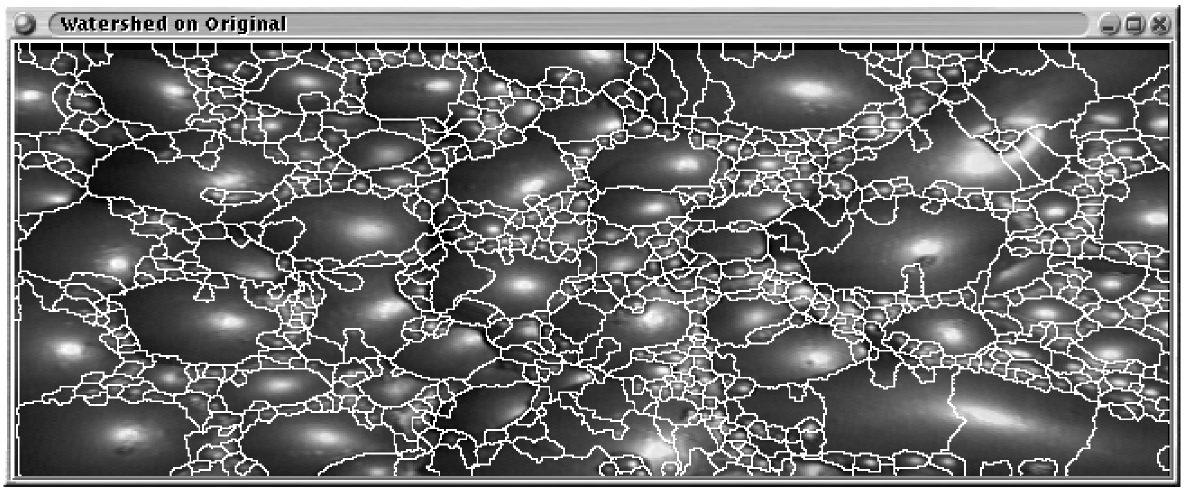
FIGURE 7.9 Denoising via Perona and Malik, $g()$ used is (7.36) (Tukey's BiWeight), 4 iterations.



(a) Denoised image – iteration 8.

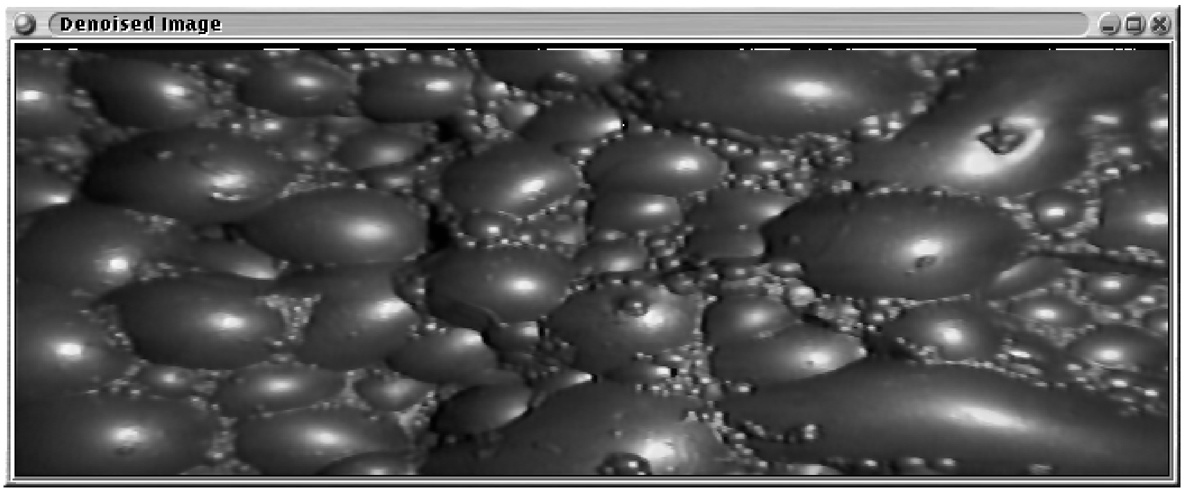


(b) Marker image – iteration 8.

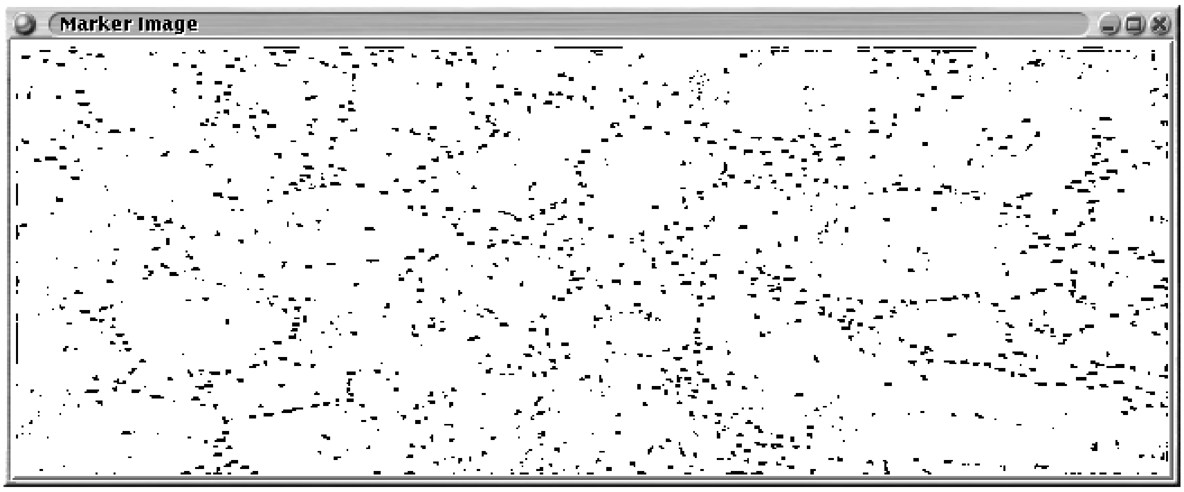


(c) Watershed image – iteration 8.

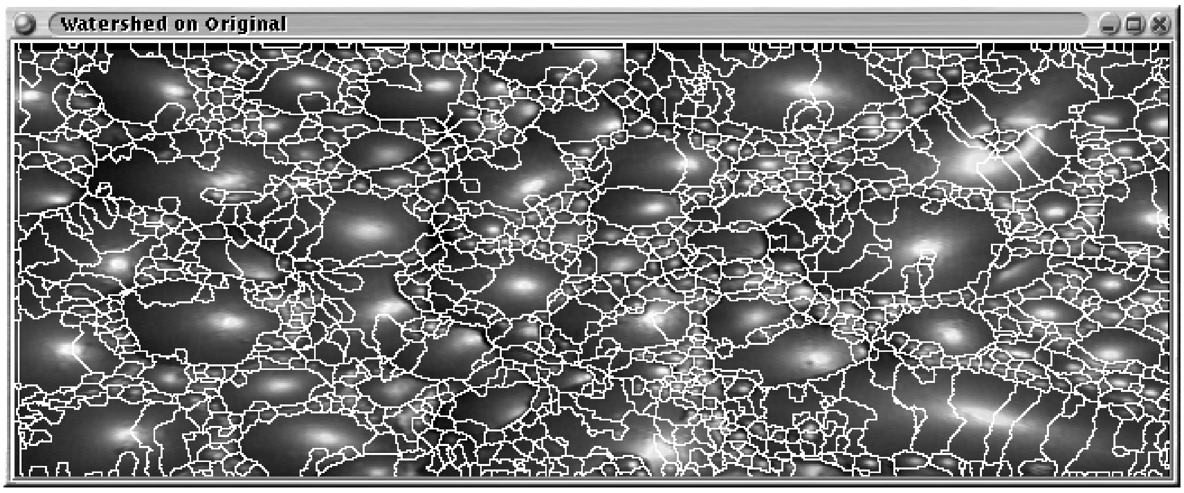
FIGURE 7.10 Denoising via Perona and Malik, $g()$ used is (7.36) (Tukey's BiWeight).



(a) Denoised image – iteration 4.

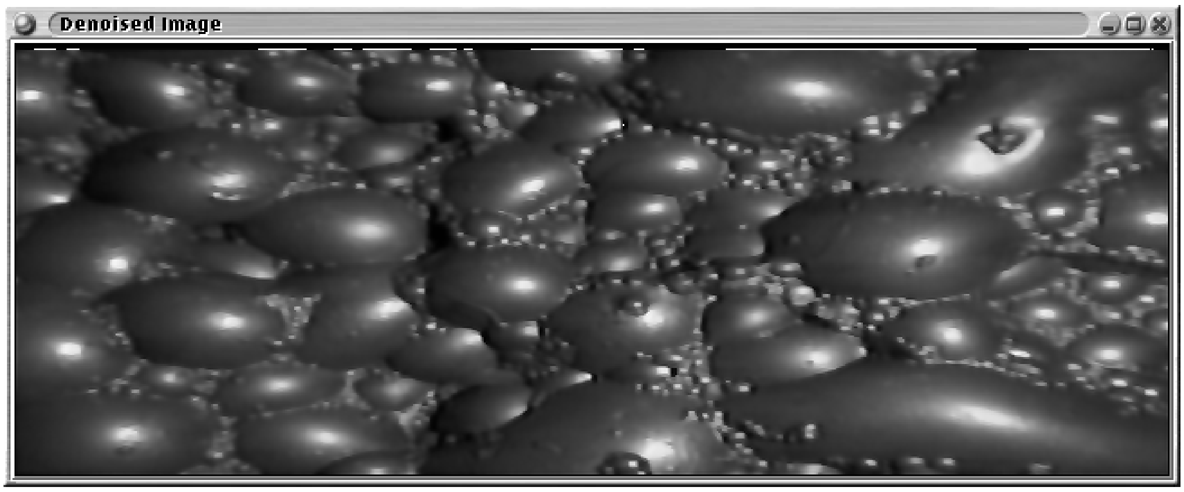


(b) Marker image – iteration 4.

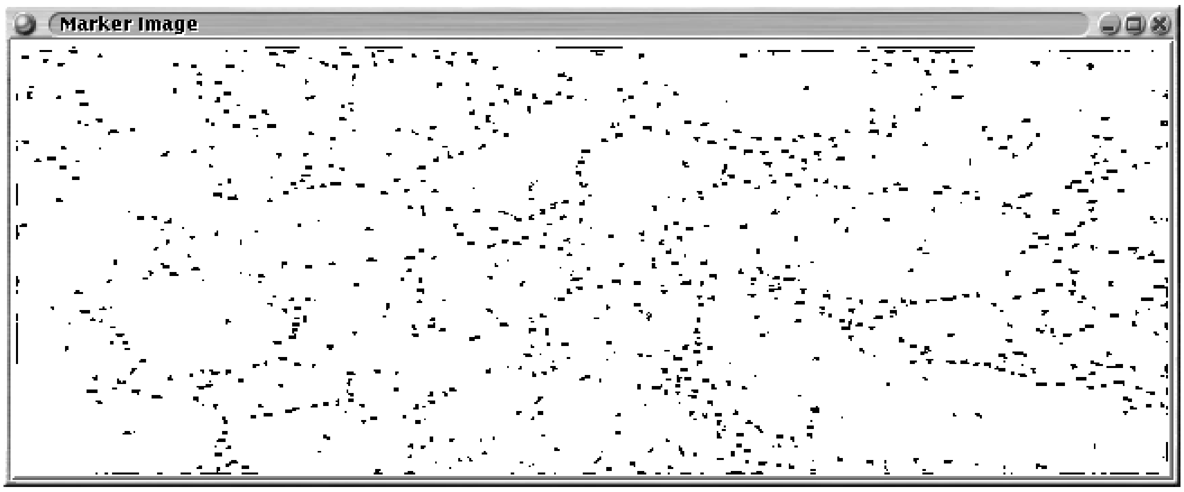


(c) Watershed image – iteration 4.

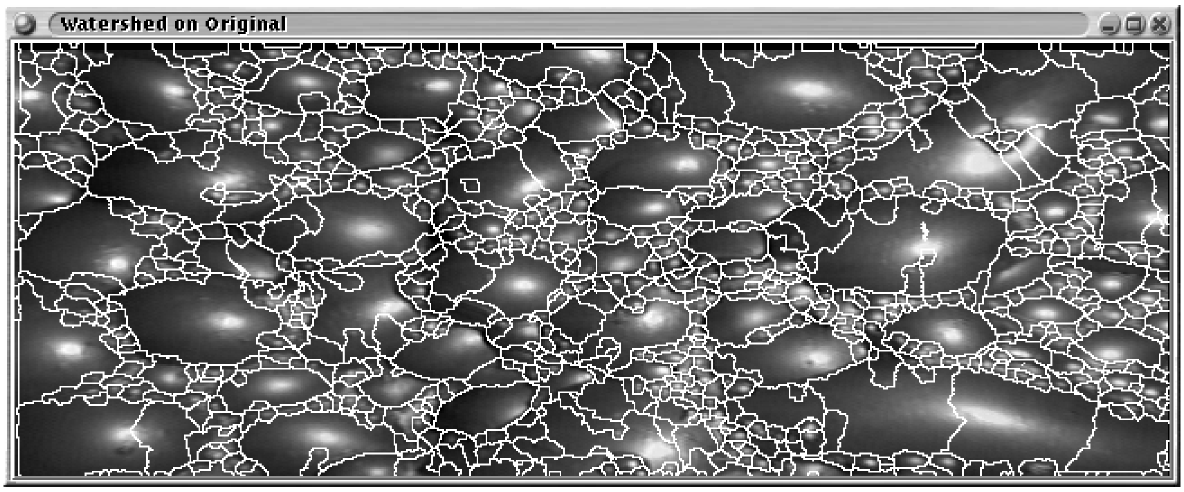
FIGURE 7.11 Denoising via Catte et al., $g()$ used is (7.36) (Tukey's BiWeight), 4 iterations.



(a) Denoised image – iteration 8.



(b) Marker image – iteration 8.



(c) Watershed image – iteration 8.

FIGURE 7.12 Denoising via Catte et al., $g()$ used is (7.36) (Tukey's BiWeight).

TABLE 7.3 Decrease in the number of watershed regions as the number of iterations increases.

Iterations	# Watershed Regions	
	Perona Malik	Catte et al.
1	2017	2660
2	1418	1933
3	1156	1509
4	1010	1261
8	750	878
16	573	709
20	528	703
24	500	682

7.7 Anisotropic diffusion without edge enhancement

It is well known that the backwards heat diffusion is ill-posed. Removal of the backwards heat diffusion component of classical Anisotropic diffusion as proposed by Perona and Malik[113] (see section 7.1) is proposed. This allows for the relative importance of edge enhancement to be gauged, at least for small number of iterations. The new diffusion equation is then:

$$I_t = g(\|\nabla I\|)\nabla \cdot \nabla I \tag{7.37}$$

The discretisation scheme used (on a four connected neighbourhood, using central differences for edges) is:

$$\text{Laplacian}(I) = I_n^k + I_s^k + I_e^k + I_w^k - 4 * I_c^k \tag{7.38}$$

$$\text{Edge} = \frac{1}{2}\sqrt{(I_e^k - I_w^k)^2 + (I_n^k - I_s^k)^2} \tag{7.39}$$

$$I_c^{k+1} = g(\|\nabla\text{Edge}(I)\|)\text{Laplacian}(I) \tag{7.40}$$

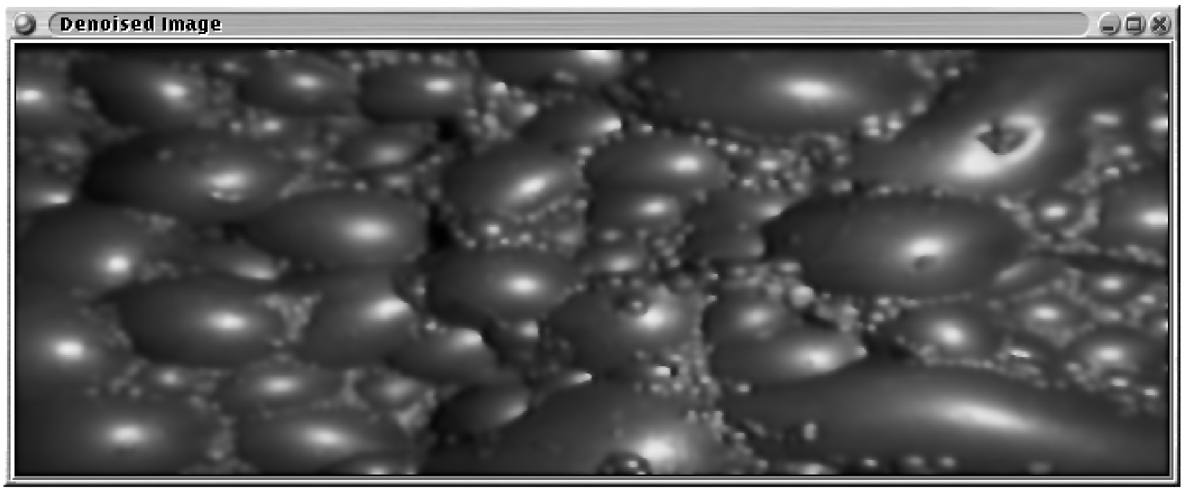
where, I_n^k denotes the pixel at $(0, 1)$ relative to I_c. Similarly for the other pixels in the other directions.

Table 7.4 shows the decrease in the number of watershed regions (and hence maxima in the image) as the number of iterations of equation 7.37 increases. Note that fewer regions are evident, than for the edge enhancement cases (see tables 7.1 and 7.2 for the Perona and Malik and Catte et al. cases). As expected the denoised images show greater blurring of edges.

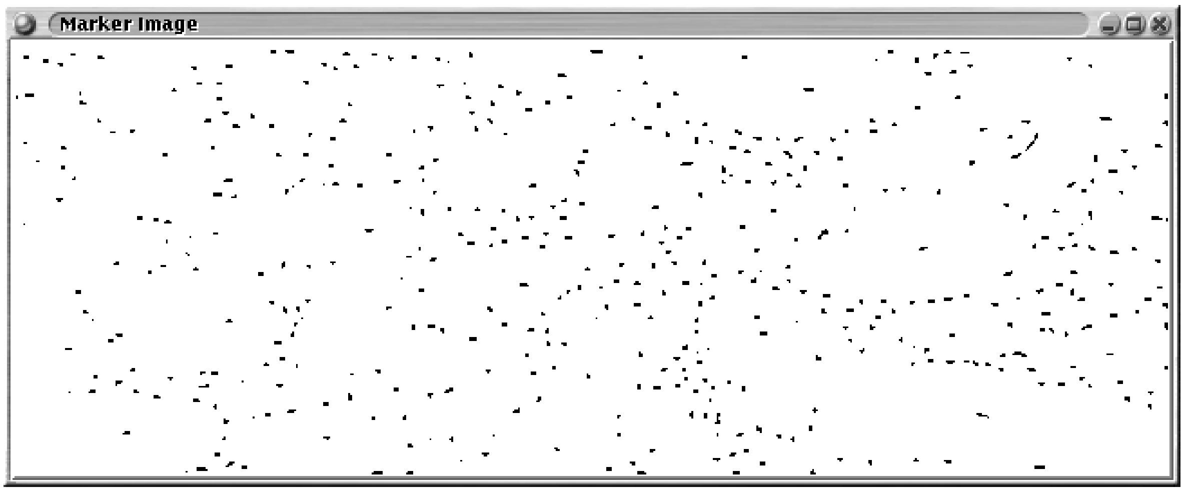
7.8 A comparison of the different Anisotropic Diffusion techniques

The following needs to be said:

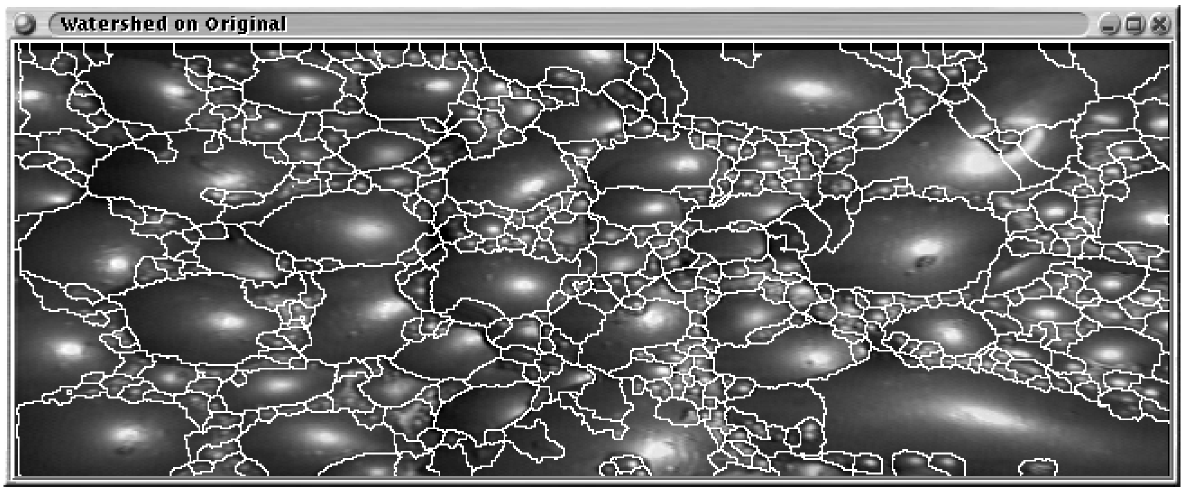
(i) The ill-posed nature of the backwards heat diffusion equation is aggravated by noise, within a



(a) Denoised image – iteration 8.

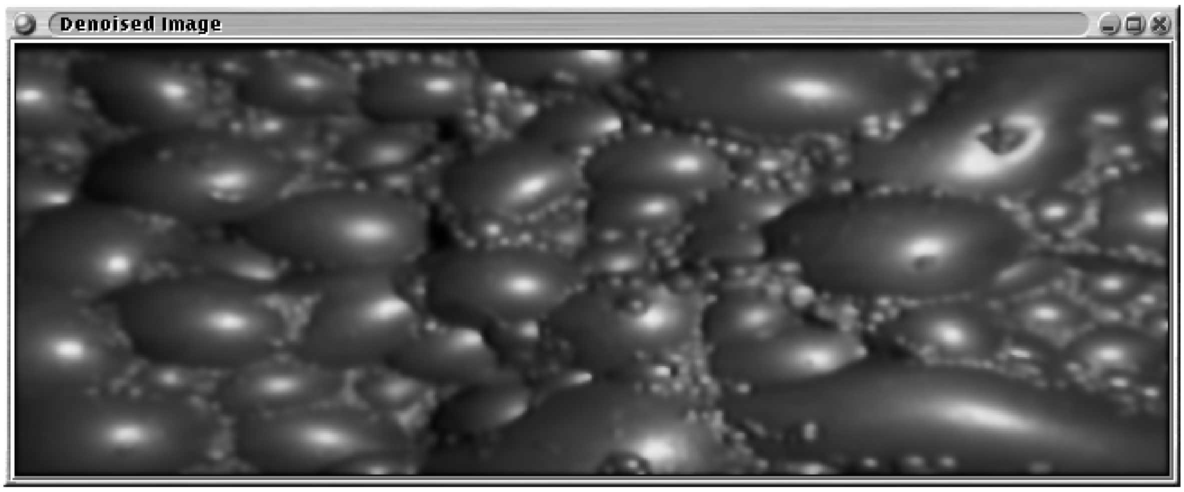


(b) Marker image – iteration 8.

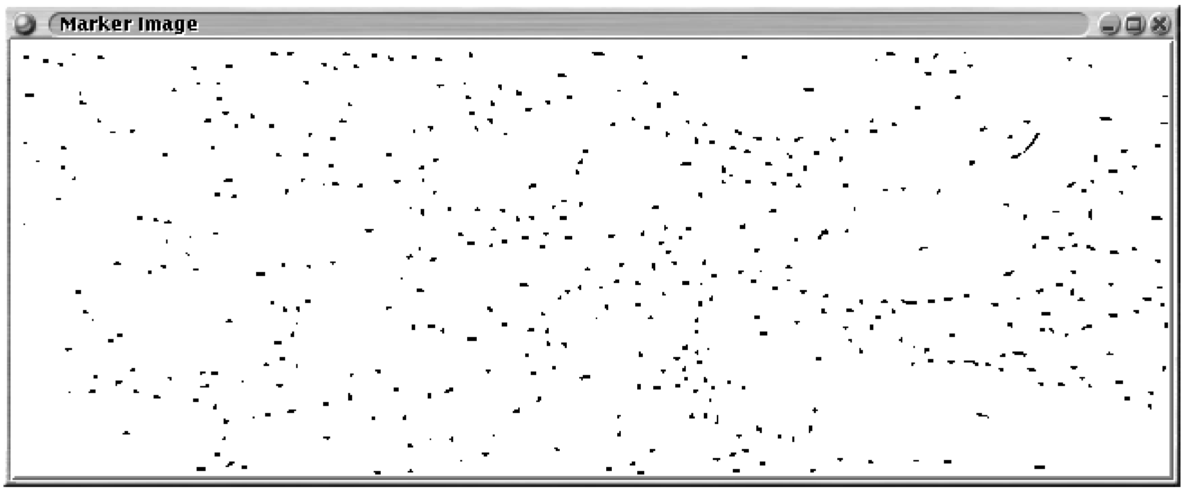


(c) Watershed image – iteration 8.

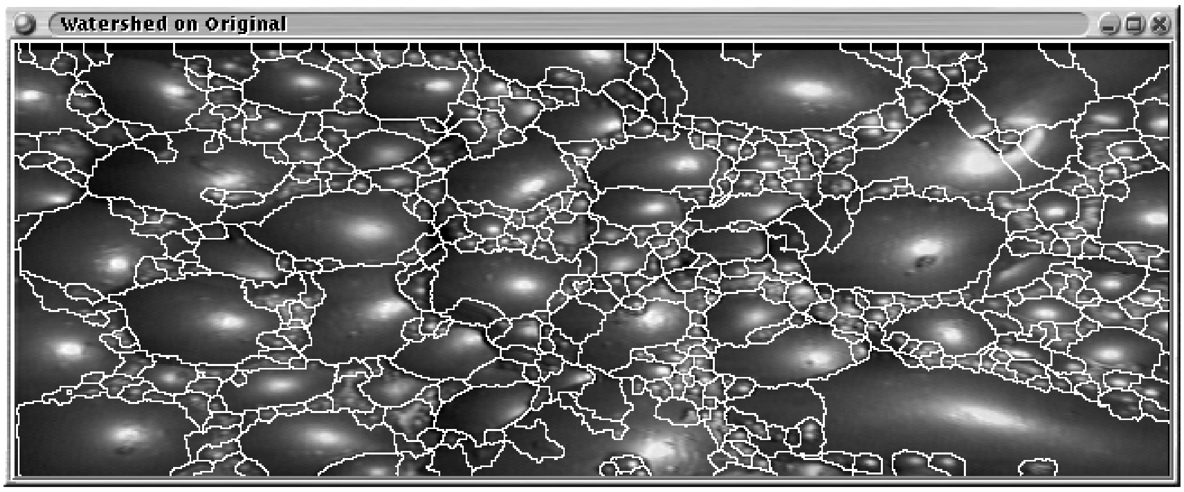
FIGURE 7.13 Diffusion without edge enhancement, $g()$ used is (7.2).



(a) Denoised image – iteration 8.



(b) Marker image – iteration 8.



(c) Watershed image – iteration 8.

FIGURE 7.14 Diffusion without edge enhancement, $g()$ used is (7.3).

TABLE 7.4 Decease in the number of watershed regions as the number of iterations increases.

Iterations	# Watershed Regions	
	$g(x) = e^{-\lvert\nabla I\rvert/K^2}$	$g(x) = \frac{1}{1+\left(\frac{\lvert\nabla I\rvert}{K}\right)^2}$
1	1224	1223
2	890	885
3	748	743
4	655	649
8	460	450
16	309	296

local neighbourhood system the few pixels under consideration do not provide sufficient information to deconvolve the edge, and the difficulty of de-convolving some types of blurring (this is especially visible in the frequency domain where some convolution kernels have zeros, at the zeros information is discarded and cannot be reconstructed). In effect, backwards heat diffusion (or anisotropic edge enhancement) is equivalent to deconvolution.

(ii) The whole anisotropic diffusion approach is weakened by the enhancement of edges by backwards diffusion, unless the edges are all less noisy and well sampled. This problem is partially addressed by convolving the image with a Gaussian, before edge estimates are taken (Catte et als. approach [19]).

(iii) The digital sampling grid imposes stair-casing in the final grey levels stored. This implies that even without noise, any of these “stairs” could potentially give rise to spurious detail if they happen on an edge strong enough and given enough iterations . This effect can be alleviated by ensuring a fine enough sample grid. In other words strict adherence to the Nyquist sampling principle, application of an anti-aliasing filter (the anti-aliasing approach is used by Perona and Malik), and limiting the number of iterations.

(iv) Even if edge enhancement were prohibited, anisotropic diffusion still provides reasonable denoising without blurring significant edges.

Chapter 8

Median Filter based Denoising

Median filters belong to a more general family of filters known as Stack Filters or Order Statistic Filters [151]. Median filters, by virtue of being nonlinear, have the advantage of being, to some degree, edge preserving [125].

A standard froth image is median filtered using one of the median filter types and then segmented via a marker based watershed segmentation. The markers are extracted by eroding the median filtered image with a rolling ball structuring element (of radius one pixel) and then greyscale reconstructing by dilation. The difference between the filtered and reconstructed image, after thresholding, yields the markers (this marker extraction process is described in detail in part I, chapter 4). The objective is to determine the suitability of applying median filters to froth images. The method used for comparisons relies on an observer rating the varieties of median filter and is hence qualitative. The criteria for comparison is based on the goodness of the segmentation and the effect on the marker and denoised images as iterations and algorithm type varies. Quantitative comparisons will be presented in chapter 12 for a variety of denoising algorithms.

Both standard and weighted median filters are tested. Weighted median filtering covers line and corner preserving filters. The following filters were examined: corner preserving; corner and horizontal and vertical line preserving; and horizontal, vertical and diagonal line preserving masks. In essence, a median filter (of any type) sorts the pixel intensities on a region into order. The output value selected is the half population point or the $N/2$th value in the sorted list (N is the number of elements in the list, or number of pixels in the region considered). This is correct when the number of pixels within the region is odd. The situation when the number of pixels is even is more complicated, essentially a new pixel is inserted into the sorted list so that the number of pixels on either side is the same. The intensity of this inserted pixel is the average of its two immediate neighbours, in other words, the median [151]. This is not used here. The middle pixel in the sorted list is chosen as the median. This ensures that the median value used is one which occurs within the input image. It should be noted that efficient algorithms for computing medians do exist (see [88] for example) which do not use sorting techniques. For small numbers of pixels, however, the computational savings achieved do not repay the added algorithmic complexity. The mask values, for weighted median filters, are weights which imply that the pixel value at that weight is repeated so many times as it is placed into the list of pixel values from which the median is computed.

8.1 Standard Median Filtering

The mask used is described in table 8.1 [125], figures 8.1 through 8.3 describe the effect of applying the filter iteratively to the test image up to four times. Table 8.2 records the decrease in the number of watershed regions (and associated image maxima) as the number of iterations increases.

One iteration did not denoise enough and more iterations denoised too much. Notice that as the number of iterations increases the size of the surviving markers increases. Notice also that some of the markers at ten iterations have a blocky and angular shape, this is a result of the small size and shape, of the rectangular region examined to find the median.

TABLE 8.1 Median filter weights.

1	1	1
1	1	1
1	1	1

TABLE 8.2 Decrease in number of watershed regions vs. median filter iterations.

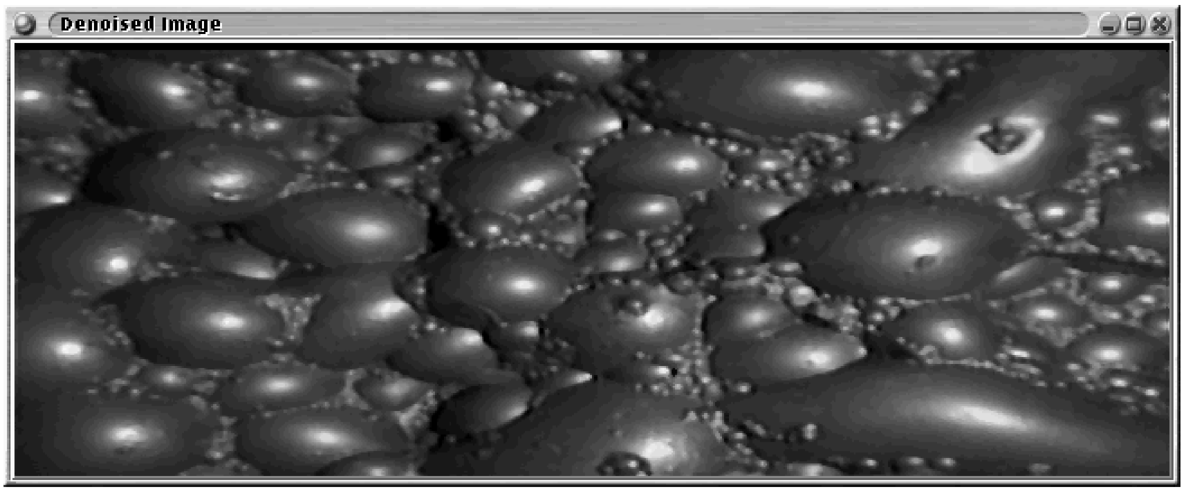
Iteration	#Watershed Regions
1	1004
2	734
3	598
4	517
5	465
10	384
20	377
50	376

One iteration of the median filter is not sufficient to adequately denoise the standard image. The large bubbles are spilt and many spurious regions can be seen. Two iterations performs better, more of the large bubbles are visible and more of the small bubbles properly segmented. As the number of iterations increases the smaller bubbles begin to merge with the larger bubbles, at four iterations (see figure 8.3) this is especially noticeable. Continued iteration of the median filter will result in the final image converging to a so-called root image [118]. The median filter as is does not preserve the gradual intensity curvature around the highlights of the bubbles, which leads to these being eroded and then eliminated. This explains the removal of the smaller bubbles at higher iterations.

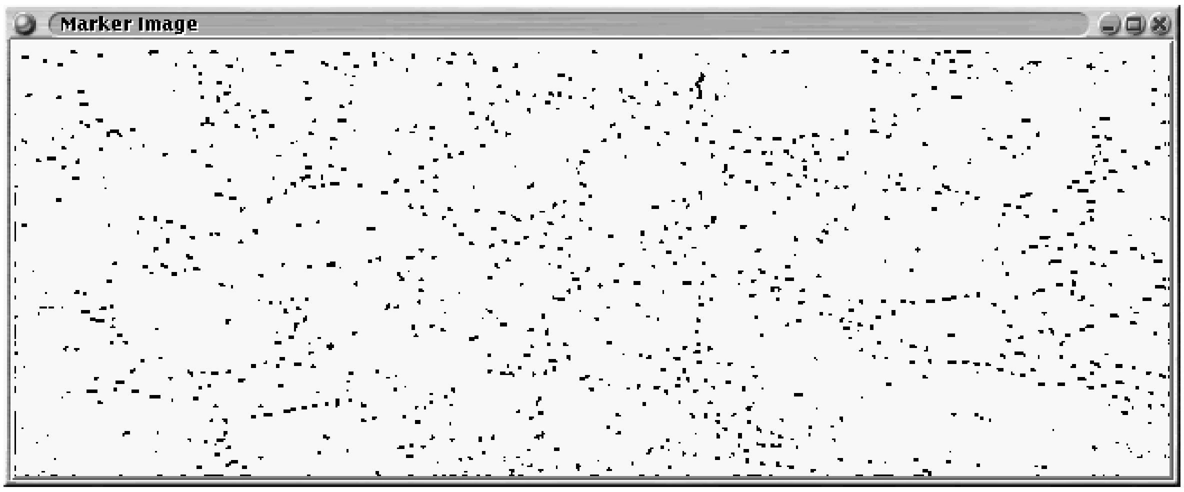
The median filter will then work well if the denoising requirement can be relaxed to extracting larger bubbles only.

8.2 Preserving corners

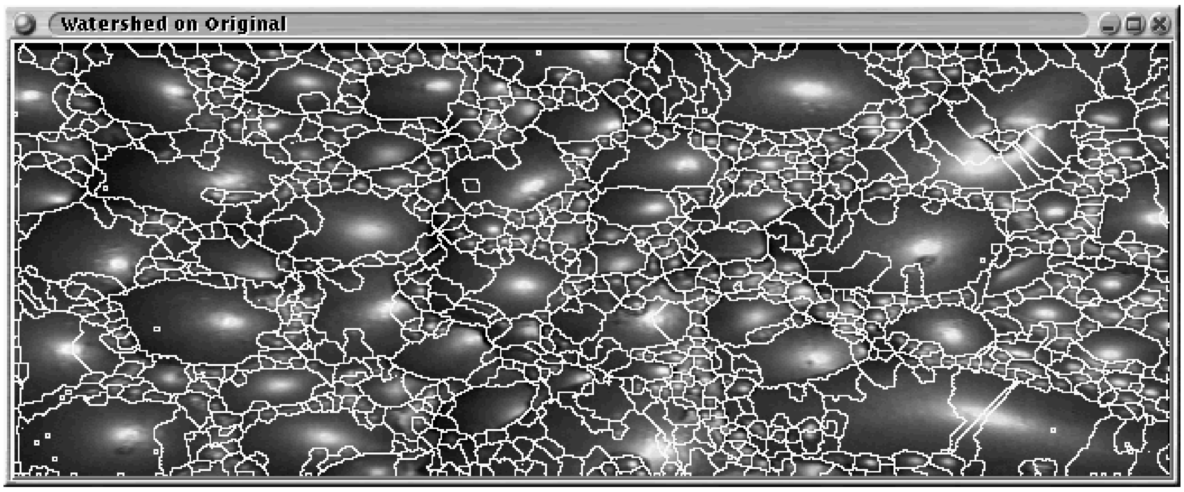
The corner preserving mask is described in table 8.3 [125]. Table 8.4 shows the decrease in the number of watershed regions (and hence minima of the image) with increasing iterations of the filter



(a) Denoised image.

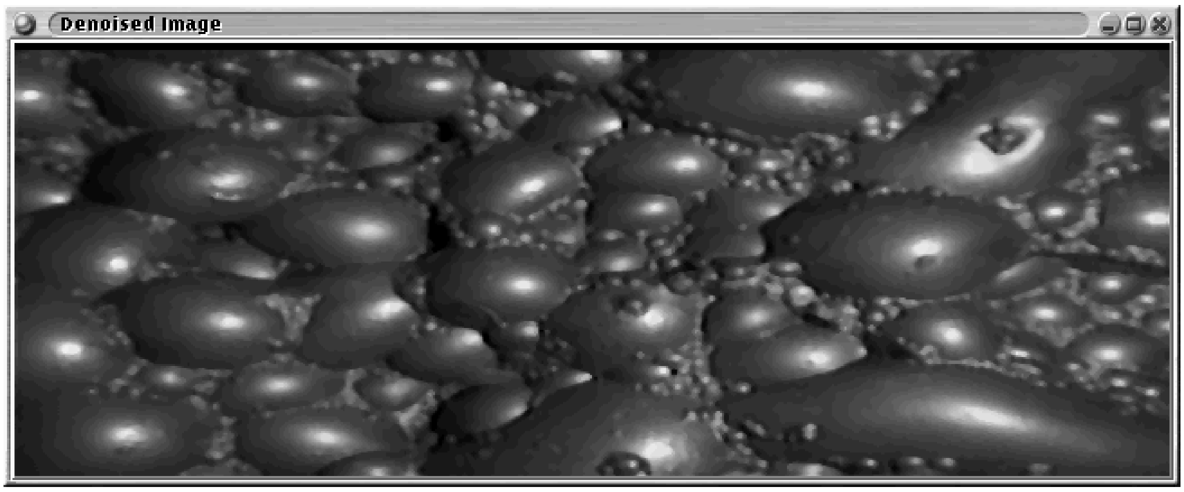


(b) Marker image.

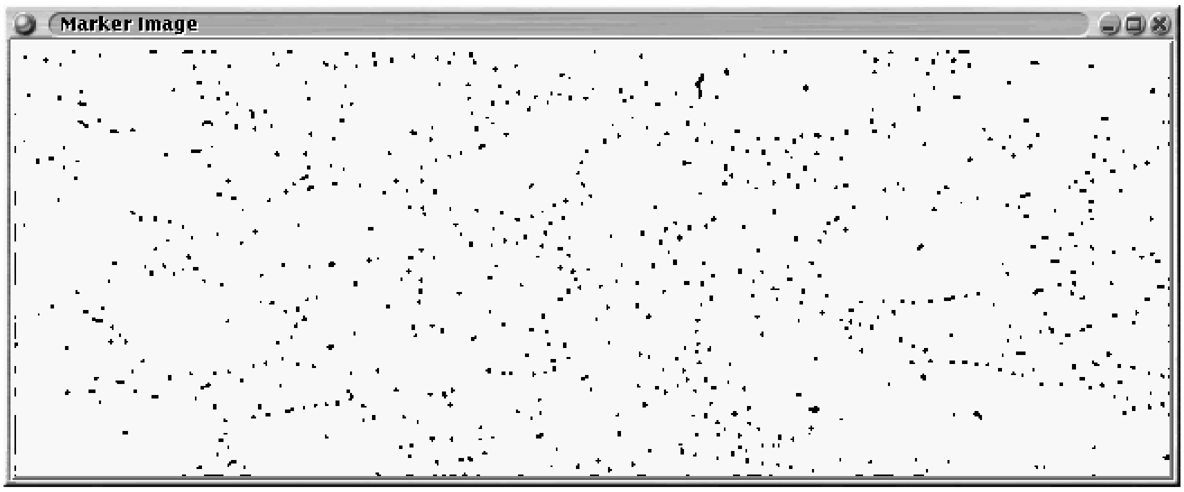


(c) Watershed segmented image.

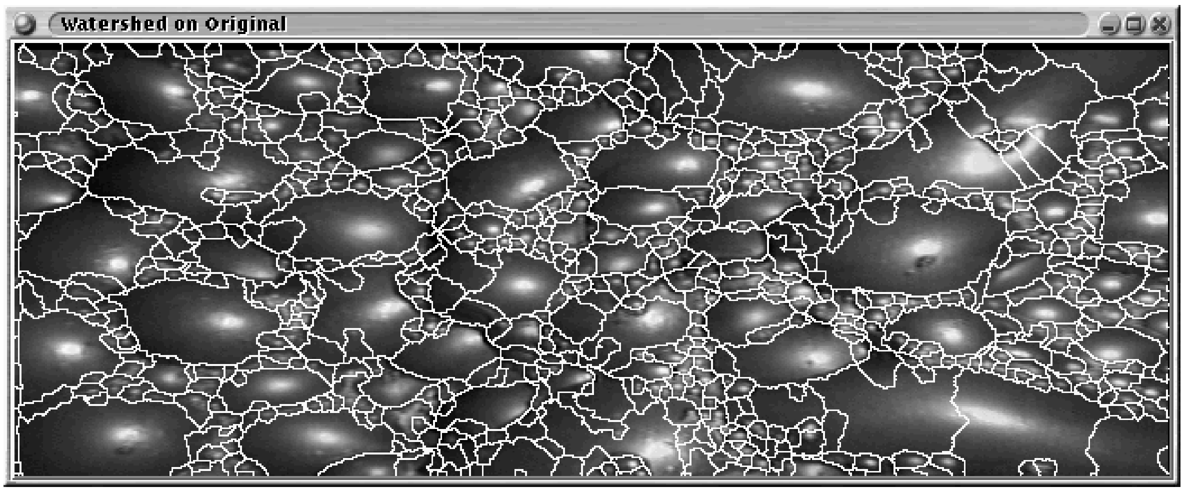
FIGURE 8.1 Median filter based denoising, 1 iteration.



(a) Denoised image.

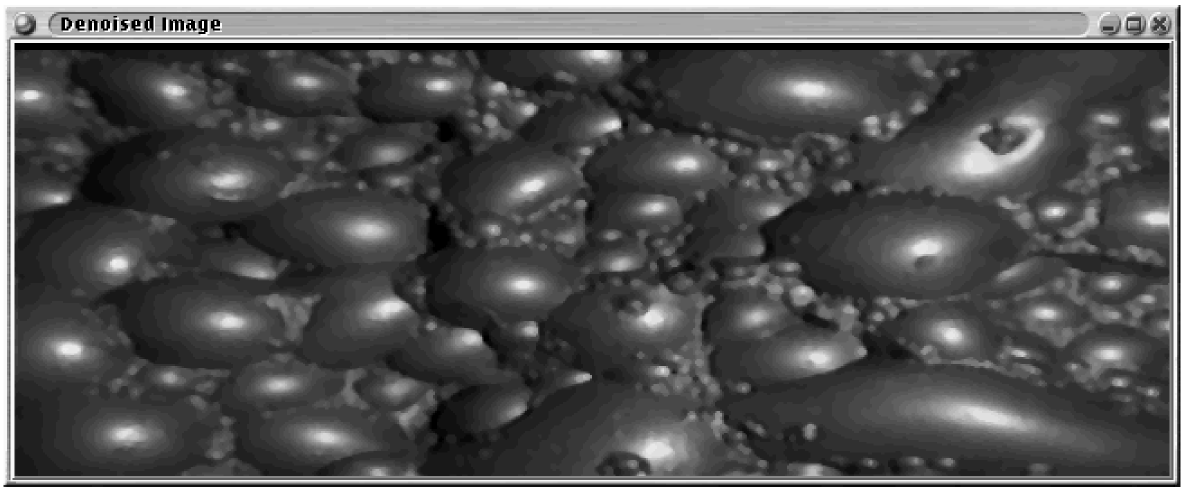


(b) Marker image.

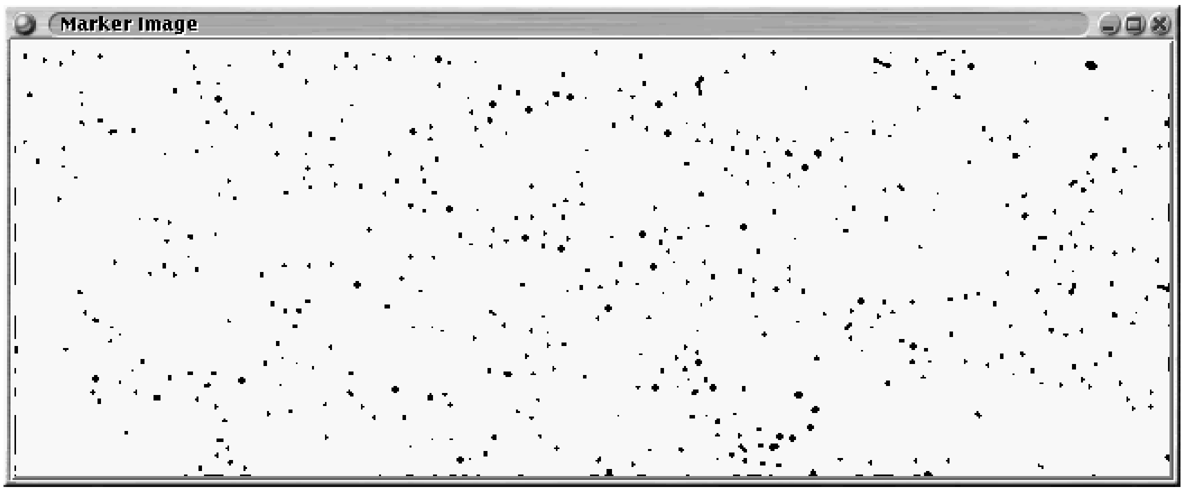


(c) Watershed segmented image.

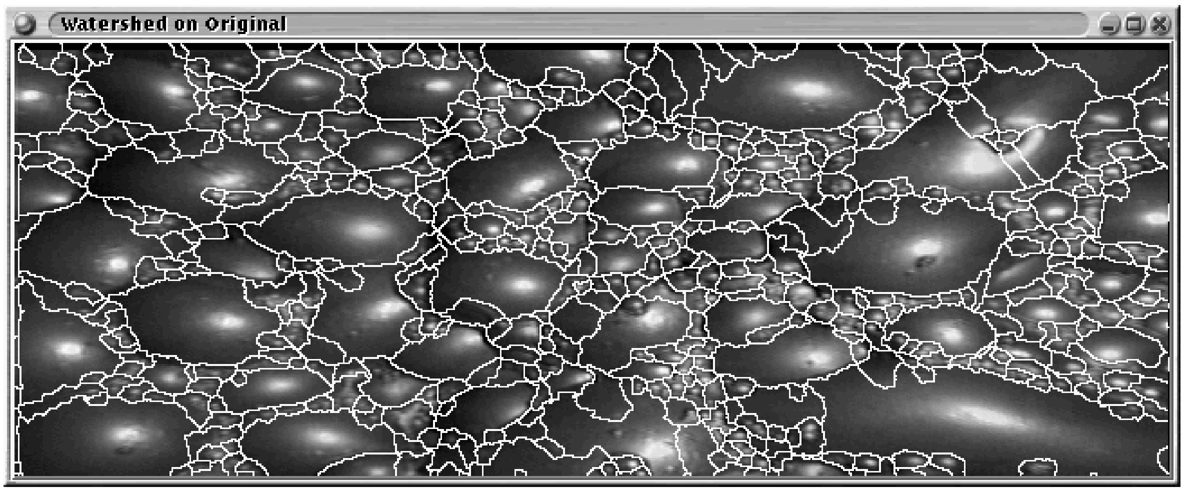
FIGURE 8.2 Median filter based denoising, 2 iterations.



(a) Denoised image.



(b) Marker image.



(c) Watershed segmented image.

FIGURE 8.3 Median filter based denoising, 4 iterations.

described.

Figures 8.4 and 8.5 show the effect on the segmentation of applying a median corner preserving filter iteratively, up to three times. Table 8.4 shows that after three iterations the number of segmented regions remains constant. This is likely due to the detail preserving property of the filter. This implies rapid convergence to the root image for this median filter. The high detail preserving property of this filter makes it unattractive for denoising froth images.

TABLE 8.3 Weighted median filter weights preserving corners.

2	3	2
3	5	3
2	3	2

TABLE 8.4 Decrease in number of watershed regions vs. corner preserving median filter iterations.

Iteration	# Watershed regions
1	1068
2	902
3	880
4	880
5	880
10	880
20	880
50	880

8.3 Preserving horizontal and vertical lines

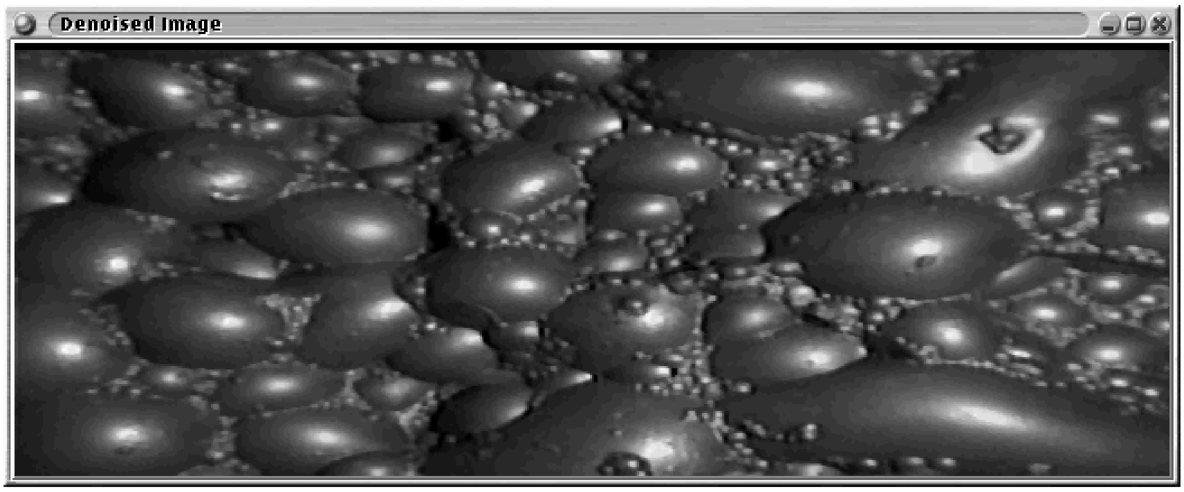
Table 8.5 describes the mask preserving horizontal and vertical lines [125]. Table 8.6 shows the decrease in the number of watershed regions vs. the number of iterations of this filter.

Figures 8.6 and 8.7 shows the effect on denoised, marker and watershed segmentation of iterating the median filter described, up to three times. Note that there is no significant difference between the number of watershed regions at the third and fiftieth iterations. Again, these slow changes per iteration are due to the detail preserving properties of the filter, specifically the high weight given to the centre pixel in the mask. The high detail preserving properties of this filter renders it unattractive for denoising froth images.

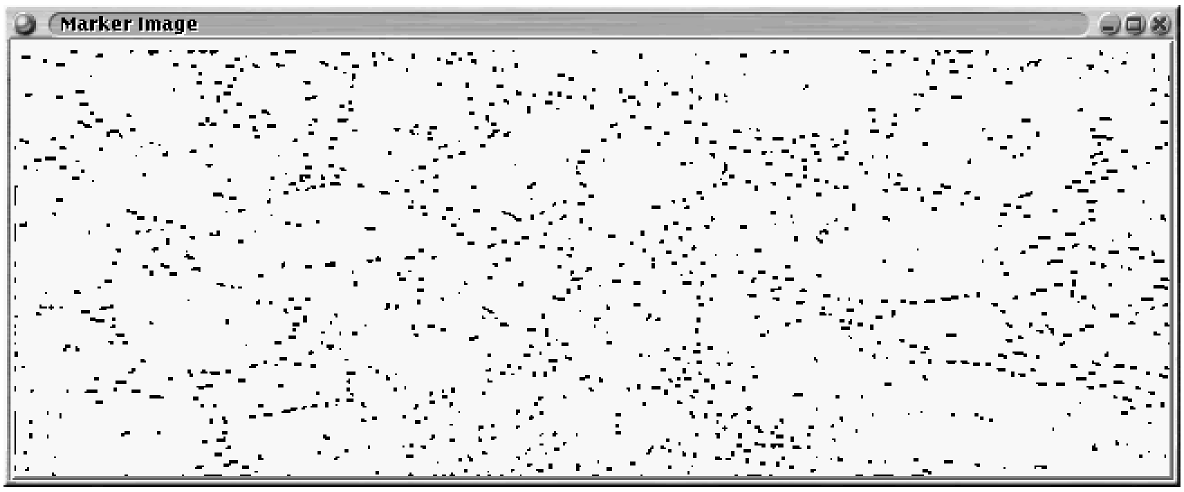
8.4 Preserving horizontal, vertical and diagonal lines

Table 8.7 describes the mask preserving horizontal, vertical and diagonal lines [125]. Table 8.8 describes the decrease in watershed regions as the number of iterations of this filter increases. The decrease is not rapid indicating that the detail preserving properties of this filter preserves noise induced minima.

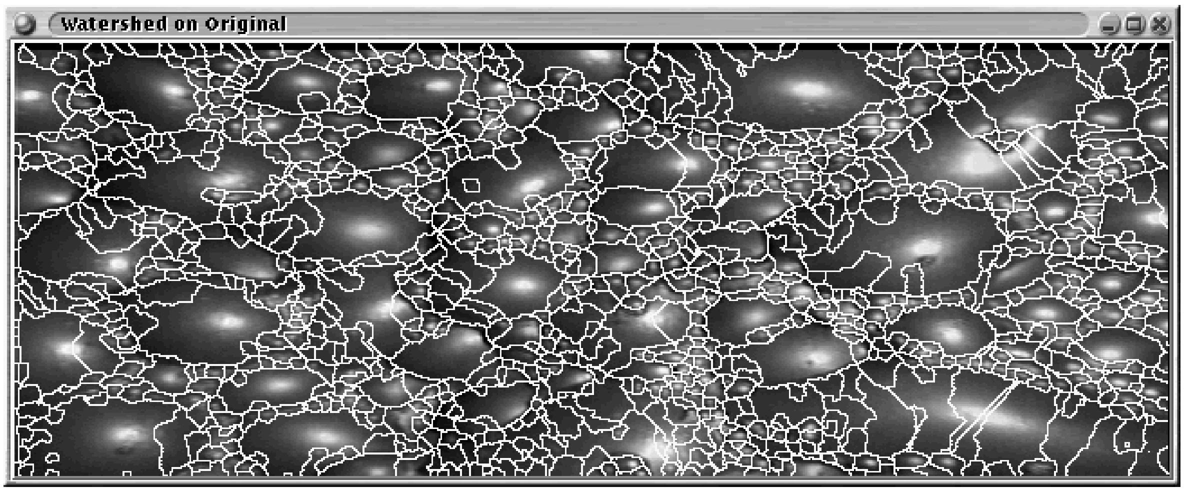
Figures 8.8 and 8.9 and describe the effect of iterating the median filter described up to five times. Examination of the figures do not show significant differences between the first and the fifth iterations.



(a) Denoised image.

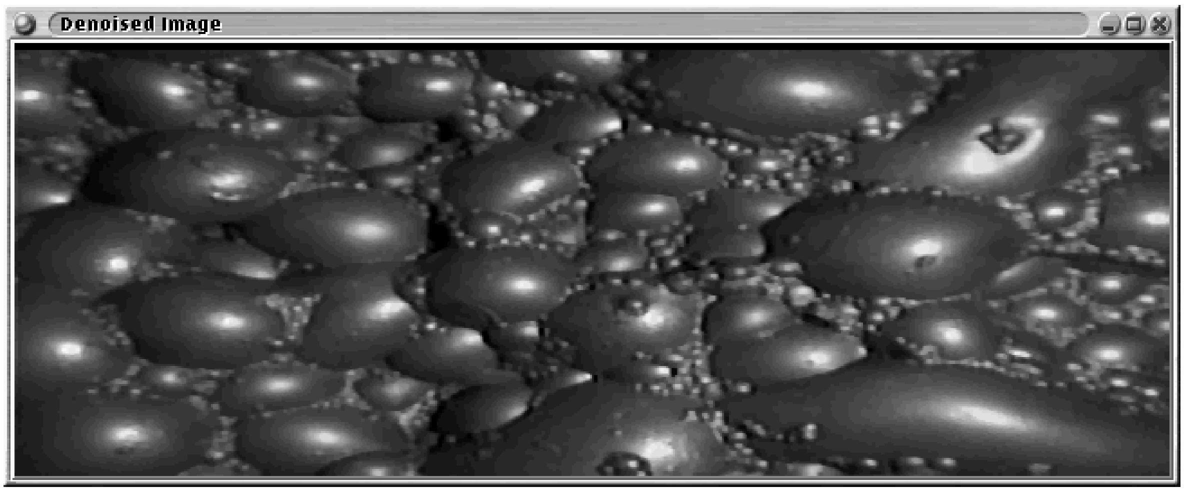


(b) Marker image.

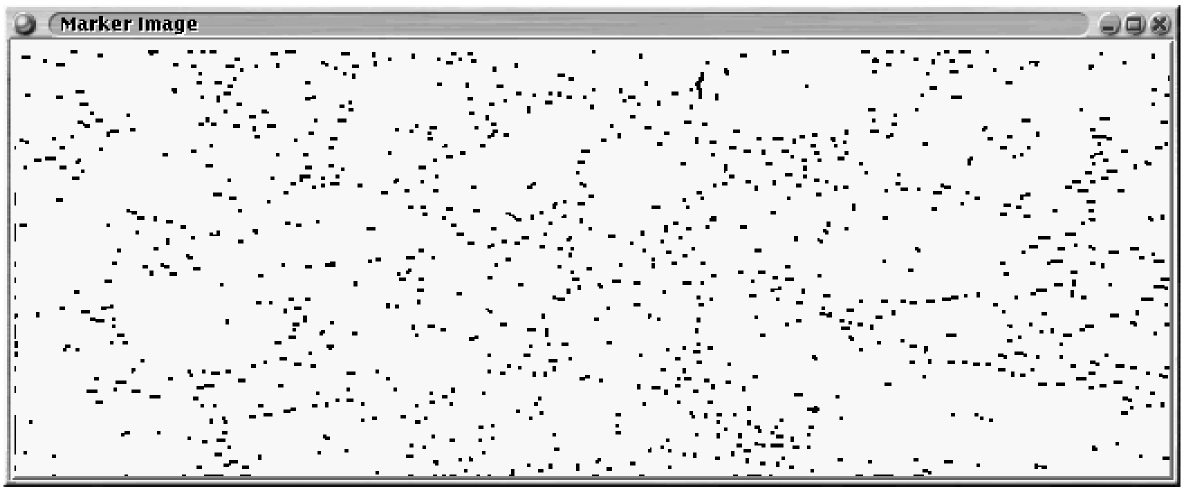


(c) Watershed segmented image.

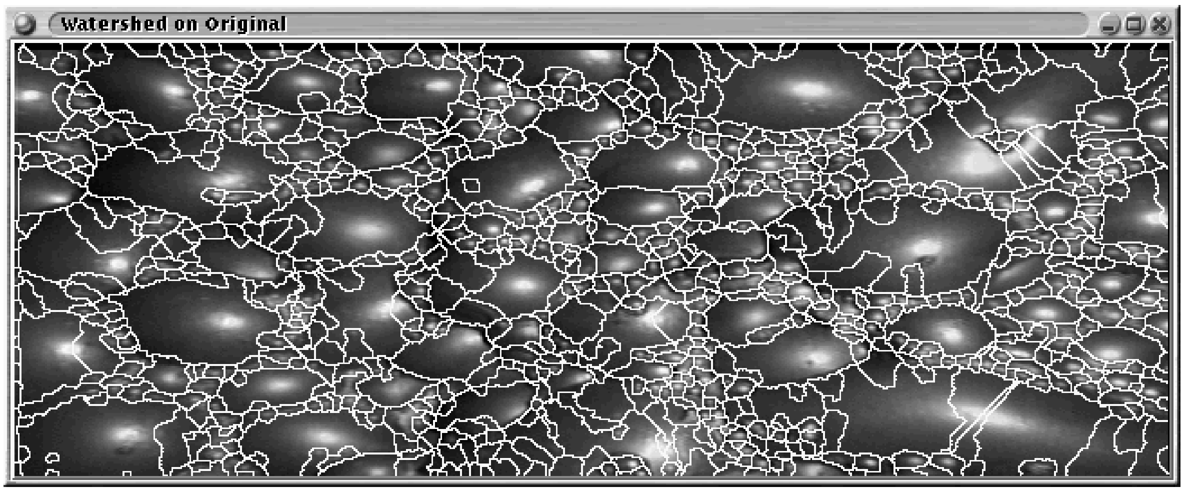
FIGURE 8.4 Corner preserving median filter based denoising, 1 iteration.



(a) Denoised image.



(b) Marker image.



(c) Watershed segmented image.

FIGURE 8.5 Corner preserving median filter based denoising, 3 iterations.

TABLE 8.5 Weighted median filter weights preserving horizontal and vertical lines.

1	3	1
3	5	3
1	3	1

TABLE 8.6 Decrease in number of watershed regions vs. horizontal and vertical line preserving median filter iterations.

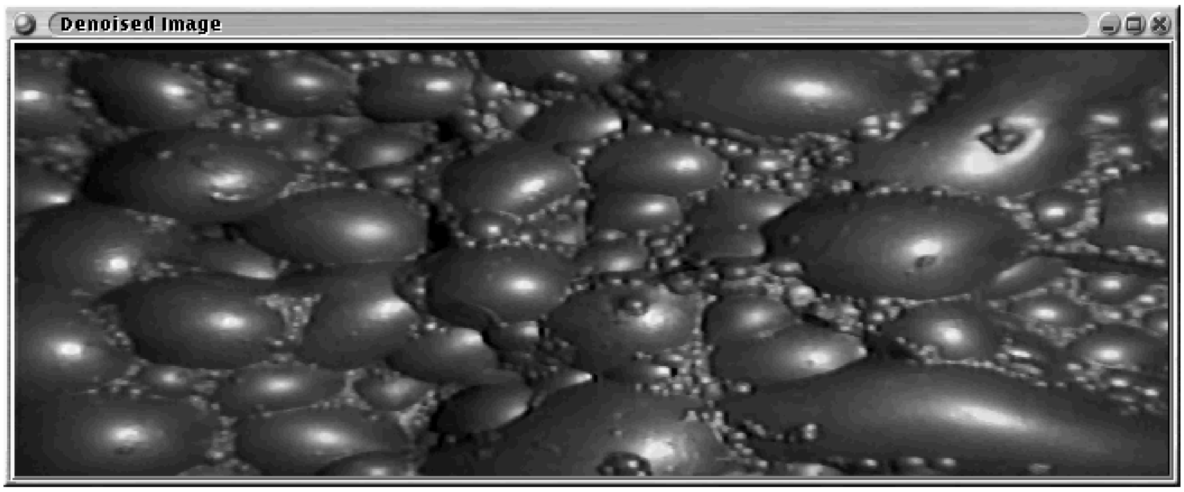
Iteration	# Watershed regions
1	1317
2	953
3	878
4	872
5	870
10	869
20	869
50	869

TABLE 8.7 Weighted median filter weights preserving horizontal, vertical and diagonal lines.

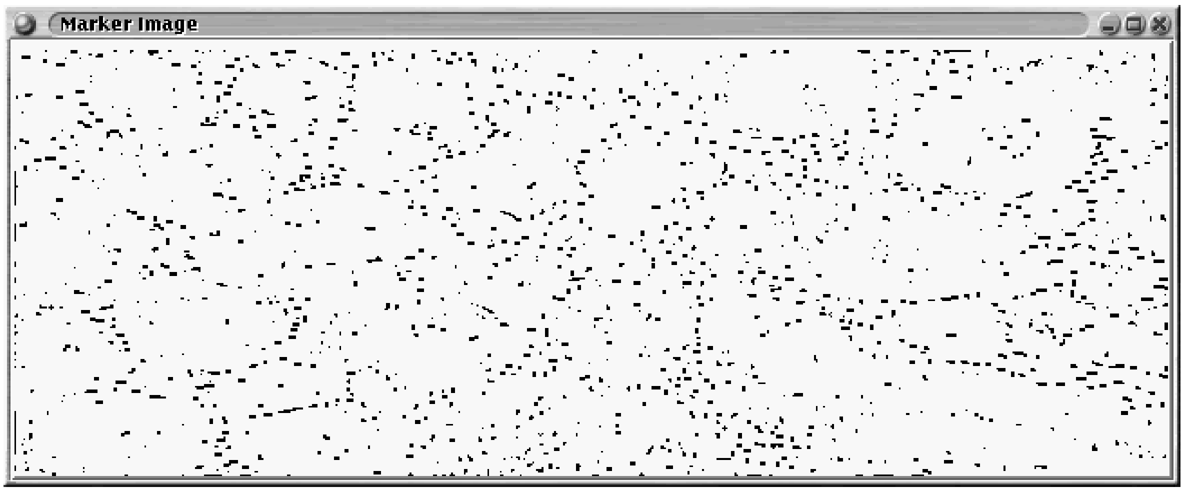
1	1	1
1	5	1
1	1	1

TABLE 8.8 Decrease in number of watershed regions vs. horizontal, vertical and diagonal line preserving median filter iterations.

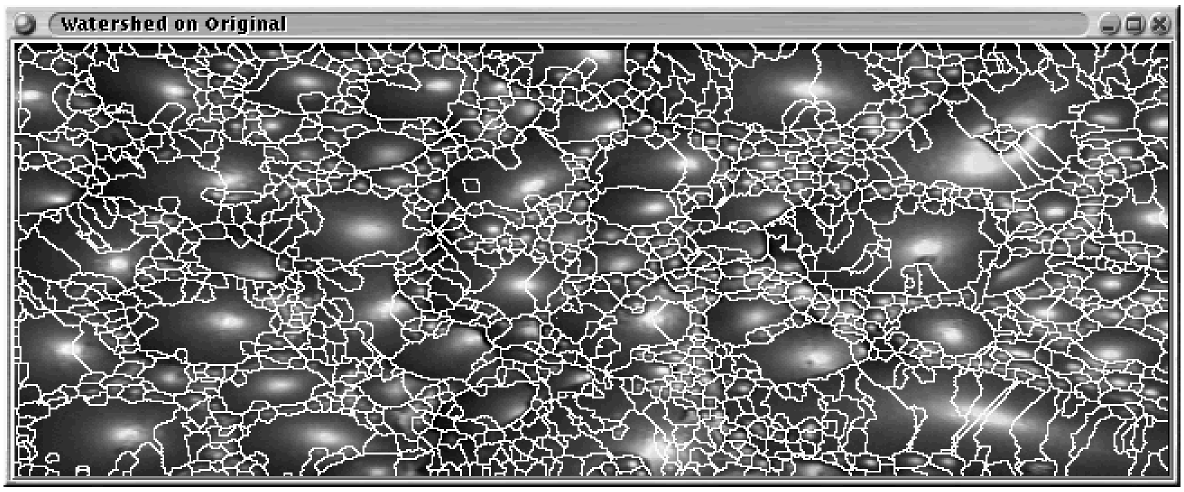
Iteration	# Watershed regions
1	1262
2	1071
3	1053
4	1051
5	1050
10	1050
50	1050



(a) Denoised image.

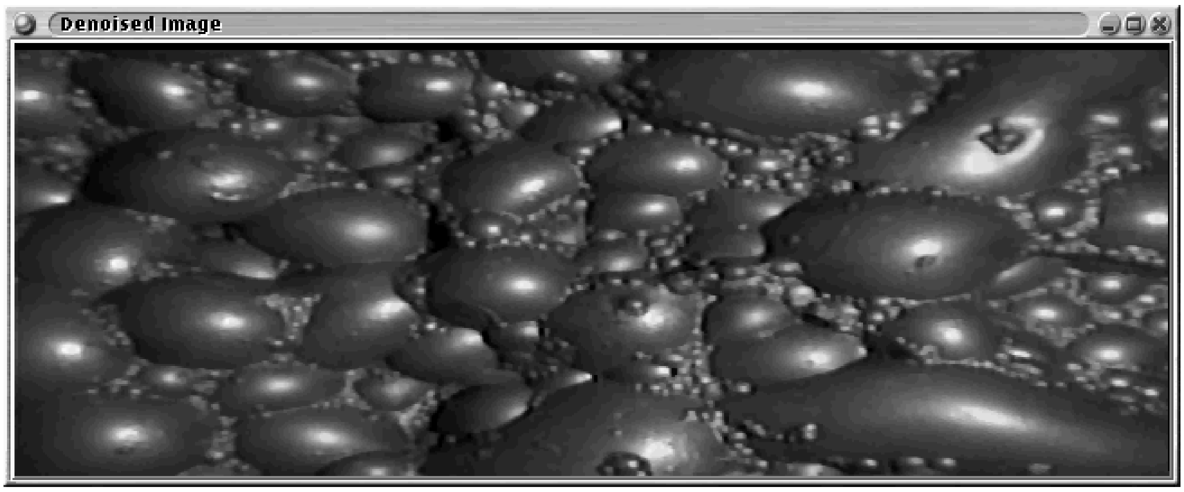


(b) Marker image.

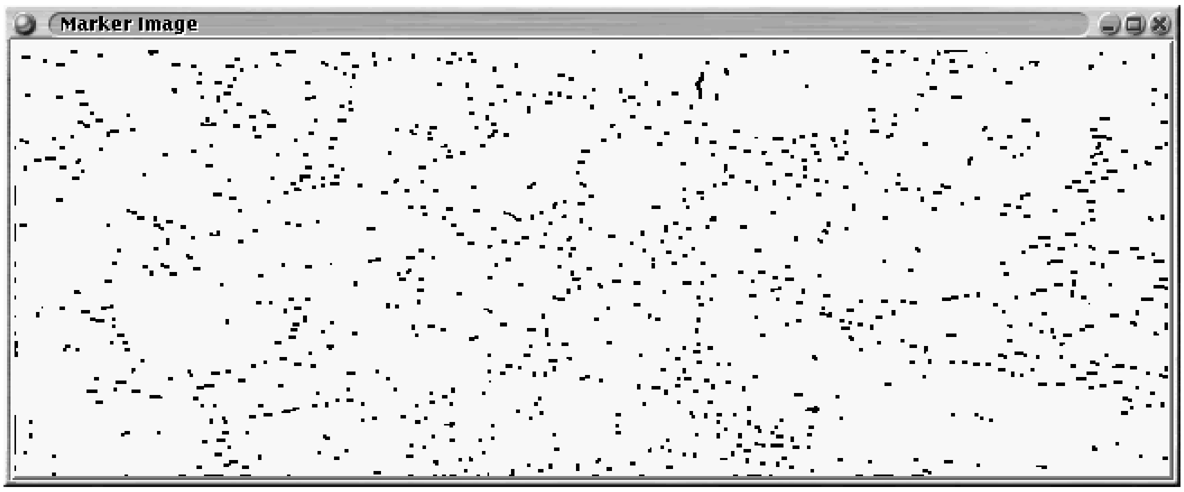


(c) Watershed segmented image.

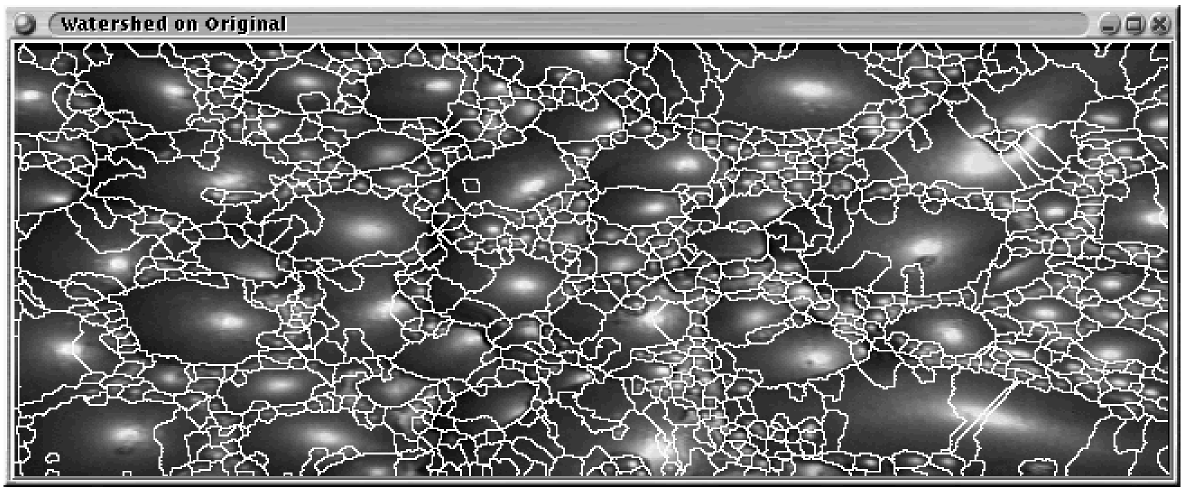
FIGURE 8.6 Horizontal and vertical line preserving median filter based denoising, 1 iteration.



(a) Denoised image.

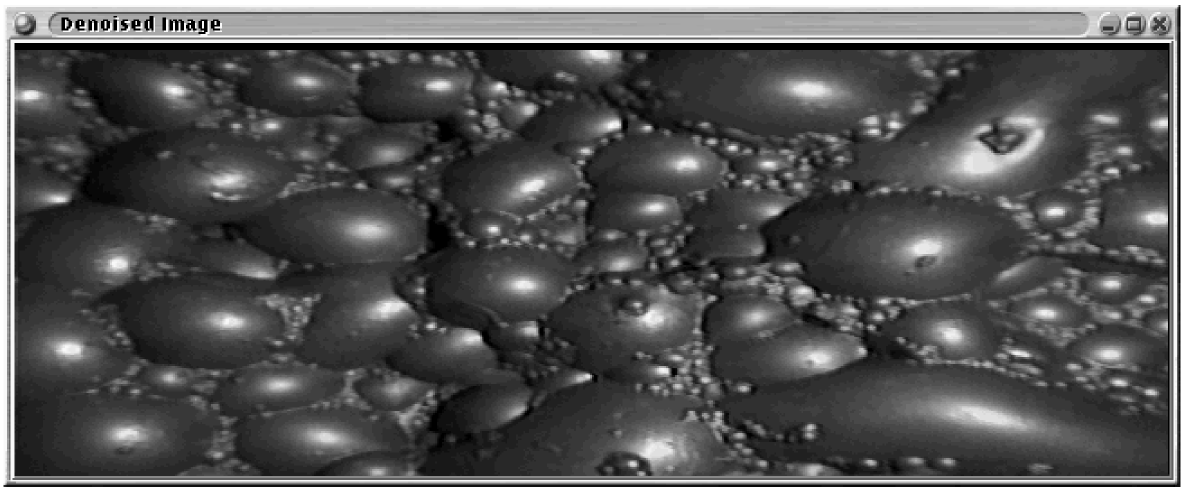


(b) Marker image.

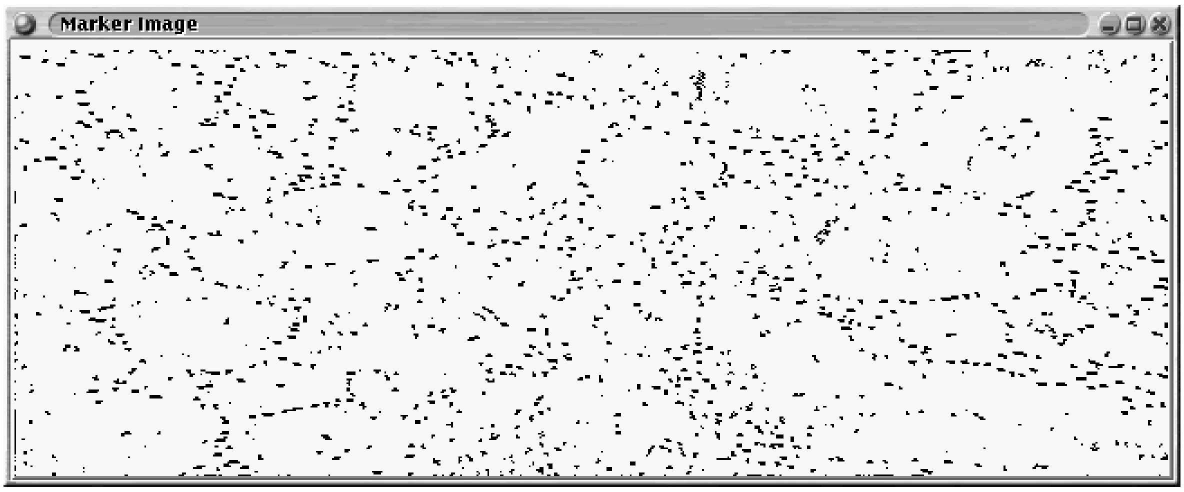


(c) Watershed segmented image.

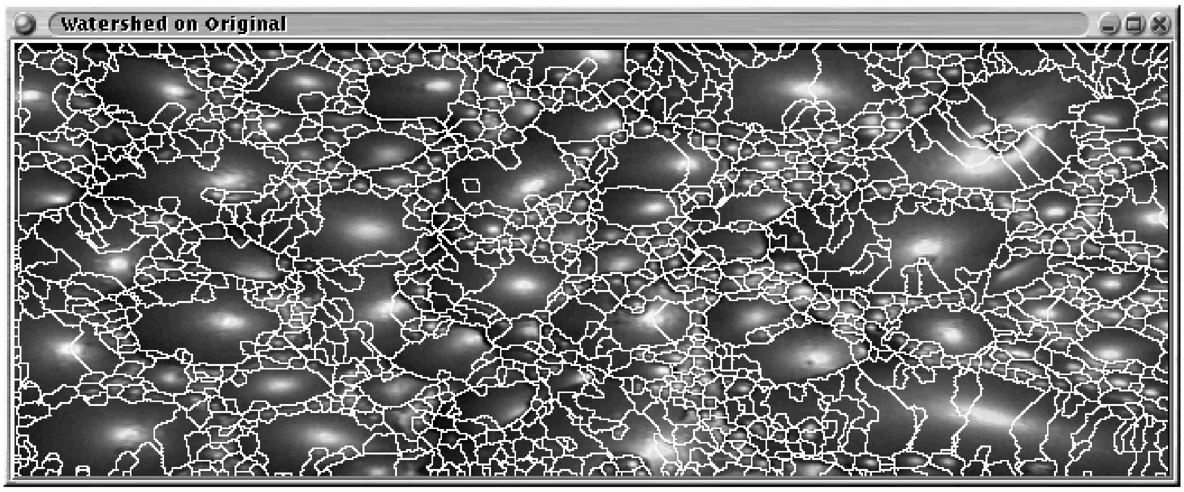
FIGURE 8.7 Horizontal and vertical line preserving median filter based denoising, 3 iterations.



(a) Denoised image.

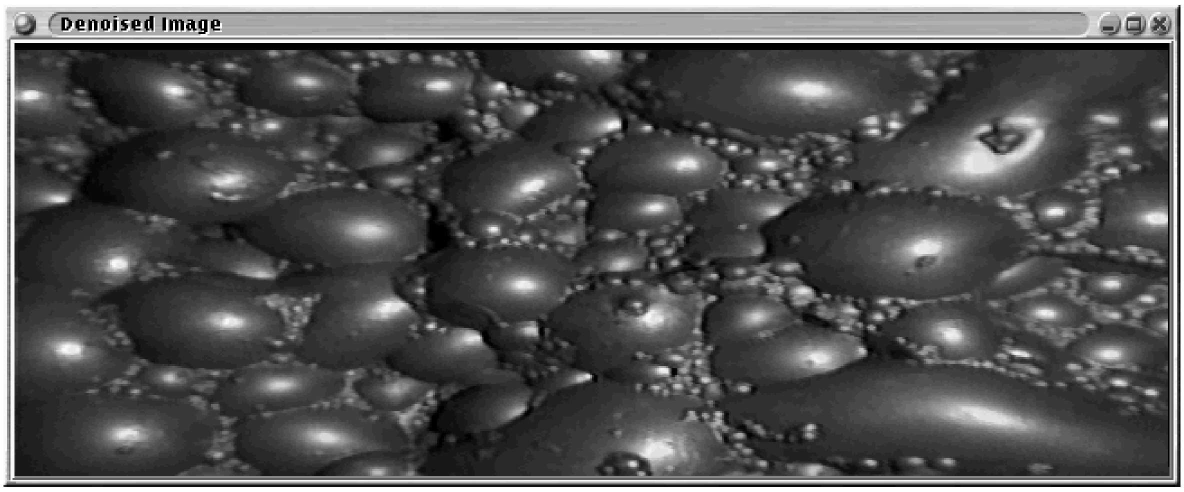


(b) Marker image.

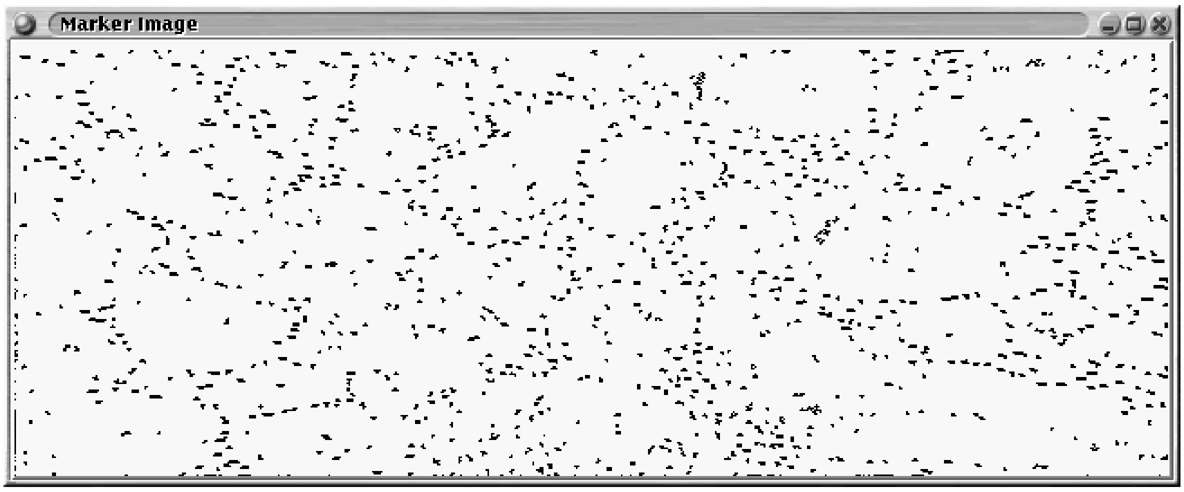


(c) Watershed segmented image.

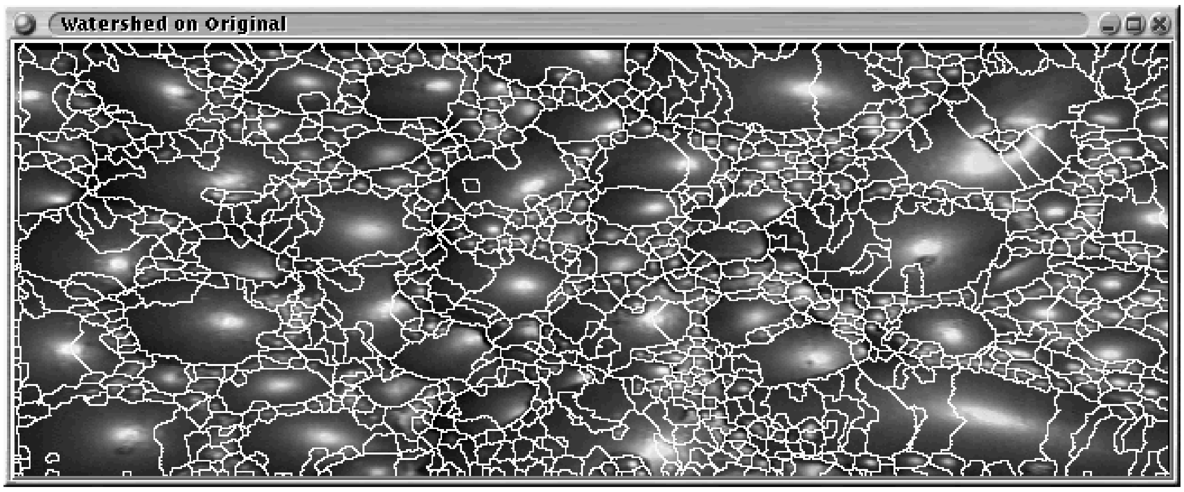
FIGURE 8.8 Horizontal, vertical and diagonal line preserving median filter based denoising, 1 iteration.



(a) Denoised image.



(b) Marker image.



(c) Watershed segmented image.

FIGURE 8.9 Horizontal, vertical and diagonal line preserving median filter based denoising, 5 iterations.

8.5 Preserving horizontal lines, vertical lines and corners

Table 8.9 describes the mask preserving corners, horizontal and vertical lines [125]. Table 8.10 shows the decrease in the number of watershed regions as the filter is iterated. Note that after four iterations, further iterating the filter does not significantly reduce the number of watershed regions.

Figures 8.10 and 8.11 represent the effect of iterating the median filter described up to five times on a standard test image. There is little difference between the segmentations shown. This is a result of the detail preserving properties of the filter in question. This is clear from examining table 8.10.

TABLE 8.9 Weighted median filter weights preserving horizontal and vertical lines, as well as corners.

1	2	1
2	5	2
1	2	1

TABLE 8.10 Number of watershed regions per iteration for median line and corner preserving filter.

Iteration	# Watershed regions
1	1329
2	962
3	890
4	880
5	879
10	876
50	876

8.6 Tri-State Median Filters

This algorithm was proposed by Chen et al [23]. The filter was originally designed for impulse noise removal. The filter chooses one of the outputs of three different filters based on a threshold. The filters are:

(i) Identity filter – returns value of the centre pixel (X_c) of neighbourhood.

(ii) Simple median (SM) filter.

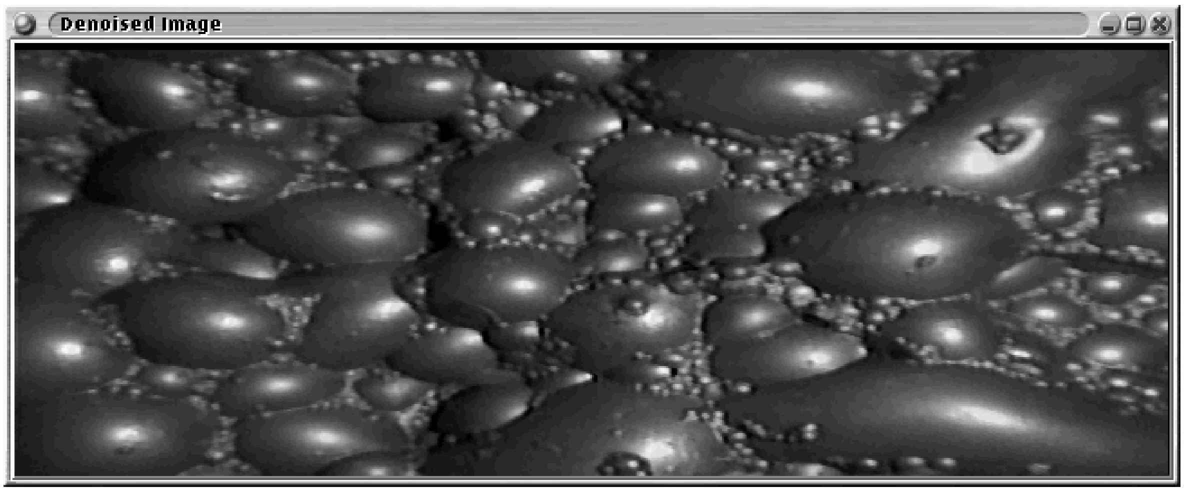
(iii) Centre weighted median (CWM) filter.

$$Y^{TSM} = \begin{cases} X_c & T \le d_1, \\ Y^{CWM} & d_2 \le T \le d_1, \\ Y^{SM} & T < d_2 \end{cases} \tag{8.1}$$

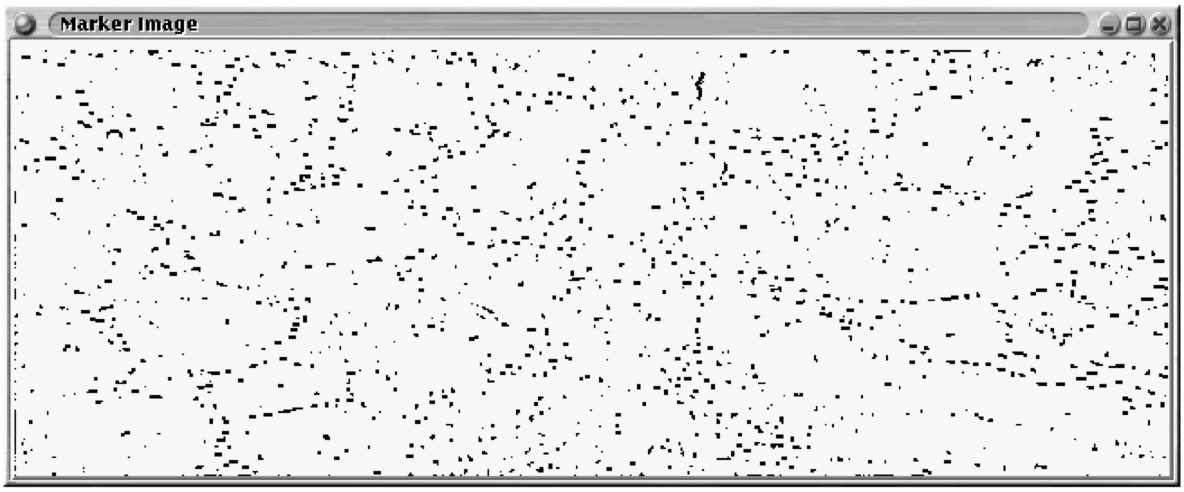
$$d_1 = \mid X_c - Y^{SM} \mid$$

$$d_2 = \mid X_c - Y^{CWM} \mid$$

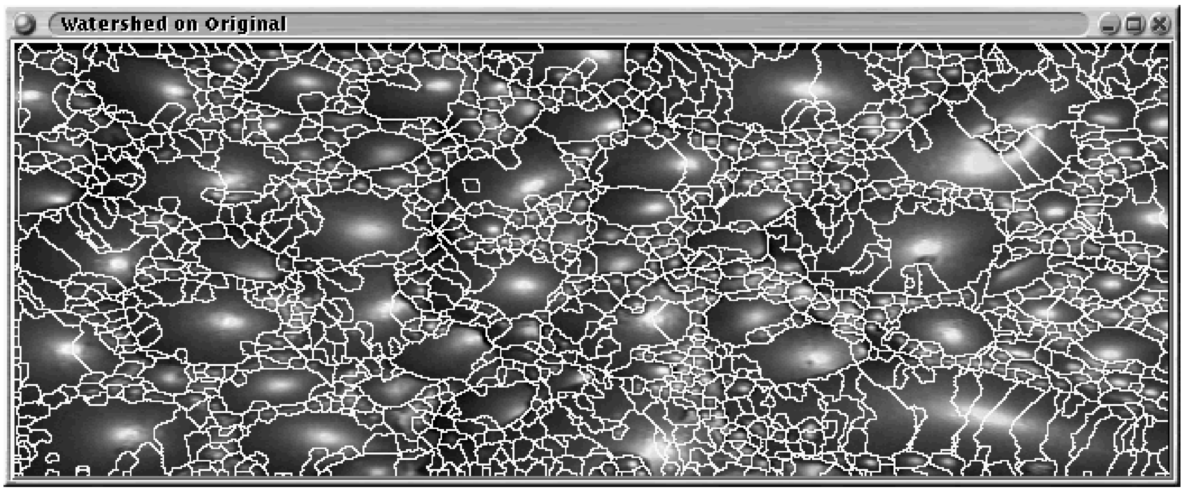
T Threshold setting denoising allowed



(a) Denoised image.

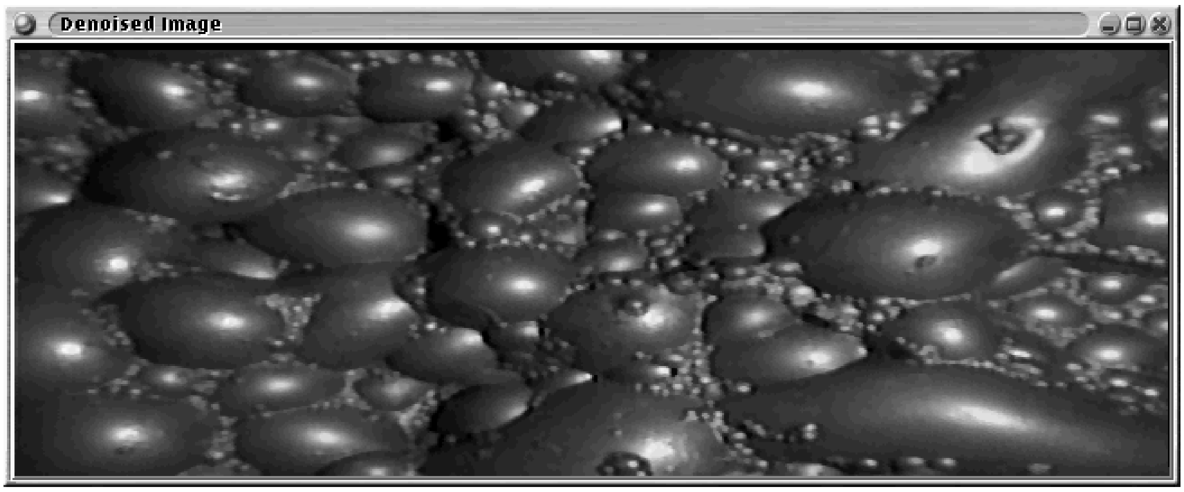


(b) Marker image.

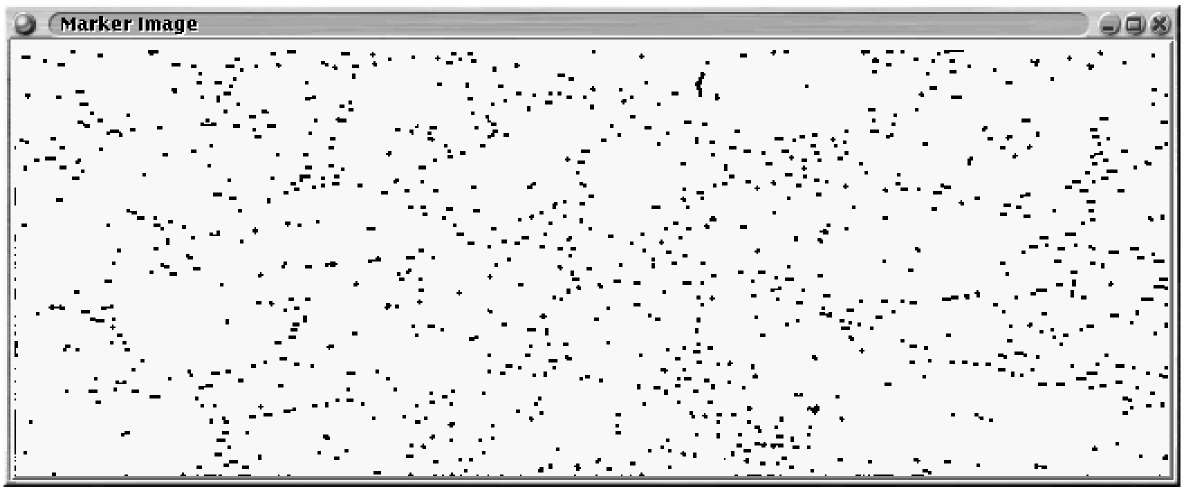


(c) Watershed segmented image.

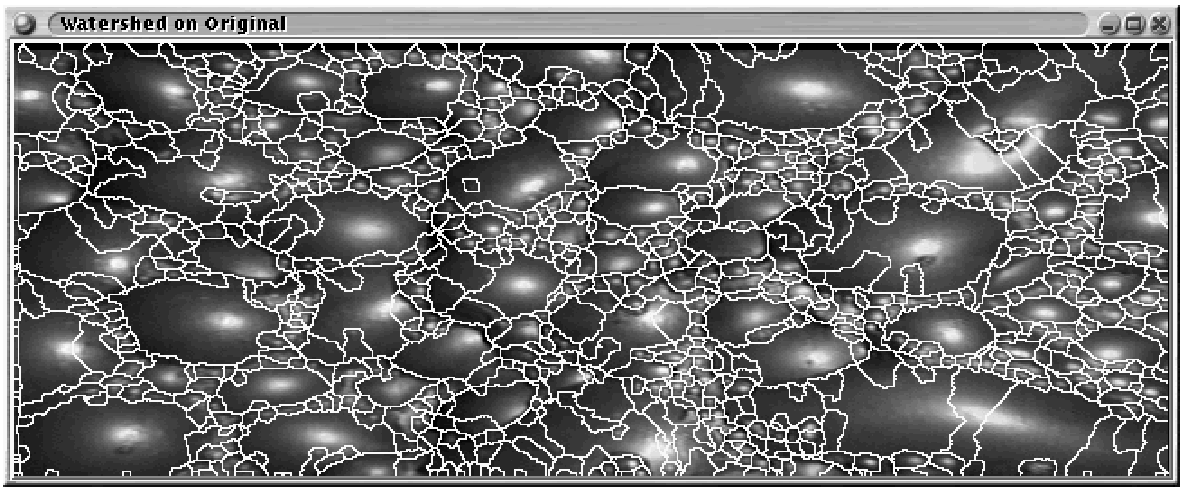
FIGURE 8.10 Corner and line preserving median filter based denoising, 1 iteration.



(a) Denoised image.



(b) Marker image.



(c) Watershed segmented image.

FIGURE 8.11 Corner and line preserving median filter based denoising, 5 iterations.

Table 8.11 shows the decrease in the number of watershed regions vs. the increasing number of iterations. Figure 8.12 shows the application of one iteration of the tri-state median filter to the standard froth image. Figure 8.13 shows the result of fifty iterations. The threshold used was 20 and the weight for the centre weighted median filter, 3. These values are selected to best denoise the test image. Comparison of the denoised, marker and segmented images in figures 8.12 and 8.13 do not show a great deal of difference. The filter does not denoise well at all, indicating then that the noise present is clearly not impulse noise.

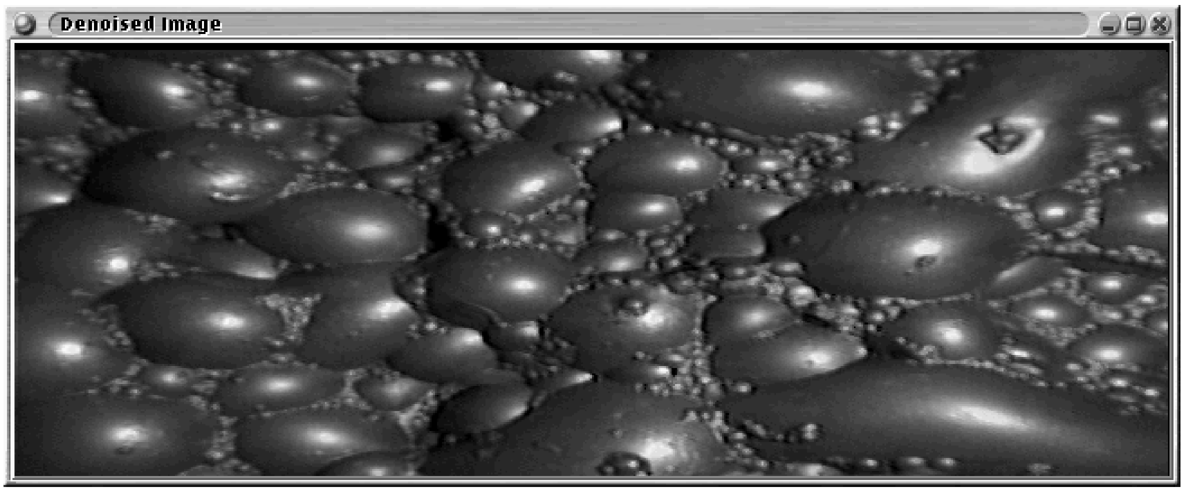
TABLE 8.11 Number of watershed regions per iteration for tri-state median filter.

Iteration	#Watershed regions
1	3352
2	3343
3	3341
4	3340
5	3339
10	3338
50	3338

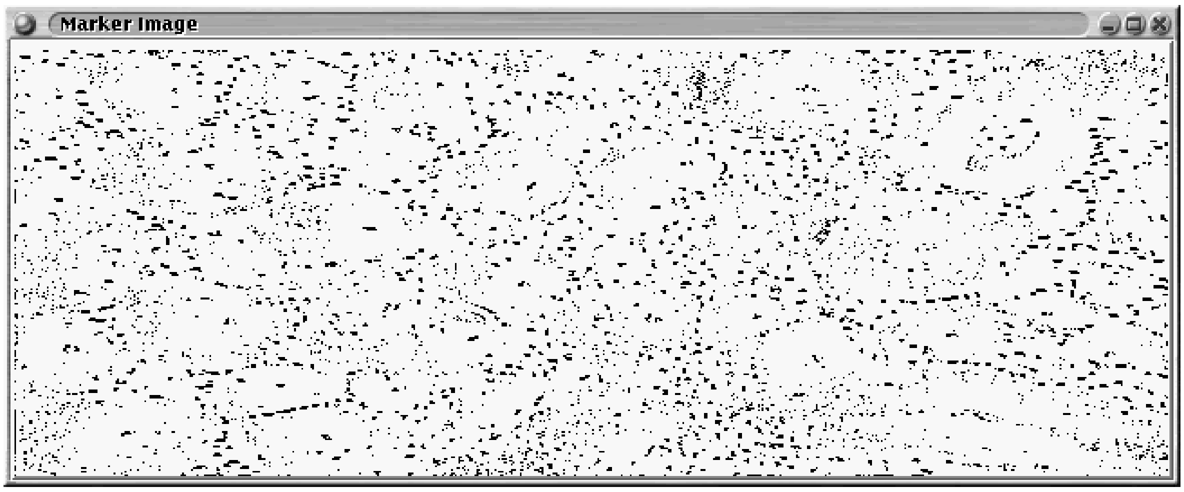
8.7 Discussion

The weighted median filter mask required is one that preserves of the gradual intensity changes of bubbles. The better the detail preserving characteristics of the median filter used, the worse that filter is from a denoising perspective. A tradeoff does exist between improving the segmentation and the computational effort involved. The detail preserving nature of the weighted median filters result in them converging rapidly to the root image. The stronger the detail preserving property the more rapid the convergence. Consider table 8.12 which ranks the various median filters in order of the number of the surviving watershed regions at iteration fifty. This gives an indication of how rapidly convergence to the root image occurs. The lowest ranking filter converges the fastest i.e. the Tri-State median filter. The detail preserving properties can clearly be seen to speed convergence to the root image. Qualitative examination of the segmentation of a standard froth image, on the other hand, suggests that detail preservation is of little use in denoising the images to improve the segmentation.

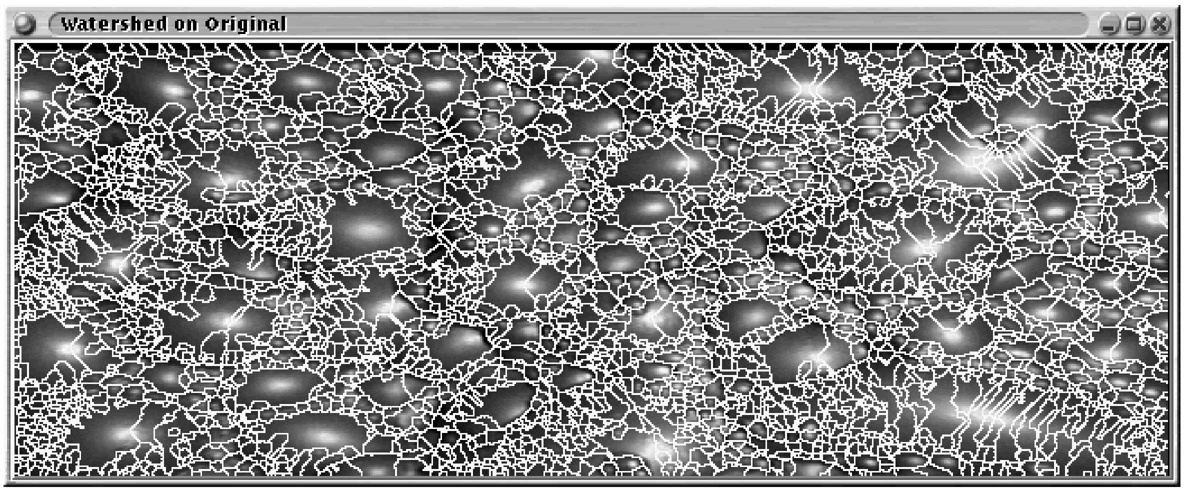
The weighted median filters would require too many iterations to significantly improve the segmentation. This is especially true of the tri-state median filter.



(a) Denoised image.

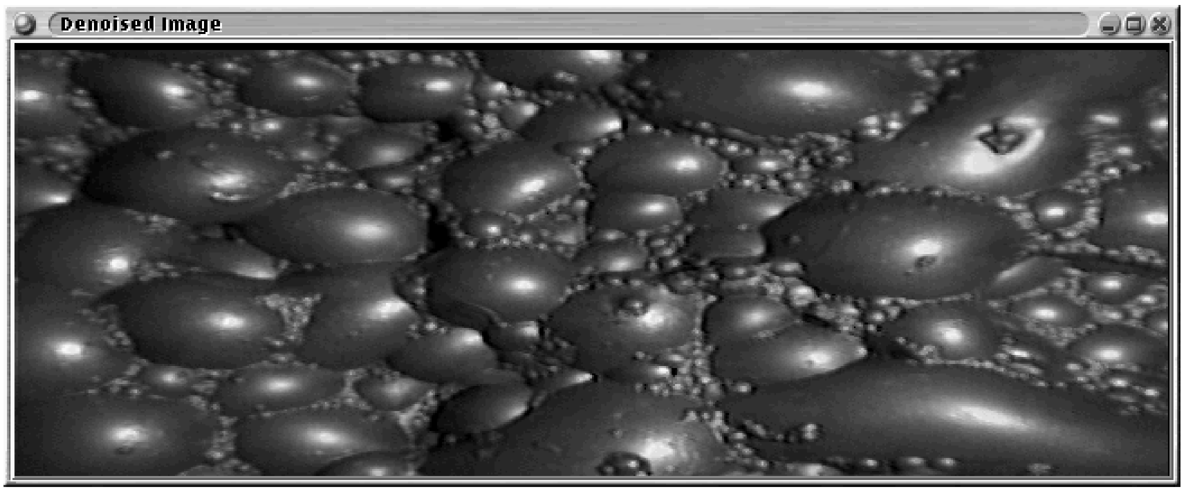


(b) Marker image.

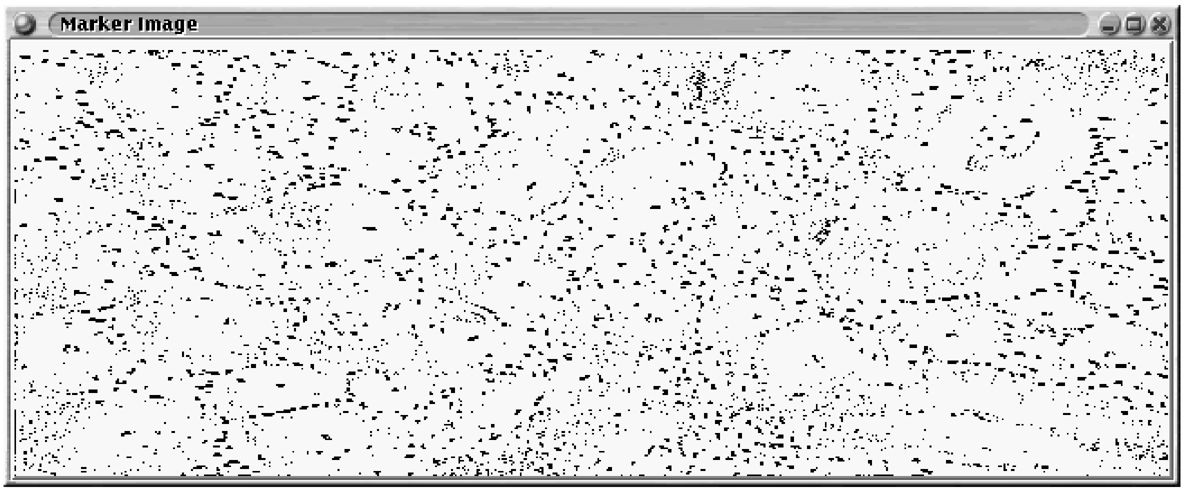


(c) Watershed segmented image.

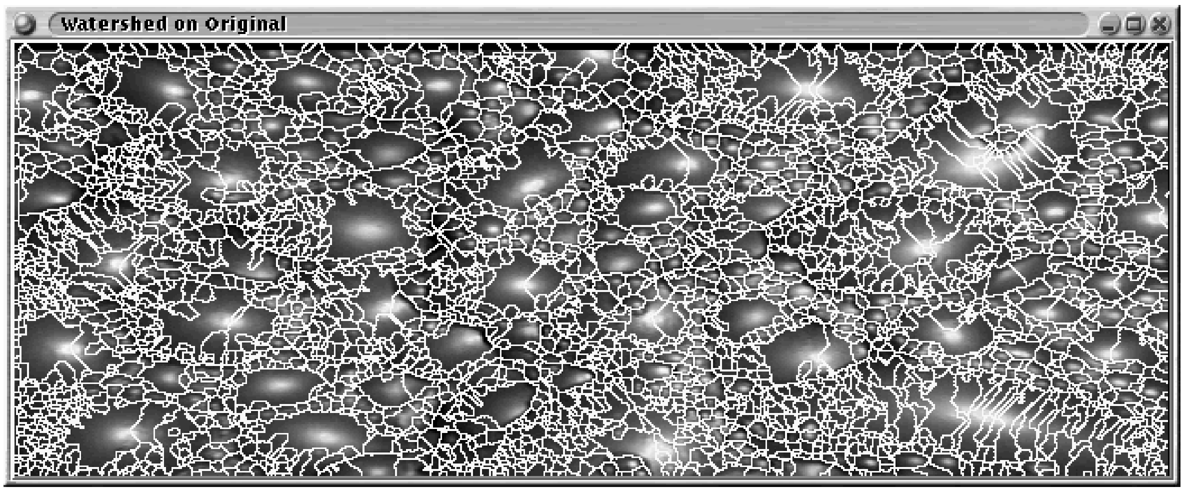
FIGURE 8.12 Tri-State median filter based denoising, 1 iteration.



(a) Denoised image.



(b) Marker image.



(c) Watershed segmented image.

FIGURE 8.13 Tri-State median filter based denoising, 50 iterations.

TABLE 8.12 Ranking of median filters. Number of watershed regions after fifty iterations.

Rank	Median filter type	# Watershed regions
1	Simple Median	367
2	Preserving corners	880
3	Preserving horizontal and vertical lines	869
4	Preserving horizontal lines, vertical lines and corners	876
5	Preserving horizontal, vertical and diagonal lines	1050
6	Tri-State	3338

Chapter 9

Morphology based denoising

Several techniques in mathematical morphology can be used for denoising. These include rolling ball filters, greyscale reconstruction via erosion and dilation, and simple flat structuring element approaches (see [34, 35, 64, 127] for an introduction to morphology). The goal of all these attempts is to fill, or eliminate, small maxima and minima likely to be associated with noise.

9.1 Flat structuring elements

This is similar to the rolling ball algorithms but rather the structuring element has no intensity. The resultant filter is an order statistic filter extracting either the first (min) or last (max) order statistic [125] of a given region of the image. The erosion and dilation operates on certain pixels in a neighbourhood as specified by the region of support of the structuring element. The structuring element used here is circular in extent. This is an attempt to make the process more isotropic in spatial extent. There is no compelling reason to consider other shapes of structuring element, though others are certainly possible.

The following flat morphological filters are considered:

(i) Dilation [34, 64].

(ii) Erosion [34, 64].

(iii) Closing. A dilation followed by an erosion [34, 64].

(iv) Opening. An erosion followed by a dilation [34, 64].

(v) Close-Openings. A closing followed by an opening [34, 64].

(vi) Open-Closings. An opening followed by a closing [34, 64].

Table 9.1 shows the effect of various flat morphological filter radii on the number of watershed regions. Figures 9.1 through 9.16 shows the effect of these filters on the image, markers and watershed.

Notice especially, in the marker image of figure 9.3 that the markers are large diamond shaped. This is because for a regularly sampled grid a disc of radius one has a diamond shape. Notice also, that as the radius increases (figures 9.2 and 9.3) the markers become better approximations to a disc.

TABLE 9.1 Number of watershed regions against structuring element radius for flat dilation, erosion.

Radius	# Watershed Regions					
	Dilation	Erosion	Closing	Opening	Clos-opening	Open-closing
1	963	1413	1831	820	638	712
2	571	898	1087	413	308	295
3	377	828	1063	224	157	151
4	279	480	787	150	113	95
5	205	350	663	100	74	78
6	165	317	547	79	65	57

9.1.1 Discussion

It is apparent that the effect of iterating a flat morphological filter is identical to filtering with a single large structuring element. Only the effect of iteration on dilations and erosions will be considered. By definition openings and closings are idempotent [34, 64, 124] i.e. iterating an opening or closing have the same effect as a single opening or closing. Openings and closings should then not be considered for iteration. The shape and size of this structuring element can be seen as follows:

$$A(x, y) = \bigcup_{c \epsilon B} B_c(x, y) \tag{9.1}$$

where,

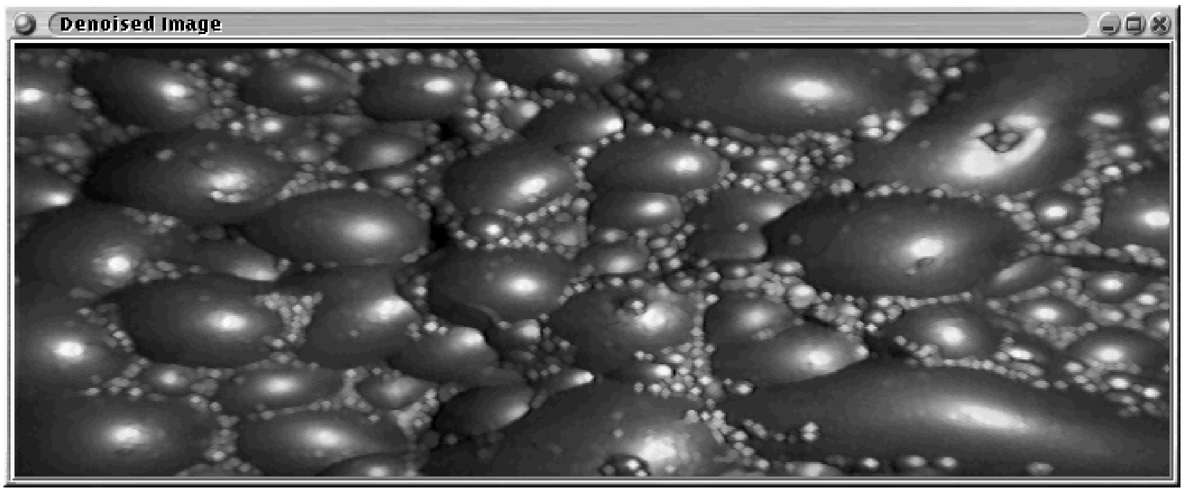
- $B(x, y)$ is the region of support of the flat structuring element.
- $B_c(x, y)$ denotes B displaced by c.

For rectangular filters the domain of support is again rectangular. For circular filters the domain of support is an approximation to a circle. As the size of the structuring element to be iterated increases, the iterated structuring element is a better approximation to a circle.

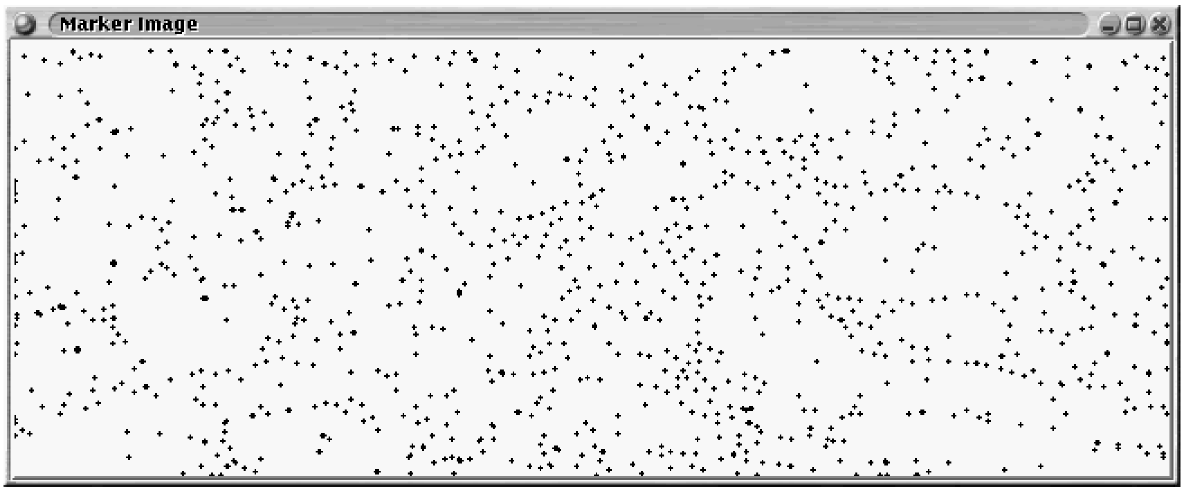
The following observations on the qualitative data can be made:

(i) Dilations merge close maxima. This leads to merging markers and reducing over-segmentation. Conversely, small bubbles close to one another tend to be merged.

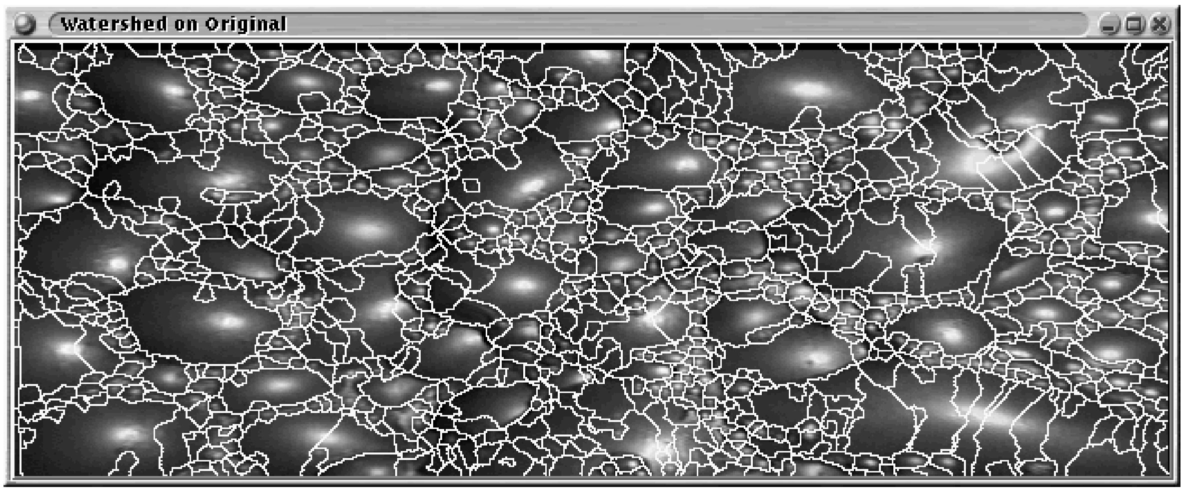
(ii) Erosions fragment maxima. This leads to fragmented markers and hence to over-segmentation.



(a) Denoised image.

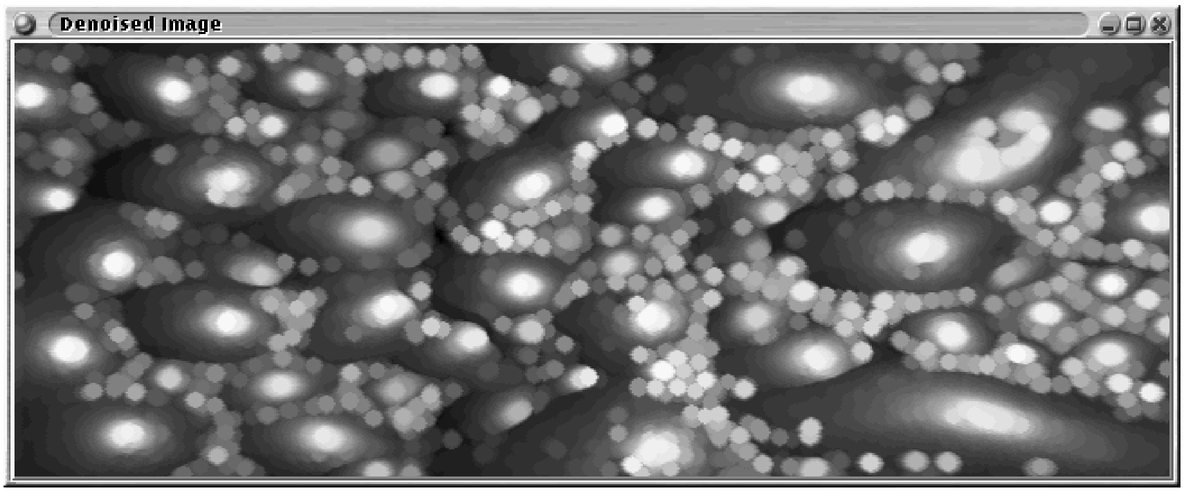


(b) Marker image.

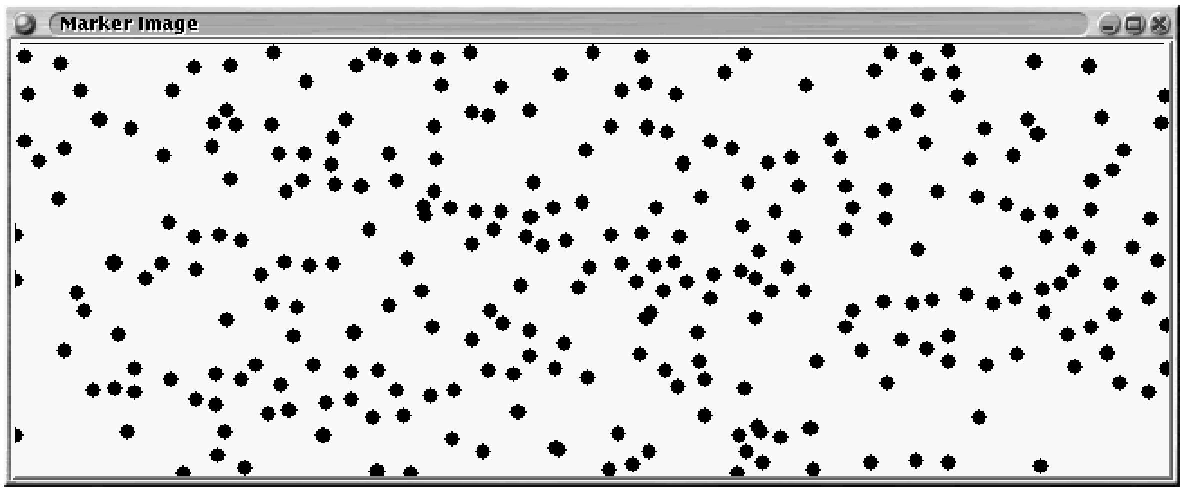


(c) Watershed segmented image.

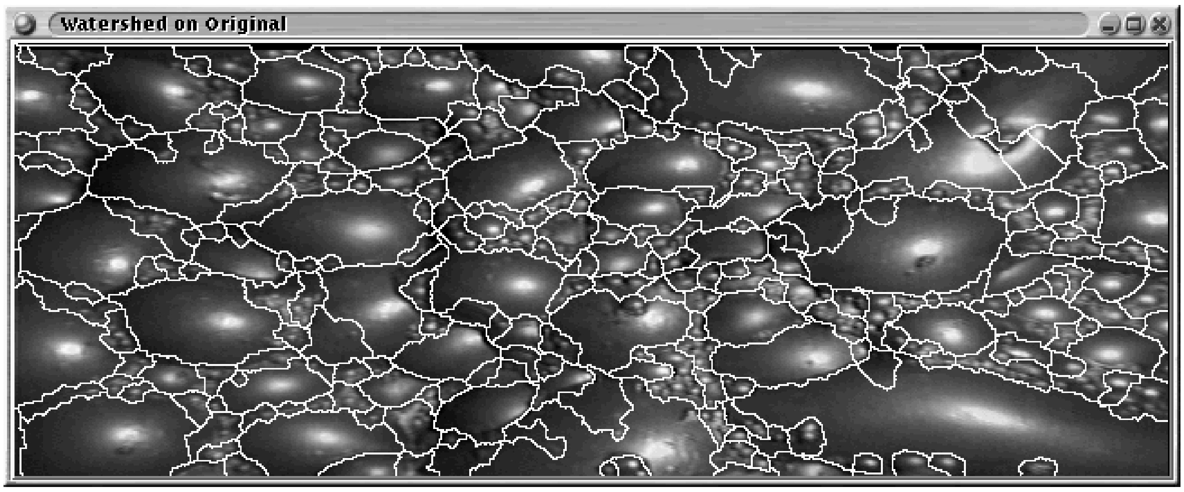
FIGURE 9.1 Result of flat dilation (radius 1) on image, marker and watershed.



(a) Denoised image.

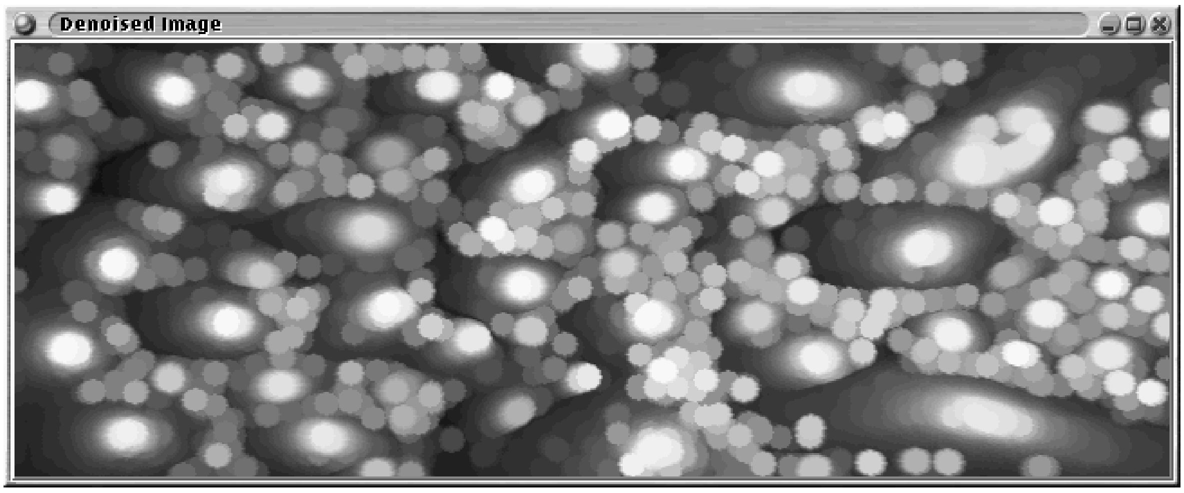


(b) Marker image.

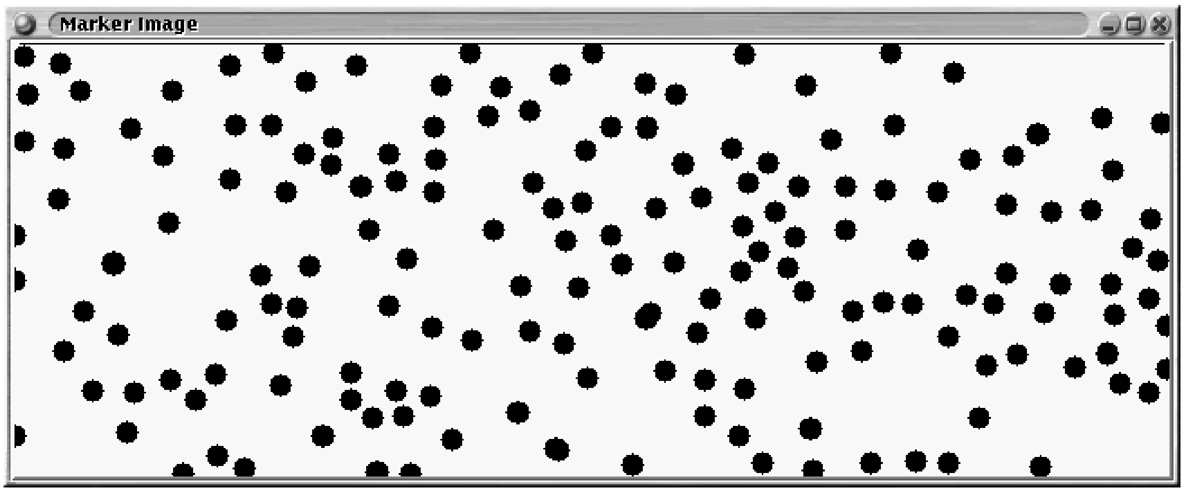


(c) Watershed segmented image.

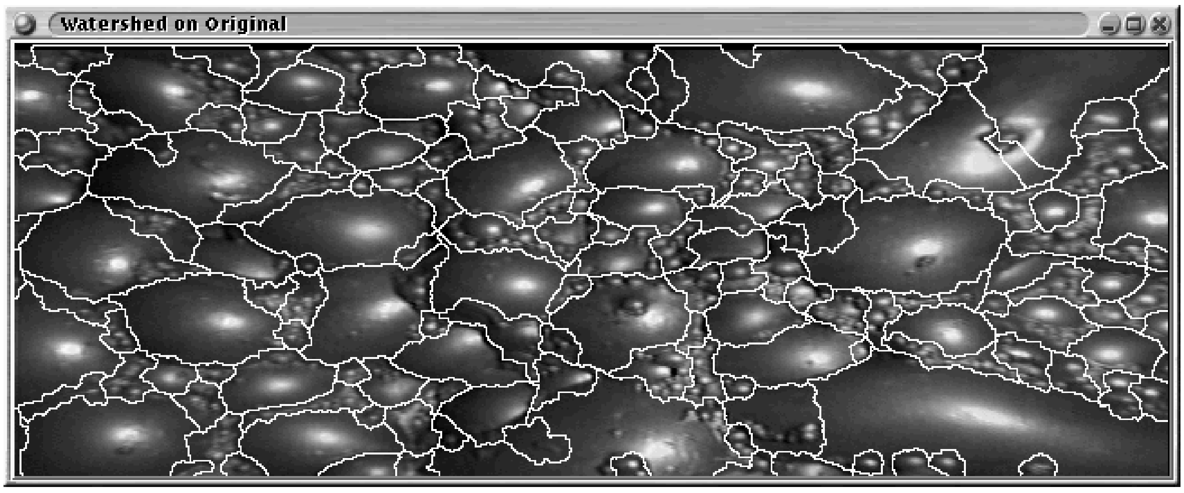
FIGURE 9.2 Result of flat dilation (radius 4) on image, marker and watershed.



(a) Denoised image.

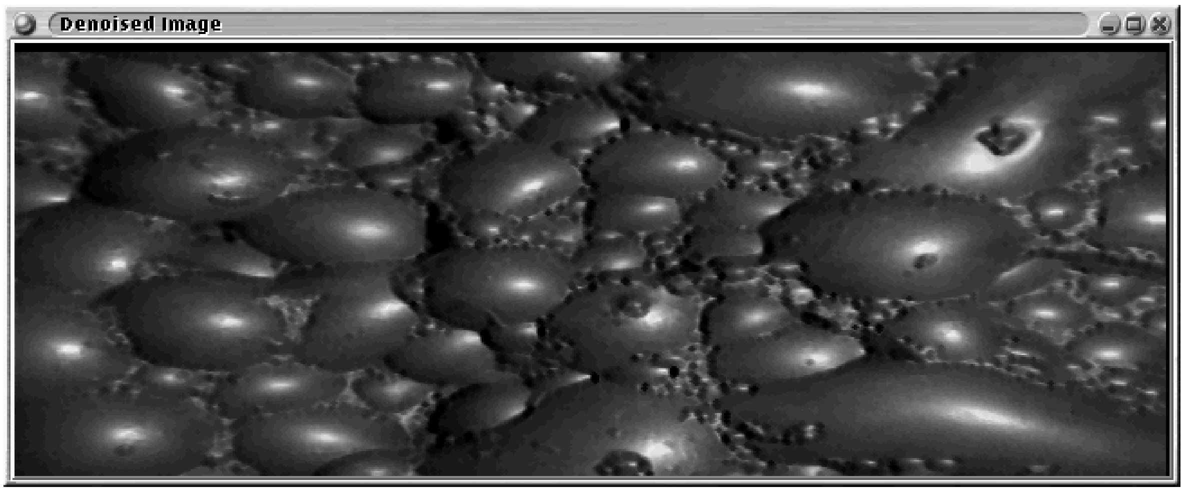


(b) Marker image.

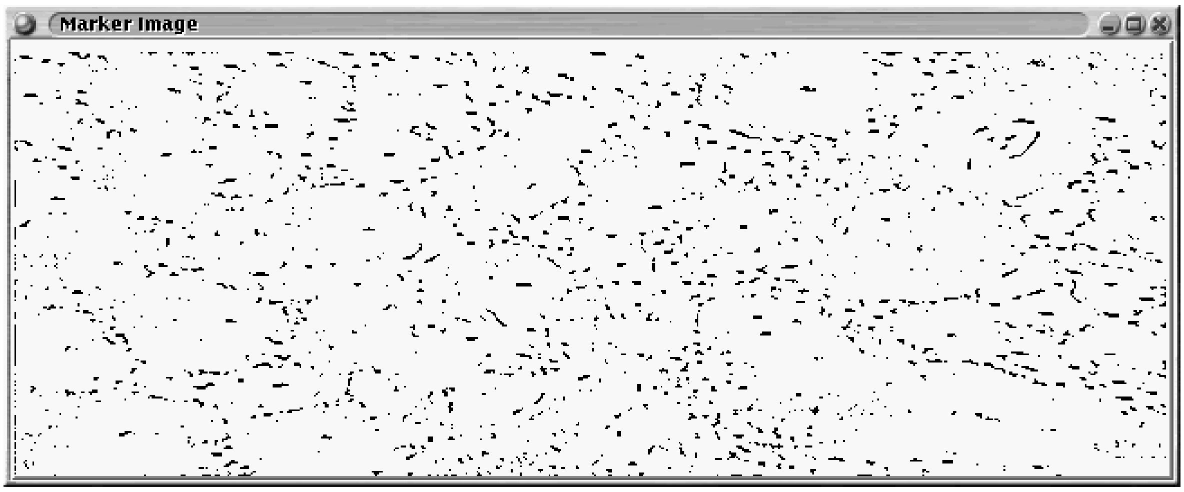


(c) Watershed segmented image.

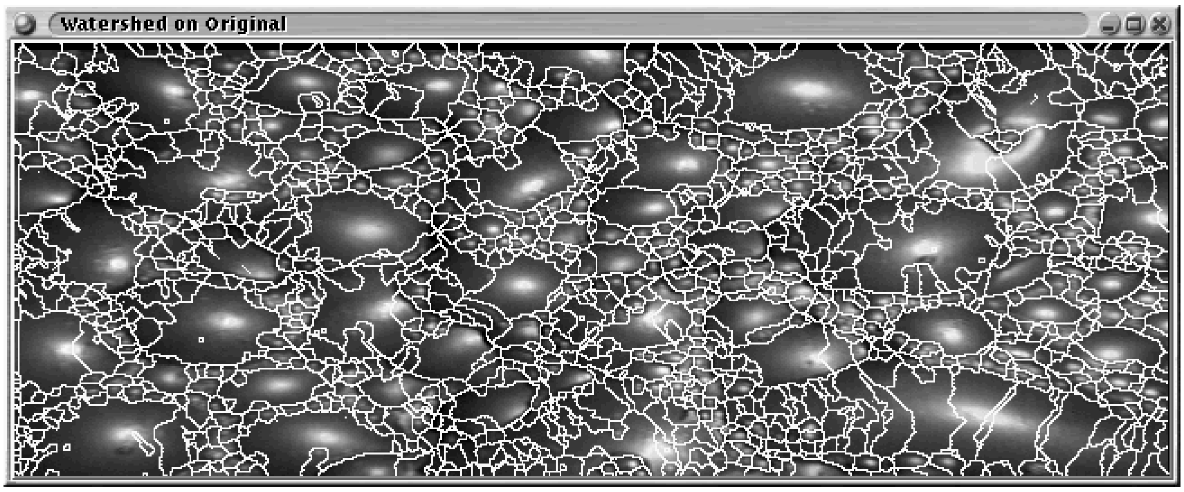
FIGURE 9.3 Result of flat dilation (radius 6) on image, marker and watershed.



(a) Denoised image.

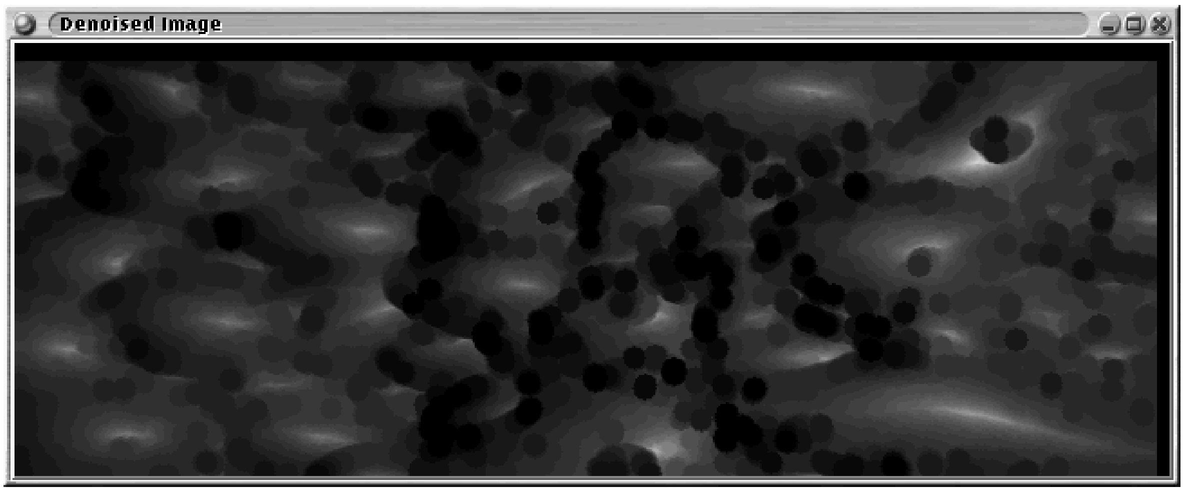


(b) Marker image.

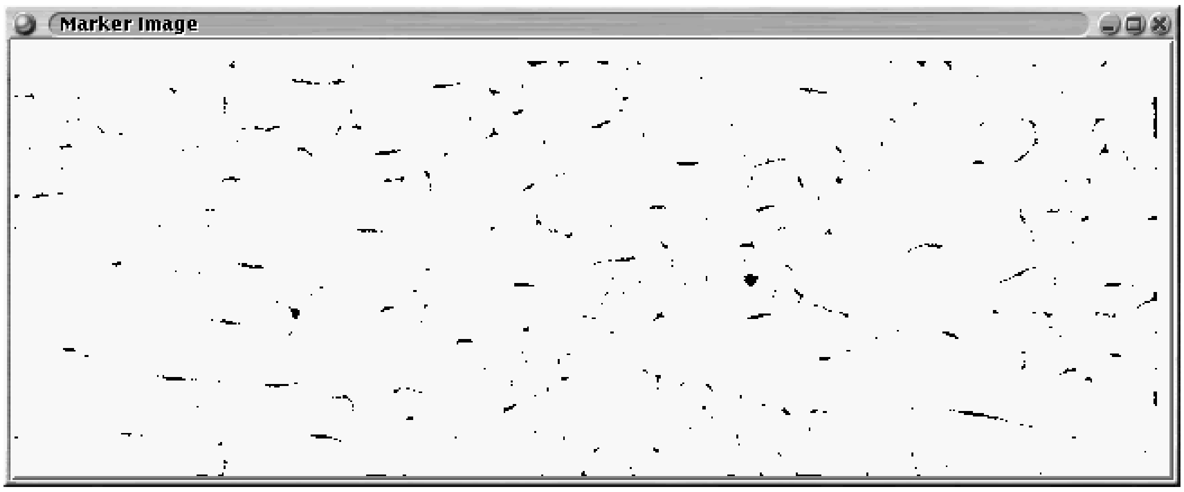


(c) Watershed segmented image.

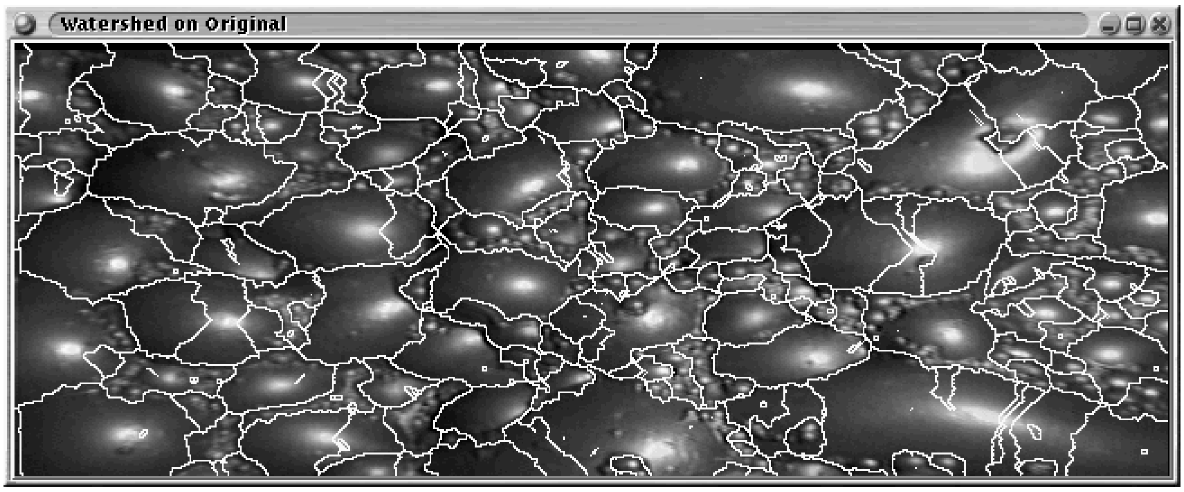
FIGURE 9.4 Result of flat erosion (radius 1) on image, marker and watershed.



(a) Denoised image.

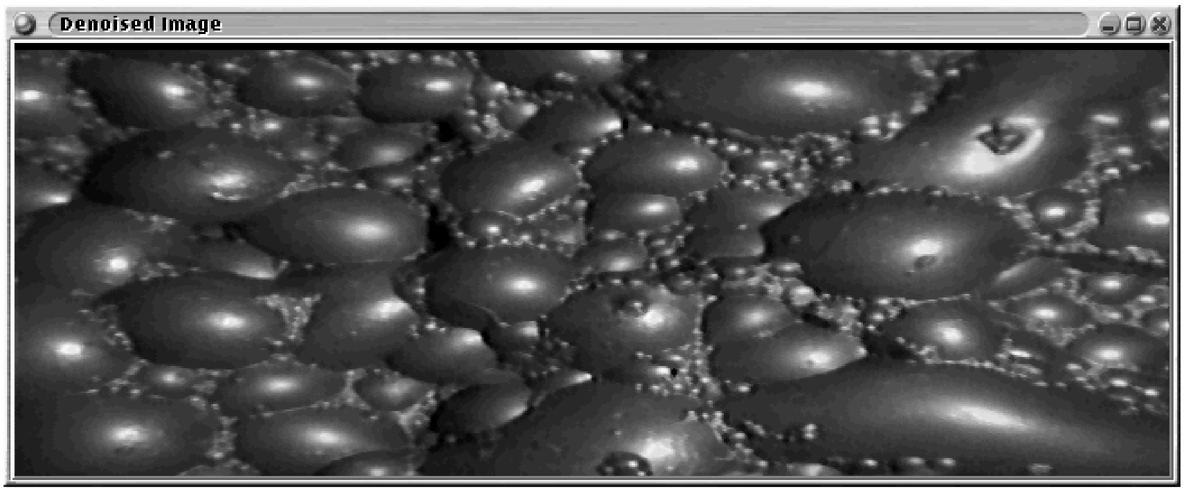


(b) Marker image.

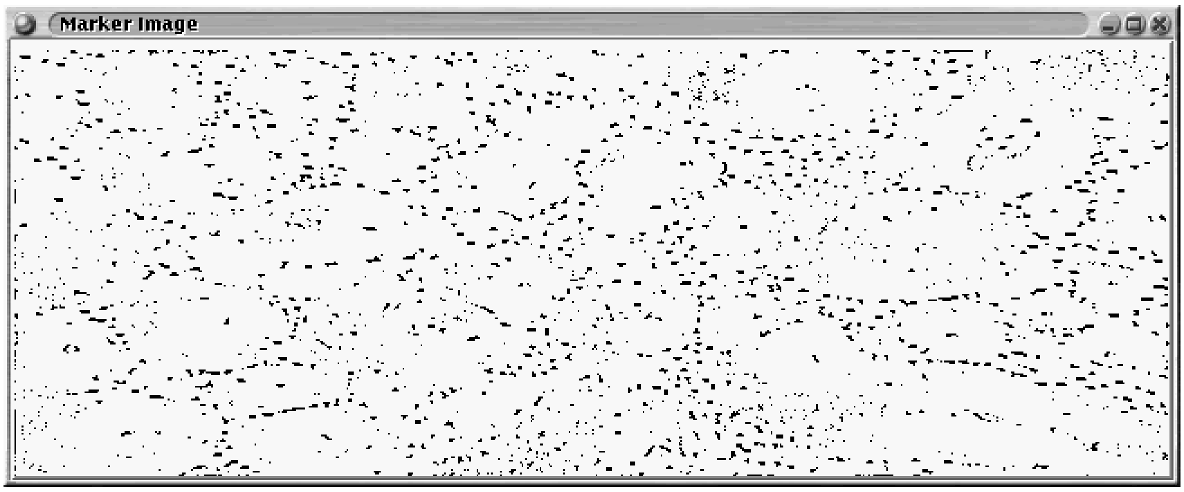


(c) Watershed segmented image.

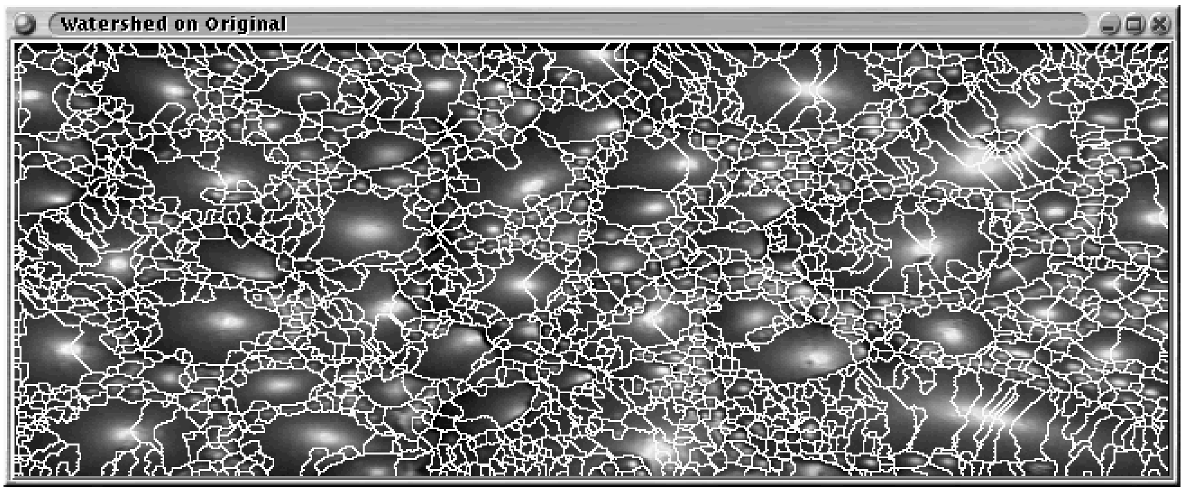
FIGURE 9.5 Result of flat erosion (radius 6) on image, marker and watershed.



(a) Denoised image.

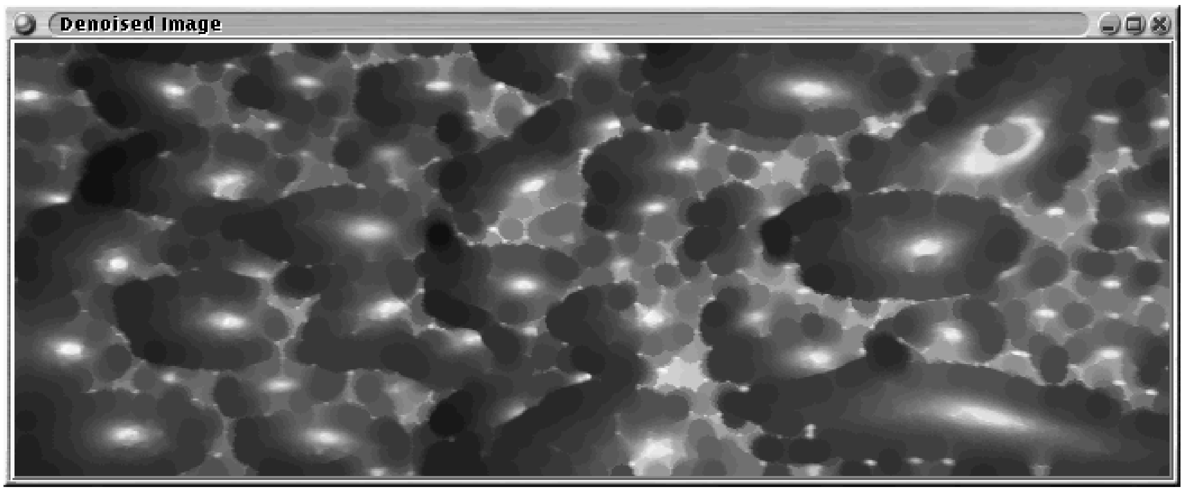


(b) Marker image.

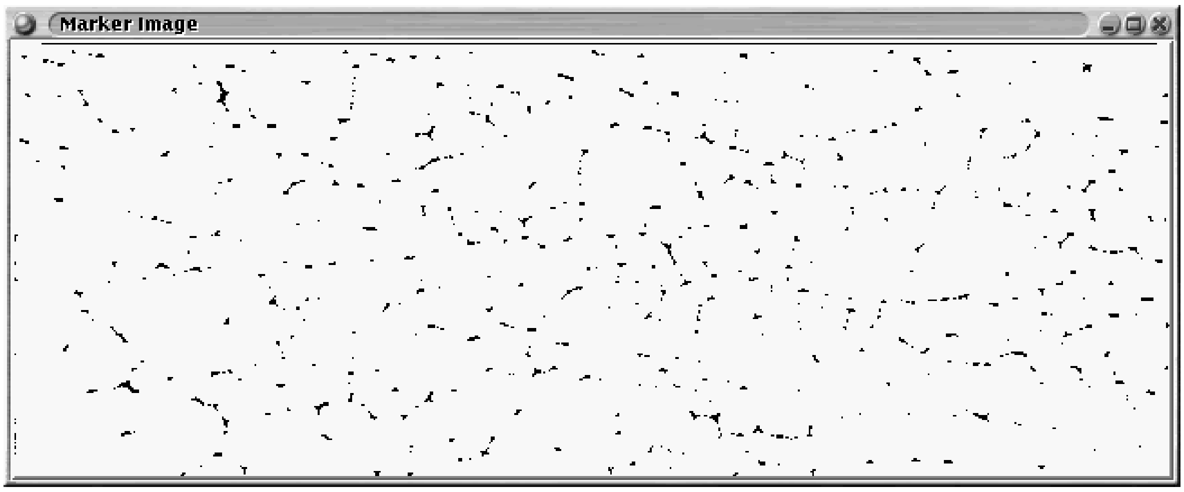


(c) Watershed segmented image.

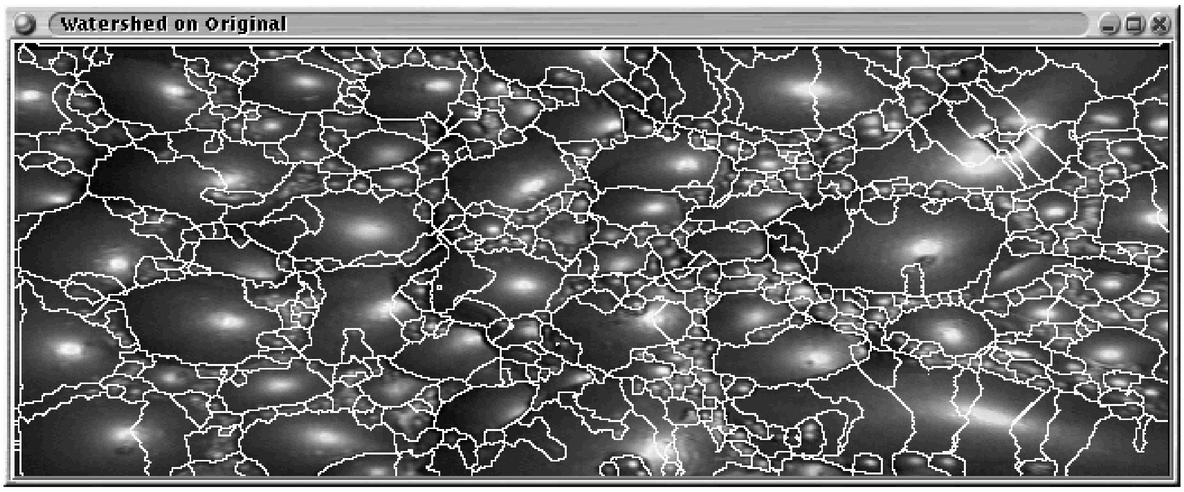
FIGURE 9.6 Result of flat closing (radius 1) on image, marker and watershed.



(a) Denoised image.

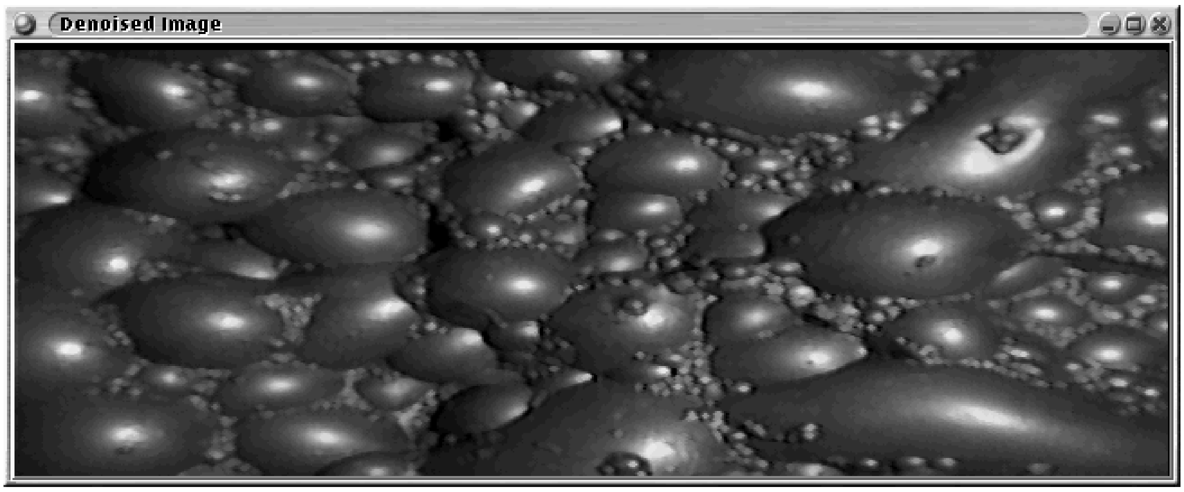


(b) Marker image.

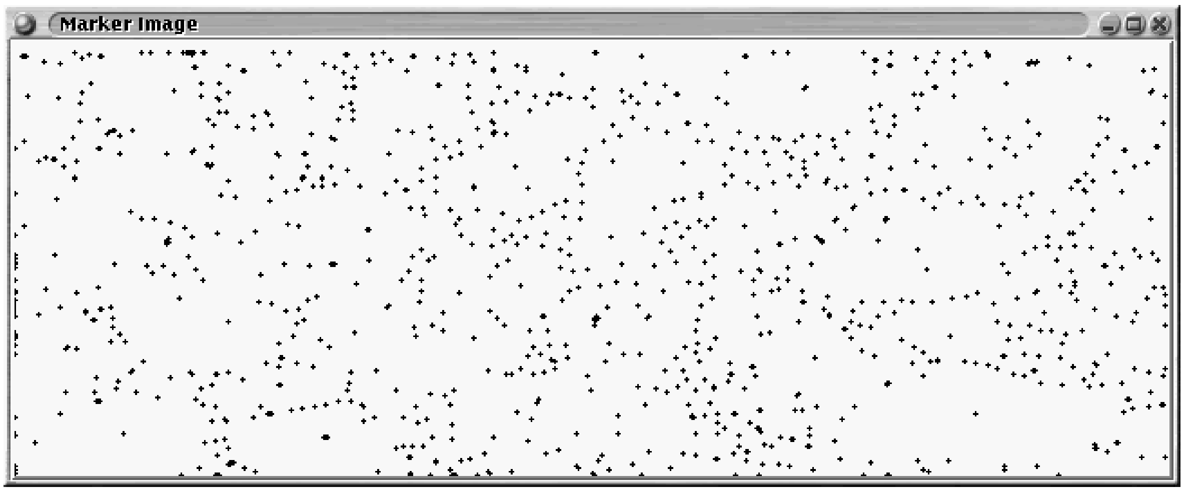


(c) Watershed segmented image.

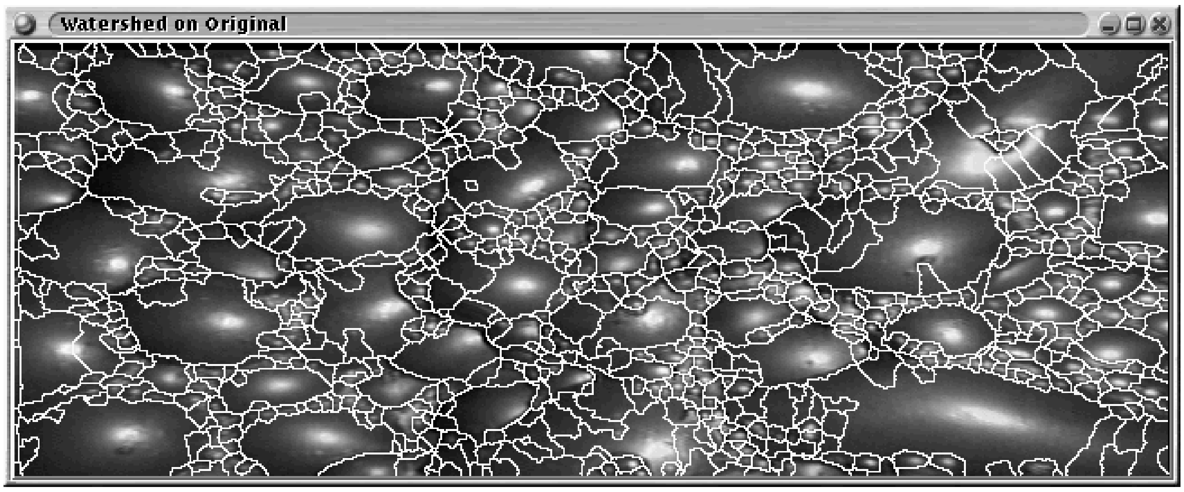
FIGURE 9.7 Result of flat closing (radius 6) on image, marker and watershed.



(a) Denoised image.

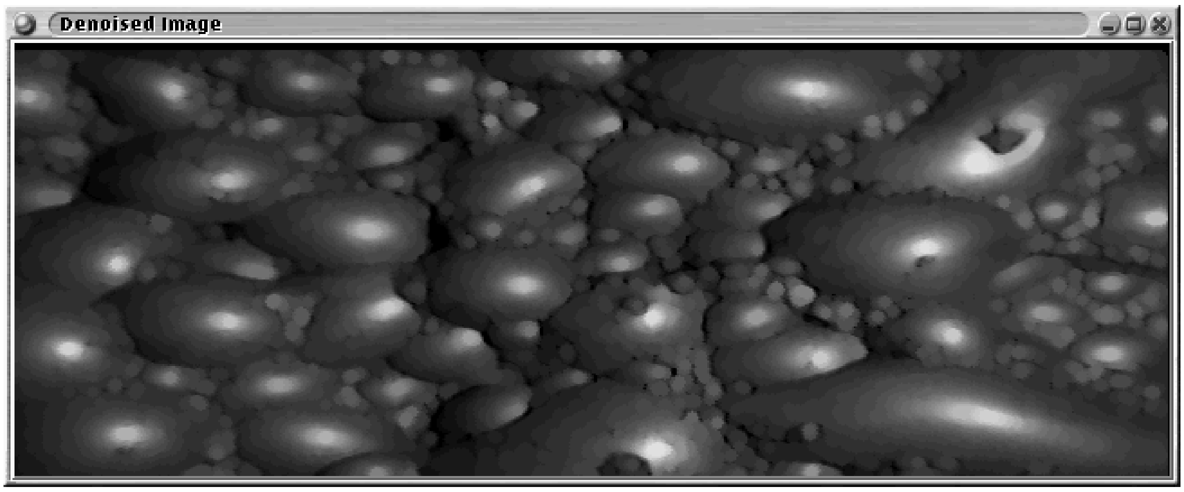


(b) Marker image.

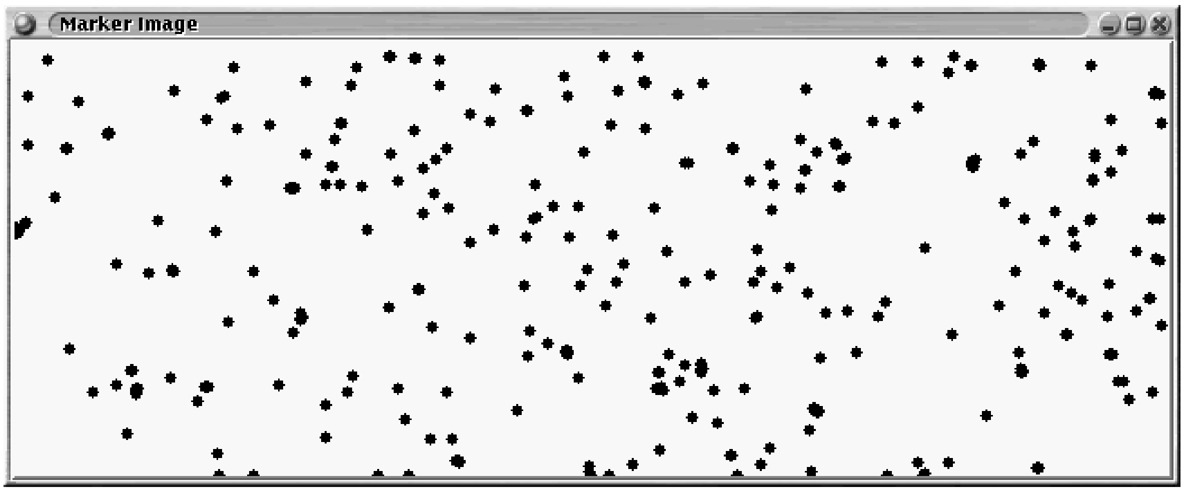


(c) Watershed segmented image.

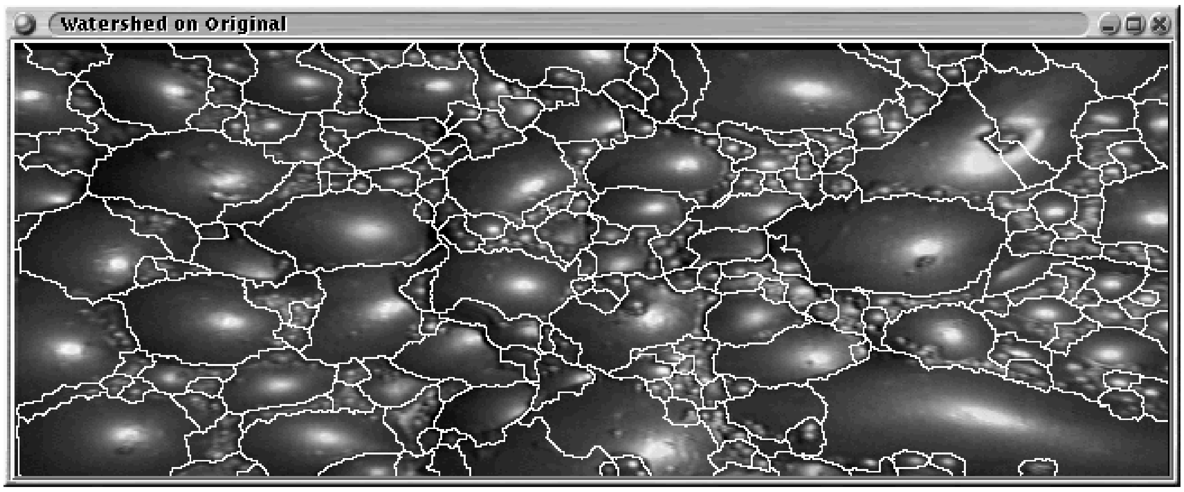
FIGURE 9.8 Result of flat opening (radius 1) on image, marker and watershed.



(a) Denoised image.

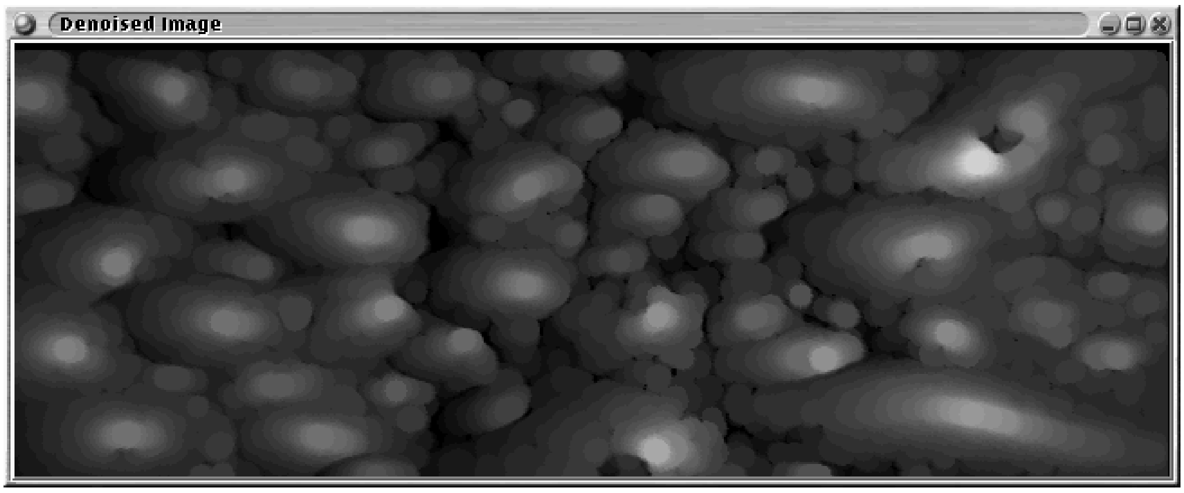


(b) Marker image.

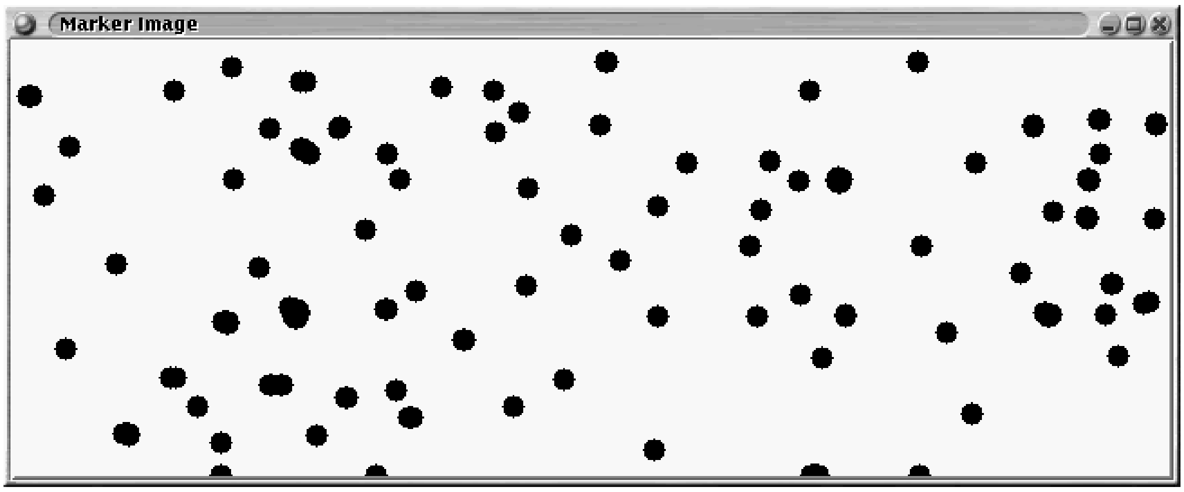


(c) Watershed segmented image.

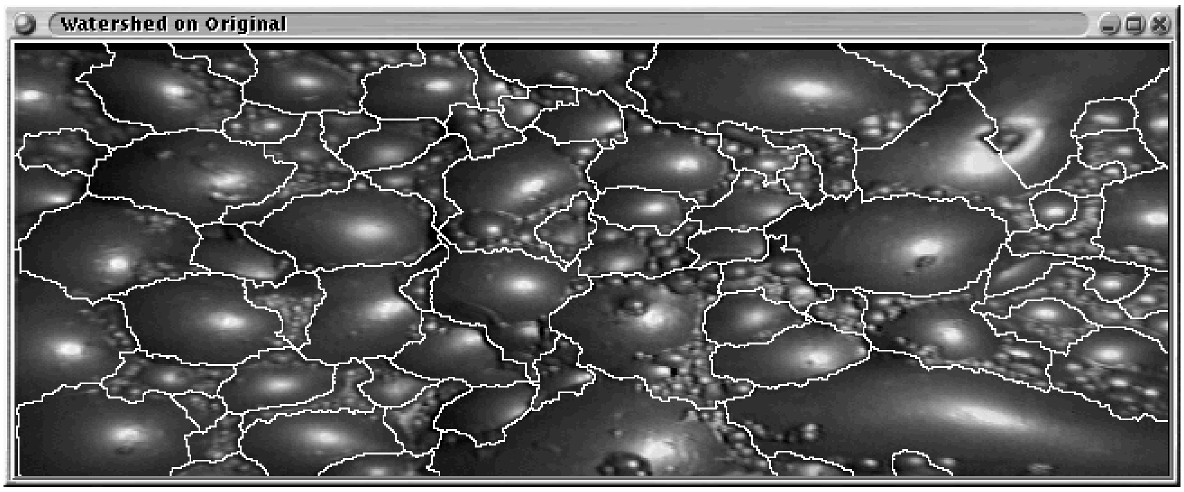
FIGURE 9.9 Result of flat opening (radius 3) on image, marker and watershed.



(a) Denoised image.

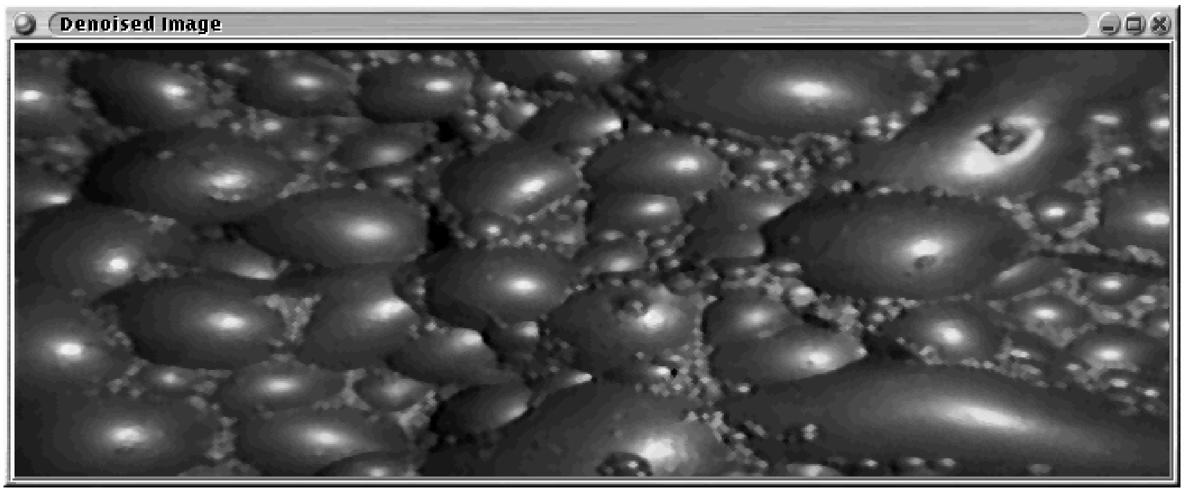


(b) Marker image.

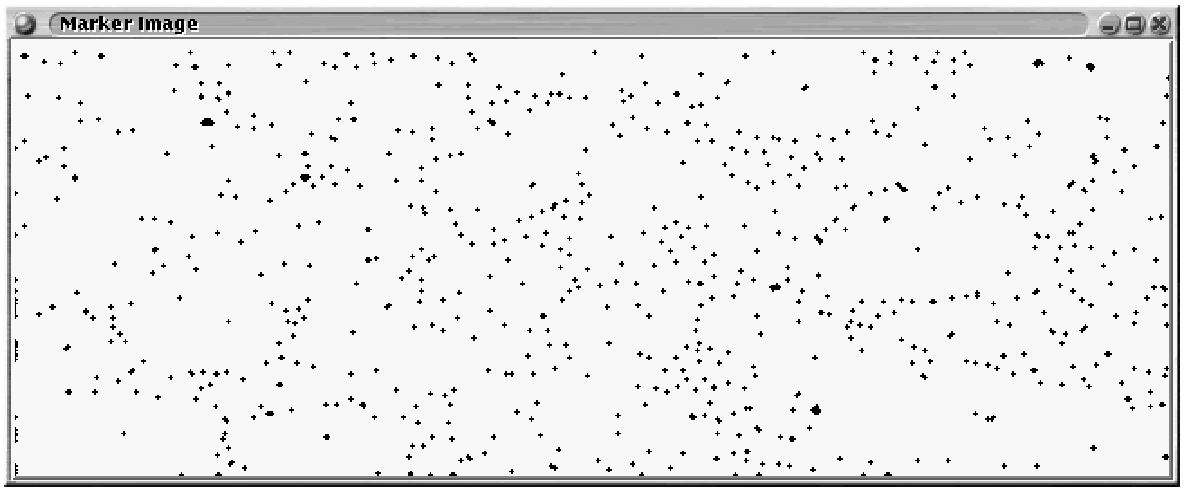


(c) Watershed segmented image.

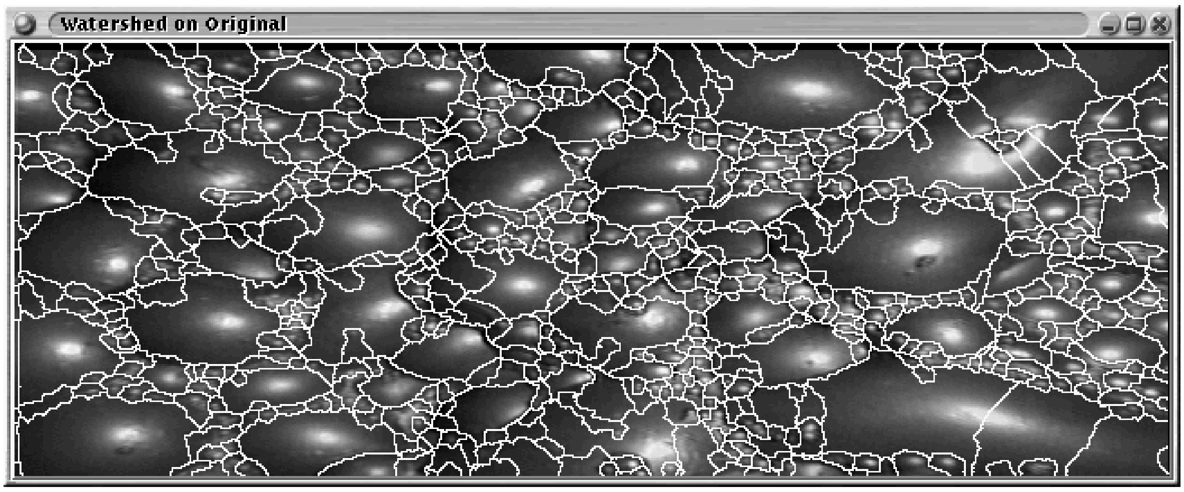
FIGURE 9.10 Result of flat opening (radius 6) on image, marker and watershed.



(a) Denoised image.

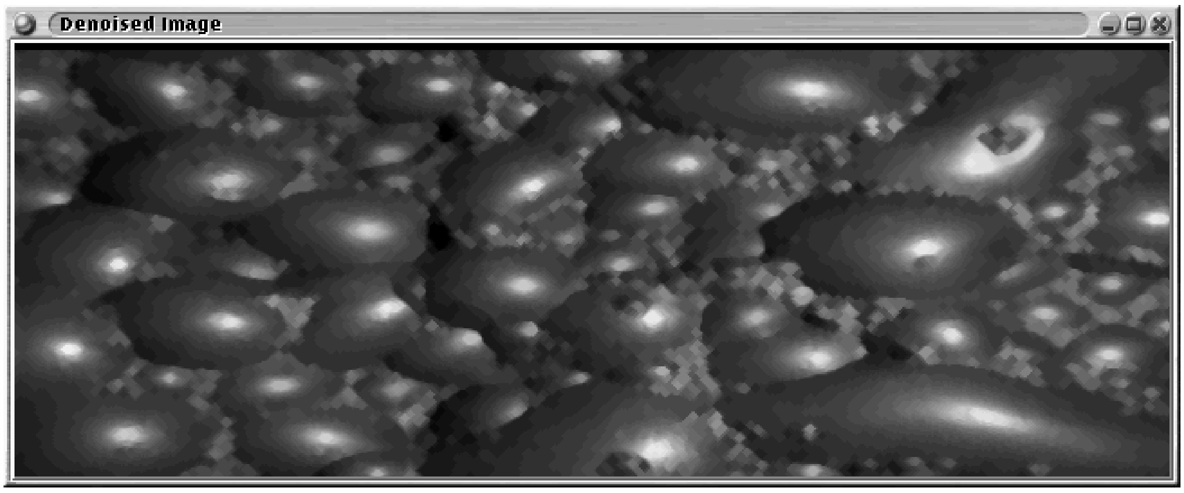


(b) Marker image.

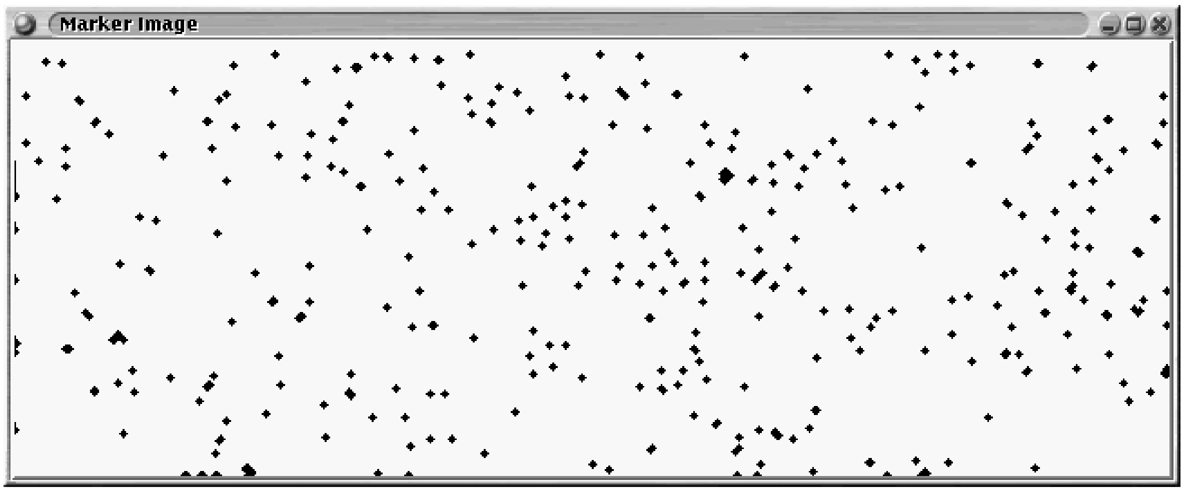


(c) Watershed segmented image.

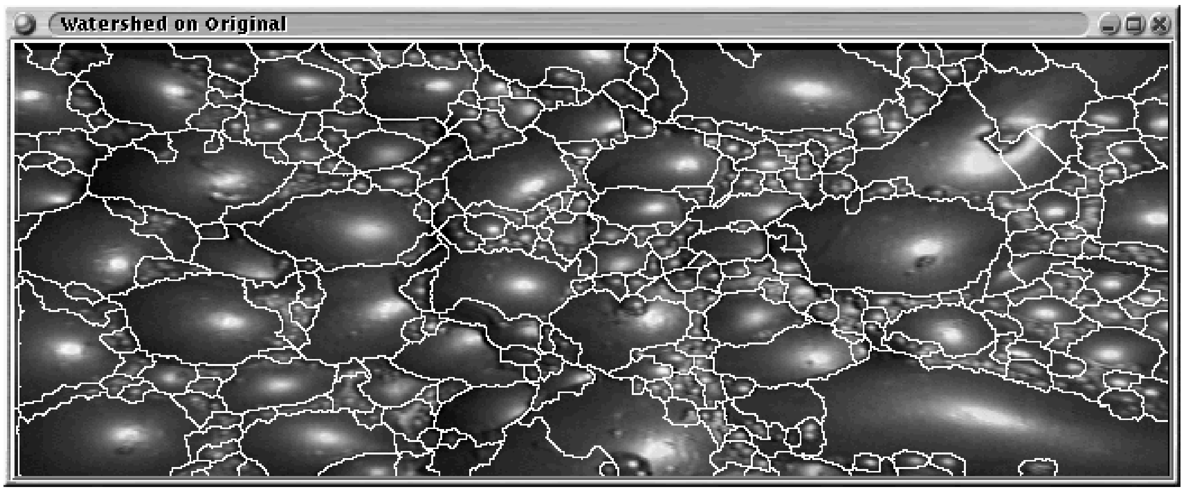
FIGURE 9.11 Result of flat close-opening (radius 1) on image, marker and watershed.



(a) Denoised image.

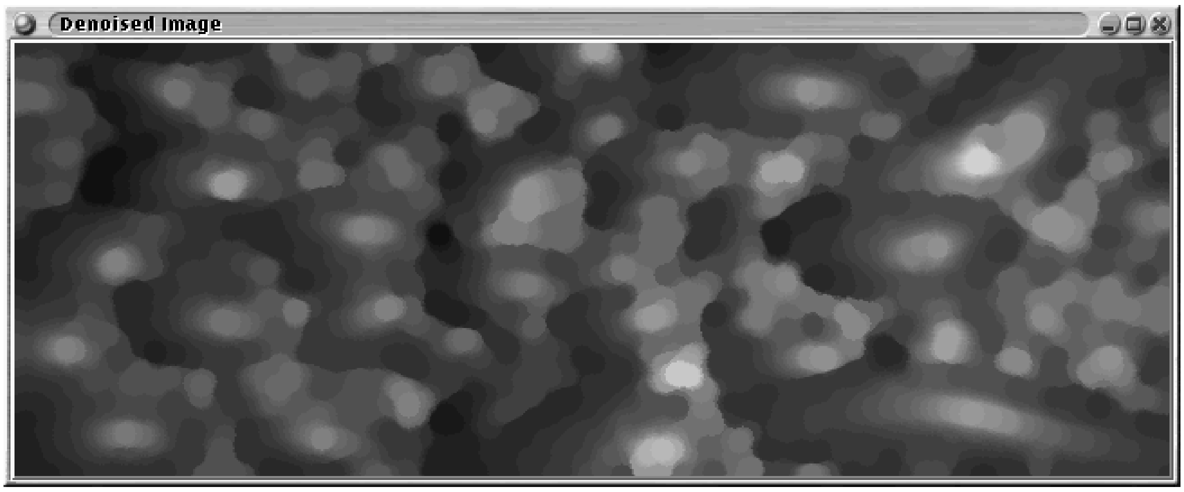


(b) Marker image.

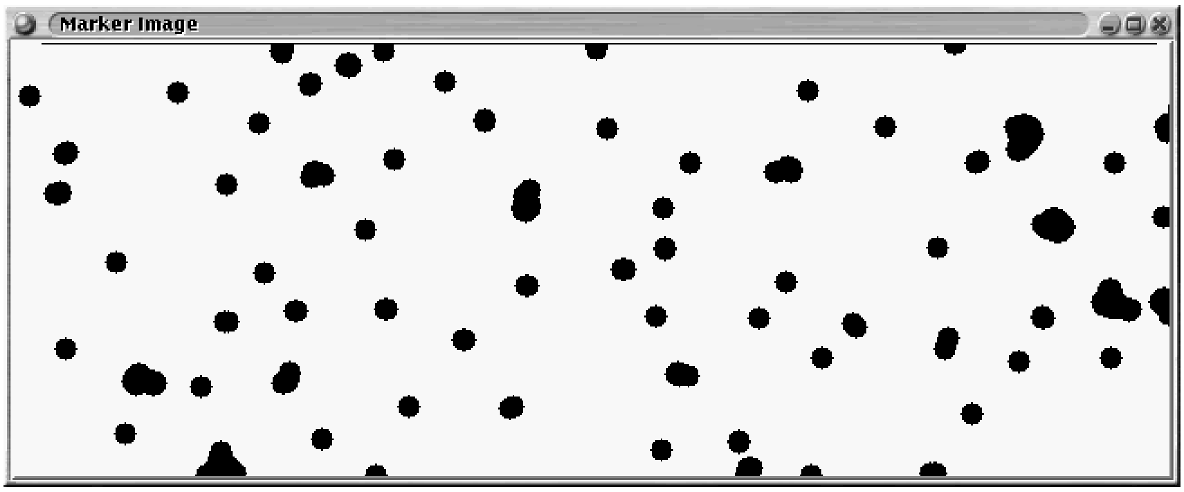


(c) Watershed segmented image.

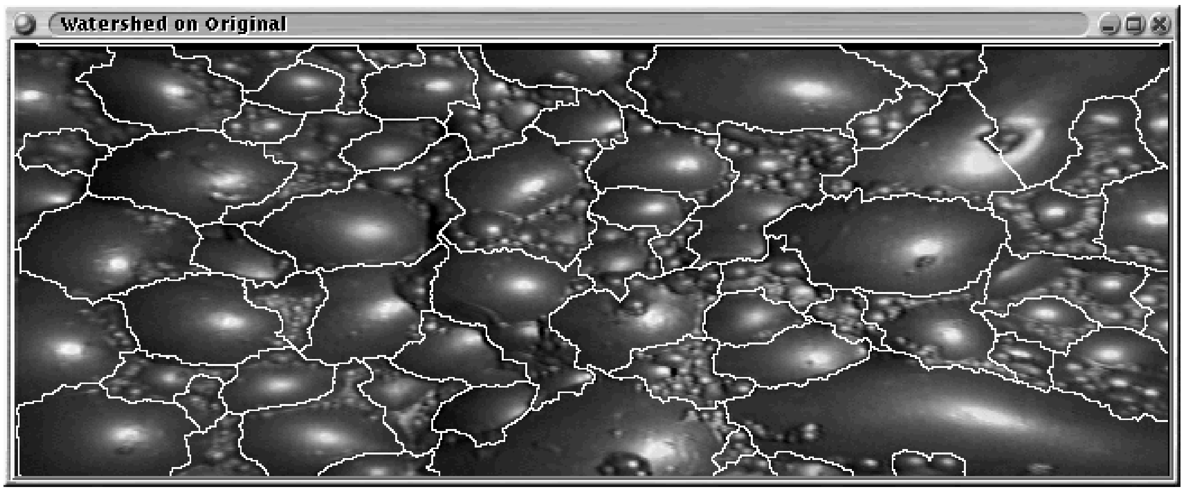
FIGURE 9.12 Result of flat close-opening (radius 2) on image, marker and watershed.



(a) Denoised image.

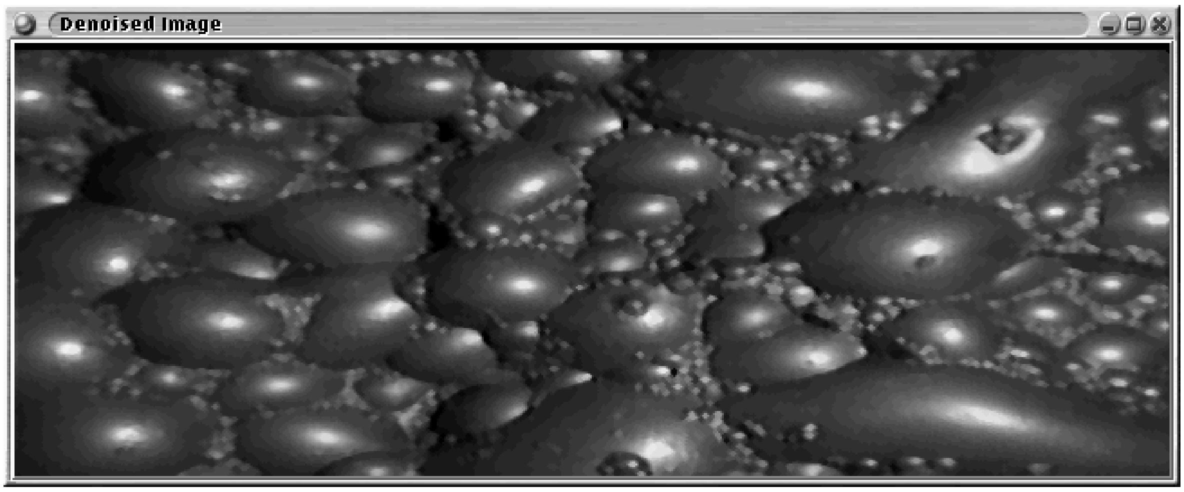


(b) Marker image.

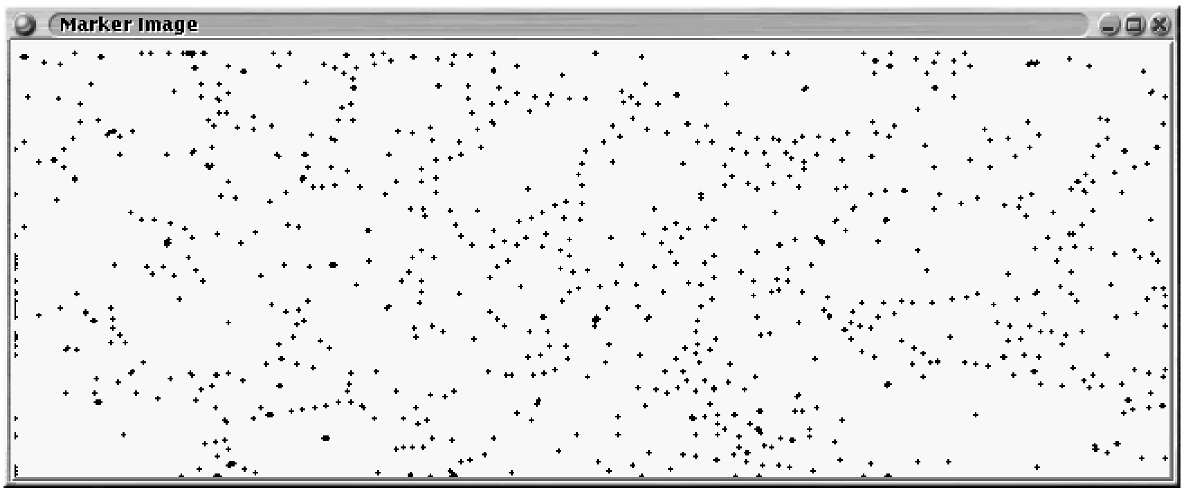


(c) Watershed segmented image.

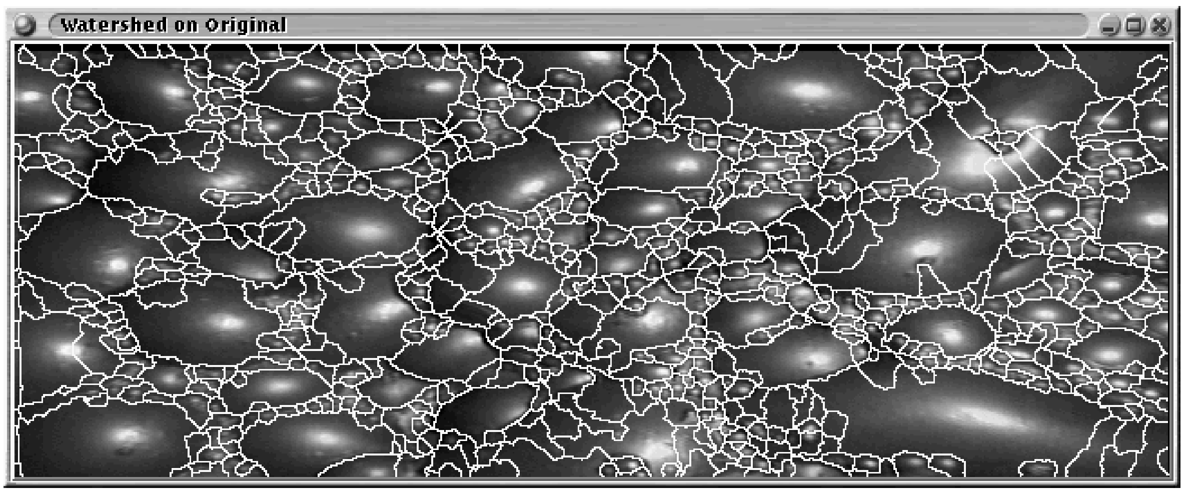
FIGURE 9.13 Result of flat close-opening (radius 6) on image, marker and watershed.



(a) Denoised image.

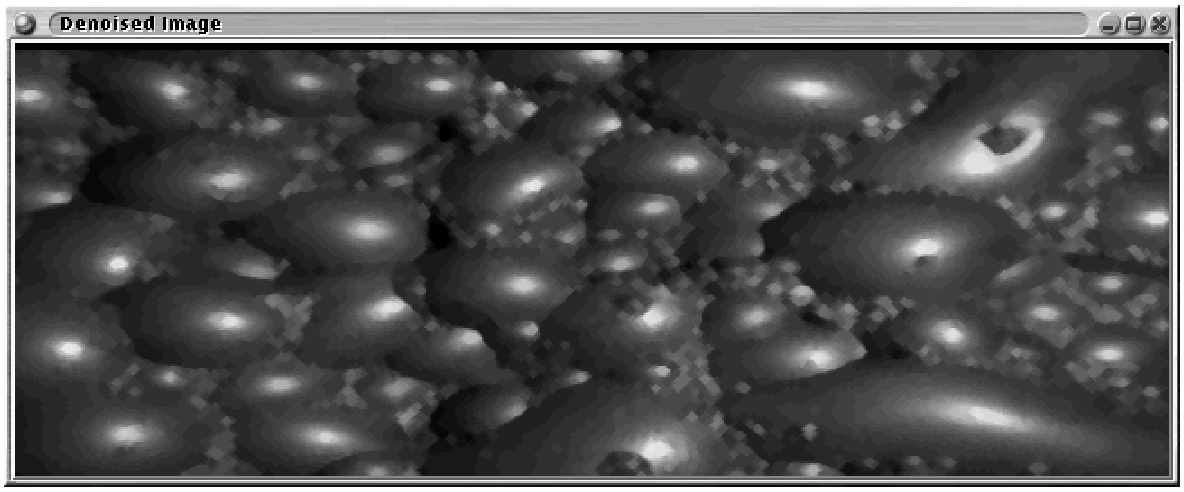


(b) Marker image.

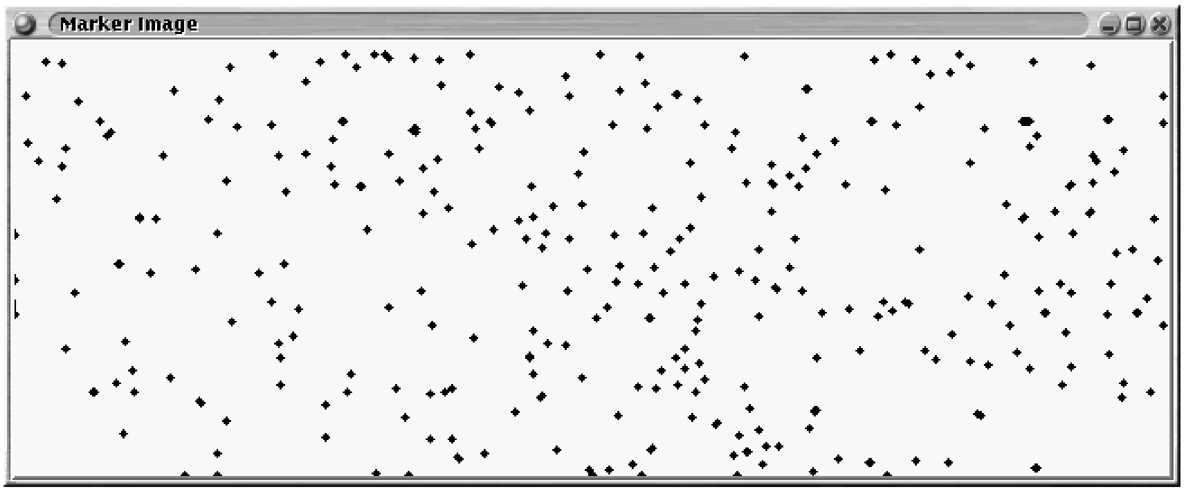


(c) Watershed segmented image.

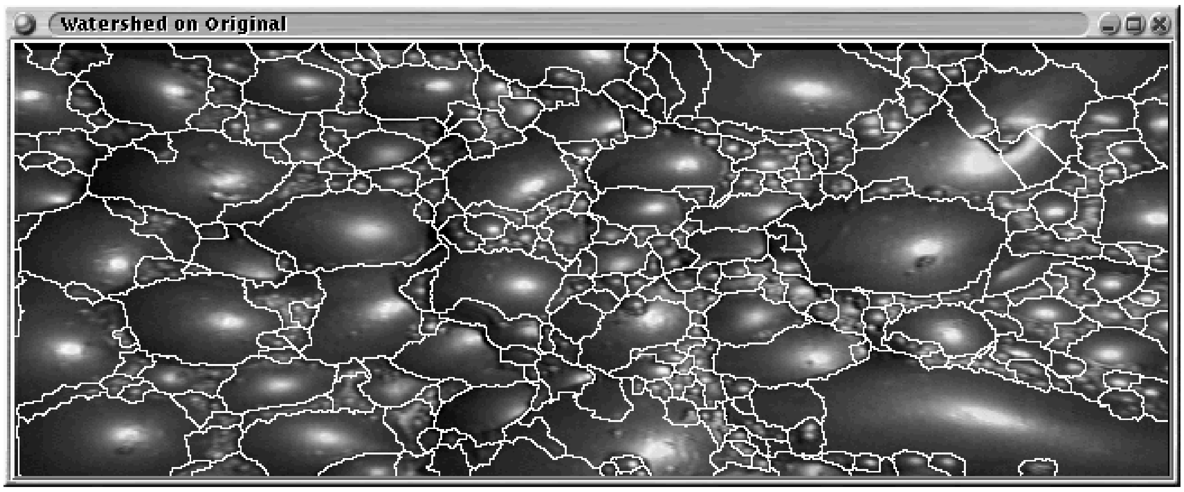
FIGURE 9.14 Result of flat open-closing (radius 1) on image, marker and watershed.



(a) Denoised image.

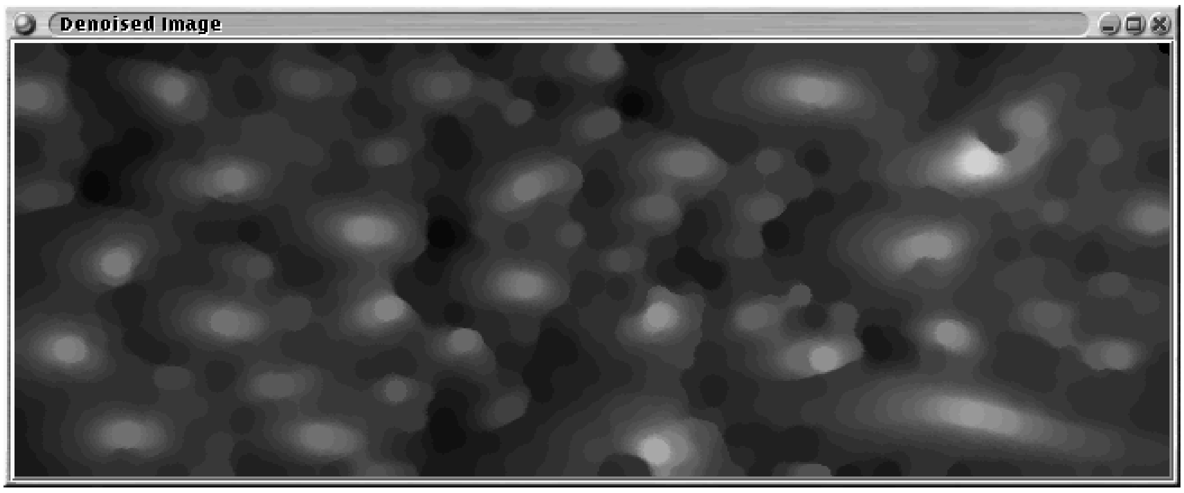


(b) Marker image.

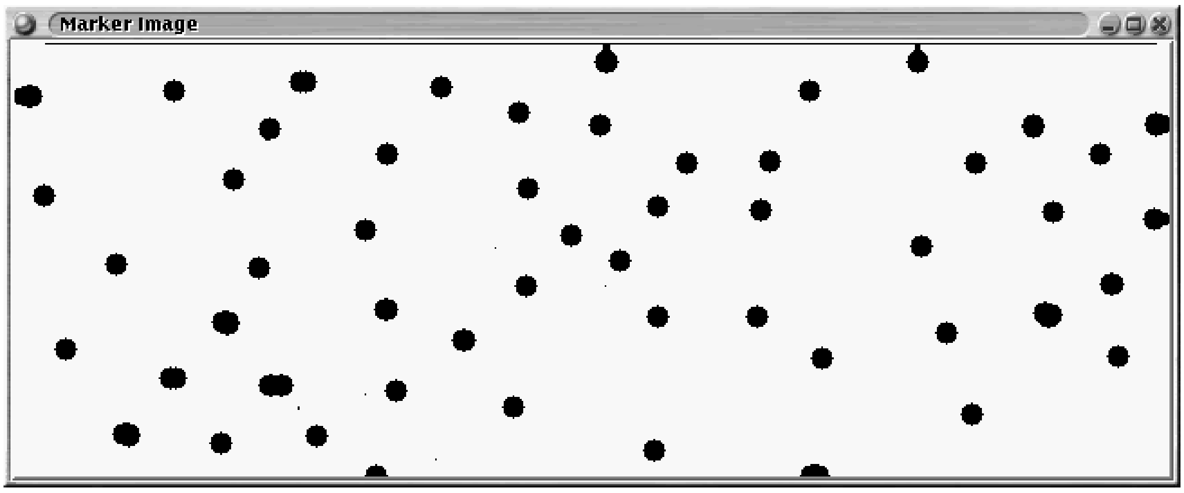


(c) Watershed segmented image.

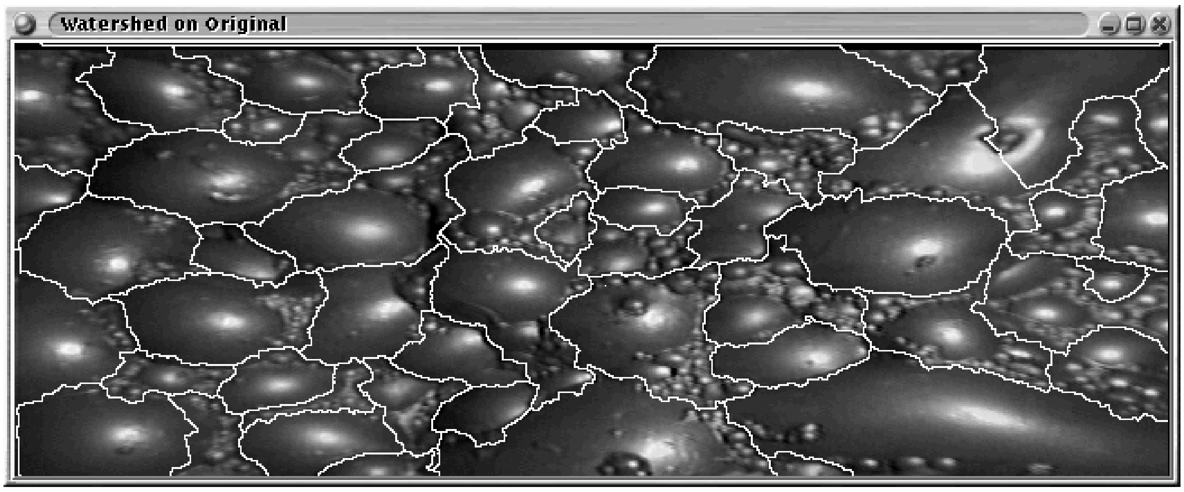
FIGURE 9.15 Result of flat open-closing (radius 2) on image, marker and watershed.



(a) Denoised image.



(b) Marker image.



(c) Watershed segmented image.

FIGURE 9.16 Result of flat open-closing (radius 6) on image, marker and watershed.

9.2 Rolling ball filters

This is somewhat traditional for froth imaging [89, 135]. The structuring element used is an upper circular hemisphere for rolling balls rolling on the top of a grey level surface. (If one chooses to interpret a grey-level image as a topological height map.) Alternately, the structuring element is a lower circular hemisphere for rolling a ball underneath the grey-level surface.

Conceptually, a rolling ball filter works by rolling a ball along either the top or underside of a grey level surface. This creates a surface traced out by the locus of the movement of the centre of the ball. Depending on the width of the ball some valleys (if rolling on the top) or some peaks (if rolling on the bottom) will be not be fully traced. In effect these peaks or valleys have been eroded or filled.

Mathematical morphology simulates this rolling ball process by means or erosion and dilation. Erosion (in the Mathematical Morphology sense [64]) of the grey-level surface and a circular shell yields the upper surface. Similarly for the dilation of the grey-level surface and a circular shell. An optimisation replaces the circular shell by a upper or lower circular hemispherical shell for dilation and erosion respectively.

One has two alternatives for rolling ball filters: one may use a erosion followed by a dilation to remove minima as well as maxima, or one may abandon symmetry and simply add a constant gray level to the eroded image equal to the radius of the rolling ball. Adding a constant preserves the intensity of flat and smoothly varying regions.

The following morphological operators are tested for denoising froth images:

(i) Dilation.

(ii) Erosion.

(iii) Dilation followed by subtracting a constant [64] (Dilation Shift).

(iv) Erosion followed by adding a constant [64] (Erosion Shift).

(v) Closing. A dilation followed by an erosion [34, 64].

(vi) Opening. An erosion followed by a dilation [34, 64].

(vii) Close-Openings. A closing followed by an opening [34, 64].

(viii) Open-Closings. An opening followed by a closing [34, 64].

Denoising for cleaning up a watershed segmentation implies that one discards minima which are not likely to be significant. This implies a dilation of the original image (since the watershed for froth images operates on the inverted froth image). This implies that a morphological closing is unnecessary (though harmless).

Table 9.2 shows the effect on the number of watershed regions (or, surviving markers) as the radius of the rolling ball structuring element increases. As expected the number of watershed regions for the Dilation Shift and Erosion Shift cases is the same as that for Dilation and Erosion respectively.

Figures 9.17 through 9.30 show the effect of applying the filters with different ball radii on the images, markers and watershed segmentation. The standard test image is used.

TABLE 9.2 Number of watershed regions against rolling ball radius.

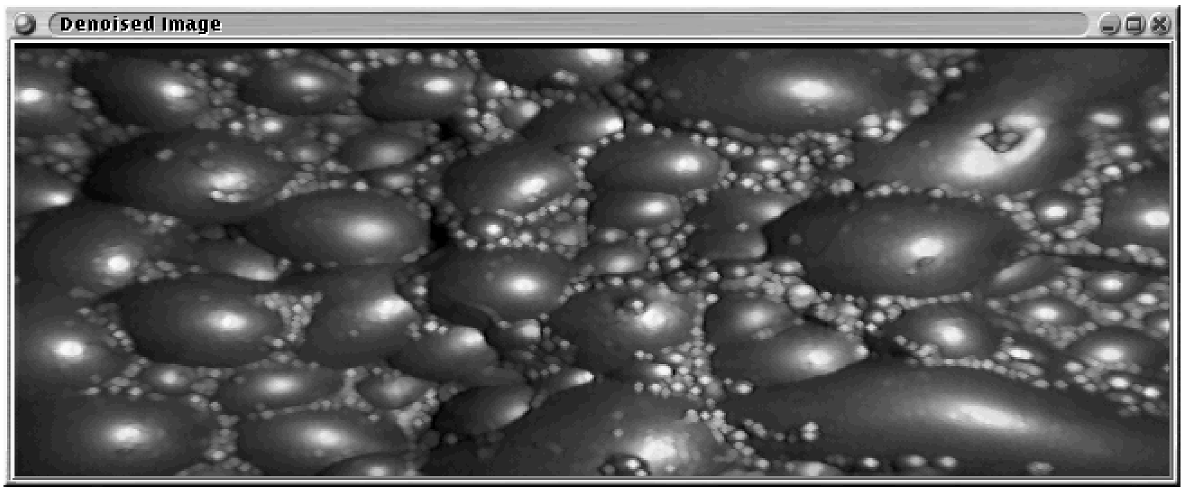
Radius	# Watershed Regions					
	Dilation Dilation-Shift	Erosion Erosion-Shift	Closing	Opening	Clos-opening	Open-closing
1	1832	1561	2080	1405	1090	1247
2	804	1222	1490	593	440	440
3	550	806	1021	352	250	223
4	461	629	864	244	176	155
5	336	463	719	161	141	105
6	284	342	567	128	106	81

9.3 Denoising via posterisation

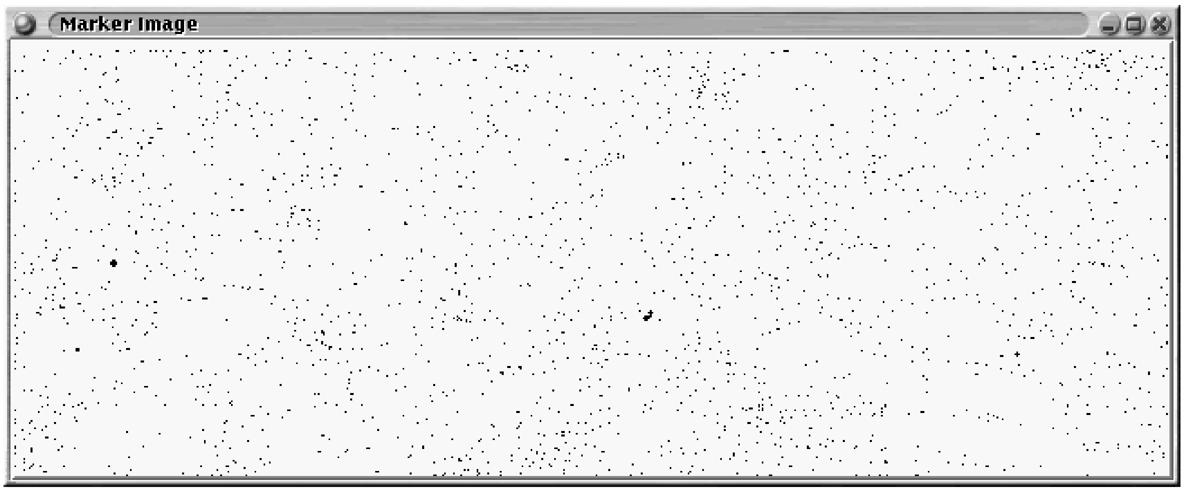
Here posterisation [141] refers to applying the watershed segmentation algorithm on a gradient image to yield the cachement areas. The gradient used here is the flat morphological gradient (the difference between the flat dilated and the flat eroded image, see Appendix B for an introduction to mathematical morphology). The markers used are then the positions of the local maxima and minima of the image (the zero intensity regions of the gradient image). Each pixel in a labelled region is replaced by the mean, or perhaps some intensity dependent function, of the pixels within the labelled area. The intensities used are those of the original image. Naturally, the algorithm may be iterated until a desired smoothing is achieved.

Table 9.3 shows the decrease in the number of watershed regions as the denoising algorithm is iterated. This is expected, since the marker for a region will disappear unless the mean is larger than the mean of the surrounding regions. Figure 9.31 shows the results of applying posterisation to a froth image. The denoising is extreme. The denoised image is made up of piece-wise constant regions.

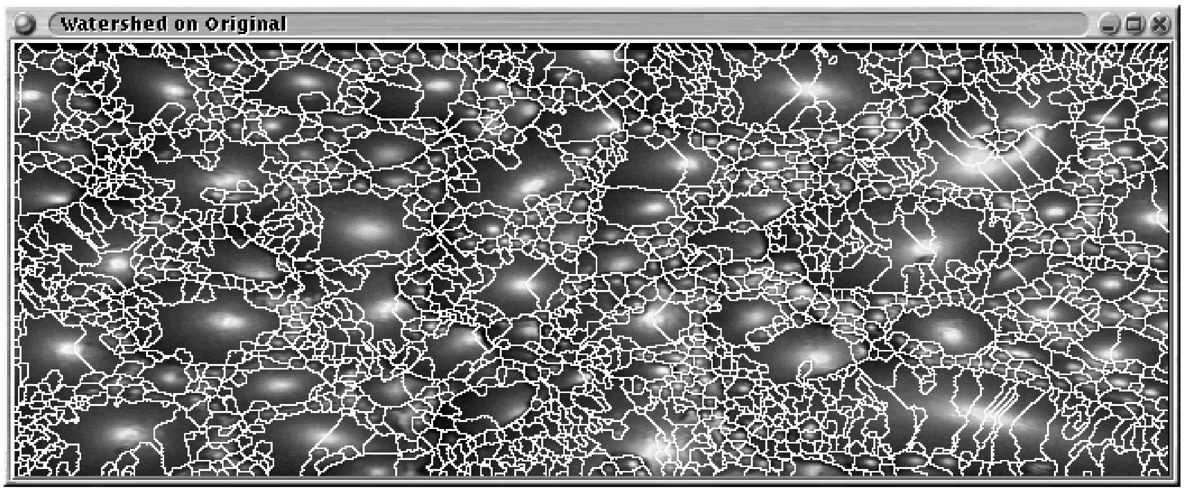
In order to test the hypothesis that adding noise can improve the posterisation, noise is added to the standard froth image. The noise is zero mean and Gaussian distributed with standard deviation 25. The posterised image, associated markers, and watershed segmentation is shown in figure 9.32. The denoised image better resembles a froth image. The marker image shows a number of small markers and blocky irregularly shaped larger markers. The segmentation is no better than in the no-noised added case, showing a number of small watershed regions and fragmented larger regions. Both observed effects can be explained by the noise added. Neither large, nor small, bubbles are well segmented.



(a) Denoised image.

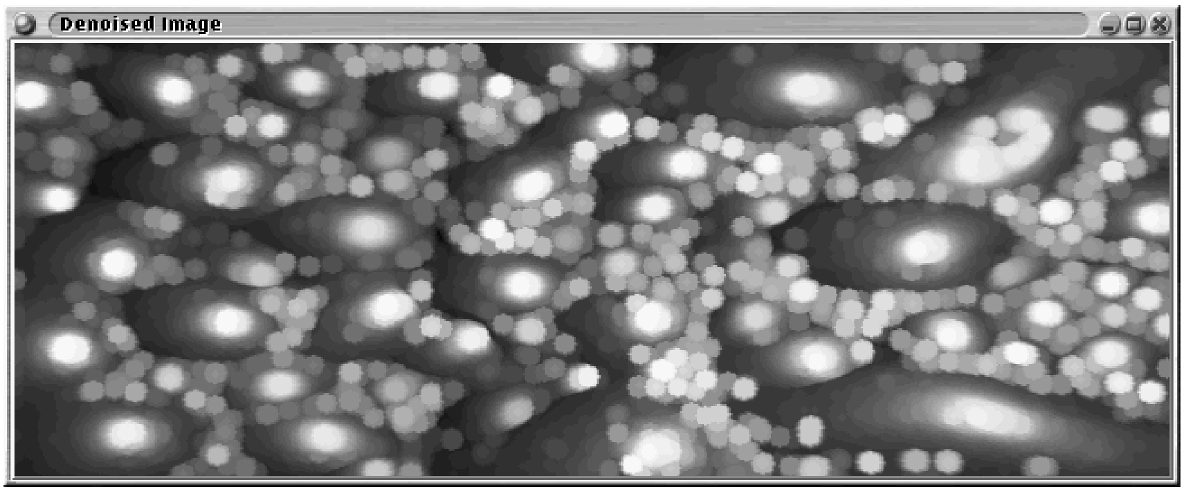


(b) Marker image.

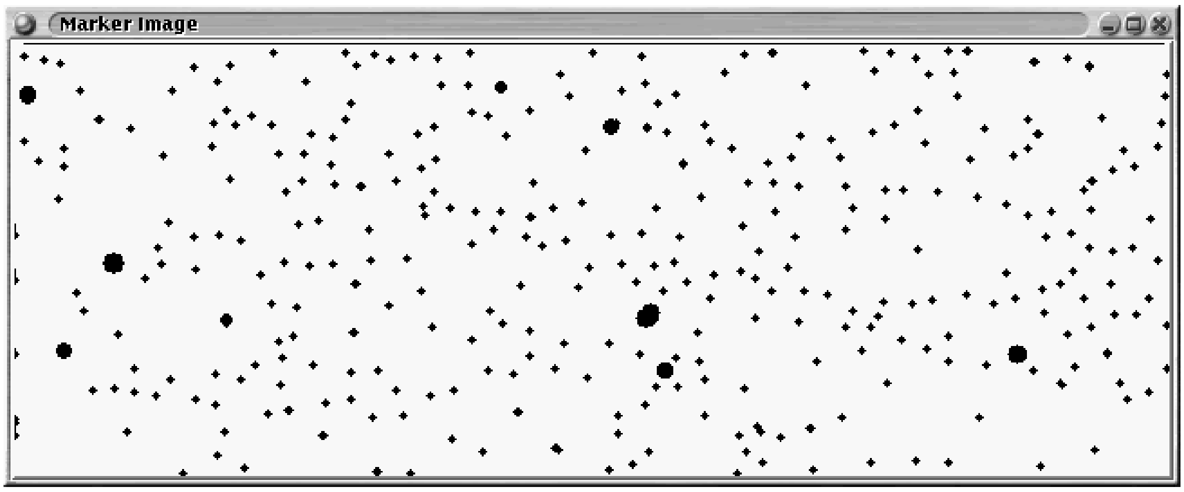


(c) Watershed segmented image.

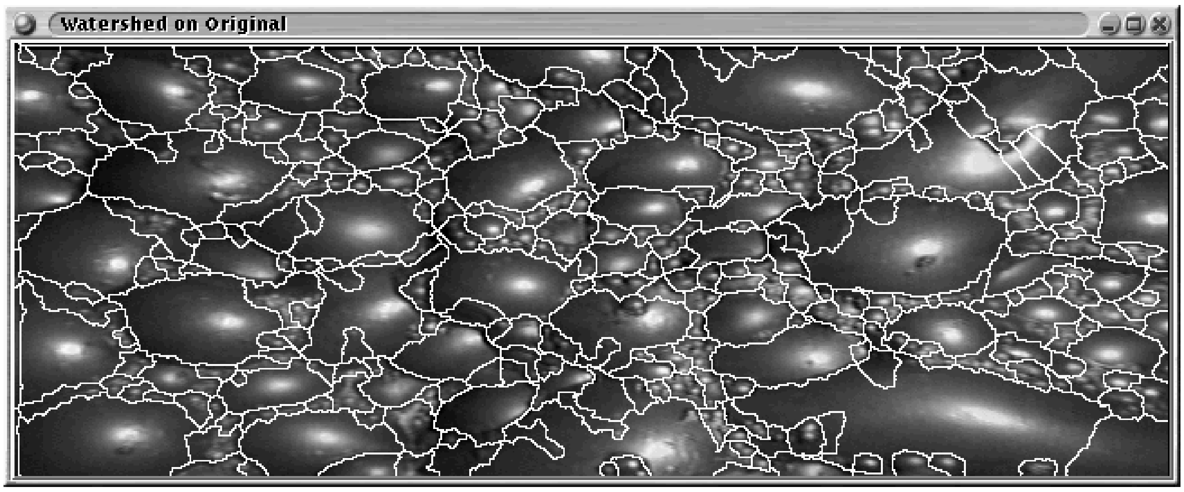
FIGURE 9.17 Result of rolling ball dilation (radius 1) on image, marker and watershed.



(a) Denoised image.

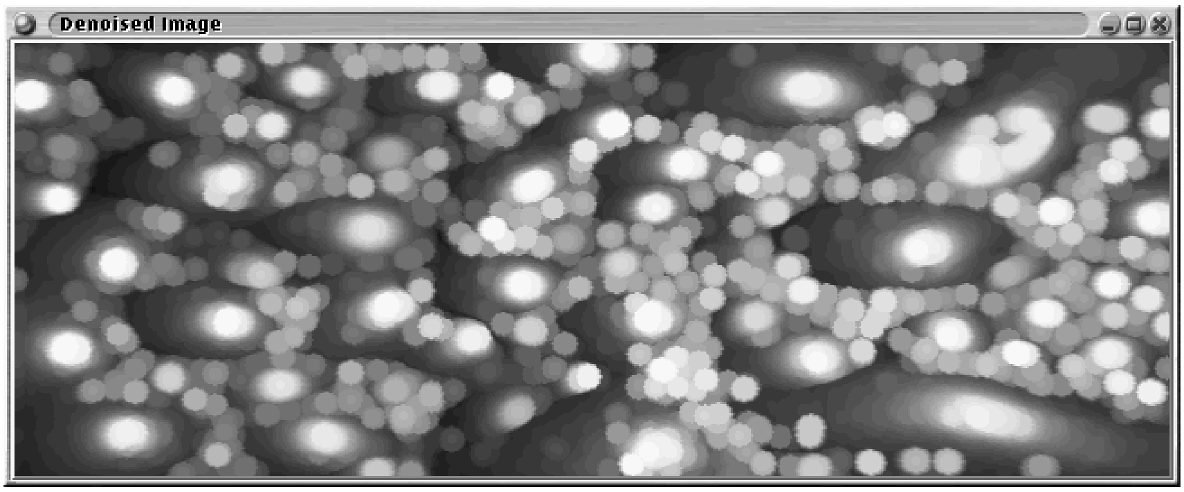


(b) Marker image.

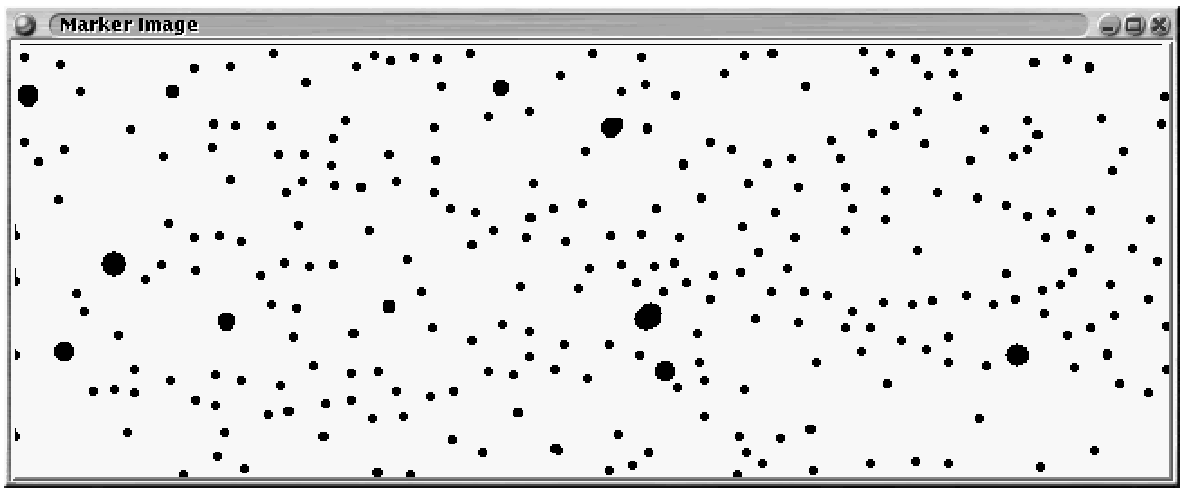


(c) Watershed segmented image.

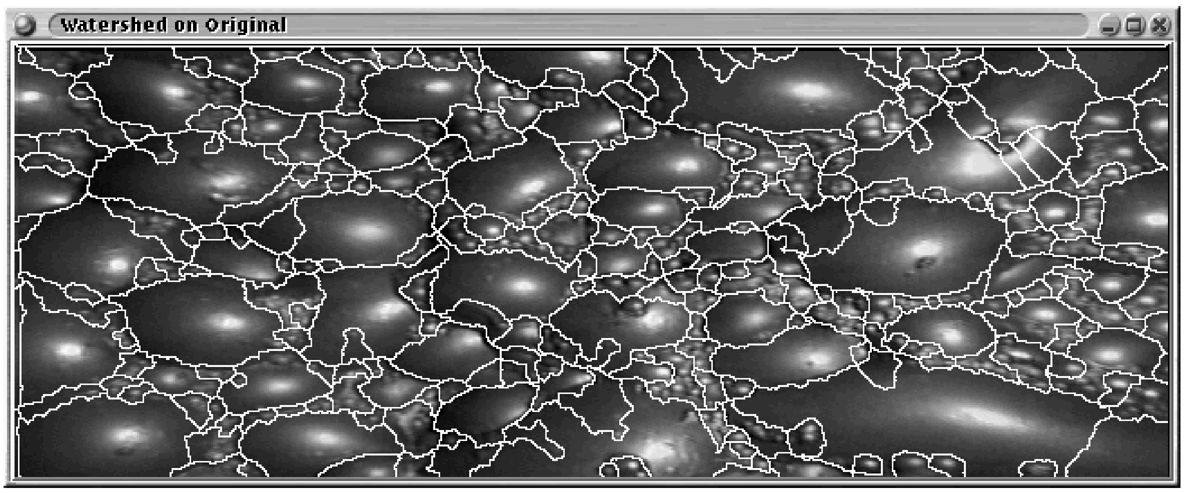
FIGURE 9.18 Result of rolling ball dilation (radius 5) on image, marker and watershed.



(a) Denoised image.

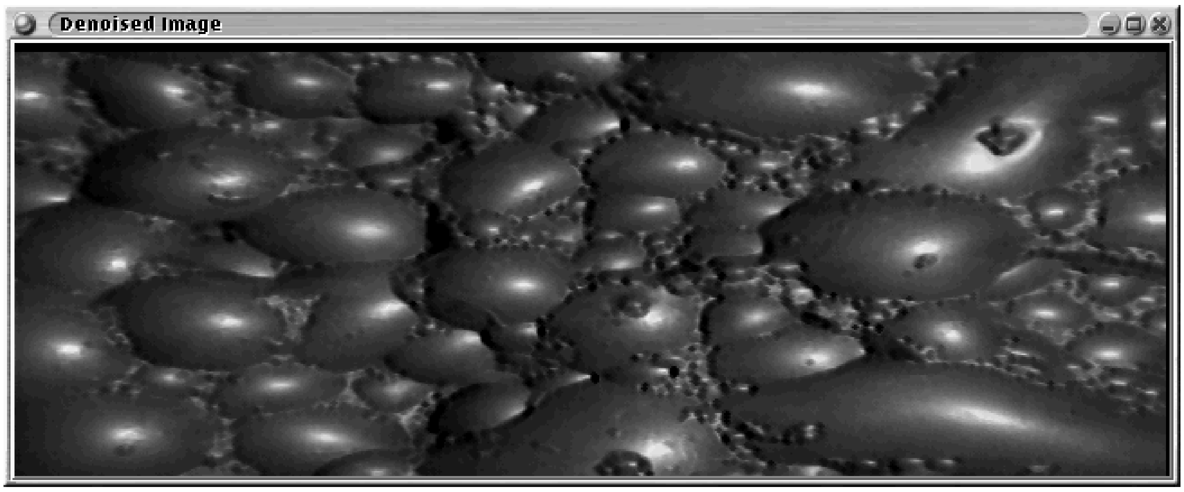


(b) Marker image.

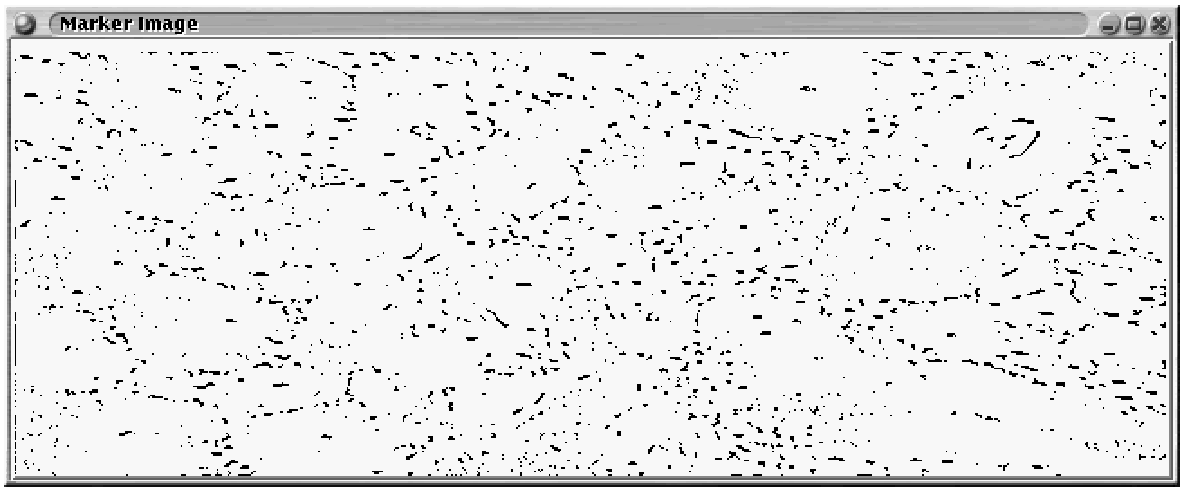


(c) Watershed segmented image.

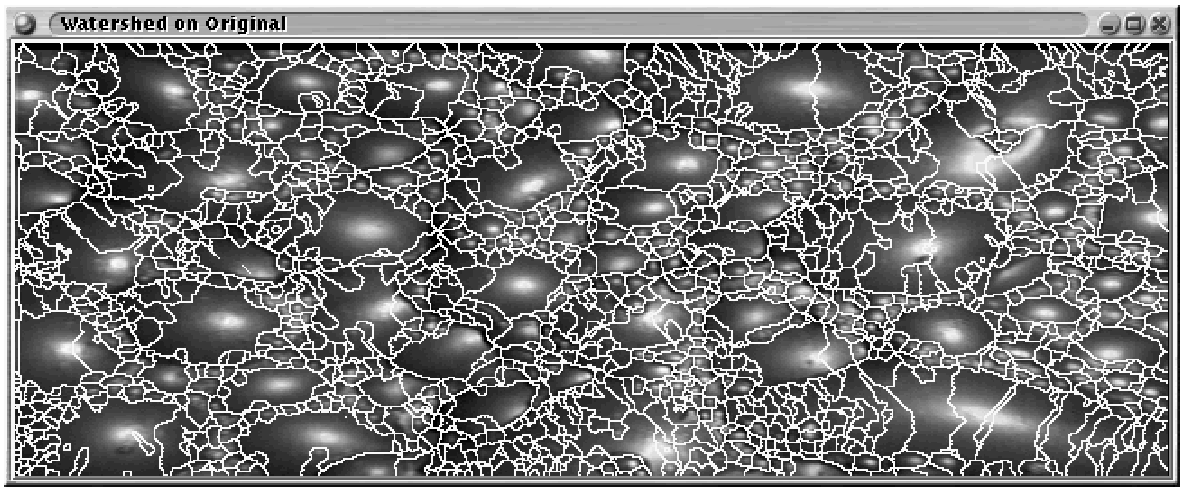
FIGURE 9.19 Result of rolling ball dilation (radius 6) on image, marker and watershed.



(a) Denoised image.

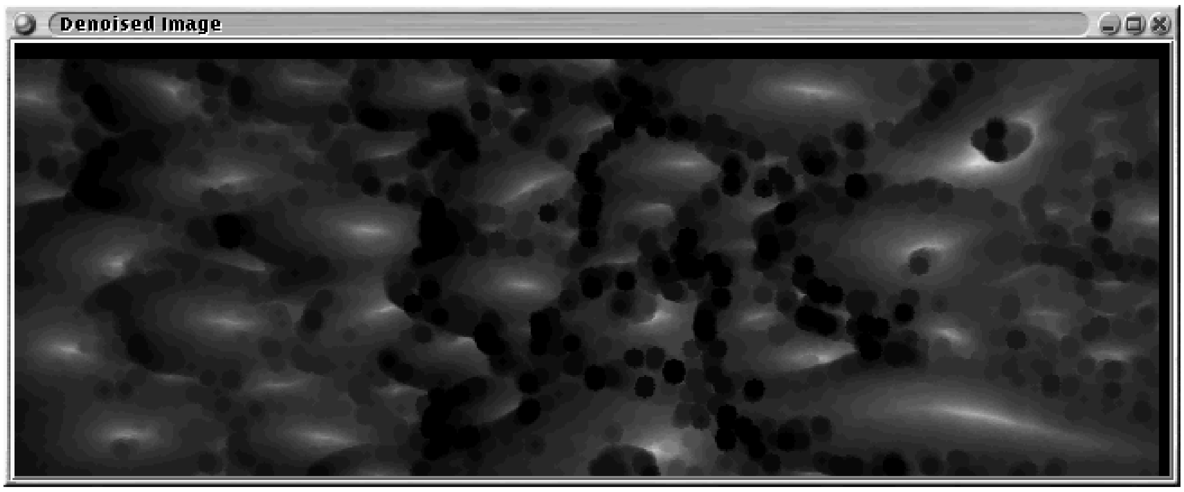


(b) Marker image.

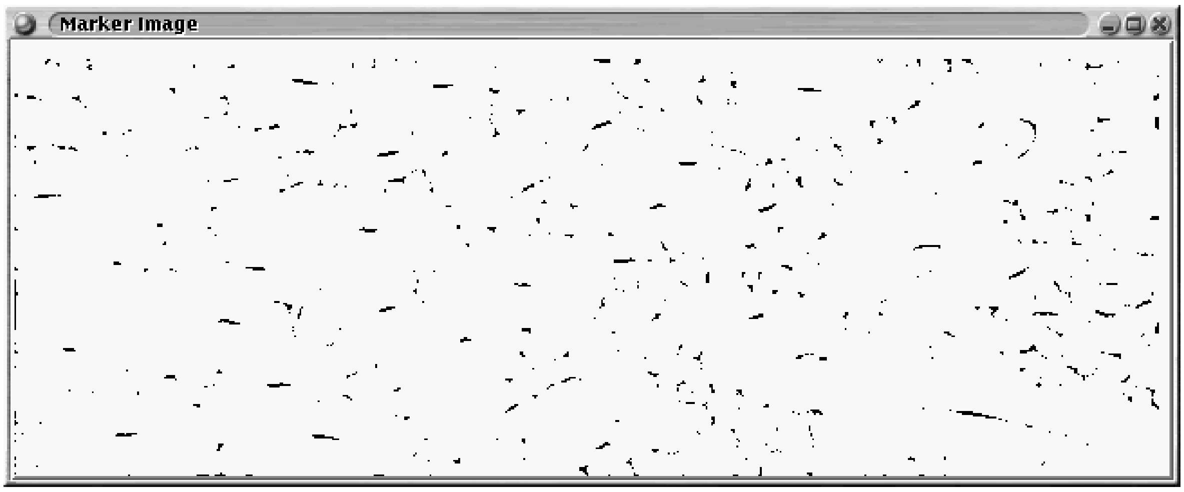


(c) Watershed segmented image.

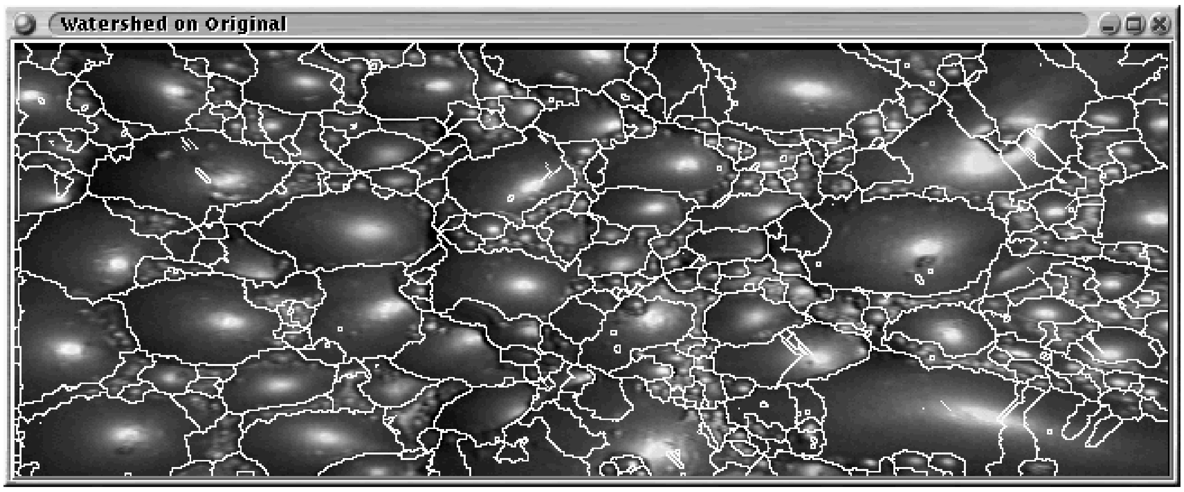
FIGURE 9.20 Result of rolling ball erosion (radius 1) on image, marker and watershed.



(a) Denoised image.

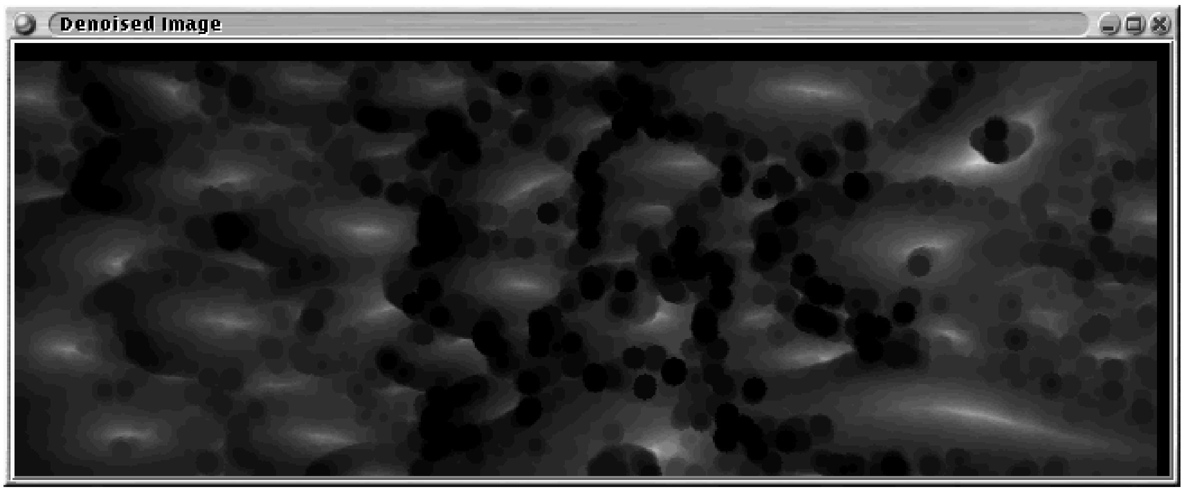


(b) Marker image.

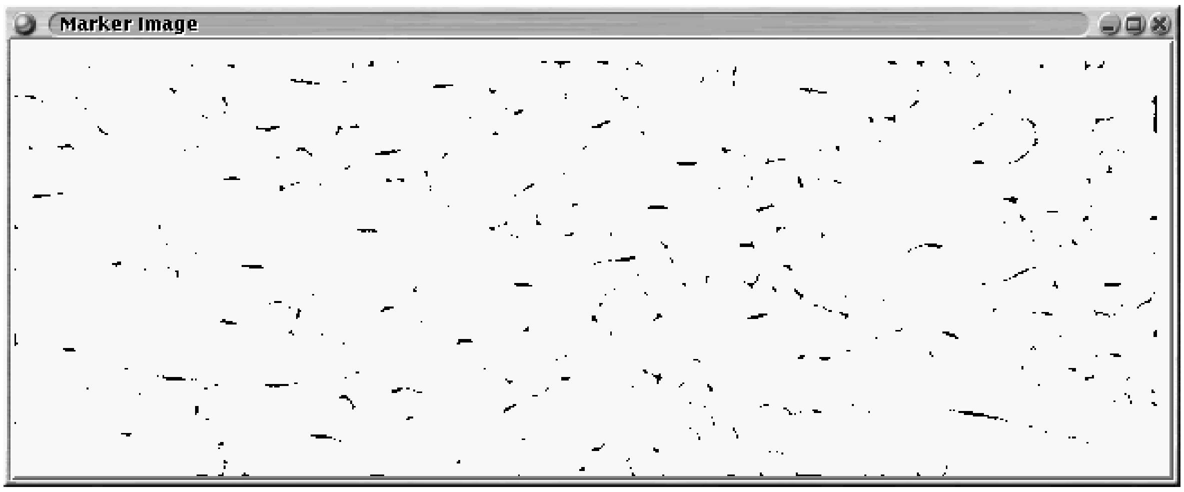


(c) Watershed segmented image.

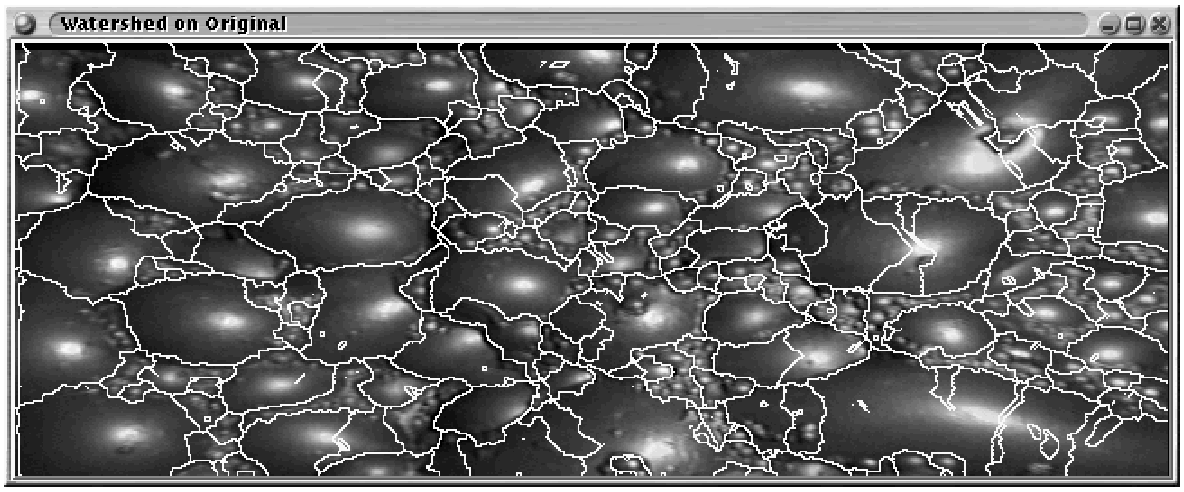
FIGURE 9.21 Result of rolling ball erosion (radius 5) on image, marker and watershed.



(a) Denoised image.

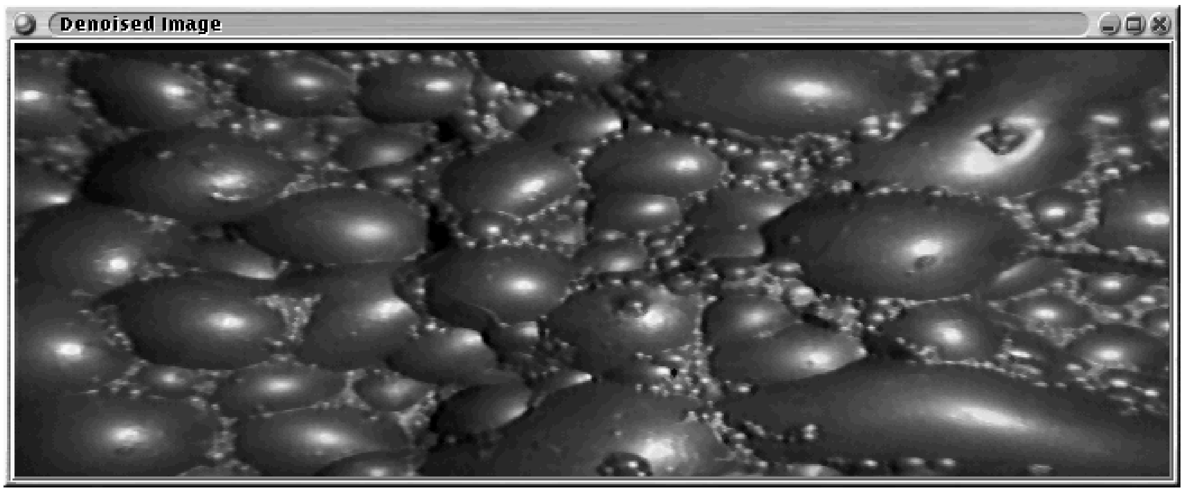


(b) Marker image.

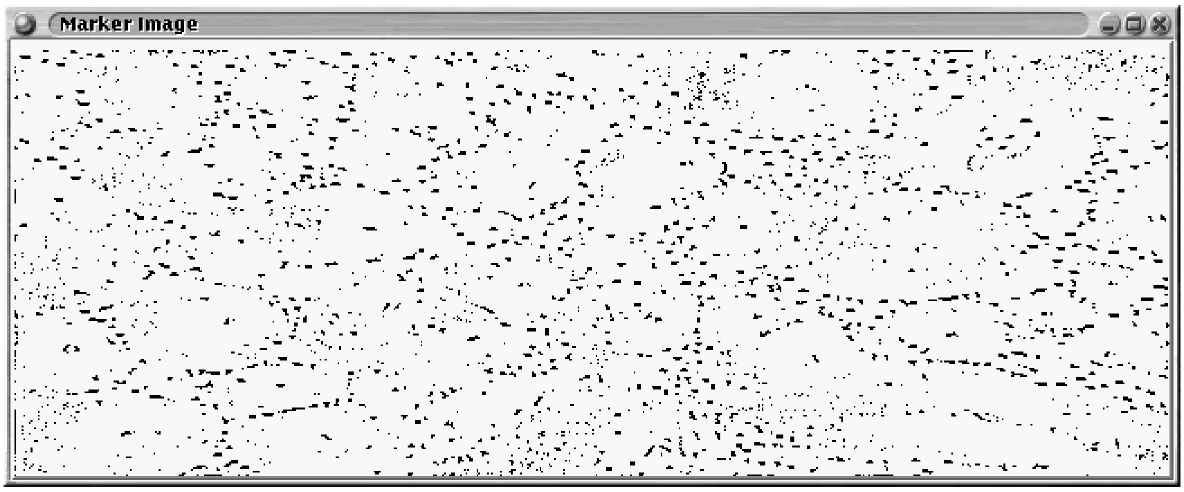


(c) Watershed segmented image.

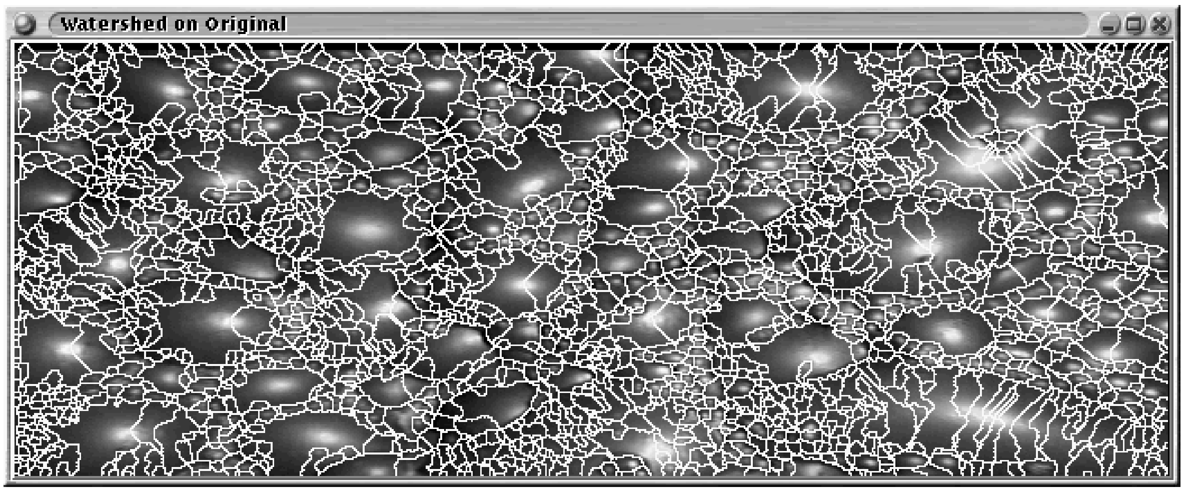
FIGURE 9.22 Result of rolling ball erosion (radius 6) on image, marker and watershed.



(a) Denoised image.

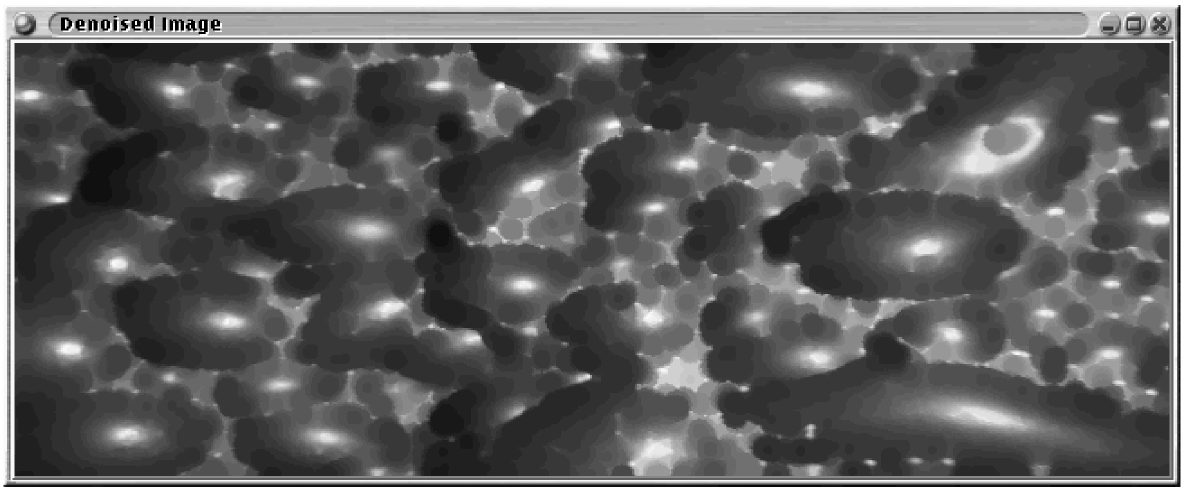


(b) Marker image.

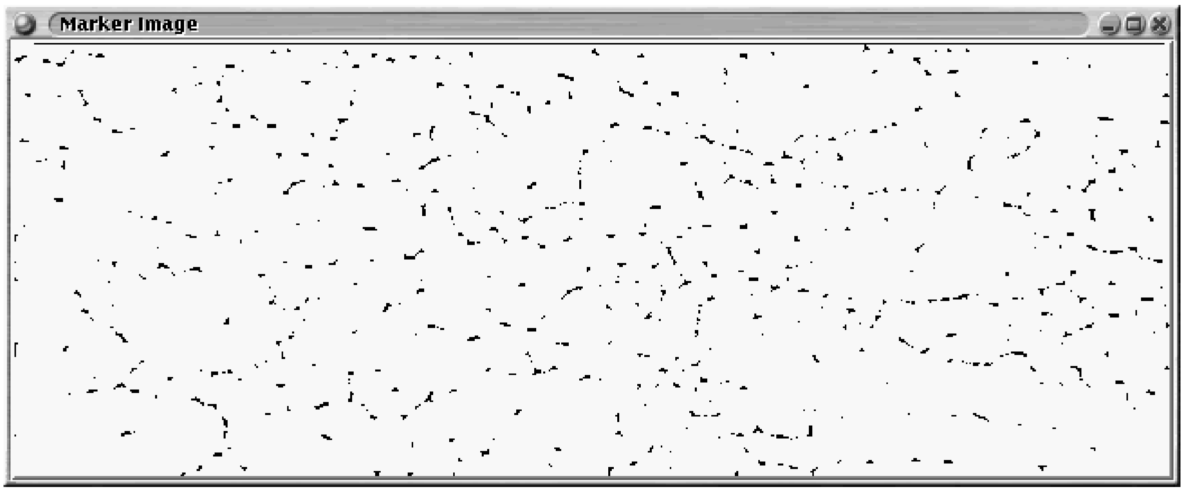


(c) Watershed segmented image.

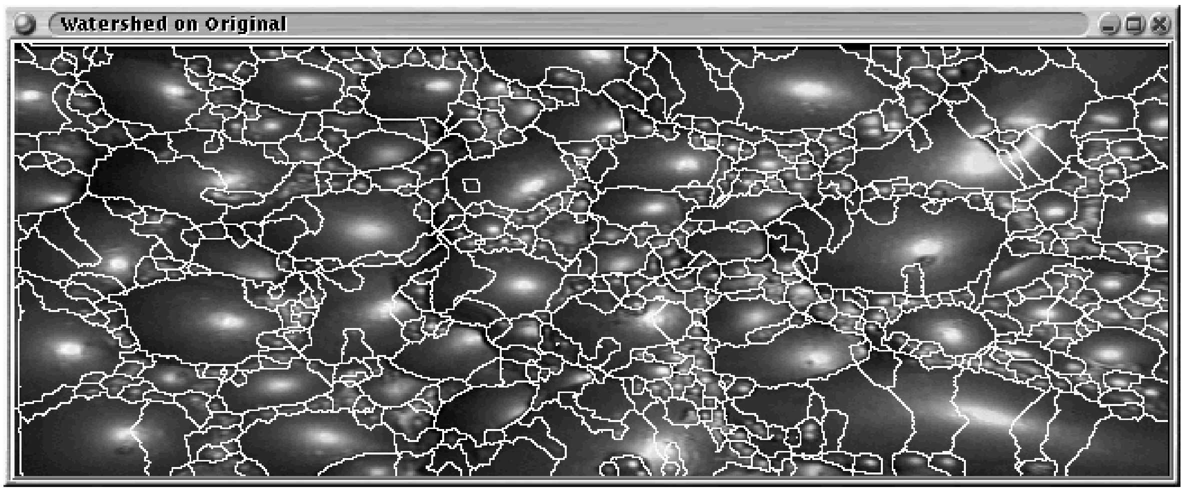
FIGURE 9.23 Result of rolling ball closing (radius 1) on image, marker and watershed.



(a) Denoised image.

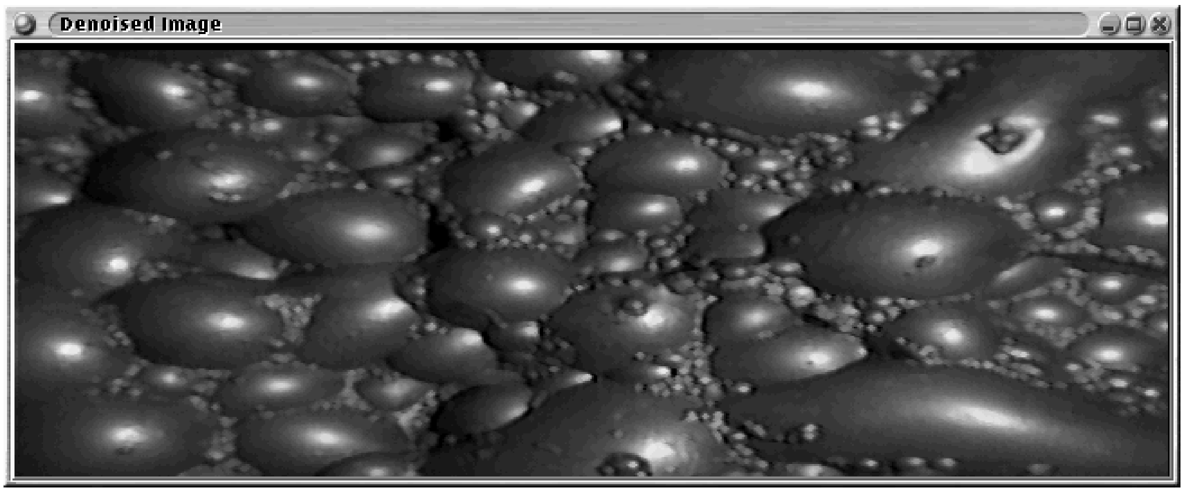


(b) Marker image.

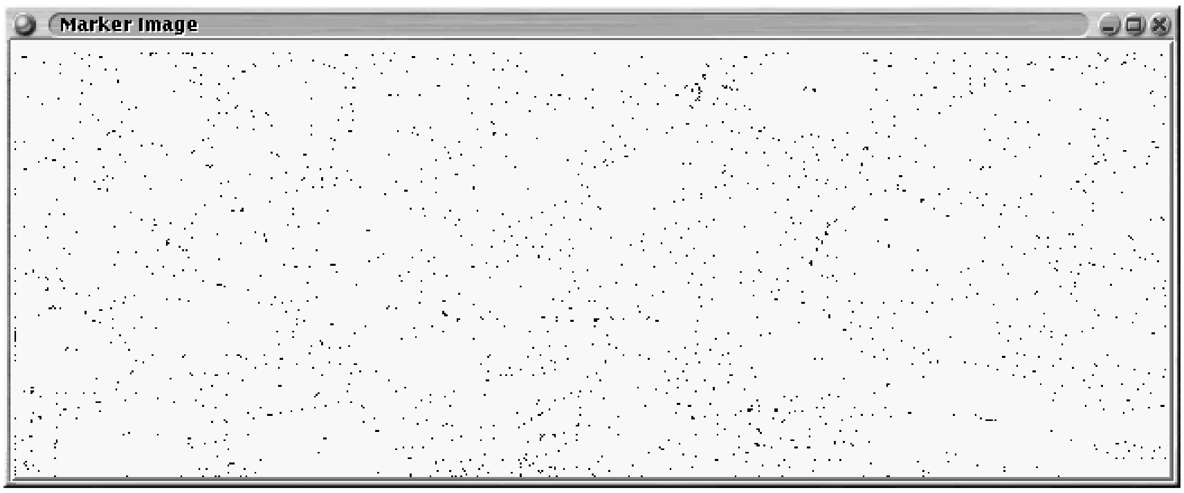


(c) Watershed segmented image.

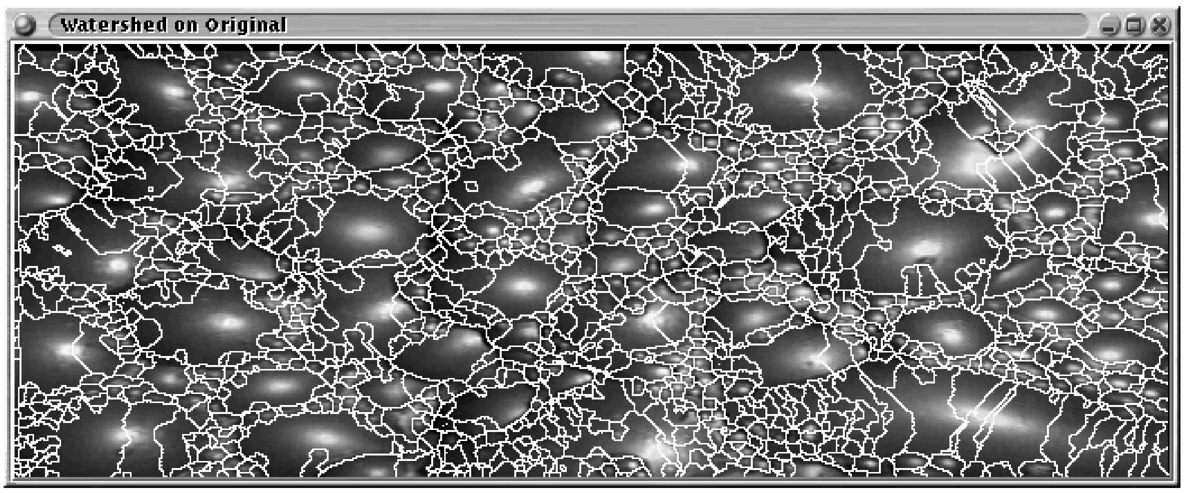
FIGURE 9.24 Result of rolling ball closing (radius 6) on image, marker and watershed.



(a) Denoised image.

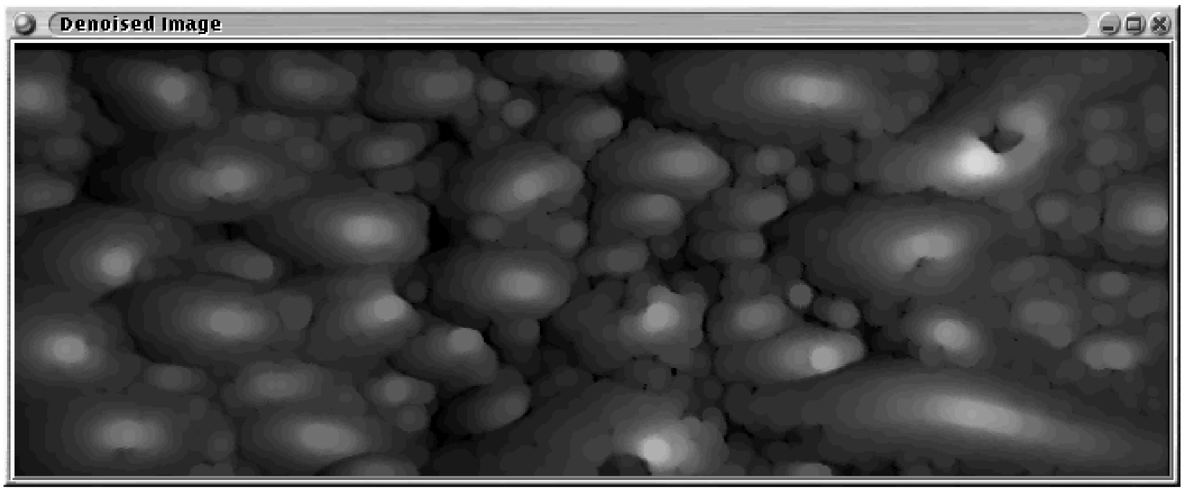


(b) Marker image.

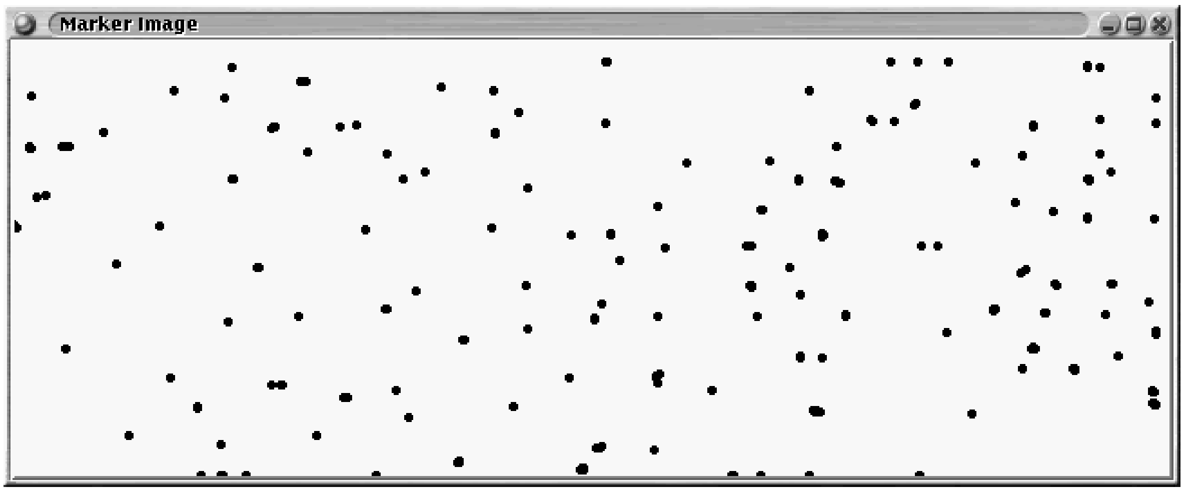


(c) Watershed segmented image.

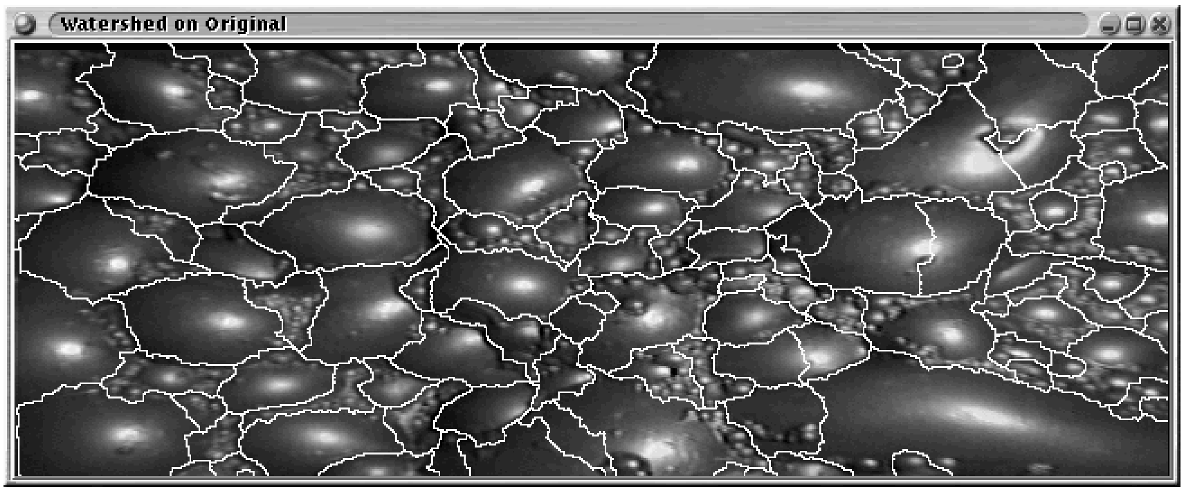
FIGURE 9.25 Result of rolling ball opening (radius 1) on image, marker and watershed.



(a) Denoised image.

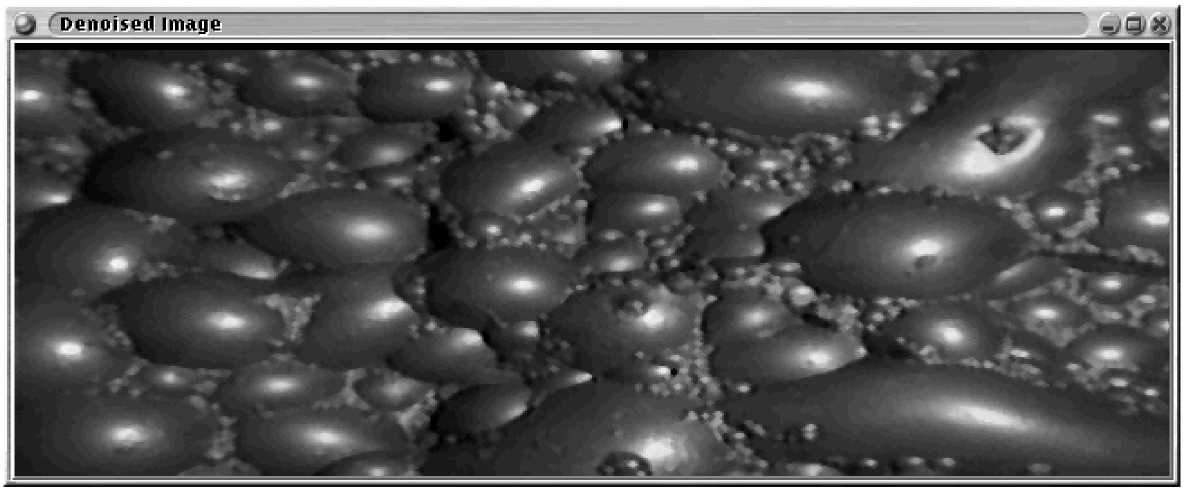


(b) Marker image.

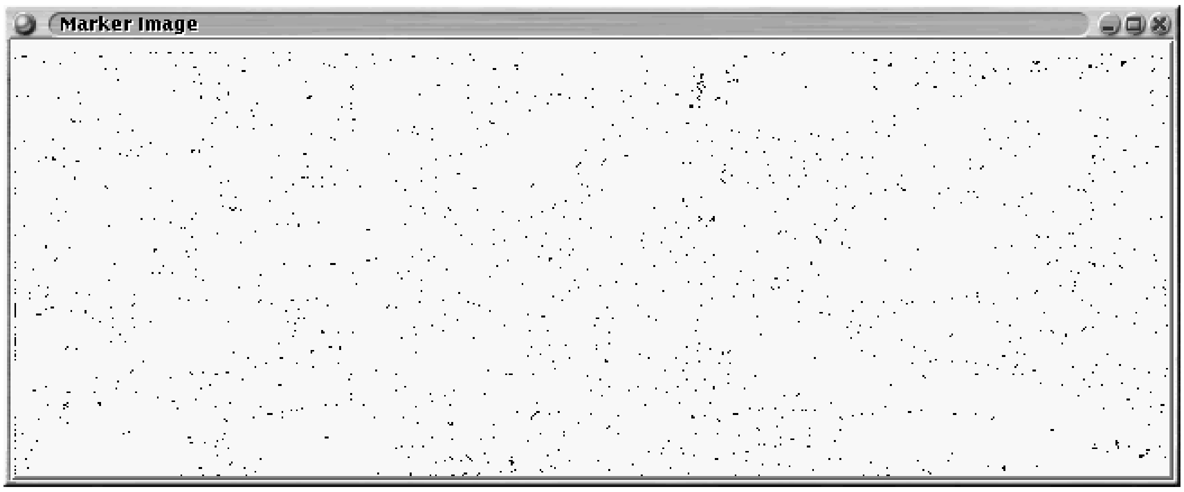


(c) Watershed segmented image.

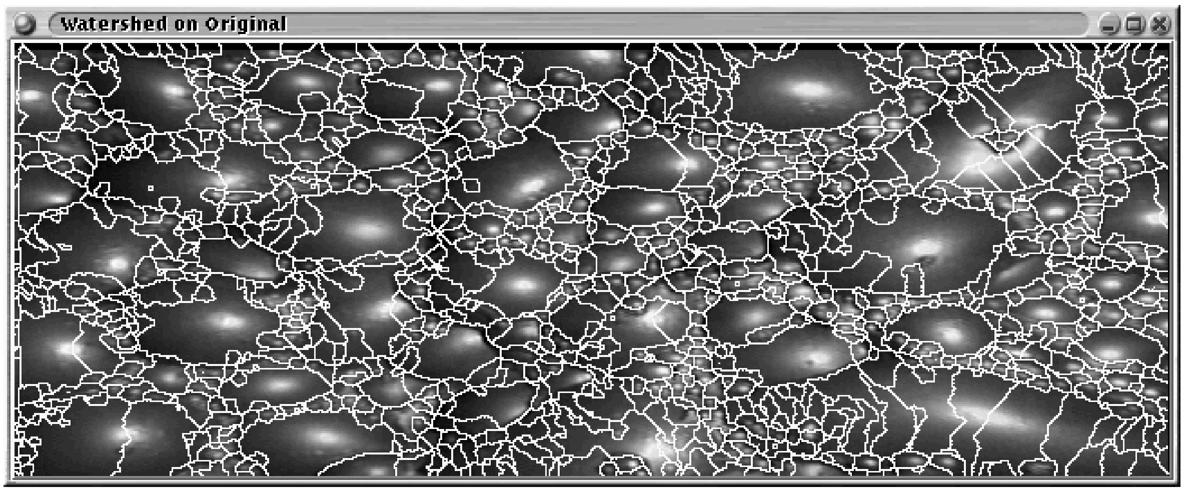
FIGURE 9.26 Result of rolling ball opening (radius 6) on image, marker and watershed.



(a) Denoised image.

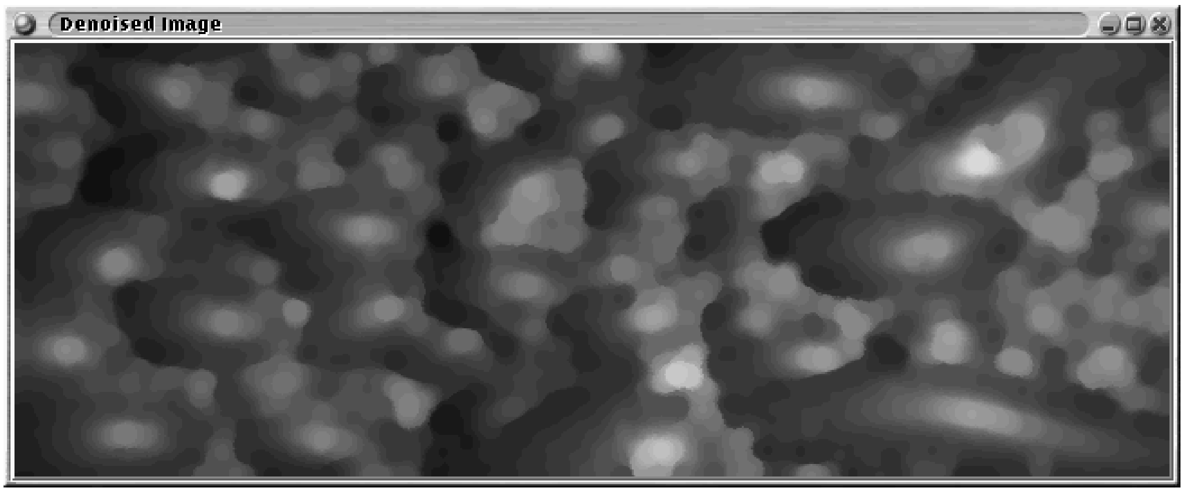


(b) Marker image.

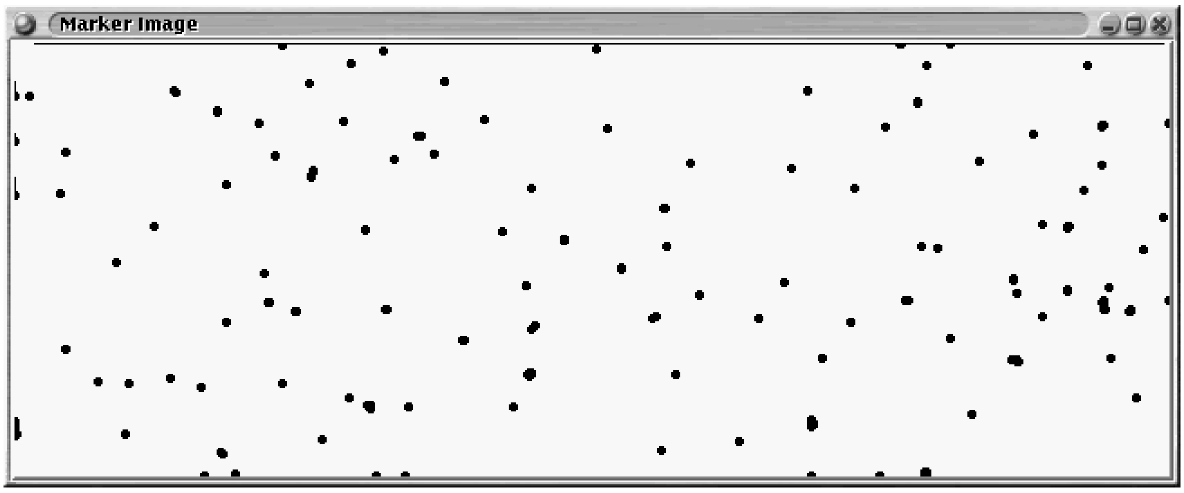


(c) Watershed segmented image.

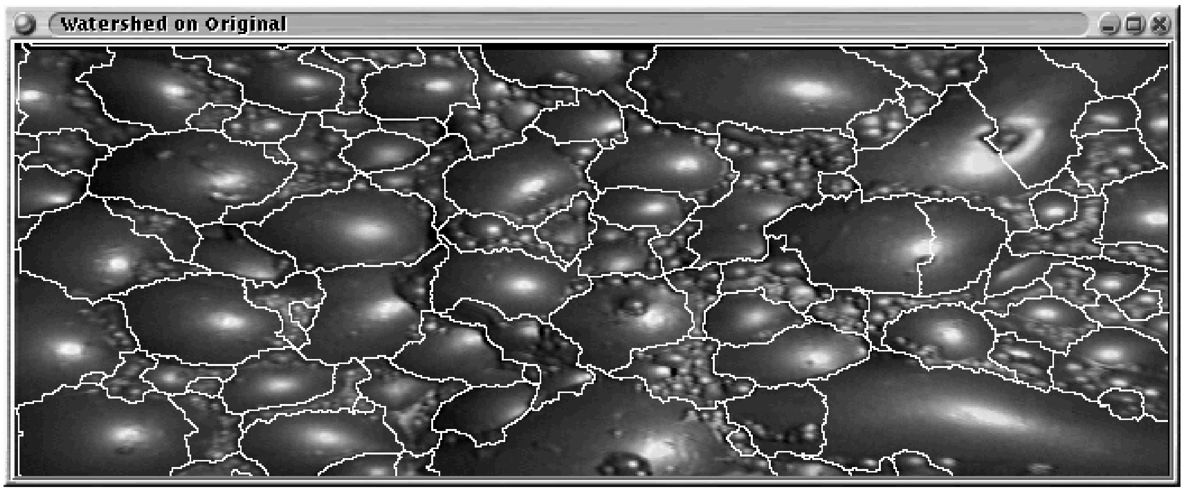
FIGURE 9.27 Result of rolling ball close-opening (radius 1) on image, marker and watershed.



(a) Denoised image.

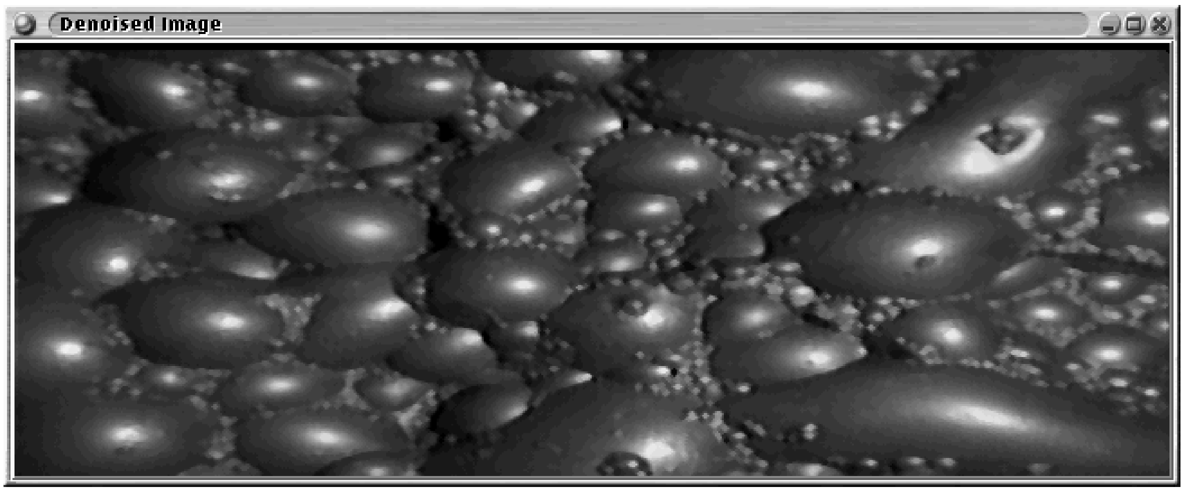


(b) Marker image.

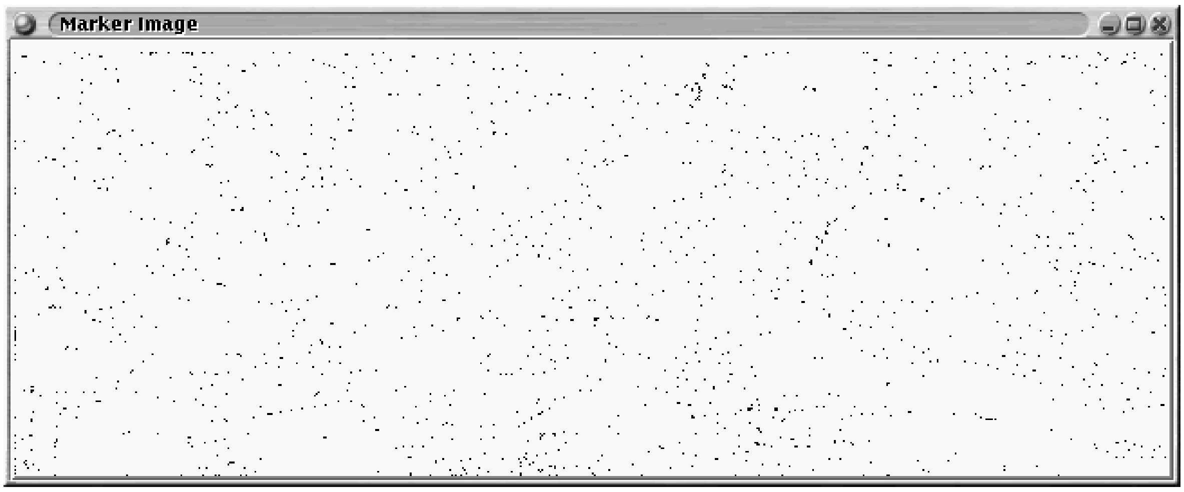


(c) Watershed segmented image.

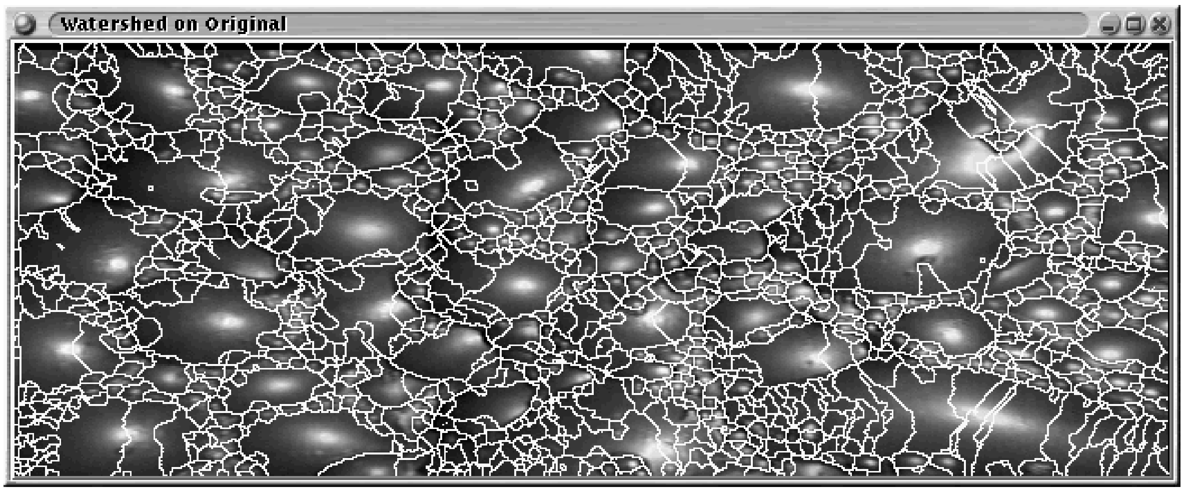
FIGURE 9.28 Result of rolling ball close-opening (radius 6) on image, marker and watershed.



(a) Denoised image.

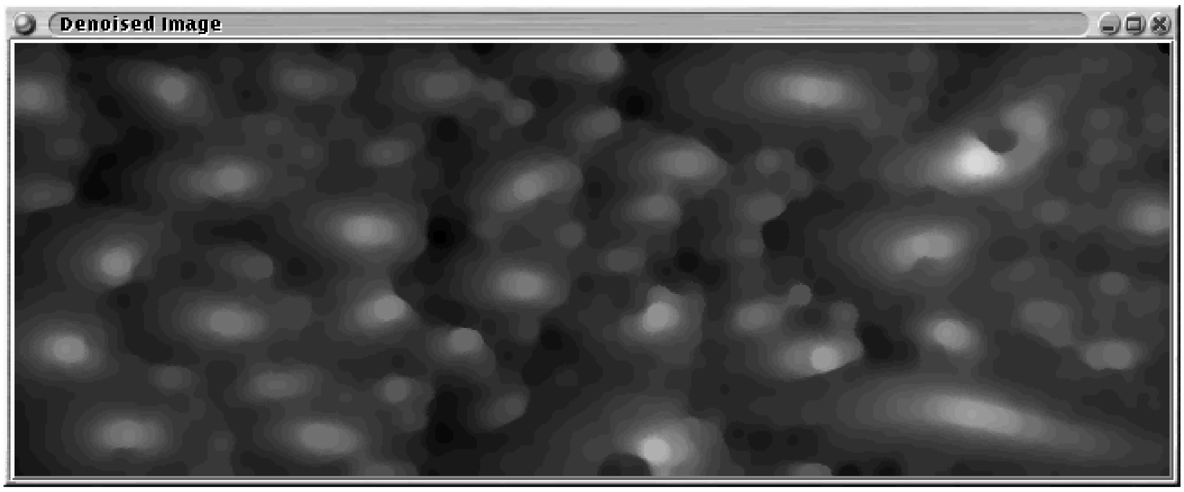


(b) Marker image.

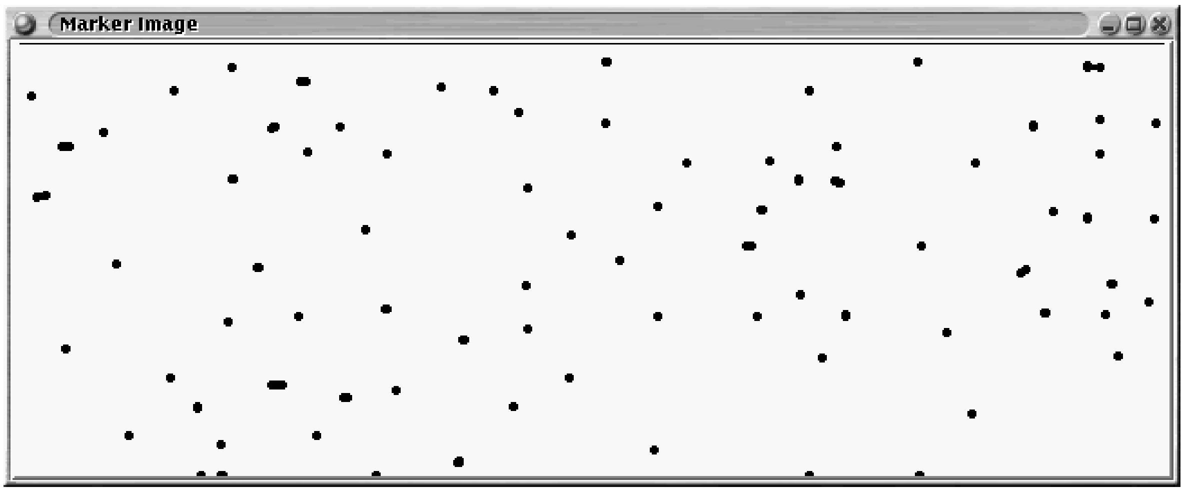


(c) Watershed segmented image.

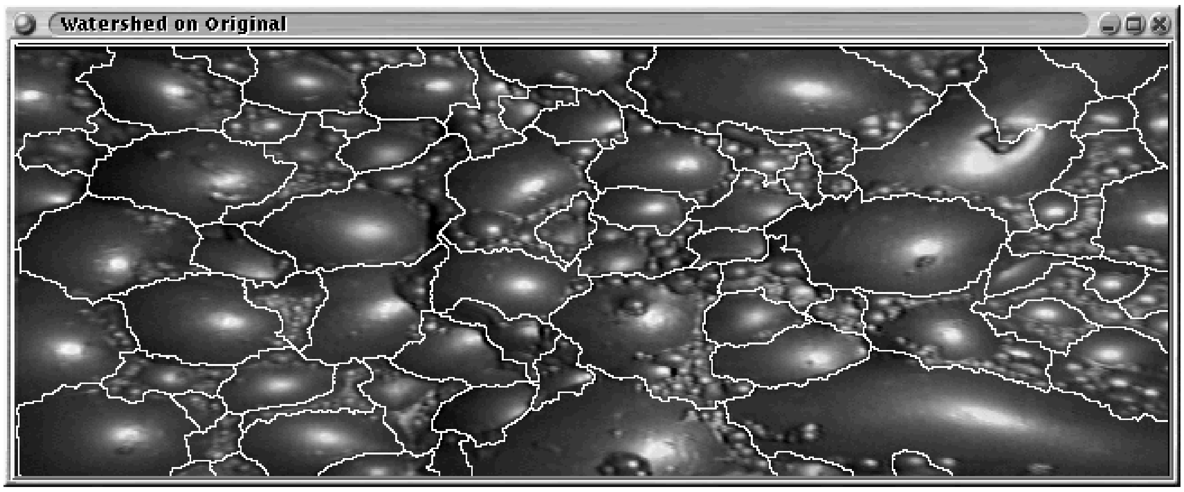
FIGURE 9.29 Result of rolling ball open-closing (radius 1) on image, marker and watershed.



(a) Denoised image.

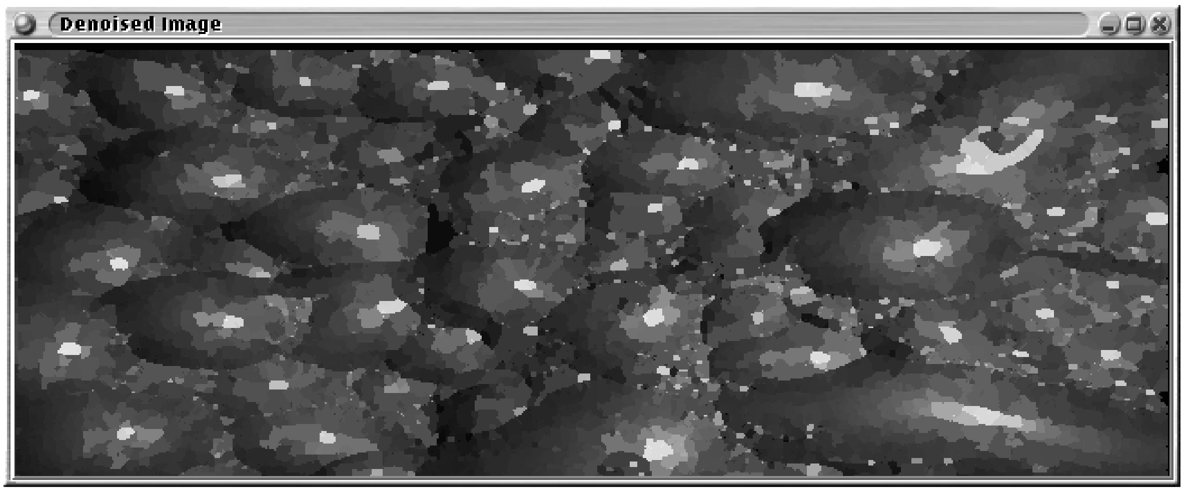


(b) Marker image.

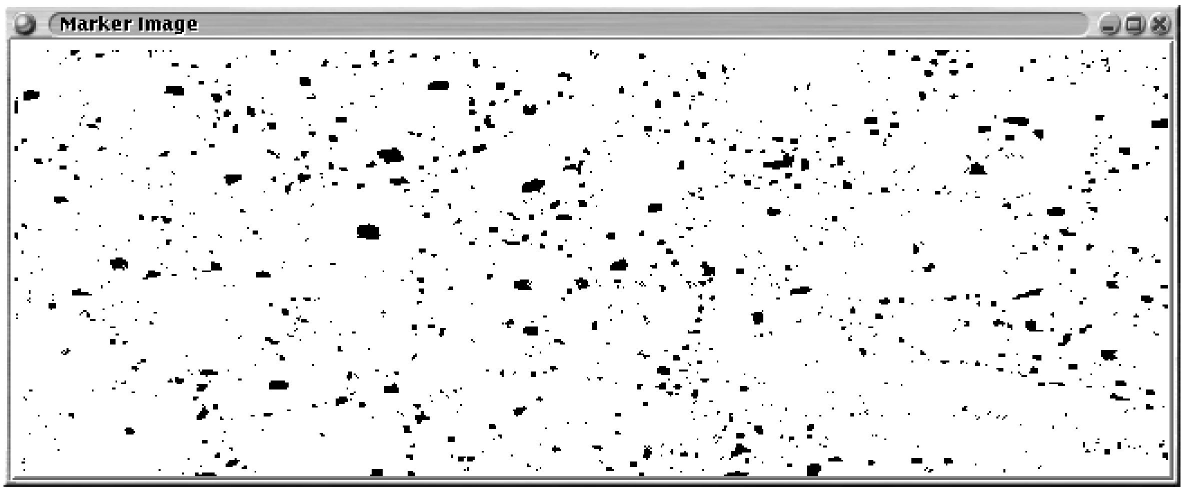


(c) Watershed segmented image.

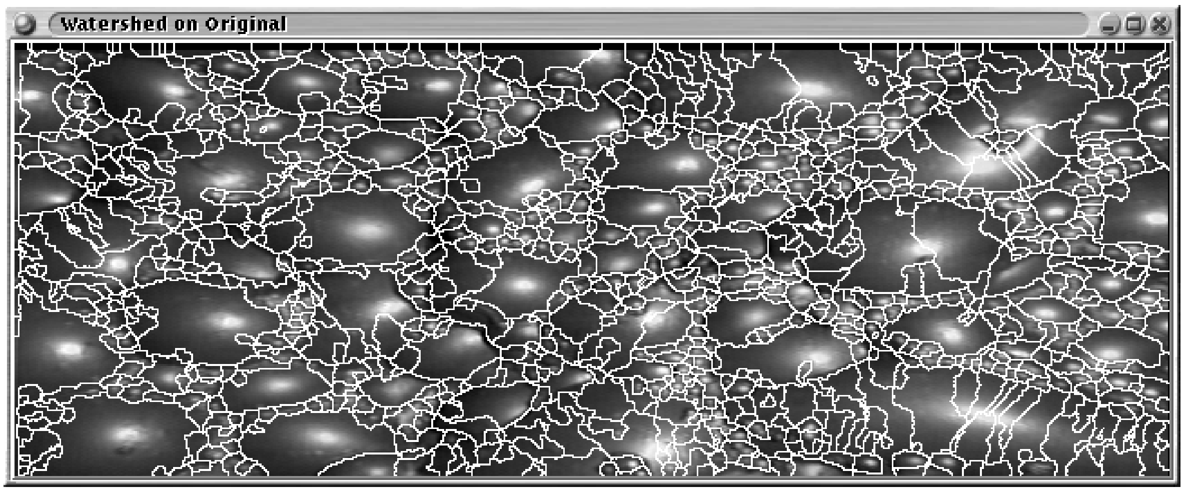
FIGURE 9.30 Result of rolling ball open-closing (radius 6) on image, marker and watershed.



(a) Denoised image.

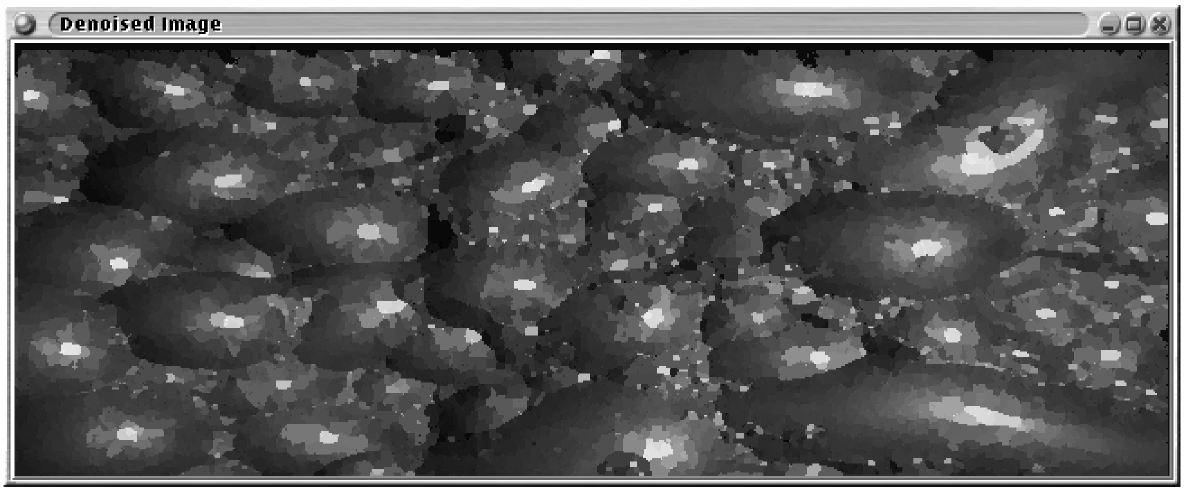


(b) Marker image.

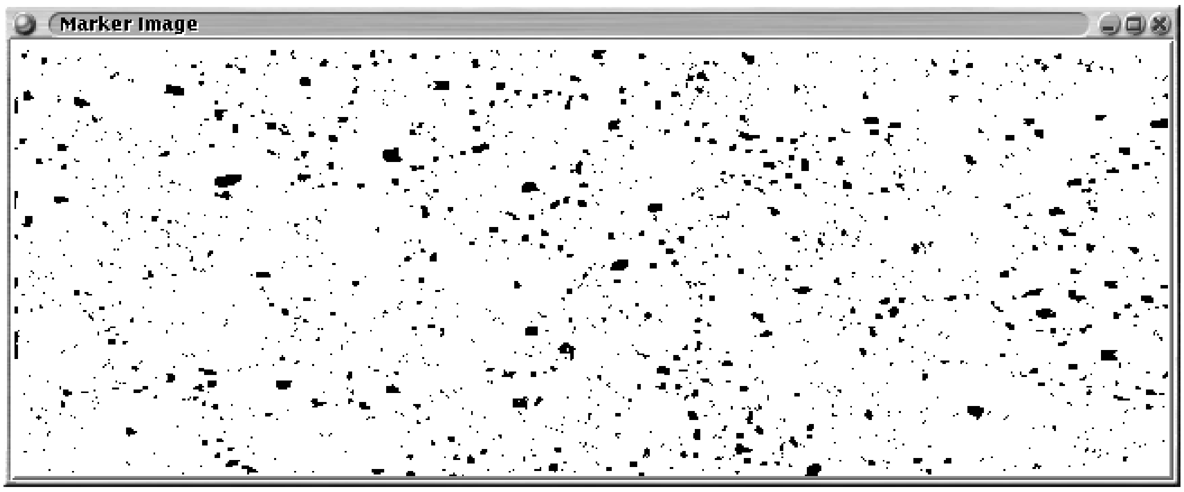


(c) Watershed segmented image.

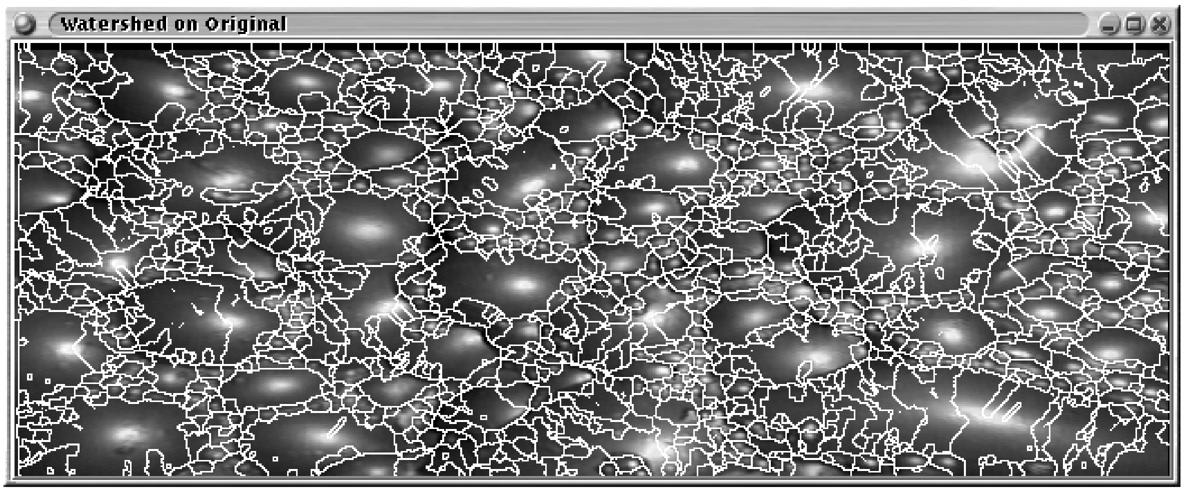
FIGURE 9.31 Result of posterisation on image, marker and watershed.



(a) Denoised image.



(b) Marker image.



(c) Watershed segmented image.

FIGURE 9.32 Result of posterisation on image, marker and watershed.

TABLE 9.3 Watershed regions per iteration for posterisation.

Iterations	# Watershed Regions
1	1071
2	18
3	2

9.3.1 Discussion

This algorithm relies on a certain amount of noise to yield perceptually useful images, or, failing enough noise, sufficient texture. The improvement in the visual appearance of the images as the noise level increases can be explained by a form of random sampling. As the noise level increases the watershed region of each noise minimum decreases (more noise maxima occur) and more watershed regions results. If the the noise amplitude is kept small, then the flat regions better approximate the underlying image. The need to add noise to improve the performance of a denoising filter makes posterisation of limited usefulness for denoising images. It is reasonable to expect a denoising algorithm should leave an image consisting of flat regions or steps largely unchanged, without adding noise this will not happen for this algorithm.

Posterisation does not denoise froth images well.

Chapter 10

Occam filters

All compression-quantisation-decompression filters are Occam filters [102]. Any filter that transforms data to some transform domain, for example generalised Fourier, Wavelet, Discrete Cosine Transform (DCT); and then discards some of the information in the transform domain can be regarded as an Occam Filter. Provided the information lost is small and confined to mostly low magnitude coefficients, the loss discards more noise than signal. The assumption made is that small magnitude coefficients are more likely to be noise induced or heavily affected by noise. Occam filtering is based on the observation that noise, being relatively uncorrelated, is more difficult to compress than signal which tends to be correlated. The implication is that slight lossy compression may *improve* the Signal to Noise Ratio (SNR) of the image. This high local correlation is prevalent in froth images especially, which makes Occam filtering attractive.

This chapter examines some attempts at compression based denoising, including Set Partitioning in Hierarchical Trees (SPIHT), JPEG based compression and various attempts at wavelet based denoising. The common properties of each are examined.

In the experimental results provided, the images are not preprocessed except to be histogram stretched to occupy the full intensity range possible (0 to 255). The reconstructed image is created by rolling ball erosion (radius 1) of the inverted denoised image, followed by greyscale reconstruction by dilation of the eroded image under the original. The domes are extracted by writing to the marker image only those pixels which are different from their reconstructed counterparts. The domes become the marker image.

The depth of the multi-resolution decomposition for wavelet based denoising is 4. At the this depth, the size of the processed images become 40×15 (from the original 640×240. Since one of the dimensions is no longer even, the multi-resolution analysis cannot continue without zero padding the image.

10.1 Common operations applied during lossy compression

Some techniques for achieving the losses in lossy compression is examined.

Some methods of lossy compression in wide spread use include:

HARD THRESHOLDING All coefficients below a certain threshold set to zero [32, 33].

SOFT THRESHOLDING Coefficients below a certain threshold are remapped [32, 33].

PERCENTAGE (QUANTILE) THRESHOLDING A percentage of smallest coefficients are set to zero [32, 33].

A variant of percentage thresholding is Lorentz wavelet filtering (as proposed by Goel and Vidakovic) [37]. Generalised, the process ranks the transform coefficients in some order (for example, increasing magnitude or frequency). The sum of all coefficients less than or equal to the current coefficient is computed. The percentage of coefficients to remove is computed by counting the number of coefficients less than or equal to the mean rank used (for example, magnitude) as a proportion of the total number of coefficients.

QUANTISATION AND INVERSE QUANTISATION The coefficients are quantised by division and discarding the fractional part. The coefficients are then reconstructed by multiplying. Selecting the divisors is determined by the compression required and perceptual errors in the reconstruction allowed. This is used by JPEG and Motion Pictures Expert Group (MPEG), for example [70, 145].

SPIHT uses a variant of this quantisation de-quantisation process, all coefficients are quantised and de-quantised with power of two divisors until the error budget is exceeded [121]. The iterative quantisation process essentially strips off bits from least to most significant until a target error is reached.

10.2 Denoising via Wavelets

This section addresses the possibilities of wavelet based denoising. An overview chapter on wavelets may be found in Appendix A.

Wavelet based denoising makes the assumption that noise affects small detail coefficients more than others [32]. The basic approach is then, transform the image into the wavelet domain using some wavelet, apply a function to the detail coefficients, finally inverse transform the filtered coefficients to yield the denoised image.

Table 10.1 shows the energy compaction properties of some wavelets after transforming the standard test image (see figure 5.4). Energy compaction is a measure of the proportion of coefficients with significant energy, in contrast to the untransformed image [139][pg. 372–376]. The Haar and Daubechies maximally flat wavelets are used [139][pg. 260]. The energy in 99% of the wavelet coefficients are shown with a view to discarding these coefficients. The wavelet with the most energy in the remaining coefficients will have better reconstruction than the others. The coefficients include the low-pass coefficients (scaling coefficients) at the lowest scale. The larger the number, the more energy is contained in that proportion of coefficients, leaving less energy in the high value coefficients. The energies are scaled to unity by dividing by the total energy (the sum of the energy of all coefficients in the Multi Resolution Analysis). It should be noted that the balance of the energy lies in the largest (in magnitude) 1% of the coefficients, and is considerable. The wavelet with the most energy in the upper 1% of the largest magnitude coefficients is the Daubechies N3. All other considerations being equal, the wavelet with the most energy in the larger coefficients is desirable. For example, image compression requires only that the coefficients with large magnitudes need to be transmitted with great fidelity, the others may be more coarsely quantised (if they are transmitted at all). This is a consequence of the approximation properties of the wavelet under consideration. For example, a piecewise constant signal in the presence of noise, requires a wavelet with $p = 1$ regularity (a piece-wise constant wavelet), higher orders of regularity will not approximate the signal better and run the risk of attempting to fit the noise as well. This is similar to the argument used to justify selecting as low an order of model as possible for fitting a polynomial model to data with noise. The Daubechies N3 wavelet is then selected to test the different wavelet denoising schemes. The higher order wavelets in the Daubechies family show similar energy compaction, The energy compaction of the Daubechies N3 indicates it a suitable wavelet for compressing froth images. (A similar argument is used to justify the selection of a particular transform for transform domain lossy coding [50][pg 375–379].)

TABLE 10.1 Energy compaction of some wavelets.

	Wavelet energy compaction (as % of total energy).					
% Coefficients	Haar	Daubechies Maximally Flat				
		N2	N3	N4	N5	N6
99	0.5164	0.3909	0.3857	0.3917	0.3936	0.3900

There are several possibilities for wavelet based denoising:

(i) One can presuppose that small detail coefficients are likely to be very noisy. This is reasonable if one presupposes additive noise. One proceeds then by scaling these coefficients down – possibly to zero. (Hard thresholding.)

(ii) Soft thresholding (Donoho [33]). Soft thresholding involves hard thresholding for small magnitude coefficients and a linear remapping for larger coefficients.

(iii) Percentage thresholding. A percentage of the total coefficients is set to zero. The smallest, in magnitude, coefficients are affected first [51][pg. 216–218].

(iv) Lorentz thresholding. A variant of percentage thresholding. The percentage threshold is selected automatically using a Lorentz curve [49].

In order to justify the assumption that small detail coefficients are likely to be heavily influenced by noise, consider figure 10.1. This is an error image formed by subtracting a hard thresholded wavelet

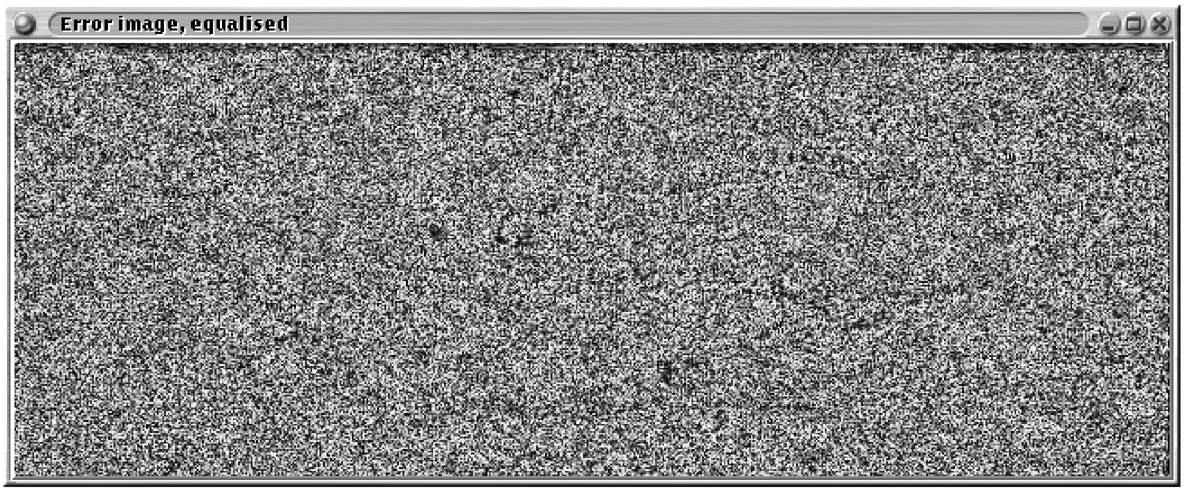


FIGURE 10.1 An error image justifying wavelet based denoising.

denoised image from the original. The hard threshold used is 5. The threshold is chosen to minimise over-segmentation while still segmenting the small bubbles correctly. The magnitude of the error is shown and the image is histogram equalised to make the errors more visible. Note the rough texture of the image. The image shows little correlation between neighbouring pixels.

10.2.1 Hard thresholding

Each of the coefficients of the multi-resolution analysis less than some threshold (possibly a spatially dependent threshold) is set to zero or reduced by some scale factor. Historically, the process of zeroing a detail coefficient is referred to as hard thresholding [3, 33]. The process has appeared elsewhere in other transform domains, for example, generalised Fourier, Karhunen-Loéve, Discrete Cosine and Discrete Sine [145, 111]. For example, in the Karhunen-Loéve transform the basis vectors/functions to be suppressed would be those with corresponding small eigen-values.

The denoised image is thresholded to ensure that the pixel intensities lies within those of the original image. This minimises the effect of excessively large or small pixels. For example, eight bit images cannot represent negative intensities or intensities larger than 255. These pixels would otherwise show up as artifacts on the denoised image, even though the associated error at the pixel in question may be no larger than for any other pixel.

Figure 10.2 shows the denoised, error, marker and watershed segmented images. The error image is the difference between the original and the denoised image. To make the changes more visible, the error image has been histogram stretched and inverted (see section 11.1 for an introduction to histogram manipulation for contrast enhancement). The hard threshold selected is 30. The threshold is selected to provide good denoising without distorting the denoised image significantly. Note that the markers become blocky. The error image shows that most of the information lost lies in the textured regions where small bubbles lie between large bubbles. The marker image is generated from the denoised image. The watershed segmentation shows over-segmentation of the large bubbles, but good segmentation of the small bubbles lying between large bubbles. This is likely due to fragmented markers and small point-like spurious markers visible on the marker image. The denoised image shows no obvious artifacts.

Table 10.2 shows the decrease in the number of watershed regions as the hard threshold increases. This decrease may be explained by the removal of small, and noise, maxima. The maxima in the image form the domes of the image which become the markers. It is clear that hard thresholding is not sufficient to denoise a froth image sufficiently to allow for a good segmentation. Clearly processing of the markers and/or watershed segmented image is required.

TABLE 10.2 Decrease in the number of watershed regions as the hard threshold increases.

Hard threshold	# Watershed regions
5	2041
10	1827
20	1671
30	1542
40	1448
80	1096

10.2.2 Soft thresholding

This was described by Donoho [33]. The application was to wavelets although there seems no reason why this cannot be applied to other transform domains.

The process is as follows (generalised from [33]):

(i) Convert the data to some transform domain (wavelet, etc.).

(ii) Modify the coefficients as follows:

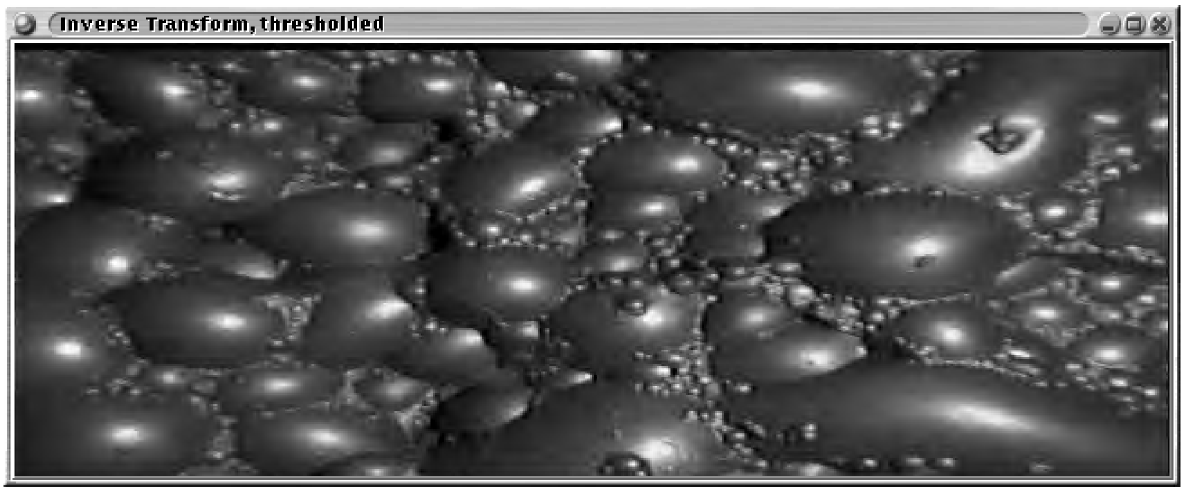
$$\eta_t(y) = \text{sign}(y)(\mid y \mid -t)_+ \tag{10.1}$$

Note that $\eta_t(y)$ represents the new coefficients, y the old, t the threshold, n the number of data points, and:

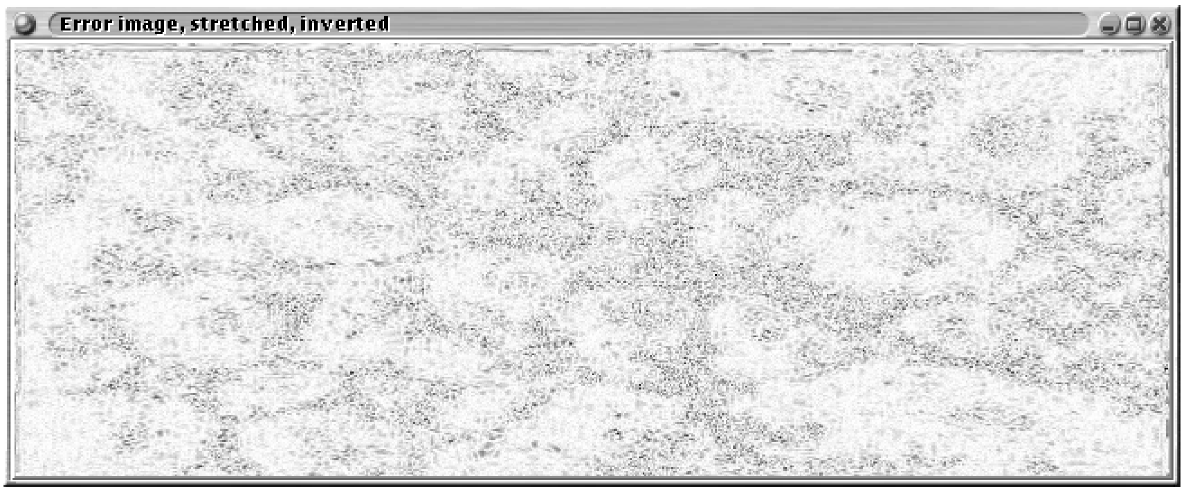
$$(\mid y \mid -t)_+ = \begin{cases} (\mid y \mid -t) & , (\mid y \mid -t) > 0 \\ 0 & , (\mid y \mid -t) \leq 0. \end{cases} \tag{10.2}$$

The threshold is computed as follows:

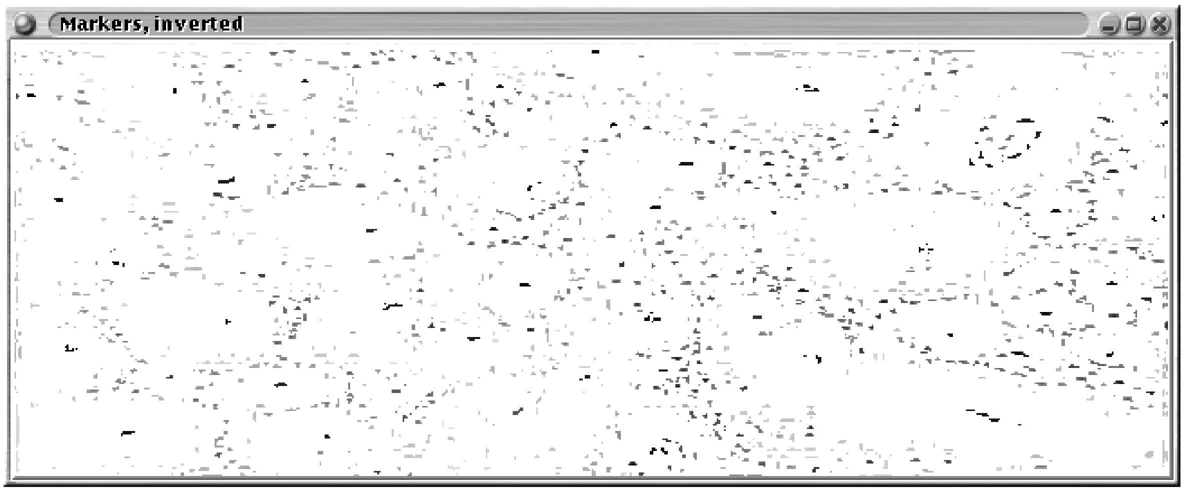
$$t_n = \sqrt{2\log(n)} \cdot \gamma_1 \cdot \frac{\sigma}{\sqrt{n}} \tag{10.3}$$



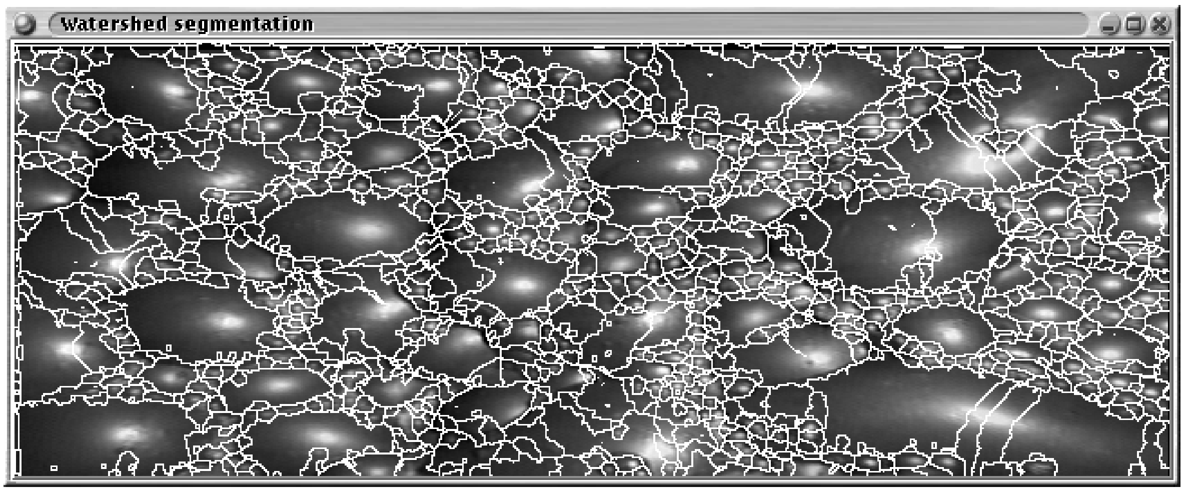
(a) Denoised image.



(b) Error image.



(c) Marker image.



(d) Watershed segmentation.

FIGURE 10.2 Wavelet denoising via hard thresholding.

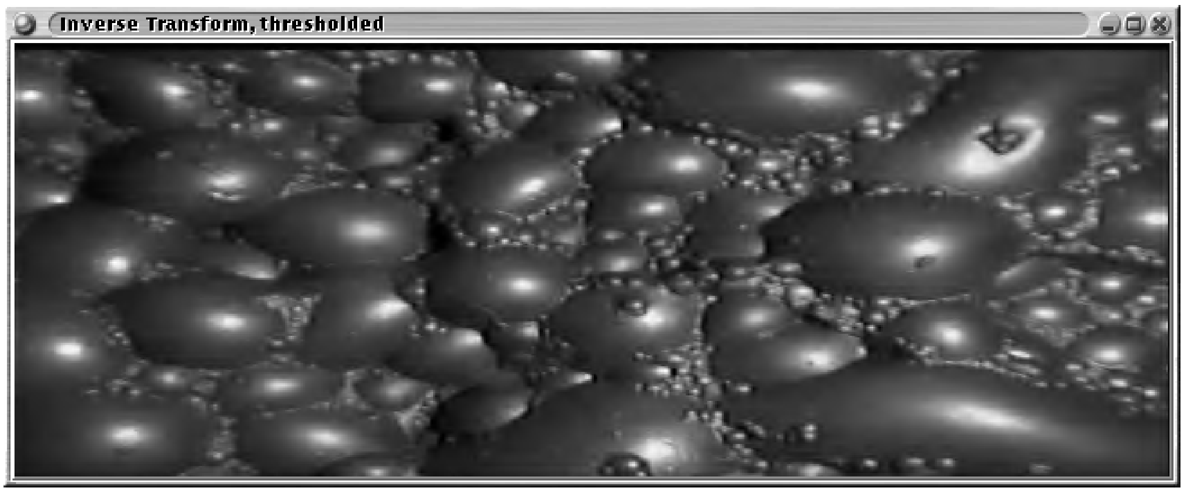
σ is the noise standard deviation. γ_1 is dependent on the wavelet transform used. Donoho does propose a simplification which sets γ_1 to 1. A difficulty seems to be estimation of σ but this can be done experimentally.

(iii) Invert the transform to convert the modified coefficients back to the spatial or time domain.

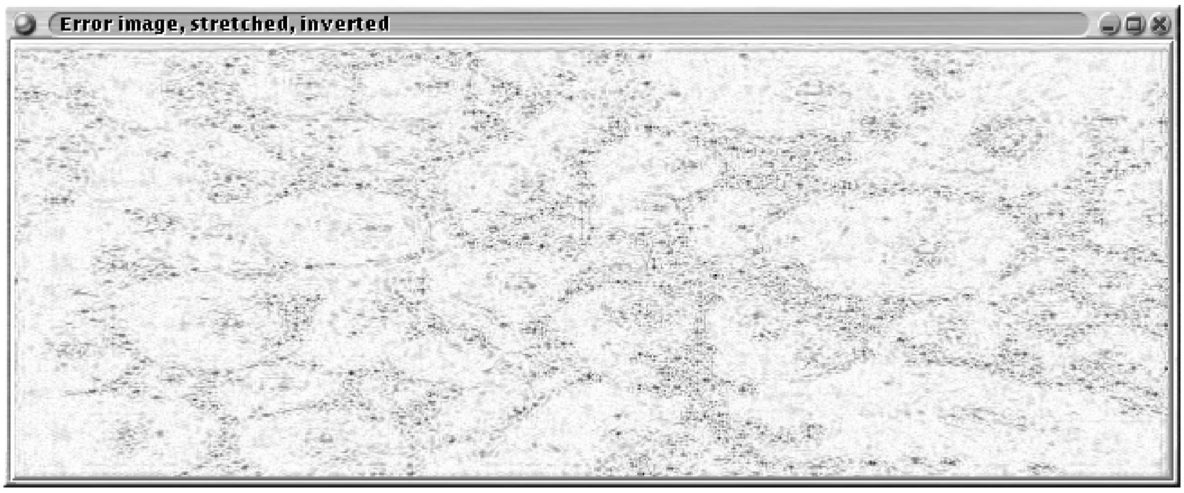
As is the case for hard thresholding, the denoised image is thresholded to ensure that the pixel intensities lie within those of the original image.

Figure 10.3 shows the denoised, error, marker and watershed segmented images. The error image is the difference between the original and the denoised image. To make the changes more visible, the error image has been histogram stretched and inverted (see section 11.1 for an introduction to histogram manipulation for contrast enhancement). The threshold is selected to provide good denoising without distorting the denoised image. The sigma value selected is 15. The marker image is generated from the denoised image. Note that the markers become blocky. The error image shows that most of the information lost lies in the textured regions where small bubbles lie between large bubbles. The watershed segmentation shows over-segmentation. This is likely due to fragmented markers and small point-like spurious markers visible on the marker image. The denoised image shows no obvious artifacts.

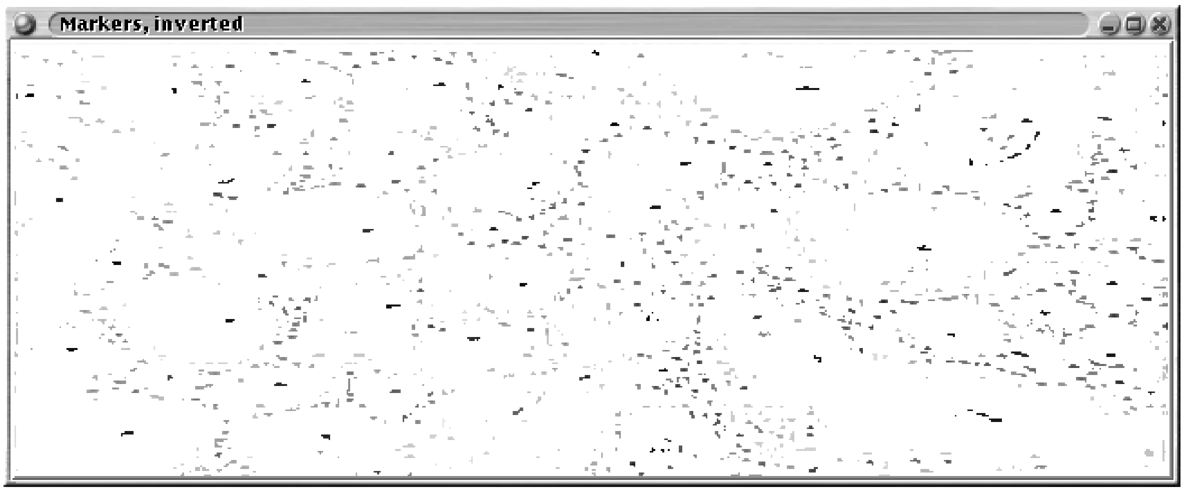
Table 10.3 shows the decrease in the number of watershed regions as the soft threshold sigma increases. This decrease may be explained by the removal of small and noise maxima. The maxima in the image form the domes of the image which become the markers. It is clear that soft thresholding is not sufficient to denoise a froth image sufficiently to allow for a good segmentation.



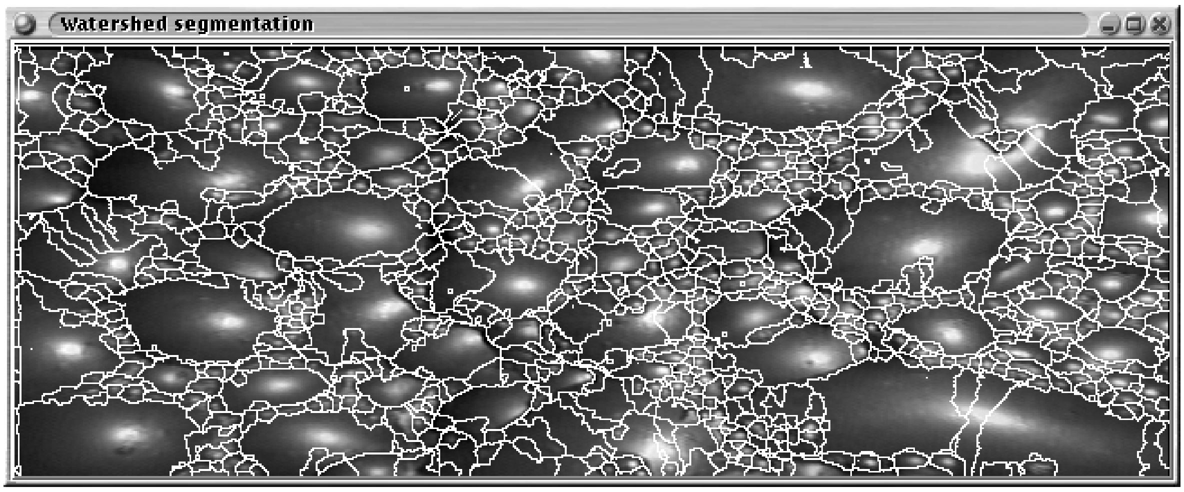
(a) Denoised image.



(b) Error image.



(c) Marker image.



(d) Watershed segmentation.

FIGURE 10.3 Wavelet denoising via soft thresholding.

TABLE 10.3 Decrease in the number of watershed regions as the soft threshold increases.

Soft threshold Sigma	# Watershed regions
1	1980
3	1354
5	1267
10	1197
15	1090
20	1027

10.2.3 Percentage thresholding

Percentage thresholding is described by Goswami and Chan [51][pg. 216-218], and involves setting to zero a specified percentage of the multi-resolution analysis coefficients. Typically the small magnitude coefficients are preferentially set to zero.

Again, the pixel intensities of denoised image are forced to lie within those of the original image.

Figure 10.4 shows the denoised, error, marker and watershed segmented images of the standard test image after 93% of the wavelet high-pass (or difference) coefficients have been removed. The threshold selected is chosen to best remove noise, while minimising denoising artifacts, while enabling good segmentation of small bubbles. Note that the denoised image shows few obvious artifacts. The marker image is generated from the denoised image. The marker image shows a number of small point-like markers which do result in over-segmentation. The histogram stretched (see section 11.1 for an introduction to histogram based contrast enhancement techniques) and inverted error image shows that most of the discarded information lies in the areas occupied by small bubbles. The error image is generated by subtracting the denoised image from the original. The segmented image, however, shows that these small bubbles have been well segmented, although larger bubbles tend to be over-segmented.

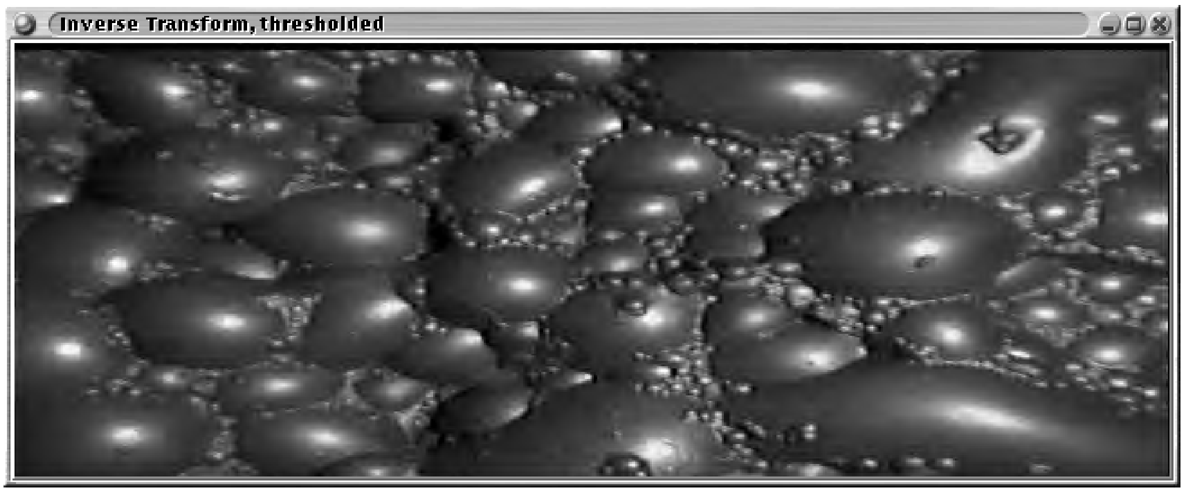
Table 10.4 shows the decrease in the number of watershed regions as the percentage of coefficients discarded increases. Clearly, as maxima are removed, these maxima no longer appear as domes, and hence the number of markers and watershed regions decreases. At some point, however, significant signal energy begins to be discarded. Some small bubbles begin to be merged, and the denoised image begins to resemble the original less and less. The number of surviving watershed regions indicates that, on its own, percentage thresholding does not suffice to remove the maxima causing over-segmentation.

TABLE 10.4 Decrease in the number of watershed regions as the percentage of coefficients discarded increases.

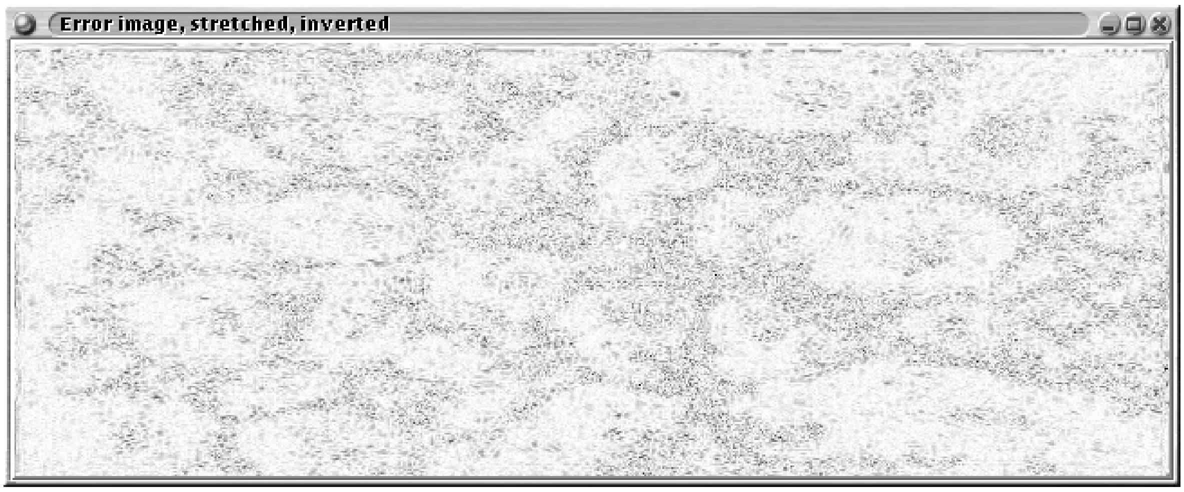
Percentage threshold	# Watershed regions
80%	1940
90%	1752
93%	1640
94%	1567
95%	1519
98%	1169

10.2.4 Lorentz filtering

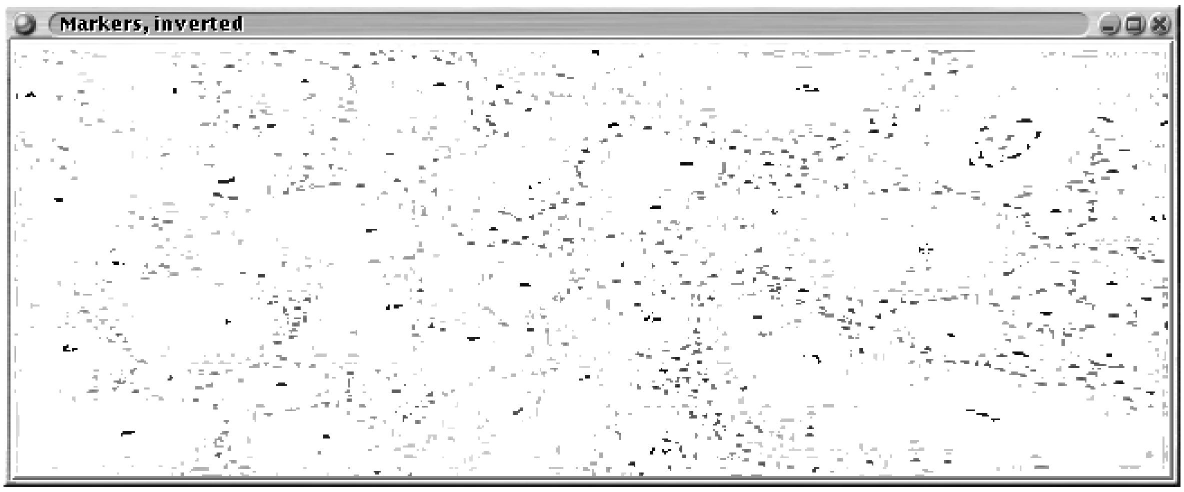
Lorentz filtering is a variant of percentage thresholding. The percentage of coefficients which are removed is found automatically. Lorentz filtering for wavelets was proposed by Goel and Vidakovic [49]. The Lorentz curve for a population is defined as follows,



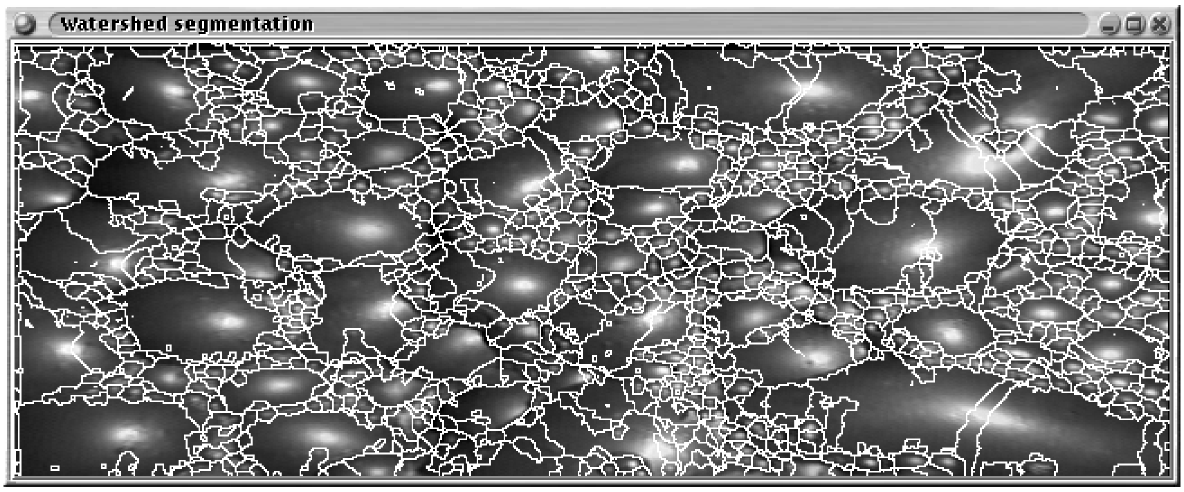
(a) Denoised image.



(b) Error image.



(c) Marker image.



(d) Watershed segmentation.

FIGURE 10.4 Wavelet denoising via percentage thresholding.

$$S(x) = \frac{\sum_{k=1}^{x} p(k)}{\sum_{i=1}^{N} p(i)} \tag{10.4}$$

where, $p(k)$ denotes the members of the population ranked in increasing magnitude. x, k, i are remapped to lie within 0 and 1 inclusive. Historically, $S(x)$ denoted the wealth of a number of individuals in a population, if the individuals were ranked in order of increasing wealth [37, 49, 79]. It is clear that the Lorentz curve, for a population with equal ranking, is a line of unit slope. (This is the case of all members of the populace being equally wealthy, Utopia perhaps.) Figure 10.5 shows some example Lorentz curves for the test image and multi-resolution analyses for different wavelets. Note that the Lorentz curves for the wavelet decompositions show most of the image energy is packed into a few coefficients.

The threshold is selected by setting coefficients whose energy is below the average energy of the coefficients (and hence image energy, for orthonormal bases [51]), to zero. The coefficients with energy above, or at, the average energy remain unchanged.

The proportion of wavelet coefficients removed, is the proportion of wavelet coefficients below, or at, the average energy of all the wavelet coefficients, or:

$$p_0 = \frac{1}{N}\sum_{k=1}^{N} I(d_k^2 \leq \overline{d^2}) \tag{10.5}$$

where, N is the number of coefficients and $I()$ is an indicator function,

$$I(w) = \begin{cases} 1, & d_k^2 \leq \overline{d^2}, \\ 0, & \text{otherwise, and,} \end{cases} \tag{10.6}$$

$$\overline{d^2} = \frac{1}{N} \sum_{k=1}^{N} d_k^2 \tag{10.7}$$

The lower $p_0 \times 100\%$ coefficients are removed as in percentage thresholding.

It can be shown that the distance between the Utopian line and the Lorentz curve is maximum at the segment of the Lorentz curve which has unit slope. Examining figure 10.5, it may be seen that this occurs at the point where the small magnitude coefficients give way to the high value coefficients. The x value of this point provides a percentage threshold, below which the wavelet coefficients have low energy, and above which, it can be argued, the coefficients are significant.

Once again, thresholding forces the denoised image intensities to lie within those of the original image.

Figure 10.6 shows the denoised, error, marker and watershed segmented images of the standard test image after Lorentz thresholding. The percentage threshold selected by this technique is 93.61%. The number of watershed regions is 1 610. Note that the denoised image shows few obvious artifacts. The marker image is generated from the denoised image. The marker image shows a number of small point-like markers which do result in over-segmentation. The histogram stretched (see section 11.1 for an introduction to histogram based contrast enhancement techniques) and inverted error image shows that most of the discarded information lies in the areas occupied by the small bubbles. The error image is generated by subtracting the denoised image from the original. The segmented image, however, shows that these small bubbles have been well segmented, although larger bubbles tend to be over-segmented.

10.3 Denoising using Set Partitioning in Hierarchical Trees (SPIHT)

This is another example of denoising taking into account the different compression characteristics of signal and noise. Set Partitioning in Hierarchical Trees (SPIHT) is a transform based compression technique. Essentially SPIHT is a form of scalar quantisation in the multi-resolution Wavelet domain [121] with progressive transmission of coefficients in order of magnitude. The SPIHT algorithm requires integer coefficient values.

The SPIHT algorithm proceeds as follows in essence, with more sophisticated coding of the data in the sorting pass [121] (the coding stage is neglected for purposes of evaluating the noise removal properties of this lossy compression method):

(i) Transform the input image via a multi-resolution analysis.

(ii) Convert the transform domain coefficients to sign and magnitude representation.

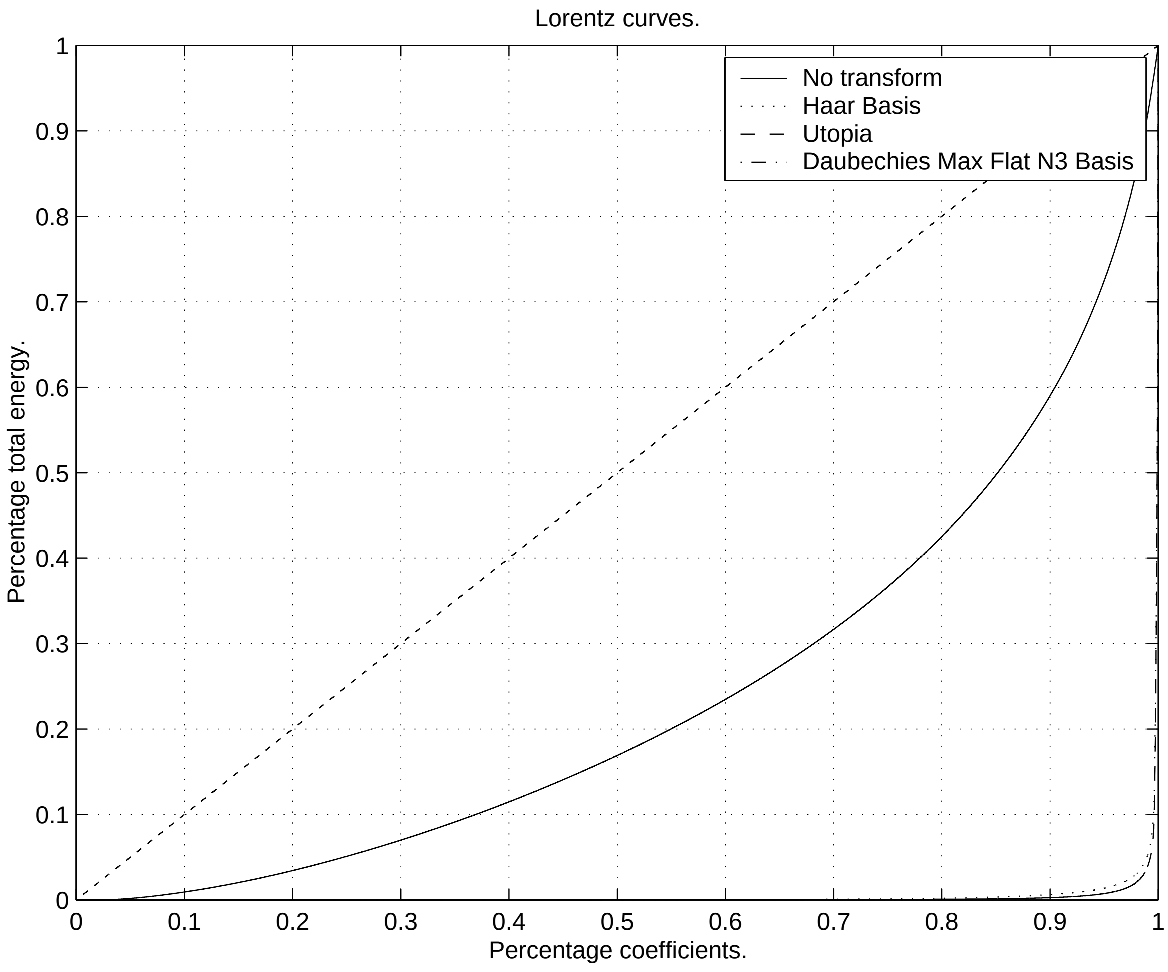
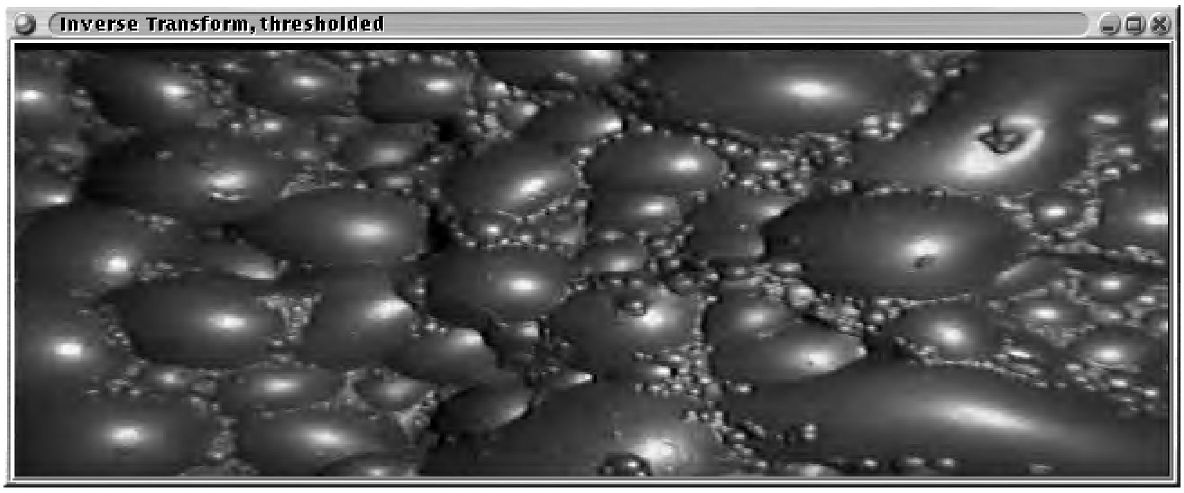
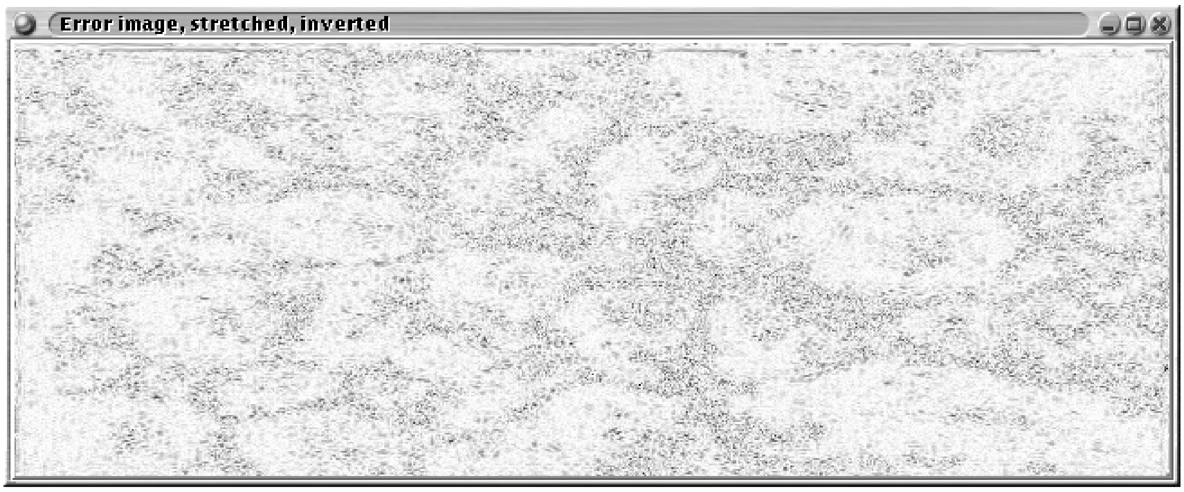


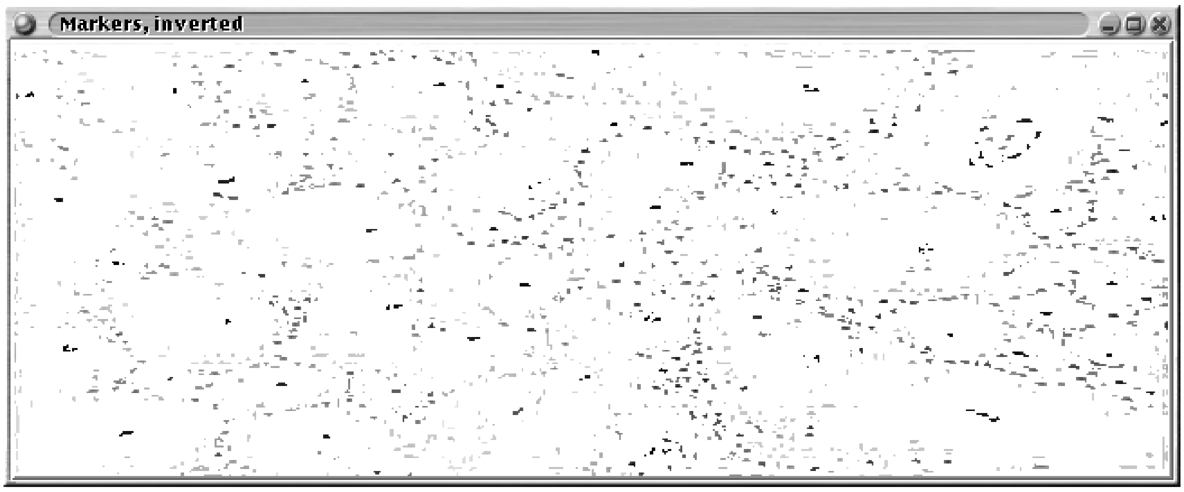
FIGURE 10.5 Some example Lorentz curves.



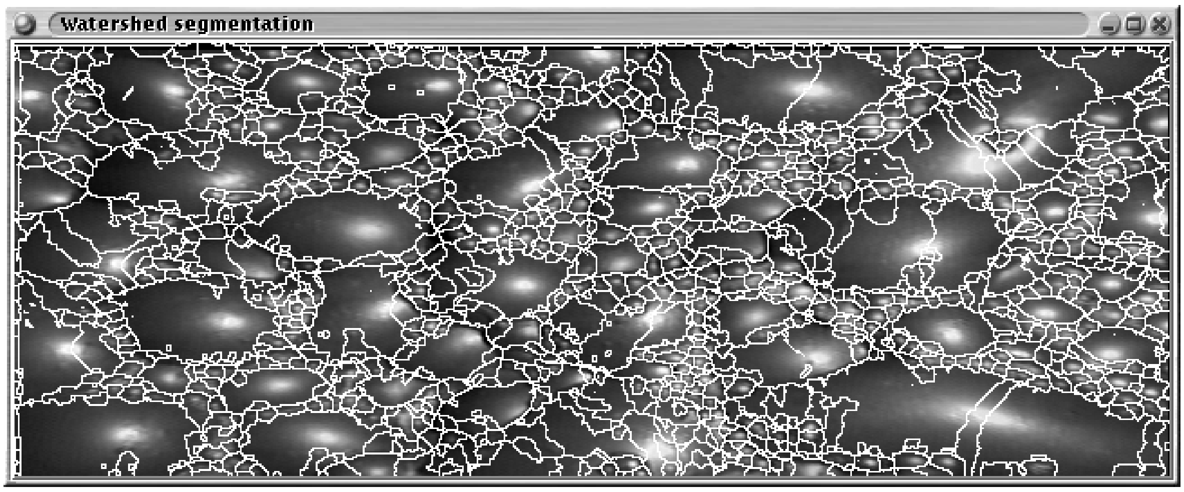
(a) Denoised image.



(b) Error image.



(c) Marker image.



(d) Watershed segmentation.

FIGURE 10.6 Wavelet denoising via Lorentz thresholding.

(iii) Output the number of bits of the largest coefficient in absolute magnitude (n).

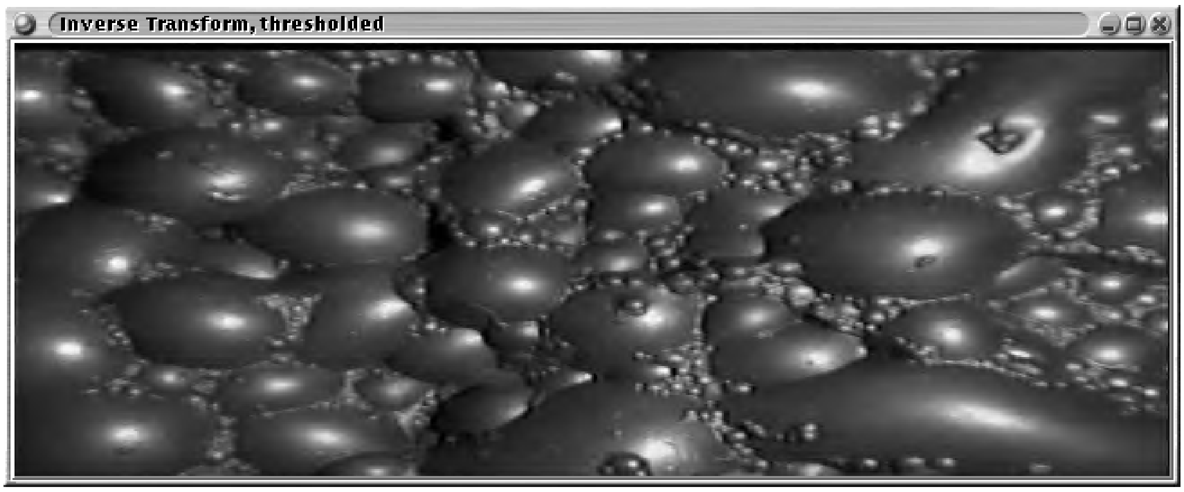
(iv) The sorting pass. Transmit the positions (p) and signs of all coefficients (C_p) satisfying, $2^n \leq \|C_p\| < 2^{n+1}$.

(v) The refinement pass. Transmit the n-th most significant bit of all pixels transmitted in previous sorting passes in the same order used previously.

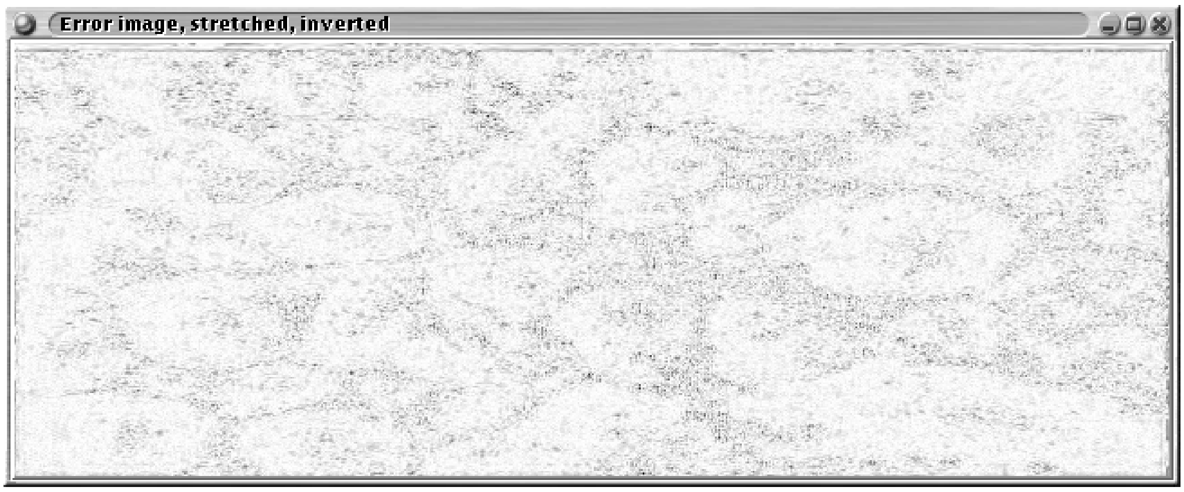
(vi) Decrement n by one and loop back to the sorting pass.

To simulate the effect of denoising via SPIHT it is then unnecessary to implement the entire algorithm. One selects a desired maximum distortion or error by choosing a MSE value. The Peak Signal to Noise Ratio (PSNR) is then inferred. The Algorithm proceeds as described, the MSE error decreasing with each bit transmitted until the MSE reaches the desired MSE, or until the transmitted bit reduces the MSE below that desired. The process stops and the resultant image is then reconstructed from the transmitted wavelet coefficients. It is a fundamental property of orthonormal linear transforms (of which Orthonormal Wavelet Transforms are examples) that the MSE in the transform domain between two images is the same as the MSE in the spatial domain. The performance of the algorithm is very dependent on the decorrelating properties of the Wavelets used. This implies a matching of the smoothness of the Wavelet (or number of vanishing moments [3]) with the expected smoothness of the image.

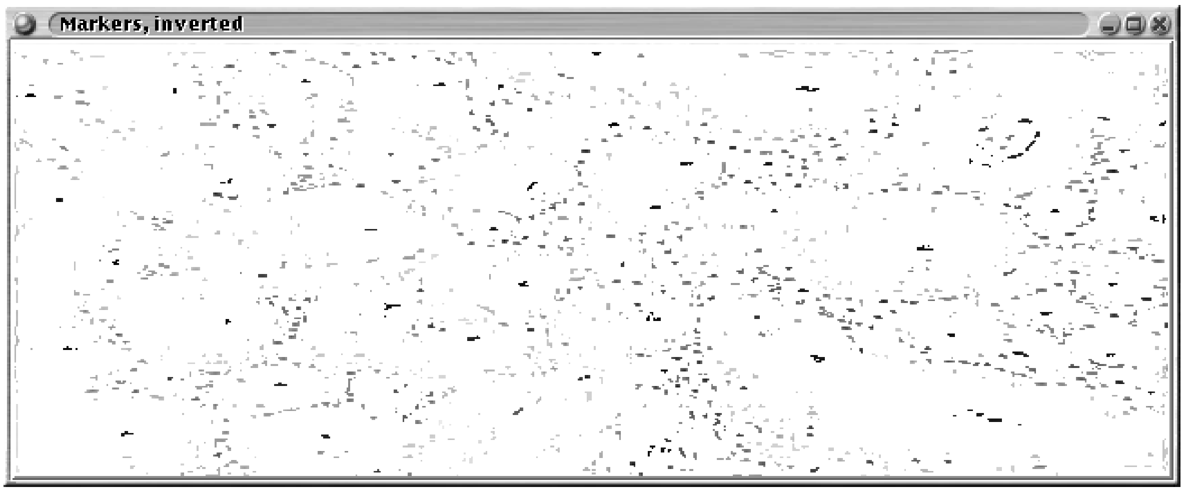
Figure 10.7 shows the effect of applying SPIHT as a means for denoising froth images. The depth of the multi-resolution was 4 and the desired MSE 25. The MSE selected allows for a tradeoff between reducing the over-segmentation of large bubbles and inhibiting the merging of small bubbles.



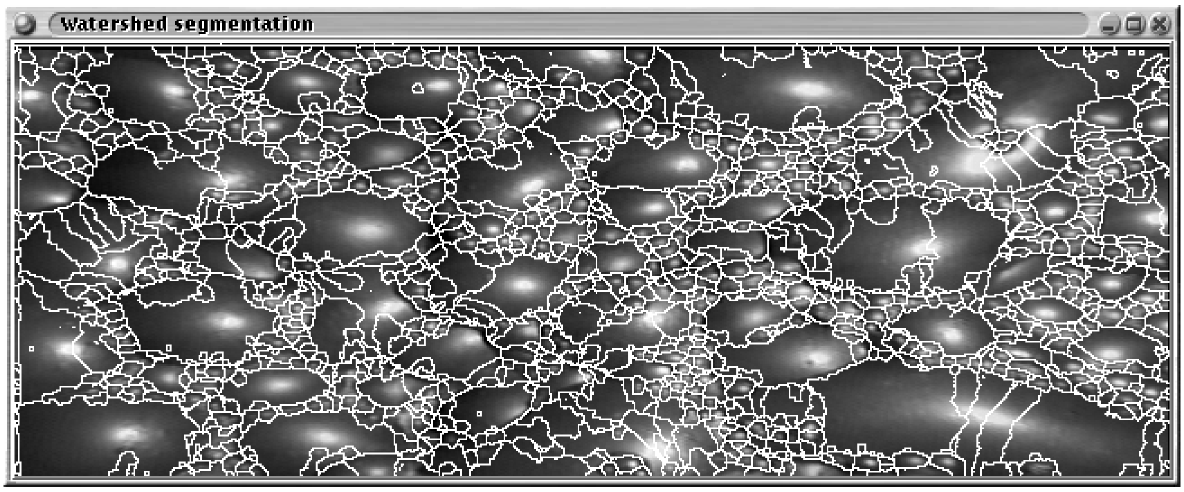
(a) Denoised image.



(b) Error image.



(c) Marker image.



(d) Watershed segmentation.

FIGURE 10.7 Wavelet denoising via SPIHT.

The generated PSNR was 31dB. The wavelet used was the Daubechies maximum flat N3 wavelet with 3 orders of regularity [139] (quadratic regions approximated exactly). The denoised region does not show any noticeable artifacts. The marker image is generated from the denoised image. As before, the error image is the difference between the original and denoised images. The error image is histogram stretched and inverted to improve the visibility of the errors (see section 11.1 for an introduction to simple contrast enhancement). The segmented image shows over-segmentation of large bubbles. This is likely due to the large number of point-like markers on the marker image. The selected MSE denoises well without introducing artifacts on the denoised image.

Table 10.5 shows the decrease in the number of watershed regions as the allowed error increases. Note that even for significant errors the number of watershed regions remains large.

TABLE 10.5 Decrease in the number of watershed regions as the MSE for SPIHT increases.

MSE	# Watershed regions
1	2456
4	1883
10	1545
25	1443
50	1341
100	1128

10.3.1 Discussion

SPIHT essentially ranks all the wavelet coefficients in order of magnitude. The coefficients are then selectively quantised from smallest to largest until the error introduced by the quantisation reaches a pre-selected limit. The quantisation is allowed to gradually increase until the desired error is achieved. The remainder of the SPIHT algorithm lies essentially in coding details and bit-plane manipulation. The bit-plane manipulation achieves the quantisation effect. The quantisation process used here, rounds down, rather than to the nearest integer as is usual for quantisation (see section 10.4 for more details on quantisation).

This process of selective quantisation can be extended to other transforms which are invertible. This selective quantisation technique is then order N ($O(N)$), where N is the number of transform coefficients. One sorts the coefficients according to the number of bits needed to represent the coefficient, then iterate though the coefficients, quantising on the absolute value.

The number of iterations, at worst, is the number of bits per coefficient. Orthonormal linear transforms possess a conservation of energy property, the energy of the coefficients in the transform domain is the same as the energy of the unprocessed signal (this is sometimes referred to as Parceval's equality)[3]. Parceval's equality allows for the error in the reconstructed image to be directly computed as the quantisation in the transform domain proceeds.

10.4 Denoising using Joint Photographic Experts Group (JPEG) quantisation matrices

The JPEG compression standard for lossy image compression defines a process of quantisation based on quantising the Discrete Cosine Transform (DCT) coefficients of 8×8 non-overlapping blocks of an image [145]. This process is similar to that used by the MPEG for intra-frame compression [70]. The use of the DCT is desirable since the de-correlation properties of the transform is close to that of the Karhunen-Loéve transform under certain conditions [3, 50]. The process of quantisation discards some data and is optimised to quantise high frequencies preferentially. Table 10.6 shows the quantisation coefficients typically used. These quantisation coefficients are scaled up as more compression is required.

The process of denoising using the DCT quantisation coefficients is similar to the other compression based schemes. The data is quantised to a particular PSNR which is directly related to the compression, and then inversely quantised. The quantisation process involved scaling the coefficients down and rounding (usually rounding the magnitude down) [145]. Provided the quantisation process is not too severe, the resulting data is largely image data and the discarded data largely noise. This assumption is justified for additive low power Gaussian white noise. The central limit theorem [93] is used to justify the Gaussian noise distribution. Low power additive noise is justified by examination of the froth images, the smooth regions do not show significant noise levels or multiplicative noise.

TABLE 10.6 Table of DCT quantisation coefficients for JPEG. Horizontal spatial frequency runs from left to right, vertical spatial frequency runs from top to bottom.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Quantisation proceeds as follows [145],

$$C_{\text{quantised}} = \text{Round}\left(\frac{C_{\text{coefficient}}}{Q_{\text{coefficient}} * M_{\text{multiplier}}}\right) \tag{10.8}$$

De-quantisation proceeds as follows,

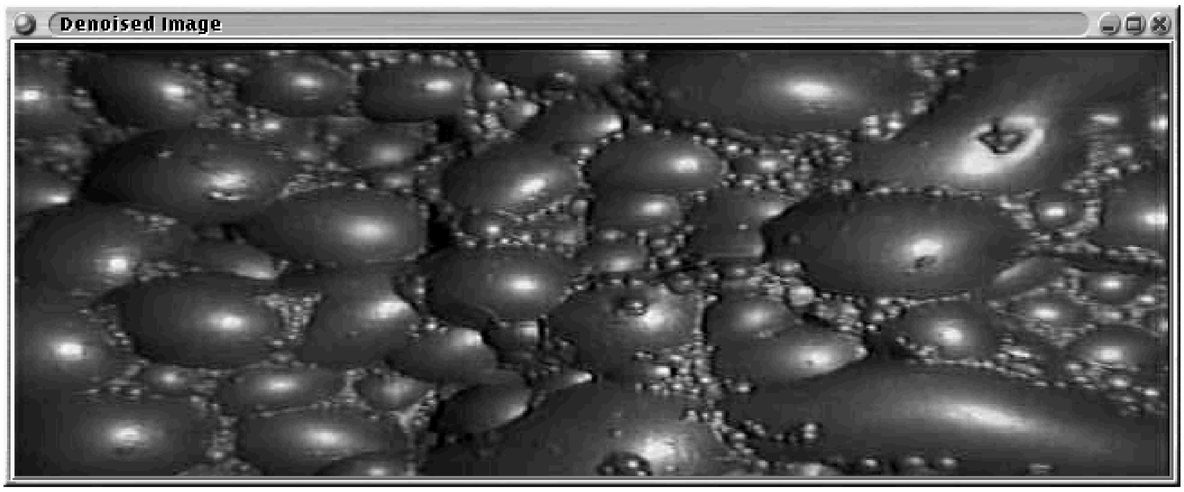
$$C_{\text{de-quantised}} = C_{\text{quantised}} * Q_{\text{coefficient}} * M_{\text{multiplier}} \tag{10.9}$$

Here,

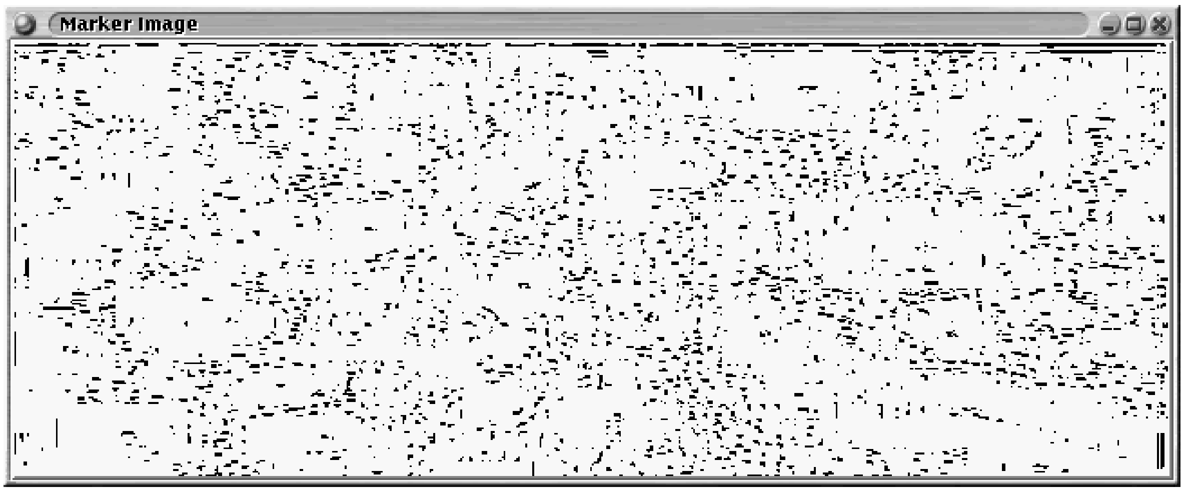
- C denotes the coefficient to be quantised.
- $C_{\text{quantised}}$, the quantised coefficient.
- $Q_{\text{coefficient}}$, the amount to quantise by.
- $M_{\text{multiplier}}$, the multiplier for $Q_{\text{coefficient}}$.

Table 10.7 shows the decrease in the number of watershed regions as the quantisation scale factor increases. Figures 10.8, 10.9 and 10.10 show the effect on the image, markers and watershed segmentation as the quantisation scale factor increases. The test image used is the standard test image. Examination of figure 10.9 shows that even for a modest multiplier (3) on the quantisation coefficients, the denoising is minimal, but the introduced blocking artifacts severe. The artifacts are caused by the discontinuities between the different blocks. These discontinuities are the result of the quantisation of the DC coefficient of neighbouring blocks.

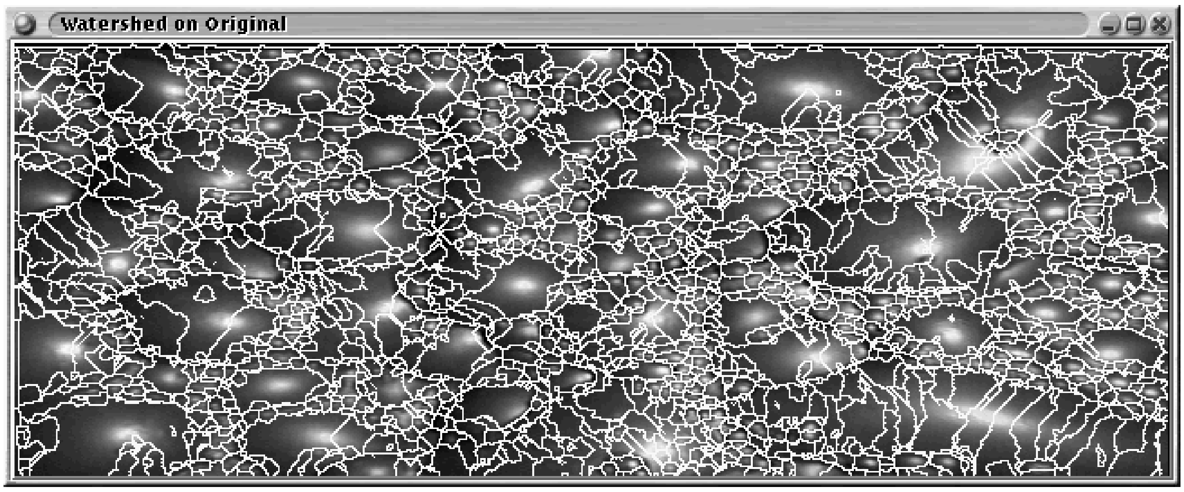
This denoising approach has implications for compressing froth image sequences. MPEG2 style compression uses a form of DCT quantisation similar to that of JPEG [70]. The quantisation process may introduce artifacts difficult to see on the denoised image, but these artifacts may well be visible in the marker image. See the denoised and marker image contained in figure 10.8, notice that the markers have a blocky rather than an irregular shape.



(a) Denoised image.

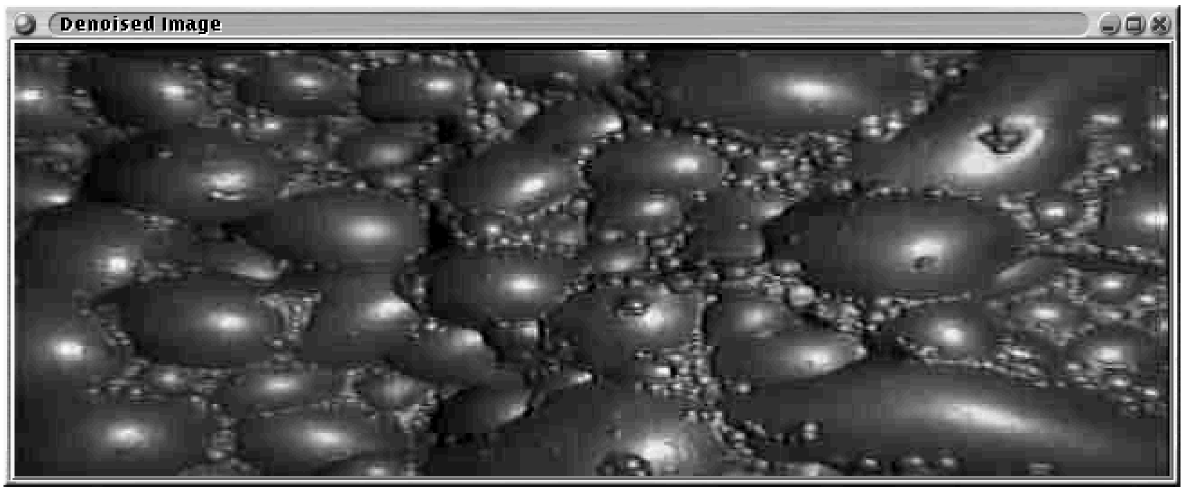


(b) Marker image.

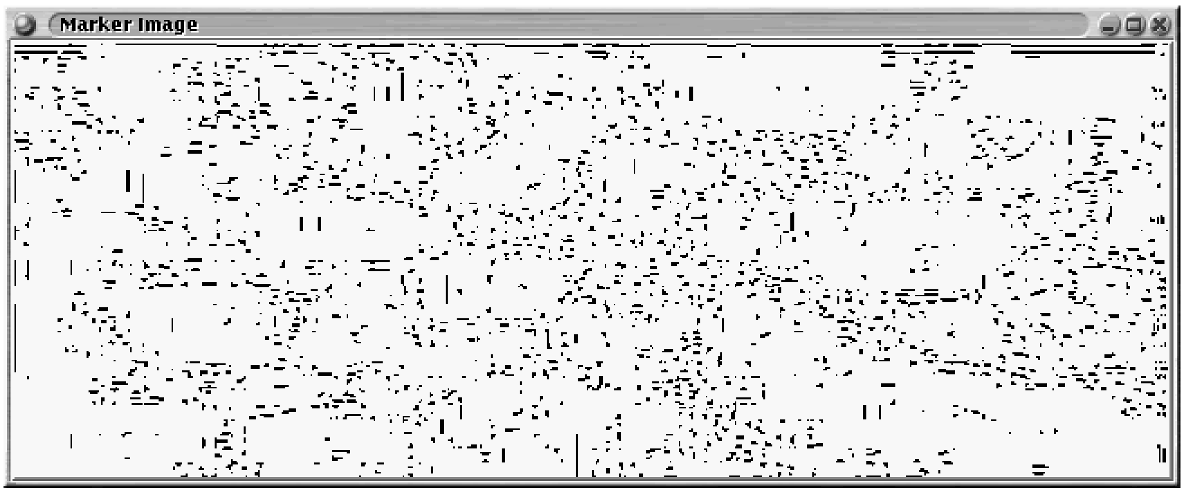


(c) Watershed segmented image.

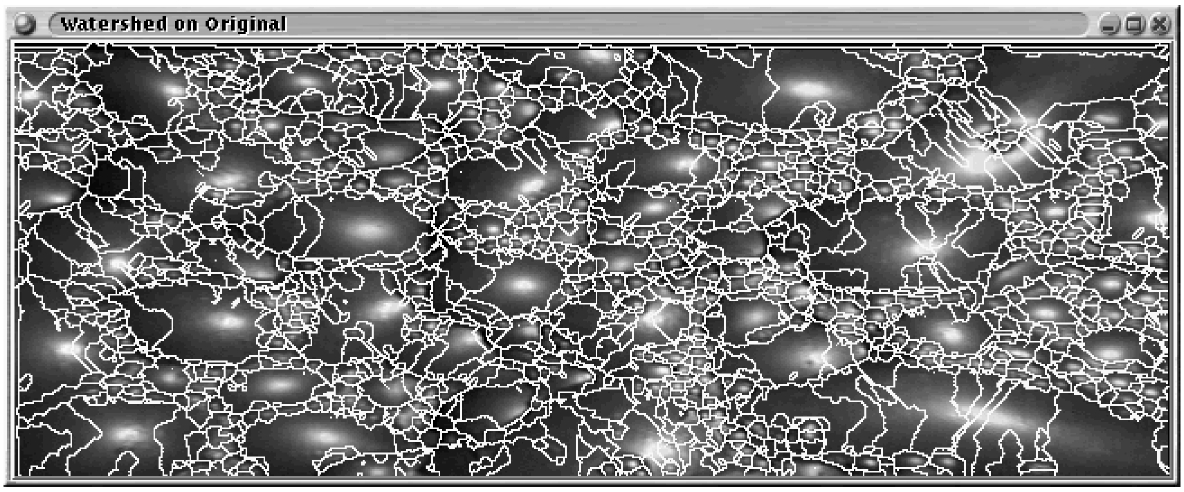
FIGURE 10.8 Result of DCT JPEG quantisation (multiplier 1) on image, marker and watershed.



(a) Denoised image.

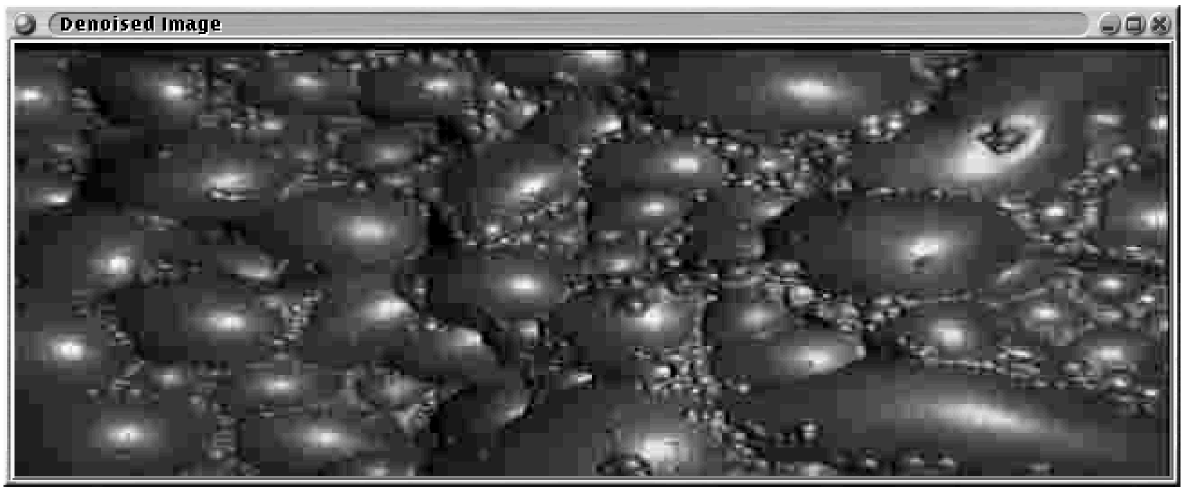


(b) Marker image.

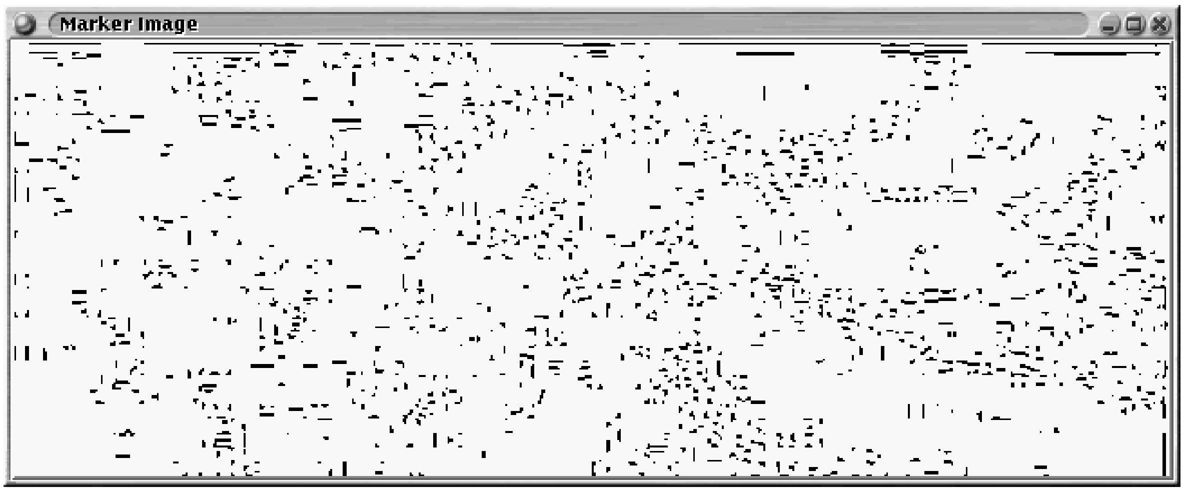


(c) Watershed segmented image.

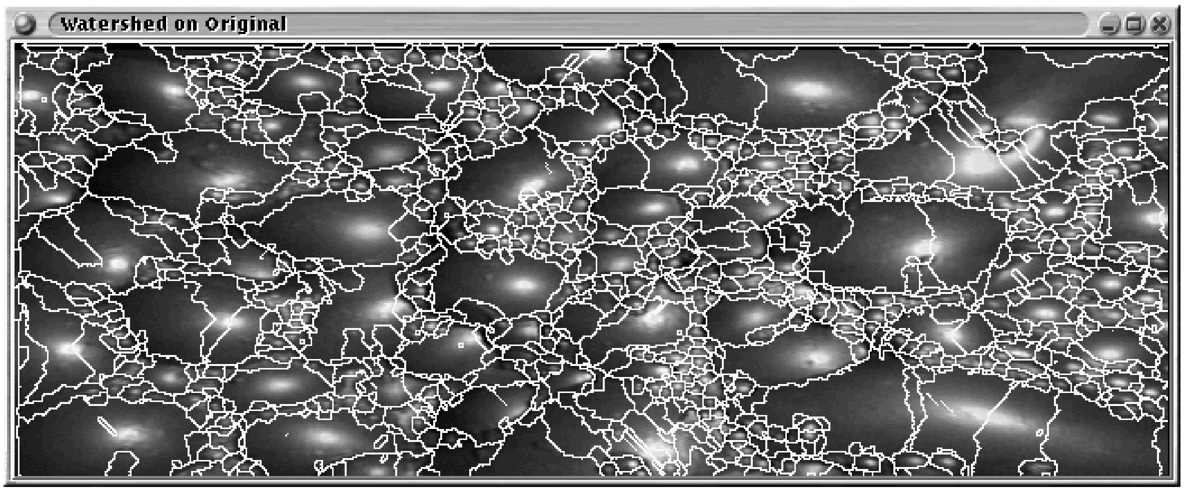
FIGURE 10.9 Result of DCT JPEG quantisation (multiplier 3) on image, marker and watershed.



(a) Denoised image.



(b) Marker image.



(c) Watershed segmented image.

FIGURE 10.10 Result of DCT JPEG quantisation (multiplier 6) on image, marker and watershed.

TABLE 10.7 Decrease in the number of watershed regions as JPEG quantisation scale factor increases.

Multiplier	# Watershed regions
1	2279
2	1981
3	1726
4	1492
5	1414
6	1180
7	1058
8	961
9	859
10	788
11	717
12	674
20	333

10.5 Is there common ground for the various Occam Filters?

Occam filters transform the data into some transform domain and then apply a function to each coefficient, usually a magnitude based function. In a sense the process is as follows,

$$\begin{aligned} g(x) &= L^{-1}\left(S^{-1}(H(S(L(f(x)))))\right) \\ \Rightarrow G(\omega) &= S^{-1}(H(S(F(\omega)))) \end{aligned}$$

(i) $L(\cdot)$ the (possibly non-linear) transform applied to the data.

(ii) $L^{-1}(\cdot)$ the inverse of the transform, reconstructing the signal from the transform domain coefficients.

(iii) $S(F(\omega))$ presents each coefficient in order of size, in effect generating a sorted list of coefficients.

(iv) $S^{-1}(\cdot)$ inverts the process and reconstructs the signal from the sorted list.

(v) $H(\cdot)$ is the function applied to each coefficient. Note that H does not affect the sorted order of the coefficients.

The MSE of the Occam filter can be derived as follows,

$$\begin{aligned}
F &= S^{-1} \cdot S \cdot F \\
G &= S^{-1} \cdot H \cdot S \cdot F \\
F - G &= (S^{-1} \cdot S \cdot F) - (S^{-1} \cdot H \cdot S \cdot F) \\
&= S^{-1} \cdot (S \cdot F - H \cdot S \cdot F) \\
&= S^{-1} \cdot [I - H] \cdot S \cdot F \\
\mathrm{trace}(\|F - G\|^2) &= \mathrm{trace}((S^{-1} \cdot [I - H] \cdot S \cdot F) \cdot (S^{-1} \cdot [I - H] \cdot S \cdot F)^*) \\
&= \mathrm{trace}(([I - H] \cdot S \cdot F) \cdot (S^{-1^*} \cdot [I - H]^* \cdot S^* \cdot F^*)) \\
&= \mathrm{trace}(S^T \cdot [I - H]^T \cdot S \cdot F \cdot S^{-1} \cdot S^T [I - H]^{*T} \cdot F^*) \\
&= \mathrm{trace}([I - H]^T \cdot S \cdot (S^{-1} \cdot S^T)^T \cdot F^T \cdot [I - H]^{*T} \cdot F^*) \\
&= \mathrm{trace}((S \cdot (S^{-1} \cdot S^T)^T)^T \cdot [I - H] \cdot [I - H]^* \cdot F \cdot F^*) \\
&= \mathrm{trace}([I - H] \cdot [I - H]^* \cdot F \cdot F^*) \\
&= \mathrm{trace}((I \cdot I^* - I \cdot H^* - H \cdot I^* + H \cdot H^*) \cdot F \cdot F^*
\end{aligned}$$

It should be noted that S, S^{-1} and their transposes are permutation matrices. It can be shown that the trace of the matrix product of a permutation matrix and a diagonal matrix, is the trace of the diagonal matrix. Specification of H then specifies the Occam filter completely. The rest of the filter is then either coding details which can be neglected, or are subsumed into the sorting step (or specification of S and S^{-1}). The sorting step can be likened to applying a series of stack filters to the data, to extract each Order statistic from the first (or maximum) to the last (or minimum).

Chapter 11

Image enhancement

The goal of image enhancement, in the context of froth imaging, is to improve the segmentation and accuracy of the motion estimation process. Some simple techniques for image enhancement are examined, including, edge and contrast enhancement.

The goal of contrast enhancement is to make the image visually more appealing and to attempt to eliminate lighting variations. The usual histogram stretch and equalisation techniques are examined. Homomorphic filtering, for example, is well known to compensate for differences in lighting conditions [50, 73].

Edge enhancement is a used for much the same reasons. Edges are well known to be important in the Human Visual System [50, 73]. In order to improve the segmentation via the watershed process it is required to take advantage of the quirks of the watershed algorithm. Watershed lines tend to be on the ridges of an image (except on flat regions where the flat region is then split). Enhancing the ridges, making them steeper and higher, can then improve the discrimination properties of the watershed. An example of this would be the case of a small bubble on top of a larger bubble. Since froth images tend to be smooth, the ridge between the large and small bubble is low. The flooding of the large bubble then does not flood around the small bubble, as it should to properly segment the small bubble. If the transition boundary (ridge) were to be exaggerated, the watershed would have a better chance of extracting the small bubble correctly.

Several examples of contrast and edge enhancement techniques will be examined. To improve the noise rejection, the images are first denoised by mean filtering with a Gaussian kernel with $\sigma = 1$. To examine the effects of edge enhancement in isolation, linear filtering is used for denoising. The kernel is truncated to a 7×7 pixel footprint. The window size in both adaptive instances was 7×7 pixels. To show that this denoising stage does not significantly affect the froth structures but serves to remove noise, consider figure 11.1. The equalised difference image (figure 11.1(b)) shows an image with most pixels being uncorrelated with their neighbours. Some pixels, especially those on the boundaries of bubbles, show some correlation. This illustrates not only that edge information is carried by high frequencies, but also that high frequencies are especially contaminated by noise.

Clearly, a smaller radius Gaussian will not remove as much noise and a larger radius Gaussian may affect the froth structures too much.

The markers shown in the results are extracted via morphological methods. First the image is eroded via a rolling ball of radius one. The eroded image is greyscale reconstructed via dilation under the original image. Domes, regional maxima, are extracted by subtracting the reconstructed image from the original image. The domes, after optional thresholding, become the markers. The watershed used for segmentation is a marker based watershed.

11.1 Simple contrast enhancement

The simple contrast enhancement algorithms examined will be histogram stretching and equalisation. Histogram stretching involves linearly stretching the histogram of the image to use the full range of grey levels available. The full greyscale variation is thus more visible.

Figures 11.2 and 11.3 and show the effect of simple contrast enhancement on the standard froth image. Figures 11.4 and 11.5 show the effect of adaptive histogram stretching and equalisation on the standard froth image. The adaptive algorithm applies the histogram manipulation algorithm to a small region of the image and remaps the centre pixel of the region accordingly.

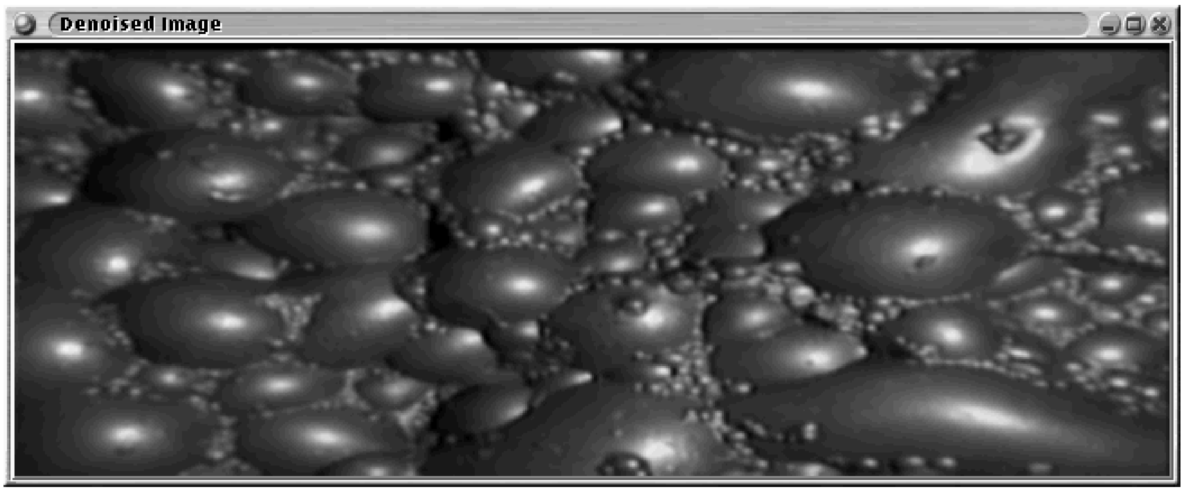
To improve the noise rejection, the images are first denoised by mean filtering with a Gaussian kernel with $\sigma = 1$. The kernel is truncated to a 7×7 pixel footprint. The window size in both adaptive instances was 7×7 pixels.

As expected, since simple stretching does not change the ordering of pixels, the watershed is unchanged. The histogram equalisation step does not change the segmentation significantly, although a few of the larger bubbles are better segmented. Histogram equalisation remaps pixel values non-linearly, and occasionally remaps pixels with similar intensities to the same intensity. This explains the improvement in the segmentation, fragmented highlights are merged and form a single marker.

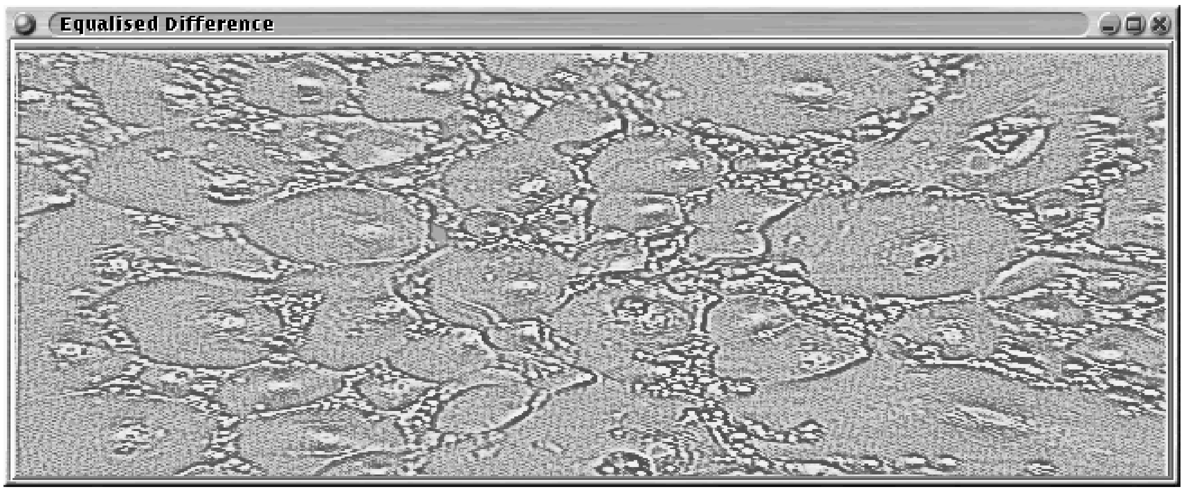
The adaptive stretching and equalisation algorithms (figures 11.4 and 11.5) can be seen to be noise sensitive. Both methods show regions which have a rough sandpaper like texture on the enhanced image. These regions show areas which are particularly sensitive to noise. Comparison of the location of these regions to the corresponding regions on the input image, show that these regions correspond to areas which have low intensity. Pixels in these low intensity regions, as expected, have lower signal to noise ratios. Notice that as the radius of the neighbourhood increases, the adaptive results become closer to the fully stretched or equalised cases.

The adaptive stretch proceeds by finding the maximum and minimum values in a circular neighbourhood of given radius. This is a simple and straightforward consequence of the usual definition of histogram stretching [73]. The intensity variation within the circular neighbourhood is linearly remapped to lie within the full variation of intensities allowed. The intensity of the centre pixel of the neighbourhood is linearly remapped to lie within the allowed range of intensities (0 to 255, for grey-scale images, for example).

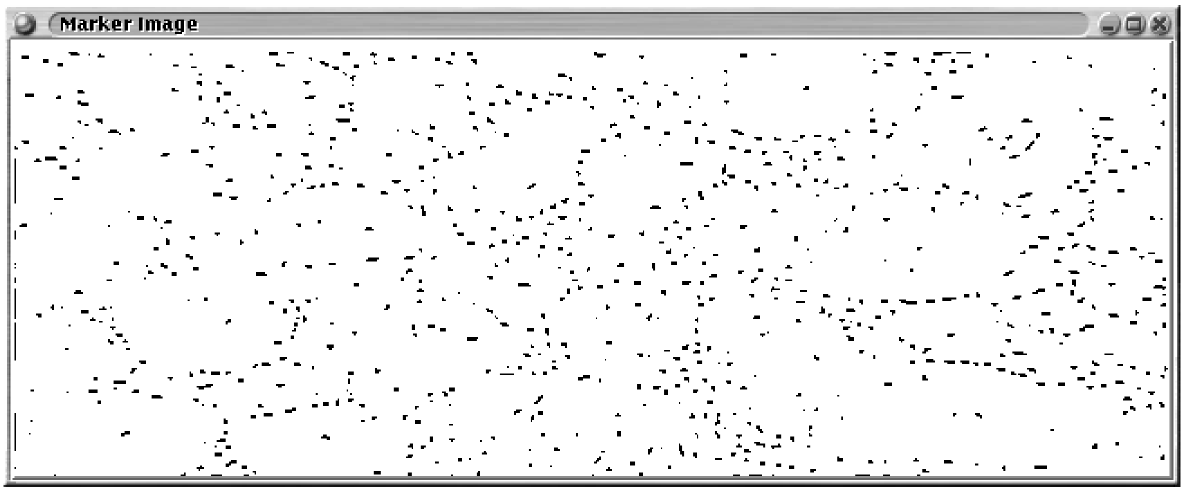
The adaptive equalisation proceeds by counting the number of pixels equal or less than the centre



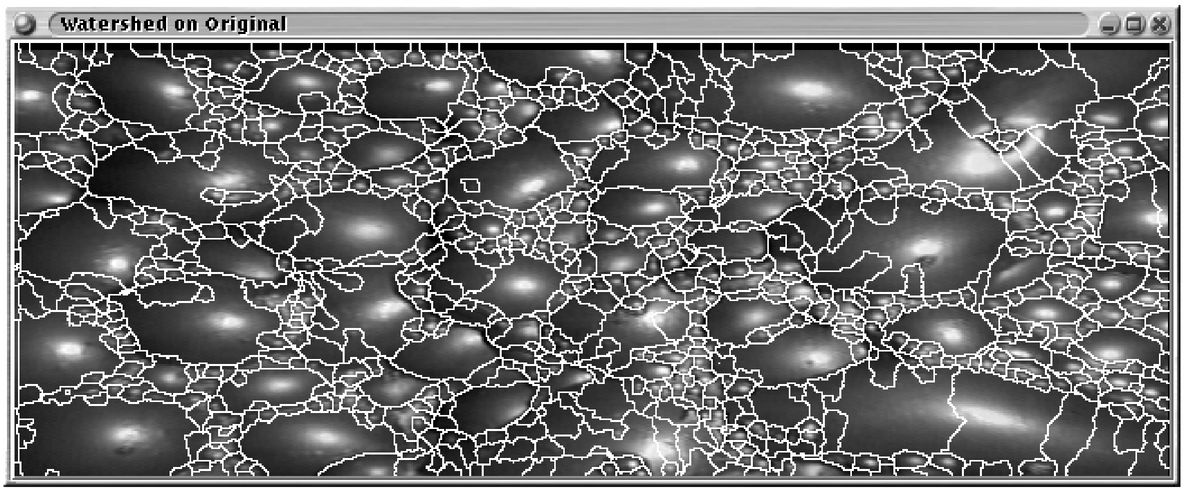
(a) The denoised image.



(b) Equalised difference.



(c) Markers of denoised image.



(d) Watershed segmented denoised image.

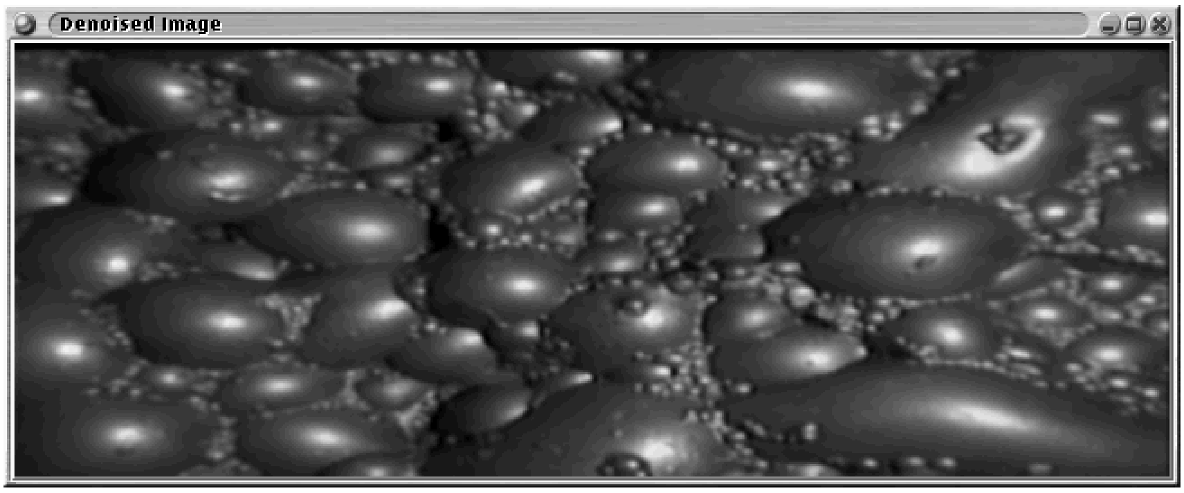
FIGURE 11.1 Justifying denoising before image enhancement.

pixel. The centre pixel intensity is remapped in proportion to the range of pixel intensities in the circular neighbourhood. The proportion is the number of pixels less than or equal to the centre pixel, to the number of pixels in the neighbourhood. This is again a simple and straightforward consequence of the usual definition of histogram equalisation (see [73] for a definition of histogram equalisation).

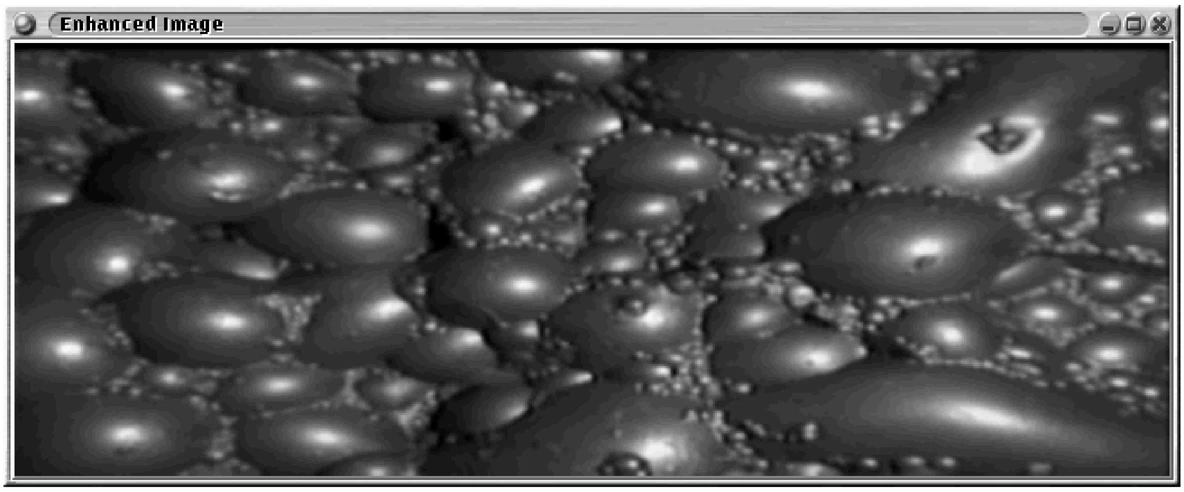
The adaptive stretching and equalisation steps are something of a disappointment, although the bubble boundaries are more visible, the segmentation is poor because of the noisiness of the enhanced image. Enhancement processes are noise sensitive because, in low contrast areas, small variations must be made more visible. The difficulty is that some of these variations are of the same magnitude as noise, which leads to noise being enhanced. This provides a case for denoising prior to enhancement. Cases for neighbourhoods of radii 5, 10 and 20 are presented (see figures 11.4 and 11.5). As expected, as the radius increases the adaptively enhanced images more resemble the simple contrast enhancement cases.

11.2 Edge enhancement

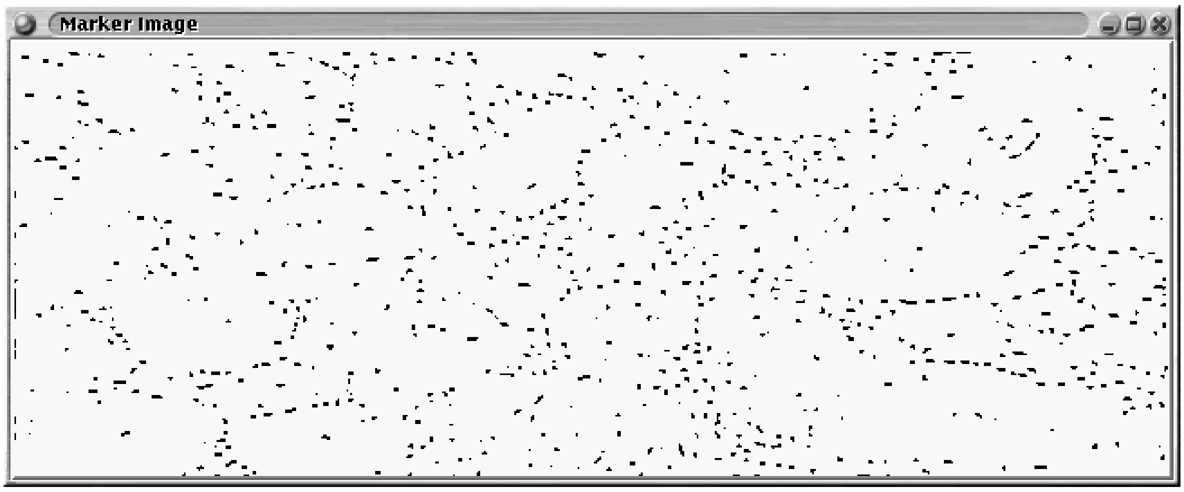
The edge enhancement algorithms used include a form of unsharp masking. The edge image is computed and added in to the original. Edge enhancement is complicated by the need to enhance the image data without enhancing noise. This requires a denoising step prior to the edge enhancement process.



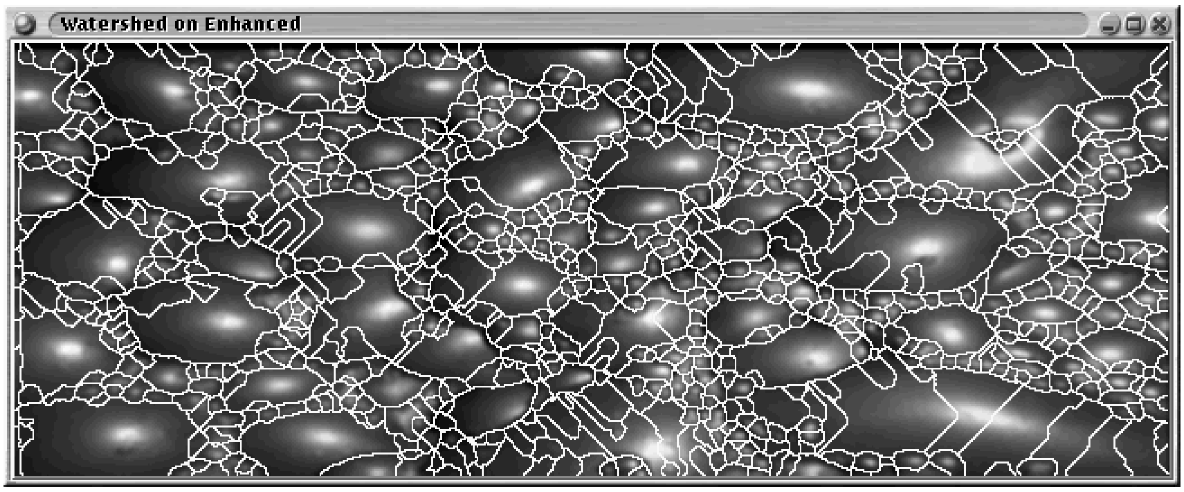
(a) Denoised image.



(b) Enhanced image.



(c) Marker image.



(d) Watershed segmented image.

FIGURE 11.2 Histogram stretching.

11.2.1 Unsharp masking

Unsharp masking was originally developed for enhancing photographs and was originally a dark-room process [73]. A high-pass image is generated by subtracting a low-pass copy from the original. This suppresses constant or slowly varying regions and improves the contrast of rapidly varying regions.

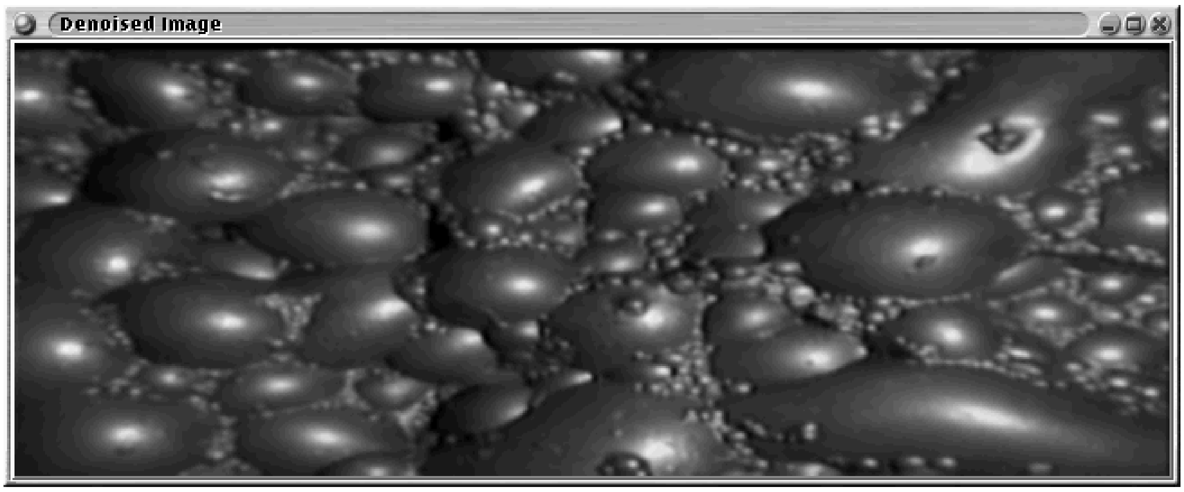
Prior to unsharp masking the images are denoised. A Gaussian kernel of $\sigma = 1$ and footprint 7×7 is used. This limits the enhancement of noise without blurring structures in the image. Figure 11.6 shows the enhanced image after a low-pass Gaussian with sigma of 1, 2 and 3 is used to generate the high-pass image as described. Close examination of the enhanced images do not show the bubble boundaries clearly delineated, or a well defined valley surrounding small bubbles on large bubbles. Unsharp masking is then not expected to improve the watershed. The fine structure of the image is visible.

11.2.2 Simple edge enhancement

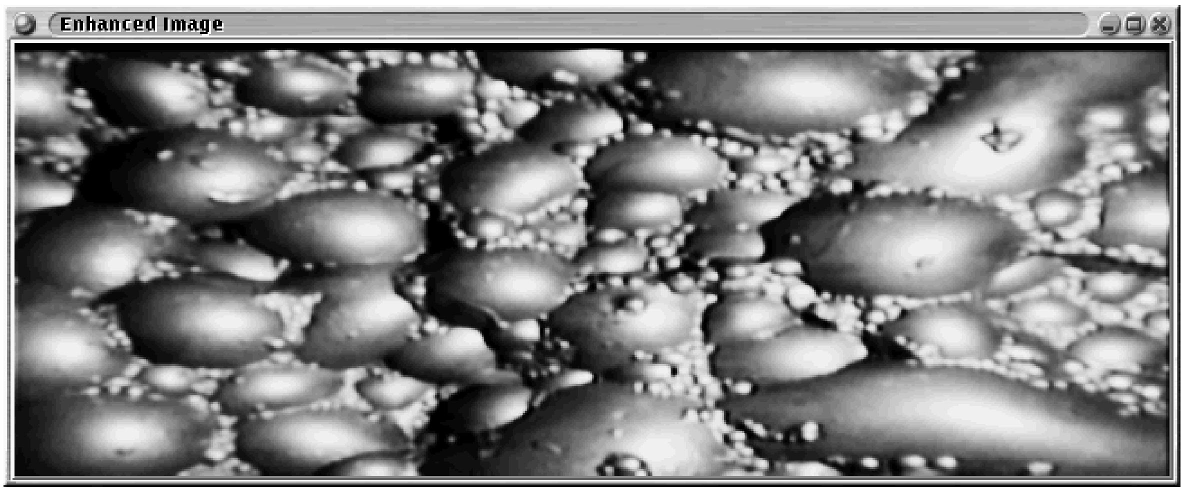
This is a variant of unsharp masking where some of the low-pass channel information is added to the high-pass channel to produce the final result. An edge image (formed by subtracting a low-pass filtered copy of the original from the original) is added to the original.

$$\text{Output Image} = \text{Input Image} + (\lambda - 1)\text{LPF}(\text{Input Image})$$

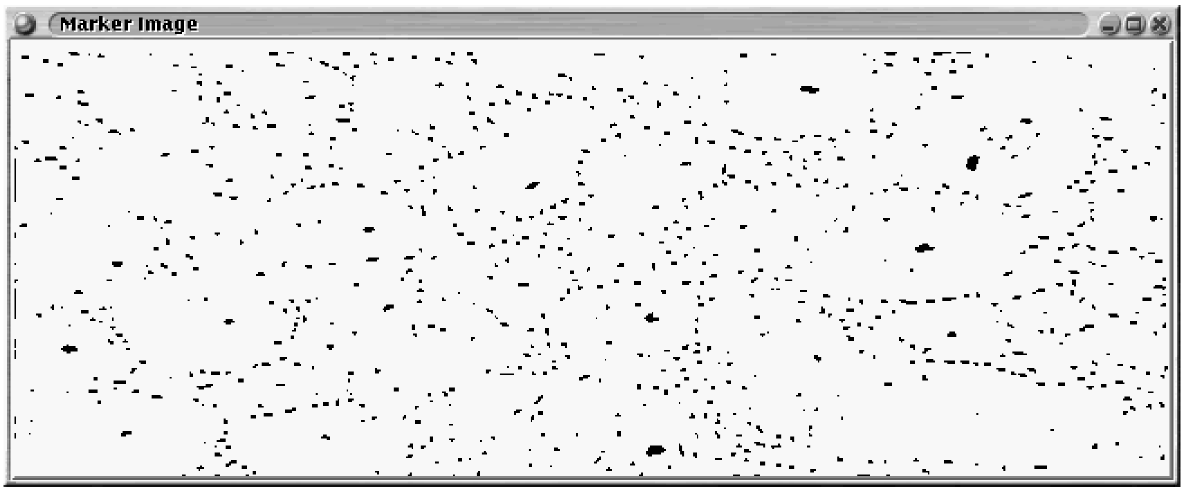
Since edge information is primarily high-pass information, (see figure 11.1(b) generated by subtracting a low-pass filtered image from the original, then histogram equalising) the edges are enhanced [18]. Since the human visual system uses edges extensively [73], the visual appearance of the image



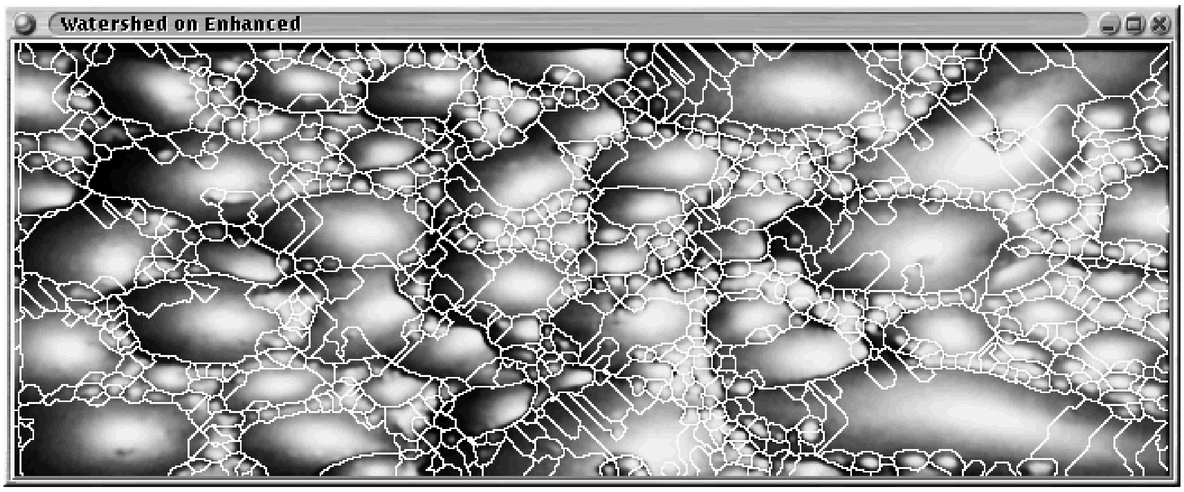
(a) Denoised image.



(b) Enhanced image.



(c) Marker image.



(d) Watershed segmented image.

FIGURE 11.3 Histogram equalisation.

improves. This is potentially useful for watershed segmentation since any enhancement which enhances the ridges, enhances the segmentation. Watershed lines tend to lie on the ridges dividing watershed regions [141].

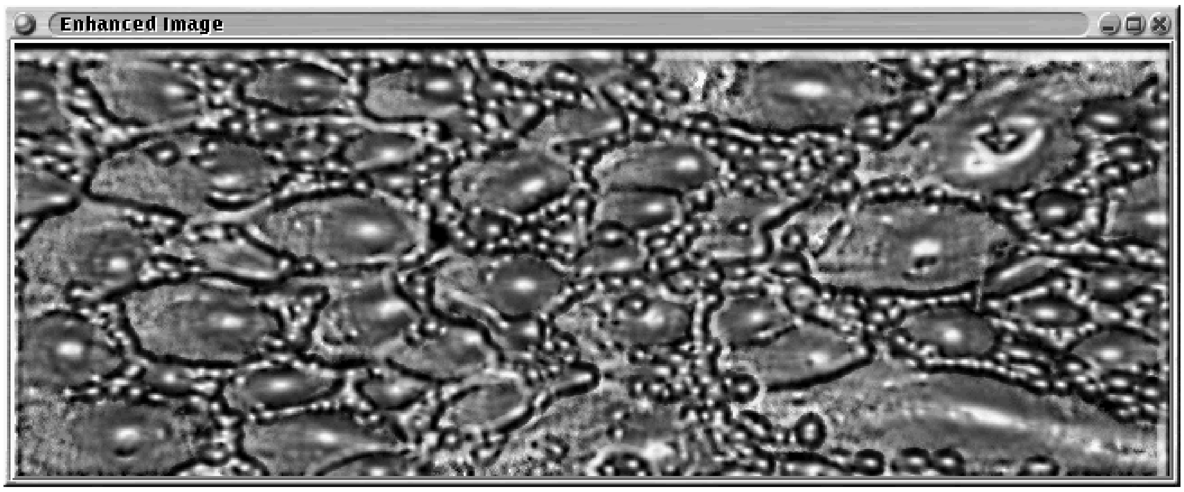
To improve the noise rejection, the images are first denoised by mean filtering with a Gaussian kernel with $\sigma = 1$. The kernel is truncated to a 7×7 pixel footprint. The window size in both adaptive instances was 7×7 pixels. The edge images are then generated as described, with one exception. In order to maintain the full greyscale variation after the edge image is added in, the image is histogram stretched. This results in the average grey-level increasing as the magnitude of the edges increases.

Figures 11.7, 11.8 and 11.9 shows the change in the visual appearance and the watershed segmentation of the enhanced image. The high-pass images are formed by applying a mean filter of sigma 1, 2 and 3 respectively. The enhanced images are formed by adding the high-pass images to the denoised image.

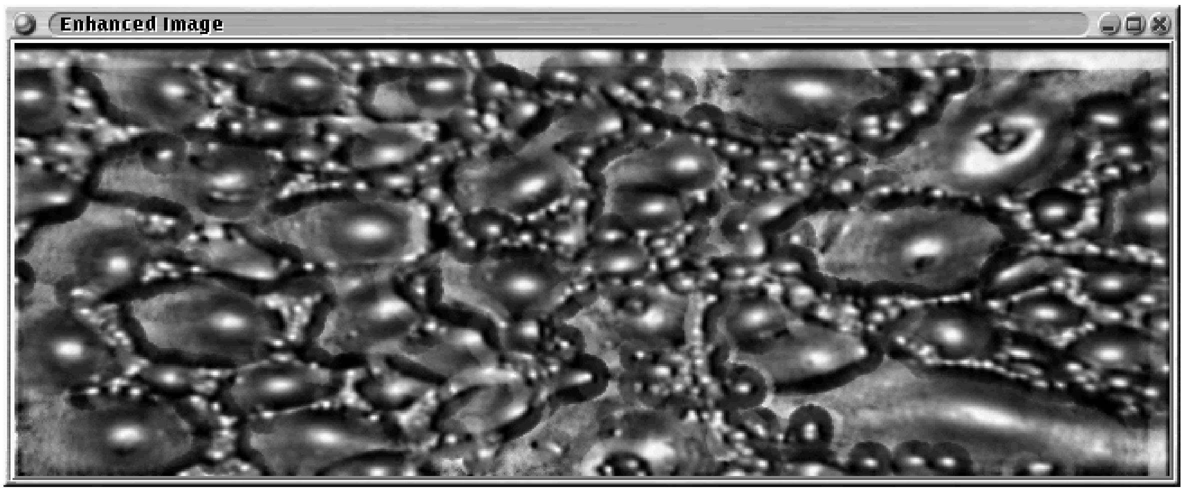
11.3 Baseline subtraction

It is expected that there will be a brightness gradient, or variation, across the image. It is unlikely that the illumination across a flotation cell will be uniform. The assumption will be made that this brightness gradient is smooth and varies slowly. Effectively this requires that the light source does not cast shadows across the flotation cell. A method for extracting this brightness gradient will be suggested. After the brightness gradient is extracted, it is subtracted from the image. The result will ideally be more illumination independent than before.

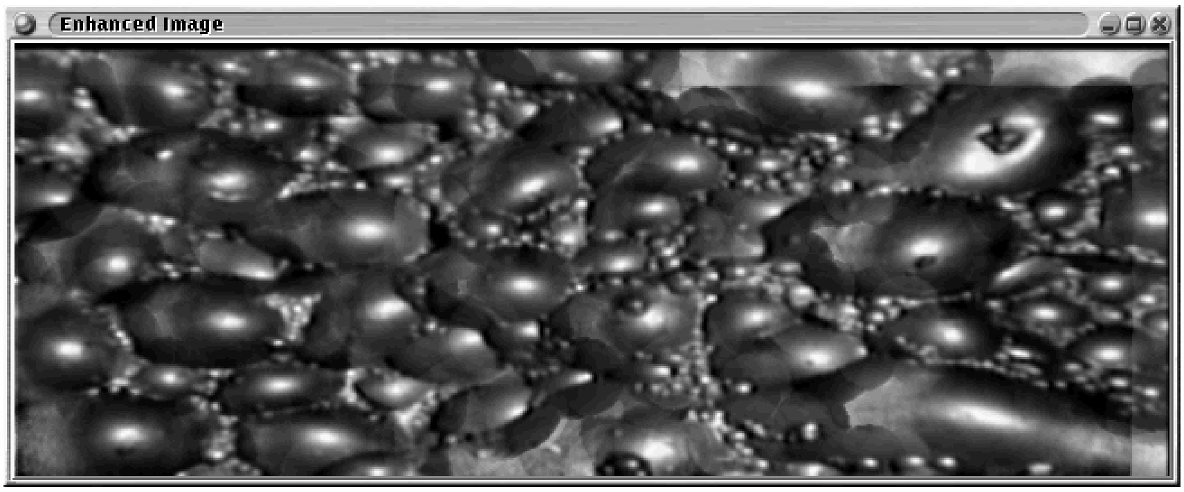
The method used here involves averaging a sequence of images (here 224, or 9.3 seconds worth).



(a) Enhanced image (radius 5).

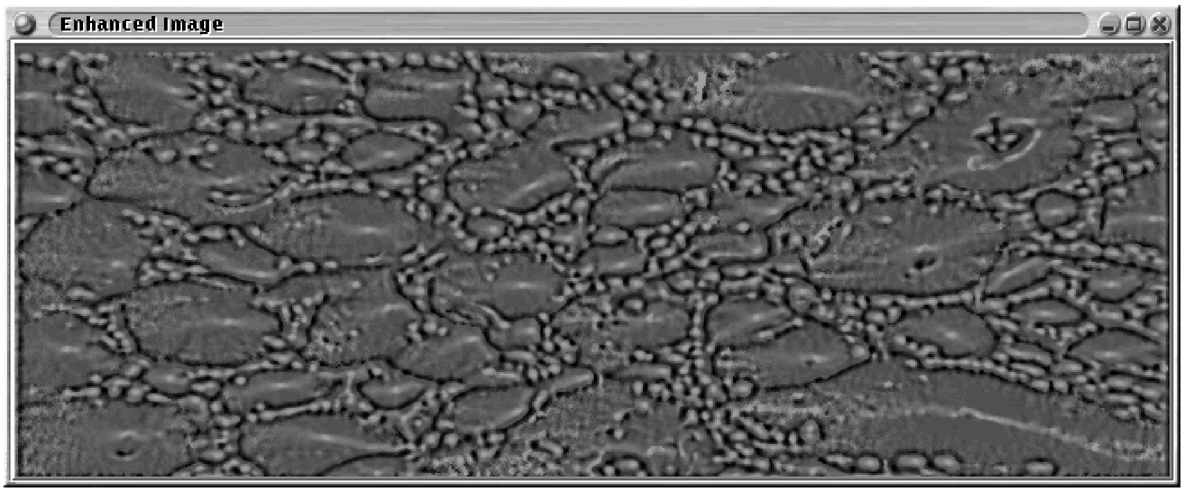


(b) Enhanced image (radius 10).

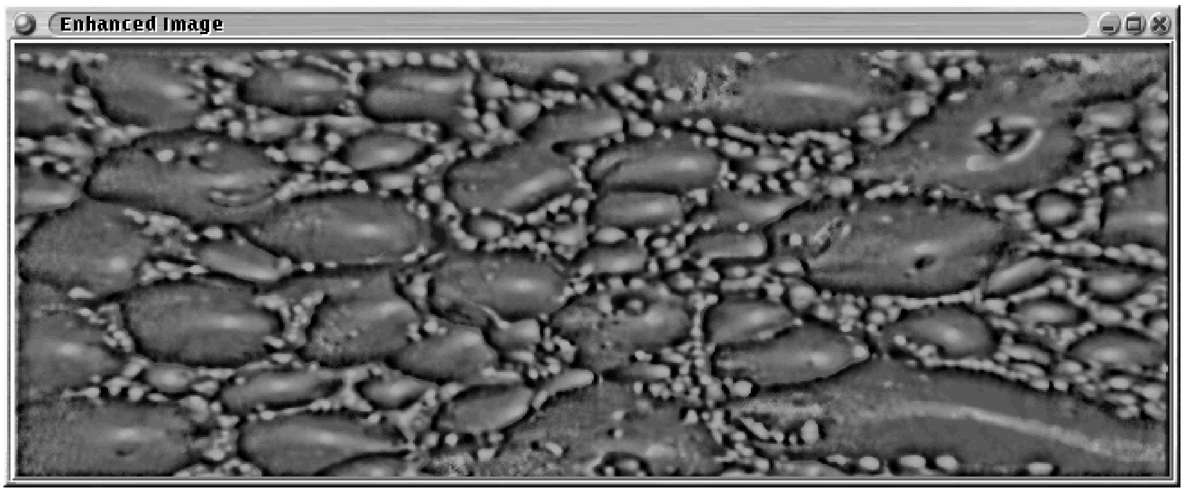


(c) Enhanced image (radius 20).

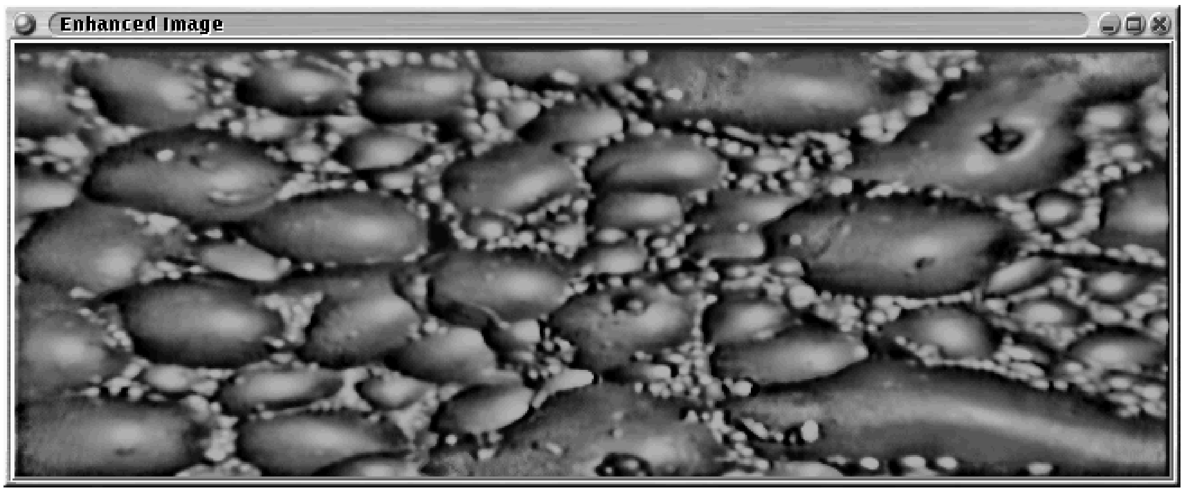
FIGURE 11.4 Adaptive histogram stretching for different radii of local window.



(a) Enhanced image (radius 5).

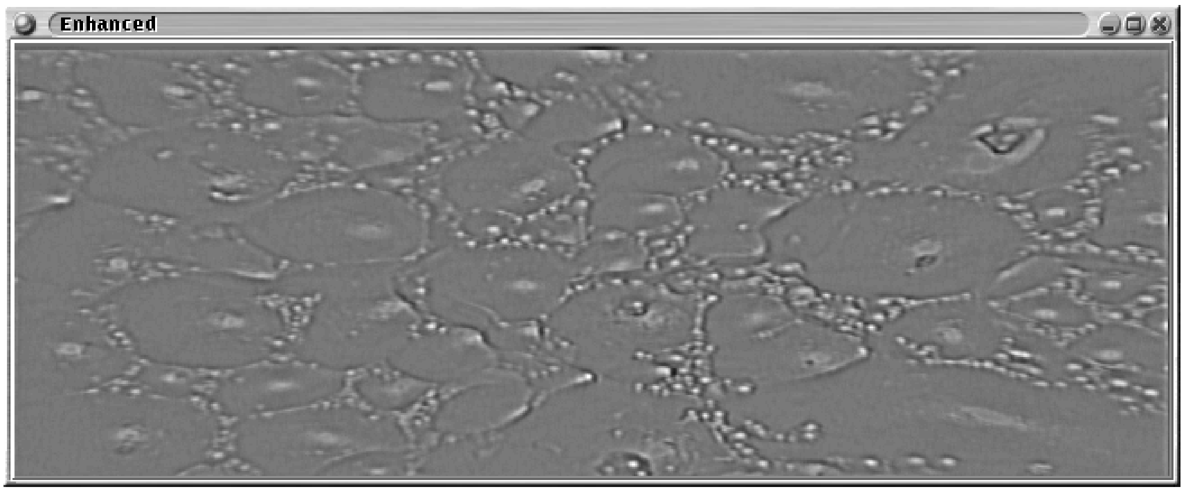


(b) Enhanced image (radius 10).

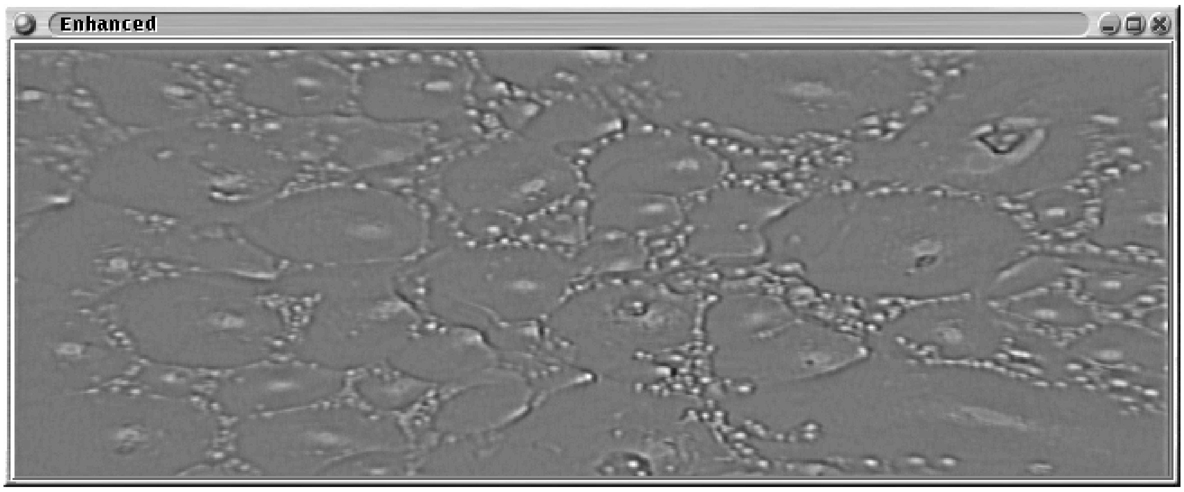


(c) Enhanced image (radius 20).

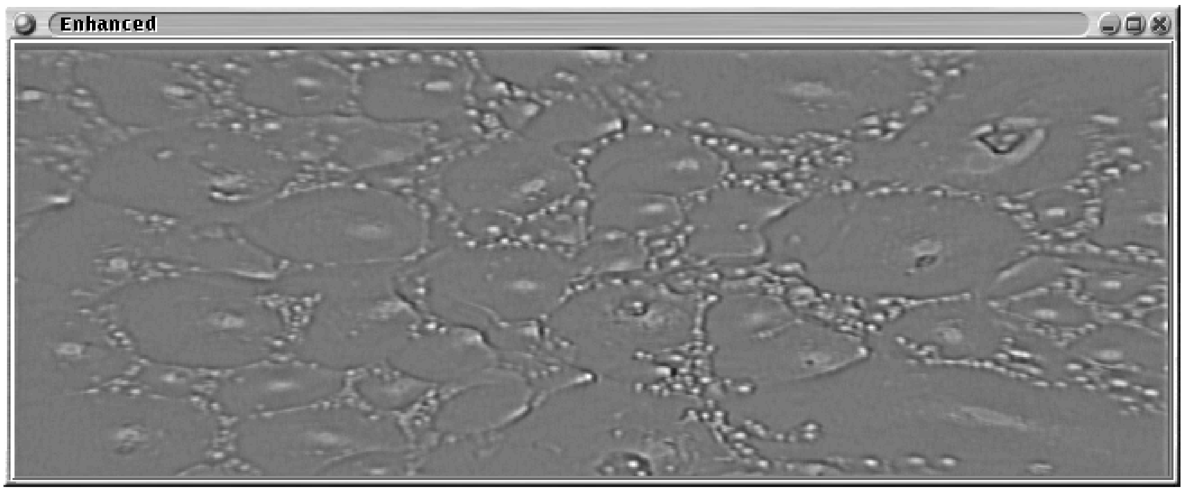
FIGURE 11.5 Adaptive histogram equalisation for different radii of local window.



(a) Unsharp masking, $\sigma = 1$.

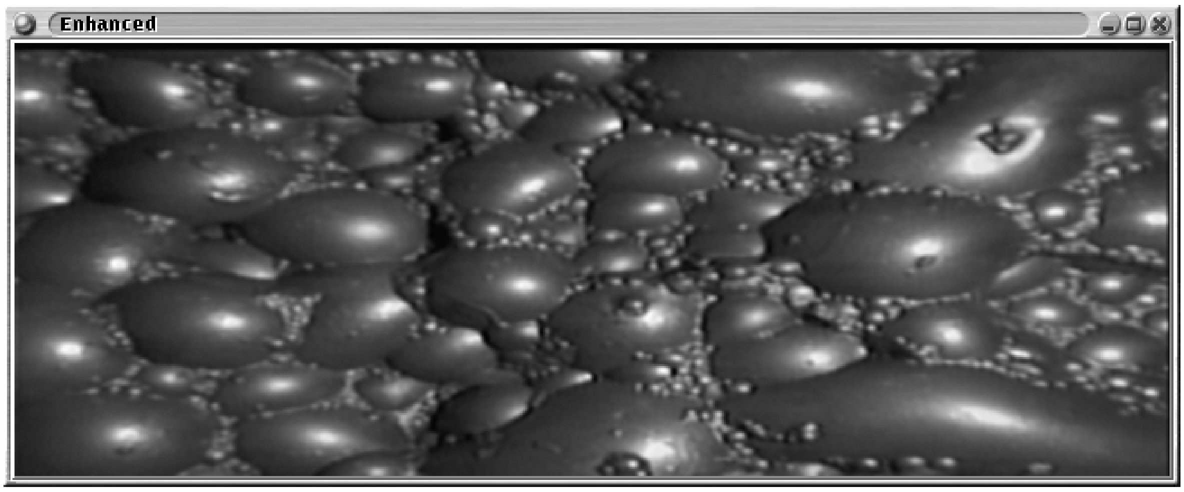


(b) Unsharp masking, $\sigma = 2$.

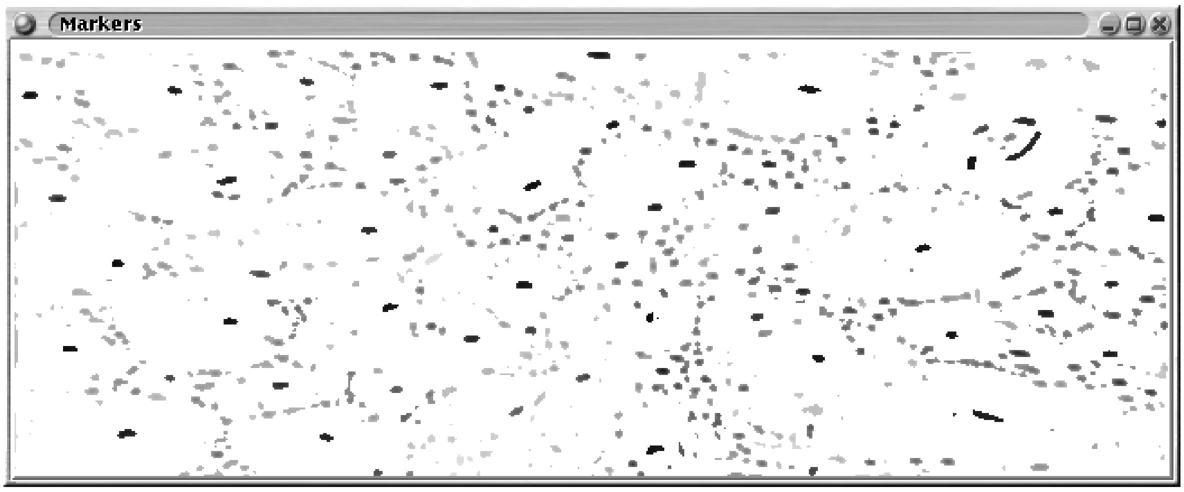


(c) Unsharp masking, $\sigma = 3$.

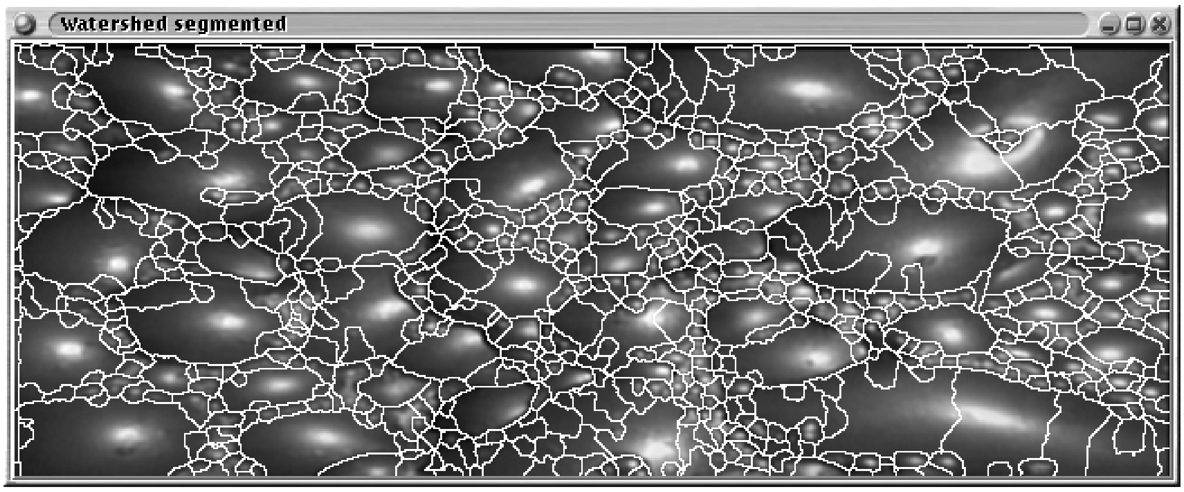
FIGURE 11.6 Unsharp masking.



(a) Enhanced image.

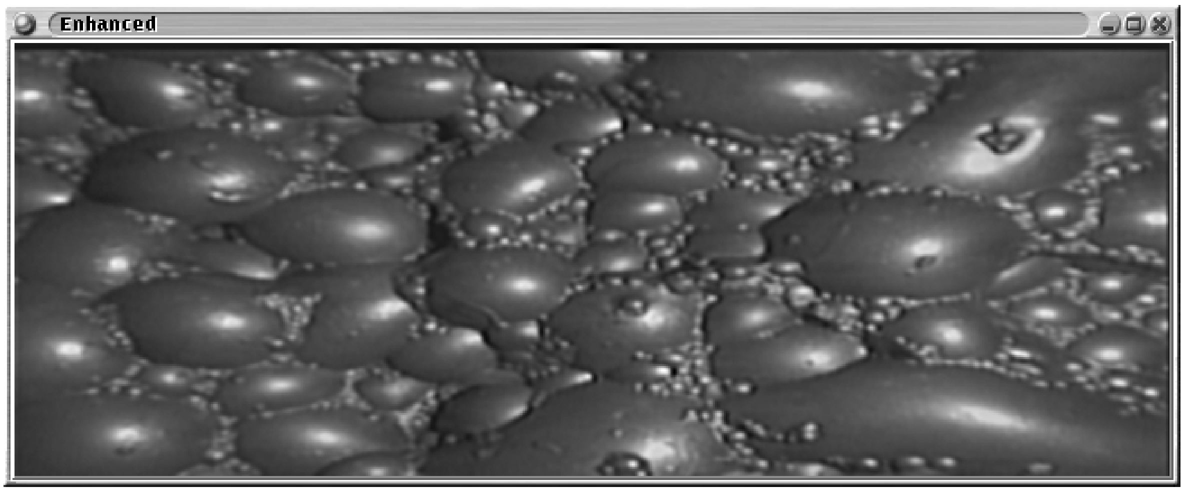


(b) Markers image.

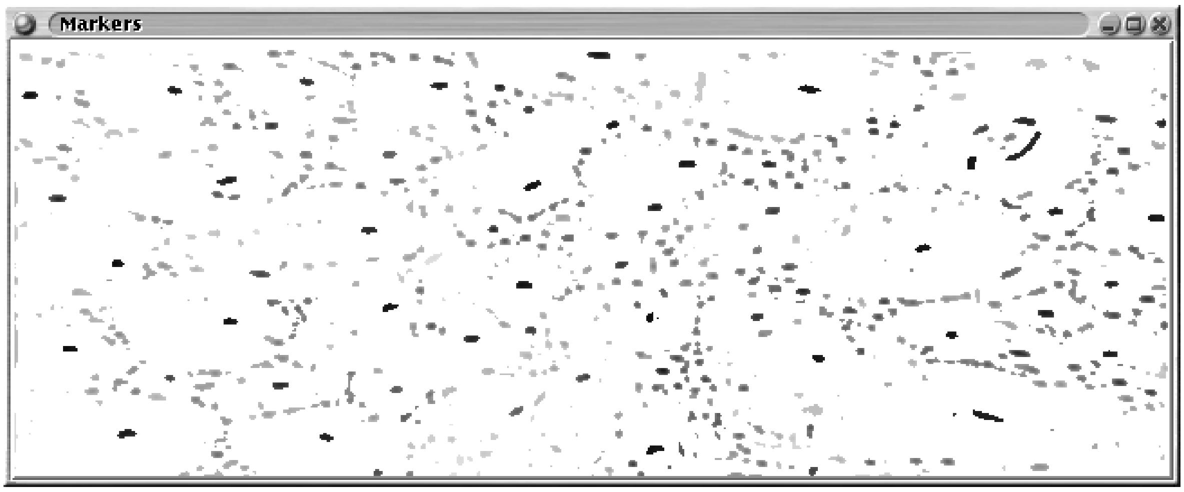


(c) Watershed segmentation.

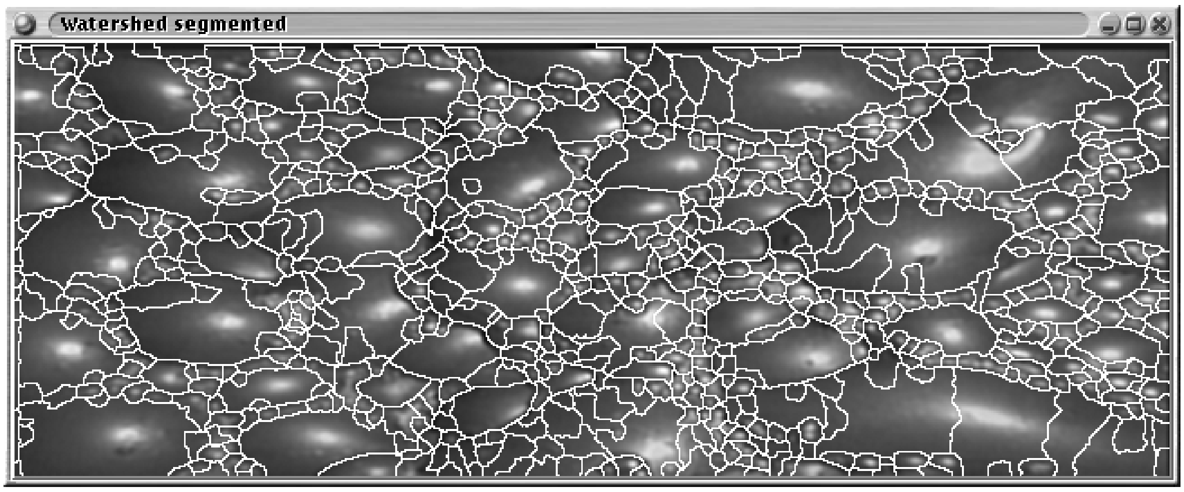
FIGURE 11.7 The effect of edge enhancement on the appearance and watershed ($\sigma = 1.0$).



(a) Enhanced image.

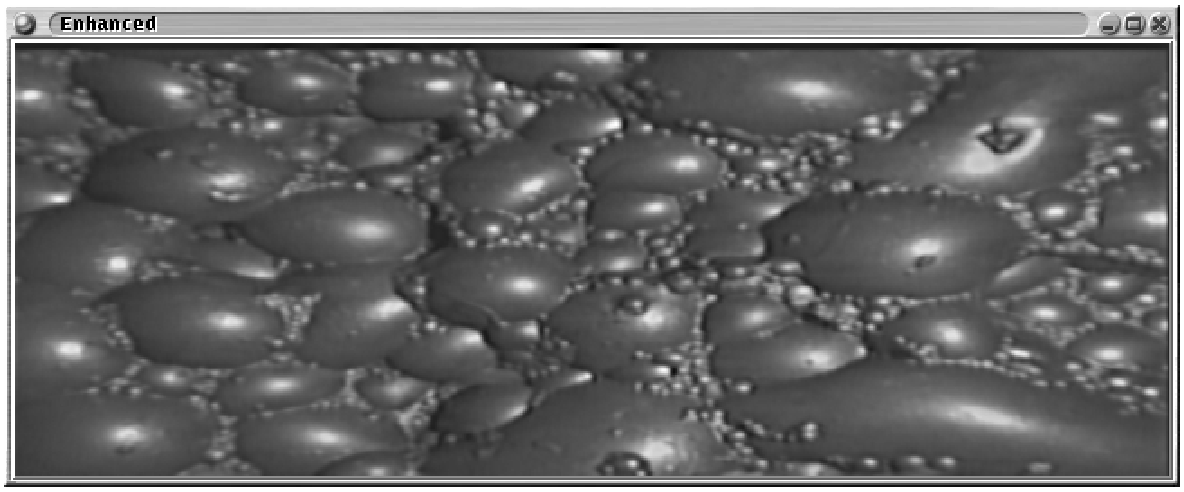


(b) Markers image.

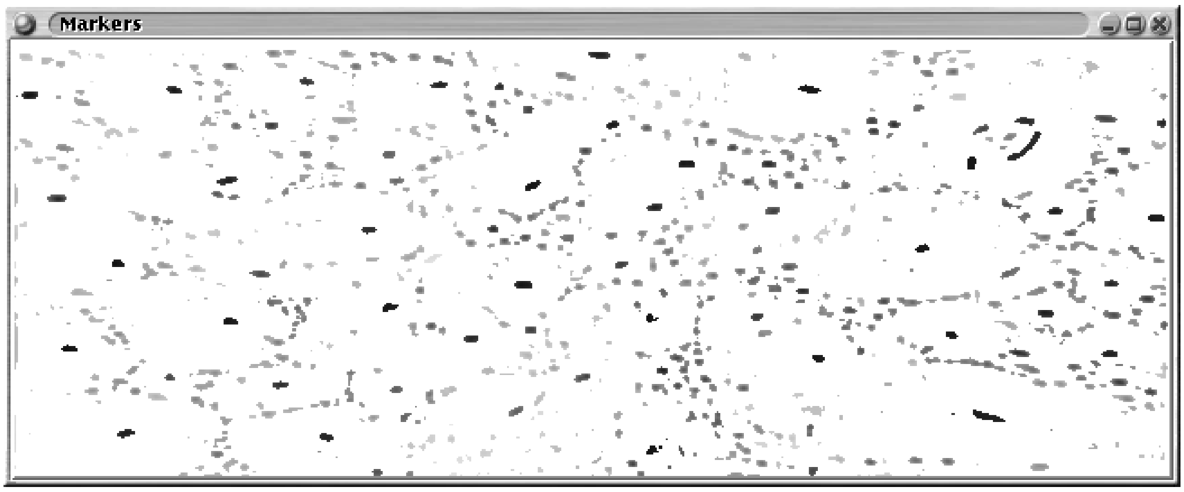


(c) Watershed segmentation.

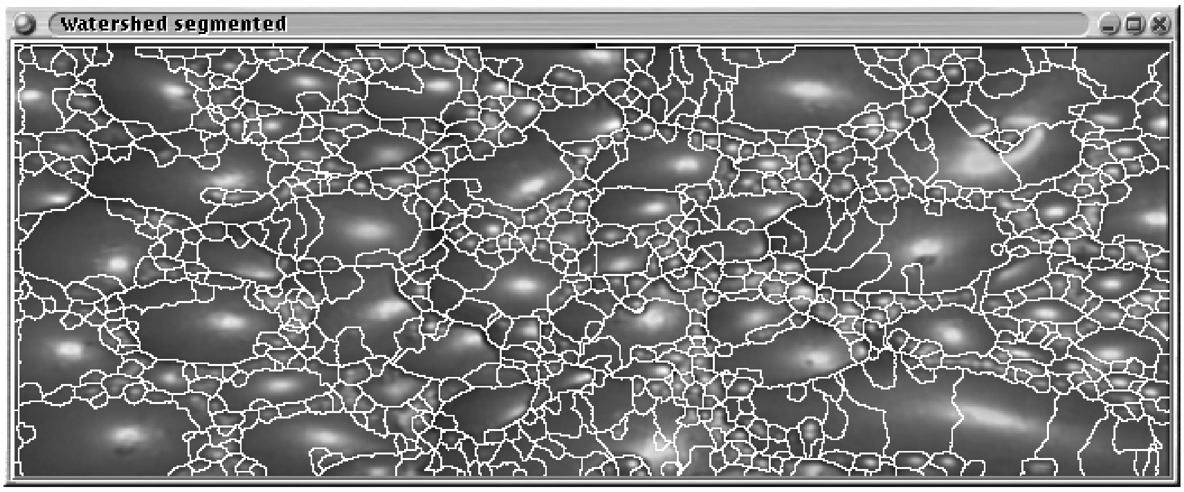
FIGURE 11.8 The effect of edge enhancement on the appearance and watershed ($\sigma = 2.0$).



(a) Enhanced image.



(b) Markers image.



(c) Watershed segmentation.

FIGURE 11.9 The effect of edge enhancement on the appearance and watershed ($\sigma = 3.0$).

Since it is expected that the froth will be in motion during this time, each pixel will be exposed to the full range of intensity variation possible given the illumination at that pixel. Clearly the longer the averaging period the better the results are expected to be. Conversely, the averaging period should not be so long that, for example, natural lighting variations affect the results.

Figure 11.10(a) shows the result of the averaging and figure 11.10(b) shows the effect of subtracting the average from the denoised standard test image. The image is denoised by filtering with a Gaussian kernel of $\sigma = 1$ and footprint 7×7. This limits the effect of noise on the segmentation process. The problem of pixels out of the 0 to 255 intensity range is addressed by linearly rescaling the pixel intensities to lie on and between 0 and 255 (this is identical to the histogram stretching operation as described in section 11.1). Figure 11.11 shows the markers and associated watershed segmentation for the baseline corrected image. Neither show a considerable improvement over the segmentations as seen before.

Close examination of figure 11.10(b) shows while structures which were previously not visible are now obvious, the regions which were previously in shadow show contouring. This contouring is an indication that the pixels in the shadowed regions are effectively of lower precision that the better illuminated pixels. Examination of the illumination gradient (figure 11.10(a)) shows some fine structure. This conflicts with the assumption that the illumination of the froth surface is slowly varying. This indicates that the averaging process did not continue for long enough.

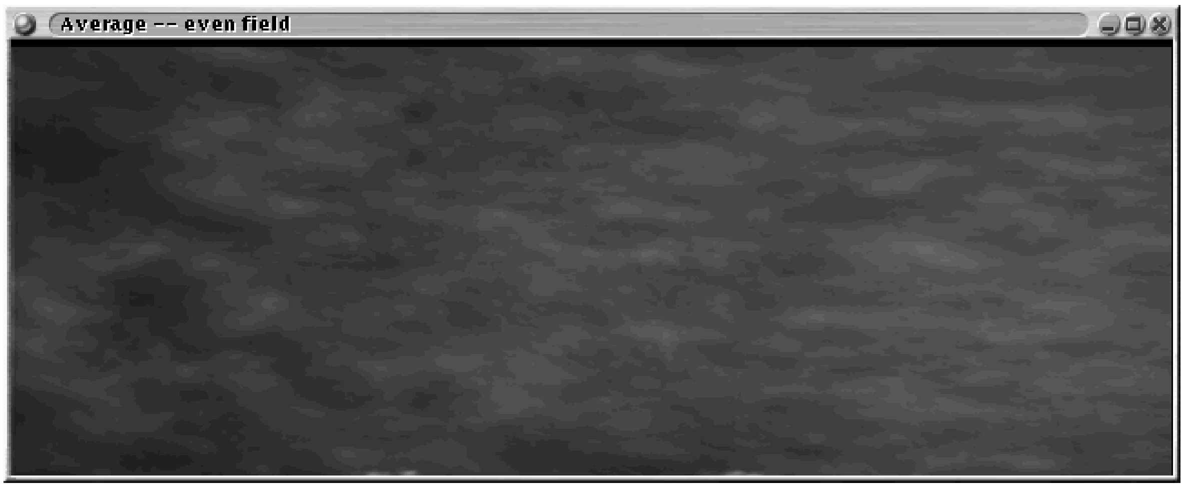
On the other hand, baseline correction does improve the visual appearance of the froth images. Specifically, more detail is visible in the shadowed areas. The watershed and marker images (figure 11.11) do not show any marked differences from the reference watershed and marker images (figure 11.1). This is to be expected. The size of the markers is small relative to the structures in the average image (figure 11.10(a)), this implies that these will survive the baseline subtraction process (the intensities of these markers may well change).

11.4 Homomorphic filtering

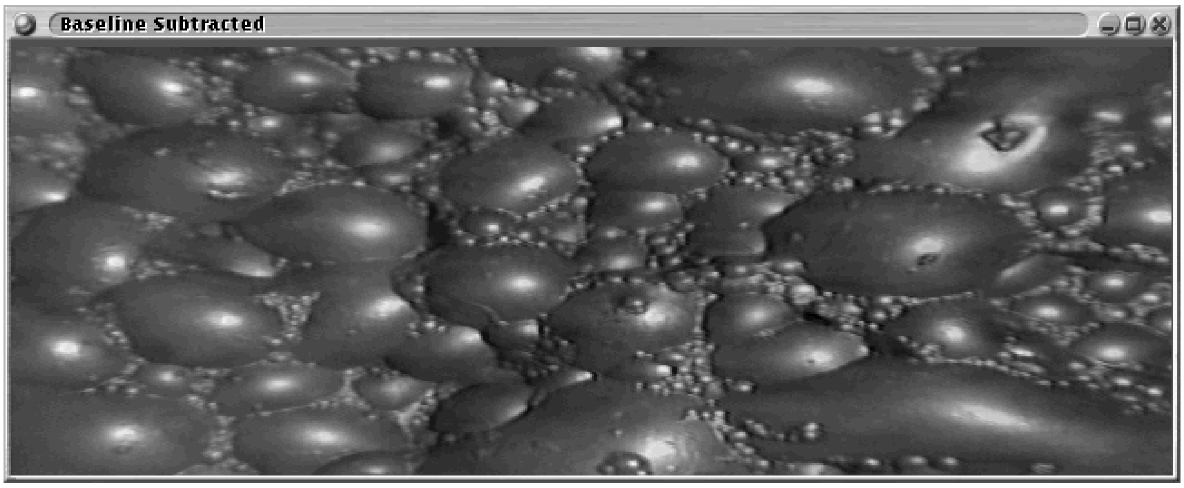
Homomorphic filtering pre-supposes that an image can be described as the product of a reflectance map and a point by point illumination function [73, 153][74][pg. 219, 315-316]. The usual simplifying assumption is that the illumination is a single point source far from the object(s) illuminated. The illumination is assumed to be smooth and the reflectance function complicated and busy. The image is modelled as the product of the illumination and reflectance map.

$$\text{Image}(x, y) = \text{Illumination}(x, y)\text{Reflectance}(x, y)$$

Conceptually, illumination invariance is achieved by separating the illumination function from the reflectance function and then setting the illumination part to a constant, then reconstructing. The option exists to completely remove the illumination component and only reconstruct the reflectance component. Traditionally this involves taking the logarithm of image intensities and filtering in the

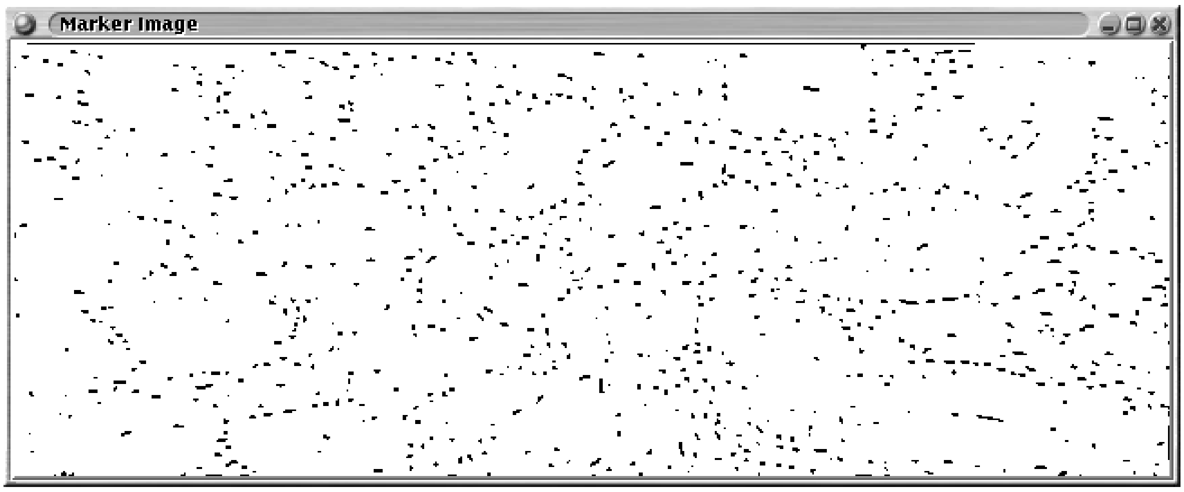


(a) The brightness variation.

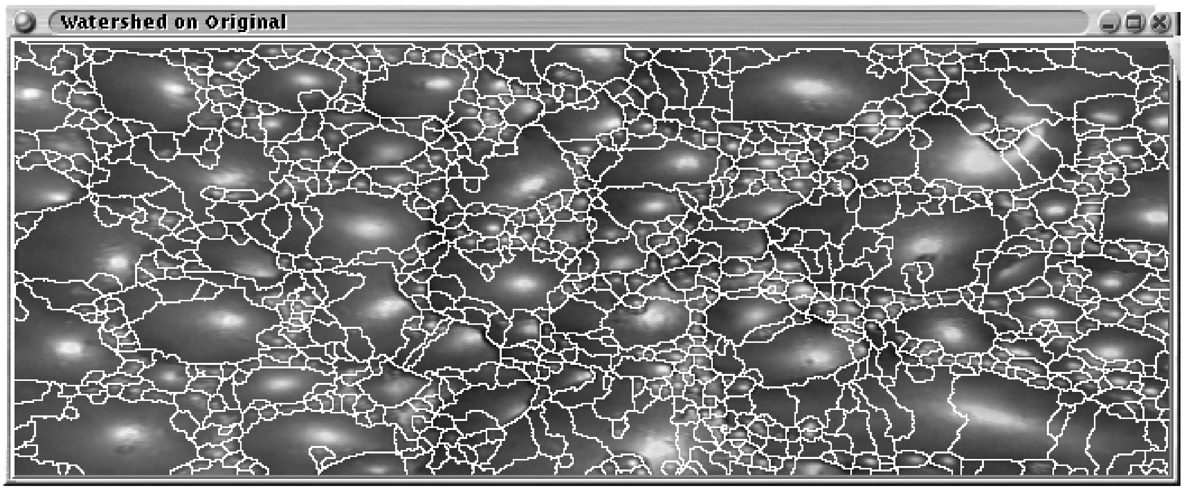


(b) After baseline subtraction.

FIGURE 11.10 Baseline subtraction.



(a) Markers.



(b) Watershed segmentation.

FIGURE 11.11 Baseline subtraction.

logarithmic domain. Reconstruction, naturally, involves taking the exponent of the logarithm domain values. Zero intensity pixels are handled by adding a small integer constant before taking logarithms and subtracting the constant after reconstructing.

Figure 11.12 shows the edge image, or reflectance function, after the illumination function has been removed from the input image. Prior to homomorphic filtering, the test image is denoised by convolving with a Gaussian kernel of $\sigma = 1$, and footprint 7×7. The low-pass channel of the homomorphic filter process is generated by convolving with a separate Gaussian kernel. The high-pass channel is formed by subtracting the low-pass channel from the logarithmic transformed image. The reflectance images are histogram stretched after reconstruction to improve appearance. Figure 11.13 shows the markers used. Figure 11.14 shows the watershed segmentation, using the given markers, on the reflectance images. The enhanced images shown represent the extreme case of all the low-pass information in the logarithmic domain suppressed. The watershed segmentation can then be made to vary between the reference segmentation (see figure 11.1) and the watersheds shown in figure 11.14 by scaling the low-pass channel down, and scaling the high-pass up [74].

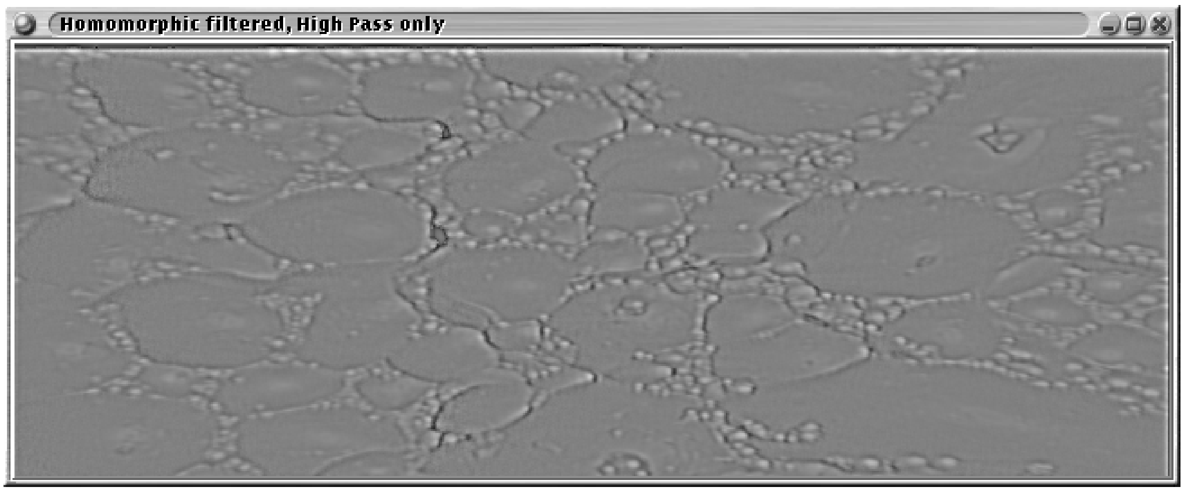
11.5 Ridge enhancement

The process of watershed segmentation involves flooding from the minima of an image. The flooding process gradually continues until the flood basins for the various minima encounter each other. Where they do, flooding at that pixel stops. The process continues until each pixel in the image has been assigned to a minimum, or if a pixel is equidistant from two minima the pixel is labelled as a watershed pixel. [13, 31, 141]. For the purposes of froth image segmentation, the inverted froth image is flooded.

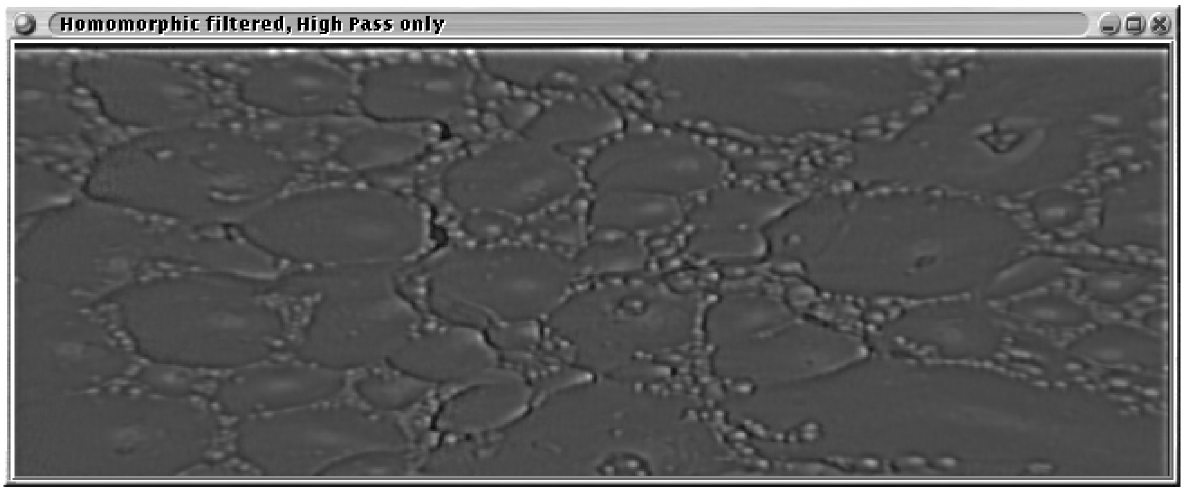
Given this summary of the watershed segmentation process, it is apparent that the ridge pixels of an image play an important role. The ridge pixels are local maxima which separate watershed basins. If one sees a grey scale image as a topographical height map, then a drop of water on one side of the ridge flows into one watershed basin, and a drop of water on the other side of the ridge flows into a different watershed basin. The ridge pixels then divide the watershed regions.

A potential enhancement technique then extracts these ridge pixels and then increases the height of these pixels. Consider a small bubble on a large bubble. If the ridge separating the small from the interior of the large bubble is high. then most, if not all, of the large bubble will be flooded before the small bubble hollow overflows. This will allow for the delineation of the small bubble within the larger, without a portion of the larger bubble being incorrectly assigned to the smaller.

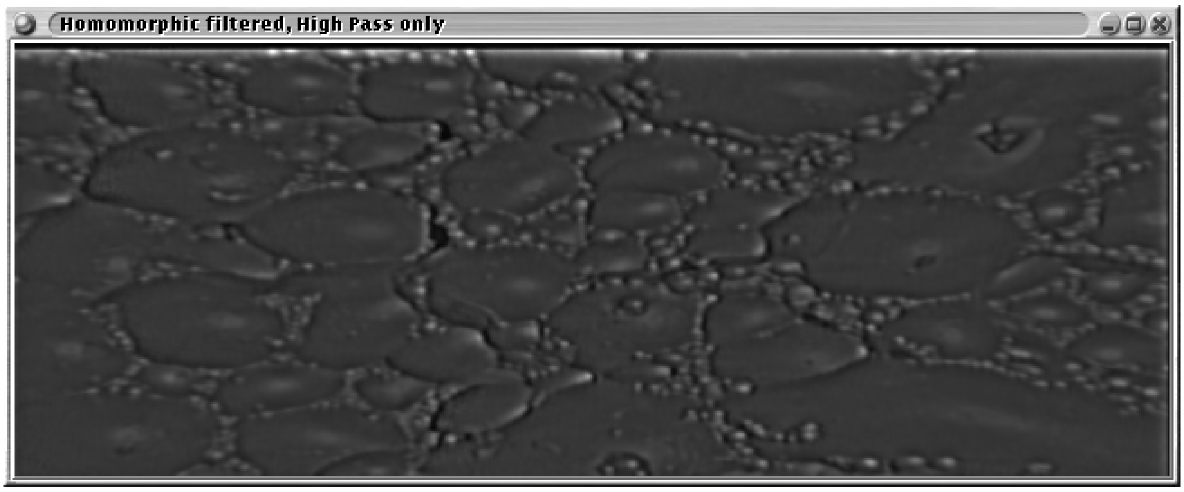
In order to simplify the ridge extraction process the image will be denoised first. Three techniques for ridge extraction will be considered, opening using a line structuring element, opening using a rolling ball, and greyscale line reconstruction. The pixels removed by the opening are likely the regional maxima of the image (see Appendix B for properties of openings). A proportion of these pixels will be ridge pixels, the remainder will be isolated maxima (peaks). The peaks and isolated maxima will be extracted by writing to an output image only those pixels which are different when



(a) Enhanced image, $\sigma = 1$.



(b) Enhanced image, $\sigma = 2$.



(c) Enhanced image, $\sigma = 3$.

FIGURE 11.12 Enhancement via Homomorphic filtering.

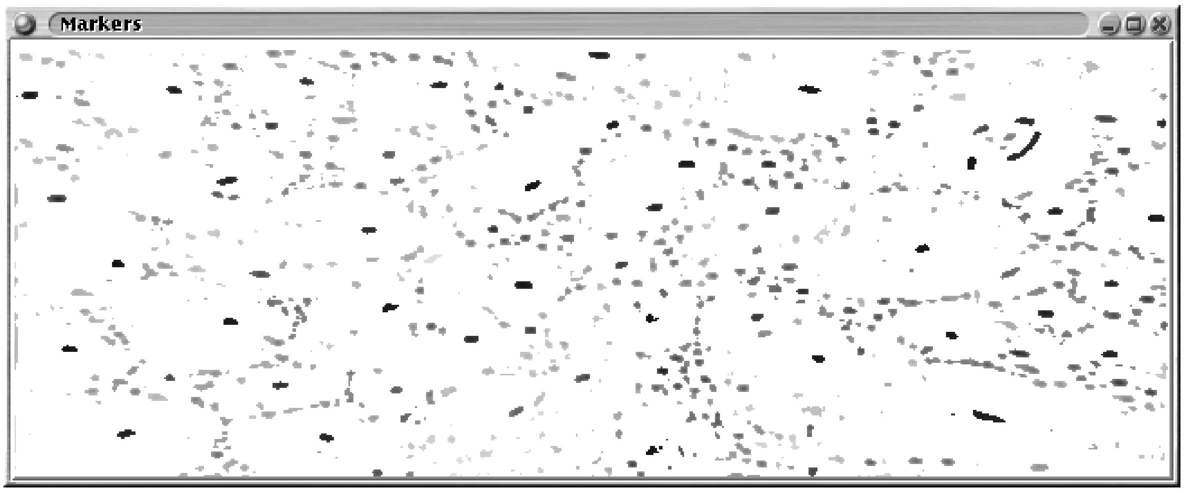


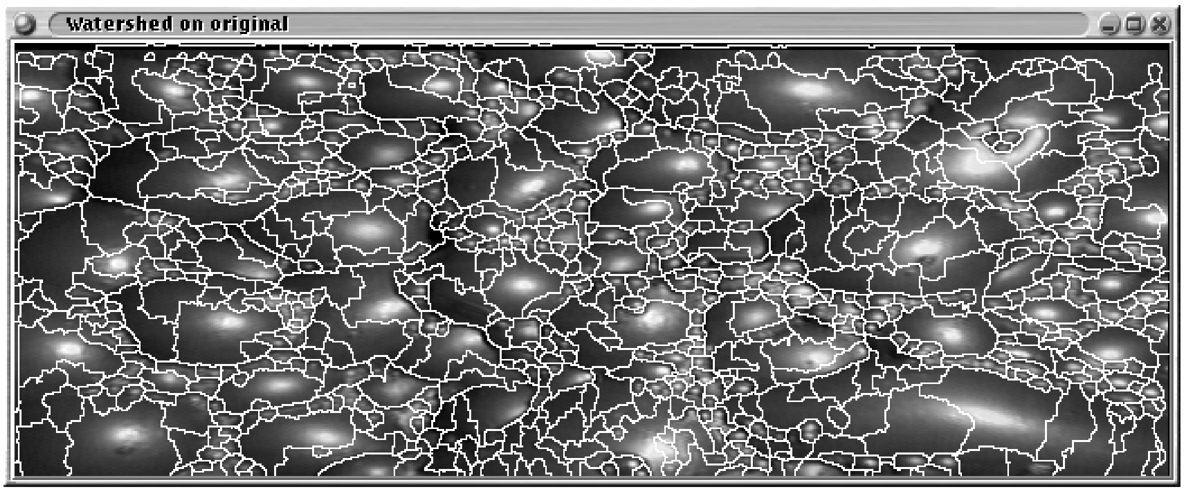
FIGURE 11.13 Markers for segmenting Homomorphic enhanced images.

comparing the opened image and the denoised image.

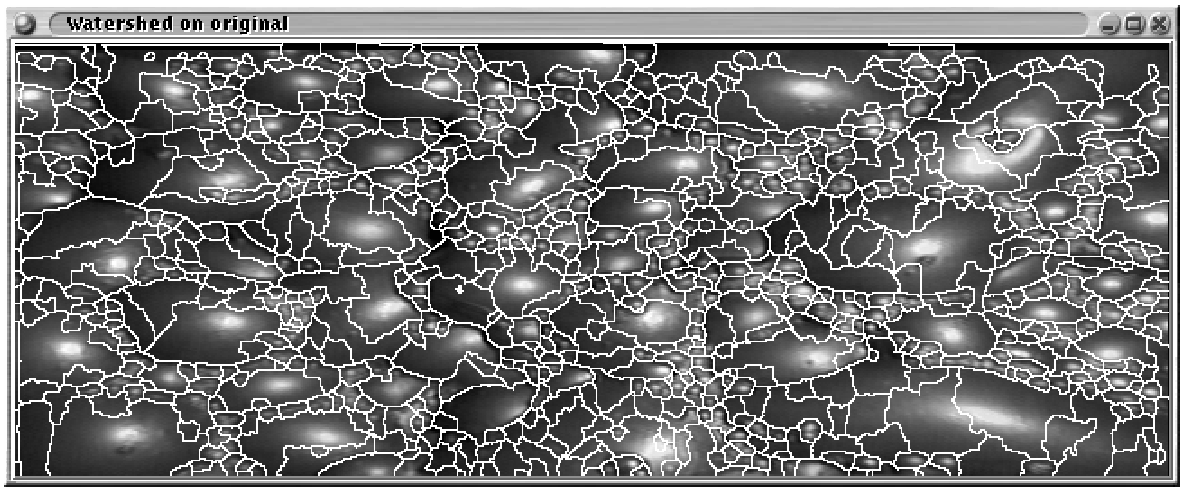
11.5.1 Line openings

Line openings are openings where the structuring element is a line of pixels, for example, a single row or column of pixels of a certain length. Line openings in the horizontal, vertical and diagonal (at 45^o and 135^o) directions will be considered. Other diagonal directions are possible but difficult to approximate using small structuring element sizes. Openings are formally defined in Appendix B.

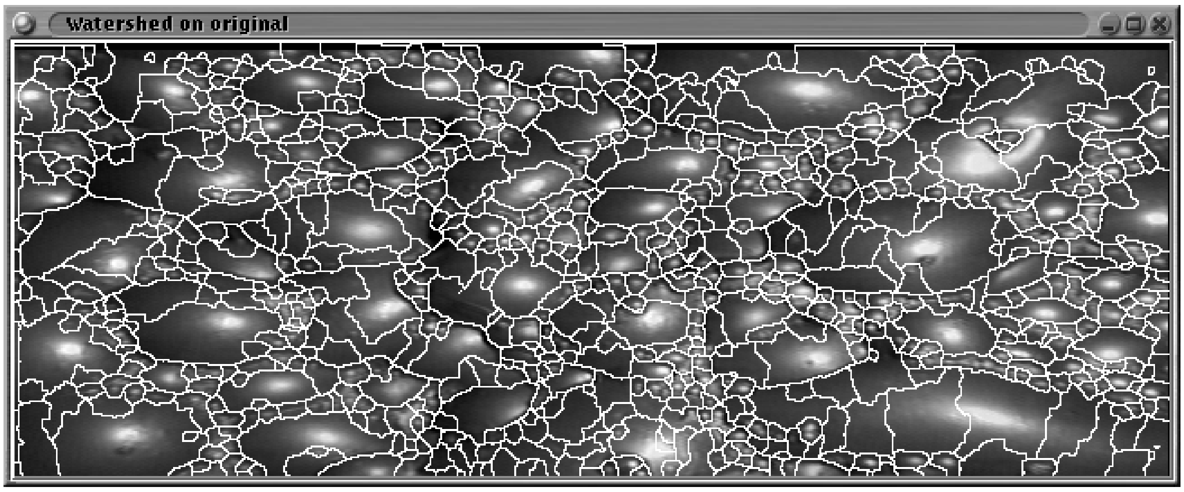
Figure 11.15 shows ridges, enhanced image and watershed segmentation. The markers used are the markers of the denoised image. The denoised image is generated by convolving with a Gaussian kernel of $\sigma = 1$ and footprint 7×7. The ridge image is computed from the inverted denoised image. The final ridge image is the constructed by combining the line reconstructions in the four direction specified. Note that the ridges found do not completely enclose the bubbles. The ridge image is cleaned by removing all ridge objects smaller than 30 pixels. This removes spurious ridge pixels due to noise and isolated local maxima (peaks). The parameter is found by investigation and provides a tradeoff between cleaning and retaining valid ridge pixels. The final segmentation is not very different from the reference segmentation (see figure 11.1). If a ridge does not completely enclose a bubble, the bubble is unlikely to be properly segmented. For the case where small bubbles occur on large bubbles, this is especially noticeable. Longer length lines are not considered, since longer line lengths produce wider ridges which are not desirable. Wider ridges tend to cover and reduce the size and magnitude of small bubble highlights in the enhanced image, this is undesirable since the highlights of bubbles are used as markers for the segmentation process.



(a) Segmented image, $\sigma = 1$.

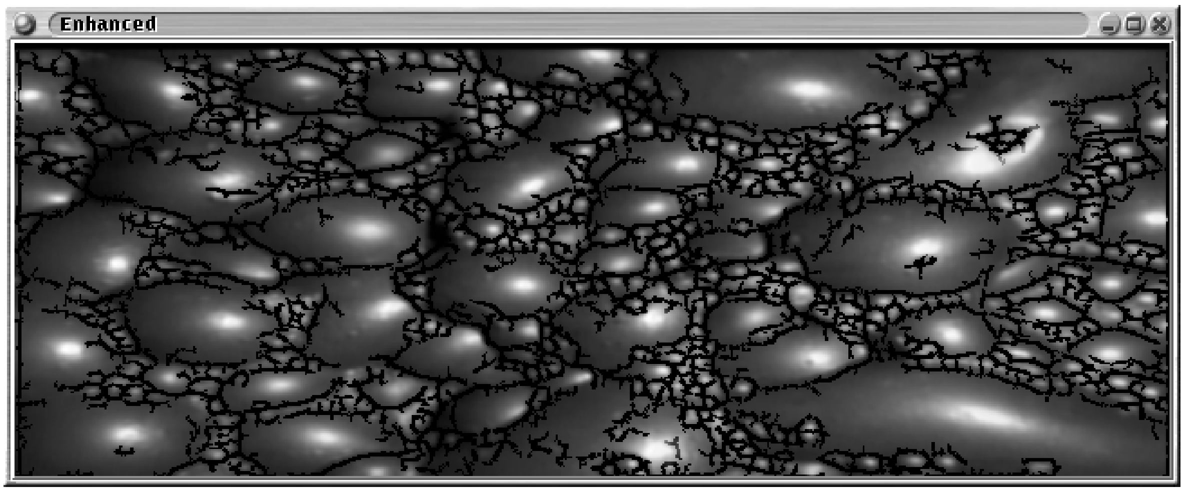


(b) Segmented image, $\sigma = 2$.

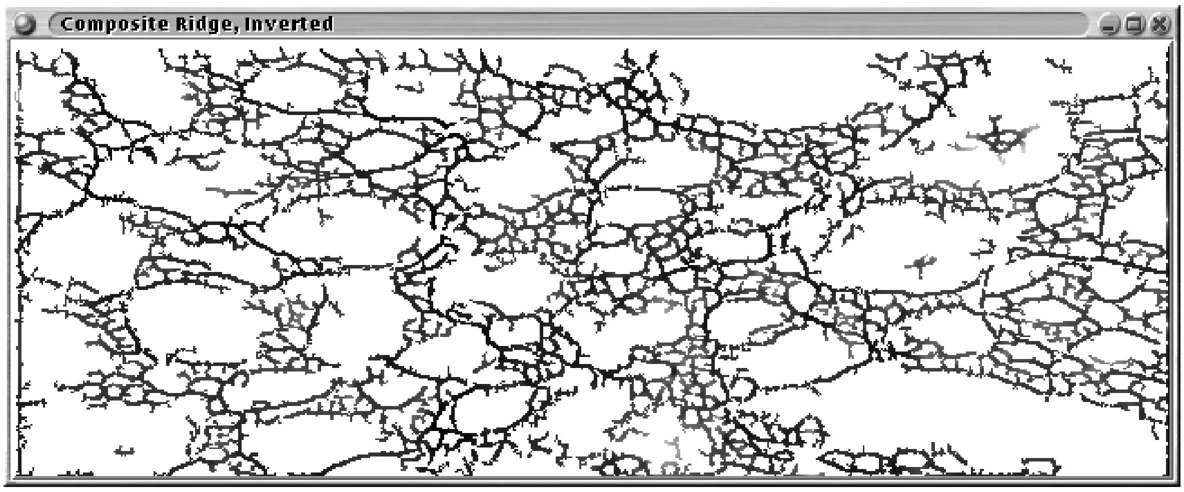


(c) Segmented image, $\sigma = 3$.

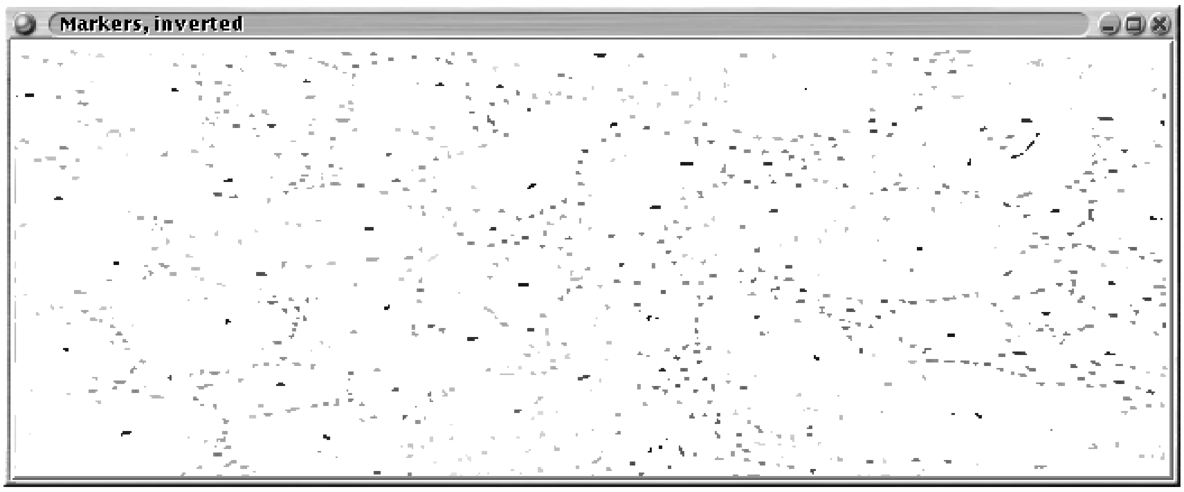
FIGURE 11.14 Segmenting Homomorphic filtered images.



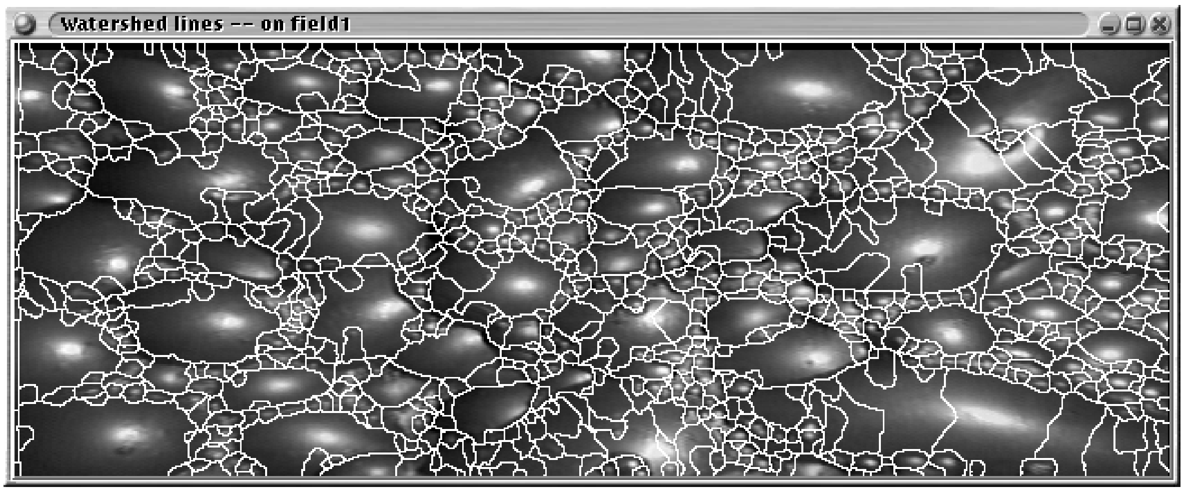
(a) Enhanced image.



(b) Ridges



(c) Markers



(d) Watershed segmented image.

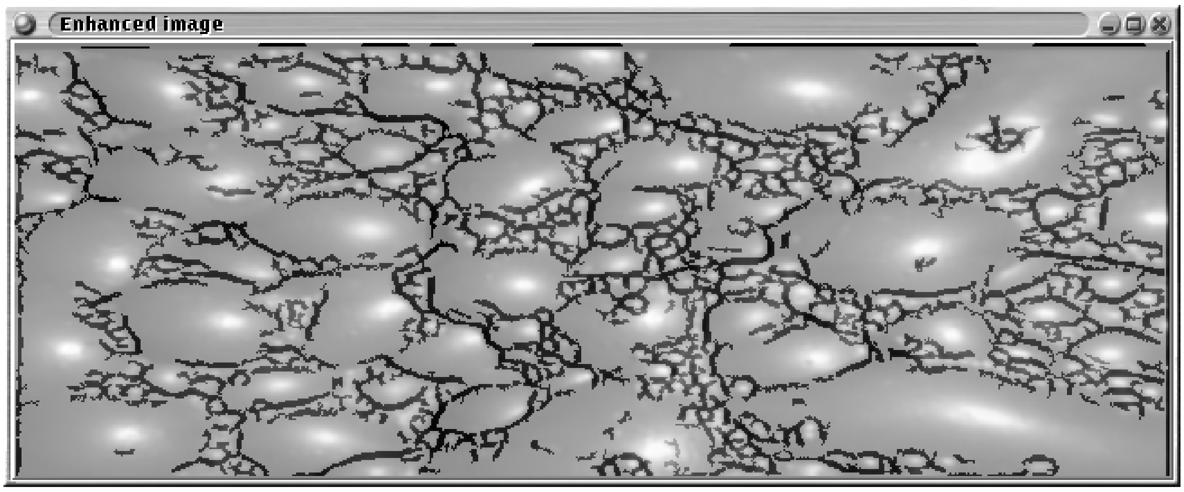
FIGURE 11.15 Ridges from line opening, line length 3.

11.5.2 Rolling ball openings

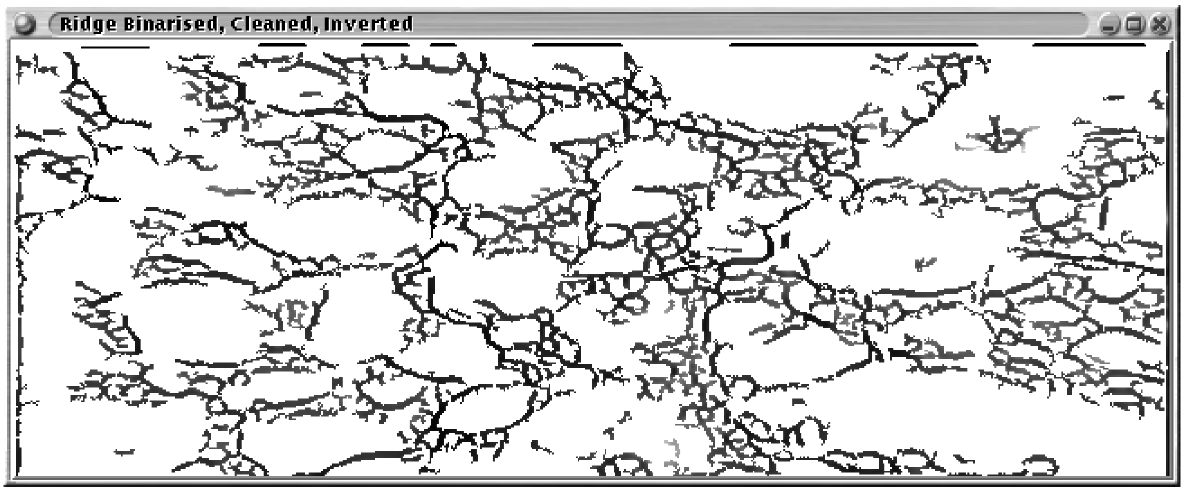
A variety of rolling ball diameters will be considered. Openings are formally defined in Appendix B. Figure 11.16 shows ridges, enhanced image and watershed segmentation. The markers used are the markers of the denoised image. The denoised image is generated by convolving with a Gaussian kernel of $\sigma = 1$ and footprint 7×7. The ridge image is computed from the inverted denoised image. The final ridge image is the constructed by subtracting the opened image from the denoised. The opened image is generated by a rolling ball opening of radius 2. Larger radii generate wider ridges which obscure small bubbles in the enhanced image. This reduces the small bubble highlights which complicates segmentation of the small bubbles. Note that the ridges found do not completely enclose the bubbles. The ridge image is cleaned by removing all ridge objects smaller than 30 pixels. This removes spurious ridge pixels due to noise and local maxima. The parameter is found by investigation and provides a tradeoff between cleaning and retaining valid ridge pixels. The final segmentation is not very different from the reference segmentation (see figure 11.1).

11.5.3 Line reconstructions

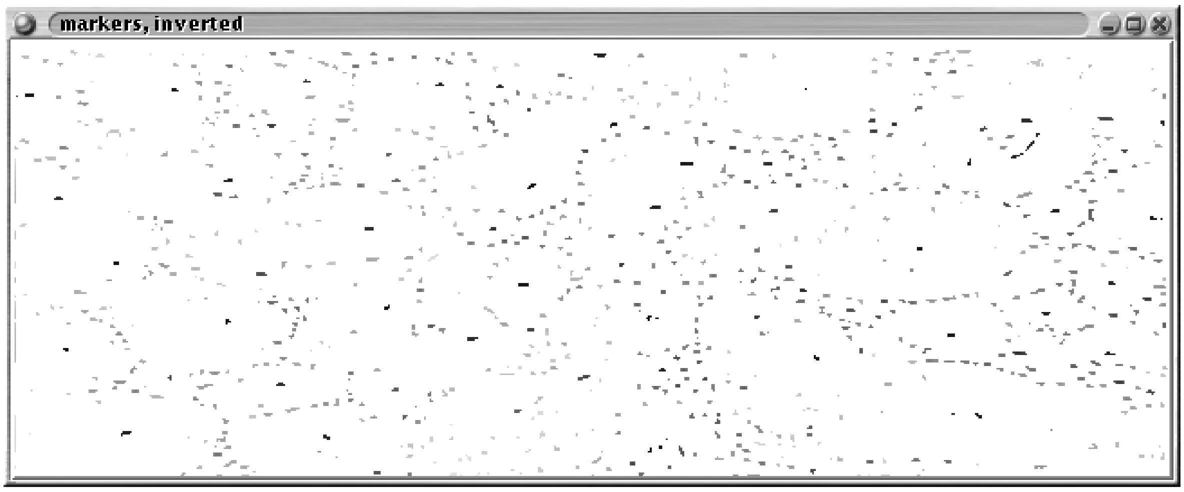
Greyscale reconstruction by dilation, followed by dome extraction, is favoured for extracting local minima of images [144]. If the crest of the ridge is everywhere of the same height, then the domes of the image include the ridge. In the terminology of greyscale reconstruction, a marker image is everywhere less than or equal to the mask image (see section 4.1). The mask image is is taken to be the unprocessed original. The dome image is the difference between the mask and the reconstructed image. For display purposes, pixels on the dome image larger than zero are remapped to the corre-



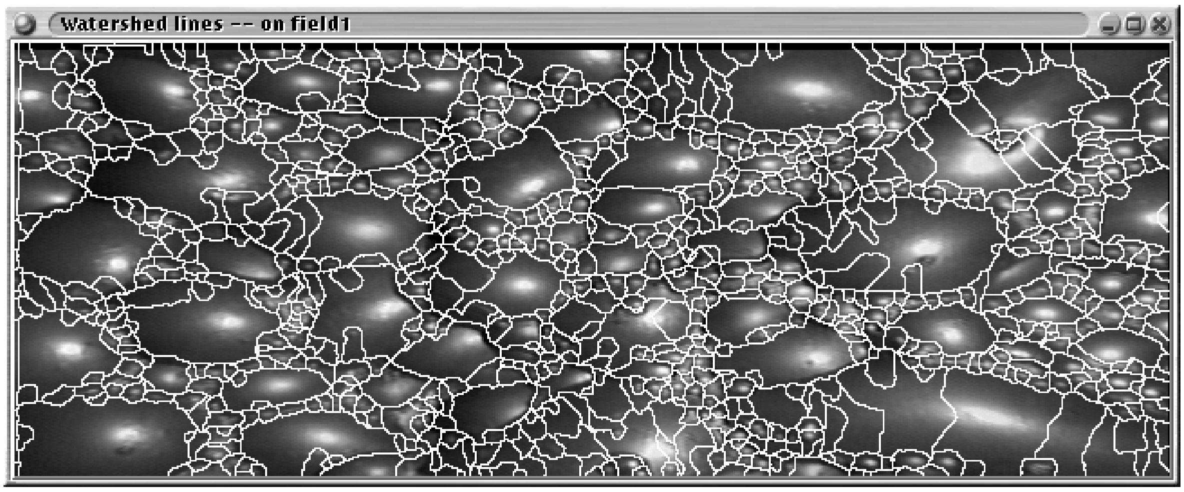
(a) Enhanced image.



(b) Ridges



(c) Markers



(d) Watershed segmented image.

FIGURE 11.16 Ridges from Rolling Ball opening, radius 2.

sponding values on the mask image.

Greyscale reconstruction by dilation is an extension of binary conditional dilation to greyscale images. The definition used by Vincent [144] is,

$$I^{k+1}(x,y) = \max(I^{k}(x,y) \oplus \mathrm{Sel}(x,y),\ \mathrm{Mask}(x.y)) \tag{11.2}$$

I^0 is initialised to the marker image, $\oplus$ denotes morphological dilation. Sel is a structuring element (usually a 3×3 flat structuring element). Mask denotes an image which constrains the intensities of the reconstructed image by placing an upper bound on the pixel intensities (see Appendix B) for an introduction to mathematical morphology). The process repeats until stability. It should be noted that faster and more sophisticated algorithms for greyscale reconstruction do exist (see [144] for some examples).

Consider a ridge running from the top to bottom in an otherwise constant intensity image. Let the ridge slowly decrease in height from the top to the bottom of the image. If one creates a marker image by shifting the ridge image down in intensity by some small constant, and then greyscale reconstructs by dilation this mask image under the ridge image, one does *not* extract the entire ridge.

Consider the crest line of the ridge on the mask and marker images. Flat greyscale dilation of the line, increases the intensity of the line, but the largest value on the line is as before. If one considers the crest line as a ramp, the top of the ramp becomes wider and the wedge portion of the ramp is shifted horizontally. Conceptually, this continues until the marker crest runs into the mask crest, the greyscale reconstruction by dilation stops. Clearly the subtracting the reconstructed from the mask image, yields only a fraction of the ridge for small shift values (constant value subtracted).

On the other hand, if one performed a dome extraction on each individual line of the ridge image, the local maximum on each line (the portion of the ridge on each line) would be extracted. This provides some justification for using line reconstructions to extract the ridges of an image. Clearly as the shift value increases the extent of the domes increases.

Figure 11.17 shows the enhanced, ridge, marker and watershed segmented image. The ridge image is computed on the inverted denoised image. The combined (point-wise maximum) of the line domes in the 0^o, 45^o, 90^o and 135^o directions form the ridges. The enhanced image is created by adding the ridge image to the inverted denoised image and then histogram stretching to ensure that the observed intensities lie between 0 and 255. The input image is denoised prior to ridge extraction by convolving with a Gaussian kernel of $\sigma = 1$ and footprint 7×7. The ridge image is cleaned by removing all binary connected components less than, or equal to, 30 pixels in extent. This reduces the effect of noise on the ridge image, and removes isolated regional maxima. The parameter chosen to provide a tradeoff between the reproducing the ridges and minimising isolated maxima and noise.

Note that the watershed is little different from before (see figure 11.1). Larger shift values will not be used as this enlarges the ridges unnecessarily.

11.6 Discussion

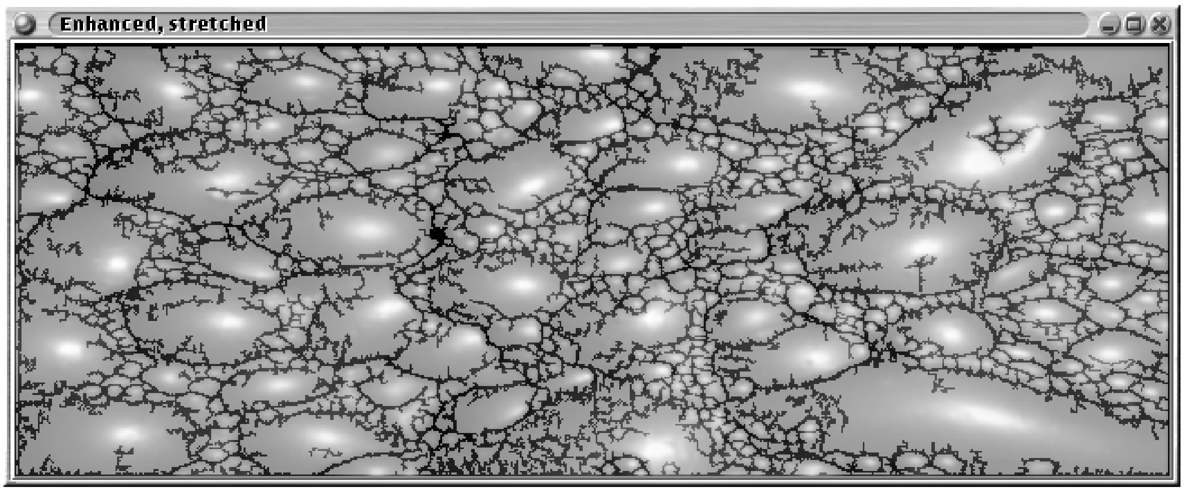
It is possible to find an image enhancement algorithm to reduce the effect of lighting variations across the image. A simple example of this is the use of histogram equalisation to provide an image with uniform illumination. Histogram equalisation has the, desirable, side-effect of eliminating some noise maxima, especially on bubble highlights. This is possible since histogram equalisation changes the relative ordering of some pixels. This provides for a better segmentation.

Histogram stretching, although, improving the visual appearance of the froth image, does not improve the segmentation, or improve contrast in the image. Since histogram stretching does not change the relative ordering of pixels at all, the watershed remains the same as that of the original image.

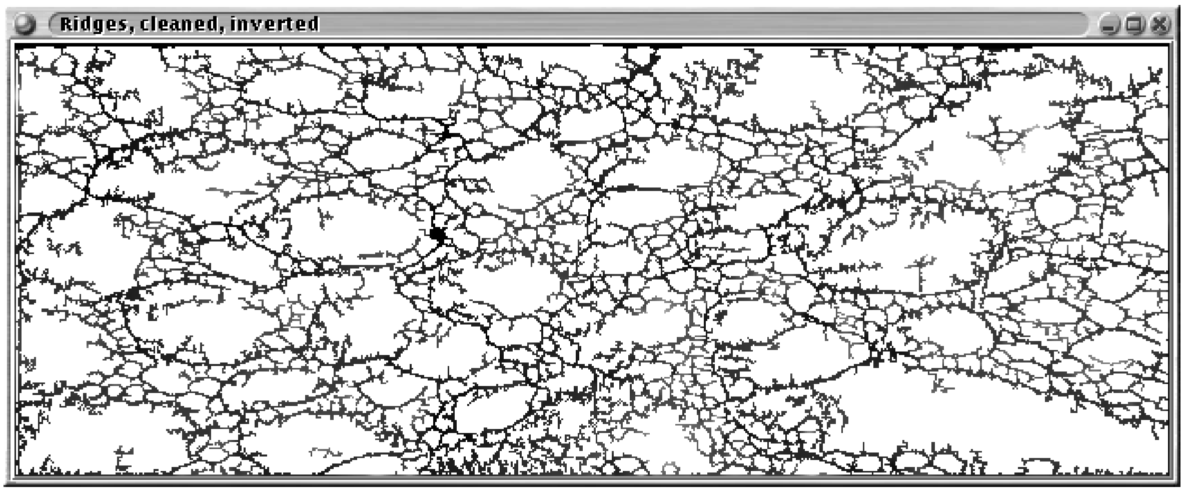
The adaptive enhancement algorithms change the relative ordering of pixels, in other words, the intensities of the pixels are different. This implies that the watershed segmentation may well be different. Adaptive enhancement has the side effect of enhancing noise maxima, this leads to the creation of spurious markers during the segmentation process.

Homomorphic filtering removes low frequency information and attempts to improve the weighting given to edge information. The edge image generated does not suffice to adequately segment the image.

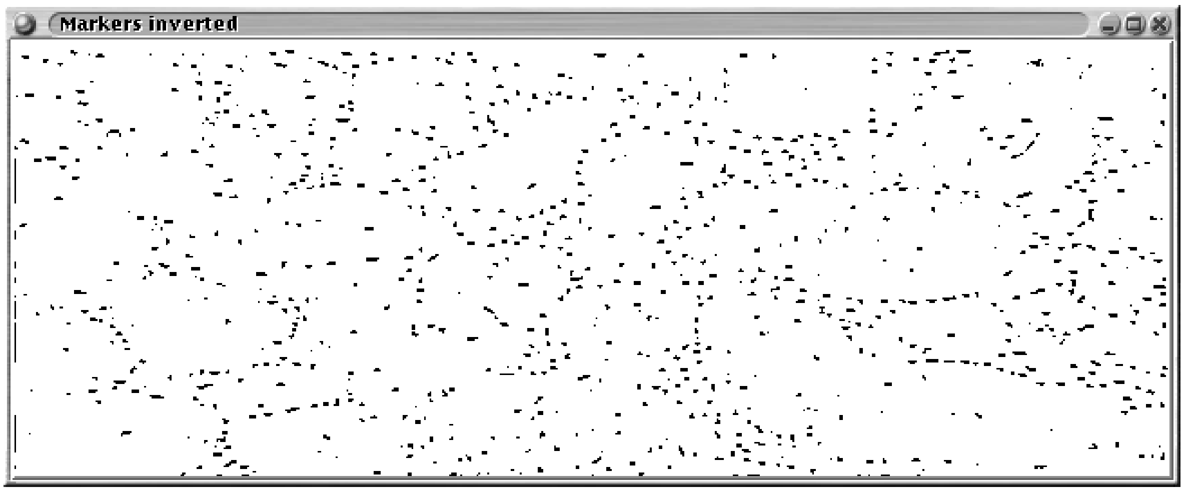
The ridge enhancement algorithms rely on a good estimate of the ridges of the inverted image (the valleys of the original image). Examining the ridge images as generated by the ridge extraction algorithms used, indicate that the ridges do not enclose small bubbles on large bubbles. The difficulty with the segmentation is not the ridges, which can be easily extracted, but generating closed ridges around small bubbles on large bubbles. Ridge enhancement then does not provide a solution for



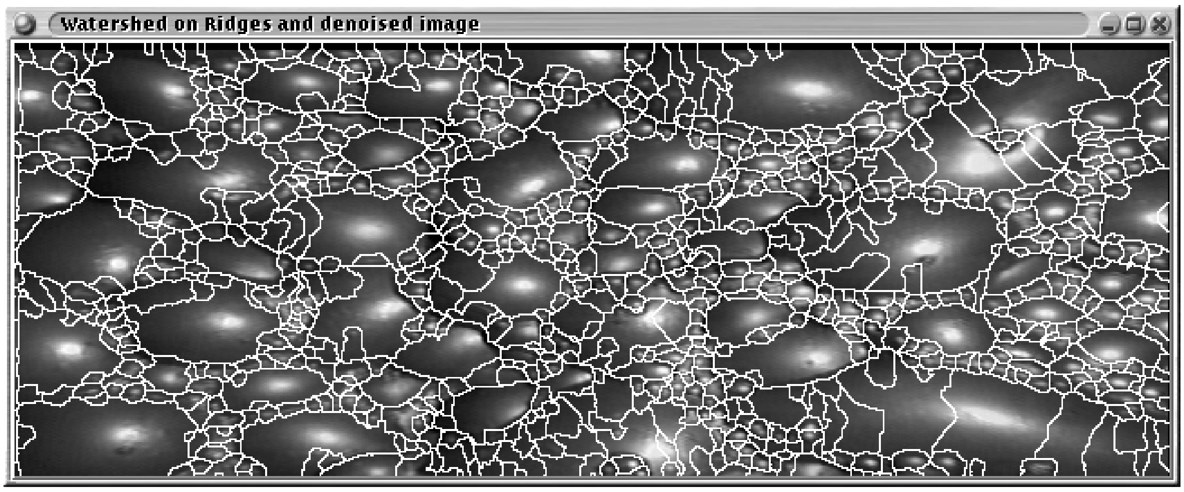
(a) Enhanced image.



(b) Ridges



(c) Markers



(d) Watershed segmented image.

FIGURE 11.17 Ridges from line domes, shift value 1.

assisting in the segmentation of the small bubbles on large bubbles.

In short, it is possible to compensate for variations in lighting across a froth image at the possible expense of enhancing noise artifacts. This indicates a need for denoising froth images before, and possibly after, contrast enhancement. The sensitivity of the enhancement process to noise suggests the use of simple contrast enhancement techniques (baseline subtraction, histogram equalisation or histogram stretching, for example). The other techniques may provide an investigative tool for human inspection of froth images, but the side effect of noise enhancement leads to spurious markers and hence over-segmentation. This is unfortunate, since edge enhancement can be seen to improve the quality of the watershed segmentation for some bubbles. The difficulty is enhancement of slight edges and valleys around small bubbles on large bubbles.

Chapter 12

Comparison of techniques

An investigation into denoising techniques does raise the question of comparison of denoising algorithms. Comparing algorithms qualitatively (i.e. by eye) becomes tedious if many algorithms are compared. A further difficulty lies in that different people may well rank the algorithms differently.

A quantitative means for comparing the algorithms is required. The algorithms will be compared by adding noise to some artificial froth images, denoising, and then comparing the error between the original and denoised images. It should be remembered that denoising is is usually a means to an end. The only real comparison can be made when examining the effect of various denoising algorithms on the ultimate use of the denoised images. For example, comparing the relative effects of the denoising algorithms on segmenting froth images. This however may not be possible, for example, how does one compare segmentations? By eye? An automatic quantitative method of comparison requires a baseline segmentation algorithm, which means there is no reason to segment again using another segmentation algorithm (the information required has already been extracted). This justifies the use of artificial data, since the data may be constructed to have a known segmentation.

The cases of no noise added, and additive Gaussian noise will be examined. Gaussian noise is the usual additive noise assumption made (noise added is usually from many sources, these noise sources add, and via the Central Limit Theorem [93] the total noise input has a Gaussian distribution). The noise is assumed to be zero mean.

Gaussian noise has the following distribution [93],

$$\text{Gaussian}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{12.1}$$

12.1 The test set used

Quantitative results are generated on artificial froth images. Artificial froth images have the advantage of being noise free and the extent of each bubble in the froth is pre-determined. This in turn, allows for the effect of noise, and hence the denoising process, on the froth images to be determined.

A single bubble, in the absence of any other forces acting on the bubble, is assumed to be spherical.

This seems intuitively reasonable (since the elastic membrane making up the surface of the bubble will tend to minimise the surface area enclosing a volume), and no further apology for this assumption will be made. The bubbles making up the surface of the froth can then be assumed to be piecewise spherical or elliptical.

This is somewhat over-simplified, there are an number of forces acting on a froth surface which would distort the bubbles. These include, the froth mass flowing over the weir (gravity) to the motion of the pulp beneath the froth surface, to bubbles bursting and merging (see chapter 2 for an introduction to froth flotation).

The artificial images are generated as follows:

(i) Randomly place a number of points equal to the desired number of bubbles on an image of the desired proportions.

(ii) Tessellate the image into Voronoi regions [38] (via a watershed, for example, using the points as markers). Note that using a watershed, while fast, does not replicate a Voronoi tessellation. This is because the distance propagation of a watershed is different to that of a Voronoi tessellation (compare [38] and [141]). It should be noted that fast algorithms for Voronoi tessellation of the plane do exist (see [131], for example).

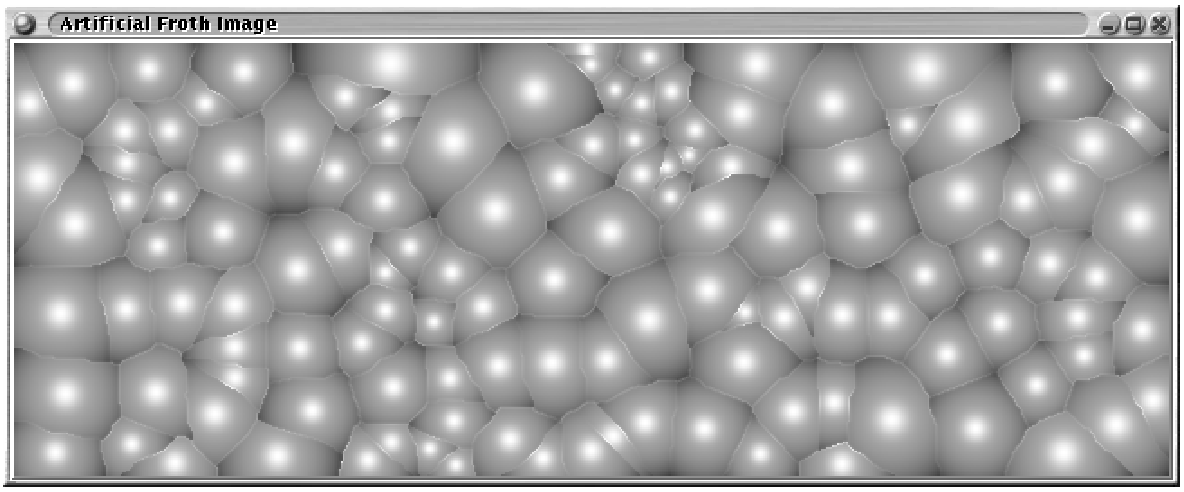
(iii) Find the centroids of the tessellated regions.

(iv) Place a half hemisphere, centred on the region centroid, on each Voronoi region to produce a height image. Select the radius as the largest distance from the centroid to the boundary of the Voronoi region. Simultaneously, generate the x and y derivatives of the half-hemisphere. This is needed during the shading stage to follow, and ensures that the bubble boundaries do not form highlights on the shaded image.

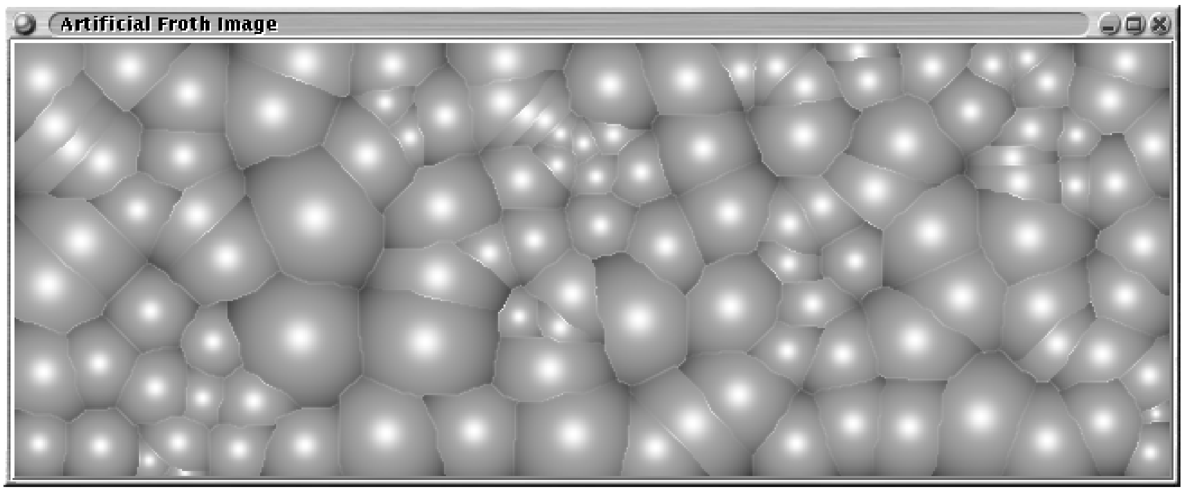
(v) Shade the height image. Simple Lambertian shading is used for the bubbles. Phong shading (see Foley and Van Dam [43][pg. 721-813] for an introduction to shading and shading models) is used for highlight formation.

The artificial image generating technique has the advantage of imposing the initial Voronoi tessellation as the watershed regions of the artificial image. This is because each Voronoi region becomes a shaded half-hemisphere which can be seen to have a single maximum only. Since shading using a single light source will not introduce extra maxima on a concave surface [43], each region has a single maxima. On the inverted image, each Voronoi region has a single minimum and the bubble boundaries become the ridges or regional maxima. Clearly, the ridges divide the watershed of the image, which implies the watershed segmentation of the artificial image is the Voronoi tessellation used to generate the image.

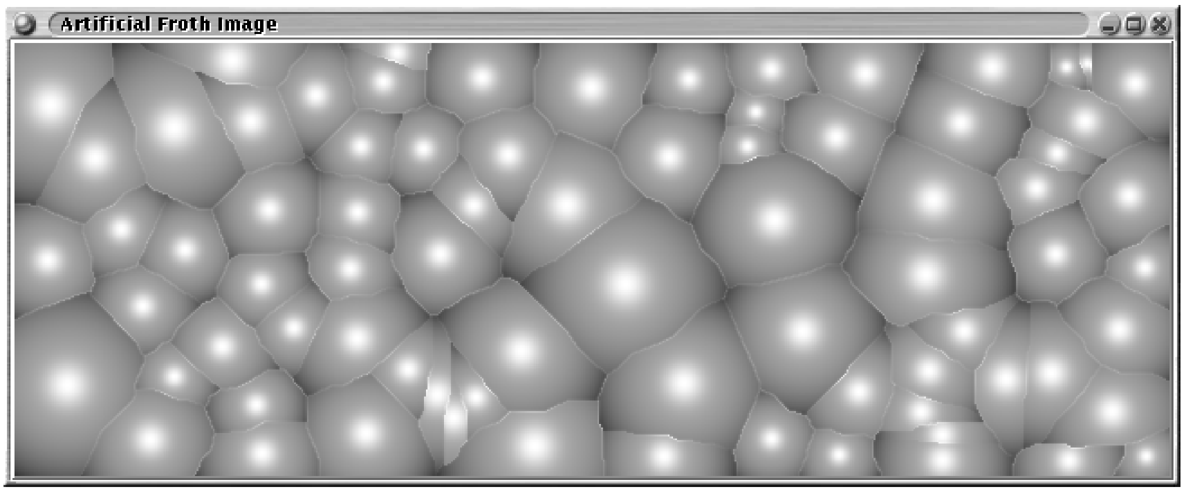
The artificial image does not share certain features of real froth images, for example, small bubbles on large bubbles. The shape of certain bubbles is distorted, especially for elongated bubbles.



(a) Image 1.



(b) Image 2.



(c) Image 3.

FIGURE 12.1 Some artificial froth images.

A test set of 100 640 $\times$ 240 artificial images is generated. To justify the model used, consider figure 12.1 which shows 3 artificial froth images. Note that the boundaries between the bubbles are not always linear, as would be expected for a Voronoi tessellation, this is a result of using a watershed algorithm to tessellate the plane instead.

12.1.1 Shading a height image

If an image representing a height map of a surface is generated, for example as described above, the issue of describing the interaction of the surface and a light source illuminating the surface is of interest. Only a very simple interaction will be described here, Foley, Van Dam *et al* [43][pg. 721–813] provide a good overview of shading from the computer graphics perspective.

The shape from shading problem details the inverse process; given a shaded image, a shading model, and modest assumptions about the illumination, reconstruct the surface [82, 160]. The shape form shading problem provides some insights into the shading models at work in real image formation.

Lambertian and Phong shading will be described. These shading techniques were chosen to provide realistic looking shading of a bubble height map (see figure 12.1, for shaded examples).

Lambertian shading

A surface with a Lambertian shading model scatters light in all directions. The model used is [43]:

$$I_L = I_S K_L \cos\theta \tag{12.2}$$

where, I_S is the intensity of the light source, K_L is a constant unique to the material of the surface. θ is the angle between the surface normal, at a point, and the direction from the point to the light source. $\cos\theta$ is then the dot product between the surface normal ($\vec{N}$) and the direction from the point to the light source ($\vec{L}$).

For practical purposes K_L may be set to unity, effectively subsumed into I_S, the final arbiter of shaded images is how realistic they appear to the eye. It is possible that $\cos\theta$ may be negative, this situation is handled by using $\max(0, \cos\theta)$ rather than $\cos\theta$ alone. A negative $\cos\theta$ indicates a surface normal larger than $\frac{\pi}{2}$ radians (90^o), from the direction to the incident light from the point on the surface. Effectively, the light rays are either parallel to the surface, and hence no interaction with the surface, at this point; or one has self-occlusion of the surface. Self-occlusion is the situation where the path to the light source, for a point on a surface, must pass *through* the surface, the point must then be in shadow, and the illumination zero.

Phong shading

Lambertian shading does not allow for highlight formation, incident light being scattered in all directions possible. Phong shading is an attempt to model a situation where the surface mirrors the incident light. Phong shading is used to account for specular reflection or highlights [43][pg 728-731].

The Phong model is:

$$I_P = I_S K_P \cos^n \alpha \tag{12.3}$$

where, K_P is a material specific constant, n is a user selected positive constant (the specular-reflection exponent) which selects the extent of the highlight, increasing n leads to a smaller highlight. α is the angle between the viewing direction and the reflected light direction ($\vec{R}$). To allow for proper formation of the specular highlight, α is not allowed to exceed 90^o, I_P for this condition is set to 0.

Composite model

For the artificial froth images generated, a composite Lambertian and Phong shading model is used. The shading parameters are:

(i) The initial viewing and illumination directions are chosen, for simplicity, to be $(0,\ 0,\ 1)$.

(ii) n is chosen to be 12.

(iii) Ambient light is chosen to be 10.

(iv) Diffuse light ($K_L I_S$) is chosen to be 256.

(v) $K_P I_S$ is chosen to be 128.

The final model used sums the contribution of the ambient, Lambertian and Phong shading models.

After shading, the artificial image is histogram stretched to lie in the range of pixel intensities allowed (0 to 255). This reduces the dependency on the initial parameters selected somewhat.

12.2 The algorithm parameters

The algorithm parameters, where necessary, were selected by hand. This combined performance improvements made possible by domain specific knowledge, while minimising the computational time.

Anisotropic Diffusion Both Perona and Malik's algorithm and Catte el al's share the following parameters. $\gamma = 0.2$. Three $g(\cdot)$ functions are explored, the GaussianG, QuadG and TukeyBiWeight functions (see sections 7.1 and 7.6 for definitions). For the first two, K is automatically selected using Canny's noise estimator. The Tukey BiWeight function selects its scale parameter using a robust Median of Absolute Differences as specified in section 7.6.

Morphology based denoising,

- Rolling ball dilations, erosion, openings and closings, use a rolling ball structuring element of the radii specified. The filter is not iterated.

- Flat dilations, erosions, openings and closings, use a disc of the radius specified as the structuring element. The filter is not iterated.
- Posterisation does not require any parameters.

Mean filtering, a Gaussian kernel of specified sigma is used. The neighbourhood of pixels examined is $4\sigma \times 4\sigma$, rounded up.

Median filtering, the median filters specified use the masks as specified in chapter 8.

JPEG based denoising, uses the scale factors specified and the quantisation matrix specified in section 10.4.

SPIHT uses the Daubechies N3 wavelet. The MSE allowed is as specified.

Wavelet based denoising uses the Daubechies N3 wavelet. The thresholds are otherwise as specified. Lorentz wavelet thresholding does not require any parameters except specifying the wavelet.

12.3 Comparing the algorithms

The algorithms are compared by adding noise to artificial froth images. Artificial froth images have the advantage of being noise free. This allows for a base-line to be established for each denoising algorithm. The base-line provides information on the image energy discarded by a denoising operation.

Table 12.1 shows the results of comparing the algorithms for the no added noise and Gaussian noise added cases. Results for MAD and MSE are shown. The additive Gaussian noise sigma is five, and the noise added image is thresholded to lie between 0 and 255, the full greyscale variation allowed. The MAD and MSE measured, are measured by comparing the denoised image to the noise added image.

Table 12.2 shows the associated running times for the algorithms. Running times are used as an indicator of the relative complexity of the algorithms. The running times are not exact, exact running times are difficult to measure on a pre-emptive multi-tasking system, due to varying load conditions. The running times given should then be used as a relative ranking of algorithms.

TABLE 12.1 Comparing denoising algorithms

Algorithm		No Noise		Gaussian Noise	
		MAD	MSE	MAD	MSE
Mean Filter	Sigma 1	1.926	7.058	3.767	23.35
	2	2.905	19.29	4.902	41.39

Table continued on next page

Table 12.1 continued from previous page

Algorithm		No Noise		Gaussian Noise	
		MAD	MSE	MAD	MSE
	3	4.122	41.93	5.928	65.35
	4	5.468	75.09	7.062	99.0
	5	6.867	116.5	8.261	140.5
	6	8.255	163.3	9.473	187.3
Median Corner Filter	Iteration 1	0.08188	0.4110	3.001	19.96
	2	0.1116	0.6077	3.104	20.58
	3	0.1116	0.6077	3.124	20.74
	4	0.1116	0.6077	3.125	20.75
	5	0.1116	0.6077	3.126	20.76
	6	0.1116	0.6077	3.126	20.77
Median Filter	Iteration 1	0.7048	3.746	3.747	27.27
	2	0.6366	3.542	3.595	25.35
	3	0.7333	4.373	3.717	26.68
	4	0.7656	4.728	3.756	27.19
	5	0.7988	5.119	3.783	27.61
	6	0.8195	5.367	3.800	27.88
Median Line Corner Filter	Iteration 1	0.05589	0.2670	2.090	12.91
	2	0.08061	0.4128	2.374	14.92
	3	0.08683	0.4523	2.445	15.45
	4	0.09189	0.4829	2.466	15.63
	5	0.09356	0.4940	2.475	15.70
	6	0.09505	0.5034	2.479	15.75
Median Line Diagonal Filter	Iteration 1	0.04216	0.2097	1.172	6.564
	2	0.05535	0.2586	1.332	7.585
	3	0.06346	0.3077	1.368	7.834
	4	0.06629	0.3206	1.377	7.898
	5	0.06869	0.3356	1.380	7.923
	6	0.06967	0.3404	1.381	7.923
Median Line Filter	Iteration 1	0.05590	0.2671	2.826	18.67

Table continued on next page

Table 12.1 continued from previous page

Algorithm		No Noise MAD	No Noise MSE	Gaussian Noise MAD	Gaussian Noise MSE
	2	0.08781	0.4495	3.031	19.99
	3	0.09402	0.4890	3.074	20.32
	4	0.09974	0.5256	3.080	20.38
	5	0.1014	0.5367	3.082	20.40
	6	0.1031	0.5475	3.083	20.41
Flat Close-Opening	Radius 1	0.4263	3.109	4.145	35.73
	2	1.324	14.09	5.750	63.14
	3	2.574	36.49	7.183	97.08
	4	4.164	74.58	8.747	144.2
	5	5.892	127.0	10.40	203.8
	6	7.726	191.1	12.11	273.1
Flat Closing	Radius 1	0.3624	2.918	3.889	35.42
	2	1.003	11.62	5.541	62.38
	3	1.738	24.76	6.730	89.77
	4	2.603	42.51	7.839	121.1
	5	3.489	63.36	8.931	156.3
	6	4.521	90.03	10.08	197.5
Flat Dilation	Radius 1	4.953	51.61	9.515	141.6
	2	9.649	148.3	14.92	298.8
	3	14.21	305.0	19.72	503.8
	4	18.72	504.7	24.29	751.4
	5	23.18	759.9	28.75	1047.0
	6	27.63	1052.0	33.12	1378.0
Flat Erosion	Radius 1	4.902	52.59	9.547	143.4
	2	9.533	151.9	14.95	304.3
	3	13.85	298.7	19.55	501.2
	4	17.86	468.4	23.69	720.8
	5	21.58	661.5	27.48	957.6
	6	25.05	862.5	30.97	1202.0

Table continued on next page

Table 12.1 continued from previous page

Algorithm		No Noise		Gaussian Noise	
		MAD	MSE	MAD	MSE
Flat Open-Closing	Radius 1	0.4249	3.081	4.072	33.77
	2	1.309	13.70	5.429	53.92
	3	2.538	35.37	6.554	78.55
	4	4.110	72.94	7.801	117.3
	5	5.837	126.8	9.224	174.3
	6	7.685	195.4	10.79	248.9
Flat Opening	Radius 1	0.0722	0.2664	3.751	32.00
	2	0.3665	2.929	5.154	50.58
	3	0.9786	13.54	6.220	72.12
	4	1.916	38.01	7.376	106.9
	5	3.133	79.43	8.727	160.2
	6	4.540	136.0	10.24	231.7
JPEG denoising	Scale factor 1	1.850	7.347	4.208	28.65
	2	2.423	11.52	4.604	34.38
	3	3.027	16.78	4.963	40.03
	4	3.659	23.34	5.364	46.70
	5	4.246	30.51	5.779	54.06
	6	4.869	39.15	6.247	62.84
	7	5.459	48.46	6.714	72.26
	8	6.042	58.62	7.190	82.43
	9	6.561	68.44	7.627	92.37
	10	7.064	79.12	8.066	103.1
Posterisation		16.81	501.4	6.218	75.53
Perona Malik Diffusion	Gaussian G Iteration 1	1.603	137.1	3.317	144.6
	2	1.807	137.8	4.139	151.9
	3	1.921	138.3	4.521	156.2
	4	1.999	138.8	4.710	158.5
	5	2.071	139.3	4.836	160.1
	6	2.140	139.8	4.924	161.3
	7	2.210	140.3	4.997	162.4

Table continued on next page

Table 12.1 continued from previous page

Algorithm		No Noise		Gaussian Noise	
		MAD	MSE	MAD	MSE
	8	2.279	140.9	5.059	163.2
Perona Malik Diffusion	Quad G Iteration 1	1.644	137.2	3.537	146.2
	2	1.867	138.1	4.276	153.5
	3	1.997	138.7	4.633	157.6
	4	2.095	139.5	4.809	159.9
	5	2.184	140.3	4.935	161.6
	6	2.268	141.0	5.028	162.9
	7	2.351	141.7	5.108	164.1
	8	2.433	142.5	5.177	165.2
Perona Malik Diffusion	Tukey BiWeight G Iteration 1	1.484	136.8	2.563	140.5
	2	1.631	137.2	3.437	145.9
	3	1.747	137.6	3.945	150.3
	4	1.844	138.0	4.263	153.5
	5	1.918	138.4	4.475	155.9
	6	1.982	138.8	4.625	157.7
	7	2.037	139.2	4.736	159.1
	8	2.086	139.5	4.824	160.2
Catte et al	Gaussian G Iteration 1	1.539	136.9	2.588	140.5
	2	1.722	137.5	3.545	146.8
	3	1.884	138.1	4.117	152.1
	4	2.014	138.8	4.466	155.8
	5	2.124	139.5	4.692	158.5
	6	2.223	140.2	4.851	160.5
	7	2.315	141.0	4.972	162.1
	8	2.403	141.7	5.067	163.4
Catte et al	Quad G Iteration 1	1.556	137.0	2.647	140.9
	2	1.757	137.6	3.619	147.5
	3	1.932	138.4	4.193	152.9
	4	2.076	139.3	4.544	156.8
	5	2.198	140.1	4.774	159.5

Table continued on next page

Table 12.1 continued from previous page

Algorithm			No Noise		Gaussian Noise	
			MAD	MSE	MAD	MSE
		6	2.309	141.0	4.935	161.6
		7	2.412	141.9	5.059	163.4
		8	2.511	142.8	5.161	164.9
Catte et al	Tukey BiWeight G Iteration	1	1.492	136.8	1.969	138.2
		2	1.576	137.0	2.662	141.1
		3	1.683	137.4	3.205	144.5
		4	1.790	137.8	3.627	147.7
		5	1.889	138.3	3.957	150.6
		6	1.978	138.8	4.215	153.3
		7	2.060	139.3	4.422	155.5
		8	2.136	139.9	4.588	157.4
Rolling Ball Close-Opening	Radius	1	0.5831	3.413	3.174	24.62
		2	1.065	8.641	4.616	43.31
		3	1.673	18.33	5.993	66.77
		4	2.432	36.36	7.130	93.76
		5	3.574	63.89	8.546	132.3
		6	4.684	97.98	9.816	173.9
Rolling Ball Closing	Radius	1	0.4950	3.037	2.669	22.23
		2	0.9159	7.697	4.318	42.82
		3	1.332	14.08	5.683	65.18
		4	1.768	23.28	6.625	85.84
		5	2.429	34.82	7.785	113.2
		6	2.962	46.83	8.673	138.1
Rolling Ball Dilation	Radius	1	4.042	35.60	7.388	95.11
		2	7.979	103.3	12.53	216.2
		3	12.19	217.2	17.55	390.1
		4	16.25	369.0	21.79	586.8
		5	20.32	566.6	26.18	837.2
		6	24.30	796.7	30.17	1105.0

Table continued on next page

Table 12.1 continued from previous page

Algorithm		No Noise		Gaussian Noise	
		MAD	MSE	MAD	MSE
Rolling Ball Dilation Shift	Radius 1	3.044	28.52	6.394	81.31
	2	5.985	75.38	10.55	170.1
	3	9.198	153.1	14.58	293.9
	4	12.27	255.0	17.83	428.4
	5	15.36	388.4	21.23	600.4
	6	18.35	541.0	24.24	778.7
Rolling Ball Erosion	Radius 1	4.022	35.95	7.415	95.92
	2	7.936	107.0	12.64	222.0
	3	12.01	220.1	17.69	400.6
	4	15.86	361.7	21.87	595.2
	5	19.57	532.3	26.05	829.5
	6	23.07	717.0	29.75	1069.0
Rolling Ball Erosion Shift	Radius 1	3.022	28.90	6.414	82.07
	2	5.936	79.27	10.65	175.4
	3	9.015	157.0	14.69	303.5
	4	11.86	250.8	17.87	436.3
	5	14.57	361.6	21.05	594.1
	6	17.05	475.5	23.74	747.3
Rolling Ball Open Closing	Radius 1	0.6129	3.350	3.146	23.81
	2	1.038	7.853	4.489	39.48
	3	1.595	16.84	5.587	55.85
	4	2.389	34.33	6.454	74.97
	5	3.438	61.18	7.592	105.7
	6	4.602	98.38	8.640	144.4
Rolling Ball Opening	Radius 1	0.4041	1.785	2.589	20.44
	2	0.5713	2.934	4.100	36.97
	3	0.7542	6.040	5.262	52.61
	4	1.147	16.82	6.119	70.21
	5	1.893	36.98	7.335	100.8
	6	2.629	65.77	8.425	138.7

Table continued on next page

Table 12.1 continued from previous page

Algorithm		No Noise		Gaussian Noise	
		MAD	MSE	MAD	MSE
Wavelet Hard Thresholding	Threshold 1	0.5031	0.5058	0.5003	0.5007
	2	0.6012	0.7041	0.6076	0.7216
	3	0.7395	1.022	0.9191	1.516
	4	0.8649	1.379	1.318	2.934
	5	0.9739	1.757	1.742	4.955
	6	1.065	2.130	2.159	7.474
	7	1.144	2.492	2.549	10.32
	8	1.215	2.852	2.900	13.28
	9	1.283	3.217	3.204	16.16
	10	1.349	3.592	3.459	18.81
Wavelet Soft Thresholding	Threshold 1	0.7838	1.108	1.142	2.150
	2	1.127	2.244	2.020	6.387
	3	1.387	3.463	2.743	11.68
	4	1.607	4.716	3.310	17.02
	5	1.806	6.002	3.741	21.81
	6	1.992	7.326	4.061	25.83
	7	2.170	8.688	4.300	29.10
	8	2.340	10.08	4.478	31.73
	9	2.502	11.50	4.620	33.94
	10	2.659	12.94	4.739	35.87
Wavelet Percentage Thresholding	Threshold 89%	0.8214	1.276	3.329	17.43
	90%	0.8674	1.421	3.418	18.38
	91%	0.9195	1.598	3.509	19.37
	92%	0.9788	1.819	3.606	20.46
	93%	1.048	2.105	3.709	21.65
	94%	1.132	2.492	3.821	23.00
	95%	1.241	3.056	3.943	24.52
	96%	1.393	3.947	4.081	26.33
	97%	1.623	5.484	4.248	28.67
	98%	2.011	8.548	4.485	32.34
	99%	2.846	16.80	4.990	41.11

Table continued on next page

Table 12.1 continued from previous page

Algorithm		No Noise		Gaussian Noise	
		MAD	MSE	MAD	MSE
Wavelet Lorentz Thresholding		1.538	4.885	4.007	25.48
SPIHT	MSE 1	0.8734	1.395	0.8724	1.333
	5	1.666	5.602	1.844	5.458
	10	2.303	10.84	2.580	10.46
	15	2.872	16.09	3.151	15.58
	20	3.332	21.35	3.605	20.58
	25	3.762	26.57	4.016	25.58
	30	4.201	31.89	4.395	30.76
	35	4.578	37.22	4.692	35.87
	40	4.908	42.45	5.014	41.17
	45	5.216	47.66	5.288	46.37
	50	5.534	52.96	5.572	51.61

12.4 Discussion

The ideal smoothing filter should preserve sharp bubble boundaries but remove noise and as far as possible remove the small bubbles which are sometimes found on large bubbles. On the other hand small bubbles which lie between the large bubbles should be, as far as possible, preserved. Preserving boundaries (edges) while denoising is shown to be feasible using non-linear filters.

12.4.1 Anisotropic diffusion

It is clear that Tukey's bi-weight function removes less energy from an image than the other functions for $g(\cdot)$. This is a result of its edge preservation properties, especially for strong edges.

The Catte et al. reformulation of diffusion blurs the edges and reduces the diffusion process. The error is then smaller than for the Perona and Malik algorithm. The Catte et al algorithm thus preserves detail better.

12.4.2 Mean filters

The detail preserving properties of mean filters are poor if compared to some of the non-linear filters, such as, median filters.

TABLE 12.2 Timing denoising algorithms.

Algorithm	Running time (ms)										
	1	2	3	4	5	6	7	8	9	10	11
	Increasing Radius $\longrightarrow$										
Flat Dilation	6512	16445	31221	50545	74608	103378					
Flat Erosion	6477	16360	31058	50379	74558	103533					
Flat Closing	13010	32880	62408	101541	149665	207444					
Flat Opening	12999	32962	62263	101344	149563	207761					
Flat Close-Opening	25914	65501	124099	201621	298221	413959					
Flat Open-Closing	25942	65645	124381	202110	299061	415816					
Rolling Ball Dilation	7375	18308	36042	58977	90311	125257					
Rolling Ball Dilation-Shift	7760	18750	36484	59482	90845	126273					
Rolling Ball Erosion	7426	18461	36311	59401	90861	126047					
Rolling Ball Erosion-Shift	7786	18840	36733	59961	91504	126634					
Rolling Ball Closing	14808	36985	72569	118514	181584	251849					
Rolling Ball Opening	14822	36889	72604	118901	181375	252278					
Rolling Ball Open-Closing	29738	73938	145355	237649	364993	504010					
Rolling Ball Close-Opening	29690	73911	145009	237867	363771	506210					
	Increasing Iterations $\longrightarrow$										
Median Corner Filter	27427	56225	84254	112374	140391	168489					
Median Filter	10499	20630	30687	40709	50740	60737					
Median Line Corner Filter	25192	51800	77663	103561	129422	155384					
Median Line Diagonal Filter	14315	26679	42153	55989	69832	83770					
Median Line Filter	26254	53958	80852	107814	134453	161816					
	Increasing Scale Factor $\longrightarrow$										
JPEG denoising	10631	10632	10662	10643	10652	10637	10651	10656	10665	10654	
	Increasing σ $\longrightarrow$										
Mean Filter	7396	13466	19509	25631	31721	37804					
Posterisation	51322										

Table continued on next page

Table 12.2 continued from previous page

Algorithm	Running time (ms)										
	1	2	3	4	5	6	7	8	9	10	11
	Increasing Iteration ⟶										
Perona Malik Diffusion											
Gaussian G	10595	20506	30288	40228	50042	59836	70204	80164			
Quad G	6083	12029	17206	23706	29029	35136	40456	46787			
Tukey BiWeight G	15556	32240	49237	66758	83212	101832	119313	136954			
Catte et al											
Gaussian G	11328	22470	33995	45327	56170	67675	79240	89250			
Quad G	9349	18596	27815	36912	46231	55388	64377	73970			
Tukey BiWeight G	12031	26526	41010	54869	70230	84119	99699	113015			
	Increasing Hard Threshold ⟶										
Wavelet											
Hard Thresholding	35652	35703	35653	35567	35758	35576	35710	35758	35550	35651	
	Increasing Soft Threshold ⟶										
Soft Thresholding	35918	35794	35792	35736	35645	36020	35810	35681	35940	35949	
	Increasing Percentage Threshold ⟶										
	89%	90%	91%	92%	93%	94%	95%	96%	97%	98%	99%
Percentage Thresholding	38760	38480	38563	38417	38462	38599	38482	38631	38725	38692	38491
Lorentz Thresholding	40385										
	Increasing MSE ⟶										
	1	5	10	15	20	25	30	35	40	45	50
SPIHT	36386	37114	37591	38163	38586	38873	39105	39797	39701	40186	40612

12.4.3 Median filters

Examining the no noise added case allows median filters to be ranked in order of detail preservation, or, Line Diagonal preserving, Line Corner preserving, Line preserving, Corner preserving, and, finally, the simple Median filter. The detail preserving properties of the median filters have the undesirable side-effect of preserving noise. This is visible when comparing the no added noise to the Gaussian added noise cases. Note that the error for two iterations of the simple Median filter is lower that one.

12.4.4 Morphology

The morphology based algorithms examined are slow for large structuring element sizes and do not denoise enough for small structuring element sizes. Large structuring element also remove significant image energy as well as noise. This suggests that the morphological type algorithms examined are unsuitable for denoising and, hence, froth imaging.

Flat operators

The detail preserving properties of flat morphological operators may be used to generate the following ranking: Opening, Closing, Open-Closing, Close-Opening, Erosion, and, finally, Dilation.

Posterisation

Posterisation shows a better error measure for additive noise than for the no noise case. This illustrates the primary deficiency of the Posterisation algorithm, namely, an image is represented in terms of flat regions. The more flat regions, or watershed regions, the better the image is reproduced.

Rolling ball operators

The detail preserving properties of rolling ball morphological operators may be used to generate the following ranking: Opening, Closing, Open-Closing, Close-Opening, Erosion-Shift, Erosion, Dilation-Shift and, finally, Dilation.

If one uses a rolling ball to extract the regional maxima of the froth image, then smaller rolling balls must be used to denoise. Recall that rolling balls will remove maxima (or minima depending on the operator used) narrower than the ball radius. This implies only extrema of larger scale remain after the denoising.

12.4.5 Occam filters

The Occam filters are fast compared to the other algorithms. This makes them attractive for real-time applications. The removal of noise energy is explicit. The disadvantage is that undesirable image artifacts may not be removed. This implies that the markers of the froth image need to be processed prior to segmentation. Archiving images generated by a real-time application requires some degree of compression. Occam filters allow for denoising and a compressed representation of an image to be generated simultaneously.

Lorentz thresholding generates a percentage threshold from an image. Percentage thresholding uses the given percentage to find a hard threshold. The hard threshold sets the lowest magnitude of wavelet coefficients maintained. This implies that the hard threshold actually selected by both Lorentz and Percentage thresholding is likely to be different for the no noise and noise added cases. This makes direct comparison between the two cases difficult.

JPEG based denoising discards a great deal of signal energy, this implies that it is unsuitable for denoising images. JPEG denoising uses non-overlapping blocks which implies potential discontinuities, and hence artifacts, on the borders of neighbouring blocks.

SPIHT is difficult to rank since the denoising is set by explicitly selecting a MSE. It is then difficult to estimate the image energy which is removed by this denoising technique. The MSE is lower for the Gaussian noise added case. The MAD is larger than for the no noise added case (except at MSE 1). This implies noise removal. The only disadvantage of SPIHT is that the denoising time increases (nearly) linearly as one requires more denoising.

12.4.6 Summary

The utility of the algorithms examined for denoising may be expressed as follows:

(i) Anisotropic diffusion performs well, but may require a number of iterations for good performance. The Tukey BiWeight G is slower, but preserves detail well. The Quad G and Gaussian G perform similarly although the Quad G is faster.

(ii) Mean filters, while fast for small footprints, do not preserve edges well. They are unsuitable.

(iii) The simple median filter, alone of the family of median filters examined, performs acceptably, although it may require a number of iterations.

(iv) The morphological type algorithms examined are unsuitable for froth imaging in terms of speed and performance.

(v) Of the Occam filters examined, wavelet based denoising performs well. The wavelet based denoising algorithms have the advantage of being fast and denoising well. They will not, however, perform well if one seeks to remove features on froth images, such as small bubbles on large bubbles, which are not noise-like. If combined with marker processing this disadvantage may be overcome for very small bubbles.

The requirements for a real time system demand fast algorithms, even though this may be at the expense of some noise induced over-segmentation. The algorithms with acceptable denoising performance may be ranked as follows in terms of speed:

(i) Median filtering, provided the number of iterations does not exceed four.

(ii) Wavelet based denoising.

(iii) Anisotropic diffusion. The $g()$ functions may be ranked as Quad G, Gaussian G and Tukey's BiWeight G.

The denoising performance of the algorithms listed above, is in the reverse order of the list. It is likely that some degree of marker post processing prior to segmentation will be employed. This would reduce the dependency on the denoising stage of a froth imaging system somewhat.

Part III

Motion Analysis of Froth Image Sequences

Chapter 13

Motion Analysis: An Introduction

Motion is a useful measurement for froth sequence analysis (see Chapter 2). The faster the froth over the weir of a flotation cell, the more material is collected (for a given pulp height).

Motion estimation in froth image sequences is complicated by the inherent smoothness of the froth images. Previous work by Horn and Schunk [68] has shown that corners are the only points at which motion can be unambiguously determined (the so called aperture problem).

Several techniques for estimation motion of froth images were examined. All, with one exception, are dense optical flow estimation algorithms. The dense optical flow estimation algorithms cover gradient-based algorithms (a typical example here is Horn and Schunk's algorithm [68]) and correlation based techniques (examples here include block motion estimation and segmentation based motion estimation).

An exception to dense optical flow methods is the Pixel Tracing algorithm [107]. The pixel tracing algorithm is examined simply for completeness, it was one of the first algorithms used for motion estimation of froth image sequences. It should be noted that other techniques for estimating motion of froth have been proposed. These techniques include texture based methods which attempted to model velocity using texture methods (see [29, 96, 97, 98] for some examples of this).

Subsequently, the layout of the motion estimation part, and the pre-processing stage prior to motion estimation, is described.

13.1 Road map

Subsequent chapters will examine the problem of estimating a dense motion field, given a pair of successive images. Chapter 14 examines correlation based algorithms (including Pixel Tracing), segmentation and clustering based algorithms. This include block matching algorithms (Li and Salari's SEA [87], for example) and segmentation based algorithms (for example, Kottke and Sun's motion from cluster matching [84]).

Chapter 15 examines gradient based optical flow algorithms. This includes Horn and Schunk's algorithm [67] and some extensions, for example Div-Curl optical flow [54].

Chapter 16 examines hierarchical motion estimation algorithms. Frameworks for embedding motion estimation algorithms and extending the results across varying spatial scales are examined.

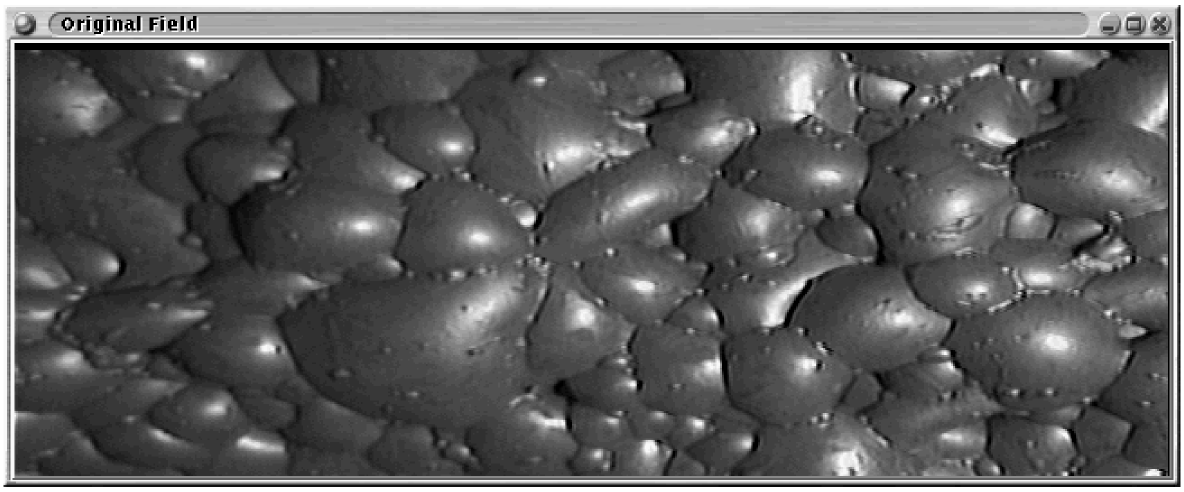
As a whole, the froth mass has an average motion (even this average motion has to be qualified, froth moves rapidly once it overflows the cell into the launder, and in the centre region of the cell the froth is more stagnant and prone to bubble bursts and merges). The effect of bubble bursts and merges on the motion estimation will be examined. It is likely that a bubble burst or merge event will correspond to a gross discontinuity in the motion vector field. This suggests an avenue for identifying bubble bursts or merges, and is explored in chapter 17.

The writeup on each algorithm is as follows: a general explanation of the algorithm, followed by a an experiment on some data. A difficulty with quantitatively evaluating motion estimation algorithms is the lack of any base-line motion data. Chapter 18 will propose a quantitative measure of comparison, and quantitatively compare the algorithms described. One method of comparison generates a reference data set by warping an image and measuring the difference between the generated motion field and the field used to warp the image.

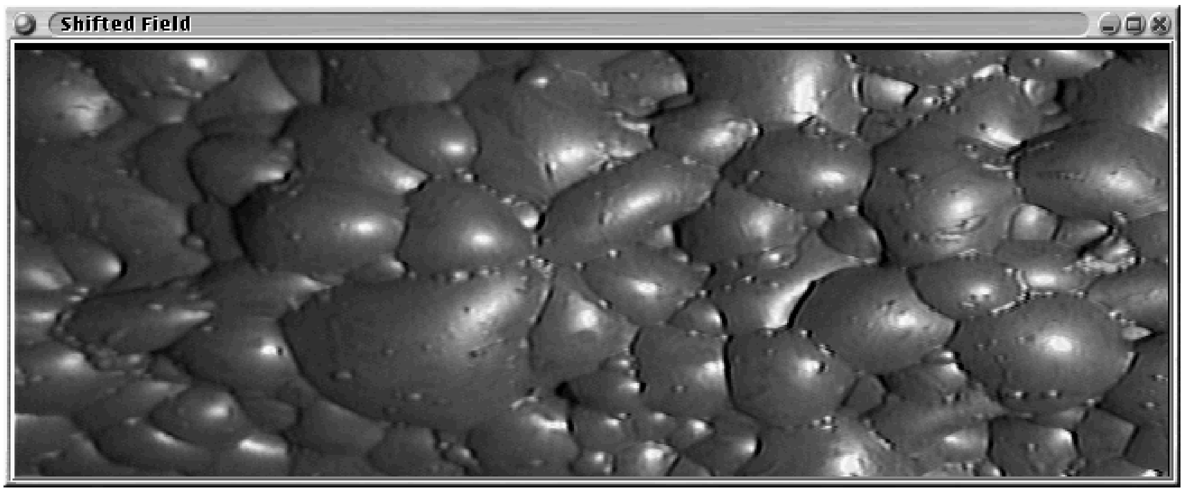
Once a motion field between two images (A and B) is generated, B can be inverse warped using this motion field. A second method of comparison compares the warped image image A using some Mean Square Error (MSE) for example [137] (this is related to existing methods for evaluating temporal compression in image sequence compression (see [139][pg. 440–442]), television frame rate conversion [10] and motion detection [154]).

13.2 Preprocessing before motion estimation

In all cases the images are de-interlaced, the even field extracted, and then denoised before motion analysis is attempted. This serves to increase the accuracy of the final motion vector field by minimising the effect of noise. A frame grabbed via a Phase Alternating Line (PAL) (a broadcast television standard) camera, consists of two fields taken $\frac{1}{50}$ of a second apart [130]. Interlacing refers to the lines within the two fields alternating with each other, for example, line 1 in the frame is line 1 of the first field (the even field) and line 2 of the frame is line 1 of the second field (the odd field). To eliminate any possible artifacts due to motion occurring between the times the fields are formed, de-interlacing is needed. De-interlacing separates a frame into its constituent fields. Because of the importance of edges and corners to the motion estimation process, a non-linear edge preserving denoising algorithm is first applied to the image. The denoising algorithm used was Perona and Malik's anisotropic diffusion algorithm [113] (see section 7.1 for a description of anisotropic diffusion). The algorithm parameters used were: 4 iterations, $\gamma = 0.2$ (or the time step in the simulation) and the edge processing function $g() = e^{\|\nabla I\|/K^2}$. The test images used are shown in figure 13.1.



(a) Even field of frame 1.



(b) Even field of frame 2.

FIGURE 13.1 The test images used for motion estimation.

13.3 Some terminology

Some terms used subsequently will be defined here. Given a sequence of images, one may generate a motion field two successive images. The first image, at time t, will be designated the **reference** image. The next image in the sequence, at time $t + \Delta t$ will be designated the **motion** image [104].

Segmentation and correlation motion estimation algorithms divide the motion and reference images into rectangular blocks, or, more generally, irregularly shaped regions. These regions are then compared in some way to produce the motion estimate for the region [104].

The **search block** (or **search region**) is a region in the reference image centred on the pixel whose motion is to be estimated. The **best-match block** is the block in the **motion** image which is most similar to the **search block**. A **best-match block** for a **search block** at position $(x,\ y)$ in the **reference** image is searched for in a small region around $(x,\ y)$ in the **motion** image.

Chapter 14

Motion estimation via segmentation and correlation

Correlation based motion estimation algorithms have roots in the temporal compression of image sequences. Block matching algorithms date to the early 1960's and are reasonably well understood [9, 78, 104, 159]. A multitude of block motion estimation algorithms exist which, in principle at least, may be applied to motion estimation [39, 44, 86]. Examples include the Three Step Search [21], block matching in the feature domain [6] and so on.

In its simplest form, correlation based motion estimation at a pixel correlates a region around a pixel (the search region) in one frame, with the a larger region (usually rectangular or square) in the next frame. The position of the largest correlation peak denotes the position of best match in the next frame, and the difference between the the best match position and start position becomes the motion vector.

This approach, naturally, has certain problems, namely, each motion vector is computed independently of every other. In contrast, one would expect neighbouring pixels to have correlated motion. In other words, as formulated, one cannot expect block based motion estimation to yield smoothly varying correlated motion fields. For temporal compression, this is not an issue, one is more concerned that the difference between the search and best match regions are small. This ensures that the the minimum information is transmitted to convert the best-match region into the search region.

Segmentation based motion estimation algorithms break up the motion and reference images into a series of regions via some segmentation algorithm. This chapter only details a motion from cluster matching algorithm, and motion from watershed segmentation. The motion in each region is then, simply, the difference between the centres of mass of the original and best match region, or slightly more complicated, an affine transformation of the best match and original regions with the warping at each point denoting the motion.

This chapter proceeds by introducing a justification for using the error norms that are used. The effect of overlapping regions on the motion estimation process is examined. The algorithms examined in detail, include Nguyen's and Holtham's Pixel Tracing [107], MSE based SEA, a regularisation of block motion estimation, Kottke and Sun's motion from cluster matching [84], and finally a motion from watershed algorithm.

With all algorithms described, an example motion field, together with the difference between the field at time t and the registered image derived from the motion field and the even field at time $t + \Delta t$, is shown for *qualitative* comparison between the techniques. To make the differences more visible, the contrast of the difference image is enhanced by histogram stretching and by inverting the image (see section 11.1 for a description of histogram modification algorithms for contrast enhancement).

In all cases the fields used are de-interlaced and then denoised by 4 iterations of Perona and Malik's Anisotropic Diffusion algorithm [113] (see section 7.1), here the time step ($\gamma = 0.2$) and the K value is determined using Canny's noise estimator (as described by Perona and Malik [113]). The choice of 4 iterations is determined by the tradeoff between removing noise (and smoothing away bubbles too small to be of significance) and merging bubbles best left separate.

14.1 A probabilistic perspective

The Mean Square Error (MSE) and Minimum Absolute Difference (MAD) error norms are commonly used in block motion estimation [104]. It can be shown that these error norms correspond to a maximum likelihood estimation process in the presence of Gaussian and Laplacian noise respectively.

14.1.1 Gaussian noise

It can be readily shown that block (or more generally, region) matching is a special case of maximum likelihood estimation of motion vectors in the presence of noise. Given that the noise in the image is Gaussian in nature, a good match at a pixel results in the noise component only being visible. The central limit theorem provides some justification for this assumption [93]. The probability of the resultant grey level in the difference image at a particular pixel $(x,\ y)$, is,

$$P(x,\ y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(A(x\ y) - B(x+u,\ y+v)\right)^2}{2\sigma^2}} \tag{14.1}$$

where, $A(x,\ y)$ is the search block, $B(x+u,\ y+v)$ is the best match block, and $(u,\ v)$ is the motion vector.

$A(x\ y) - B(x+u,\ y+v)$ is the grey level difference between original and best match pixels, and σ the noise standard deviation. Clearly then a region of pixels then has the probability of match as the product of the individual pixel matches (for some trial motion field). The parameters of the motion field is then the ones that maximise the aggregate probability. The problem reduces to finding the Mean Square Error (MSE) between a region and its displaced copy in the next frame. This suggests that MSE is optimal for Gaussian noise. It is also unnecessary to estimate the noise variance since the noise standard deviation occurs as part of a multiplicative constant and an exponent of the result. Both processes do not affect the position of the maxima and minima of functions with only one maximum or minimum. A motion vector which minimises the sum of the squared pixel differences (MSE) clearly also then maximises equation 14.1, or the probability of the match being accepted.

14.1.2 Laplacian noise

The process can then be repeated for the Laplacian noise case [91],

$$L(x) = \frac{1}{\sqrt{2}\sigma} e^{-\frac{\sqrt{2}\,|A(x\ y) - B(x+u,\ y+v)|}{\sigma}} \tag{14.2}$$

Here $A(x\ y) - B(x+u,\ y+v)$ represents the pixel difference as before, and σ determines the variance of the distribution. The aggregate probability for a region is then related to the sum of the absolute values of the pixel differences. This is the justification for the Minimum Absolute Difference (MAD) mismatch measure popular in Block Motion Estimation (BME). This is useful since impulse noise (speckle) is sometimes modelled as a heavy-tailed distribution, of which the Laplacian distribution is an example [36].

14.2 Correlation between neighbouring motion vectors increases as the size of the search block increases

The relationship between neighbouring motion vectors will be considered. It can be shown by a simple argument that as the search block becomes larger, that neighbouring motion vectors become more correlated. Consider two overlapping search blocks. The second search block can be thought of as two separate regions:

(i) A region corresponding to the intersection of the domains of the two search blocks.

(ii) A region not within the first search block.

A similar argument for the first search block may be made. It can be shown that if the size of the overlapping region is large, compared to the regions unique to the individual search blocks, the correlation between the blocks is large. Consider the following ratio (which could be trivially extended to n dimensions and arbitrarily shaped regions):

$$\frac{MSE_1}{MSE_2} = \frac{\sum_{k\epsilon \text{block 1}} (f_k - g_k)^2}{\sum_{m\epsilon \text{block 2}} (f_m - g_m)^2} \qquad \text{Note that } A/B \text{ denotes set subtraction here.} \tag{14.3}$$

$$= \frac{\sum_{p\epsilon \text{block 1}\cap\text{block 2}} \left(f_p - g_p\right)^2 + \sum_{a\epsilon \text{block 1}/(\text{block 1}\cap\text{block 2})} (f_a - g_a)^2}{\sum_{q\epsilon \text{block 1}\cap\text{block 2}} \left(f_q - g_q\right)^2 + \sum_{b\epsilon \text{block 2}/(\text{block 1}\cap\text{block 2})} (f_b - g_b)^2} \tag{14.4}$$

$$= \frac{1 + \dfrac{\sum_a (f_a - g_a)^2}{\sum_p \left(f_p - g_p\right)^2}}{1 + \dfrac{\sum_b (f_b - g_b)^2}{\sum_p \left(f_p - g_p\right)^2}} \tag{14.5}$$

The ratio will then be one provided the MSE of the non-overlapping regions is small, and the MSE of the overlapping regions dominate. This is easy to achieve provided the number of pixels in the overlapping region is large. The position of the best match and the error measure will then be close. A similar argument may be constructed for MAD (or, in fact, any error norm which is summed over a region of pixels).

This argument provides some justification for the common practice of subsampling the search space common in BME algorithms (see [86, 159] for some examples). It should be noted that the size of the search blocks is usually too small (typical values include 7×7 and 15×15) for the massive subsampling used (see, for example, the Three Step Search and its derivatives [9, 21, 86, 159]). For example, consider the subsampling used by the Three Step Search in its first stage, a factor of 4. For a 15×15 block the ratio of the overlapping pixels to the pixels in the block is, $(11 \times 11)/(15 \times 15) = 0.538$, at worst. This implies that approximately half of the pixels do not overlap. The correlation between the subsampled motion vectors is then not expected to be good.

14.3 The Pixel Tracing algorithm

This algorithm was used in an attempt to build a fast froth inspection system by Nguyen and Holtham [107]. It is a correlation based motion estimation algorithm which makes no attempt to calculate a dense optical flow field. Instead a block in the centre of the reference field (frame at time t) is correlated with the motion frame (frame at time $t + \Delta t$) at certain points in an attempt to optimise the search. The average motion is then estimated by the best match position at these distinct points. The algorithm proceeds as follows:

(i) A square region of around 100 pixels wide is selected from the centre of the reference image (at time t).

(ii) In the motion image (at time $t + \Delta t$) starting from the centre of the image, eight directions are searched: North, South, East, West, North-East, North-West, South-East and South-West. The number of pixels searched along each direction is chosen from between twenty to thirty.

(iii) The position of the best match is found by taking the position of minimum sum of squared error.

(iv) The motion vector is then the difference between the two positions; start and best match.

14.3.1 Some experimental results

The following figure (figure 14.1) shows the application of pixel tracing to the standard pair of test images (see figure 13.1). The size of the search block is 101×101 and the number of pixels searched along each direction is 25. These parameters were chosen to minimise the energy in the error image. The dense motion field is generated by replicating the single motion vector generated to cover the entire image. The motion vector (also the average) is measured as $(1,\ -1)$.

14.3.2 Discussion

Some difficulties with this algorithm as a general average motion estimator include:

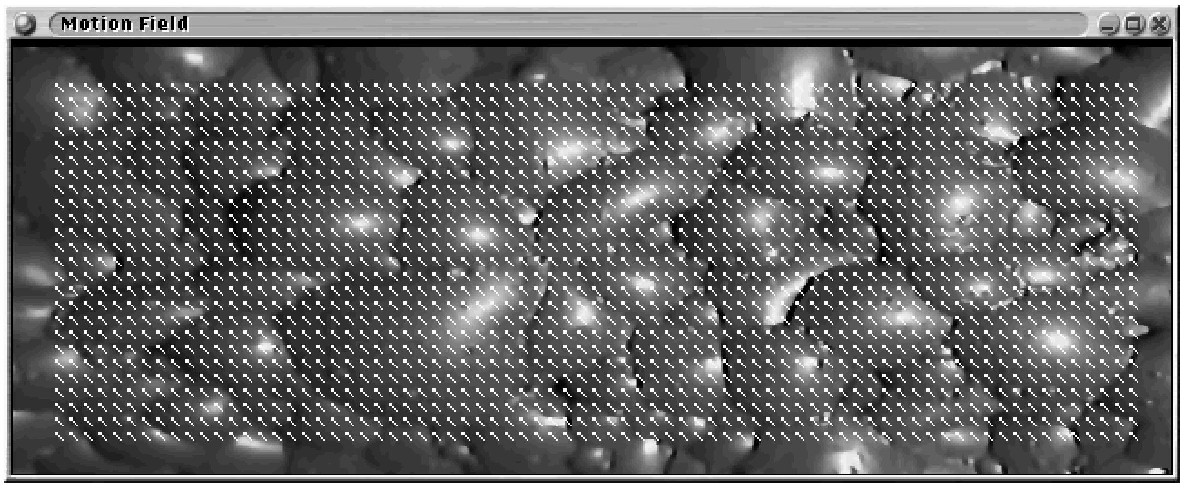
(i) It is possible that two (or more) matches with exactly the same error norm occur, it is not clear how this ambiguous case is to be handled. In the absence of other information, the match with the shortest motion vector should be chosen.

(ii) The motion vector space is not completely searched. For example, how would motion in direction $(10,\ 15)$ be handled? While the Pixel Tracing algorithm was developed with speed in mind, several possible methods exist to extend the search range to include more possible motion vectors. A simple speed up would include use of MSE based SEA as described in section 14.4. Further possibilities involve feature based methods to reduce the size of the search space [6, 87, 123].

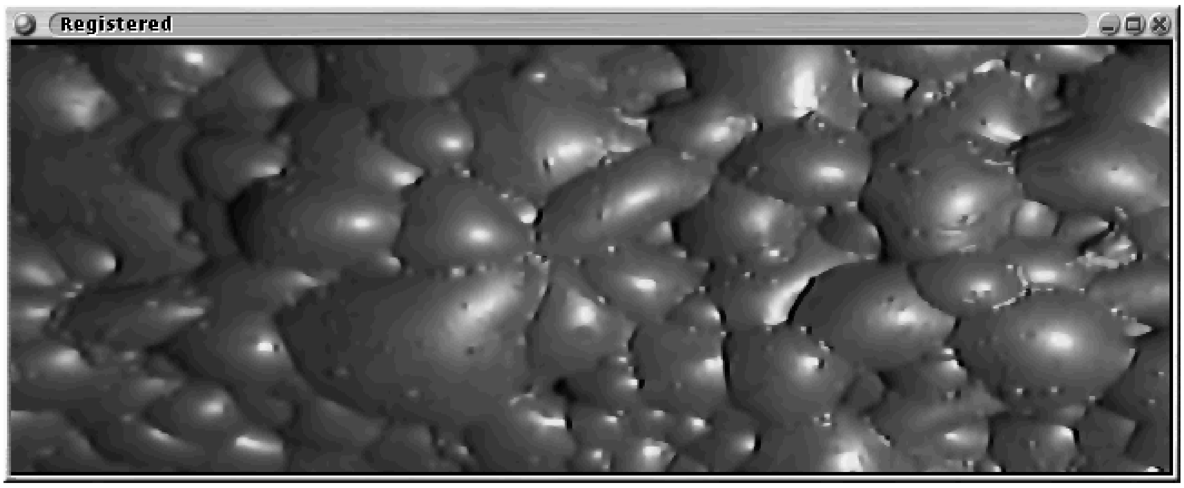
(iii) The limiting case for pixel tracing, as the block size increases, results in the correlation between the images at time $t + \Delta t$ (motion image) and time t (reference image). This is fast to compute in linear transform domains, for example, the Fourier domain using the Fast Fourier Transform (FFT) [73].

14.4 Mean Square Error (MSE) based Successive Elimination Algorithm (SEA)

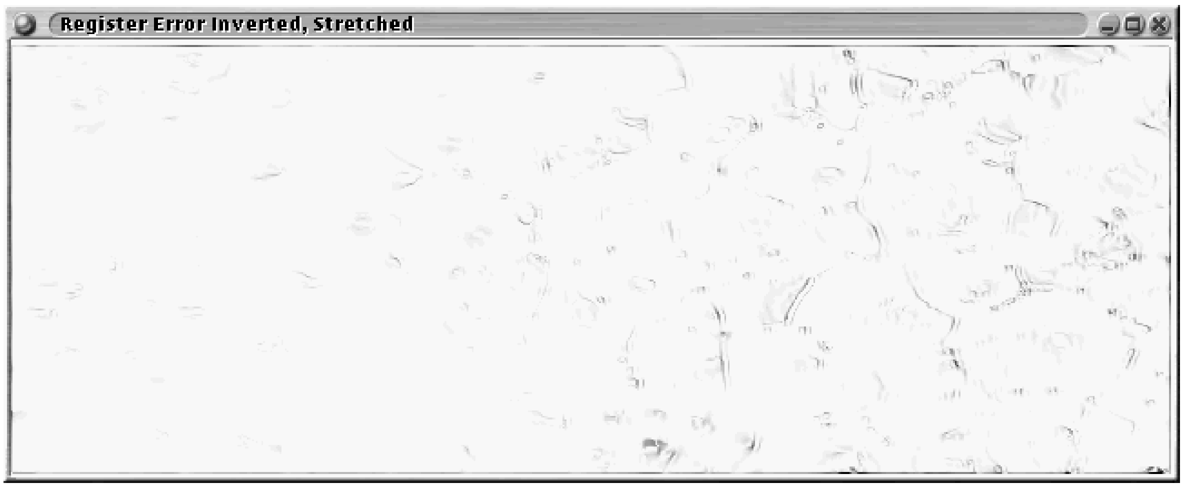
Block Motion Estimation (BME) is used for estimating motion between images. Image sequence compression algorithms (notably, MPEG) generate a difference block for each search block by subtracting the search from the best match block [70]. The difference is transmitted. Optimal temporal compression demands minimising the difference between blocks and this is best accomplished by having as large a search area as possible. A large search area (the area on the search image examined)



(a) An example motion field due to Pixel Tracing. Motion field scaled by 4. Average motion $(1,\ -1)$.



(b) Registered image.



(c) Registered error, stretched. Energy in error image is 6 326 305.

FIGURE 14.1 Motion estimation via Pixel Tracing.

makes finding a better match more likely. Motion estimation, on the other hand, demands a correlated motion vector field. This is a consequence of requiring that neighbouring pixels in an object have similar motion. The correlated motion field is a tradeoff between the size of the search block and the size of the search area (see section 14.2).

Whether computing motion, or attempting to find temporal changes for image sequence compression, the search for a best matching block can be truncated by finding suitable upper and lower bounds on the Sum Squared Error (SSE) and not considering a search position with SSE outside these bounds. This algorithm develops upper and lower bounds on the SSE in terms of the energies of the search blocks and potential best match blocks, and describes a fast method for calculating the energies of overlapping blocks.

BME is use to find temporal differences between frames in several video compression algorithms, notably MPEG-2 [69]. BME involves dividing a frame (called the motion frame or search frame) into a series of blocks (the search blocks) and for each block finding the position of the best match block, via some criterion, in the next frame (called the reference frame) [100]. Usually one minimises the SSE or the MAD (or their averages).

The compression achieved by block motion estimation depends on the search space. Examining every possible point in the search space leads to best compression at the expense of speed. Several algorithms have been proposed in the literature which examine only a small neighbourhood around the origin of the search, these include, Two Dimensional Logarithmic Search[129], Block Based Gradient Descent Search [90], Conjugate Directional Search [129], Conjugate Search [100], Cross Pattern Search [6], Cross Search [47], Dynamic Search Window Adjustment [85], Four Step Search [114], Improved Three Step Search [21], Modified Conjugate Search [100], Modified Three Step Search [86], New Direction of Minimum Distortion [78], One Dimensional Full Search [22], Parallel Hierarchical One Dimensional Search [20], Plus Pattern Search [6], Subblock Full Search [100] and Three Step Search [100].

Until the development of the SEA by Li and Salari [87] a full search involved applying the MAD or SSE (the MSE is the average SSE [73]) at every point on the motion frame. The SEA generates upper and lower bounds on the sum of pixels in the best match block and, having pre-calculated the sum of pixels for every possible best match block, reduces the size of the search space for every improvement of the match. The SEA is tied to the use of the MAD to measure the match. The algorithm described here uses a similar process of calculating upper and lower bounds, but uses the SSE error estimate. The SSE has the advantage of being directly related to the correlation of the search and candidate best-match blocks.

A brief overview of the SEA algorithm and block motion estimation is given, then the bounds on the SSE are derived. An experimental comparison of the SSE based algorithm and the MAD based algorithm is made and finally the results are discussed.

14.4.1 An overview of the SEA

The SEA algorithm is a refinement of the BME process. BME involves dividing a frame (the reference frame) and a successive frame (the motion frame) into a series of blocks, the search and candidate best-match blocks, respectively. These blocks overlap for motion estimation and do *not* overlap for temporal compression. The search blocks are individually correlated with another frame (the motion frame) to find the best match position. The difference between the best match position and the corresponding $(x,\ y)$ co-ordinates of the search block on the reference frame, gives the motion vector.

Two error norms are commonly used: the Sum Squared Error (SSE) (which is MSE without averaging over the number of pixels) and the MAD. The SEA algorithm uses the MAD to derive bounds on the sum of pixels within the best match block. Li and Salari derive the following bound on the sum of pixels in the best match block in terms of the MAD [87]:

$$R - MAD(m,n) \leq M(x,y) \leq R + MAD(m,n) \tag{14.6}$$

where, R denotes the sum of pixels in the search block, $M(x,\ y)$ denotes the sum of pixels in the current candidate best-match block under consideration, and $MAD(m,\ n)$ is the current best match block's MAD.

Clearly, re-computing the sum of pixels of overlapping blocks can be computationally expensive. The SEA proposed a fast method for calculating the sum of pixels.

The calculation proceeds as follows for block size $N \times N$:

(i) For the first element of each row calculate the sum of the first N points on the row ($\sum_{i=0}^{N-1} f(i,k)$, k is the row) and save as $C_{11}, C_{12}, ...$ For example C_{11} is the sum of the first eight pixels on row one. The second element of each row strip is calculated as follows:

$$\begin{aligned} C_{21} &= C_{11} - f(1,1) + f(N+1,1) && \text{and} \\ C_{2w} &= C_{1w} - f(1,w) + f(N+1,w) && \text{and hence,} \\ C_{(v+1)w} &= C_{vw} - f(v,w) + f(N+1+v,w) && \text{at pixel } (v,w). \end{aligned}$$

(ii) For the first element of each column calculate the sum of the first N points on the column ($\sum_{j=0}^{N-1} C_{jk}$, k is the row) and save as $D_{11}, D_{12}, ...$ The second element of each column strip is calculated as follows:

$$\begin{aligned} D_{21} &= D_{11} - C_{11} + C_{(N+1)1} && \text{and,} \\ D_{v2} &= D_{v1} - C_{v1} + C_{v(N+1)} && \text{and hence,} \\ D_{v(w+1)} &= D_{vw} - C_{vw} + C_{v(N+1+w)} && \text{at pixel } (v,w). \end{aligned}$$

The values of D are the sumnorms of the blocks at the given positions.

14.4.2 Derivation of the bounds on the SSE

A bound on the energy in the best-match block, in terms of the MSE between best-match and search block is derived. This is an attempt to optimise block motion estimation using the MSEprocess. SSE can be defined as follows for a motion vector $(a,\ b)$ [73]:

$$SSE = \sum_{y=-N/2}^{N/2} \sum_{x=-N/2}^{N/2} (f(x,y) - g(x+a, y+b))^2 \qquad \text{if N odd, and,} \tag{14.7}$$

$$SSE = \sum_{y=-N/2+1}^{N/2} \sum_{x=-N/2+1}^{N/2} (f(x,y) - g(x+a, y+b))^2 \qquad \text{if N even.} \tag{14.8}$$

Correlation can be defined as follows for a motion vector $(a,\ b)$:

$$f * g = \sum_{y=-N/2}^{N/2} \sum_{x=-N/2}^{N/2} f(x,y)g(x+a, y+b) \qquad \text{for N even, and,} \tag{14.9}$$

$$f * g = \sum_{y=-N/2+1}^{N/2} \sum_{x=-N/2+1}^{N/2} f(x,y)g(x+a, y+b) \qquad \text{for N odd.} \tag{14.10}$$

The output of the correlation process at the moment of complete overlap of the search and reference blocks can be regarded as an inner product since it fits the definition of an inner product [80], where $\langle f, g\rangle$ denotes the inner product of f and g, namely [75][pg. 150–152]:

$$\langle f, g\rangle = \overline{\langle g,\ f\rangle} \tag{14.11}$$

$$\langle \alpha f, g\rangle = \langle f, \alpha g\rangle = \alpha\langle f, g\rangle \tag{14.12}$$

$$\langle f + h, g\rangle = \langle f, g\rangle + \langle h, g\rangle = \langle f, g + h\rangle \tag{14.13}$$

$$\langle f,\ f\rangle \geq 0 \tag{14.14}$$

These follow from the usual definition of the inner product [80]:

$$\langle f, g\rangle = \int_{-\infty}^{\infty} d\tau\ f(\tau)\overline{g(\tau)} \tag{14.15}$$

By inspection equation (14.11) is satisfied for real images, equations (14.12) and (14.13) follow from definition of correlation, equation (14.14) denotes the energy within the block at complete overlap and is always positive or zero. The norm is defined as the Euclidean, or inner norm, and is derived from the inner product as follows [80]:

$$|f| = \sqrt{\langle f,\ f\rangle} \tag{14.16}$$

Closer inspection shows that, evaluated at the point where autocorrelation completely overlaps, the norm denotes the square root of the energy within the block. Denoting correlation by $*$ the SSE at this point can be rewritten as:

$$(f-g)*(f-g) = f*f - f*g - g*f + g*g \tag{14.17}$$

Treating correlation as an inner product,

$$\begin{aligned}
\langle (f-g),(f-g)\rangle &= \langle f, f\rangle - 2\langle f, g\rangle + \langle g, g\rangle &(14.18)\\
|\langle f, g\rangle|^2 &\leq |f|^2|g|^2 \quad \ldots \quad \text{Cauchy-Swartz inequality [75]} &(14.19)\\
SSE &= |f|^2 - 2\langle f, g\rangle + |g|^2 &(14.20)\\
&\geq F^2 - 2\sqrt{|f|^2|g|^2} + G^2 &(14.21)
\end{aligned}$$

which has roots,

$$(F-G+\sqrt{SSE})(F-G-\sqrt{SSE}) \leq 0 \tag{14.22}$$

which implies,

$$\begin{aligned}
(F-G+\sqrt{SSE}) &\leq 0 \qquad \text{and,} &(14.23)\\
(F-G-\sqrt{SSE}) &\geq 0 &(14.24)\\
\Rightarrow G-\sqrt{SSE} &\geq F \geq G+\sqrt{SSE} &(14.25)
\end{aligned}$$

or,

$$\begin{aligned}
(F-G+\sqrt{SSE}) &\geq 0 \qquad \text{and,} &(14.26)\\
(F-G-\sqrt{SSE}) &\leq 0 &(14.27)\\
\Rightarrow G+\sqrt{SSE} &\geq F \geq G-\sqrt{SSE} &(14.28)
\end{aligned}$$

Only $G+\sqrt{SSE} \geq F \geq G-\sqrt{SSE}$ (equation (14.28)) is consistent with real numbers. This equation then gives a upper and lower bound on the square root of the energy of the candidate best match block. The SSE, which is expensive to compute, is only computed for pixels which satisfy the above inequality. It is tempting to consider,

$$G^2 + 2G\sqrt{SSE} + SSE \geq F^2 \geq G^2 - 2G\sqrt{SSE} + SSE \tag{14.29}$$

Since for positive numbers $a \geq b \Rightarrow a^2 \geq b^2$, however, one cannot guarantee that the lower bound of equation (14.28) will be positive. Equation (14.29) apparently expresses the bounding equation as a bound on the energy of the best match blocks. It has the further advantage of being easier to compute since fewer square root operations are required.

It should be noted that the same method employed by the SEA to pre-calculate the sumnorms, or sum of pixels within a block, can be used by the proposed algorithm. The only modification is before the sumnorms are calculated as in the traditional SEA the pixels are replaced by their squared values. A further speedup for motion estimation involves pre-calculating the energies for both frames since the search blocks now overlap.

14.4.3 Experimental results

The result of applying the suggested MSE based SEA algorithm is shown in figure 14.2. The average motion vector is measured to be $(1.2, \ -0.70)$. The result of applying the MAD based SEA to the standard test images is shown in figure 14.3. The average motion vector is measured to be $(1.2, \ -0.70)$. Note that the two algorithms perform similarly in measuring the average motion. The parameters for both algorithms were a search block of 9×9 and a search area of 5×5. This provided the best compromise between performance and time taken to generate the motion field.

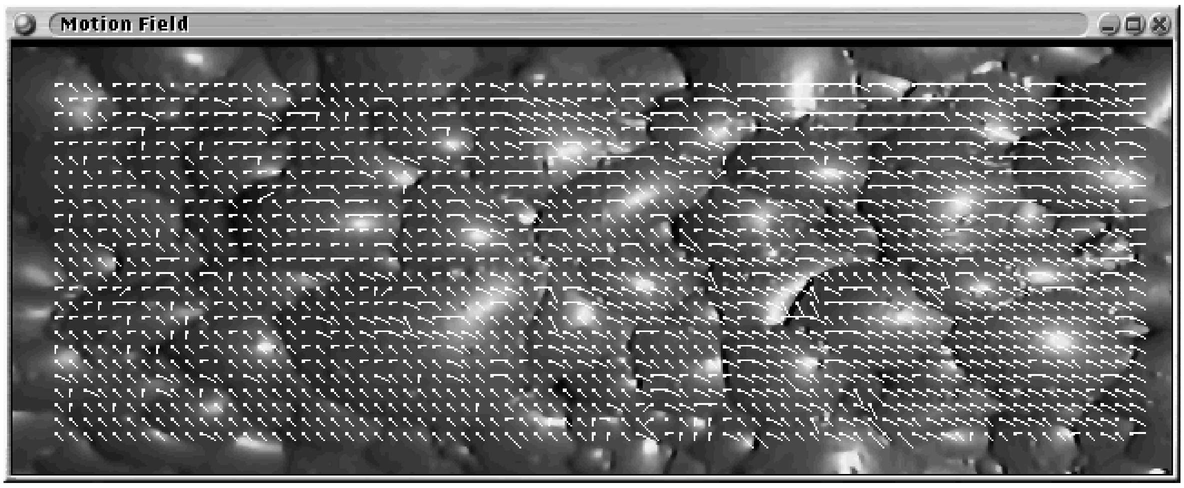
14.4.4 Discussions

The proposed algorithm provides the same results as the full search algorithm but with fewer computations. The algorithm places bounds on the energy of the candidate best match blocks. This allows the search space to be reduced each time a better match is found without sacrificing accuracy. The algorithm is suitable for motion estimation (and its use of small search areas) and searching the entire frame, if needed, for temporal compression. The expected savings in computations can be expected to increase as the search window and size of the blocks increases. This algorithm may useful for extending the Pixel Tracing algorithm (see section 14.3) to allow for motion in any direction while reducing the number of computations involved.

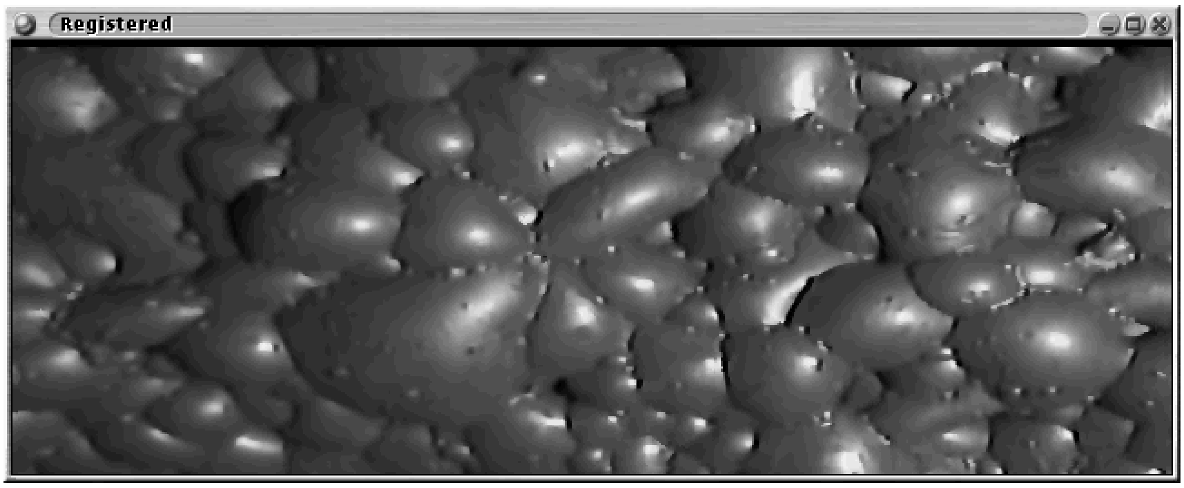
14.5 Block Motion Estimation with regularised flow field

This is a merging of the Horn and Schunk smoothing of the flow field (see section 15.1) with the robustness of block motion estimation. This is an attempt to provide a correlated motion field for block matching using small blocks. This attempts to provide speed (by using small blocks) while still ensuring a correlated motion field. A correlated motion field ensures that neighbouring pixels will have similar motion. The theoretical justification is as follows, one attempts to minimise the block mis-match error coupled with a smoothness constraint.

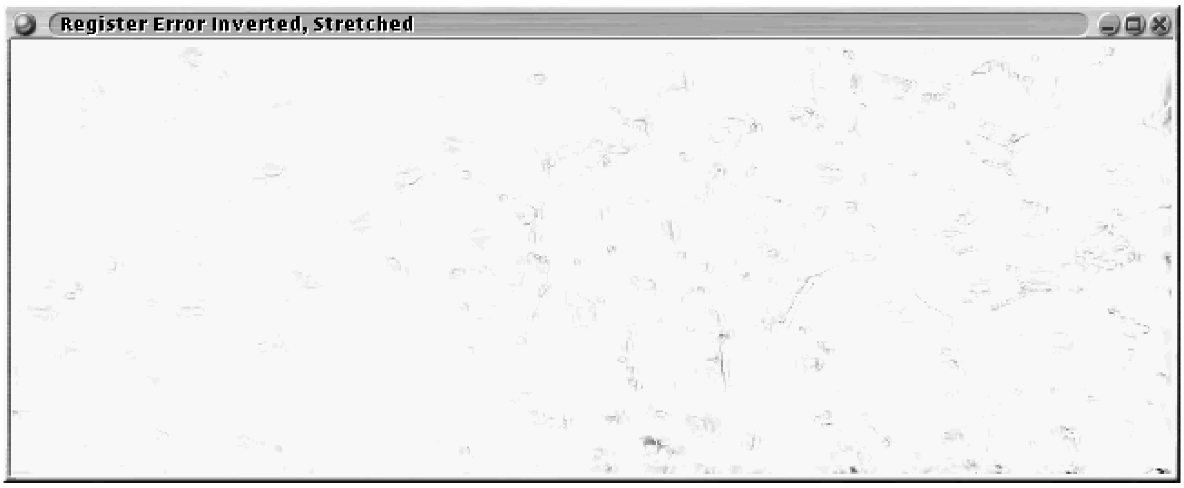
$$\varepsilon = \iint dx\,dy\ \left(MSE + \lambda\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right)\right) \tag{14.30}$$



(a) An example motion field due to MSE based SEA. Motion field scaled by 4. Average motion $(1.2, \ -0.70)$.

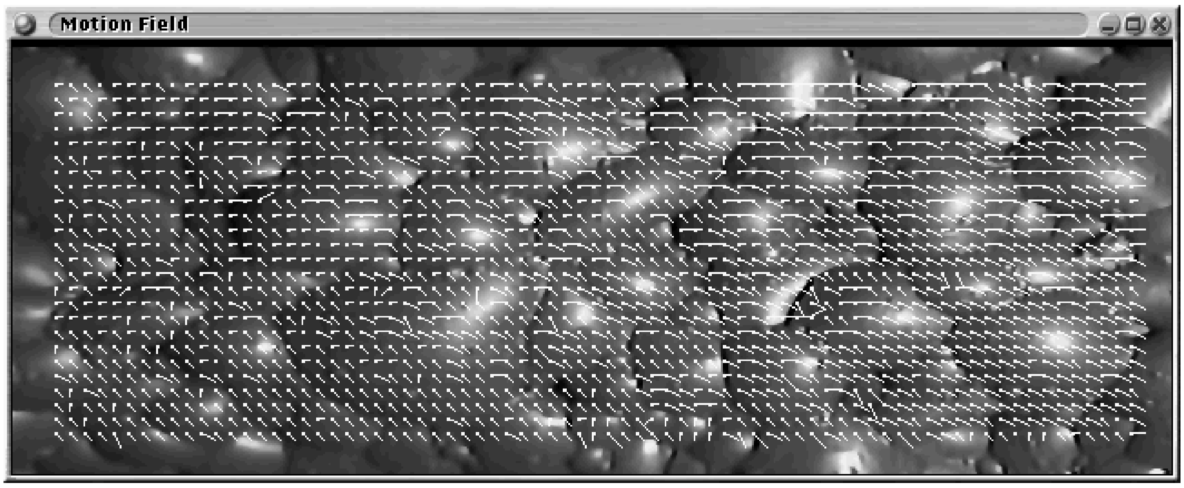


(b) Registered image.

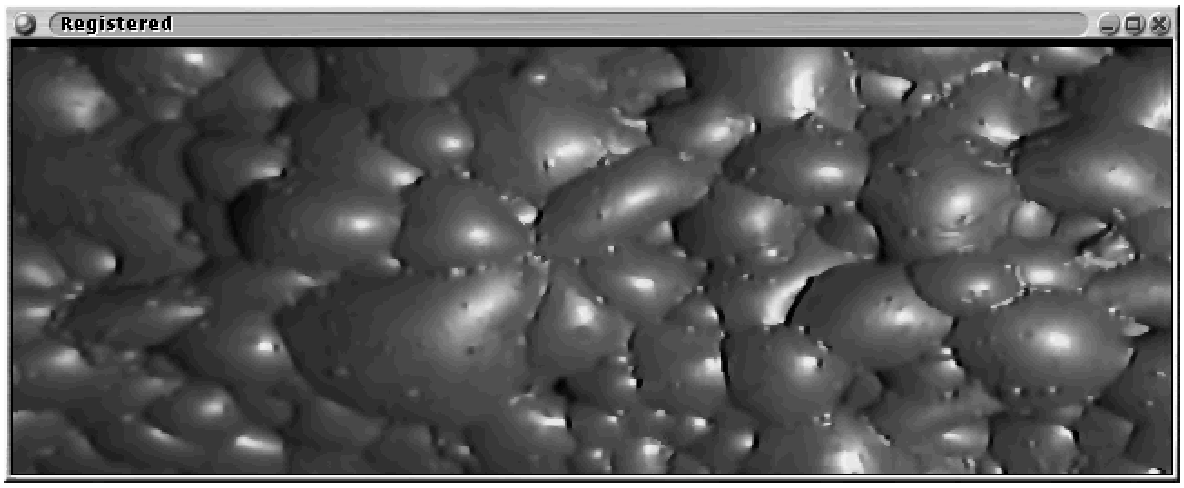


(c) Registered error, stretched. Energy in error image is 2 208 070

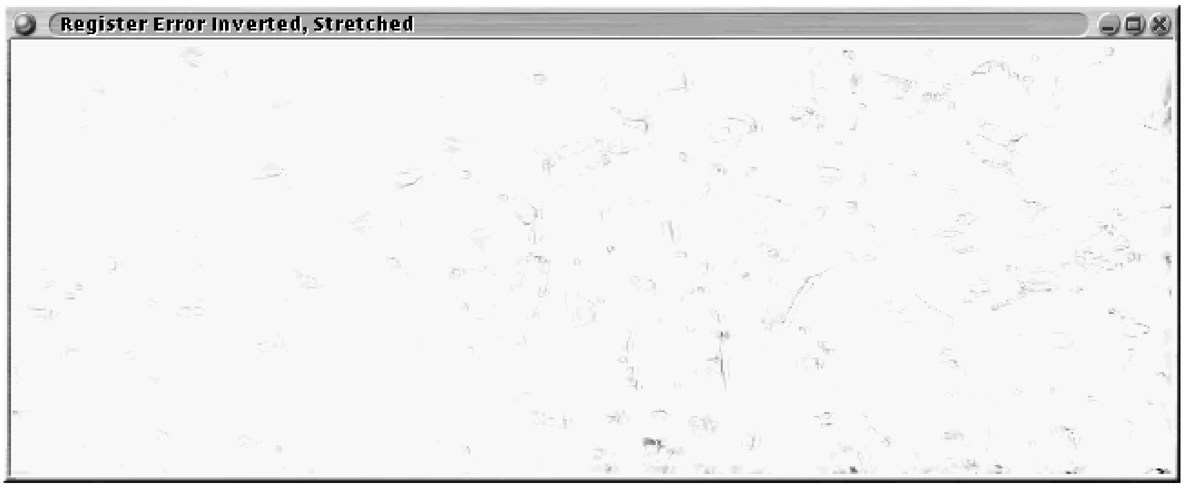
FIGURE 14.2 Motion estimation via MSE based SEA.



(a) An example motion field due to SEA. Motion field scaled by 4. Average motion $(1.2, -0.70)$.



(b) Registered image.



(c) Registered error, stretched. Energy in error image is 2 148 359.

FIGURE 14.3 Motion estimation via SEA.

Note that $u(x,\ y)$ is the x component of the motion vector at $(x,\ y)$ and $v(x,\ y)$ is the y component. ε is the total error to be minimised and depends on the block mis-match error and a smoothness of motion vector field term. Using the calculus of variations [46, 28],

$$\begin{aligned}
\varepsilon &= \iint dx\, dy\ \left(MSE + \lambda\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right)\right) \\
&= \iint dx\, dy\ F(u, v, u_x, u_y, v_x, v_y, x, y) \\
&= \iint dx\, dy\ F(\overline{u} + \epsilon\eta(x,y),\ \overline{v} + \phi\beta(x,u),\ \overline{u_x} + \epsilon\eta_x(x,y),\ \overline{v_x} + \phi\beta_x(x,u), \\
&\qquad\qquad \overline{u_y} + \epsilon\eta_y(x,y),\ \overline{v_y} + \phi\beta_y(x,u),\ x,\ y)
\end{aligned}$$

If the integral is minimum at $F(\overline{u}, \overline{v}, \overline{u_x}, \overline{v_x}, \overline{u_y}, \overline{v_y}, x, y)$, then slightly away at $u = \overline{u} + \epsilon\eta(x,y)$ and $v = \overline{v} + \phi\beta(x,y)$ for some defined functions $\overline{u}, \overline{v}, \eta$ and β, the integral now only depends on ϵ and ϕ. This then implies that,

$$\begin{aligned}
\left.\frac{\partial\varepsilon}{\partial\epsilon}\right|_{\epsilon=0} &= 0 \\
\left.\frac{\partial\varepsilon}{\partial\phi}\right|_{\phi=0} &= 0
\end{aligned}$$

Differentiating under the integral yields,

$$\begin{aligned}
\frac{\partial F}{\partial\epsilon} &= \frac{\partial F}{\partial u}\frac{\partial u}{\partial\epsilon} + \frac{\partial F}{\partial u_x}\frac{\partial u_x}{\partial\epsilon} + \frac{\partial F}{\partial u_y}\frac{\partial u_y}{\partial\epsilon} \\
&= \frac{\partial F}{\partial u}\eta + \frac{\partial F}{\partial u_x}\eta_x + \frac{\partial F}{\partial u_y}\eta_y
\end{aligned}$$

and,

$$\begin{aligned}
\frac{\partial F}{\partial\phi} &= \frac{\partial F}{\partial v}\frac{\partial v}{\partial\phi} + \frac{\partial F}{\partial v_x}\frac{\partial v_x}{\partial\phi} + \frac{\partial F}{\partial v_y}\frac{\partial v_y}{\partial\phi} \\
&= \frac{\partial F}{\partial v}\beta + \frac{\partial F}{\partial v_x}\beta_x + \frac{\partial F}{\partial v_y}\beta_y
\end{aligned}$$

or,

$$\begin{aligned}
\left.\frac{\partial \varepsilon}{\partial \epsilon}\right|_{\epsilon=0} &= \iint dx\,dy\,\left(\frac{\partial F}{\partial u}\right)_{u=\overline{u}} \eta + \left(\frac{\partial F}{\partial u_x}\right)_{u_x=\overline{u_x}} \eta_x + \left(\frac{\partial F}{\partial u_y}\right)_{u_y=\overline{u_y}} \eta_y = 0 \\
\left.\frac{\partial \varepsilon}{\partial \phi}\right|_{\phi=0} &= \iint dx\,dy\,\left(\frac{\partial F}{\partial v}\right)_{v=\overline{v}} \beta + \left(\frac{\partial F}{\partial v_x}\right)_{v_x=\overline{v_x}} \beta_x + \left(\frac{\partial F}{\partial v_y}\right)_{v_y=\overline{v_y}} \beta_y = 0
\end{aligned}$$

Integrating by parts leads to,

$$\iint dx\,dy\,\frac{\partial F}{\partial u_x}\eta_x = \left(\int dy\,\frac{\partial F}{\partial u_x}\eta\right) - \left(\iint dx\,dy\,\eta\frac{\partial}{\partial x}\frac{\partial F}{\partial u_x}\right)$$

$$\left(\iint dx\,dy\,\left(F_u - \frac{\partial}{\partial x}\frac{\partial F}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial F}{\partial u_y}\right)\eta\right) + \left(\int dy\,\frac{\partial F}{\partial u_x}\eta\right) + \left(\int dx\,\frac{\partial F}{\partial u_y}\eta\right) = 0$$

By careful choice of η, $\left(\int dy \frac{\partial F}{\partial u_x}\eta\right)$ and $\left(\int dx \frac{\partial F}{\partial u_y}\eta\right)$ can be made to disappear, leaving:

$$\left(\iint dx\,dy\,\left(F_u - \frac{\partial}{\partial x}\frac{\partial F}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial F}{\partial u_y}\right)\eta\right) = 0$$

Choosing, say,

$$\begin{aligned}
\eta(x,y) &= \omega(x,y)\left(F_u - \frac{\partial}{\partial x}\frac{\partial F}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial F}{\partial u_y}\right) \\
\omega(x,y) &> 0
\end{aligned}$$

Implies,

$$F_u - \frac{\partial}{\partial x}\frac{\partial F}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial F}{\partial u_y} = 0$$

Which is, formally, the Euler equation for this system [46] [28][pg. 497]. The equation for ϕ can be shown to be,

$$F_v - \frac{\partial}{\partial x}\frac{\partial F}{\partial v_x} - \frac{\partial}{\partial y}\frac{\partial F}{\partial v_y} = 0$$

It can then be shown that the pair of equations describing the evolution of u and v are,

$$MSE_u - 2\alpha\nabla\cdot\nabla u = 0$$
$$MSE_v - 2\alpha\nabla\cdot\nabla v = 0$$

Since the Laplacian in the discrete arena can be formulated as $\overline{u} - u$ where $\overline{u}$ is now the mean u (excluding the centre pixel of the neighbourhood used to compute the mean) [68]. The iterative version of the above equations now becomes,

$$u_{k+1} = \frac{-MSE_u + 2\alpha\overline{u}}{2\alpha}$$
$$v_{k+1} = \frac{-MSE_v + 2\alpha\overline{v}}{2\alpha}$$

Where,

$$\overline{u} = \frac{1}{6}\left(u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}\right) + \frac{1}{12}\left(u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}\right)$$
$$\overline{v} = \frac{1}{6}\left(v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}\right) + \frac{1}{12}\left(v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}\right)$$

Where, i denotes the x co-ordinate, j the y and k time. The MSE is usually defined as follows [9],

$$MSE(x,y,u,v) = \frac{1}{MN}\sum_{n=-\frac{N}{2}}^{\frac{N}{2}}\sum_{m=-\frac{M}{2}}^{\frac{M}{2}} h(x,y)\left(I(x+m,y+n,t) - I(x+m+u,y+n+v,t+\Delta t)\right)^2 \tag{14.31}$$

Usually, the weight $h(x,y)$ is one everywhere on its region of support $N \times M$. Weighted MSE requires that $h(x,y)$ be a smoothing (or low-pass filter) kernel. The iterative equations then become,

$$u_{k+1} = \left(\frac{-1}{\alpha MN} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \sum_{m=-\frac{M}{2}}^{\frac{M}{2}} h(x,y) \left(I(x+m, y+n, t) - I(p, q, t+\Delta t) \right) \frac{\partial I(p,q,t+\Delta t)}{\partial p} \right) + \overline{u} \tag{14.32}$$

$$v_{k+1} = \left(\frac{-1}{\alpha MN} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \sum_{m=-\frac{M}{2}}^{\frac{M}{2}} h(x,y) \left(I(x+m, y+n, t) - I(p, q, t+\Delta t) \right) \frac{\partial I(p,q,t+\Delta t)}{\partial q} \right) + \overline{v} \tag{14.33}$$

where,

$$p = x + m + u_k \qquad \text{and,} \tag{14.34}$$

$$q = y + n + v_k \tag{14.35}$$

The differences in the MSE_u and MSE_v terms are smoothed by convolving with a 2D function of support $M \times N$. The choice of region of support for the mean computation (and the kernel) becomes another algorithm parameter. This would then affect the approximation used for computing the Laplacian. To maintain consistency it is natural to select the region of support for the smoothing filter for u_k and v_k to be the same as the region of support for MSE_u and MSE_v. For simplicity, the kernel may be chosen to be one within it's regions of support.

14.5.1 Experimental results

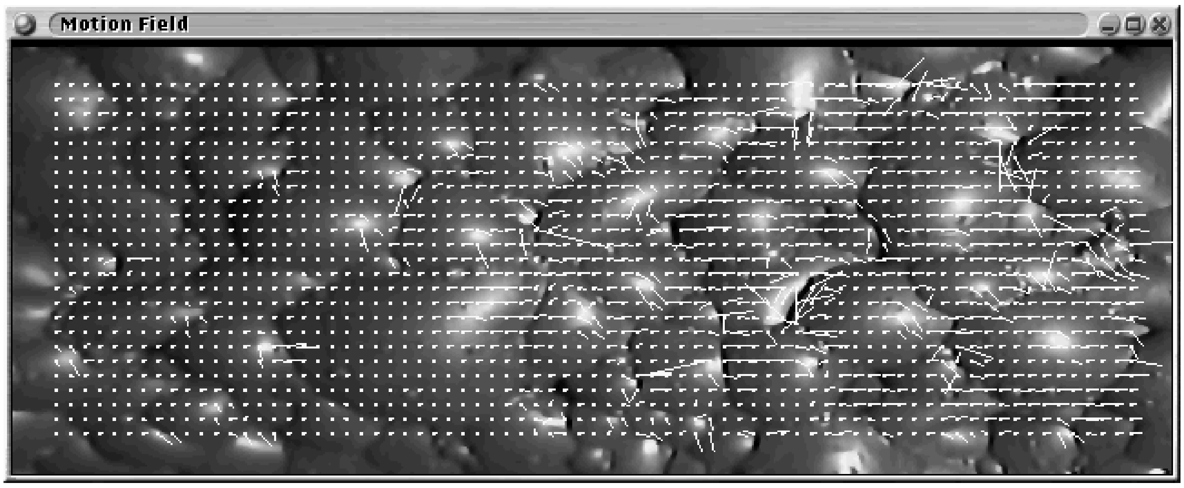
The algorithm was applied to the standard test images (see figure 13.1). The result of the test is shown in figure 14.4. The block size used was 15×15, the α value was 100, and the number of iterations 8. The parameters were chosen to provide the best performance while maintaining stability, and minimising the time needed to generate the motion field. The average motion vector is measured as $(0.77, \ -0.12)$. The energy in the error image is measures as 22 174 896. The performance is seen to be poor. It is likely the gradient descent used is becoming trapped in a local minimum.

14.6 Motion from Cluster matching

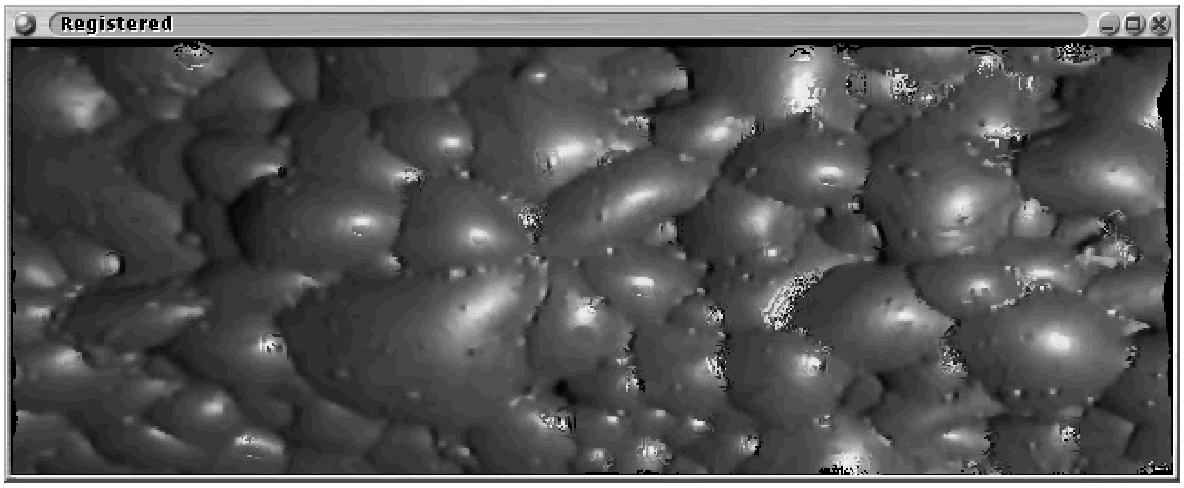
This algorithm was described by Kottke and Sun [84]. The algorithm is divided into a clustering phase and, once stable clusters are formed, a cluster matching and thus displacement estimation phase. Essentially the clustering phase of the algorithm is a variant of K-means clustering [94].

The clustering phase of the algorithm proceeds as follows:

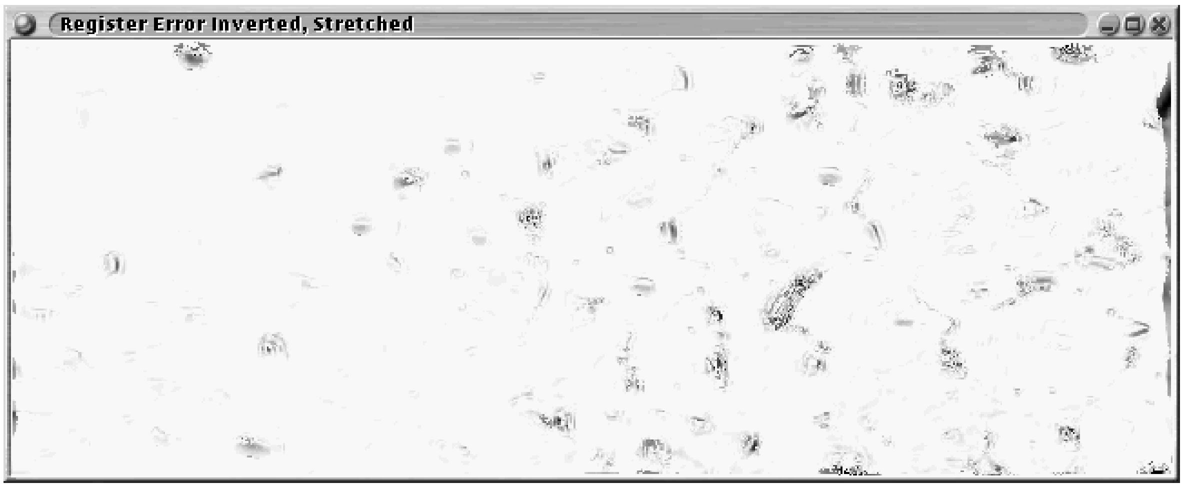
(i) Decide on the number of clusters to use.



(a) An example motion field due to BME with regularised flow field. Motion field scaled by 4. Average motion $(0.77, -0.12)$.



(b) Registered image.



(c) Registered error, stretched. Energy in error image is 22 174 896.

FIGURE 14.4 Motion estimation via BME with regularized flow field..

(ii) Tile the image with blocks until all clusters allocated.

(iii) Each cluster has three features, namely, the mean of the x and y positions of the pixels in the cluster and the mean of the intensities of the pixels in the cluster, or (m_x^j, m_y^j, m_z^j) respectively.

(iv) Iterate until the largest shift of the cluster centres fall below a certain threshold.

(v) Each pixel p^i is assigned to cluster k^j if the distance in the following in feature space is minimum,

$$\epsilon^{ij} = \left(\mathbf{p}^i - \mathbf{m}^j)\mathbf{W}^j(\mathbf{p}^i - \mathbf{m}^j\right) \tag{14.36}$$

(vi) $\mathbf{W}^j$ is a diagonal matrix with the following weights on the main diagonal w_x^j, w_y^j and w_z^j.

(vii) The weights are:

$$w_x^j = \frac{c^j}{\sigma_x^j} \tag{14.37}$$

$$w_y^j = \frac{c^j}{\sigma_y^j} \tag{14.38}$$

$$w_z^j = \frac{c^j}{\sigma_z^j} \tag{14.39}$$

(viii) The constant c is set as follows,

- If in a noise free region $\sigma_z^j == 0$ set, $w_x^j = w_y^j = 0$ and $w_z^j = 1$.
- Otherwise set $c^j = \sigma_x^j \sigma_y^j \sigma_z^j$
- σ_ϕ^j is defined as the unbiased standard deviation in the ϕ direction and is defined as usual,

$$\sigma_\phi^j = \sqrt{\frac{1}{N_j - 1} \sum_{\phi \epsilon k^j}^{N_j} (\phi - m_\phi^j)^2} \tag{14.40}$$

The cluster matching phase is as follows:

(i) The following features are added to each cluster's feature description.

- A measure of the average grey-scale difference,

$$m_s^j = \sum_{x,y \epsilon k_j} I(x,y,t) - I(x,y,t+\Delta t) \tag{14.41}$$

- A shape descriptor, the sum of the squared central moments,

$$m_m^j = \sum_{x,y \epsilon k_j} \left[(x-\overline{x^j})^2 + (y-\overline{y^j})^2\right] I(x,y) \tag{14.42}$$

Where $\overline{x^j}$ and $\overline{y^j}$ are the weighted central moments defined as follows,

$$\overline{x^j} = \frac{\sum_{x,y \epsilon k_j} x I(x,y)}{\sum_{x,y \epsilon k_j} I(x,y)} \tag{14.43}$$

$$\overline{y^j} = \frac{\sum_{x,y \epsilon k_j} y I(x,y)}{\sum_{x,y \epsilon k_j} I(x,y)} \tag{14.44}$$

(ii) The match criterion is the weighted squared Euclidean distance:

$$D^{jl} = \left(\mathbf{b}^j - \mathbf{c}^l\right) \mathbf{U} \left(\mathbf{b}^j - \mathbf{c}^l\right) \tag{14.45}$$

Here,

- $\mathbf{c}^l$ is the cluster which minimises the match between clusters $\mathbf{b}^l$ and $\mathbf{c}^l$.
- $\mathbf{U}$ is a 5×5 with main diagonal elements u_x, u_y, u_z, u_m and u_s. These are the cluster weights and described later.

(iii) The match process iterates between weight fixing and cluster matching until the total displacement $D_T = \sum_{j=1}^{K^l} D^{jl}$ does not change between iterations. For each iteration the weights are recalculated as follows,

$$u_y = \frac{u_x \sum_{j=1}^{K^j} \left(m_x^j - m_x^l\right)^2}{\sum_{j=1}^{K^j} \left(m_y^j - m_y^l\right)^2} \tag{14.46}$$

$$u_z = \frac{u_x \sum_{j=1}^{K^j} \left(m_x^j - m_x^l\right)^2}{\sum_{j=1}^{K^j} \left(m_z^j - m_z^l\right)^2} \tag{14.47}$$

$$u_m = \frac{u_x \sum_{j=1}^{K^j} \left(m_x^j - m_x^l\right)^2}{\sum_{j=1}^{K^j} \left(m_m^j - m_m^l\right)^2} \tag{14.48}$$

$$u_s = \frac{u_x \sum_{j=1}^{K^j} \left(m_x^j - m_x^l\right)^2}{\sum_{j=1}^{K^j} \left(m_s^j - m_s^l\right)^2} \tag{14.49}$$

$$u_x = \left[\sum_{j=1}^{K^j} \left(m_y^j - m_y^l\right)^2\right]^{\frac{1}{5}} \left[\sum_{j=1}^{K^j} \left(m_z^j - m_z^l\right)^2\right]^{\frac{1}{5}} \left[\sum_{j=1}^{K^j} \left(m_m^j - m_m^l\right)^2\right]^{\frac{1}{5}} \left[\sum_{j=1}^{K^j} \left(m_s^j - m_s^l\right)^2\right]^{\frac{1}{5}} \left[\sum_{j=1}^{K^j} \left(m_x^j - m_x^l\right)^2\right]^{-\frac{4}{5}} \tag{14.50}$$

(iv) Generating a threshold,

$$\tau = \alpha \max_j \|m_s^j\| \qquad \cdots \qquad \alpha \text{ a user supplied parameter} \tag{14.51}$$

The motion vector for each cluster is then,

$$\begin{cases} d_x = m_x^l - m_x^j,\, d_y = m_y^l - m_y^j, & \text{if} \tau < \|m_s^j\| \\ d_x = d_y = 0 & \text{otherwise} \end{cases} \tag{14.52}$$

14.6.1 Experimental results

The algorithm was applied to the usual pair of test froth images with the following results as shown in figure 14.5. The initial number of clusters chosen were 100 with $\alpha = 0.1$. These parameters provided the best performance. The average motion vector was measured as $(1.7, -0.33)$. The energy in the error image is 27 572 317.

14.6.2 Discussion

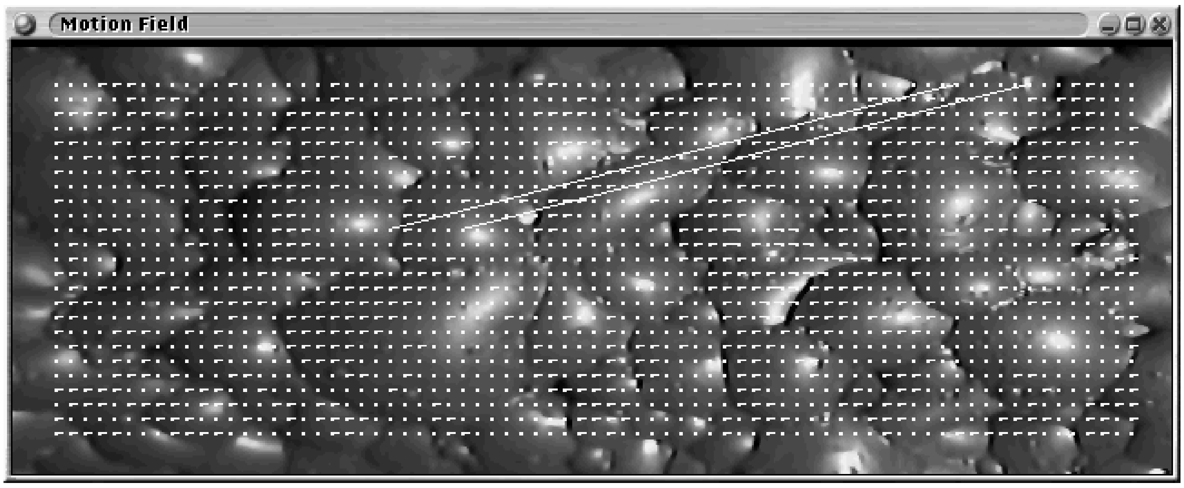
The advantages of using more clusters than there are bubbles is that deformation can be better handled. The further advantage of this algorithm is that the search space is greatly reduced from examining every pixel with other methods to examining a (in comparison) small number of clusters. The disadvantage of this method is that, effectively, a sparse motion field is generated. The method replicates the motion vector of the centroid of the region to each pixel within the cluster. This replication cannot capture the subtleties of the motion of pixels within each bubble. This explains the poor performance.

14.7 Motion from Watershed segmentation

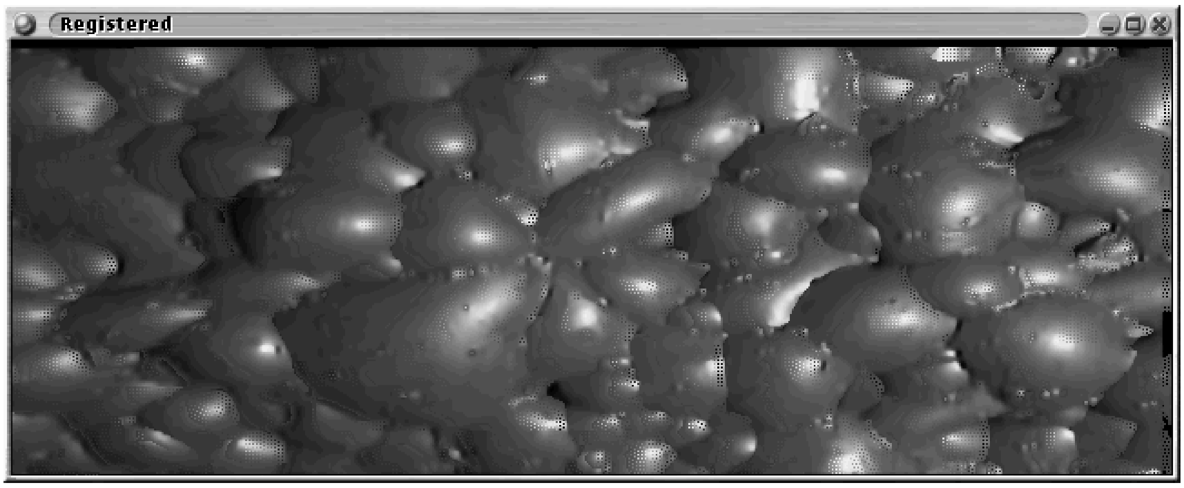
The watershed algorithm is used after some denoising to segment the individual bubbles from the froth. Given that the segmentation process has then occurred it is natural to attempt to perform motion estimation using these segmented regions. Several features are computed from the the bubbles. This algorithm is inspired by the Motion from Cluster matching algorithm described in section 14.6.

The algorithm proceeds as follows:

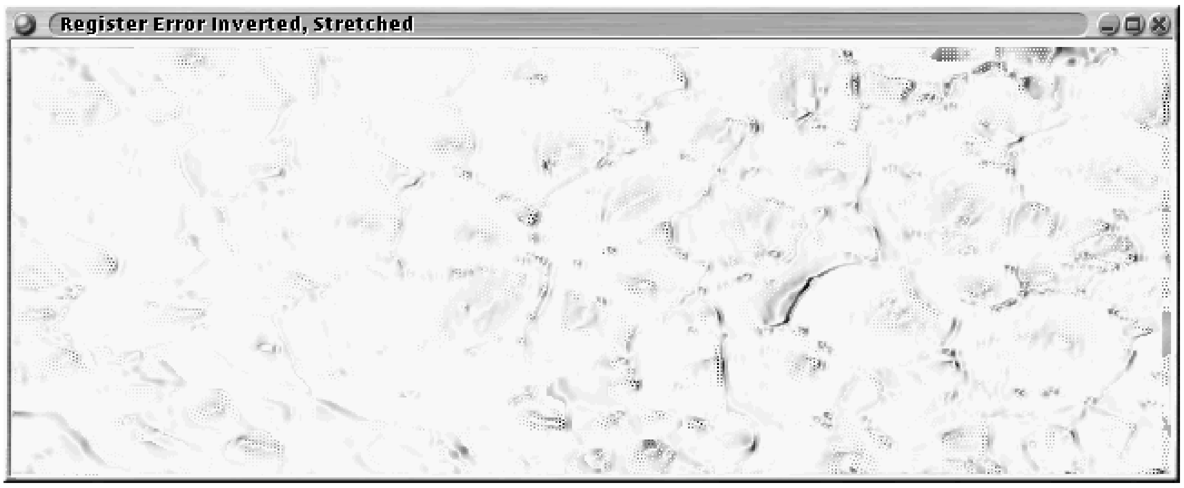
(i) After Watershed segmentation a number of features are calculated on each Watershed region.



(a) An example motion field due to Kottke and Sun. Motion field scaled by 4. Average motion $(1.7, -0.33)$.



(b) Registered image.



(c) Registered error, stretched. Energy in error image is 27 572 317.

FIGURE 14.5 Motion estimation via Kottke and Sun cluster matching.

(a) The grey level mean.

(b) The grey level standard deviation.

(c) The standard deviation of the x coordinates of the pixels.

(d) The standard deviation of the y coordinates of the pixels.

(e) The number of pixels.

(ii) The features (from both frames) are jointly normalised to unit variance and zero mean.

(iii) The the best match region in frame $t + \Delta t$ is found by finding a q which minimises:

$$ERROR = \left(R_t^p - R_{t+\Delta t}^q\right)^2 + \alpha((x_p - x_q)^2 + (y_p - y_q)^2)$$

Where p represents the search region in frame t, q the reference region(s) in frame $t + \Delta t$ and R is the feature vector. α is a composite factor, it consists of a scale factor (β) to ensure that the longest distance between any two regions is one, and incorporates a scale factor (γ) to allow for distance to have a larger or smaller weight as required.

14.7.1 Experimental results

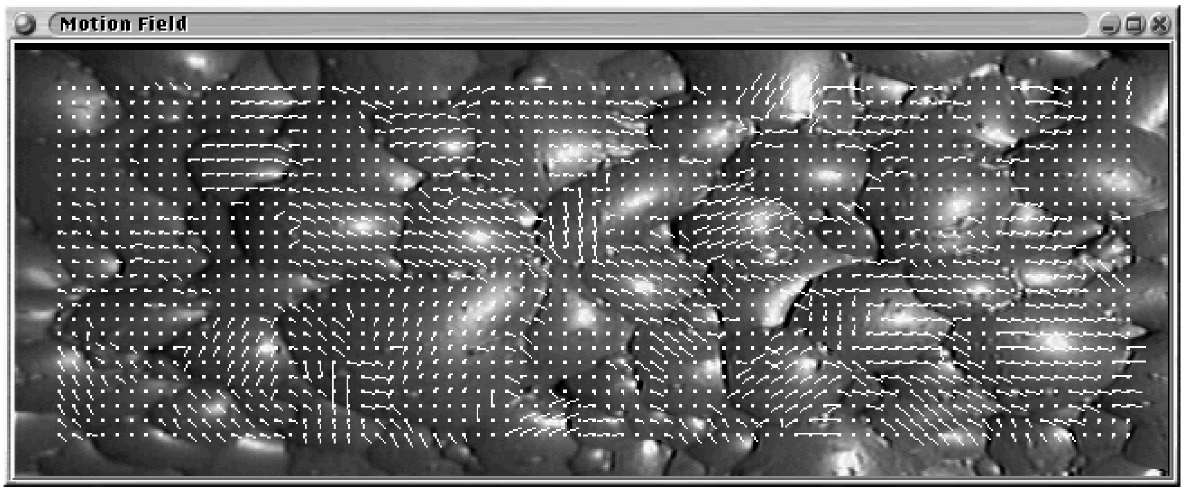
The algorithm was applied to the usual two froth images (as shown in figure 13.1). A marker based watershed was used to segment the images. The markers used were the domes of the image [144]. The domes of an image are found by subtracting the original image from the grey level reconstruction of the original image less some integer [144]. The parameters used were, search radius of 40 pixels and the distance part of the error measure (γ) is weighted by 10. The parameters chosen penalise very long motion vectors and provide for the best results. The results of the application of this algorithm is shown in figure 14.6. The average motion vector was measured as $(1.3,\ -0.53)$. The energy in the error image is 27 411 424.

As an aid to visualisation of the operation of the algorithm consider the motion and reference images watershed segmented after the usual smoothing as shown in figure 14.7.

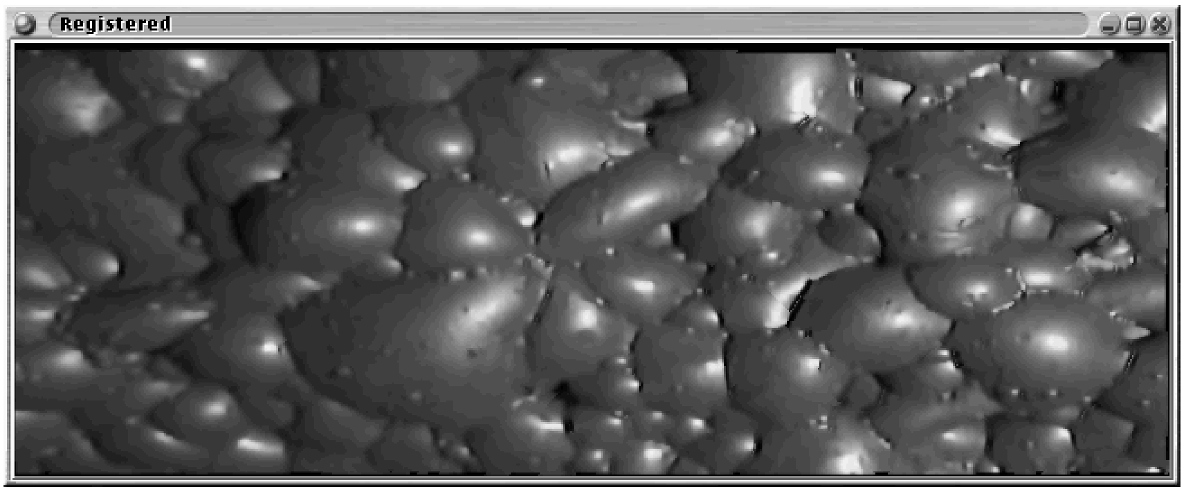
14.7.2 Discussion

This algorithm is very dependent on the segmentation. Over-segmentation generates more regions which allows for finer estimation of motion. Under-segmentation (in the case of froth this would be two or more bubbles regarded as a single bubble), is not desirable since the final motion of the region would be regarded as a weighted average of the motion of the individual bubbles.

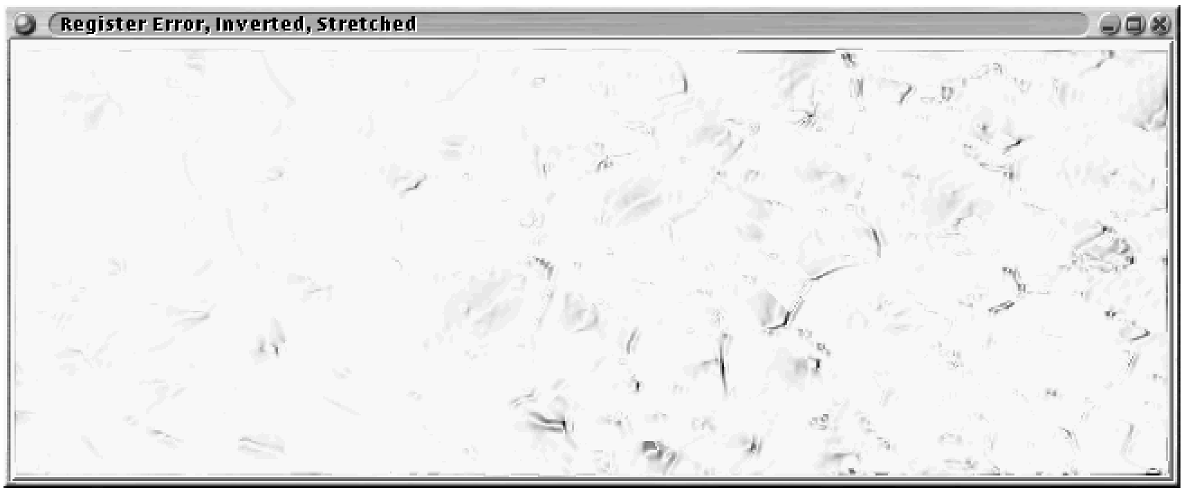
The method used for evaluating the algorithms relies on the computation of a *dense* motion field. The method under consideration only generates vectors at the bubble centroids. A centroidal vector is then replicated to cover the rest of the pixels in a region. The weakness of this algorithm lies in the replication of the motion vectors. The sparse motion field generated, is hampered by the lack of correlation between the various regions. This is the reason for the poor quantitative performance.



(a) An example motion field due to Watershed based Motion Estimation. Motion field scaled by 4. Average motion $(1.3, -0.53)$.

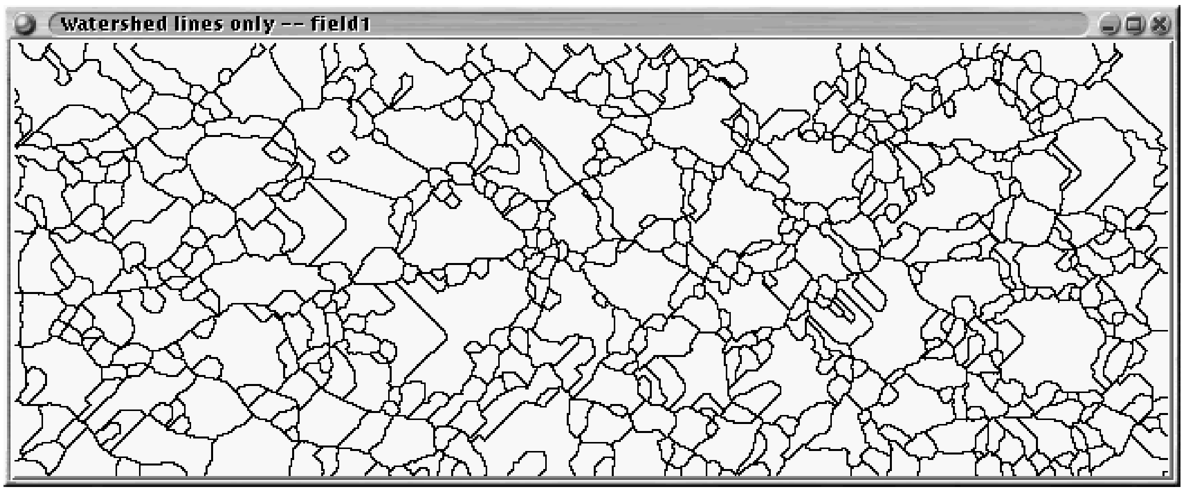


(b) Registered image.

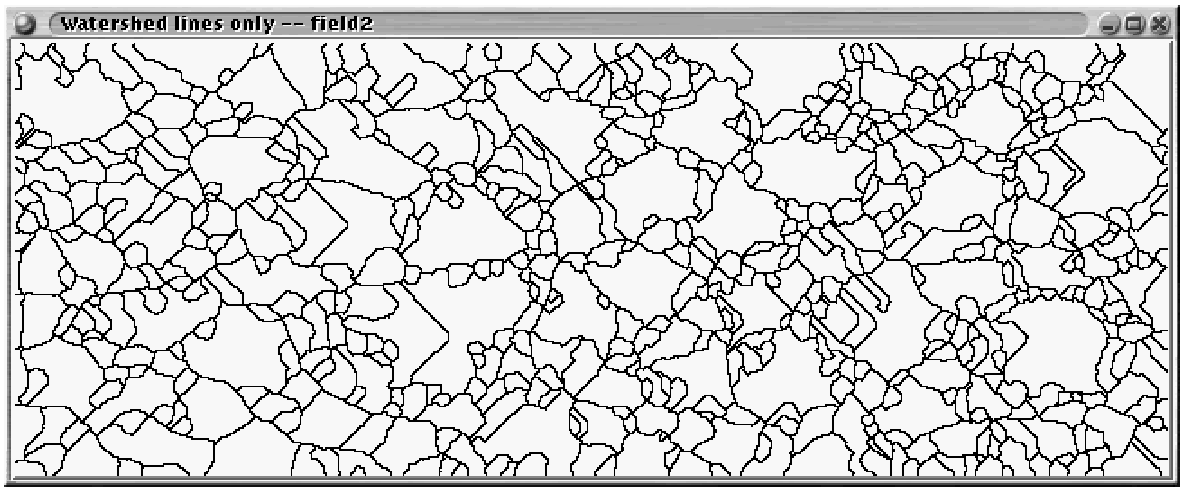


(c) Registered error, stretched. Energy in error image is 27 411 424.

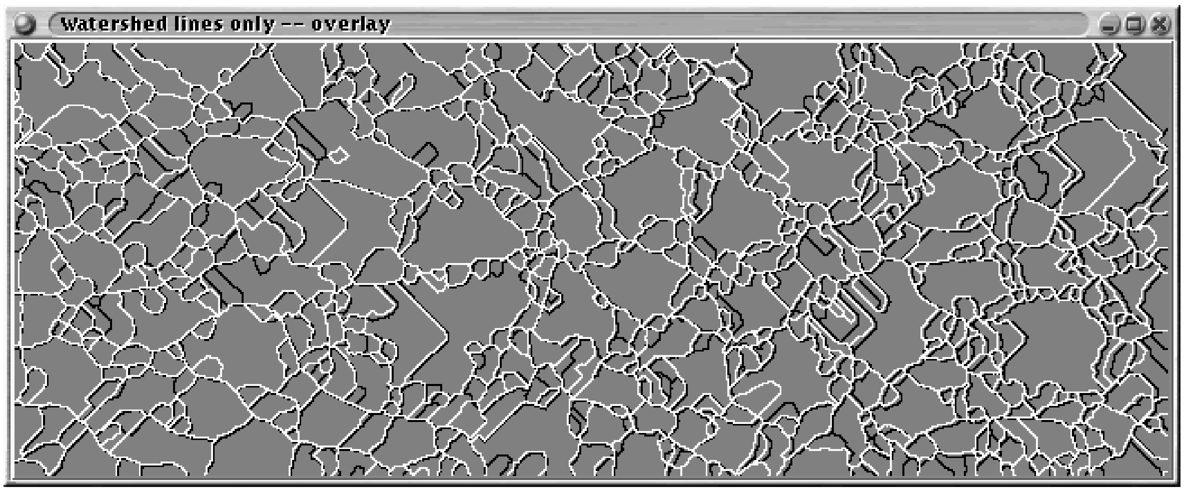
FIGURE 14.6 Watershed based motion estimation.



(a) Watershed segmented, even field at time t.



(b) Watershed segmented, even field at time $t + \Delta t$



(c) Watershed lines overlaid. White denotes field at time t. Black denotes field at time $t + \Delta t$.

FIGURE 14.7 Watershed lines, individually and overlaid, of the test images.

Chapter 15

Gradient based dense optical flow

This chapter examines some of the gradient-based optical flow algorithms in the literature. The suitability of the approach for motion estimation of froth images is examined. Optical flow is usually defined as the apparent motion of the brightness patterns within an image. Other possible definitions exist such as optical flow in feature space (for example, multiple constraint optical flow [138]).

Horn and Schunk [68] first described a solution to the problem of estimating a dense optical flow field via gradient methods. Since the optical flow constraint equation is under-constrained, a solution requires regularisation. Different forms of regularisation are possible and are extensively explored in the literature [2, 11, 117]. The notion of multiple constraint optical flow was developed to overcome the need for regularisation (see Tistarelli [138]).

Div-Curl stochastic models for optical flow [54, 132] regularise the usual optical flow constraint equation by using a div-curl spline. Suter [132] then proves equivalence to Horn and Schunks's optical flow algorithm.

In all cases examined, the fields used are first de-interlaced and the even field extracted, then denoised by 4 iterations of Perona and Malik's Anisotropic Diffusion algorithm [113] (see section 7.1), here the time step (γ) used is 0.2 and the K values determined using Canny's noise estimator as described by Perona and Malik. The choice of 4 iterations is determined by the tradeoff between removing noise (and smoothing away bubbles too small to be of significance) and merging bubbles best left separate.

15.1 Horn and Schunk

This is regarded as a standard work on gradient based optical flow. The algorithm [68] relates the intensity changes at a pixel to the motion at the pixel. Since the intensity of a point is the same at its position in one frame and its corresponding position in the next.

$I_x u + I_y v + I_t = 0$ is known as the Optical Flow constraint equation, and is derived by assuming that the pixel intensities between two frames are preserved. The pixel intensities in the next frame is expanded via a Taylor expansion as follows:

$$
\begin{aligned}
I(x,y,t) &= I(x+\Delta x, y+\Delta y, t+\Delta t)\\
I(x+\Delta x, y+\Delta y, t+\Delta t) &= I(x,y,t) + \frac{\partial}{\partial x}I(x,y,t)\Delta x + \frac{\partial}{\partial y}I(x,y,t)\Delta y + \frac{\partial}{\partial t}I(x,y,t)\Delta t\\
0 &= \frac{\partial}{\partial x}I(x,y,t)\Delta x + \frac{\partial}{\partial y}I(x,y,t)\Delta y + \frac{\partial}{\partial t}I(x,y,t)\Delta t\\
I_x\Delta x + I_y\Delta y + I_t\Delta t &= 0\\
I_x\frac{\Delta x}{\Delta t} + I_y\frac{\Delta y}{\Delta t} + I_t\frac{\Delta t}{\Delta t} &= 0\\
I_x u + I_y v + I_t &= 0 \qquad \text{taking, } \lim \Delta t \to 0
\end{aligned} \tag{15.1}
$$

Note that u is the velocity component in the x co-ordinate and v in the y direction. Since the optical flow constraint equation is an equation in two unknowns, a second constraint is required to solve for $(u,\ v)$. Horn and Schunk do this by specifying that the motion vector field is smooth. In other words, the motion flow field has no sudden discontinuities. The extra constraint is then, $u_x^2 + u_y^2 + v_x^2 + v_y^2$, and must be minimised. In integral form the constraints are,

$$
\varepsilon = \iint dx\, dy\ \left(I_x u + I_y v + I_t\right)^2 + \alpha\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) \tag{15.2}
$$

and ε needs to be minimised. The calculus of variations is required to find a u and v minimising ε since both u and v vary according to $(x,\ y)$ [46]. The corresponding Euler equations [28, 46] are,

$$
\begin{aligned}
I_x^2 u + I_x I_y v &= \alpha \nabla\cdot\nabla u - I_x I_t\\
I_x I_y u + I_y^2 v &= \alpha \nabla\cdot\nabla v - I_x I_t
\end{aligned}
$$

Approximating the Laplacian by,

$$
\nabla\cdot\nabla u = \overline{u} - u,
$$

where $\overline{u}$ denotes the mean without the centre pixel $(x,\ y)$ in the neighbourhood, yields,

$$
\begin{aligned}
\left(\alpha + I_x^2\right) u + I_x I_y v &= \alpha\overline{u} - I_x I_t\\
I_x I_y u + \left(\alpha + I_y^2\right) v &= \alpha\overline{v} - I_x I_t
\end{aligned}
$$

This is a simple system which can be readily solved. The iterative update equations computes the new motion estimate at a pixel $\left(u^{(n+1)},\, v^{(n+1)}\right)$ as follows,

$$u^{(n+1)} = \overline{u}^{(n)} - \frac{I_x \left(I_x \overline{u}^{(n)} + I_y \overline{v}^{(n)} + I_t \right)}{\alpha + I_x^2 + I_y^2}$$
$$v^{(n+1)} = \overline{v}^{(n)} - \frac{I_y \left(I_x \overline{u}^{(n)} + I_y \overline{v}^{(n)} + I_t \right)}{\alpha + I_x^2 + I_y^2}$$

15.1.1 Experimental results

The algorithm was applied to the standard set of test images (see figure 13.1). The number of iterations used is 16 and $\alpha = 1$. These parameters are chosen to provide good performance. The result of the test is shown in figure 15.1. The average motion vector is measured as $(0.94, \ -0.55)$. The energy in the error image is 2 876 750.

15.1.2 Discussion

The smoothing of the motion vector field results in the motion of pixels on the bubbles boundaries being affected by the motion of neighbouring bubbles. The effect of noise on the motion estimation process is considered.

The effect of noise on gradient based optical flow

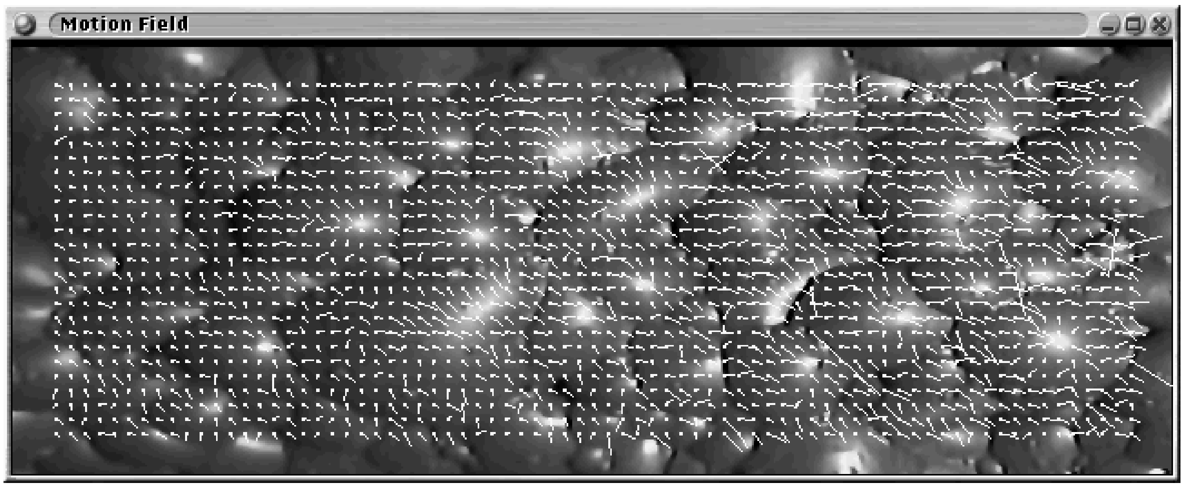
The optical flow constraint equation, formulated in the usual way, assumes that the intensity of pixels is conserved under motion (see Horn and Schunk's seminal paper [68] described in section 15.1). This implies,

$$f(x + u(x, y), y + v(x, y), t + \Delta t) = f(x, y, t) \tag{15.3}$$

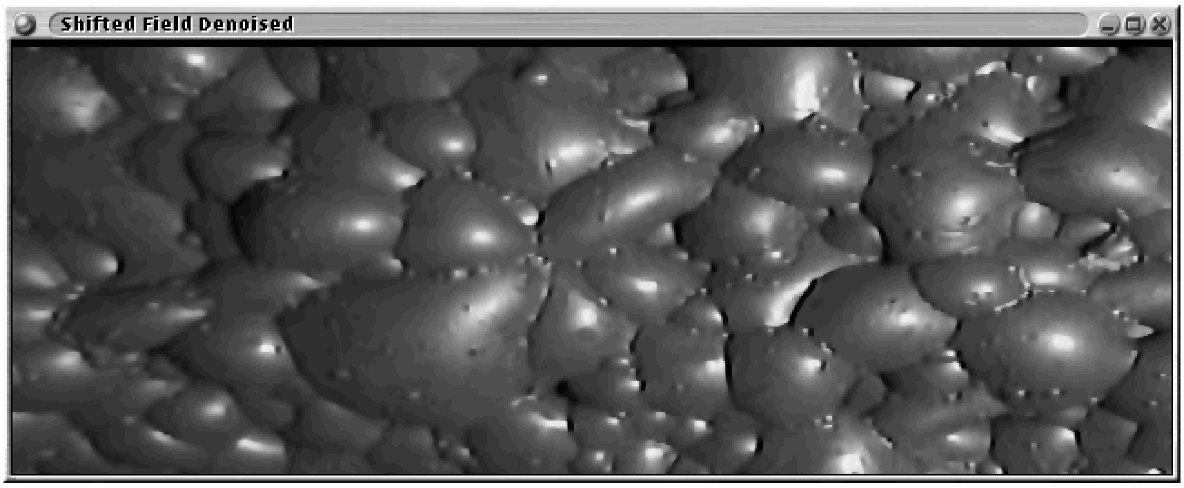
The presence of noise immediately confounds this assumption. The true state for additive noise, is closer to,

$$f(x + u(x, y), y + v(x, y), t + \Delta t) + \eta_1(x, y) = f(x, y, t) + \eta_2(x, y) \tag{15.4}$$

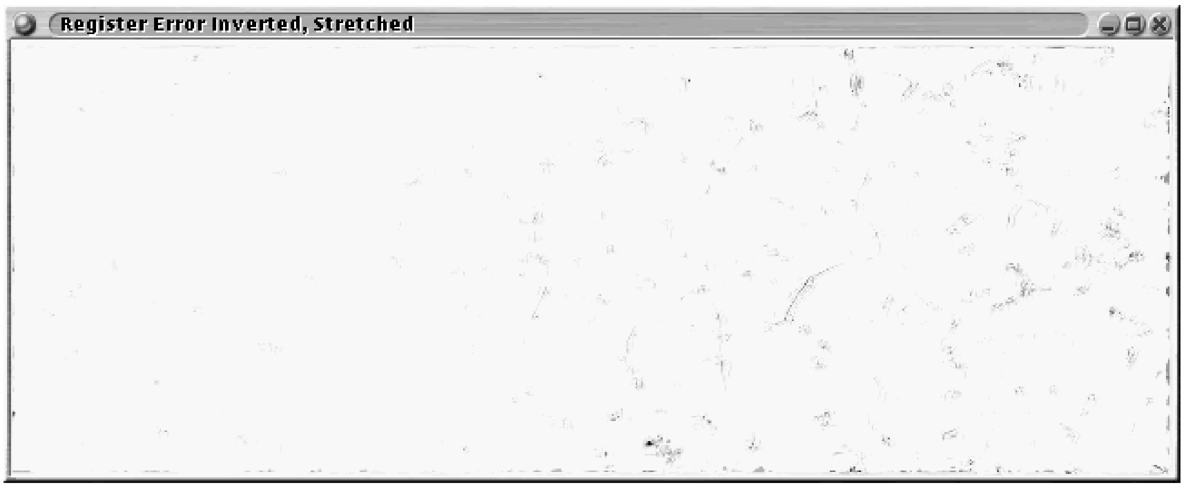
Where η_1 and η_2 are noise processes. Following the derivation in Horn and Schunk's paper [68], $f(x + u(x, y), y + v(x, y), t + \Delta t)$ is Taylor expanded, and the higher order terms truncated to form,



(a) Motion field, scaled by factor of 4. Average motion $(0.94,\ -0.55)$.



(b) Registered image.



(c) Registered error, stretched. Energy in error image is 2 876 750.

FIGURE 15.1 Optical flow via Horn and Schunk.

$$f(x,y,t) + \Delta x \frac{\partial}{\partial x} I(x,y,t) + \Delta y \frac{\partial}{\partial y} I(x,y,t) + \Delta t \frac{\partial}{\partial t} I(x,y,t) + \eta_1(x,y) = f(x,y,t) + \eta_2(x,y) \tag{15.5}$$

$$\Delta x \frac{\partial}{\partial x} I(x,y,t) + \Delta y \frac{\partial}{\partial y} I(x,y,t) + \Delta t \frac{\partial}{\partial t} I(x,y,t) = \eta_2(x,y) - \eta_1(x,y) \tag{15.6}$$

Under the assumption of zero mean noise,

$$E\{\Delta x \frac{\partial}{\partial x} I(x,y,t) + \Delta y \frac{\partial}{\partial y} I(x,y,t) + \Delta t \frac{\partial}{\partial t} I(x,y,t)\} = E\{\eta_2(x,y) - \eta_1(x,y)\} \tag{15.7}$$

$$= 0$$

Where $E\{\cdot\}$ denotes the expectation operator.

$$E\{\Delta x \frac{\partial}{\partial x} I(x,y,t)\} + E\{\Delta y \frac{\partial}{\partial y} I(x,y,t)\} + E\{\Delta t \frac{\partial}{\partial t} I(x,y,t)\} = 0 \tag{15.8}$$

The expectation operator denotes an averaging process [99, 93]. Since the problem is formulated in a three dimensional environment, averaging over time and the two spatial dimensions is possible. This provides some justification for the common practice of averaging gradient based motion estimates either spatially or temporally. Occasionally estimates are smoothed both spatially *and* temporally [68].

15.2 Div-Curl Optical Flow Methods

These were developed by Suter [132] and constrain the optical flow constraint equation ($I_x u + I_y v + I_t = 0$) using a div-curl spline. Essentially, this is a different form of regularising the optical flow constraint equation.

The divergence represents the smoothness of the vector field. The curl is dependent on the rotation and shear of the vector field.

$$\iint \left((I_x u + I_y v + I_t)^2 + \alpha \parallel \nabla \cdot \mathbf{v} \parallel^2 + \beta \parallel \operatorname{curl} \mathbf{v} \parallel^2 \right) dx\, dy \tag{15.9}$$

A motion vector field $(u,\ v)$ is found which minimises equation 15.9. The implementation involved three steps: creating a variational framework for the equations, solving the variational equations, and finally generating a numerical scheme for the solution.

Variational calculus may be used to find the Euler equations for the system,

$$\alpha u_{xx} + \beta u_{yy} + (\alpha - \beta) v_{xy} = I_x(I_x u + I_y v + I_t) \qquad \text{and,} \tag{15.10}$$

$$(\alpha - \beta) u_{xy} + \beta v_{xx} + \alpha v_{yy} = I_y(I_x u + I_y v + I_t) \tag{15.11}$$

Approximating [73],

$$u_{xx} \approx u_{(i-1,j)} - 2u_{(i,j)} + u_{(i+1,j)} \tag{15.12}$$

$$u_{yy} \approx u_{(i,j-1)} - 2u_{(i,j)} + u_{(i,j+1)} \tag{15.13}$$

$$v_{xx} \approx v_{(i-1,j)} - 2v_{(i,j)} + v_{(i+1,j)} \tag{15.14}$$

$$v_{yy} \approx v_{(i,j-1)} - 2v_{(i,j)} + v_{(i,j+1)} \tag{15.15}$$

Where (i, j) denotes a pixel's position. The iterative update equations can be shown to be,

$$u = (\alpha + \beta + I_y^2)\gamma_1 - I_x I_y \gamma_2 \tag{15.16}$$

$$v = -I_x I_y \gamma_1 + (\alpha + \beta + I_x^2)\gamma_2 \tag{15.17}$$

where,

$$\gamma_1 = \alpha(u_{(i-1,j)} + u_{(i+1,j)}) + \beta(u_{(i,j-1)} + u_{(i,j+1)}) + (\alpha - \beta) v_{xy} - I_x I_t \tag{15.18}$$

$$\gamma_2 = \alpha(v_{(i,j-1)} + v_{(i,j+1)}) + \beta(v_{(i-1,j)} + v_{(i+1,j)}) + (\alpha - \beta) u_{xy} - I_y I_t \tag{15.19}$$

15.2.1 Experimental results

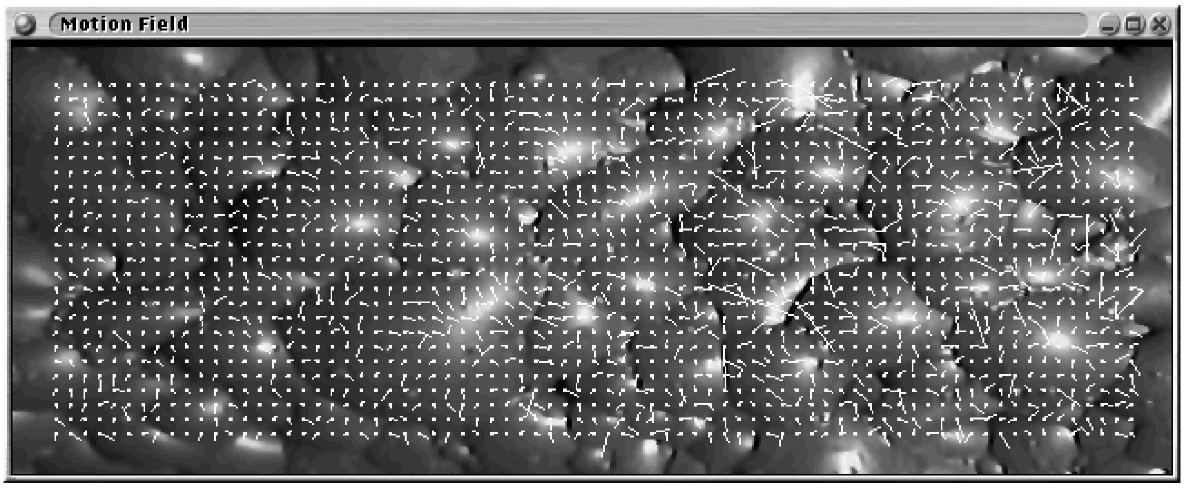
Div-curl optical flow was applied to the standard set of test images (see figure 13.1). The parameters chosen were, number of iterations 16, alpha 0.8, and beta 0.2, in an attempt to provide good performance. The result of the application is shown in figure 15.2. The average motion is measured as $(0.44, -0.29)$. The energy in the error image is $8\,160\,649$.

15.2.2 Discussion

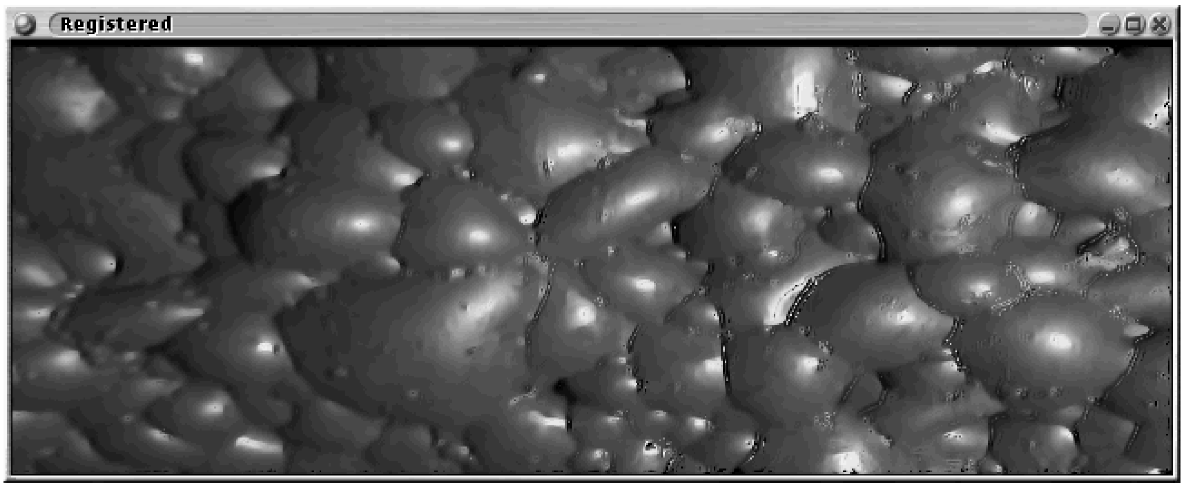
Essentially this is a different form of regularising the optical flow constraint equation. The div-curl regularisation incorporates Horn and Schunk's method as a special case ($\alpha = \beta$ in equation (15.9)).

15.3 Multiple constraint optical flow

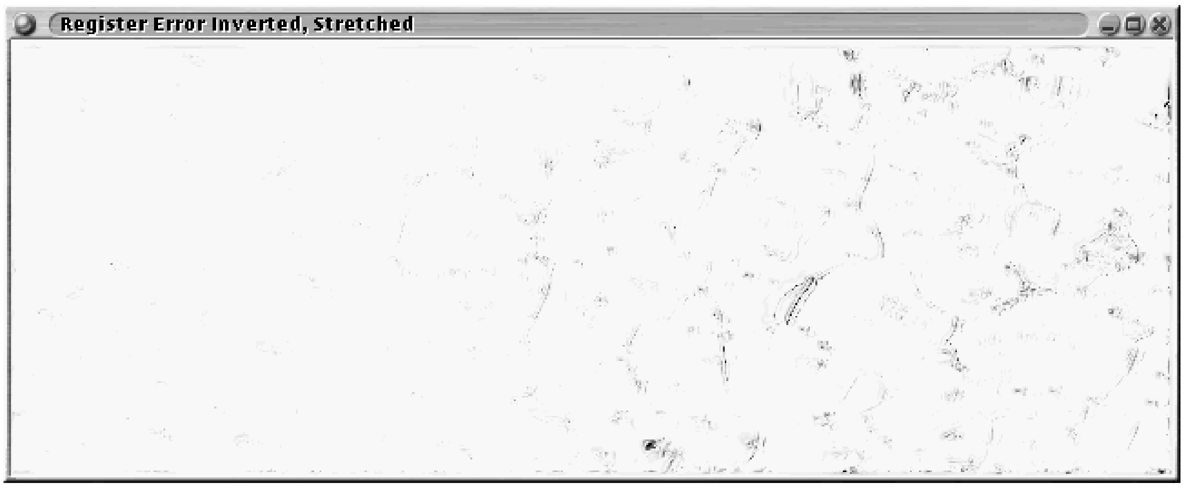
This was proposed by Tistarelli [138] to overcome the need to regularise the optical flow constraint equation. In essence a series of images is generated from each image and the optical flow constraint equation is applied. An overdetermined set of linear equations results which can then be solved as



(a) Motion field, scaled by factor of 4. Average motion $(0.44,\ -0.29)$.



(b) Registered image.



(c) Registered error, stretched. Energy in error image is 8 160 649.

FIGURE 15.2 Div-Curl optical flow.

required. A flaw is that it is possible that at certain pathological points, or even regions, in the image enough constraint equations may disappear or become ill-posed, to make the set of linear equations unsolvable. It should be noted that any set of features may be used.

Tistarelli proposed the following scheme,

$$I_{xx}u + I_{xy}v + I_{xt} = 0 \tag{15.20}$$

$$I_{yx}u + I_{yy}v + I_{yt} = 0 \tag{15.21}$$

Combining the above equations ((15.20) and (15.21)) with the optical flow constraint equation (15.1) provides three equations in two unknowns. Tistarelli explores the following methods for computing the optical flow,

(i) Finding the pair of equations with highest determinant and solving for (u, v) from these.

(ii) Computing the pseudo-inverse of the three equations, and solving.

(iii) Averaging the optical flow due to the *pair* of equation pairs with highest determinant.

Tistarelli proposes the third. While Tistareli proposes extensions, for example, accounting for linear variations in the motion field, these extensions have been neglected in this implementation. The added accuracy is not judged worth the extra complexity.

This technique may also be applied in feature space to generate extra constraint equations. Gupta and Prince et al. proposed a bandpass scheme to provide the extra constraint equations [55]. An oriented bandpass filter is applied in four directions to yield four images. The filters are orientated as follows: north to south, east to west, north-east to south-west, and north-west to south-east. The constraint equations (15.20), (15.21) and the optical flow constraint equation (15.1) are applied to the four images to yield an overdetermined system of twelve equations. Gupta et al. solve these using least-squares.

Tistarelli suggests weighting the solutions by the determinant. This results in [138],

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum_{i=l}^{m+l} N_i^u \\ \sum_{i=l}^{m+l} N_i^v \end{pmatrix} \frac{1}{\sum_{i=l}^{m+l} D_i} \tag{15.22}$$

where, N^u and N^v represent the x and y co-ordinates of the solution of a pair of l constraint equations. D_i represents the determinant of the minor matrix in question. Here, only m of the l equations are selected, for example, the m pairs of equations with largest determinants. N^u and N^v may be computed by direct inverse of the minor matric concerned (where this is possible). The final step of the algorithm is to smooth the optical flow field in the x and y components using a Gaussian kernel. A sigma of $\sigma = 1$ is used.

15.3.1 Experimental results

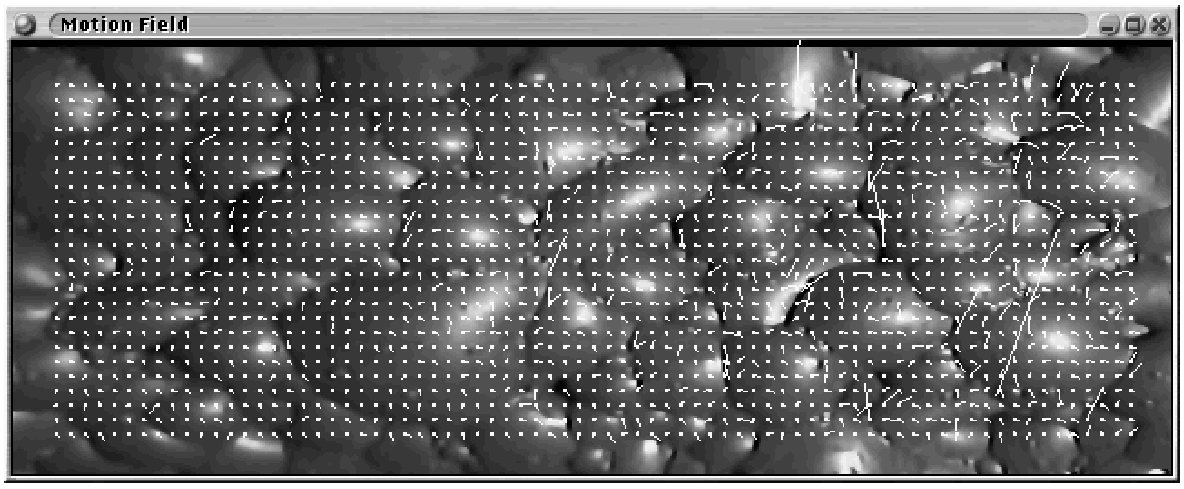
The algorithm was applied to the standard set of test images (see figure 13.1). The threshold testing the determinants was chosen to be $\tau = 1$ and the standard deviation for smoothing set to $\sigma = 2$. A smaller τ leads to more singular equation pairs and hence less reliable motion vectors at pixels with small first and second derivatives. A larger τ tends to reject motion vectors where the estimate is still reliable. The choice of σ lies in a tradeoff between over-smoothing the motion field, and not regularising the motion estimates enough. The result of the test is shown in figure 15.1. The average motion vector is measured as $(0.45, \ -0.25)$. the energy in the error image is $10\,795\,532$.

15.3.2 Discussion

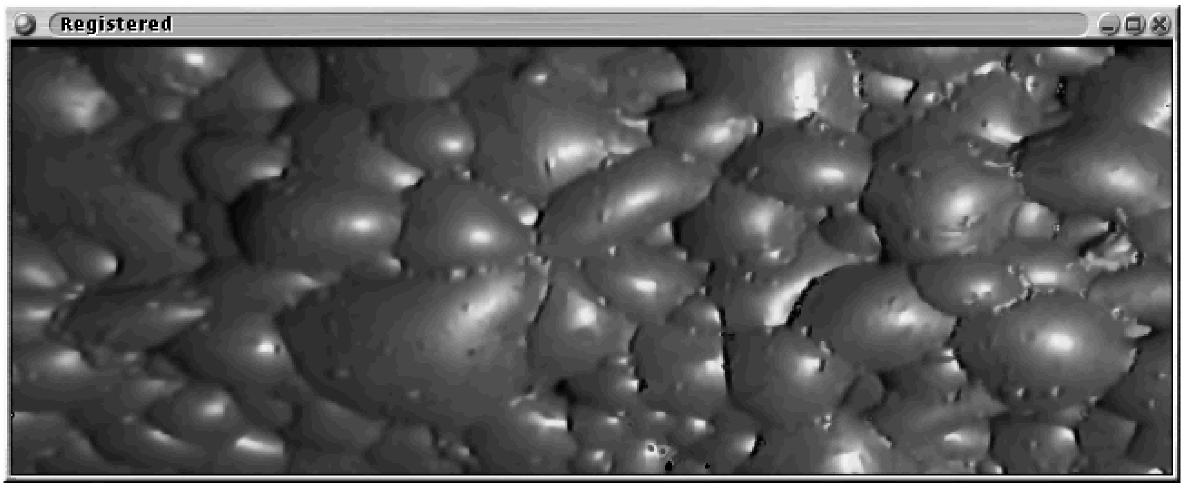
The method has certain difficulties. There are pixels where the sets of equations may be all ill-conditioned. While these pixels may be detected and discarded, the question of how to compute a motion estimate for these pixels may be posed. For individual pixels, or small groups of pixels, the final smoothing stage propagates the neighbouring motion estimates into the discarded pixels.

The second problem lies in the weighting of the solutions. It is possible for a matrix determinant to be negative. This could result, for example, from using different features (perhaps, variance and second derivatives) to generate the constraint equations. It is proposed that a weighting using the absolute value of the determinant be used instead. The relevant equation (equation 15.22) becomes,

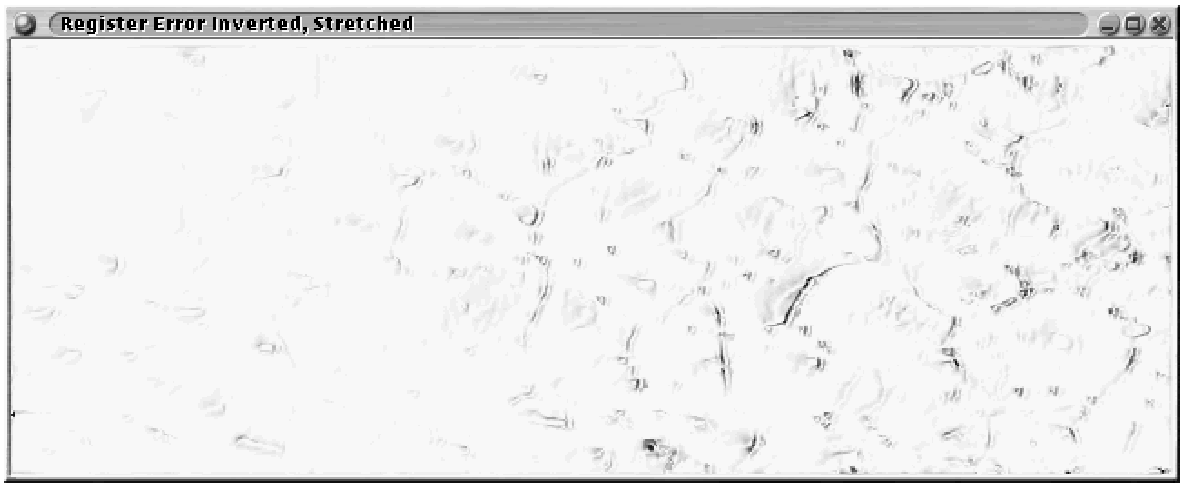
$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum_{i=l}^{m+l} |D_i| N_i^u \\ \sum_{i=l}^{m+l} |D_i| N_i^v \end{pmatrix} \frac{1}{\sum_{i=l}^{m+l} |D_i|} \tag{15.23}$$



(a) Motion field, scaled by factor of 4. Average motion $(0.45, -0.25)$.



(b) Registered image.



(c) Registered error, stretched. Energy in error image is 10 795 532.

FIGURE 15.3 Optical flow via Multiple Constraint Optical Flow.

Chapter 16

Hierarchical motion estimation

Several hierarchical motion estimation algorithms are described in the literature [10]. This chapter describes a general hierarchical framework in which an arbitrary motion estimation algorithm may be embedded. The algorithm is then used to compute motion at several scales.

Scale space representation of images represent an image in terms of detail. At the highest detail, or smallest scale, all structures in an image are visible (high resolution). At the lowest detail, or highest scale, only the large (in spatial extent) features of the image are visible (low resolution). This is the basis of multi-resolution analysis as used in wavelet analysis of images [3, 51]. One can generate a scale space pyramid for an image, by low-pass filtering, and then decimating the image. The low-pass filtering removes high frequency components, which lowers the Nyquist frequency of the image. This implies the filtered image is over-sampled which allows for decimation to remove the redundant information (pixels) [3, 130]. At the apex of the scale space pyramid is the low scale (low spatial resolution) image which is a fraction of the size of the original image.

Generating scale space pyramids can be very efficient. The low-pass filter process may be implemented in the spatial domain as a correlation, but since 75% of the pixels are discarded by the decimation process, only the correlations at the retained pixels need to be computed. The limiting factor for computational complexity is then the length of the low-pass filter kernel. A natural optimisation for images is the selection of a separable low-pass filter kernel [139]. Separability implies a two dimensional convolution may be achieved by first row-wise then column-wise convolution (or vice versa). One would perform the row-wise convolution on each row, and then the column-wise convolution only at the retained pixels, for example.

The motion vectors are interpolated for the next highest scale, and used to translate the search area (for Block Motion Estimation (BME)), or to warp the motion frame (frame at time $t + \Delta t$) to form a new motion frame. The advantages of this system include: smaller search areas may be used at each stage, and correlation between neighbouring motion vectors is enforced by the interpolation of motion vectors. The low-pass filtering stages improve the noise immunity at higher scales.

One previous approach (see [10]) required registering the motion frame using the interpolated motion field, and then computing the motion at the new scale using the registered image as the reference frame.

In all cases the fields used are first de-interlaced to remove motion artifacts produced by the interlacing process, then denoised by 4 iterations of Perona and Malik's Anisotropic Diffusion algorithm [113] (see section 7.1), here the time step (γ) used is 0.2 and the K values determined using Canny's noise estimator as described by Perona and Malik. The choice of 4 iterations is determined by the tradeoff between removing noise (and smoothing away bubbles too small to be of significance) and merging bubbles best left separate.

The chapter proceeds by first describing the frameworks. The application of the frameworks to various motion estimation algorithms is described. Finally the results are examined.

16.1 The proposed frameworks

Two frameworks will be considered. One takes advantage of the structure of block (or region) matching algorithms. The other is a more general framework, in that no assumptions of the motion estimation algorithm used are made, with the exception that the motion estimation algorithm used, produces a dense motion field (in other words, a motion estimate at every pixel).

16.1.1 Framework 1

A possible modification to block based motion estimation algorithms is simple: whenever a search block is accessed, an offset is added to the position of the search block. This offset corresponds to the interpolated motion field from the previous stage. In essence the origin of the search space used when computing the motion vector at a particular pixel, is translated by the interpolated motion field as generated by the previous stage of the algorithm. The same process is used for segmentation and cluster based motion estimation algorithms.

The framework (Framework 1) is as follows (similar to the algorithm described in [139][pg. 438–441]:

(i) Generate scale-space pyramids for a preselected number of scales for the motion and reference images by low-pass filtering and then decimation of the rows and columns of the images by a factor of 2:

    (a) Low-pass filter via a Gaussian filter with $\sigma = 1$ and $radius = 4\sigma$. Gaussian filters possess a maximum principle which prevents maxima (or minima) from being formed which were not present in the unfiltered image [7, 113]. The width of the Gaussian is chosen to eliminate aliasing after the decimation process.

    In the calculations following, a normalised frequency is used. The normalisation is achieved by dividing the angular frequencies through by the sampling frequency (in Hertz). The angular sampling frequency then occurs at 2π.

It can be shown that the -3dB frequency of a Gaussian is at $\omega = \frac{1}{\sigma}\sqrt{\frac{3}{20} ln(10)}$, effectively for sigma of one, $\omega = 0.588$, which is well before the desired new Nyquist frequency of $\omega = \pi$. At the new Nyquist frequency the gain of the filter (with a sigma of one) is -85dB. This justifies the sigma value used.

The Gaussian kernel is truncated to speed the low-pass filtering, the Gaussian is otherwise of infinite extent. The size of the footprint chosen allows most of the energy within the Gaussian kernel to lie within the footprint selected. The choice of neighbourhood size is otherwise arbitrary.

(b) Decimate each row and column by 2 to generate an image at the new scale.

(c) Repeat at the next scale using the old scale image as input.

(ii) For each scale starting at the lowest:

(a) Interpolate the motion field by a factor of two. The interpolation proceeds as follows,

(1) Super-sample by factor of two. This introduces zero value pixels between valid pixels.

(2) Convolve each component of the motion vector field with a Gaussian of $\sigma = 1$ and $radius = 4\sigma$ to interpolate between the samples.

(b) Apply the block motion estimation algorithm on the motion and reference image at this new scale to generate a new motion field. Note that the search window is translated by the interpolated motion estimate from the previous stage. The new motion estimate at a pixel lies within the search region. This new motion estimate is added to the old motion estimate from the previous stage, and passed on to subsequent stages.

16.1.2 Framework 2

It is not clear, however, how Framework 1 may be extended to gradient based optical flow algorithms. At a particular scale, an estimate of the motion field may be generated. After the interpolation to the next higher scale, this motion field may be used to warp the frame at time $t + \Delta t$ (the motion frame). The warped image becomes the reference frame for the next application of the motion estimation algorithm. The thresholded difference between the warped and the reference frame (the frame at time t) yields points, or regions, which need better motion estimates. The motion estimation stage may then focus exclusively on generating motion estimates for these pixels. This promises a speed improvement.

In temporal compression of image sequences a similar technique of examining differences is used for comparing different block motion estimation algorithms [139][pg 440–442]. Hierarchical frame rate conversion uses a scheme of using a motion field to interpolate between frames [10].

The alternate framework (Framework 2) is then,

(i) Select a motion estimation algorithm.

(ii) Generate a scale pyramid for a preselected number of scales (scale zero is the original image) for the motion and reference images,

   (a) Low pass filter via a Gaussian filter with $\sigma = 1$ and $radius = 4\sigma$.

   (b) Decimate each row and column by 2 to generate image at new scale.

   (c) Repeat at next scale using old scale image as input.

(iii) For each scale starting at the highest,

   (a) Interpolate by a factor of two for the motion field. The interpolation proceeds as follows,

      (1) Super-sample by factor of two. This introduces zero value pixels between valid pixels.

      (2) Convolve each component of the motion field with a Gaussian of $\sigma = 1$ and $radius = 4\sigma$ to interpolate between the samples.

   (b) Generate a new reference frame from the interpolated motion field and the current motion frame. The intensity at each pixel in the new reference frame may be found by averaging a small neighbourhood of pixels located at the terminus of the motion vector at the pixel in question (interpolation). A Gaussian kernel of $\sigma = 0.5$ is suggested for the averaging, with a neighbourhood of size $4\sigma \times 4\sigma$.

   (c) Apply the motion estimation algorithm to the new reference and old motion frames to generate a new motion field.

   (d) Add the new and old motion fields. This is the result motion field passed to the next stage.

16.2 Application of the frameworks

The two algorithm frameworks described are applied to some motion estimation algorithms and the results examined. The full search algorithm, Minimum Absolute Difference (MAD) based Successive Elimination Algorithm (SEA), Mean Square Error (MSE) based Successive Elimination Algorithm (SEA), motion from watershed and Horn and Schunk's algorithm are examined.

16.2.1 Application to a full search

A simple full search is applied at each level of the pyramid. The full search algorithm finds the MSE at every pixel in a small search window (sometimes the entire image) [9, 87, 159]. The motion vector is then the difference in position between the best-match position and the origin of the search space. The block size and search area is 3×3 pixels. The small search blocks and search space implies that the full range of motion within the froth images will not be explored. A 3×3 search space cannot estimate motion vectors larger than one pixel, this implies that any pixels with larger motion will have inaccurate motion estimates. A single stage full search algorithm with block sizes and search areas as specified, cannot then be expected to do well. Any benefits, therefore, is due to the hierarchical nature of the algorithm frameworks examined.

The smoothing and interpolation kernel used is a Gaussian with $\sigma = 1$. The Gaussian is truncated to lie in a 4×4 pixel area. At the coarsest scale (lowest spatial resolution or detail) a 3×3 block size and search area on the sub-sampled image, results in an effective 24×24 block size and search area on the full sized image.

Framework 1

Figure 16.1 shows the application of the hierarchical method (Framework 1) to the standard test images (see figure 13.1). The average motion vector is $(0.76, -0.51)$. The energy in the error image is $2\,821\,950$.

Framework 2

Applying the full-search algorithm as part of Framework 2 is shown in figure 16.2. The average motion vector is $(0.92 \ -0.52)$. The energy in the error image is $1\,717\,858$. The average motion is close to that found for Framework 1, and the energy in the error image is smaller.

Comparing the frameworks

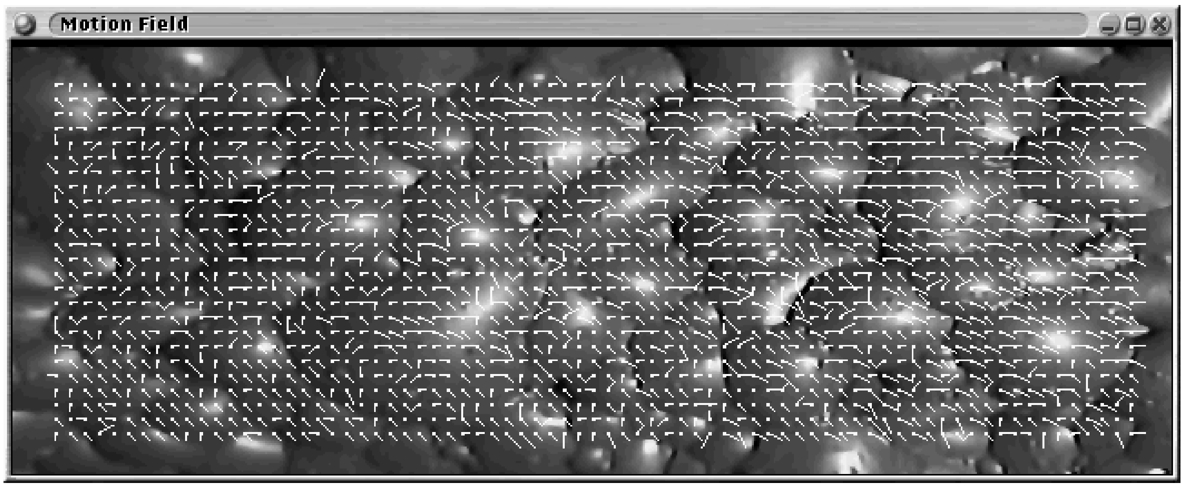
The two motion fields as generated by Frameworks 1 and 2 can be compared. The allowed difference between corresponding motion vectors from the different frameworks is arbitrary, but selected to pass motion vectors with approximately the same orientation and length. A variation of up to 50% of the length and orientation of the motion vectors of Framework 2 will be allowed. In other words the corresponding motion vector of Framework 1 may only vary by up to 50% in length and orientation, in order to be counted as correct.

This finds that only 53.0% of the total number of motion vectors generated by Framework 1, lies within the specified variation of the motion vectors as generated by Framework 2. The percentage of motion vectors whose length lies within the variation is 74.6%, and the percentage of motion vectors whose orientation lies within the variation is 62.3%. The motion fields as generated by these two frameworks are thus very different.

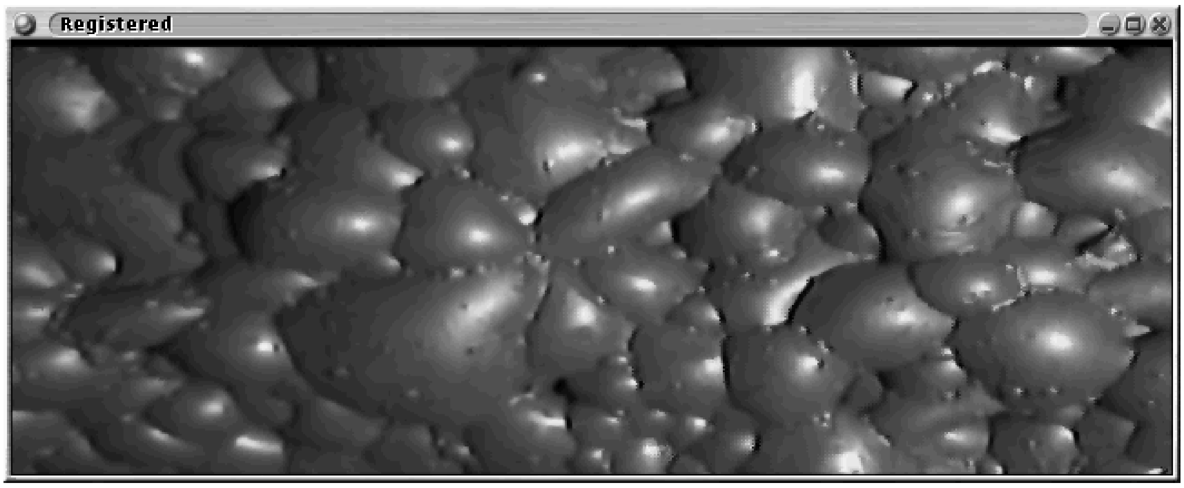
Visual inspection of the two motion fields in figures 16.1 and 16.2 show them to be similar looking. Framework 2 produces a significantly lower error energy than Framework 1.

16.2.2 Application to Minimum Absolute Difference (MAD) Successive Elimination Algorithm (SEA)

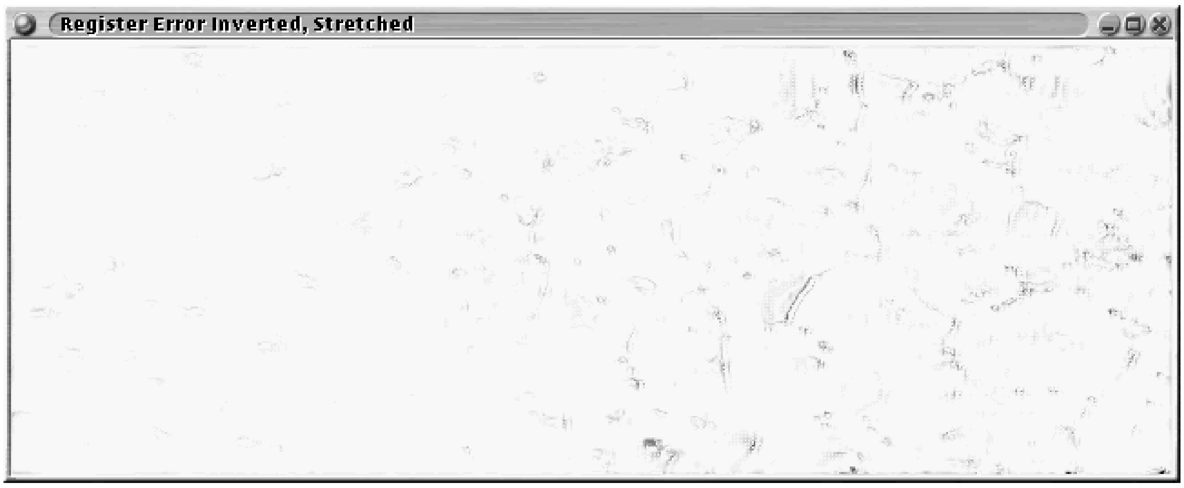
This is Li and Salari's algorithm [87] (see subsection 14.4.1). The only modification made was to limit the size of the search area to 15×15 pixels. The size of the search block used is 7×7. The height of the pyramid used is 4. The application of the block size and search area specified on the subsampled



(a) Motion field, scaled by factor of 4. Average motion $(0.76,\ -0.51)$.

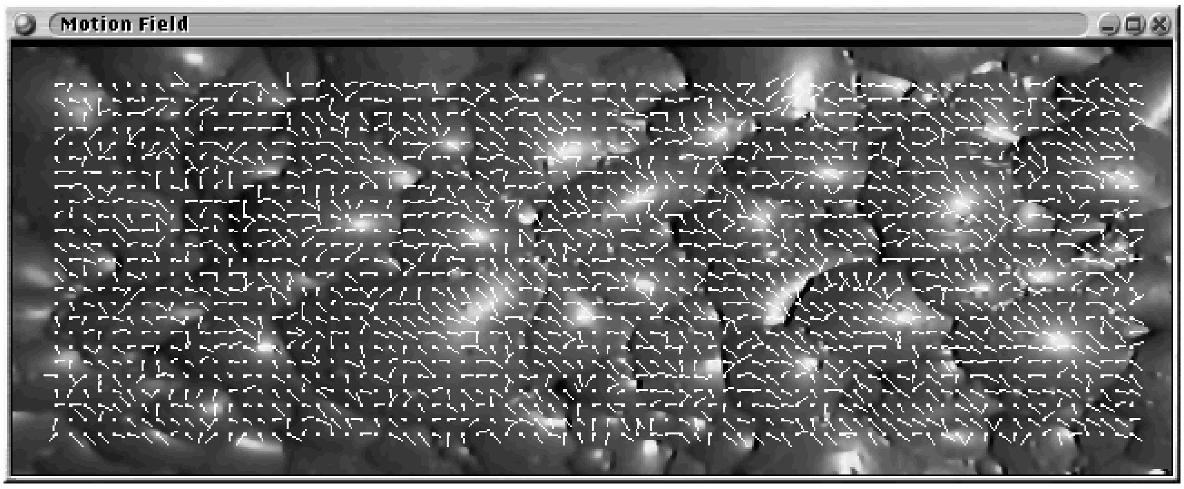


(b) Registered image.

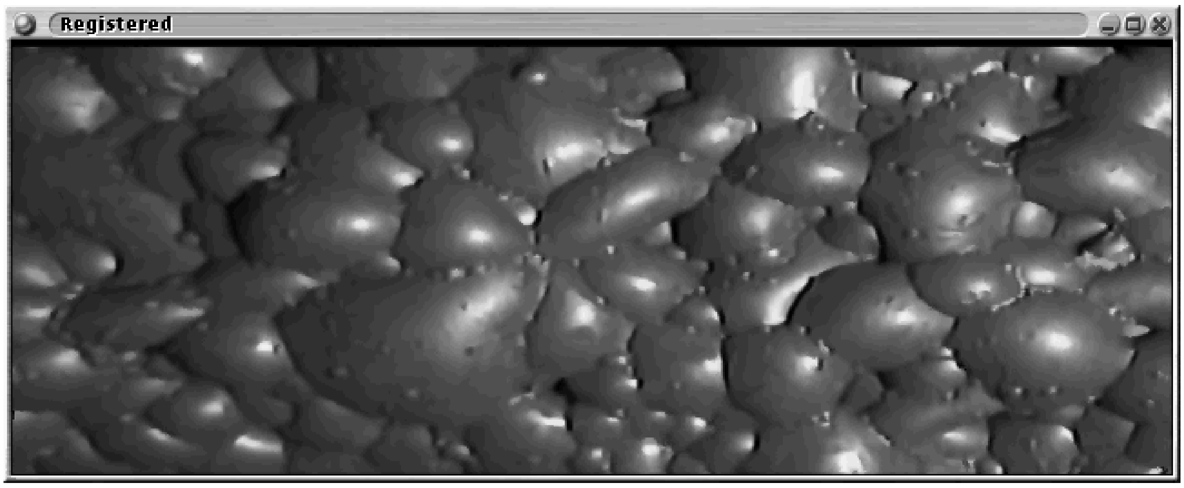


(c) Registered error, stretched. Energy in error image is 2 821 950.

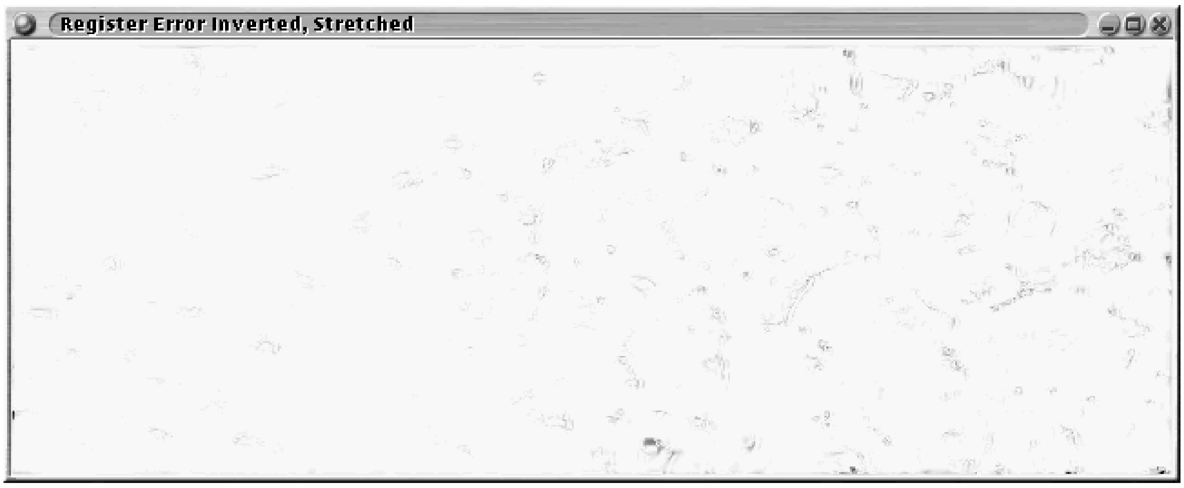
FIGURE 16.1 Hierarchical motion estimation via full search, Framework 1.



(a) Motion field, scaled by factor of 4. Average motion $(0.92\ -0.52)$.



(b) Registered image.



(c) Registered error, stretched. Energy in error image is 1 717 858.

FIGURE 16.2 Hierarchical motion estimation via full search, Framework 2.

image of lowest resolution, results in a maximum search area of 120×120, and a maximum search block of 56×56 pixels, at the coarsest scale (lowest detail) on the full sized image.

Framework 1

Figure 16.3 shows the application of Framework 1 to the standard test images (figure 13.1). The average motion vector is $(1.3, -0.70)$. The energy in the error image is $1\,774\,367$. Note the correlated motion field and the low error energy.

Framework 2

Figure 16.4 shows the application of Framework 2 to the standard test images (figure 13.1). The average motion vector is $(1.4, -0.67)$, and the energy in the error image is $2\,081\,428$.

Comparing the frameworks

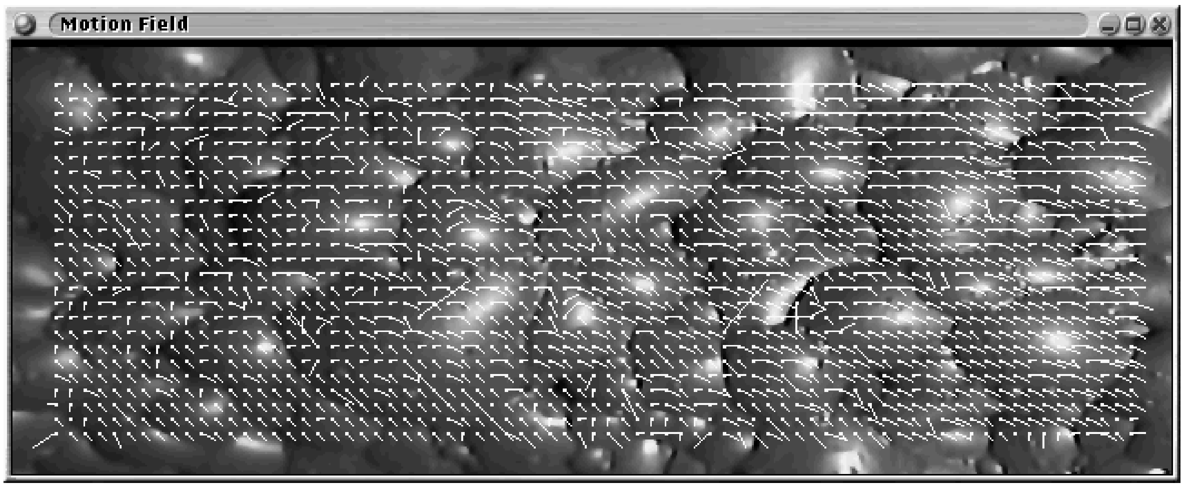
The two motion fields as generated by Frameworks 1 and 2 can be compared. The allowed difference between corresponding motion vectors from the different frameworks is arbitrary, but selected to pass motion vectors with approximately the same orientation and length. A variation of up to 50% of the length and orientation of the motion vectors of Framework 2 will be allowed. In other words the corresponding motion vector of Framework 1 may only vary by up to 50% in length and orientation, in order to be counted as correct.

This finds that only 59.7% of the total number of motion vectors generated by Framework 1, lies within the specified variation of the motion vectors as generated by Framework 2. The percentage of motion vectors whose length lies within the variation is 80.7%, and the percentage of motion vectors whose orientation lies within the variation is 68.0%. The motion fields as generated by these two frameworks are thus very different.

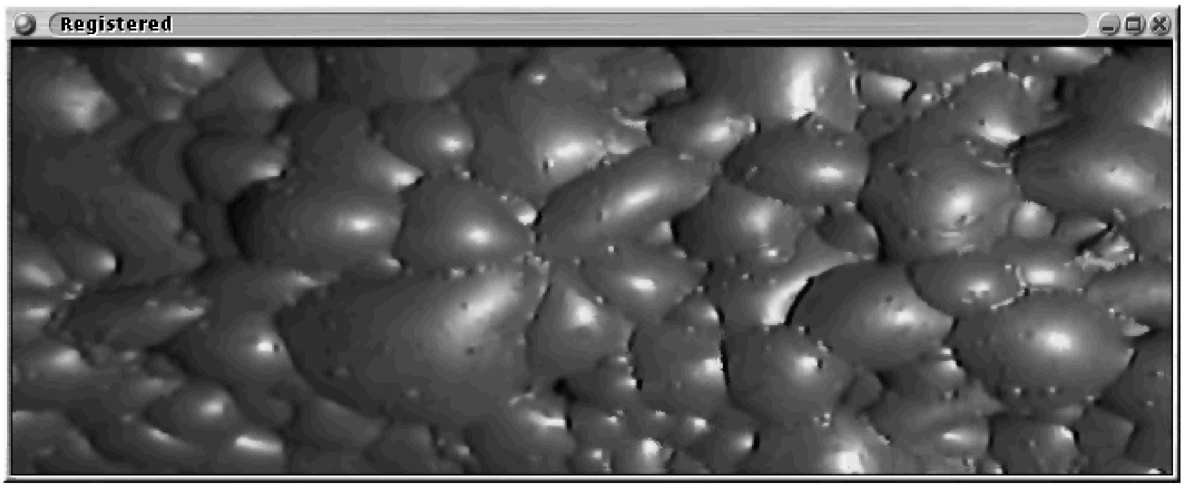
Visual inspection of the two motion fields in figures 16.3 and 16.4 show them to be similar looking. The average motion vectors are close, but the error energies differ significantly (by more than 10%), for the two frameworks.

16.2.3 Application to Mean Square Error (MSE) based Successive Elimination Algorithm (SEA)

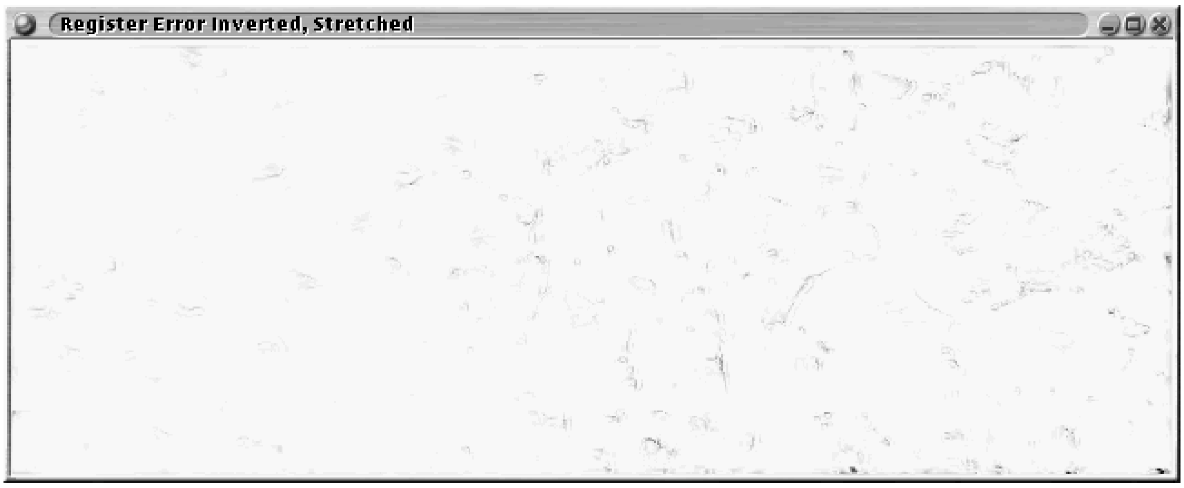
The size of the search block is 7×7 pixels. The smoothing and interpolation kernel is Gaussian with $\sigma = 1$. The Gaussian is truncated to lie within a 4×4 window. The size of the search area is limited to 15×15 pixels. The height of the pyramid used is 4. On the subsampled image of lowest spatial resolution, this then allows for a maximum search area of 120×120, and a maximum search block of 56×56 pixels, if one were to interpolate this low resolution image to full image size (the size of the high detail image).



(a) Motion field, scaled by factor of 4. Average motion $(1.3,\ -0.70)$.

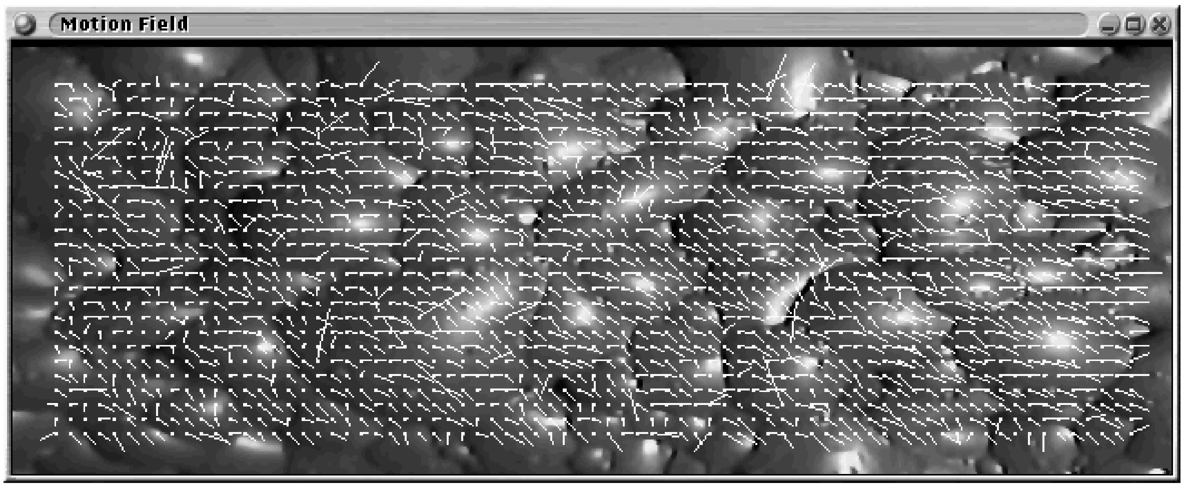


(b) Registered image.

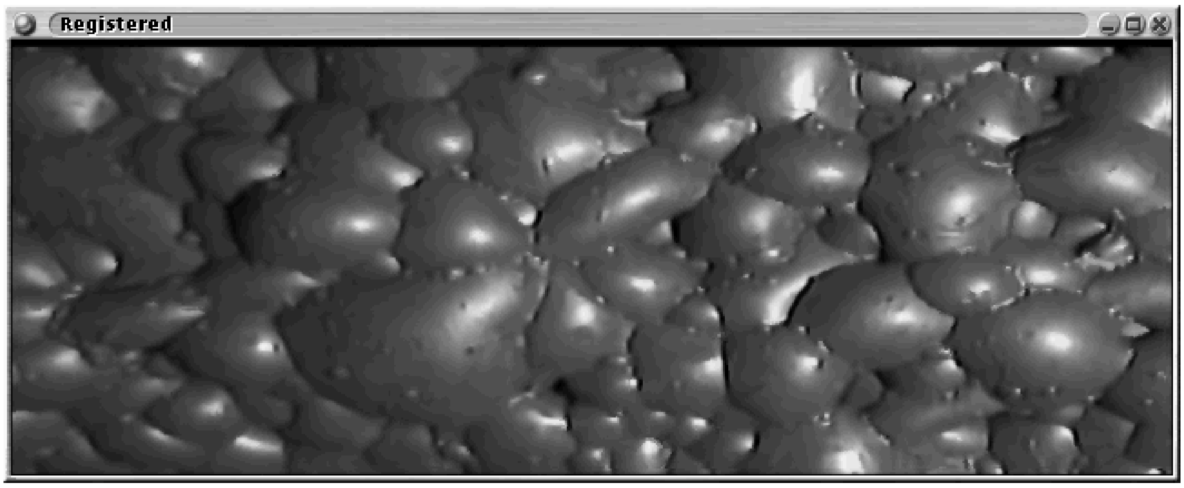


(c) Registered error, stretched. Energy in error image is 1 774 367.

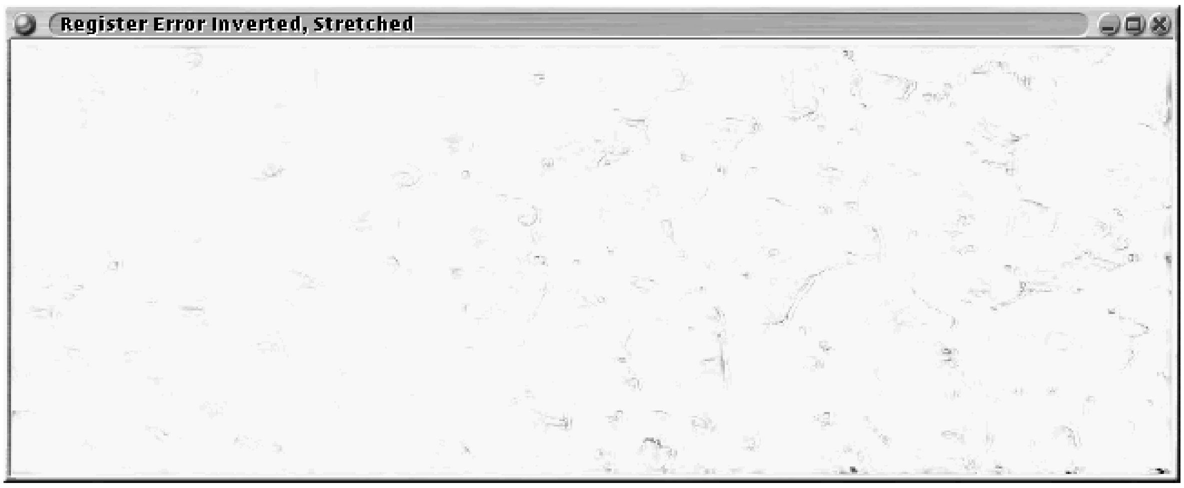
FIGURE 16.3 Hierarchical motion estimation, test algorithm MAD based SEA, Framework 1.



(a) Motion field, scaled by factor of 4. Average motion $(1.4,\ -0.67)$.



(b) Registered image.



(c) Registered error, stretched. Energy in error image is 2 081 428.

FIGURE 16.4 Hierarchical motion estimation, test algorithm MAD based SEA, Algorithm 2.

Framework 1

Figure 16.5 shows the application to the standard test images (figure 13.1). The average motion is $(1.3, -0.70)$. The energy in the error image is $1\,832\,788$. Note the small error and the correlated motion field. Comparison with section 16.2.1 shows the improvement in performance as the size of the search area and the search block increases.

Framework 2

Figure 16.6 shows the application of Framework 2 to the standard test images (figure 13.1). The average motion vector is $(1.3, -0.68)$, and the energy in the error image is $1\,924\,062$.

Comparing the frameworks

The two motion fields as generated by Frameworks 1 and 2 can be compared. The allowed difference between corresponding motion vectors from the different frameworks is arbitrary, but selected to pass motion vectors with approximately the same orientation and length. A variation of up to 50% of the length and orientation of the motion vectors of Framework 2 will be allowed. In other words the corresponding motion vector of Framework 1 may only vary by up to 50% in length and orientation, in order to be counted as correct.

This finds that only 60.1% of the total number of motion vectors generated by Framework 1, lies within the specified variation of the motion vectors as generated by Framework 2. The percentage of motion vectors whose length lies within the variation is 81.2%, and the percentage of motion vectors whose orientation lies within the variation is 68.2%. The motion fields as generated by these two frameworks are thus very different.

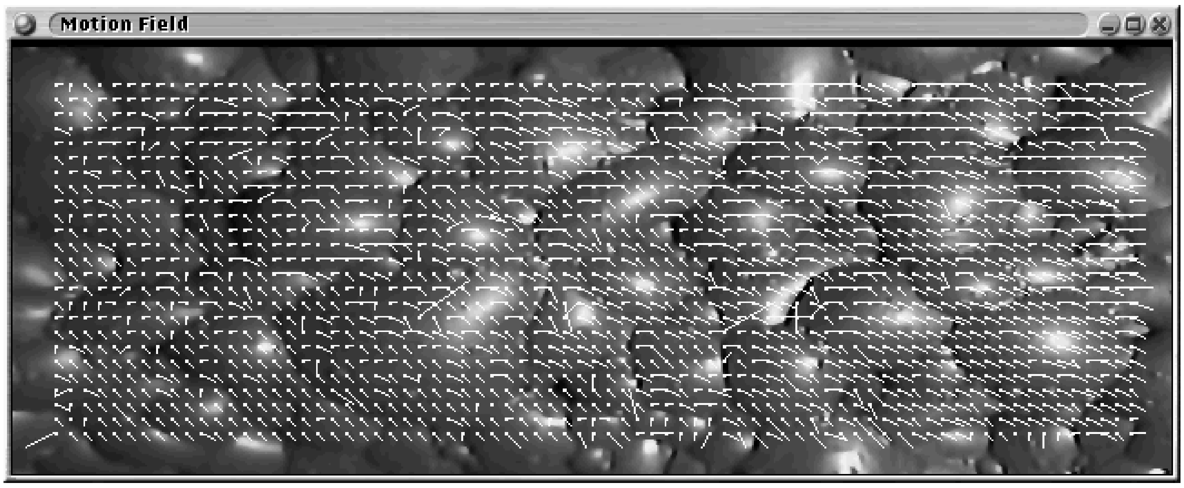
Visual inspection of the two motion fields in figures 16.5 and 16.6 show them to be similar looking. The two frameworks generate similar average motion vectors. The registered error energies differ by less than 5%, implying the performance of the two frameworks is similar for this motion estimation algorithm (for the algorithm parameters specified).

16.2.4 Application to Watershed based motion estimation

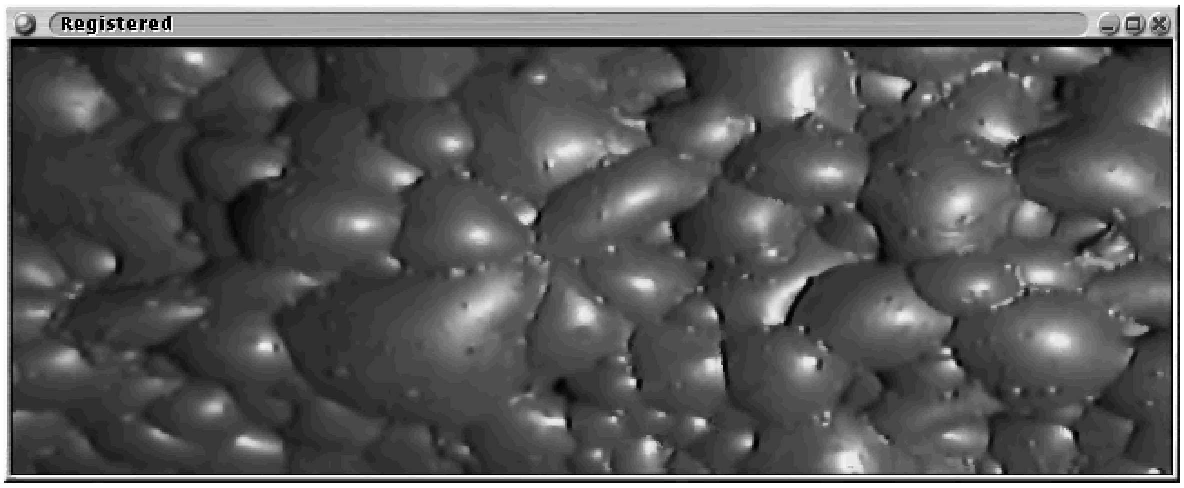
The watershed based motion estimation algorithm is described, in detail, in section 14.7.

Framework 1

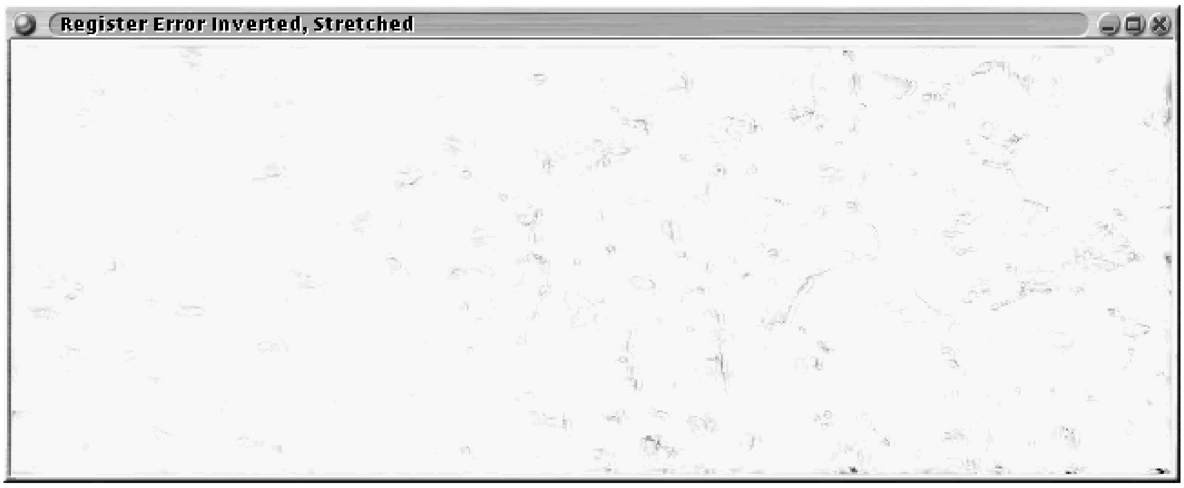
The application of Framework 1 is shown in figure 16.7. The average motion vector is $(1.8, 0.22)$, and the energy in the error image is $20\,903\,965$.



(a) Motion field, scaled by factor of 4. Average motion $(1.3,\ -0.70)$.

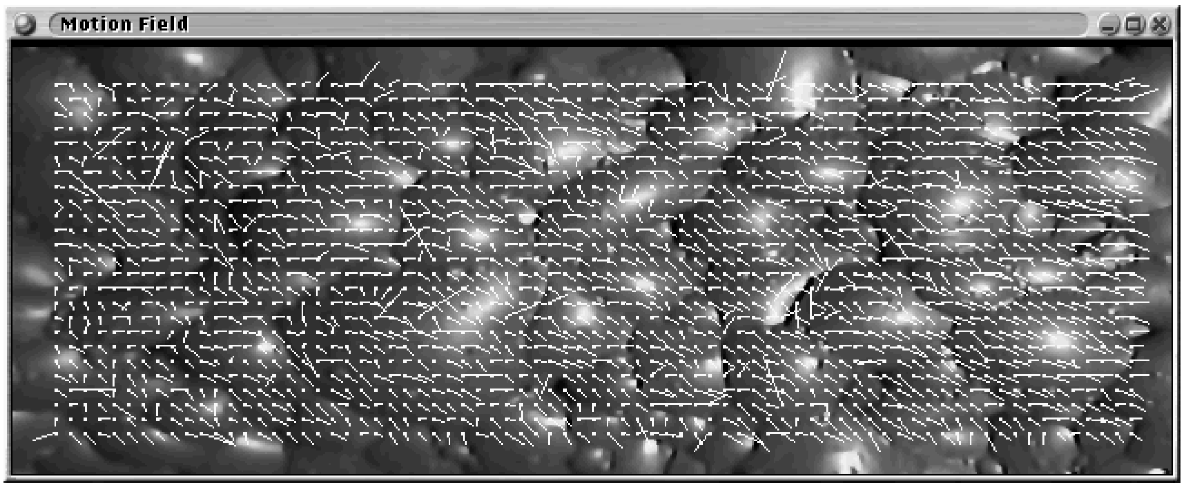


(b) Registered image.

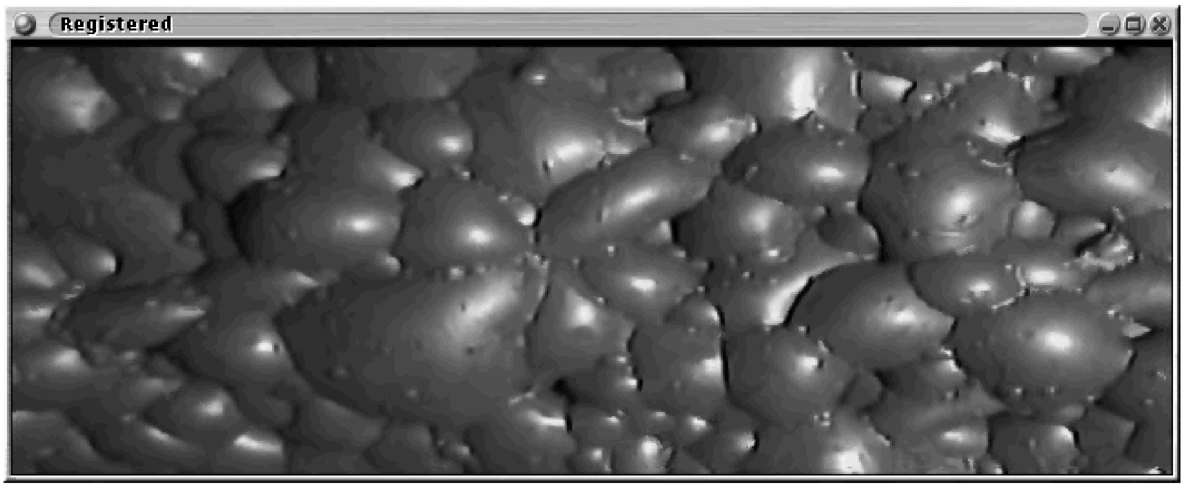


(c) Registered error, stretched. Energy in error image is 1 832 788.

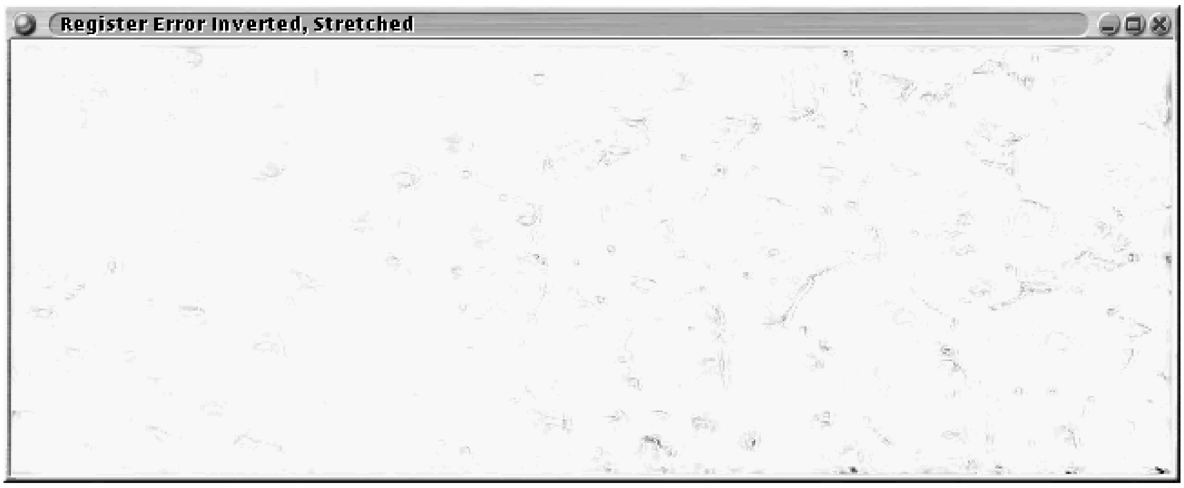
FIGURE 16.5 Hierarchical motion estimation, test algorithm MSE based SEA, Algorithm 1.



(a) Motion field, scaled by factor of 4. Average motion $(1.3, -0.68)$.

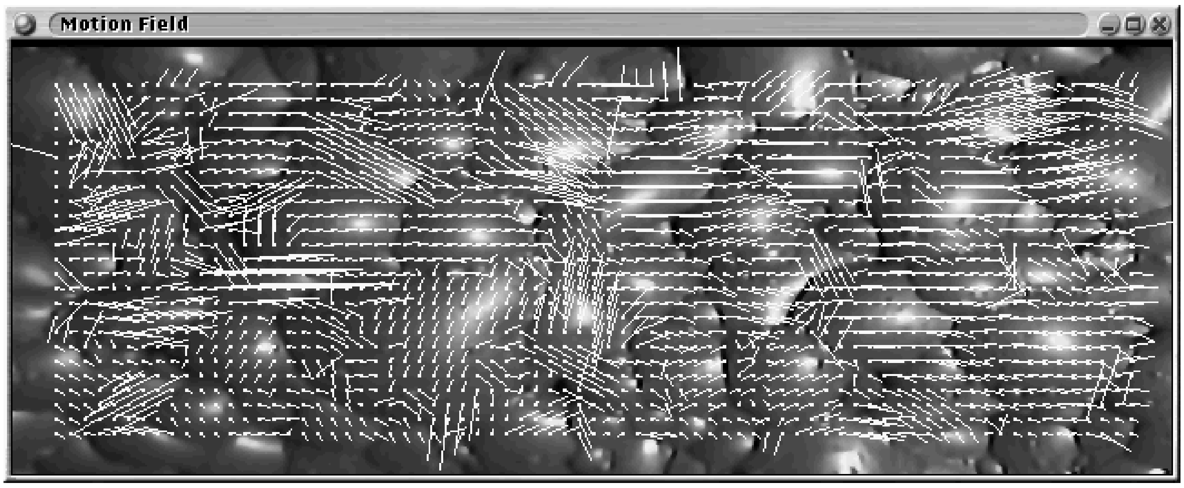


(b) Registered image.

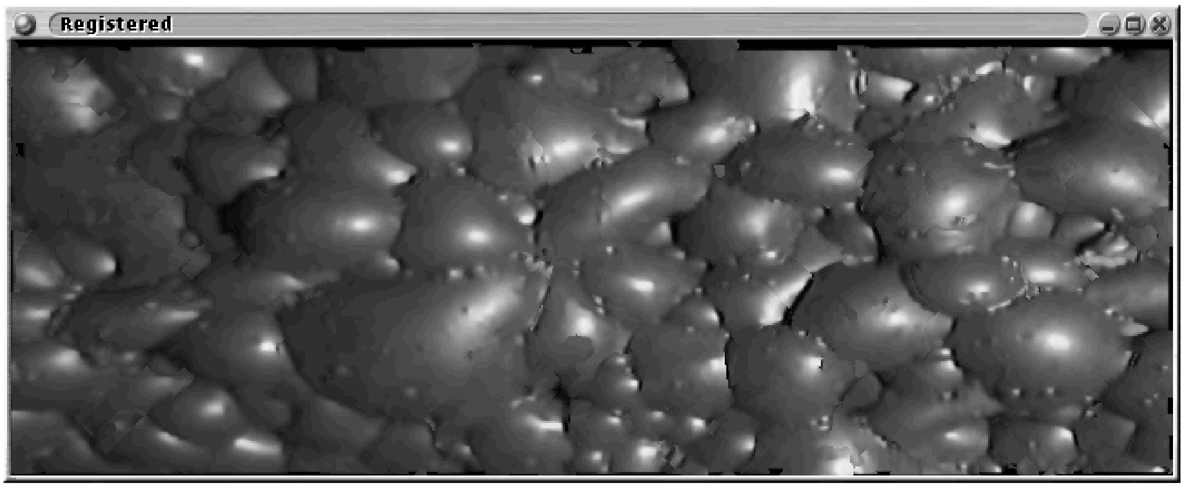


(c) Registered error, stretched. Energy in error image is 1 924 062.

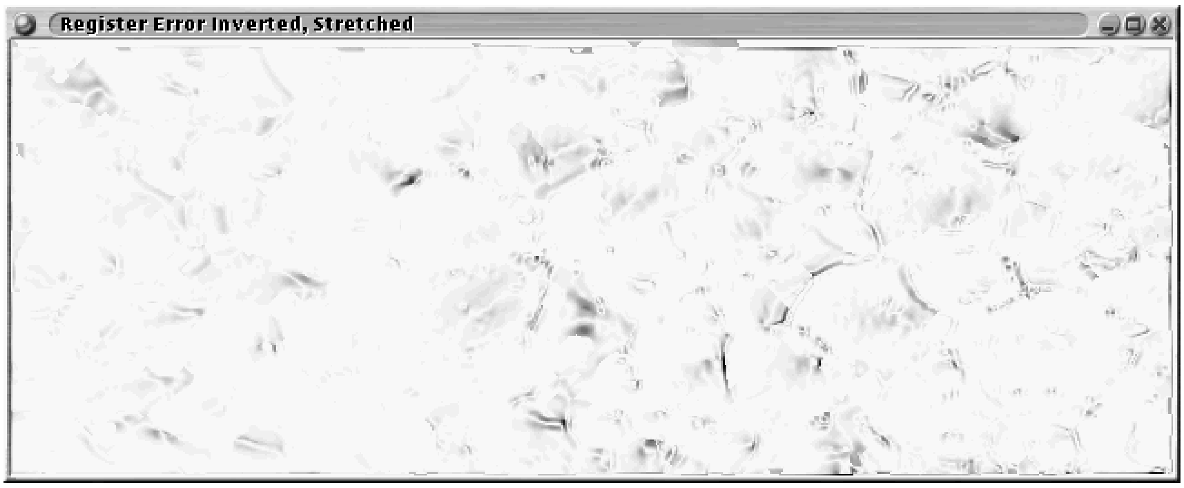
FIGURE 16.6 Hierarchical motion estimation, test algorithm MSE based SEA, Algorithm 2.



(a) Motion field, scaled by factor of 4. Average motion $(1.8,\ 0.22)$.

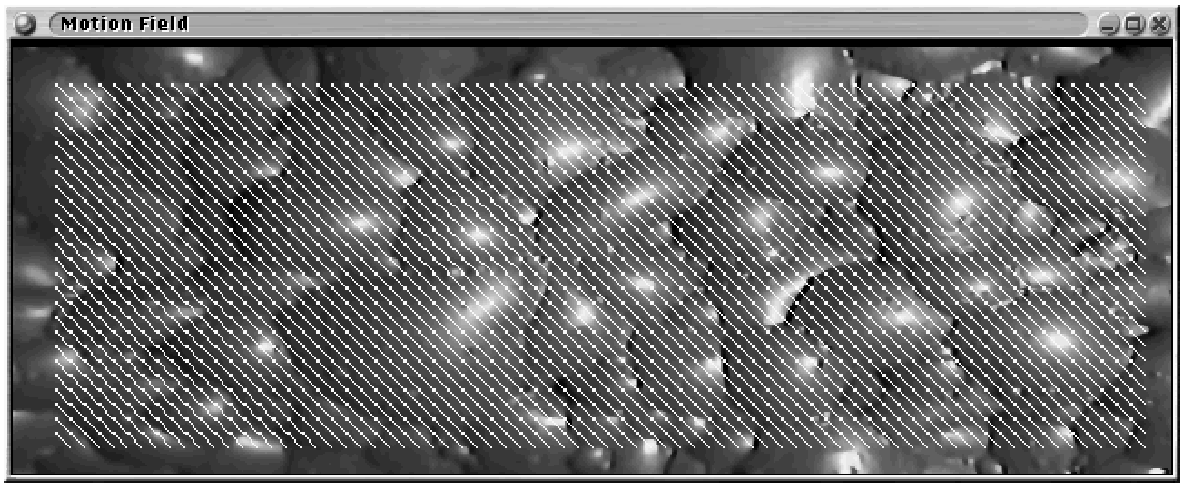


(b) Registered image.

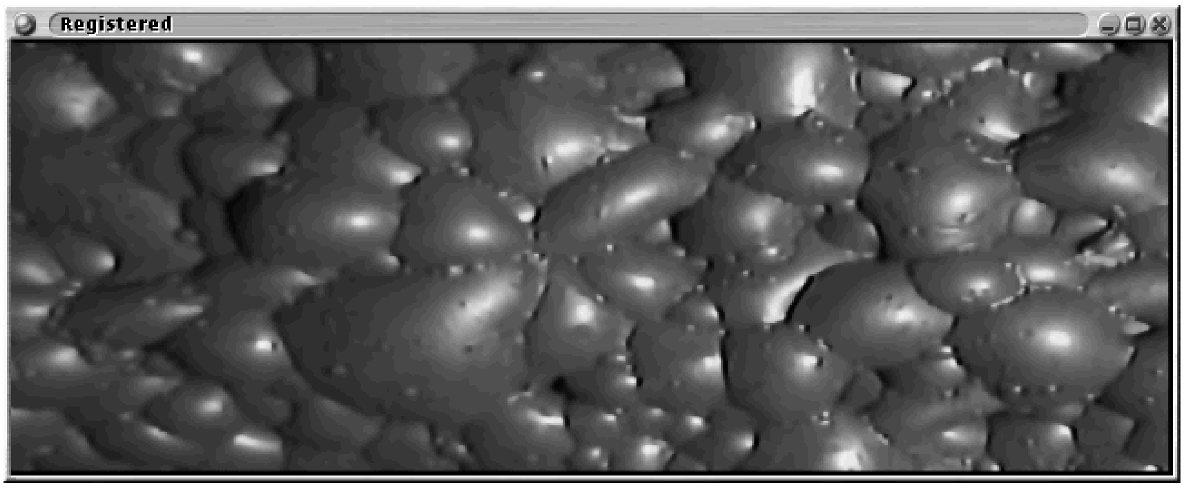


(c) Registered error, stretched. Energy in error image is 20 903 965.

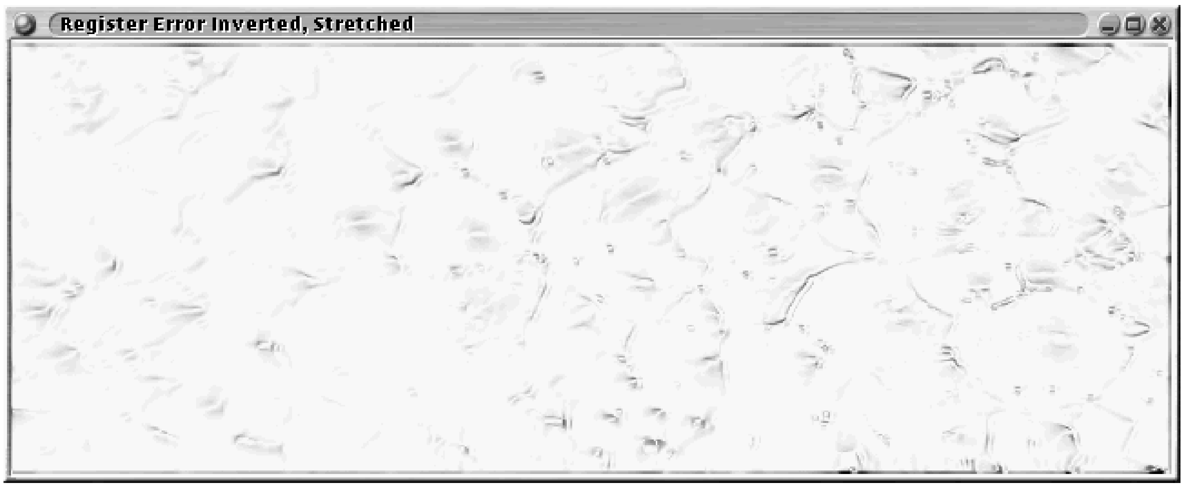
FIGURE 16.7 Hierarchical motion estimation, via Watershed based motion estimation, using Framework 1.



(a) Motion field, scaled by factor of 4. Average motion $(1.98, -1.89)$.



(b) Registered image.



(c) Registered error, stretched. Energy in error image is 12 839 272.

FIGURE 16.8 Hierarchical motion estimation, via Watershed based motion estimation, using Framework 2.

Framework 2

The effect of using Framework 2 is shown in figure 16.8. The average motion vector is $(1.98,\ -1.89)$, and the energy in the error image is $12\,839\,272$. Note the highly correlated motion field and the considerably smaller error than for Framework 1.

Comparing the frameworks

The two motion fields as generated by Frameworks 1 and 2 can be tested. The allowed difference between corresponding motion vectors from the different frameworks is arbitrary, but selected to pass motion vectors with approximately the same orientation and length. A variation of up to 50% of the length and orientation of the motion vectors of Framework 2 will be allowed. In other words the corresponding motion vector of Framework 1 may only vary by up to 50% in length and orientation, in order to be counted as correct.

This finds that only 34.4% of the total number of motion vectors generated by Framework 1, lies within the specified variation of the motion vectors as generated by Framework 2. The percentage of motion vectors whose length lies within the variation is 57.9%, and the percentage of motion vectors whose orientation lies within the variation is 53.9%. The motion fields as generated by these two algorithms are thus very different.

This is also apparent from a visual inspection of the motion fields in figures 16.7 and 16.8. The average motion vectors as well as the register error energies are very different for the two frameworks.

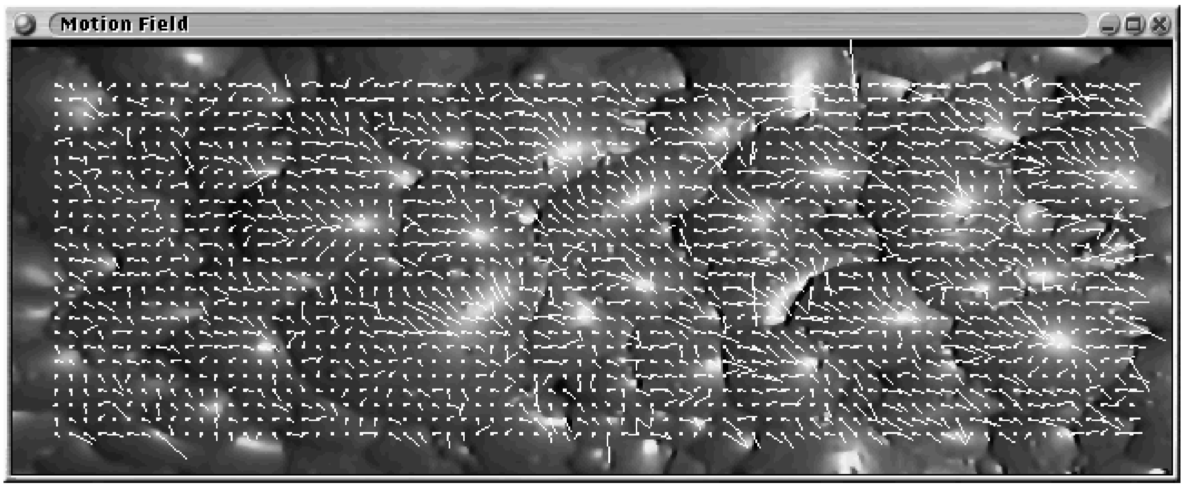
16.2.5 Application to Horn and Schunk Optical Flow

Horn and Schunk's motion estimation algorithm is embedded in Framework 2 and applied to the standard test images for motion (see figure 13.1). Horn and Schunk optical flow is described in section 15.1 (see [68] for details).

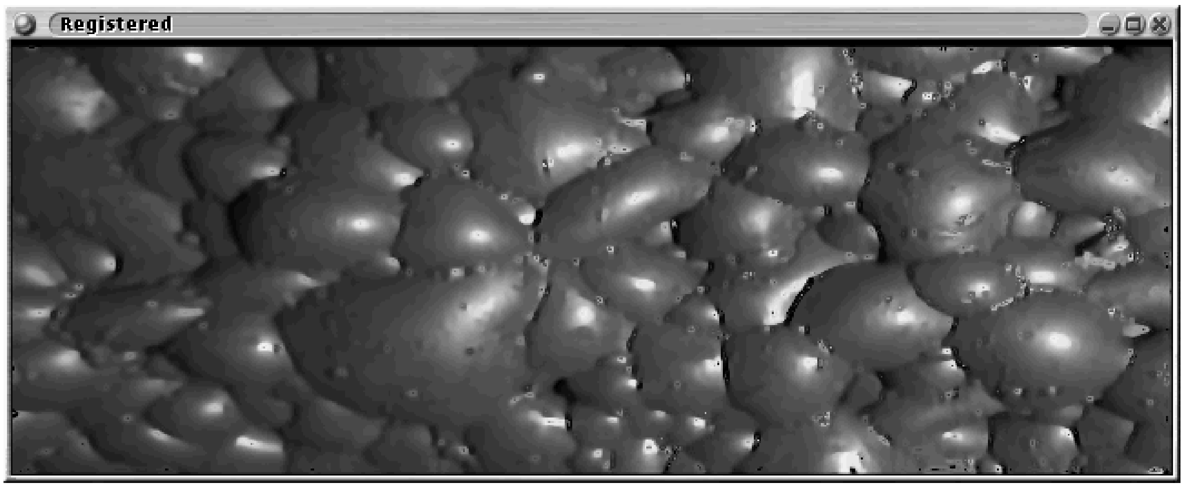
The application of Horn and Schunk's is shown in figure 16.9. The average motion vector is $(1.0,\ -0.53)$, and the energy in the error image is $4\,054\,200$.

16.3 Discussion

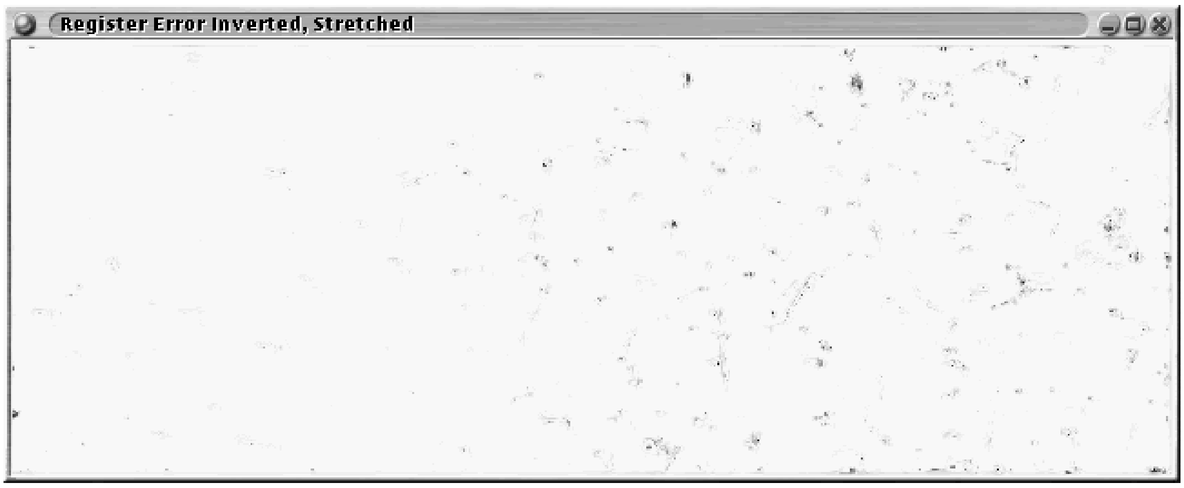
The frameworks have the advantage of creating a correlated motion field. For froth images where there is a definite trend in the motion of the froth, this can be beneficial. For image sequences composed of separate objects moving with different velocities in different directions (the general motion estimation problem) this can be a disadvantage. Framework 2 is a general framework for any motion estimation algorithm, and across the range of motion estimation algorithms examined performs well. As an example of the benefits a hierarchical framework offers, consider the full search case. The energy in the error image is low, despite the small search blocks and search regions used.



(a) Motion field, scaled by factor of 4. Average motion $(1.0, -0.53)$.



(b) Registered image.



(c) Registered error, stretched. Energy in error image is 4 054 200.

FIGURE 16.9 Hierarchical motion estimation, via Horn and Schunk, using Framework 2.

For block motion estimation algorithms Framework 1 offers the better framework, provided the BME algorithm covers the full range of motion within the image. This is apparent if one compares the full search to the SEA cases. The full search algorithm examines a small part of the motion search space compared to the SEA algorithms. For watershed based motion estimation Framework 2 significantly improves the results.

The motion vectors generated by Frameworks 1 and 2, for the cases where both frameworks are used, do not agree well, even with the large variation allowed, see table 16.1. This illustrates a difficulty in comparing motion estimation algorithms in the absence of some base-line (or, knowledge of the actual motion).

Motion Estimation	Motion Vectors within 50%			Motion fields
Algorithm	Length	Orientation	Both	compared visually
Full Search	74.6	62.3	53.0	Similar
MAD SEA	80.7	68.0	59.6	Similar
MSE SEA	81.2	68.2	60.1	Similar
Watershed based motion estimation	57.9	53.9	34.4	Different

Chapter 17

Detecting bubble bursts and merges

This chapter examines the effect of bubble bursts or merges on the motion estimation process. It is hypothesised that a burst or merging bubble will result in a discontinuity within the motion vector field. This chapter will consider the suitability of several motion estimation algorithms for detecting burst or merge events. A framework for the detection of these events will be proposed.

Once a motion field is generated for a pair of successive images, and one suspects the presence of burst or merge events, the motion field needs to be processed. This may be done by computing the variance of the X and Y components of the motion vector field and thresholding. Several thresholding techniques exist, of these Lorentz thresholding will be employed. Lorentz thresholding is an example of a parameter free thresholding technique and is explored subsequently.

17.1 Lorentz thresholding

This is an instance of a percentage thresholding technique [51][pg. 216-218]. The technique was proposed by Goel and Vidakovic [49] in the context of wavelet based denoising. The Lorentz curve of a data set is defined as follows,

$$S(x) = \frac{\sum_{k=1}^{x} m(k)}{\sum_{i=1}^{N} m(i)} \tag{17.1}$$

where, $m(k)$ denotes the members of the population ranked in increasing magnitude. x, k , i are remapped to lie within 0 and 1. Originally, $S(x)$ denoted the total wealth of a number of individuals in a population, if the individuals were ranked in order of increasing wealth [49, 79, 37].

The threshold is selected by setting the pixels below the average energy of the image to zero. Lorentz thresholding sets the pixels at, or above, the average energy to the maximum value allowed (for example, 255). Lorentz filtering leaves coefficients with energy above, or at, the average energy alone.

The proportion of pixels remapped is the proportion of pixels with energy less than, or at, the average energy of the pixels. The lower $p_0 \times 100$ pixels are removed (set to zero).

$$p_0 = \frac{1}{N} \sum_{k=1} N I(m^2(k) \leq \overline{m^2}) \tag{17.2}$$

where n is the number of coefficients, and $I()$ is an indicator function,

$$I(w) = \begin{cases} 1, I(m^2(k) \leq \overline{m^2} \\ 0, \text{otherwise} \end{cases} \tag{17.3}$$

and,

$$\overline{m^2} = \frac{1}{N} \sum_{k=1} N m^2(k) \tag{17.4}$$

A Lorentz thresholded image may not localise the regions of interest (pixels with large energies) sufficiently, this implies a pre-processing stage is needed. To reduce the number of pixels thresholded, the high energy pixels should be increased in magnitude and the low energy pixels decreased. This makes the Lorentz curve more convex, and more coefficients then lie below the average energy, this implies that fewer pixels are thresholded. This suggests the use of an increasing function for re-mapping the pixels intensities. An increasing function, by definition, has the following property, $f(x + \Delta x) \geq f(x)$, for some $\Delta x \leq 0$, which implies $f'(x + \Delta x) \geq f'(x)$ also holds [75, 134].

17.2 Comparing motion estimation algorithms

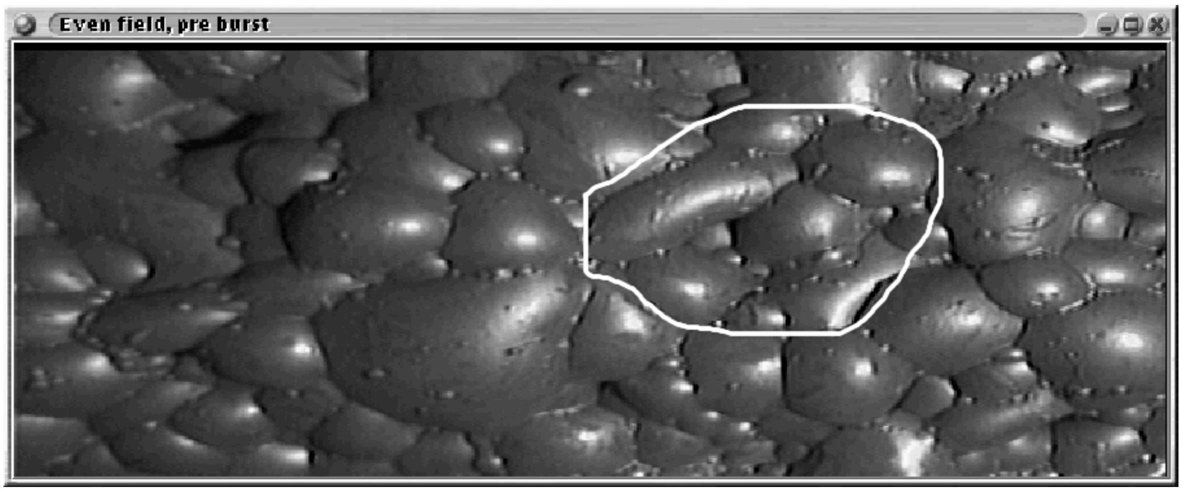
Consider a bubble bursting, provided the surrounding bubbles are unaffected, the bubble (or bubbles) beneath the burst bubble are exposed. Similarly two bubbles merging result in the highlights of both bubbles disappearing and a new highlight being generated elsewhere on the new bubble. A burst or merge event generates a new region, or regions, in an image which have no analogue in the previous image. This implies that a region matching motion estimation algorithm will have difficulty matching regions in the unmerged, or burst, bubbles to regions in the bubbles of the next image. In optical flow type algorithms, this leads to large temporal errors which implies large motion.

It is then reasonable to expect that large errors will be generated in the motion estimation process if bubbles burst or merge. This suggests that the difference between the warped and reference image (the error image) will show large errors at the discontinuity, or in the region of this burst or merge. Pixels with large intensities on the error image may then be used as indicators of motion discontinuities. In view of the qualitative performance of the algorithms considered in the previous two chapters, and brevity, only Horn and Schunk's gradient based optical flow algorithm [67] (see section 15.1), and the hierarchical MSE based SEA algorithm will be considered. It is important that the motion estimation algorithm used generates small error images (i.e. is a good motion estimator). Small error images imply that a large error is significant, and should have an identifiable cause in the image pair considered. This prevents false positives (incorrectly detecting a burst or merge event), and improves localisation. Figure 17.1 shows a bubble burst, and the images immediately before and after. Similar for figure 17.2, only for bubble merges. Given that the motion estimates at the edges of the images are suspect (this depends on how pixels outside the boundaries of the image are handled by the motion estimation algorithm), the edge pixels, and pixels close to the edges, should be ignored for the purposes of post processing of the motion field.

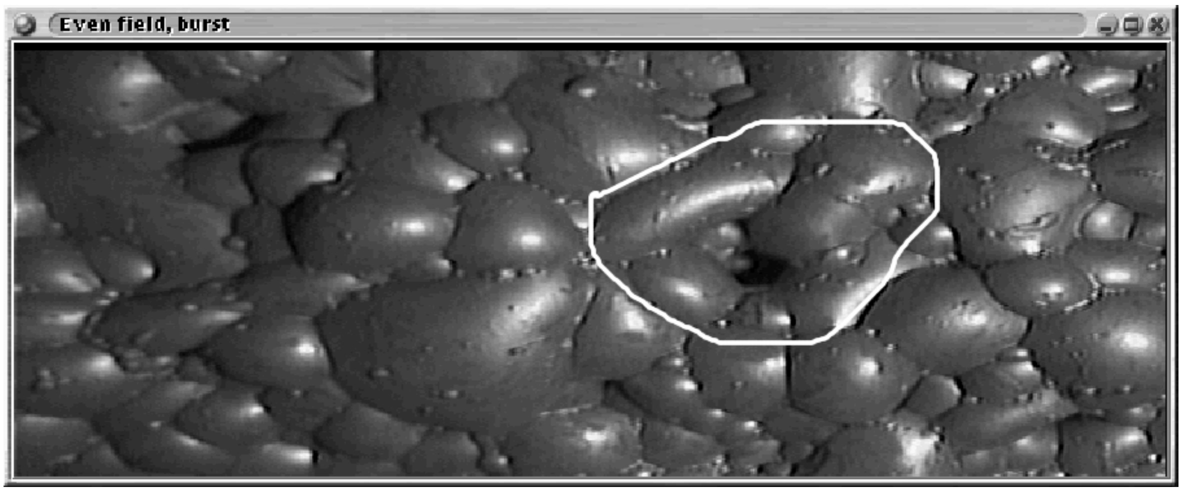
Once the motion fields are computed, the variance of the X and Y components of the motion field are computed. The variance is computed within a disc shaped neighbourhood of radius 8. This large variance allows for local deviations form the average motion to be smoothed away, while not smoothing away the discontinuity caused by the burst or merge event. The radius is chosen to localise the motion field discontinuities well, but at the same time prevent fragmentation of the discontinuity on the thresholded variance image. This explains the somewhat circular regions visible on the variance, and thresholded variance, images (see figure 17.4 and 17.5 for example). The composite variance image is formed by the pointwise maximum of the X and Y variances. This preserves pixels where the variances are large.

Thresholding the variance images provides localisation of the motion vector discontinuities. Lorentz filtering [37, 49] is used in an attempt to threshold the variance components of the motion field in a parameter free way. Prior to thresholding, the variance images are pre-processed by taking each pixel to its 5-th exponent, and then rescaling the pixel values to lie between their original maximum and minimum values. The pre-processing stage preferentially maintains the larger pixel intensities. Admittedly, using a pre-processing stage replaces the choice of a simple threshold, for example, by the choice of the shape of the pre-processing function and its parameters. Lorentz thresholding, and pre-processing prior to Lorentz thresholding, is explored in section 17.1. Investigation shows that the preprocessing used provides good localisation of the events, without including minor eddies and swirls in the motion field.

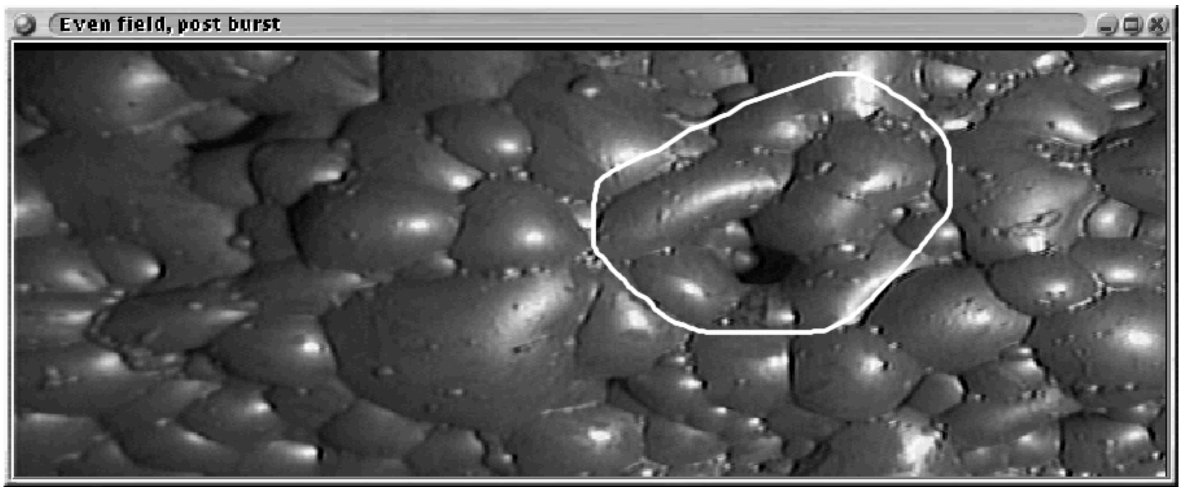
A motion field allows for the warping of the next image in the sequence. This is essentially an estimation of the current image, given the motion estimate between the current and next image. The absolute difference between the estimated and current image, yields the error in approximation. The thresholded approximation error yields pixels where the motion estimation process does not predict the change in pixel intensities due to motion. This implies that the thresholded pixels occur where sudden changes in intensity occur. This includes bubble bursts and merges. Examination of the burst test images, show changes in the position and intensity of highlights. It is expected then, that these highlight changes will show themselves in the thresholded approximation. Again, Lorentz filtering is used with a pre-processing stage which takes the 5-th exponent of the pixels in an image before re-mapping the pixels intensities to lie in the range of intensities of the original image. The pre-processing stage allows for significant errors to be localised without including less significant ones.



(a) Image at $t - \Delta t$.

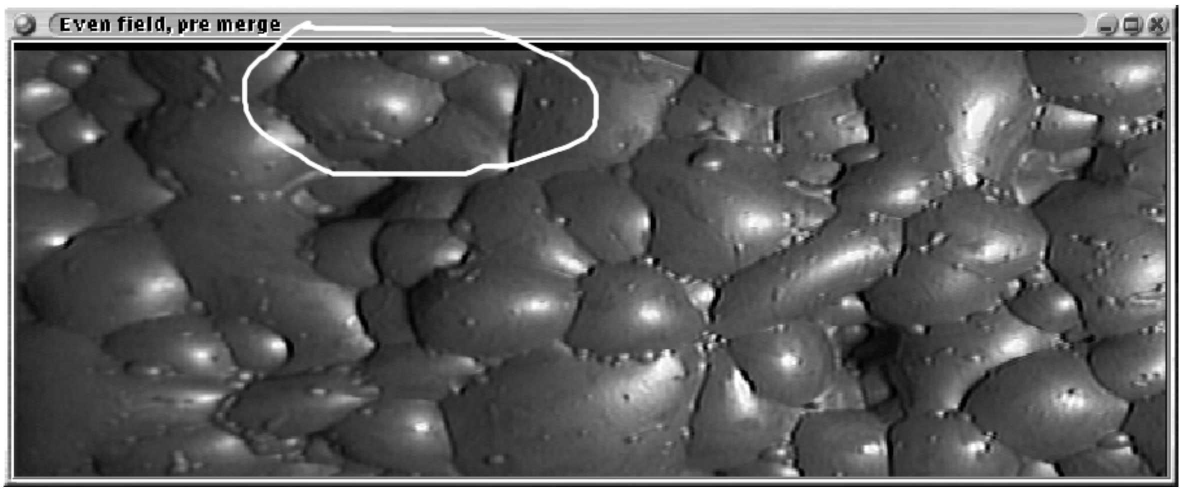


(b) Image at t.

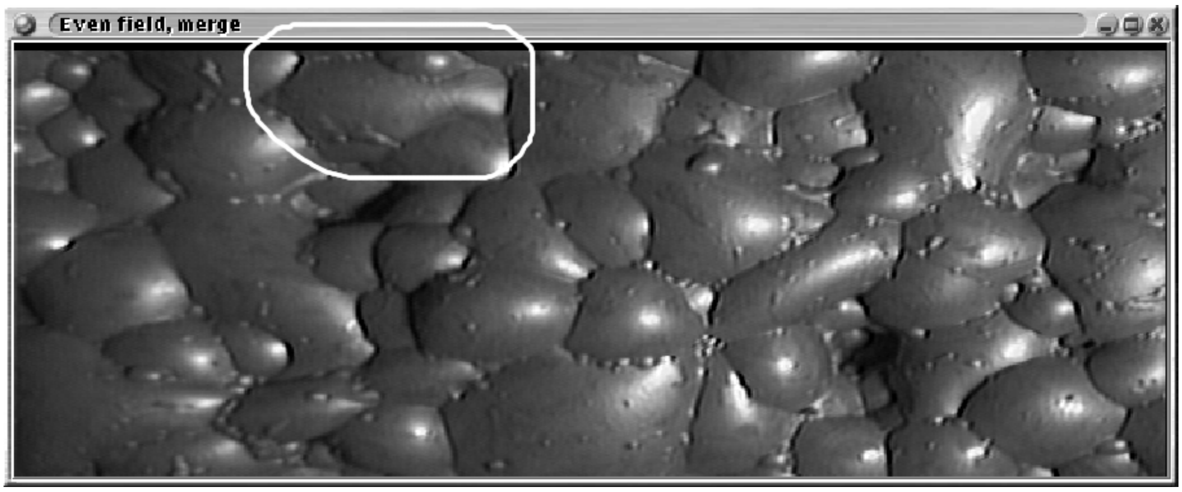


(c) Image at $t + \Delta t$.

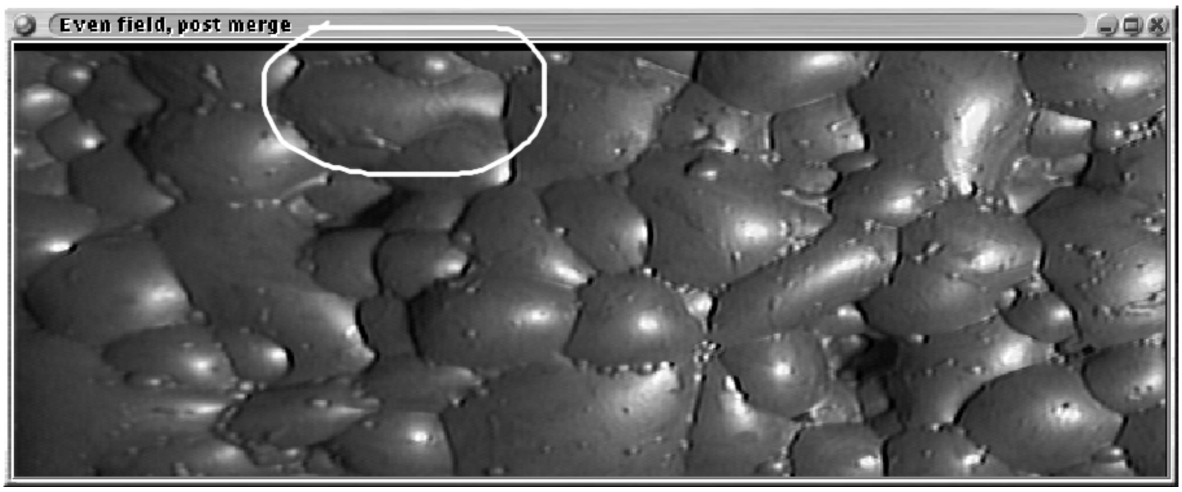
FIGURE 17.1 A bubble burst event.



(a) Image at $t - \Delta t$.



(b) Image at t.



(c) Image at $t + \Delta t$.

FIGURE 17.2 A bubble merge event.

The parameters are found by investigation.

This does suggest an iterative improvement motion estimation scheme. If the prediction error at a pixel is larger than some threshold, or the average error is larger than some threshold, then a new motion estimate for that pixel is computed with finer accuracy. With block motion estimation algorithms, for example, this would involve larger block and search window sizes. For optical flow type algorithms this may involve more iterations during the update stage.

The remainder of this chapter examines the issue of localising the discontinuities within the motion vector field.

17.2.1 Horn and Schunk's algorithm

Horn and Schunk's algorithm is evaluated for its utility in detecting burst of merge events. The algorithm parameters used are: alpha is $\alpha = 1$ and the number of iterations 16. These are the same parameters used for motion estimation (see section 15.1). These parameters are selected to give the best backward predicted frame at time t, given the motion field and the next image in the sequence at time $t + \Delta t$.

A bubble burst

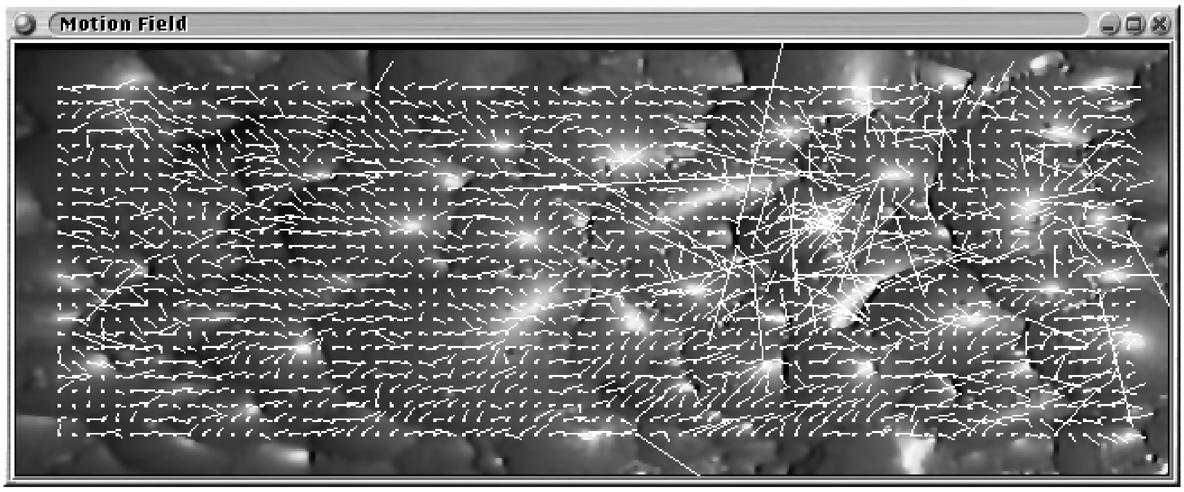
Horn and Schunk's algorithm is applied to the images at $t - \Delta t$ and t of the burst test images (figure 17.1). Figure 17.3 shows the effect of a bubble burst on the motion field. Figure 17.4 shows the X, Y, and the point-wise maximum of the X and Y variances of the motion field. Figure 17.5 shows the thresholded variances. Note that the Lorentz thresholded composite variance image shows only two objects of significant size, these correspond to the bubble burst and a merge event respectively. The smaller objects correspond to simple highlight changes. The energy in the difference between the image at time t and the predicted image is $11\,258\,428$.

The energy in the difference image when one repeats the process for the next image, that is, generating a new motion field for images at time t and $t + \Delta t$, and then using the motion field to predict the image at time t using the image at time $t + \Delta t$, yields the energy in the error image as $6\,079\,348$. Examination of the images in question (see figure 17.1) show that no burst or merge event is visible, although some highlight changes do occur.

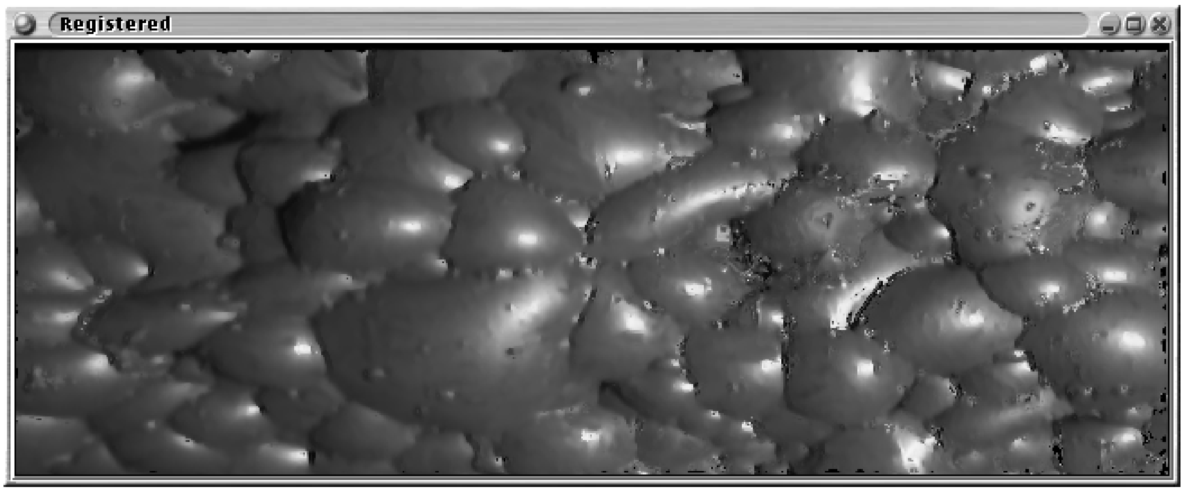
Figure 17.6 shows the thresholded highlights which are coincident with the thresholded motion discontinuities. These show that the burst event is coincident with some highlight changes and the merge event with a significant one. The smaller objects are then highlight changes, but not significant.

A bubble merge

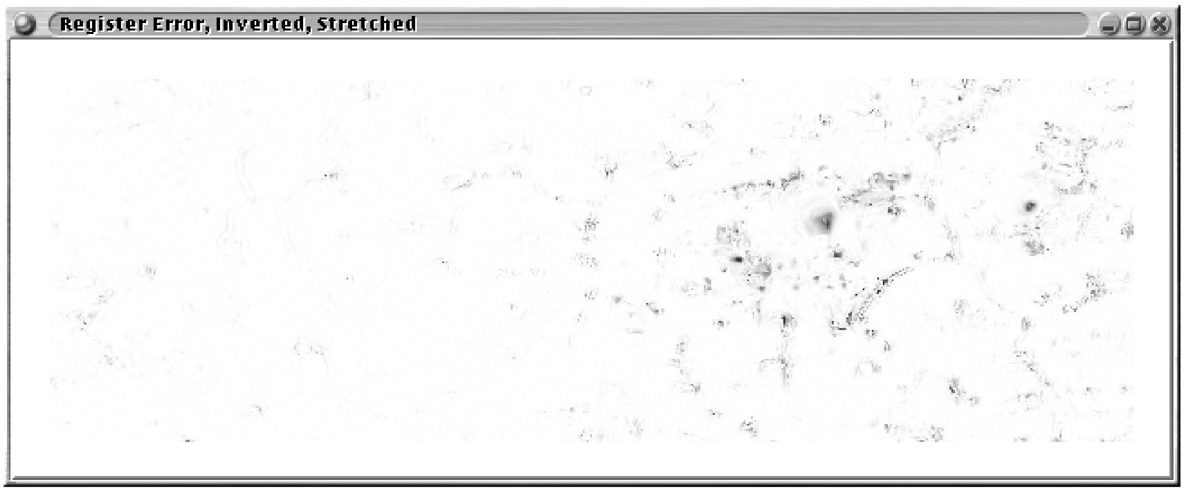
Similarly to the burst detection case, figure 17.7 shows the effect of a bubble merge. Figure 17.8 shows the X, Y, and the point-wise maximum of the X and Y variances of the motion field. Figure 17.9 shows the thresholded variances. Note that the Lorentz thresholded composite variance image shows a single



(a) Motion field, scaled by factor of 4. Average motion $(1.6, \ -0.11)$.

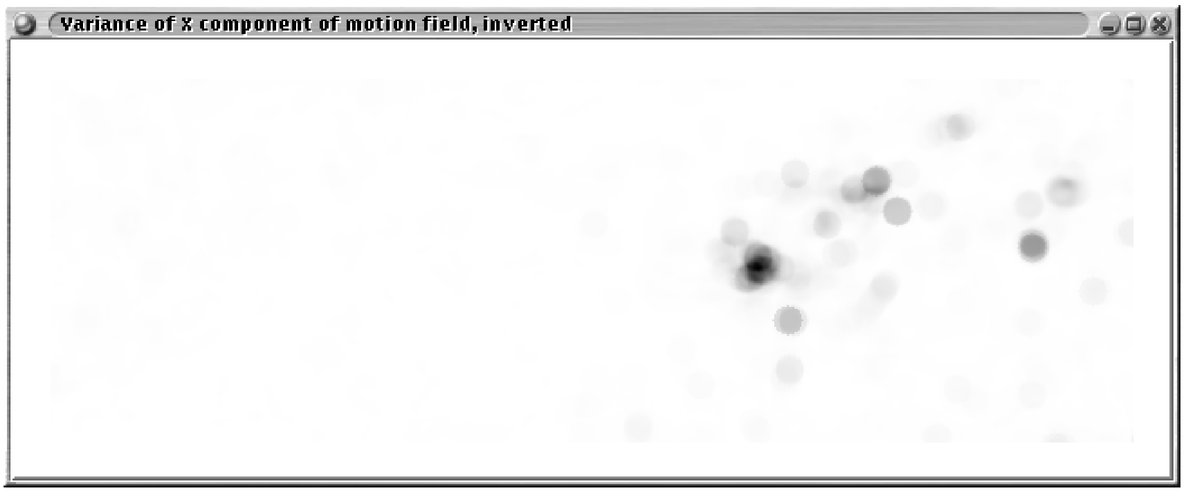


(b) Registered image.

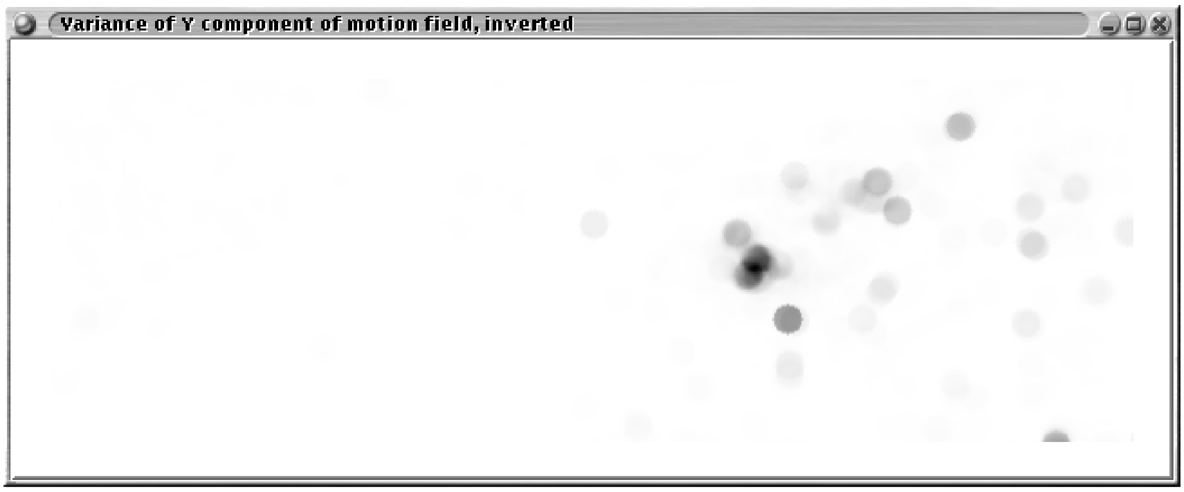


(c) Registered error, stretched. Energy in error image is 11 258 428.

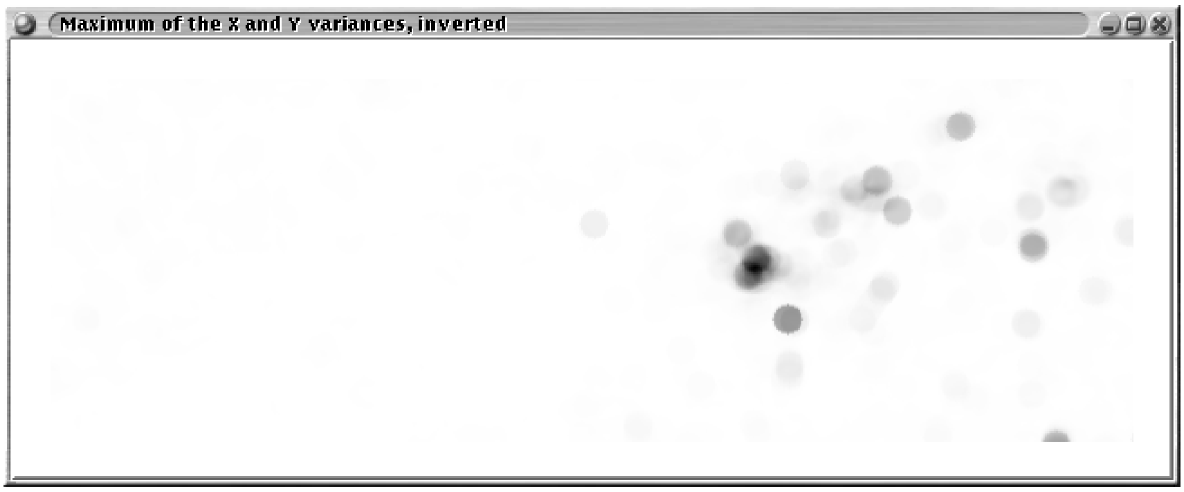
FIGURE 17.3 Effect of a bubble burst, measured using Horn and Schunk's algorithm.



(a) Variance of the X component of the motion field, inverted.

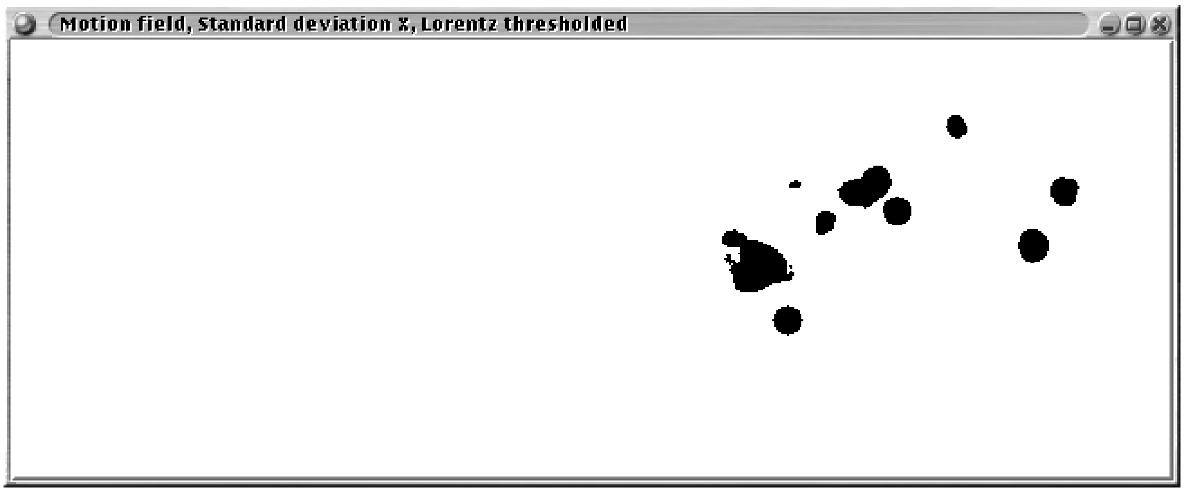


(b) Variance of the Y component of the motion field, inverted.

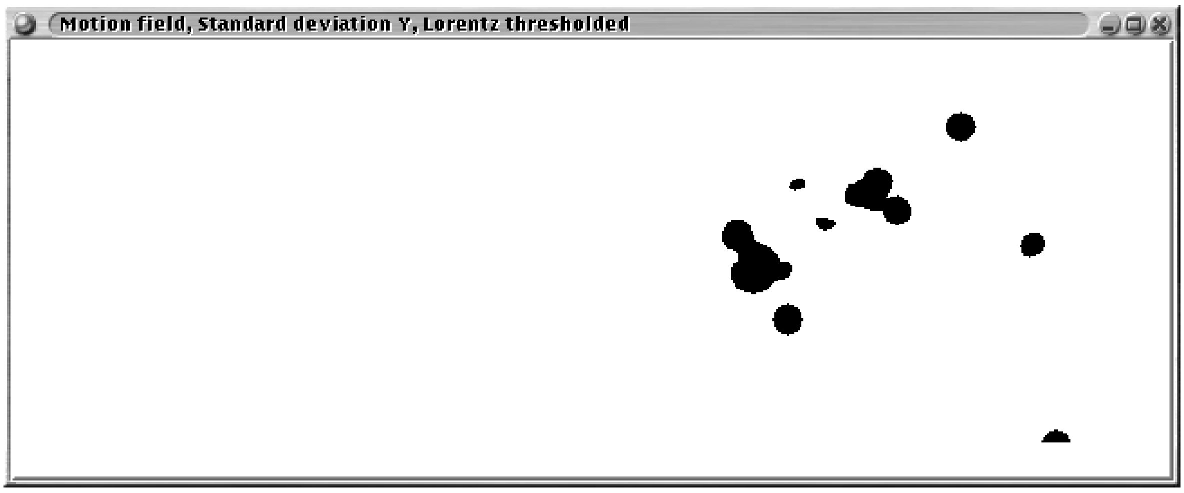


(c) Maximum of the variance of the X and Y components of the motion field, inverted.

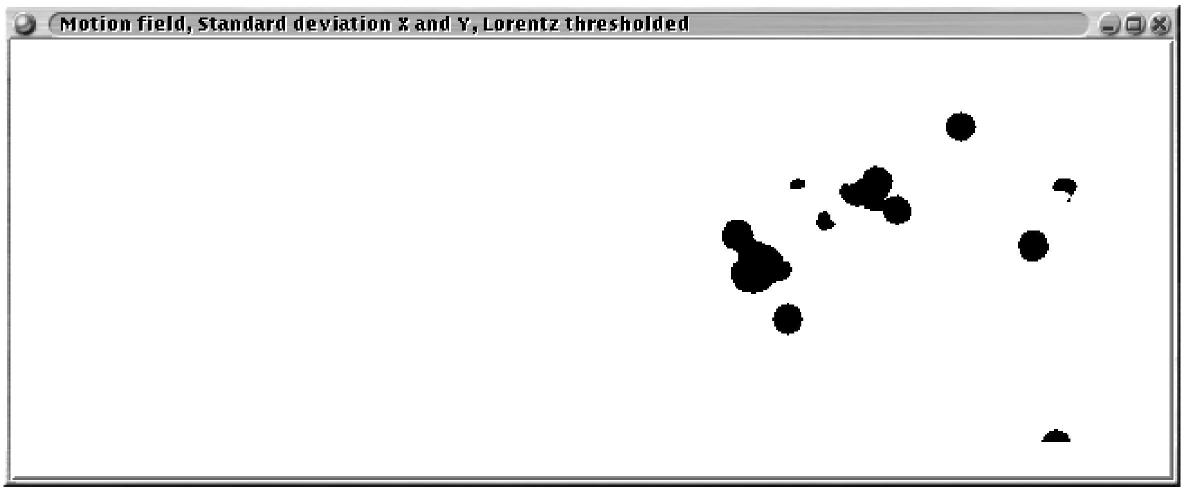
FIGURE 17.4 The variances of the X and Y components of the motion field, via Horn and Schunk.



(a) Variance of the X component of the motion field, thresholded, inverted.

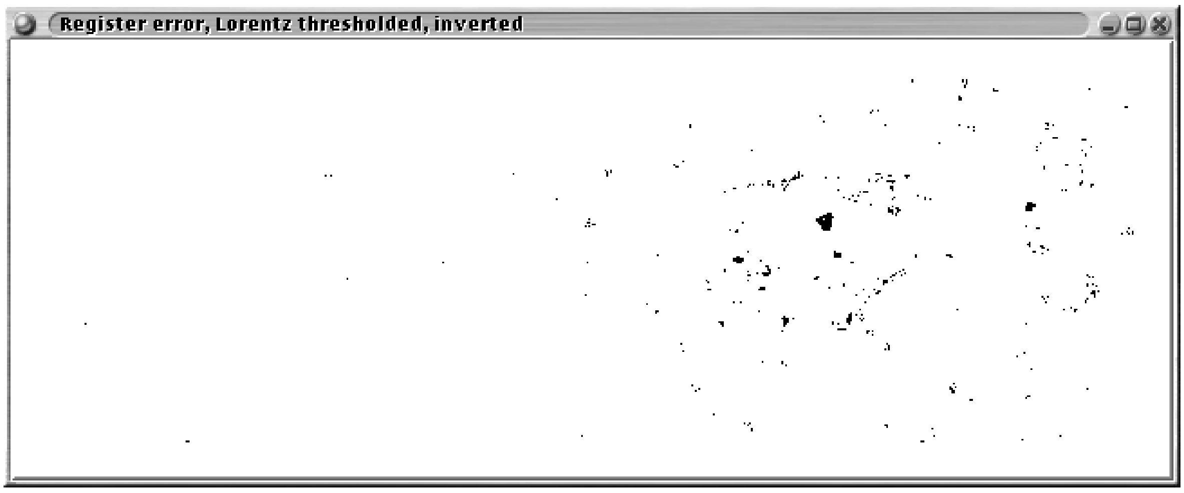


(b) Variance of the Y component of the motion field, thresholded, inverted.

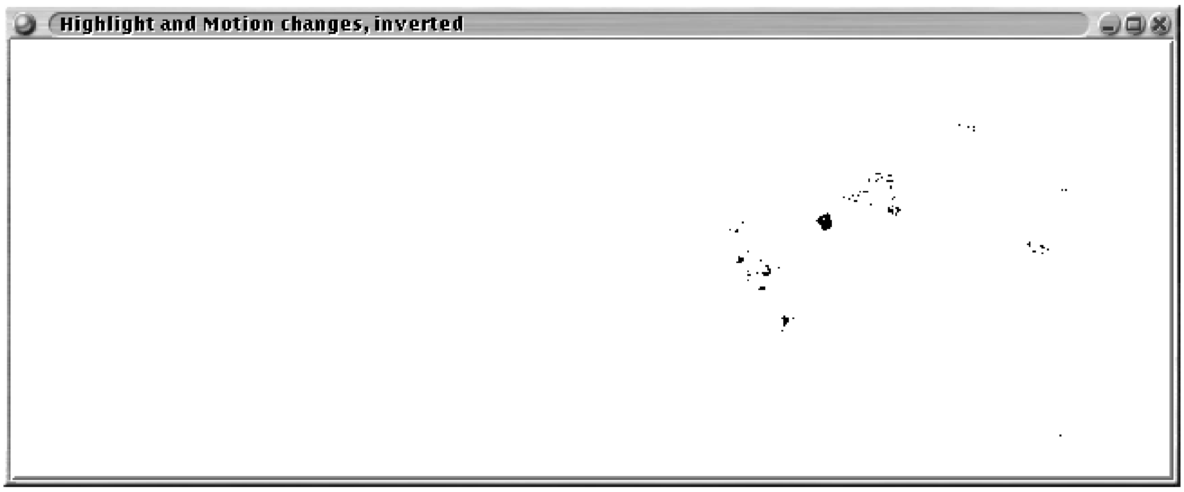

Motion field, Standard deviation X and Y, Lorentz thresholded

(c) Intersection of the thresholded X and Y component of the motion field, inverted.

FIGURE 17.5 The thresholded variances of the X and Y components of the motion field, via Horn and Schunk.



(a) Lorentz thresholded error image.



(b) Thresholded highlights coincident with motion discontinuities.

FIGURE 17.6 Thresholded highlights and coincident with motion discontinuities, via Horn and Schunk.

object, which corresponds to the merge event. The energy in the error image is $12\,083\,425$. The error image is generated by predicting the image at time t using the motion field and the image at time $t + \Delta t$. If the error estimation process is repeated for the images at time t and $t + \Delta t$, then the error is $12\,147\,623$. Examination of the images in question (figure 17.2) show that no burst or merge event is visible.

Figure 17.10 shows the thresholded highlights which are coincident with the thresholded motion discontinuities. These show that the merge event coincides with a single significant highlight change.

17.2.2 Hierarchical Minimum Absolute Difference (MAD) based Successive Elimination Algorithm (SEA)

The Hierarchical MAD based SEA algorithm is examined to test its suitability for burst and merge detection. The algorithm parameters used are: a search block size of 7×7 and a search area of 15×15. These parameters are the ones used in the general motion estimate in (see section 16.2.2) and allow for a tradeoff of performance vs. speed.

A bubble burst

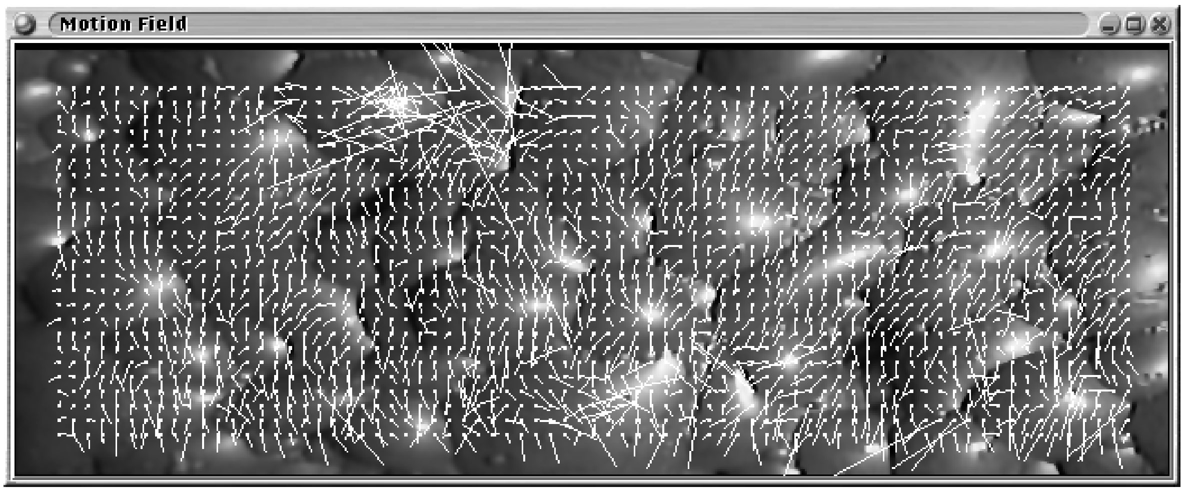
Hierarchical SEA is applied to the images at time $t - \Delta t$ and t of the burst test images (figure 17.1). Figure 17.11 shows the effect of a bubble burst on the motion field. Note the motion field is especially uncorrelated at the burst. Figure 17.12 shows the variance of the X and Y components of the motion field. Figure 17.13 shows the variance images thresholded. Note that several regions are extracted by the thresholding of the point-wise maximum of the variance of the X and Y components of the motion field. The point-wise maximum operation ensures that the composite variance is large whenever either of its components is large. Careful comparison of the positions of these regions to the test images (see figure 17.1), show the following: a bubble burst, and regions corresponding to highlight changes. The energy in the error between the predicted and the image at time t is $2\,893\,428$.

Figure 17.14 shows the thresholded highlights coincident with the thresholded motion discontinuities. A significant highlight change, and three clustered smaller highlight changes are detected. The clustered trio of highlight changes occur in the vicinity of the bubble burst.

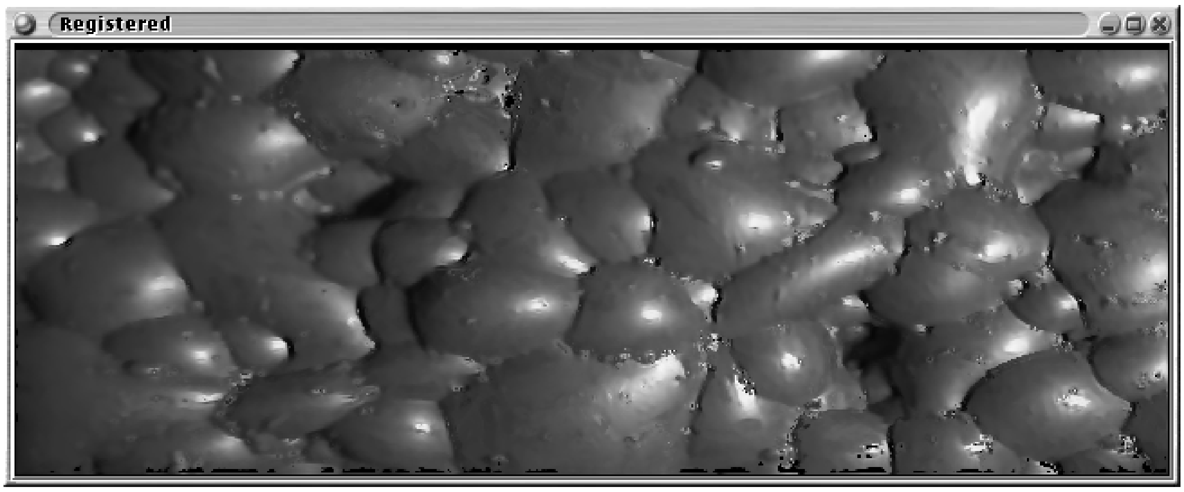
If the process is repeated for the two succeeding images, at time t and time $t + \Delta t$ respectively of the burst test images (figure 17.1), then the energy in the error between the predicted and the image at time t is $2\,019\,299$. Examination of the images in question (figure 17.1) show that no burst or merge event is visible, though some highlight changes do occur.

A bubble merge

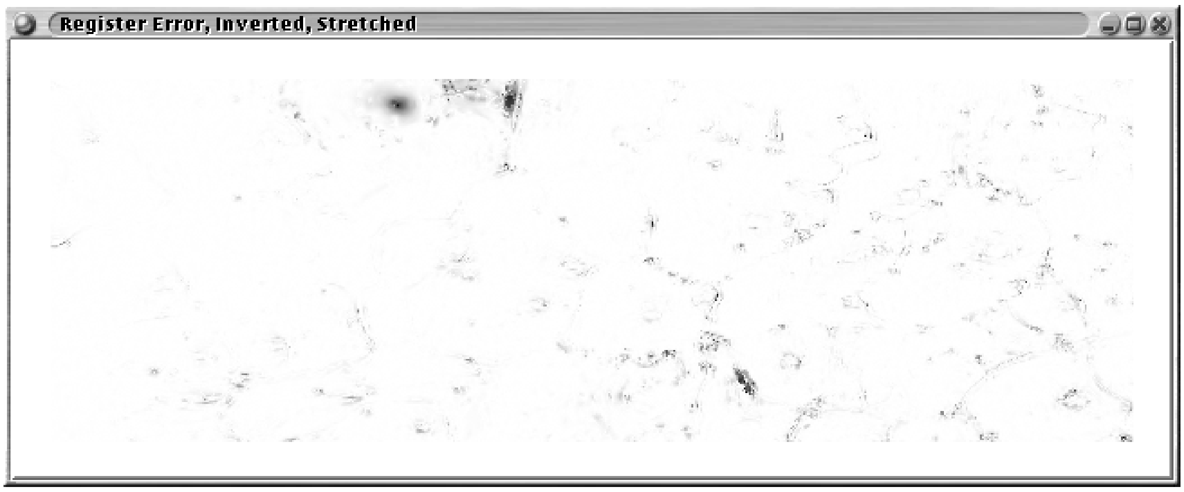
The process is repeated, but for a bubble merge event. Figure 17.15 shows the effect of a bubble merge. Note the correlated motion field. Figure 17.16 shows the variance of the X and Y components



(a) Motion field, scaled by factor of 4. Average motion $(-0.17,\ -1.6)$.

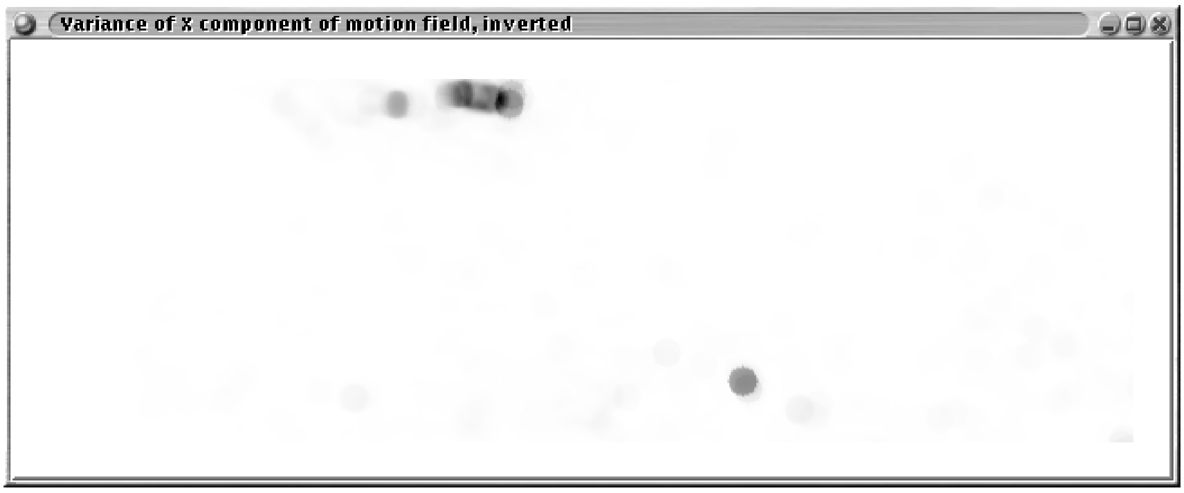


(b) Registered image.

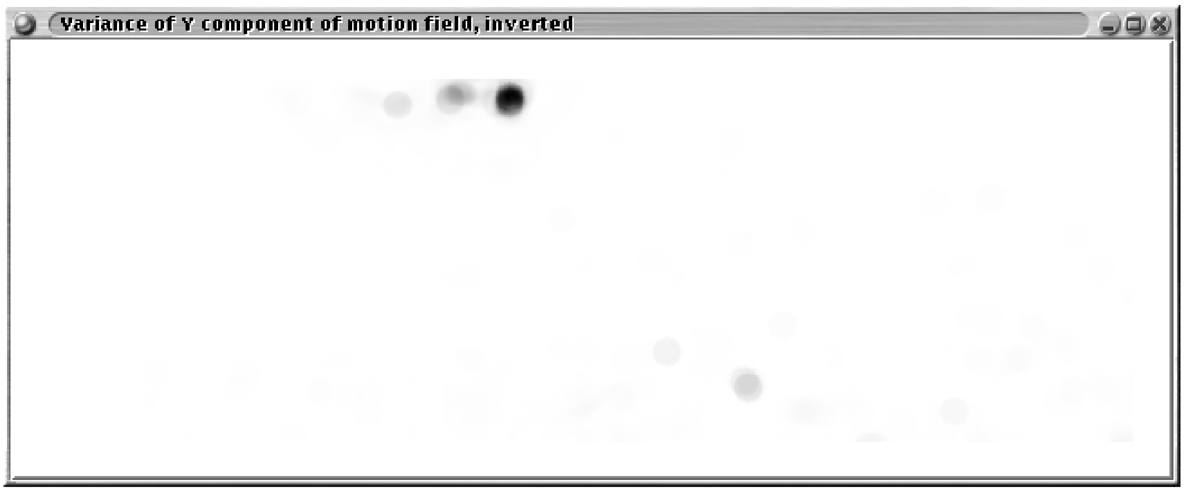


(c) Registered error, stretched. Energy in error image is 12 083 425.

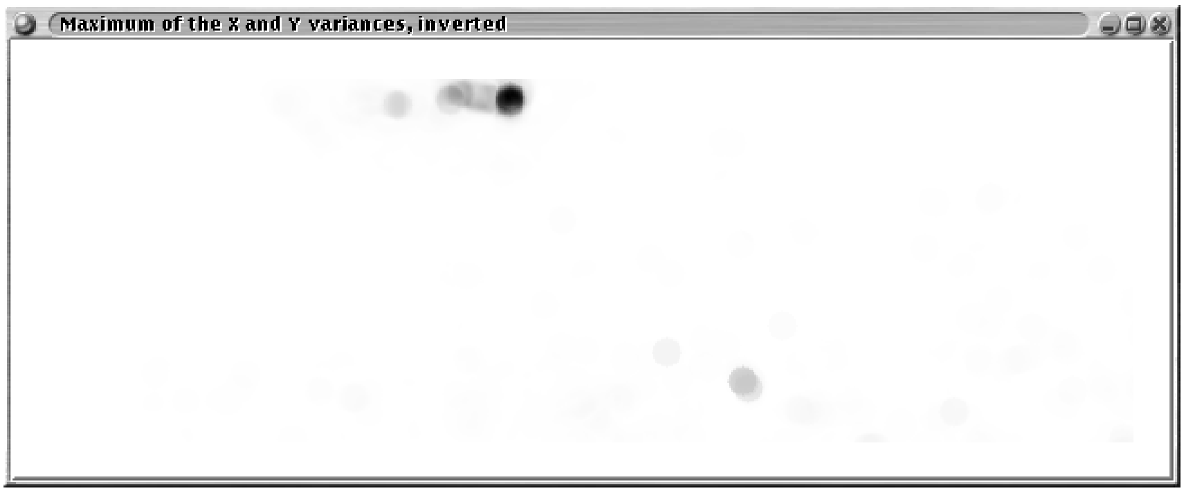
FIGURE 17.7 Effect of a bubble merge, measured using Horn and Schunk's algorithm.



(a) Variance of the X component of the motion field, inverted.

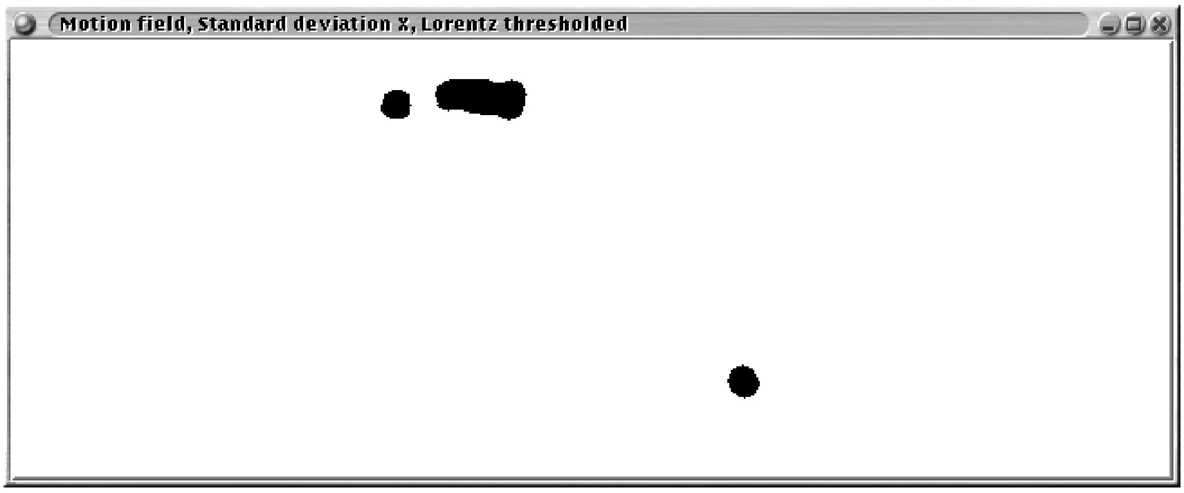


(b) Variance of the Y component of the motion field, inverted.

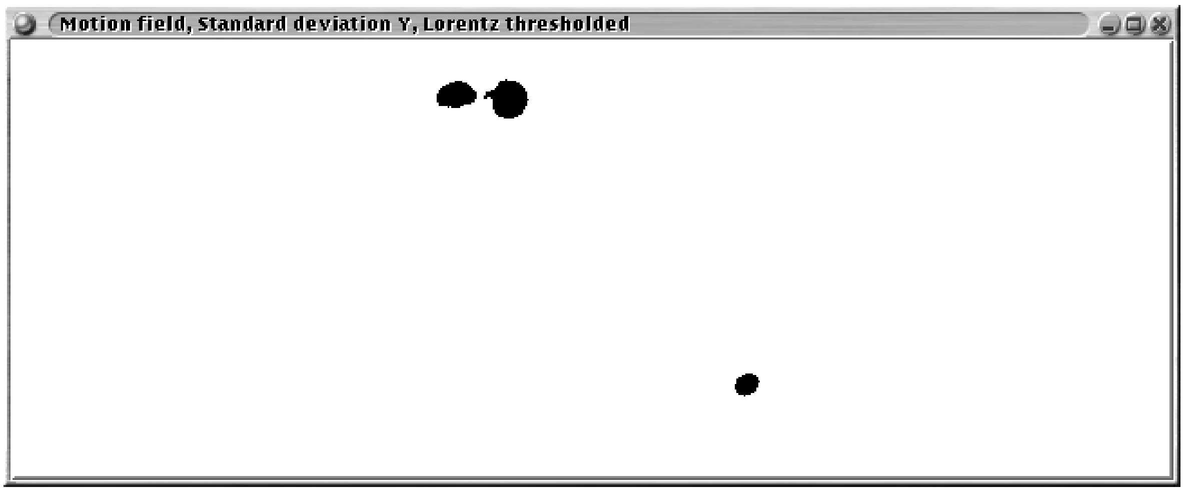


(c) Maximum of the variance of the X and Y components of the motion field, inverted.

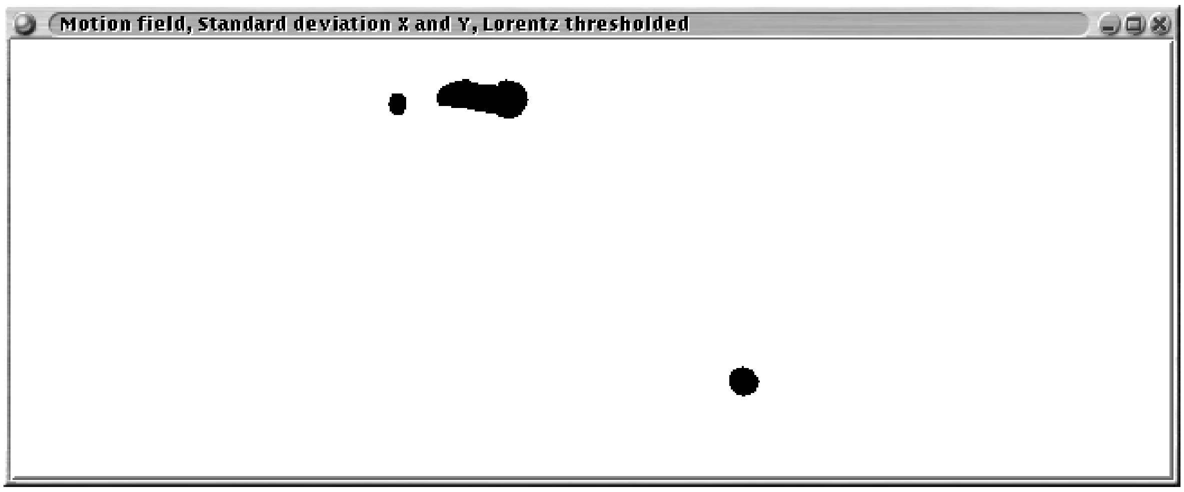
FIGURE 17.8 The variances of the X and Y components of the motion field, merge event, via Horn and Schunk.



(a) Variance of the X component of the motion field, thresholded, inverted.

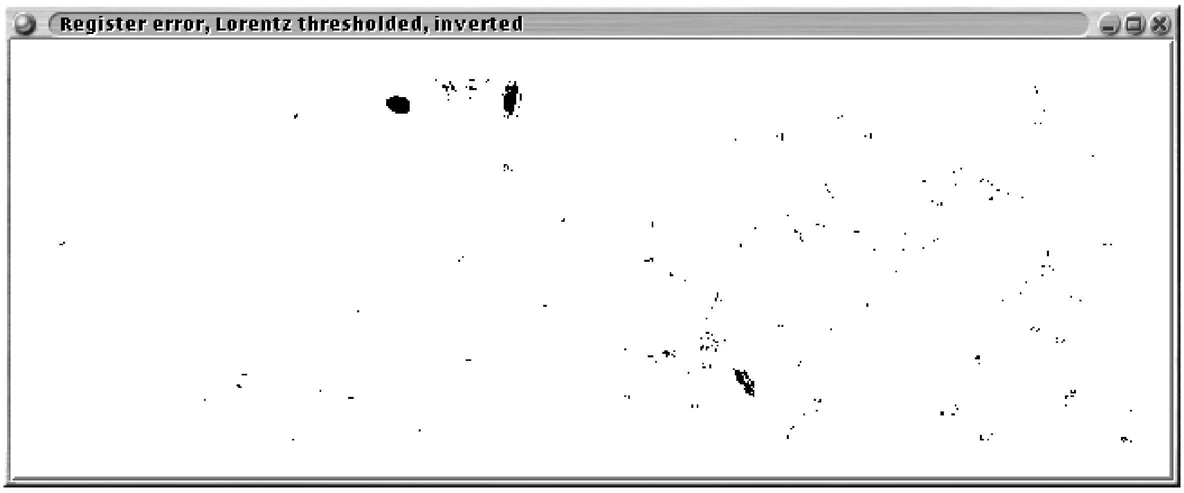


(b) Variance of the Y component of the motion field, thresholded, inverted.

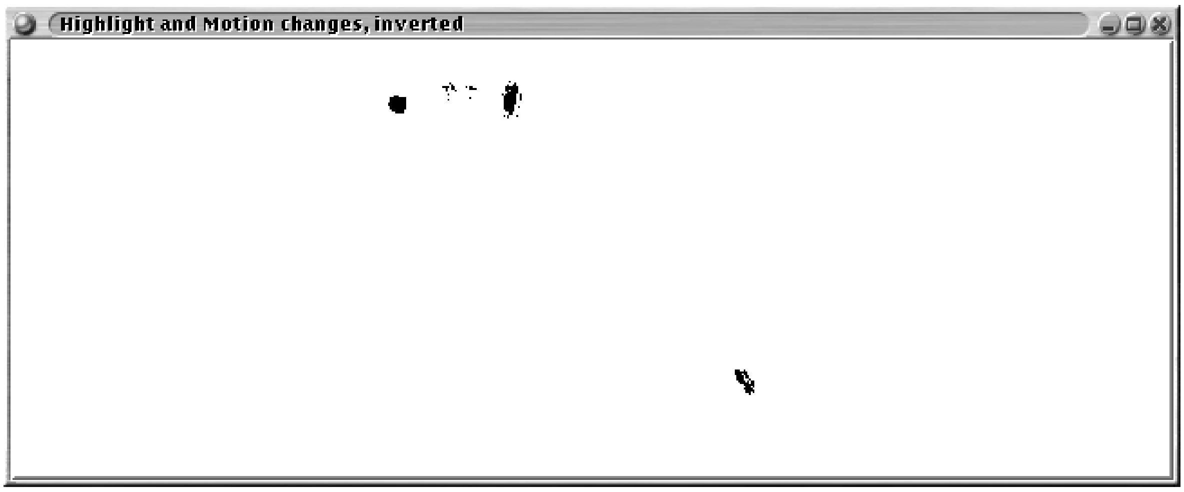

Motion field, Standard deviation X and Y, Lorentz thresholded

(c) Intersection of the thresholded X and Y component of the motion field, inverted.

FIGURE 17.9 The thresholded variances of the X and Y components of the motion field, merge event, via Horn and Schunk.



(a) Lorentz thresholded error image.



(b) Thresholded highlights coincident with motion discontinuities.

FIGURE 17.10 Thresholded highlights and coincident with motion discontinuities, merge event, via Horn and Schunk.

of the motion field. Figure 17.17 shows the variance images thresholded. Note that a single object is visible after Lorentz thresholding. The object corresponds to the position of the merge event.

Figure 17.18 shows a single significant highlight coincident with the thresholded motion composite variance. Figure 17.18 shows that the highlights of the merged bubbles are not predicted well.

The energy in the error between the predicted and the image at time t is $3\,236\,513$. If the process is repeated for the two succeeding images, then the energy in the error between the predicted image and the image at time $t + \Delta t$ is $1\,438\,988$. Examination of the frames in question (figure 17.2) show that no burst or merge event is visible.

17.3 Discussion

Clearly, a bubble burst, merge, or highlight change causes a discontinuity in the motion vector field. Thresholding the variance of the X and Y components of the motion field results in the localisation of the discontinuities in the motion vector field. Discontinuities in the motion field are not limited to bursts or merges, to a lesser extent discontinuities may be caused by different parts of the froth having different local motion. A second cause lies in the change of aspect of a bubble, causing the highlight to change position.

Of the pair of algorithms considered for burst detection, Horn and Schunk's algorithm performs better than the Hierarchical MSE based SEA algorithm. Horn and Schunk's algorithm provides good localisation and shows a large difference between the average error energy between the image pair with the burst event and the next pair of images.

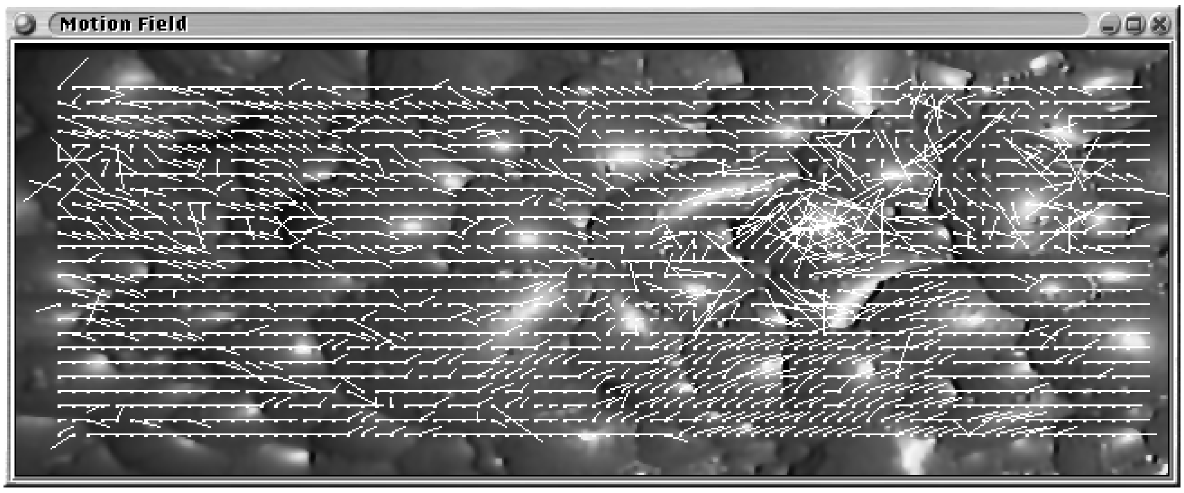
Merge detection, however, shows Horn and Schunk's algorithm providing good localisation. The difference in the average error between the image pair containing the merge and the next image pair is not significant. The Hierarchical MSE based SEA algorithm does not localise as well, but shows a clear distinction in the average prediction error between the image pair with the merge and the successive pair of images.

The localisation difficulty with the Hierarchical MSE based SEA algorithm is likely due to the good motion estimates provided at each pixel. (A motion estimate can be ranked in terms of the prediction error for the associated motion estimate.) The fragmentation may be partially offset by less pre-processing of the error image prior to Lorentz thresholding. This, clearly visible in the Lorentz filtered images, has the result of localising pixels with less significant prediction errors of motion estimate variances. On the other hand, these other regions sometimes correspond to highlight changes.

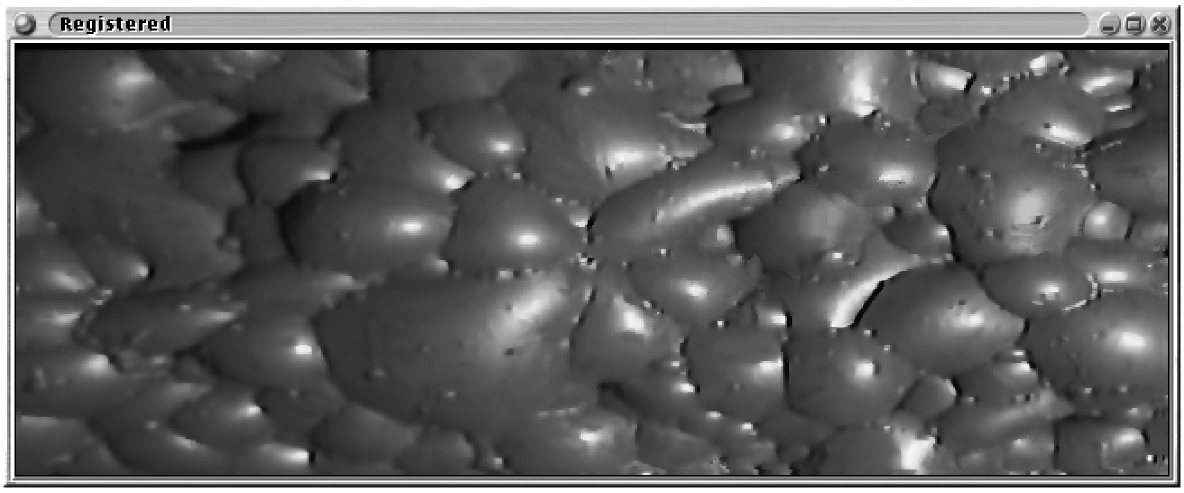
The Hierarchical MSE based SEA algorithm shows significant average prediction errors for burst and merge events. This suggests a method for finding images in a sequence where changes have occurred, images are selected for further analysis if the prediction error is above some threshold, or significantly larger than the average prediction error for the previous image pair.[1]

It is clearly feasible to find candidate images in a video sequence for significant bubble bursts,

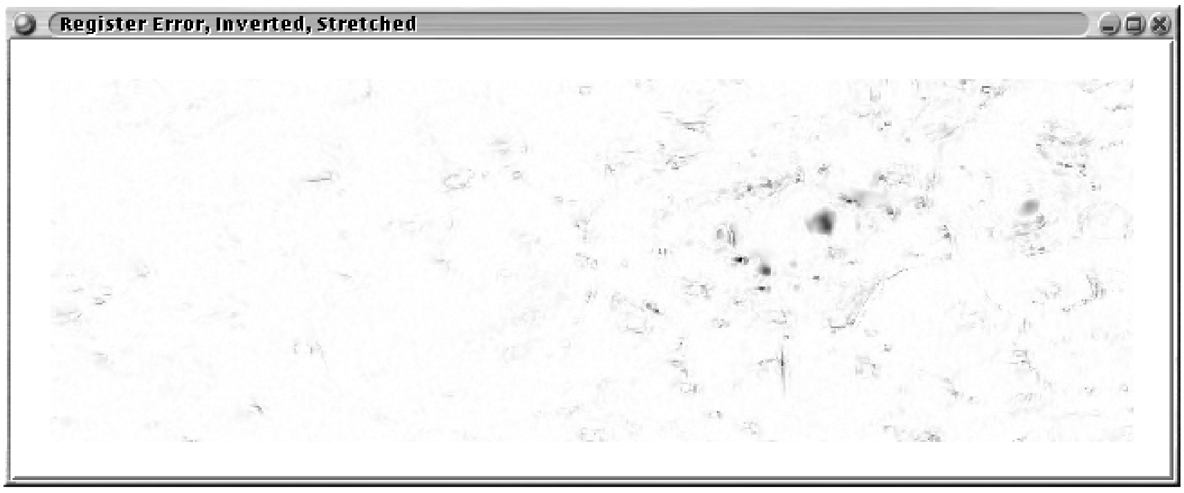
[1]This may also be utilised for camera, or scene, change detection in television sequences, for example.



(a) Motion field, scaled by factor of 4. Average motion $(2.2, -0.10)$.

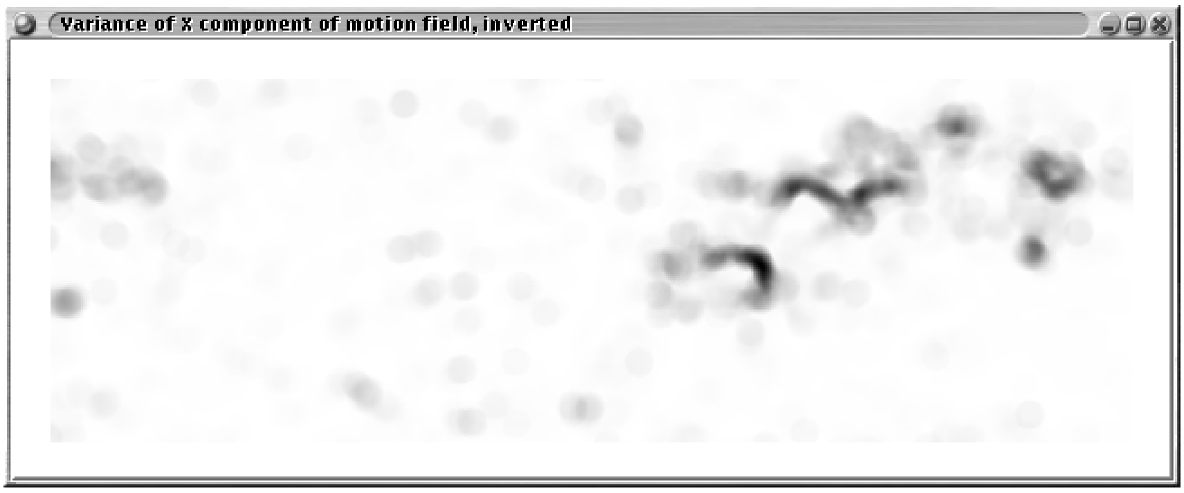


(b) Registered image.

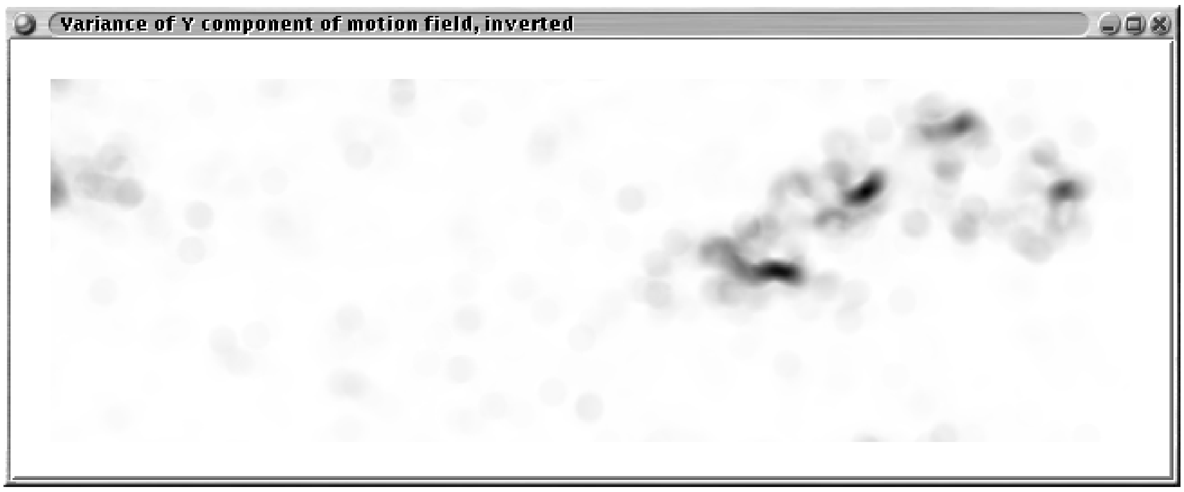


(c) Registered error, stretched. Energy in error image is 2 893 428.

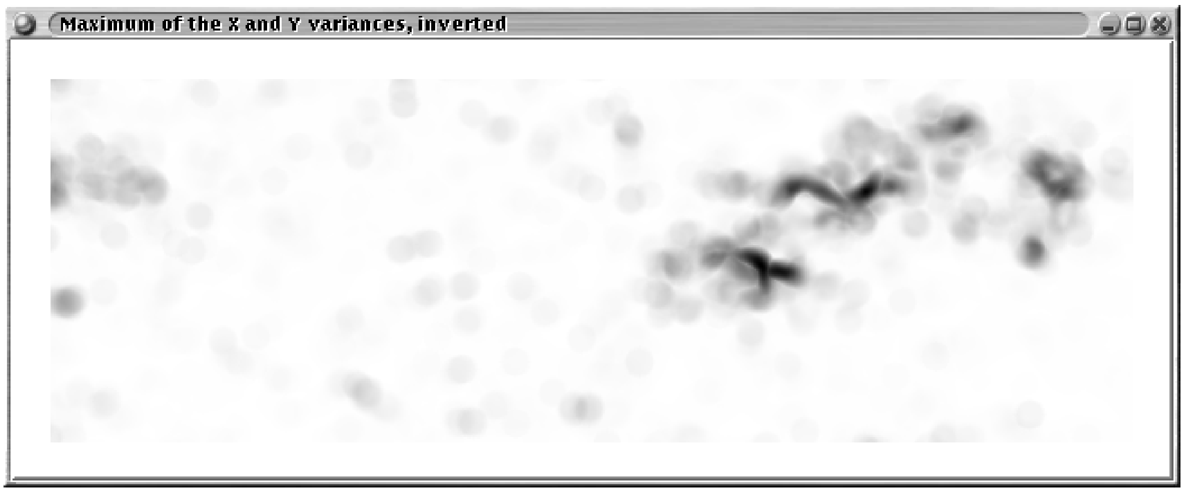
FIGURE 17.11 Effect of a bubble burst, measured using MAD based SEA.



(a) Variance of the X component of the motion field, inverted.

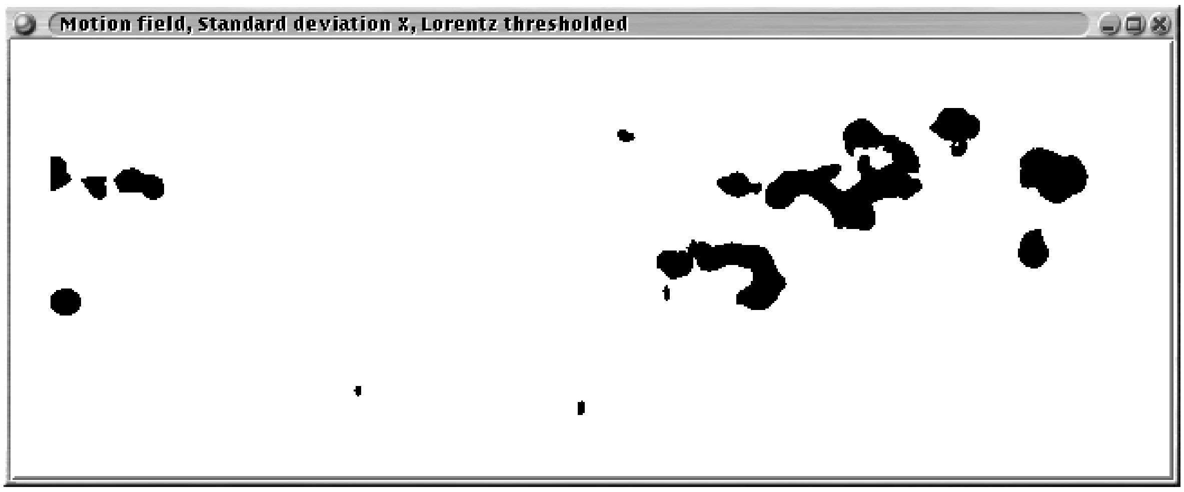


(b) Variance of the Y component of the motion field, inverted.

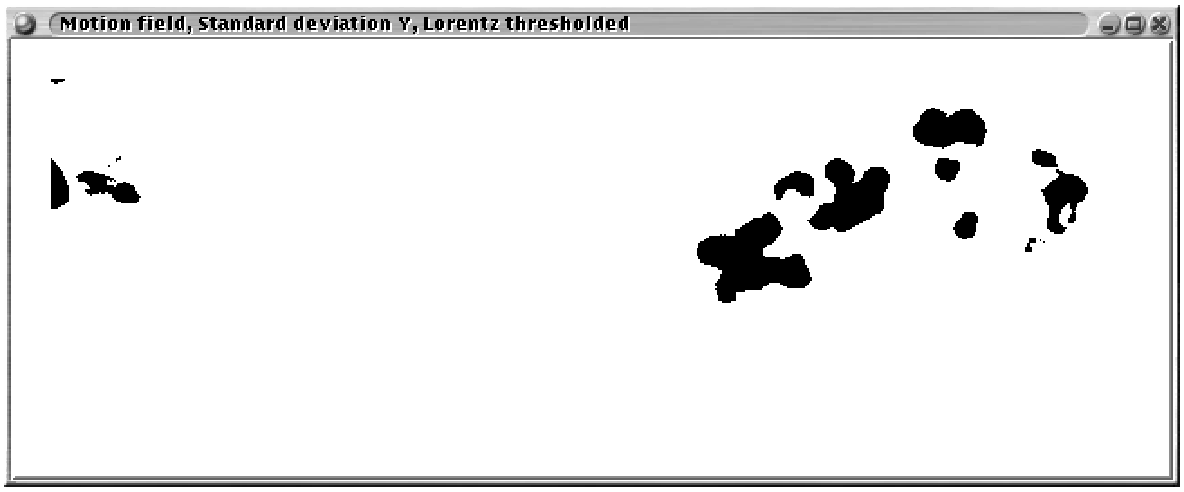


(c) Maximum of the variance of the X and Y components of the motion field, inverted.

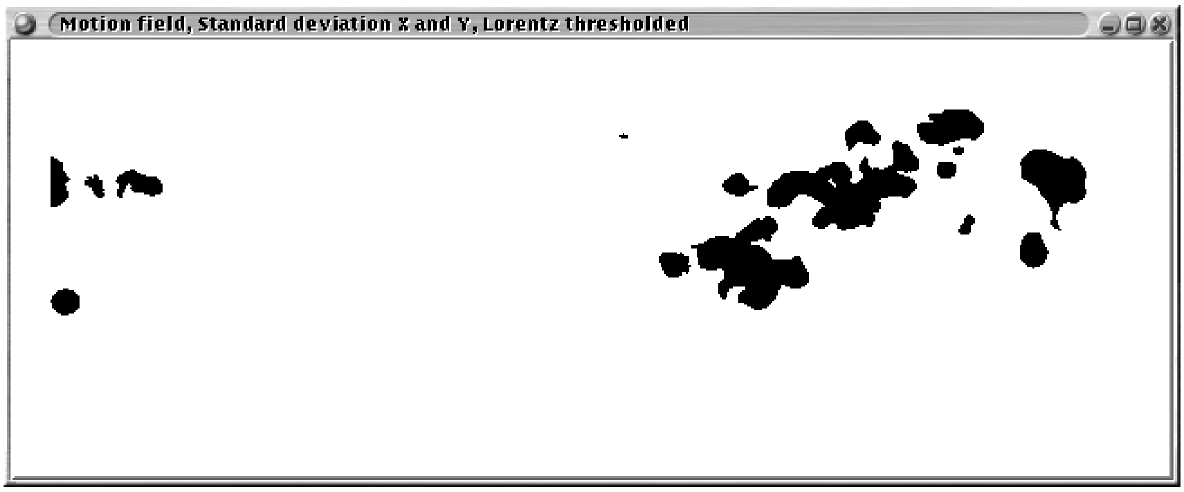
FIGURE 17.12 The variances of the X and Y components of the motion field, via MAD based SEA, burst event.



(a) Variance of the X component of the motion field, thresholded, inverted.

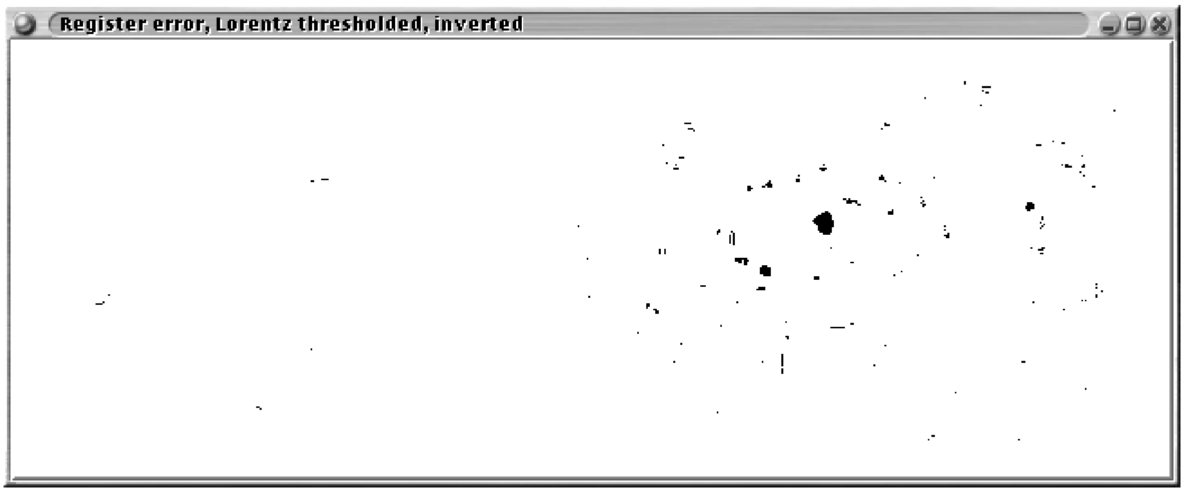


(b) Variance of the Y component of the motion field, thresholded, inverted.

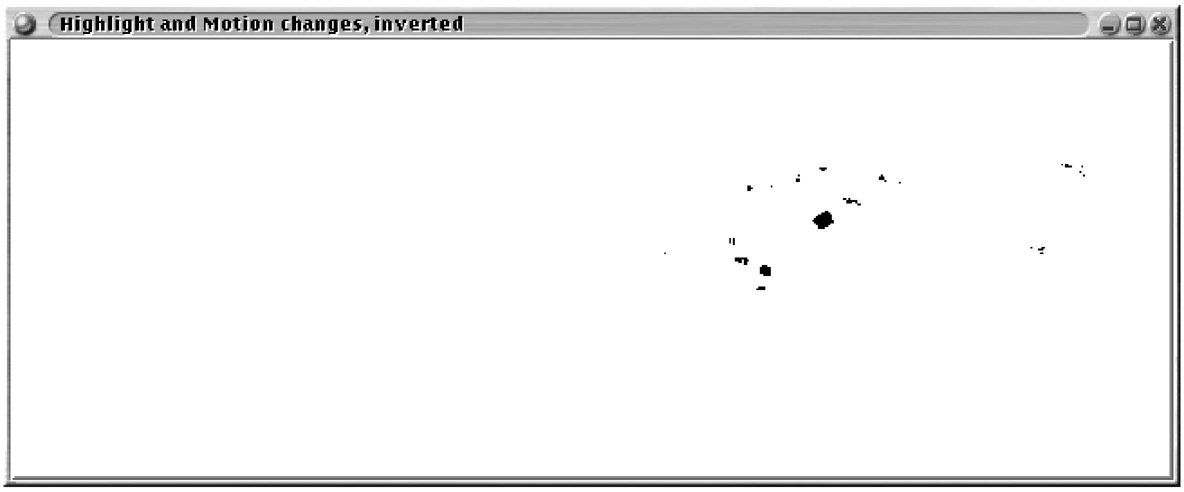

Motion field, Standard deviation X and Y, Lorentz thresholded

(c) Maximum of the variance of the X and Y components of the motion field, inverted.

FIGURE 17.13 The thresholded variances of the X and Y components of the motion field, via MAD based SEA, burst event.

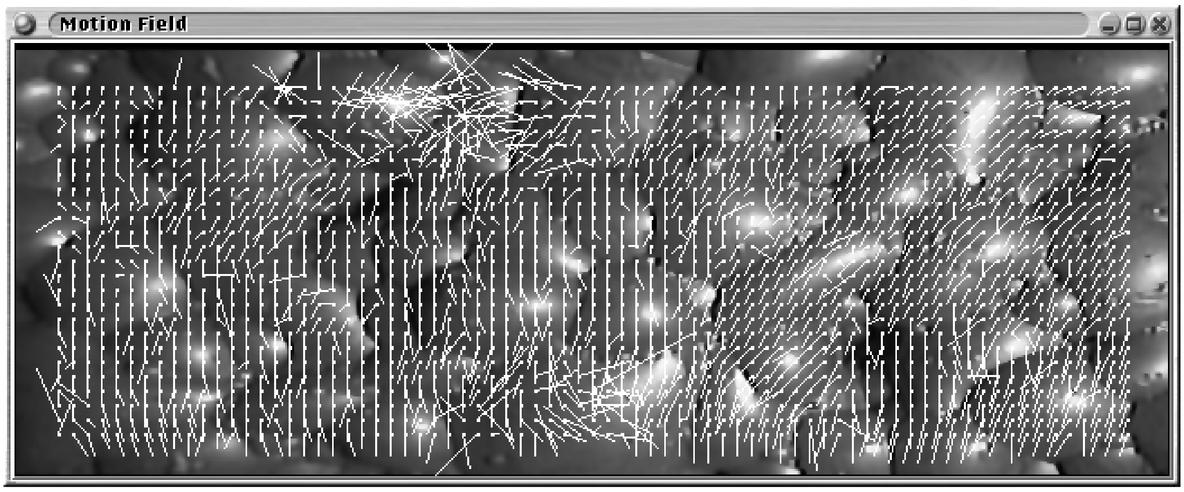


(a) Lorentz thresholded error image.

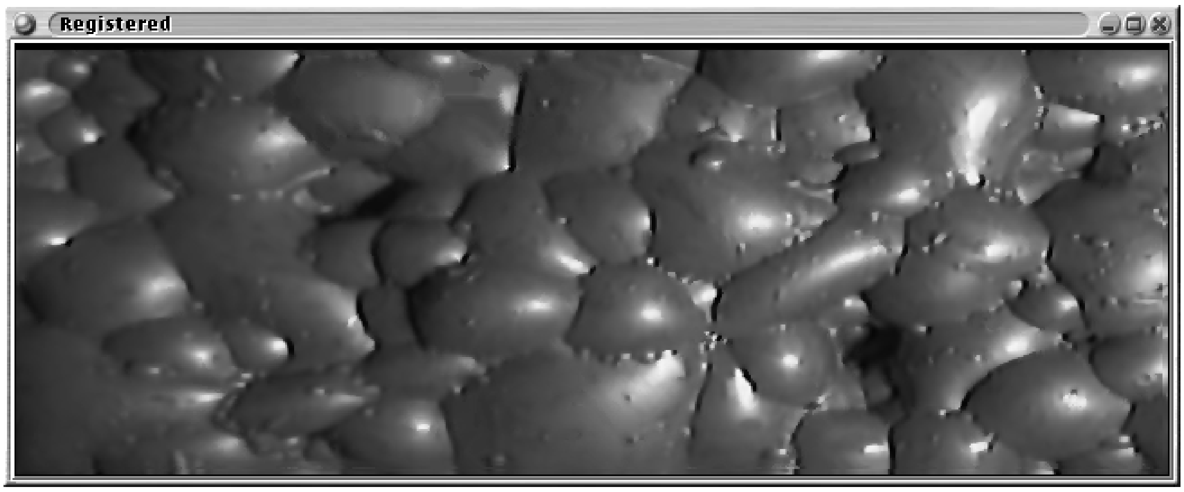


(b) Thresholded highlights coincident with motion discontinuities.

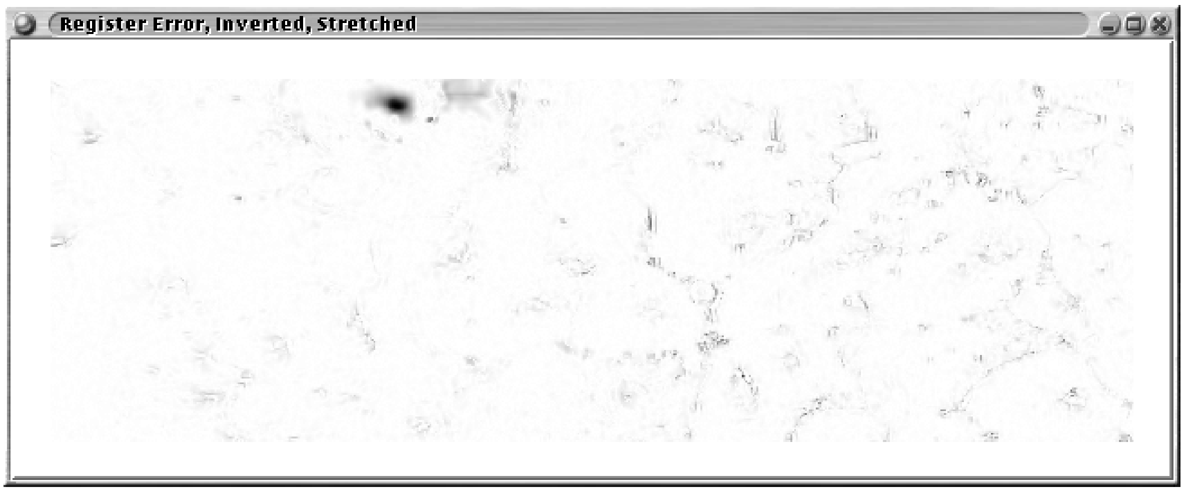
FIGURE 17.14 Thresholded highlights and coincident with motion discontinuities, via MAD based SEA.



(a) Motion field, scaled by factor of 4. Average motion $(-0.20,\ -2.0)$.

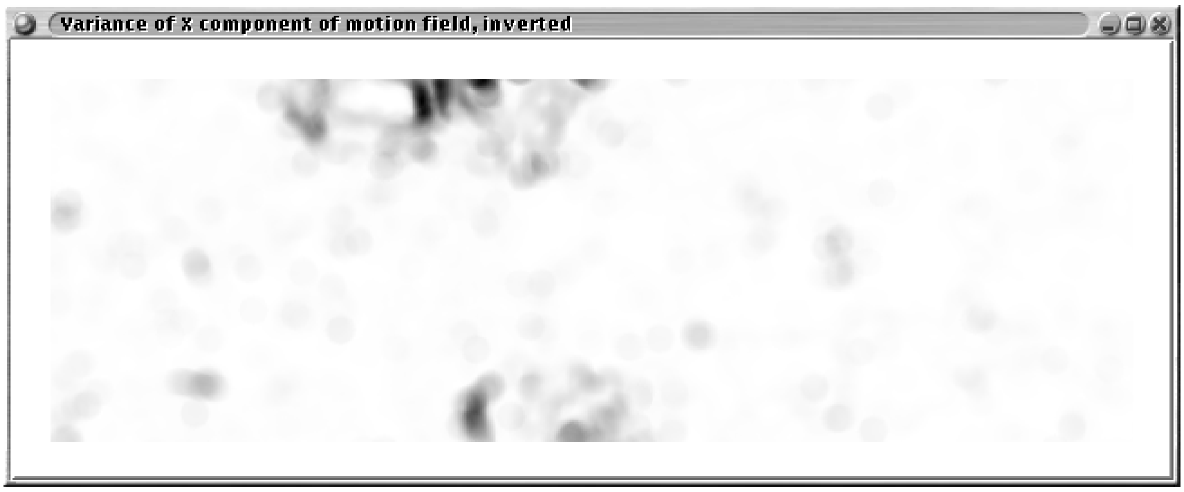


(b) Registered image.

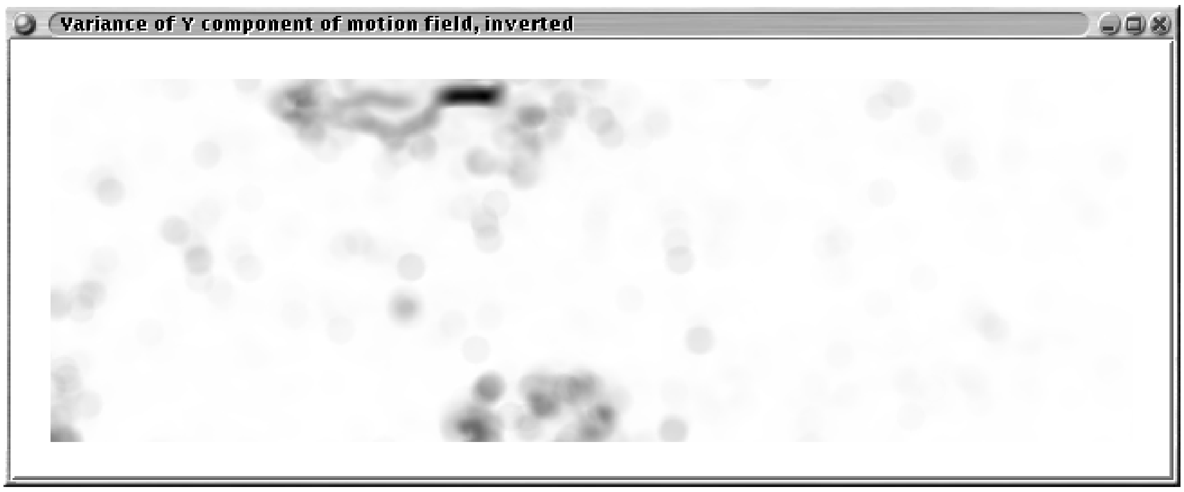


(c) Registered error, stretched. Energy in error image is 3 236 513.

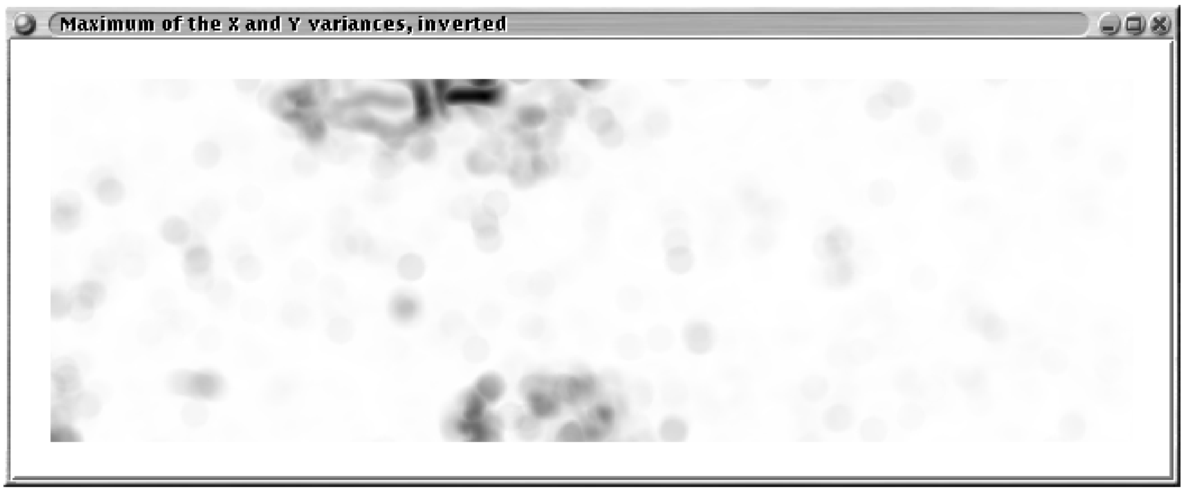
FIGURE 17.15 Effect of a bubble merge, measured using MAD based SEA.



(a) Variance of the X component of the motion field, inverted.

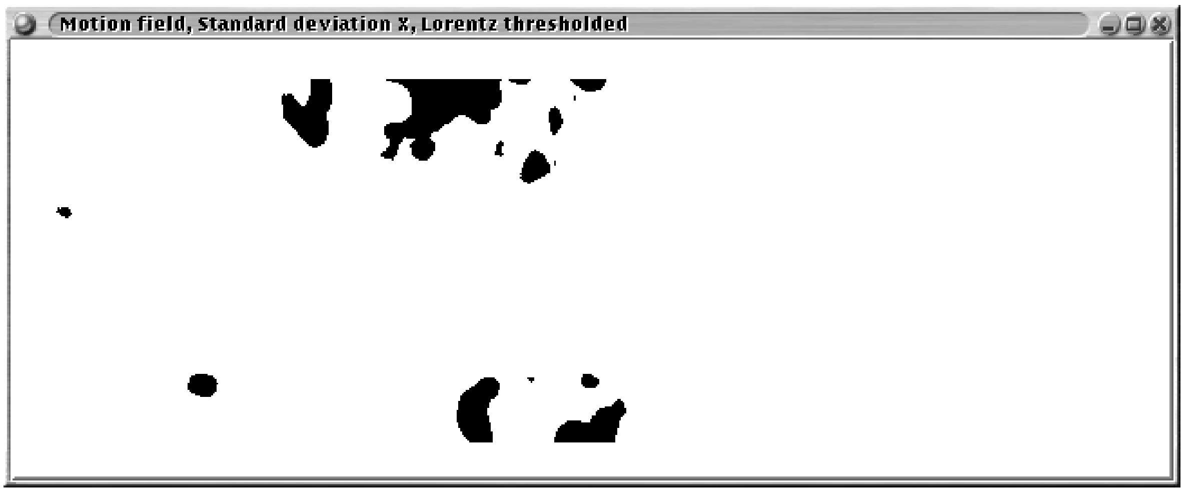


(b) Variance of the Y component of the motion field, inverted.

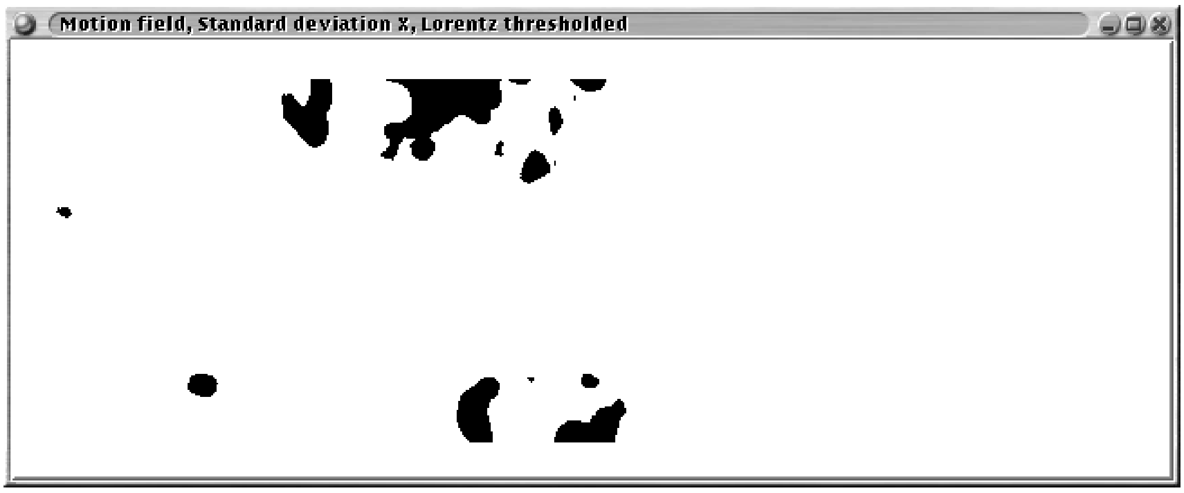


(c) Maximum of the variance of the X and Y components of the motion field, inverted.

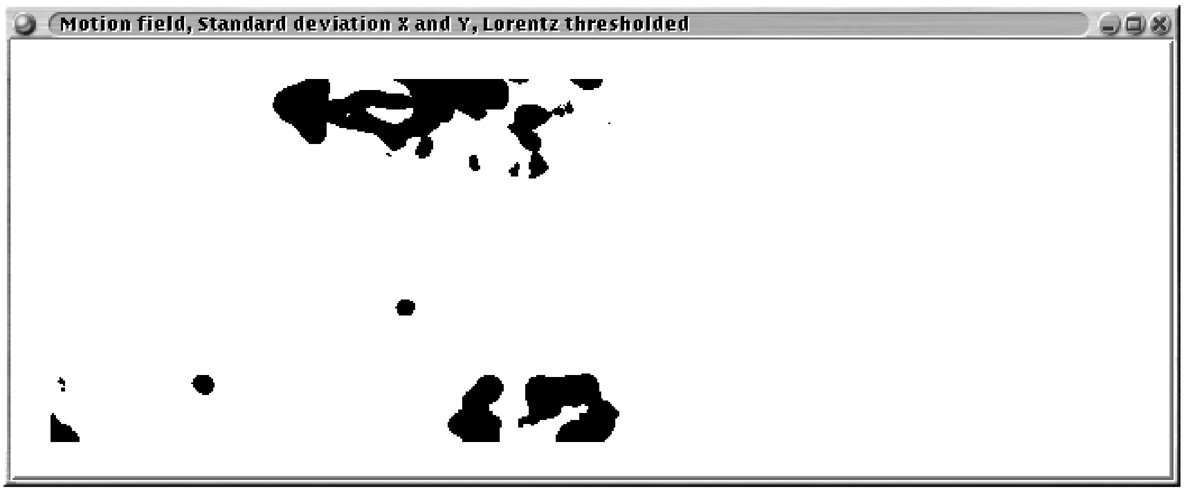
FIGURE 17.16 The variances of the X and Y components of the motion field, via MAD based SEA, merge event.



(a) Variance of the X component of the motion field, thresholded, inverted.

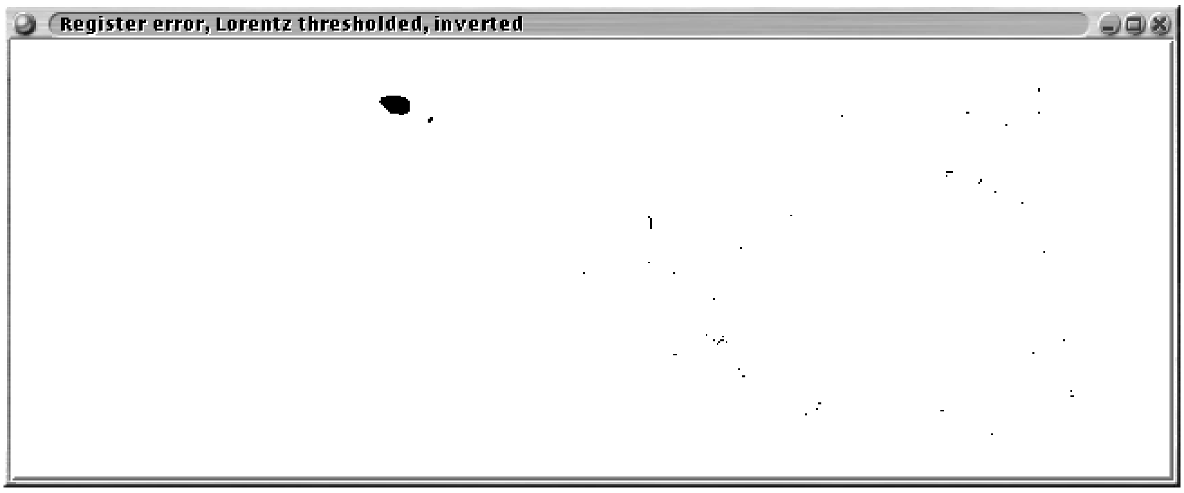


(b) Variance of the Y component of the motion field, thresholded, inverted.

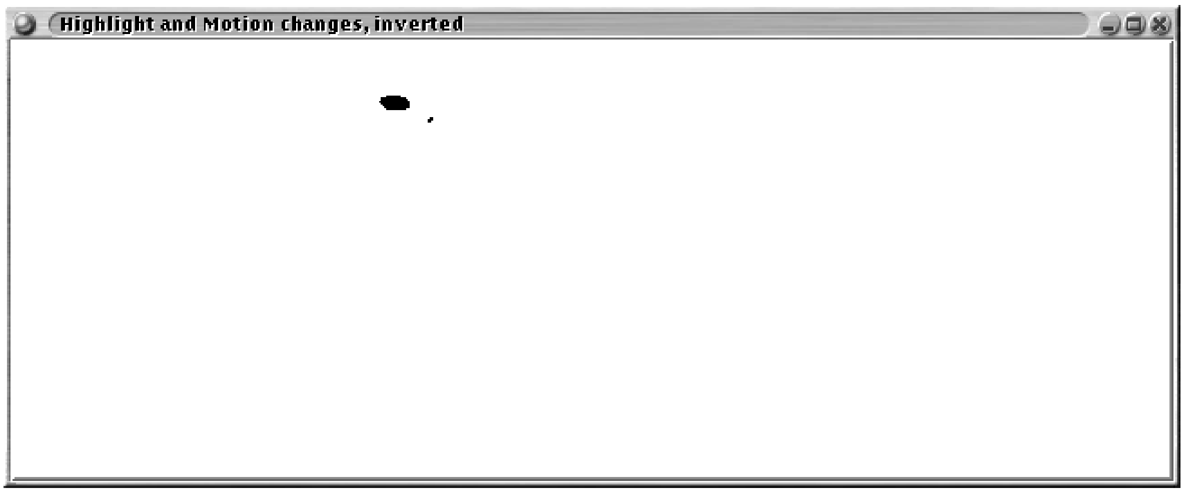

Motion field, Standard deviation X and Y, Lorentz thresholded

(c) Maximum of the variance of the X and Y components of the motion field, inverted.

FIGURE 17.17 The thresholded variances of the X and Y components of the motion field, via MAD based SEA, merge event.



(a) Lorentz thresholded error image.



(b) Thresholded highlights coincident with motion discontinuities.

FIGURE 17.18 Thresholded highlights and coincident with motion discontinuities, via MAD based SEA, merge event.

merges or highlight changes. These candidate images may then be further processed in an attempt to localise the event in question. Highlight changes tend to occur in conjunction with high variances in the motion field. This has the effect of over-estimating the number of burst or merge events.

Chapter 18

Comparisons of the various motion estimation algorithms

It is fairly difficult to compare the various motion estimation algorithms on a sequence of froth images without introducing some observer bias. After all, if the motion of the froth at each pixel were known already for calibrating the motion estimation algorithms, then there would be no need for testing algorithms since the problem would already have been solved. In the absence of known motion on a pixel by pixel basis, all that can be done is observing that the candidate algorithms find the trend of motion and comparing the algorithms on artificially warped froth images.

The alternative proposed here is to use the motion field and one of the images to estimate the other. The energy in the difference between the image at time t, and the interpolated image generated from the motion field and the image at time $t + \Delta t$, can be used to rank algorithms. It is possible to generate the image at time $t + \Delta t$, from the motion field and the image at time t but this would involve interpolation of the motion field as well as the image in question. A further issue is the selection of optimal parameters for the various algorithms to render them suitable for froth image sequences.

18.1 The test set used

The test set used consists of 224 frames ($\approx 9s$) taken taken on nine flotation banks as shown in table 18.1. The data set comprises 1 116 frames in all. The data is all collected from the Merensky flotation circuit an the AngloPlats Amandelbult Concentrator. The images used are the even fields of two consecutive frames. This is not the only way to overcome the interlacing problem, odd fields may be used, or the even and odd fields interpolated up to full frame size before being processed. De-interlacing removes motion artifacts introduced by interlacing. Recall that the even and odd fields in a frame are taken at different times (a $\frac{1}{50}$th of a second apart).

TABLE 18.1 The data set used for motion estimation.

Image collection	Bank collected from	Time	Date	Operational notes
Cleaner 1	Cleaner	17h04	June 1997	Normal
Rougher 1	Rougher	19h55	June 1997	Normal
Rougher 2	Rougher	20h07	June 1997	Normal
Rougher 3	Rougher	10h52	18 June 1998	Normal
Rougher 4	Rougher	19h03	17 June 1998	Normal
Rougher 5	Rougher	19h31	17 June 1998	Higher interface
Rougher 6	Rougher	19h40	17 June 1998	Low airflow
Rougher 7	Rougher	19h55	17 June 1998	Low depressant
Rougher 8	Rougher	20h07	17 June 1998	High frother

18.2 Selection of parameters for various motion estimation algorithms

The parameters used in the different motion estimation algorithms are selected by hand. This allows for good performance, since domain specific knowledge is used, coupled with reasonable running times. In principle, selection of optimal parameters is a combinatorial optimisation problem.

A cost function can be proposed, for an algorithm's parameters, as follows. Once the motion field has been generated, the motion field can be used to warp an image to produce an estimate of the current image in the sequence of images. This estimate can then be compared to the original and an error measure applied.

This allows for an optimisation algorithm to be used to compute, for a given training sequence of images, the parameters for a motion estimation algorithm which best captures the motion between successive images. There are a large number of optimisation algorithms described in the literature (see [115] for algorithms and a succinct overview of some popular types). The better known combinatorial optimisers include Population Based Incremental Learning (PBIL) [8] a variant of genetic algorithm. Simulated annealing [1] is another. Other deterministic optimisers include gradient descent type algorithms.

The parameters used for the different motion estimation algorithms are,

- Block and region matching algorithms,

  Pixel Tracing Block size 101 $\times$ 101 pixels and 30 pixels searched along each search direction.

  MAD based SEA Block size of 9 $\times$ 9 and a search area of 5 $\times$ 5.

  MSE based SEA Block size of 9 $\times$ 9 and a search area of 5 $\times$ 5.

  Kottke and Sun 400 regions used, and confidence (α) of 1. The clustering stops when the largest cluster shift is less than 1 pixels.

  Watershed based motion estimation The search radius is 40 pixels and the distance to best match weighted by 10.

- Gradient based optical flow,

  Horn and Schunk 16 iterations, with $\alpha = 1$.

  Div-Curl Optical Flow 16 iterations, $\alpha = 0.8$ and $\beta = 0.2$.

  Tistarelli A sigma of $\sigma = 2$ and $\tau = 1$.

- Hierarchical algorithms. The depth of the scale space pyramid used is 4, and the sigma of the low-pass Gaussian kernel used is $\sigma = 1$. The spatial extent of the Gaussian kernel is selected to be 4×4.

  MAD based SEA Block size of 5×5 and a search area of 3×3.

  MSE based SEA Block size of 5×5 and a search area of 3×3.

  Motion from Watershed The search radius is 40 pixels and the distance to best match weighted by 10.

  Horn and Schunk 16 iterations, with $\alpha = 1$.

18.3 Comparison of algorithms on the standard test set

A number of motion estimation algorithms are evaluated on the froth image sequences. The algorithms are compared by treating motion estimation as an image registration problem and computing the registration error. Traditionally, proposed motion estimation algorithms are compared on data where the motion is known beforehand (for, example, on artificial data) [68]. A second technique relies on an observer judging by eye whether a motion field is reasonable [84].

Given two successive images, the motion field is computed using some algorithm. The motion field shows the pixels moving from their positions in the first image (the reference image), to their new positions as shown in the successive image (the motion image). The comparison method used inverse warps the motion image using the motion field, in other words, an estimated reference image is generated from the motion image using the motion field. The warp used estimates the pixel intensities, interpolated using a small sigma Gaussian kernel ($\sigma = 0.5$ used), at the predicted position of the pixel. The sigma is chosen to interpolate without smoothing the warped image significantly. This warping procedure is significantly simpler than the alternative, which is projecting the pixels in the first image onto their new positions as predicted by the motion field, and then interpolating to find the intensities on the rectangular grid. The warped image is compared to the first image.

This technique is related to existing methods for comparing temporal compression in image sequence compression (see [139][pg. 440–442]), television frame rate conversion [10] and motion detection [154].

Szeliski proposes a similar metric for comparing motion estimation and stereo correspondence algorithms [137]. Szeliski, however, incorporates a second warping stage of less than one pixel at worst, in order to correct for possible mis-registrations in the first registration stage. It should be noted, however, that the registration algorithm used will bias the results of all the motion estimation algorithms compared. Since this bias is common to all the results, correction of mis-registrations may well be unnecessary.

If one can regard a motion estimation algorithm as a form of image registration, then the registration error can be used as a means of comparison. The registration error is the difference between the reference image and the motion image warped by the final motion flow field. Tables 18.3 and 18.4 show the MAD and MSE for different motion estimation algorithms. Figures 18.1 and18.2 show the MAD and MSE of the cumulative registration error.

Another criterion for evaluating algorithms is the processing time required for generating the motion flow field. This is especially important for potential real time applications. Table 18.2 includes the time taken to execute the algorithm on a FreeBSD dual processor Pentium II machine. The algorithms were all implemented in Java running on a Java 1.3.0 virtual machine.

The difficulty in comparing algorithm processing required is comparing a multiple correlation algorithm like MSE based SEA to a feature based region comparison algorithm (such as watershed based motion estimation). One cannot here look at the numbers of pixels processed, or even the putative number of operations (can one really claim that a floating point operation is similar in execution time to an integer operation?). Examining computational time, while not ideal, is (under the same process load conditions) simpler and perhaps more indicative of the complexity of an algorithm.

TABLE 18.2 Comparison of running times.

Algorithm	Running Time(s)
BME regularised	3254052
MAD based SEA	118920
MSE based SEA	38137
Pixel Tracing	932
Kottke and Sun	7408
Motion from Watershed	35533
Div Curl	50055
Horn and Schunk	9205
Tistarelli	243460
Hierarchical Framework 1 MAD based SEA	13540
Hierarchical Framework 1 MSE based SEA	38137
Hierarchical Framework 1 Motion from Watershed	55497
Hierarchical Framework 2 Horn and Schunk	66118
Hierarchical Framework 2 MAD based SEA	32221
Hierarchical Framework 2 MSE based SEA	51976
Hierarchical Framework 2 Motion from Watershed	81098

TABLE 18.3 Registered Error (as MAD), test set.

Algorithm	Data Set									
	Cleaner 1	Rougher 1	Rougher 2	Rougher 3	Rougher 4	Rougher 5	Rougher 6	Rougher 7	Rougher 8	All Data
Region matching										
BME regularised	12.24	10.13	11.17	10.86	10.32	11.52	9.228	12.73	10.38	10.93
Kottke and Sun	13.97	12.96	12.80	14.78	12.52	13.93	11.12	12.58	12.80	13.04
MAD based SEA	3.813	2.668	2.652	2.889	2.836	3.637	2.410	2.683	2.652	2.917
MSE based SEA	3.937	2.753	2.734	2.964	2.937	3.748	2.485	2.761	2.731	3.004
Motion from Watershed	6.519	5.587	5.745	7.186	5.964	6.489	6.008	5.603	5.745	6.094
Pixel Tracing	6.067	5.151	5.171	4.587	5.570	6.625	4.558	5.131	5.171	5.390
Optical flow										
Div Curl	5.176	3.958	4.059	4.363	3.893	5.544	3.205	3.870	4.059	4.236
Horn and Schunk	3.379	2.582	2.607	2.833	2.711	3.783	2.243	2.572	2.646	2.822
Tistarelli	4.622	3.453	3.574	4.093	3.481	5.005	2.940	3.385	3.574	3.792
Hierarchical Framework 1										
MAD based SEA	4.744	3.480	3.577	3.972	3.384	4.799	2.862	3.420	3.577	3.757
MSE based SEA	4.830	3.542	3.641	4.033	3.447	4.875	2.918	3.483	3.641	3.823
Motion from Watershed	8.619	7.481	7.414	8.735	7.751	8.337	7.486	7.465	7.414	7.856
Hierarchical Framework 2										
Horn and Schunk	4.214	3.343	3.248	3.329	3.415	4.264	2.845	3.310	3.248	3.468
MAD based SEA	4.087	3.202	3.135	3.411	3.298	4.270	2.729	3.175	3.135	3.383
MSE based SEA	4.134	3.270	3.204	3.460	3.368	4.345	2.792	3.242	3.204	3.446
Motion from Watershed	13.76	13.61	12.91	14.34	14.18	13.73	12.62	13.60	12.91	13.52

TABLE 18.4 Registered Error (as MSE), test set.

Algorithm	Data Set									
	Cleaner 1	Rougher 1	Rougher 2	Rougher 3	Rougher 4	Rougher 5	Rougher 6	Rougher 7	Rougher 8	All Data
Region matching										
BME regularised	616.8	469.4	547.0	607.8	444.2	549.9	435.2	553.1	495.8	524.2
Kottke and Sun	662.5	570.3	570.6	758.3	518.0	626.6	478.5	548.6	570.6	588.6
MAD based SEA	79.98	37.26	38.47	51.62	39.66	71.43	31.94	37.89	38.47	47.46
MSE based SEA	82.14	38.37	39.50	52.50	41.00	73.39	32.69	38.87	39.43	48.61
Motion from Watershed	201.2	170.5	181.9	266.6	175.9	214.7	189.1	170.5	181.9	194.7
Pixel Tracing	181.2	167.6	164.8	150.2	171.2	244.6	136.2	166.3	164.8	175.6
Optical flow										
Div Curl	171.7	110.3	117.4	153.8	102.1	184.7	85.02	106.8	117.4	127.7
Horn and Schunk	107.0	70.92	75.40	94.49	73.80	133.1	59.37	70.53	77.00	84.99
Tistarelli	124.0	73.42	79.24	122.4	71.94	136.5	60.70	71.86	79.24	91.03
Hierarchical Framework 1										
MAD based SEA	113.8	63.99	66.39	92.55	58.44	111.3	47.16	62.96	66.39	75.89
MSE based SEA	115.5	64.61	67.11	93.14	59.11	112.5	47.69	63.58	67.11	76.70
Motion from Watershed	295.0	248.9	249.1	338.7	248.2	297.4	252.6	247.4	249.1	269.6
Hierarchical Framework 2										
Horn and Schunk	130.6	83.07	81.75	98.74	80.85	129.5	68.35	82.29	81.75	92.99
MAD based SEA	87.76	52.06	51.00	67.59	51.90	89.65	40.68	51.84	51.00	60.39
MSE based SEA	87.81	52.93	51.96	67.72	52.79	90.99	41.46	52.60	51.96	61.14
Motion from Watershed	602.5	614.7	578.2	726.2	617.2	641.2	567.2	610.9	578.2	615.1

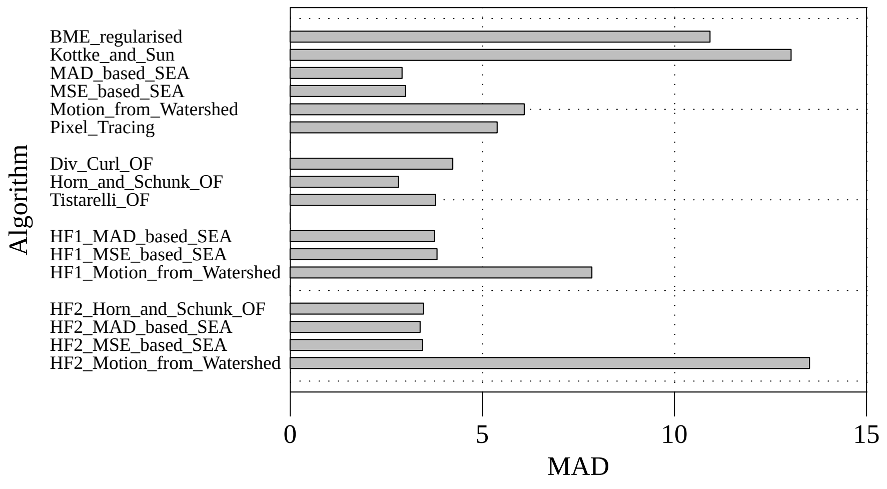


FIGURE 18.1 Registered Error (as MAD), test set.

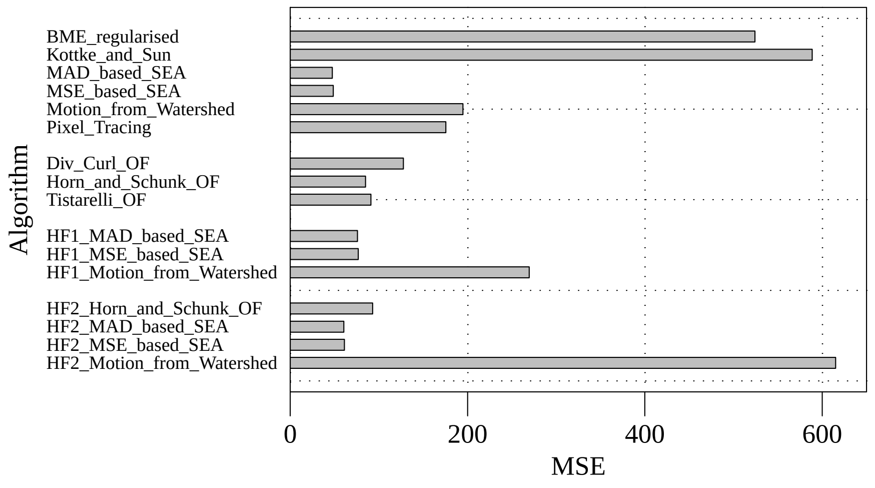


FIGURE 18.2 Registered Error (as MSE), test set.

18.4 Comparison of algorithms on artificially warped images

A second data set is generated by translating the pixel intensities of the first data set by $(5, 3)$. The choice is arbitrary, but needs to be away from the North-East and North-West diagonal of the image as well as from the axes. This enables errors in motion estimation to be localised as errors in the x or y direction as needed. Since the pixel by pixel motion field is known, the error between the computed and known motion fields can be examined. To facilitate the comparisons, only a subregion of the motion fields are compared, a rectangular region from $(20,\ 20)$ to $(620,\ 220)$, this is chosen to avoid possible image boundary effects. Table 18.5 and 18.6 summarises the error between the known and computed motion fields for the various algorithms tested.

A final criterion is the error in the difference image between the reference image, and the motion image as inverse warped by the motion field. This gives some indication of the registration capability of the algorithm. The better the algorithm the lower this error should be. Again, the performance of the algorithms with regard to this criterion is summarised in tables 18.7 and18.8 for MAD and MSE respective.

Figures 18.3, 18.4 show the error in recovering the motion field used to generate the artificial data from the froth image data set. Figures 18.5 and 18.6 compare the error in the difference image between the reference image and the reference image as warped by the recovered motion field.

TABLE 18.5 Reconstructed warp field (as MAD).

Algorithm	Data Set									
	Cleaner 1	Rougher 1	Rougher 2	Rougher 3	Rougher 4	Rougher 5	Rougher 6	Rougher 7	Rougher 8	All Data
Region matching										
BME regularised	2.015	2.055	2.065	2.081	2.051	2.062	2.045	2.045	2.066	2.054
Kottke and Sun	4.202	4.503	4.325	3.927	4.276	3.960	3.916	4.500	4.325	4.212
MAD based SEA	3.119	3.093	3.064	3.057	3.079	3.062	3.083	3.092	3.064	3.079
MSE based SEA	3.115	3.073	3.044	3.019	3.063	3.037	3.062	3.073	3.044	3.059
Motion from Watershed	3.901	3.920	3.921	3.942	3.928	3.912	3.918	3.920	3.921	3.920
Pixel Tracing	4.000	4.000	4.000	4.000	4.000	4.000	3.979	4.000	4.000	3.998
Optical flow										
Div Curl	12.00	20.99	20.18	17.00	20.67	17.79	23.42	20.85	20.18	19.22
Horn and Schunk	3.329	3.526	3.495	3.422	3.538	3.465	3.447	3.527	3.495	3.472
Tistarelli	3.210	3.297	3.264	3.206	3.304	3.233	3.150	3.296	3.264	3.247
Hierarchical Framework 1										
MAD based SEA	2.399	2.365	2.350	2.355	2.363	2.342	2.358	2.365	2.350	3.757
MSE based SEA	2.405	2.368	2.353	2.353	2.367	2.343	2.361	2.368	2.353	2.363
Motion from Watershed	4.206	4.220	4.224	4.269	4.226	4.214	4.237	4.220	4.224	7.856
Hierarchical Framework 2										
Horn and Schunk	3.188	3.306	3.279	3.243	3.301	3.288	3.240	3.308	3.279	3.270
MAD based SEA	2.823	2.829	2.807	2.804	2.820	2.809	2.805	2.829	2.807	2.815
MSE based SEA	2.829	2.833	2.811	2.804	2.824	2.813	2.810	2.833	2.811	2.819
Motion from Watershed	2.802	2.810	2.853	2.901	2.828	2.833	2.814	2.819	2.853	2.835

TABLE 18.6 Reconstructed warp field (as MSE).

Algorithm	Data Set									
	Cleaner 1	Rougher 1	Rougher 2	Rougher 3	Rougher 4	Rougher 5	Rougher 6	Rougher 7	Rougher 8	All Data
Region matching										
BME regularised	7.782	8.165	8.212	8.503	8.103	8.248	7.969	7.981	8.206	8.130
Kottke and Sun	228.7	321.5	323.7	270.9	301.9	245.8	264.6	324.5	323.7	288.5
MAD based SEA	15.84	15.74	15.57	15.55	15.65	15.56	15.65	15.73	15.57	15.65
MSE based SEA	15.81	15.61	15.44	15.31	15.55	15.40	15.52	15.61	15.44	15.52
Motion from Watershed	27.49	27.09	27.30	28.80	27.14	26.91	27.75	27.09	27.30	27.43
Pixel Tracing	24.67	24.67	24.67	24.67	24.67	24.67	24.44	24.67	24.67	24.64
Optical flow										
Div Curl	691.3	1552	1136	854.7	1186	960.4	1380	1567	1136	1161
Horn and Schunk	22.5	29.51	30.31	28.71	29.68	29.00	28.96	29.51	30.33	28.72
Tistarelli	54.82	88.30	84.22	75.02	86.25	77.85	63.31	88.25	84.22	78.03
Hierarchical Framework 1										
MAD based SEA	9.759	9.662	9.600	9.603	9.672	9.555	9.598	9.665	9.600	9.635
MSE based SEA	9.798	9.683	9.621	9.591	9.697	9.568	9.620	9.685	9.621	9.654
Motion from Watershed	34.20	33.56	33.73	35.71	33.53	33.34	34.50	33.59	33.73	33.99
Hierarchical Framework 2										
Horn and Schunk	17.98	20.19	20.11	19.73	20.21	20.20	19.36	20.22	20.11	19.79
MAD based SEA	13.31	13.52	13.39	13.32	13.48	13.42	13.29	13.52	13.39	13.41
MSE based SEA	13.35	13.55	13.42	13.32	13.51	13.45	13.33	13.55	13.42	13.43
Motion from Watershed	18.49	17.44	17.99	20.07	17.69	17.56	18.31	17.54	17.99	18.12

TABLE 18.7 Registered Error (as MAD), warped images.

Algorithm	Data Set									
	Cleaner 1	Rougher 1	Rougher 2	Rougher 3	Rougher 4	Rougher 5	Rougher 6	Rougher 7	Rougher 8	All Data
Region matching										
BME regularised	16.93	15.19	14.44	15.83	15.22	14.45	13.69	14.47	14.38	14.95
Kottke and Sun	15.25	15.50	14.56	14.56	15.32	14.67	13.25	15.48	14.56	14.79
MAD based SEA	6.499	7.470	7.092	7.012	7.391	7.363	6.426	7.464	7.092	7.090
MSE based SEA	6.645	7.685	7.320	7.264	7.627	7.607	6.631	7.680	7.324	7.310
Motion from Watershed	4.093	4.194	4.302	5.661	4.141	4.108	4.652	4.196	4.302	4.405
Pixel Tracing	6.607	6.418	6.241	6.040	6.517	6.296	5.907	6.395	6.241	6.285
Optical flow										
Div Curl	25.60	32.57	32.78	33.75	33.74	29.97	34.70	32.48	32.78	32.02
Horn and Schunk	6.711	8.294	7.636	7.861	8.060	8.125	7.024	8.284	7.654	7.737
Tistarelli	8.428	9.909	9.192	9.443	9.630	9.672	8.379	9.895	9.192	9.304
Hierarchical Framework 1										
MAD based SEA	9.208	9.934	9.329	9.525	9.860	9.640	8.511	9.927	9.329	9.473
MSE based SEA	9.250	9.870	9.231	9.459	9.782	9.578	8.406	9.862	9.231	9.407
Motion from Watershed	7.760	7.789	7.569	8.514	7.792	7.443	7.505	7.784	7.569	7.747
Hierarchical Framework 2										
Horn and Schunk	6.312	6.898	6.341	6.570	6.687	6.680	5.873	6.876	6.341	6.508
MAD based SEA	7.594	8.255	7.774	7.892	8.167	8.020	7.154	8.245	7.774	7.875
MSE based SEA	7.575	8.152	7.635	7.808	8.044	7.936	7.028	8.144	7.635	7.773
Motion from Watershed	14.95	14.62	13.77	16.38	14.61	13.81	13.80	14.56	13.77	14.48

TABLE 18.8 Registered Error (as MSE), warped images.

Algorithm	Data Set									
	Cleaner 1	Rougher 1	Rougher 2	Rougher 3	Rougher 4	Rougher 5	Rougher 6	Rougher 7	Rougher 8	All Data
Region matching										
BME regularised	876.0	713.0	687.6	864.4	678.8	702.2	645.7	650.7	682.1	722.6
Kottke and Sun	658.9	714.1	652.1	682.3	656.0	676.6	577.1	710.7	652.1	664.3
MAD based SEA	198.2	275.2	266.2	292.2	261.2	284.2	237.7	273.6	266.2	261.6
MSE based SEA	204.8	287.0	278.6	307.2	274.0	297.9	248.8	285.4	279.0	273.7
Motion from Watershed	126.9	150.4	160.4	220.7	149.5	154.3	170.1	150.2	160.4	160.3
Pixel Tracing	181.0	197.8	197.7	207.1	195.6	202.8	189.6	196.0	197.7	196.1
Optical flow										
Div Curl	1574	2137	2174	2399	2189	1950	2330	2126	2174	2115
Horn and Schunk	252.5	372.0	339.1	399.0	345.7	381.0	310.0	370.0	340.5	345.4
Tistarelli	287.8	390.9	353.2	424.8	354.6	394.8	315.8	388.1	353.2	362.6
Hierarchical Framework 1										
MAD based SEA	307.8	382.1	355.1	392.4	358.0	385.3	315.6	379.9	355.1	359.0
MSE based SEA	300.6	363.0	332.3	373.1	337.1	366.9	293.9	361.1	332.3	340.0
Motion from Watershed	250.0	270.2	267.3	331.4	263.2	264.1	266.6	268.8	267.3	272.1
Hierarchical Framework 2										
Horn and Schunk	217.8	265.2	241.6	284.0	243.4	268.2	219.5	262.6	241.6	249.3
MAD based SEA	87.76	52.06	51.00	67.59	51.90	89.65	40.68	51.84	51.00	278.4
MSE based SEA	87.81	52.93	51.96	67.72	52.79	90.99	41.46	52.60	51.96	257.0
Motion from Watershed	663.8	683.3	635.9	852.2	652.8	655.9	640.2	676.2	635.9	677.3

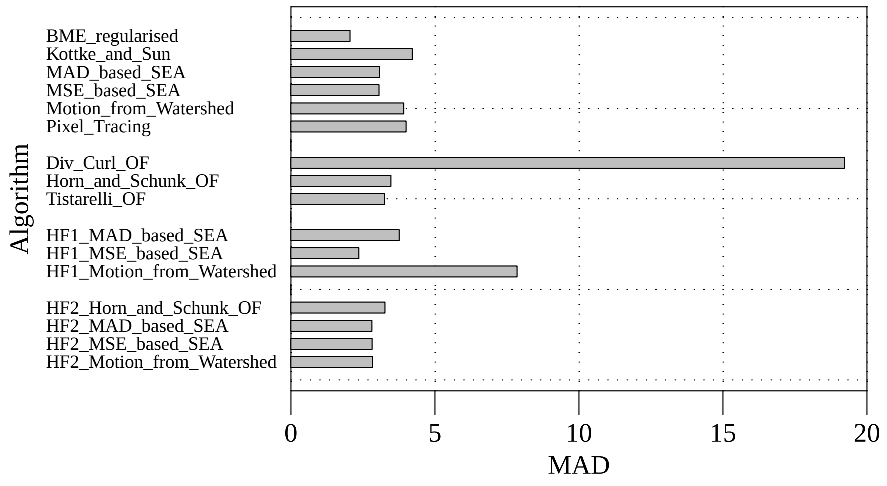


FIGURE 18.3 Error in reconstructed warp field (as MAD), artificial data.

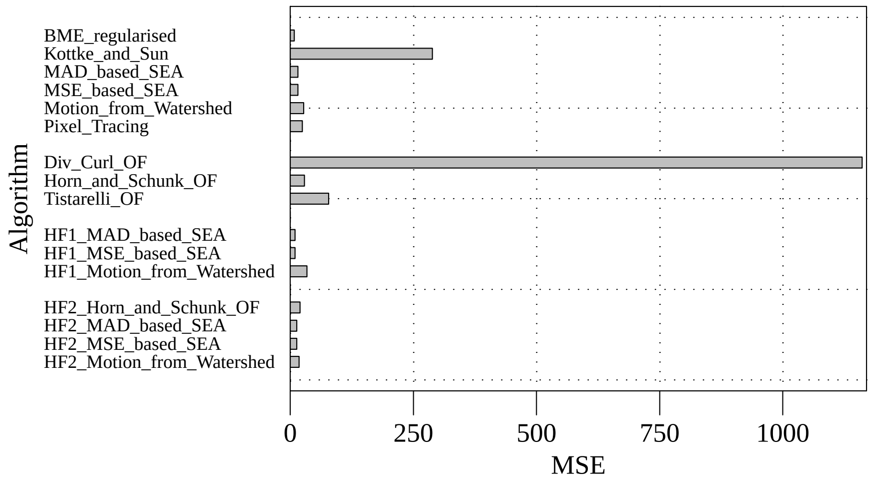


FIGURE 18.4 Error in reconstructed warp field (as MSE), artificial data.

18.5 Discussion

The rankings of algorithms in terms of smallest error is different for the error norms used. This can be attributed to the relative weighting of large errors. MAD weights each error linearly, whereas, MSE weights each error quadratically. Table 18.9 shows the different rankings of the various motion estimation algorithms in terms of increasing error.

Clearly the BME estimation algorithms, for the MSE measure, minimises the occurrence of large errors. This is a likely consequence of the robustness of correlation based methods. The warped data shows BME estimating large motion well.

Framework 1 is shown to better estimate the warping field than Framework 2. This implies that Framework 1 is a better estimator of the actual motion field. The implication is that the method used

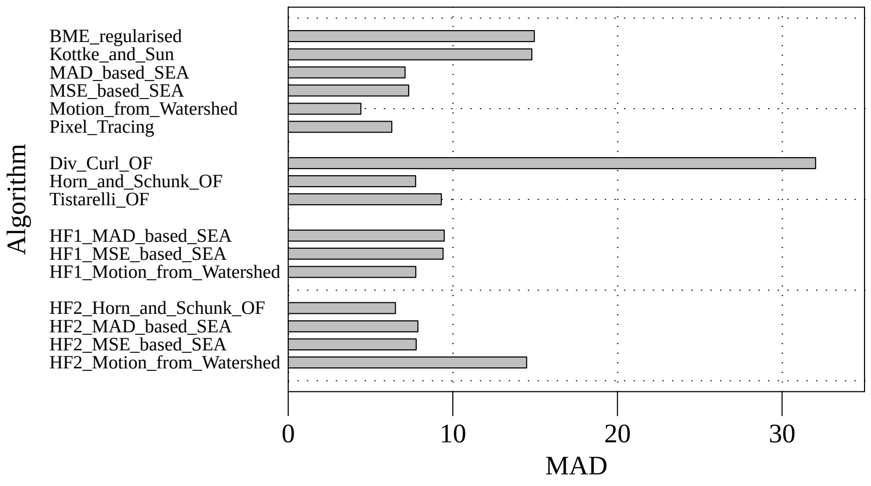


FIGURE 18.5 Registered Error (as MAD), artificial data.

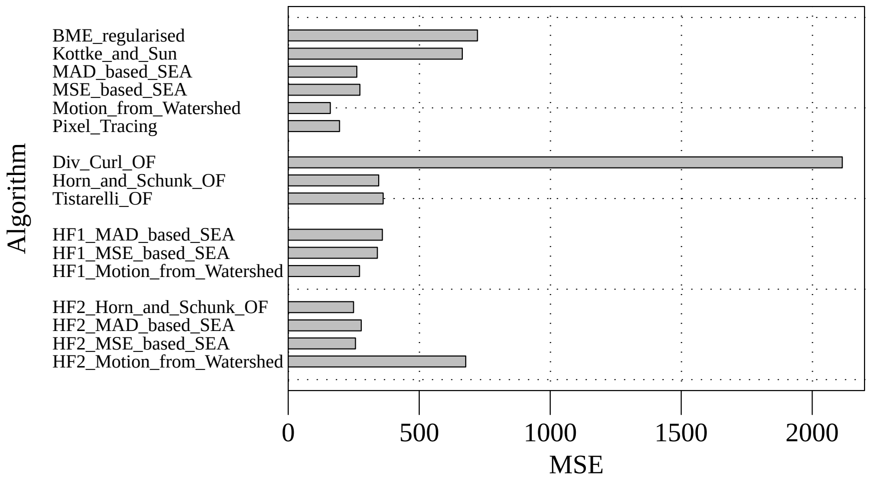


FIGURE 18.6 Registered Error (as MSE), artificial data.

TABLE 18.9 Ranking motion estimation algorithms.

Standard test set		Warped images			
		Warp field		Registered error	
MAD	MSE	MAD	MSE	MAD	MSE
Horn Schunk	MAD SEA	HF1 MSE SEA	HF1 MAD SEA	Watershed	Watershed
MAD SEA	MSE SEA	HF2 MAD SEA	HF1 MSE SEA	Pixel Tracing	Pixel Tracing
MSE SEA	HF2 MAD SEA	HF2 MSE SEA	HF2 MAD SEA	HF2 Horn Schunk	HF2 Horn Schunk
HF2 MAD SEA	HF2 MSE SEA	HF2 Watershed	HF2 MSE SEA	MAD SEA	HF2 MSE SEA
HF2 MSE SEA	HF1 MAD SEA	MSE SEA	MAD SEA	MSE SEA	MAD SEA
HF2 Horn Schunk	HF1 MSE SEA	MAD SEA	MSE SEA	Horn Schunk	HF1 Watershed
HF1 MAD SEA	Horn Schunk	Tistarelli	HF2 Watershed	HF1 Watershed	MSE SEA
Tistarelli	Tistarelli	HF2 Horn Schunk	HF2 Horn Schunk	HF2 MSE SEA	HF2 MAD SEA
HF1 MSE SEA	HF2 Horn Schunk	Horn Schunk	Pixel Tracing	HF2 MAD SEA	HF1 MSE SEA
Div Curl	Div Curl	HF1 MAD SEA	Watershed	Tistarelli	Horn Schunk
Pixel Tracing	Pixel Tracing	HF1 Watershed	Horn Schunk	HF1 MSE SEA	HF1 MAD SEA
Watershed	Watershed	Watershed	HF1 Watershed	HF1 MAD SEA	Tistarelli
HF1 Watershed	HF1 Watershed	Pixel Tracing	Tistarelli	HF2 Watershed	Kottke Sun
Kottke Sun	Kottke Sun	Kottke Sun	Kottke Sun	Kottke Sun	Watershed
HF2 Watershed	HF2 Watershed	Div Curl	Div Curl	Div Curl	Div Curl

for comparing motion fields, namely the error between the reference image and the estimated reference image (created from the motion field and the motion image) should be used with care. However, when baseline motion data is unavailable, the method of comparison (a modified form of that due to Szeliski [137]) may be the only way to quantatively compare motion estimation algorithms.

Note that Tistarelli's algorithm always performs nearly as well, or worse than Horn and Schunk's algorithm. This indicates the utility of diffusing neighbouring motion estimates into regions were Tistarelli's algorithm cannot estimate the motion.

Motion in froth image sequences is a simpler problem than the general motion estimation problem. The general motion estimation problem must be capable of accounting for the motion of different objects in an image. The objects may well have very different motion, and the motion estimation algorithm must respect the boundaries of the objects, and not assign incorrect motion of one object to another or to the background. This, for example, leads to the complex regularisations used in gradient based optical flow in the literature [2, 11, 103]. Froth image sequences, however, exhibit different motion, the motion of a pixel is highly correlated to its neighbours. This is unsurprising, each pixel on a froth surface is physically coupled to its neighbours. Thus, examining complex regularisations becomes unnecessary.

It is clear that optical flow algorithms do not estimate large motion well. This is a result of the gradient estimation operating on a small local neighbourhood only. Embedding Horn and Schunk's algorithm in a hierarchical framework is seen to perform well, however.

While the motion fields generated by Watershed based motion estimation and Kottke and Sun's motion from cluster matching, show discontinuities, and qualitatively, does not estimate the motion well, the artificial data set does show it estimates large average motion well.

Div Curl optical flow is not suited to estimating uniform motion. The regularisation, while capable of being reduced to Horn and Schunk's, assumes a complex motion field.

There exists a performance vs. processing time tradeoff. An application may settle for a lower performance algorithm provided this algorithm was fast, compared to another algorithm with better performance but is slower. An extreme example of this is Horn and Schunk's algorithm (section 15.1) vs. MSE based SEA (section 14.4). Block motion estimation algorithms provide good performance for large block sizes and area of the motion image searched. The computational time, however, is slow. Contrasting are the hierarchical methods, small block sizes and regions searched on the motion image perform well in a hierarchical framework. The results show Framework 2 performs better than framework 1 for the standard test set.

Real time applications demand fast algorithms. This suggests, for relatively small motion, using Horn and Schunk's algorithm followed by hierarchical MAD or MSE based SEA within Framework 1. While Framework 2 does perform better, it is considerably slower because of the extra interpolation stage, and does not estimate the actual motion field as well for uniform motion. If a fast uniform motion froth, close to a cell's launder, for example, is of interest, then a Framework 1 is preferable to Horn and Schunk's algorithm.

Part IV

Final conclusions and recommendations

Chapter 19

Conclusions and Recommendations

The issues of denoising, segmentation and motion analysis are examined. Finally, future work is proposed.

19.1 Denoising

Some form of denoising or smoothing of the froth images is essential. The over-segmentation is not simply an additive noise problem (although this does play a role) but is complicated by miniscule bubbles on top of large bubbles, fragmentation of the highlights and what appears to be *reflections* of small bubble highlights off of the large bubbles. The nature of the denoising, unfortunately, is that some decision will have to be made regarding a tradeoff between correctly segmenting the large bubbles and removing the small bubbles. Correctly preserving bubble boundaries wherever possible demands the use of a non-linear denoising algorithm. Denoising has the added advantage of improving the robustness of the motion estimation process.

Of the denoising algorithms examined, mean filtering, using a Gaussian kernel, performed well, but does not preserve small bubbles. The added requirement of preserving small bubbles is satisfied by the family of anisotropic diffusion algorithms. These provided acceptable performance without being too computationally demanding. Perona and Malik's algorithm (at least for a small number of iterations) performed acceptably. Tukey's bi-weight function, used as the $g(\cdot)$ function, preserves edges well and is recommended. It should be noted that few iterations are required for adequate denoising. Catte et al's algorithm did not denoise as much, but preserved detail better, implying more iterations to achieve the same denoising as Perona and Malik's algorithm. Catte et al's algorithm does, however, avoid certain difficulties with Perona and Malik's formulation, for example, noisy edges.

Median filters are shown to combine good preservation of edges, noise rejection, and speed. The simple median filter provides the best denoising performance among the family of median filters examined. The better detail preserving filters denoising performance is significantly worse, implying noise is being preserved.

Wavelet based denoising is fast, and preserves edges well. Wavelet denoising will not remove artifacts, such as small bubbles on large bubbles, but does remove random noise well. Use of wavelet denoising will require marker processing to reduce the effect of artifacts on the segmentation.

The remaining algorithms lack either the detail preservation requirements, or are too slow. Rolling ball openings, for example, preserve detail but, especially for larger radii, are too slow for real-time use.

The artificial images generated are visually very similar to natural froth images. This suggests that the piecewise spherical or elliptical model for a froth surface is potentially useful, for example, for shape from shading.

The speed requirements for a real (or near real) time algorithm ranks the algorithms discussed in the following order: simple median, Wavelet denoising, Perona Malik and, finally, Catte et al's algorithm.

19.2 Segmentation

The use of the watershed algorithm for segmentation raises the issue of choosing markers (seed regions) for the watershed regions. The issue of correct choice of the markers is at least as difficult as segmenting the image correctly, since correct choice of the markers will result in perfect, or near perfect, segmentation. The issues of extracting and then processing the markers were examined. The literature only speaks of using subtraction and then greyscale reconstruction to create the domes. Investigation showed that any anti-extensive filter followed by greyscale reconstruction is capable of providing acceptable domes. Erosion of the inverted image (using a rolling ball) followed by greyscale reconstruction was finally selected as the most appropriate technique. The radius of the rolling ball selects the minimum scale of the bubble highlights actually maintained. Since highlights are used for locating the markers, it is reasonable to reject any potential marker generated by the previous step whose average intensity (on the original image) is too dark. Rejecting markers whose average intensity is less that 50% of the maximum intensity improves the segmentation without significantly merging bubbles. Dilating (or closing) the markers serves to join close markers. This is not unreasonable since it is observed that, except for small bubbles clustered together, highlights are not very close to each other. Merging close markers also serves to merge fragmented markers.

The process of filtering the markers (essentially via a connected components approach) makes the segmentation more robust to a denoising algorithm not performing well. A real, or near real time, system demands speedy algorithms. This does suggest using wavelet based denoising followed by marker processing. It should be remembered that wavelet denoising runs in linear time, and the running time of a multi-resolution analysis is proportional to $Nlog(N)$ (N is the number of pixels in an image).

19.3 Motion estimation

Motion estimation of froth images remains a difficult problem. The first difficulty is that froth structures are not rigid by any means. The motion estimation problem is severely limited by the lack of any means of determining the actual motion of the froth. This lack makes it difficult to quantitatively compare the motion fields generated.

Clearly there is a relationship between running time of an algorithm and the performance of the algorithm. The block matching algorithms are slow but perform well (excepting hierarchical algorithms). Provided the motion is small, the optical flow algorithms perform well.

The performance of the motion estimation algorithms is compared by, essentially, an image registration approach. Once a motion field is generated, however this may be, an image at time $t + \Delta t$ may be warped onto its predecessor at time t. The difference between the predicted and the actual pixel intensities is clearly related to how well the motion estimation algorithm has performed. The literature usually test motion estimation algorithms on artificial data (where the motion is known), or show images and motion fields and appeal to reader to judge what is reasonable (the literature is replete with such examples, see [72, 84] for example).

The comparison technique has the advantage of a numerical means of comparison (the MSE of the error image for example). This allows for a motion estimation algorithm (or algorithms) to be embedded in an optimisation framework and the algorithm's parameters estimated. This, once an algorithm has been selected, turns the embedded motion estimation process into a "black-box" which is, essentially, self-calibrating.

It is shown that discontinuities in the motion field correspond to bubble bursts, bubble merges and highlights changing position or shape. It is shown that detection of these events is feasible.

It is shown that when robustness, speed, and motion estimation ability, especially for large uniform displacement is considered, then MAD or MSE based SEA embedded in Framework 1 is preferred. For small motion, Horn and Schunk's algorithm, followed by MAD or MSE based SEA embedded in Framework 2 is suggested.

19.4 How the thesis goals have been addressed

The following has been addressed within this work,

(i) For segmentation,

  (a) Careful marker processing improves the noise immunity of marker based watersheds.

  (b) Watershed on the distance function of an edge map, provides a simple technique for closing gaps in an edge detected froth image.

  (c) Anti-extensive filters, after dome extraction, produce better markers than simple subtraction alone.

(ii) For denoising,

  (a) Edge preserving denoising is shown to produce better watershed segmentations than linear filtering.
  (b) The artificial froth image model is shown to be useful for quantitatively comparing denoising algorithms for this application.
  (c) Fast denoising algorithms which do not affect the segmentation are shown to exist.
  (d) Wavelet based Occam filters denoise well and fast. The principles of Occam flters are based on image coding, which implies wavelet based compression for froth images. JPEG based denoising is shown to introduce artifacts without appreciable denoising. This indicates that froth image sequences intended for processing should not be compressed using a JPEG based coder (for example MPEG and MPEG2).

(iii) For motion analysis,

  (a) The influence of increasing block size and correlation between neighbouring motion vectors is demonstrated.
  (b) A motion estimation algorithm, Mean Square Error Successive Elimination Algorithm is proposed.
  (c) It is shown that bubble bursts and merges may be detected by finding discontinuities in dense motion fields.
  (d) An algorithm suitable for fast and robust dense motion field estimation is shown to exist.

19.5 Future work

Given the number of research groups active in this area now, and work at the University of Cape Town, further examination of classical watershed based techniques on froth images now seems of limited use.

The problems encountered in the use of watershed include “comet-tailing”, this could be reduced if a shape constraint were imposed on growth of the watershed regions. This issue of constrained contour evolution has been extensively addressed in the literature on snakes (see [116] and references therein). In the context of froth imaging this corresponds to model based segmentation.

Colour analysis of froth images needs to be examined. Platinum is somewhat unusual since the froth is grey. In copper and tin, for example, the colour of the froth is a useful indicator of process performance [59, 61, 97]. The issue of conflicts between illumination for the machine vision system and the general plant illumination needs to be examined. The requirement that there be a single highlight per bubble will not hold if other strong light sources are present. If the extraneous light sources have a different colour, for example, general plant lighting has a strong red component, as opposed to the machine vision system lighting being essentially white, then colour analysis could be used to find, and remove, extraneous highlights.

Shape from $\mathcal{X}$ ($\mathcal{X}$ could be shading, texture, motion, etc.) could be used to generate a height map of a froth image. Watershed on this height map would then yield a better estimate of the bubble boundaries. This is because the grey level intensity is dependent in a nonlinear way on shape.

Motion estimation remains an area of further work, especially since bubble merge and burst events need robust and reliable detection. It has been shown that discontinuities in the motion field correspond to bubble bursts, bubble merges, and highlight changes. Classifying these events remains an area of further work.

A metric used to compare motion estimation algorithms involves subtracting an estimated reference image from the reference image. The average error using MAD or MSE is an indication of how well a motion estimation algorithm performs. The estimated image is generated by inverse warping the motion image using the motion field. It is possible to use an optimisation algorithm to find the parameters of a motion estimation algorithm which optimises the performance of the algorithm (this is in fact suggested by Szeliski [137]). Finding a fast and robust optimiser would turn the motion estimation process into a self-calibrating component.

Use of wavelet based Occam filters for denoising requires a wavelet matched to the characteristics of froth images. This problem may be framed as finding a wavelet which best compresses froth images (or classes of froth images, perhaps). Is froth appearance under different operating conditions (froth types) different enough to require different wavelets for optimal compression? Would this allow for classification of froth images based on appearance?

The alternative definition of diffusion (equation 7.4) needs to be compared against Perona and Malik's model.

Part V

Appendices

Appendix A

Wavelets – a brief overview

This chapter presents a brief overview of wavelets. A number of good introductions to wavelets exist in the literature. A difficulty is that most seem to written from a mathematician's perspective. A good introduction from the perspective of filter banks is Vetterli and Kovacevic [139]. Goswami and Chan [51] provide a good introduction with some example applications.

A.1 A brief description of wavelets

Wavelet analysis covers two areas: multi-resolution analysis and the wavelet transform. For simplicity the wavelet transform is examined first.

A wavelet is constructed in terms of a scaling function which is itself defined in terms of itself at higher scales [3].

$$\phi(x) = 2\sum_k h_k \phi(2x - k) \tag{A.1}$$

The wavelet is then defined,

$$\psi(x) = 2\sum_k g_k \phi(2x - k) \tag{A.2}$$

In the frequency domain as one increases the domain of the scaling function equation (A.1) becomes.

$$\Phi(\omega) = 2H(\frac{\omega}{2})\Phi(\frac{\omega}{2}) \tag{A.3}$$

It becomes clear then that as one iterates the scaling function then,

$$\Phi(\omega) = \left[\prod_{k=1}^{n} H(\frac{\omega}{2^k}) \right] \Phi(\frac{\omega}{2}) \tag{A.4}$$

or, after iterating to infinity,

$$\Phi(\omega) = \prod_{k=1}^{\infty} H(\frac{\omega}{2^k}) \tag{A.5}$$

This explains why the scaling function is sometimes referred to as the refinement equation. The refinement equation can be used to generate the scaling function at different scales.

A.1.1 Constraints and convergence

Uniqueness of the coefficients h_k and g_k is ensured by the following, (by integrating),

$$\int \phi \, dx = 2 \sum_k h_k \int \phi(2x - k) \, d(2x - k)$$
$$\int \psi \, dx = 2 \sum_k g_k \int \psi(2x - k) \, d(2x - k)$$

which ensures,

$$\sum_k h_k = 1$$
$$\sum_k g_k = 1$$

Both the above equations are normalised, in the sense that, $\int \phi \, dx = 1$ and $\int \psi \, dx = 1$. All the constraints provide is that particular set of h_k and g_k will have one particular scaling function and wavelet. Naturally, these constraints are very dependent on the definition of wavelet and scaling functions used.

Invertibility of the wavelet transform, or multi-resolution analysis, is the so-called *admissibility* criterion,

$$C_\psi = \frac{1}{2\pi} \int_{-\infty}^{\infty} \|\Psi(\Omega)\|^2 \frac{d\Omega}{\|\Omega\|} < \infty \tag{A.6}$$

A.1.2 Polynomial approximation

Equation (A.6) must converge for the wavelet transform to be invertible. Alternatively, this implies zero mean of wavelet,

$$\Psi(0) = 0 \tag{A.7}$$

To improve the efficiency of the wavelet at finding singularities, the wavelet may be required to have a number of vanishing moments,

$$\int_{-\infty}^{\infty} x^n \psi(x) dx = 0 \tag{A.8}$$

The following Fourier identity explains why the number of vanishing moments is related to the number of zeros at $w = 0$.

$$\int_{-\infty}^{\infty} x^n \psi(x) dx \implies \frac{1}{2\pi} \frac{1}{(-j)^n} \frac{d\Psi(\omega)^n}{d\omega^n} \bigg|_{\omega=0} \tag{A.9}$$

Any $F(\omega)$ can be rewritten as a rational function factorised in to poles and zeros. Differentiating this rational function and evaluating the result at $\omega = 0$ results in only the zeros or poles at $\omega = 0$ being significant. The other poles and zeros can be ignored. N zeros at $\omega = 0$ correspond to differentiation of the function in the time domain N times. This implies that for a power series approximation of the function, the low order terms up to a power of $N - 1$ can be ignored. A wavelet having N vanishing moments, is then only non-zero at a point where the function has significant energy in the higher order terms in a power series expansion around this point.

Put another way, the wavelet transform can approximate functions which can be locally approximated by polynomials of up to power $N - 1$ with zero or small detail coefficients. The approximation is then done by the scaling function. Equation (A.9) is only zero if the integrand is zero identically.

This approximation quality of wavelets translates to the scaling function as follows (after all, what information cannot be carried by the wavelet coefficients must be contained in the approximation coefficients):

- The first $N - 1$ zeros must exist. Zeros exist at multiples of 2π.
- The scaling function cannot have zero mean.
- Given (from equation (A.4),

$$\Phi(\omega) = \left[\prod_{k=1}^{\infty} H\left(\frac{\omega}{2^k}\right) \right]$$

then the constraint on zeros of $\Phi(\omega)$ translates to zeros of $H(\omega)$ at $\omega = 2\pi n$. Each factor of $\Phi(\omega)$ is $H(\frac{\omega}{2^k})$ at $k = 1, 2, 3, ...$ Since the constraint on the mean prohibits any zero at $\omega = 0$ all other possible zeros of $\Phi(\omega)$ must lie at $\omega = \pi$. (By periodicity any odd multiple of π is the same as π.) The different factors of $\Phi(\omega)$ divide the the input frequencies by powers of 2. The input frequency is a multiple of two. Each factor is the evaluated either on a multiple of 2π or an odd multiple of π.

The zeros of Φ are then controlled by the zeros of H.

- The zeros of $H(\omega)$ at $\omega = \pi$ are a necessary condition for regularity, but, of itself, not sufficient to ensure regularity. Regularity ensures that the scaling and wavelet functions are the impulse responses of low and high pass filters respectively. This also corresponds to the wavelets from filter banks approach. The scaling function is set by the low pass channel, and the wavelet by the high pass channel.

A.1.3 Construction of the Wavelet from the Scaling Function and vice versa

In order to provide an efficient decomposition the wavelet is always orthogonal to the scaling function. One can be constructed from the other. Orthogonal implies that the inner product of the two functions is zero, irrespective of position.

A wavelet can be created from the scaling function as follows,

$$\psi(x) = \sum_k W_k \phi(2x - k) \tag{A.10}$$

$$\langle \phi(t-k), \psi(t) \rangle = 0$$

$$\int \Phi(\omega) e^{j\omega k} \Psi^*(\omega)\, d\omega = 0 \tag{A.11}$$

The wavelet from scaling function is usually defined as,

$$\psi(x) = \sum_k (-1)^k c_{1-k} \phi(2x - k) \tag{A.12}$$

This equation does follow from the requirement that the wavelet and scaling function be orthogonal. Some common wavelets are summarised in table A.1.

A.1.4 Time-frequency resolution

The time-frequency resolution of wavelets is measured by the product of the Root Mean Squared (RMS) spread of the scaling and function in the time and frequency domain [3].

$$\sigma_T^2 = \frac{\int (t - \bar{t}) \|\phi(t)\|^2 dt}{E} \tag{A.13}$$

$$\sigma_\Omega^2 = \frac{\int (\Omega - \overline{\Omega}) \|\Phi(\Omega)\|^2 d\Omega}{E} \tag{A.14}$$

Where E is the energy in the scaling function or,

$$E = \int_{-\infty}^{\infty} \|\phi(t)\|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} \|\Phi(\Omega)\|^2 d\Omega \tag{A.15}$$

Name	Index	Scaling Coefficients	Wavelet Coefficients
Haar	0	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$
	1	$\frac{1}{\sqrt{2}}$	$\frac{-1}{\sqrt{2}}$
Daubechies N2	0	0.4829629131445341	-0.1294095225512603
	1	0.8365163037378077	-0.2241438680420134
	2	0.2241438680420134	0.8365163037378077
	3	-0.1294095225512603	-0.4829629131445341
Daubechies N3	0	0.3326705529500825	0.0352262918857095
	1	0.8068915093110924	0.0854412738820267
	2	0.4598775021184914	-0.1350110200102546
	3	-0.1350110200102546	-0.4598775021184914
	4	-0.0854412738820267	0.8068915093110924
	5	0.0352262918857095	-0.3326705529500825
Daubechies N4	0	0.2303778133088964	-0.010597401785069
	1	0.7148465705529155	-0.0328830116668852
	2	0.6308807679398587	0.0308413818355607
	3	-0.0279837694168599	0.1870348117190931
	4	-0.1870348117190931	-0.0279837694168599
	5	0.0308413818355607	-0.6308807679398587
	6	0.0328830116668852	0.7148465705529155
	7	-0.010597401785069	-0.2303778133088964
Daubechies N5	0	0.1601023979741929	0.0033357252854738
	1	0.6038292697971895	0.012580751999082
	2	0.7243085284377726	-0.0062414902127983
	3	0.1384281459013203	-0.0775714938400459
	4	-0.2422948870663823	-0.0322448695846381
	5	-0.0322448695846381	0.2422948870663823
	6	0.0775714938400459	0.1384281459013203
	7	-0.0062414902127983	-0.7243085284377726
	8	-0.012580751999082	0.6038292697971895
	9	0.0033357252854738	-0.1601023979741929
Daubechies N6	0	0.1115407433501095	-0.0010773010853085
	1	0.4946238903984533	-0.0047772575109455
	2	0.7511339080210959	5.538422011614E-4
	3	0.3152503517091982	0.0315820393174862
	4	-0.22626469396544	0.0275228655303053
	5	-0.1297668675672525	-0.0975016055873225
	6	0.0975016055873225	-0.1297668675672525
	7	0.0275228655303053	0.22626469396544
	8	-0.0315820393174862	0.3152503517091982
	0	5.538422011614E-4	-0.7511339080210959
	10	0.0047772575109455	0.4946238903984533
	11	-0.0010773010853085	-0.1115407433501095

and,

$$\bar{t} = \frac{\int_{-\infty}^{\infty} t\|\phi(t)\|^2 dt}{E} \tag{A.16}$$

and,

$$\overline{\Omega} = \frac{\frac{1}{2\pi}\int_{-\infty}^{\infty} \Omega\|\Phi(\Omega)\|^2 d\Omega}{E} \tag{A.17}$$

A.2 Bi-orthogonality

The notion of Bi-orthogonality involves the generating a wavelet transform or multi-resolution using one scaling function and wavelet and then reconstructing using a different scaling function and wavelet. If ϕ and ψ are Bi-orthogonal to $\tilde{\phi}$ and $\tilde{\psi}$ then,

$$< \tilde{\phi}, \psi(\cdot - l) > = < \tilde{\psi}, \phi(\cdot - l) > = 0 \tag{A.18}$$

and;

$$< \tilde{\phi}, \phi(\cdot - l) > = < \tilde{\psi}, \psi(\cdot - l) > = \delta_l \tag{A.19}$$

As an aside on notation,

$$< \phi\ , \psi(\cdot - l) > \overset{def}{=} \int_{-\infty}^{\infty} \overline{\phi\ (x)}\psi(x - l)dx \tag{A.20}$$

Bi-orthogonality can be illustrated as follows:

(i) If one has an N dimensional vector space then a basis is defined as orthogonal if the dot product of each basis vector with every other basis vector is zero identically.

(ii) A Bi-orthogonal basis in contrast has some correlation between the various basis vectors. Note that none of the basis vectors is redundant, or put another way, every point in this N dimensional vector space can be reconstructed using these N slightly correlated basis vectors.

(iii) Projecting a vector onto these Bi-orthogonal basis vectors leaves the problem of perfect reconstruction. Unlike the orthogonal basis case, reconstruction is not simply adding the basis vectors weighted by the inner product of the input vector and the basis vectors. A vector used for reconstruction must correlate with the corresponding vector used to project an input vector into the basis space, but still be orthogonal to other basis vectors. As an example suppose $(1, 0)$ and $(1, 1)$ happen to be the projection be-orthogonal basis pair for a two dimensional vector space.

The reconstruction pair will then be $(1, -1)$ and $(0, 1)$ respectively. Note that $(1, 0)$ and $(0, 1)$ are orthogonal as are $(1, 1)$ and $(1, -1)$. On the other hand $(1, 0)$ and $(1, -1)$ correlate as do $(1, 1)$ and $(0, 1)$.

A.3 Wavelets from filter banks

A close relationship between wavelets and filter banks exist, a wavelet transform is essentially a perfect reconstruction filter bank with either/both high-pass and low-pass channels filtered again into high and low pass bands and so on. Figure A.1 shows a simple filter bank with reconstruction from the high and low-pass channels.

To maintain the possibility for Bi-orthogonality the different filters are denoted as follows:

(i) $\tilde{g}$ is the high pass filter.

(ii) $\tilde{h}^0$ is the low pass filter.

(iii) g^0 reconstructs the high pass coefficients after super-sampling.

(iv) h reconstructs the low pass coefficients after super-sampling.

The perfect reconstruction property ensures that the output is identical to the input.

This can be shown as follows for images provided the filters are separable [3],

$$HP(z^2) = \frac{1}{2}\Big(\tilde{g}(z)I(z) + \tilde{g}(-z)I(-z)\Big) \tag{A.21}$$

Adding the $\tilde{g}(-z)I(-z)$ term ensures that the odd powers of z cancel.

$$LP(z) = \frac{1}{2}\Big(\tilde{h}^0(z)I(z) + \tilde{h}^0(-z)I(-z)\Big) \tag{A.22}$$

The reconstructed signal is then,

$$\begin{aligned} R(z) &= g^0(z)HP(z^2) + h(z)LP(z^2) \\ &= g^0(z)\frac{1}{2}\Big[\tilde{g}(z)I(z) + \tilde{g}(-z)I(-z)\Big] + h(z)\frac{1}{2}\Big[\tilde{h}^0(z)I(z) + \tilde{h}^0(-z)I(-z)\Big] \\ &= \frac{1}{2}\Big[g^0(z)\tilde{g}(z) + h(z)\tilde{h}^0(z)\Big]I(z) + \frac{1}{2}\Big[g^0(z)\tilde{g}(-z) + h(z)\tilde{h}^0(-z)\Big]I(-z) \\ &= T(z)I(z) + S(z)I(-z) \end{aligned} \tag{A.23}$$

Perfect reconstruction demands,

$$\begin{aligned} T(z) &= 1 \\ S(z) &= 0 \end{aligned}$$

$S(z) = 0$ implies no aliasing and for this to hold,

$$\begin{aligned} h &= F(z)\tilde{g}(-z) \\ g^0 &= -F(z)\tilde{h}^0(-z) \end{aligned}$$

$T(z) = 1$, implies $F(z)$ must be,

$$F(z) = \frac{2}{\tilde{h}^0(z)\tilde{g}(-z) - \tilde{h}^0(-z)\tilde{g}(z)} \tag{A.24}$$

$F(z)$ is derived by solving $S(z)$ for $g^0(z)$ in terms of $\tilde{g}^0$, $\tilde{h}^0$ and h; substituting into $T(z)$ and solving for h. Similarly for g^0. The form of $F(z)$ in equation (A.24) implies $F(z) = -F(-z)$. $T(z) = 1$ then becomes,

$$T(z) = \frac{1}{2}\Big(g^0(z)\tilde{g}(-z) + h(z)\tilde{h}^0(-z)\Big) = 1 \tag{A.25}$$

and,

$$T(z) = \frac{1}{2}\Big(\tilde{g}^0(z)g^0(z) + \tilde{g}^0(-z)g^0(-z)\Big) = 1 \tag{A.26}$$

By substituting for $\tilde{g}(z)$ and $\tilde{g}(-z)$ to yield the *duality condition.* $g^0(z)\tilde{g}(-z)$ then has no even terms in z, and is then the convolution expression,

$$\sum_k g^0[k]\tilde{g}[2n-k] = \delta[n] \tag{A.27}$$

Similarly for $\tilde{h}^0$ and $\tilde{g}$,

$$T(z) = \frac{1}{2}\Big(- \tilde{g}(z)h(z) - \tilde{g}(-z)h(-z)\Big) = 1 \tag{A.28}$$

and again $\tilde{h}^0(z)h(-z)$ then has no even terms in z, and is then the convolution expression,

$$\sum_k \tilde{h}^0[k]h[2n-k] = \delta[n] \tag{A.29}$$

(A.30)

Note that:

$$\begin{aligned} \tilde{g}(z)h(z) &= F(z)h(-z)h(z) && \text{and} \\ \tilde{h}^0(z)g^0(z) &= -F(z)g^0(-z)g^0(z) && \text{are both odd polynomials.} \end{aligned}$$

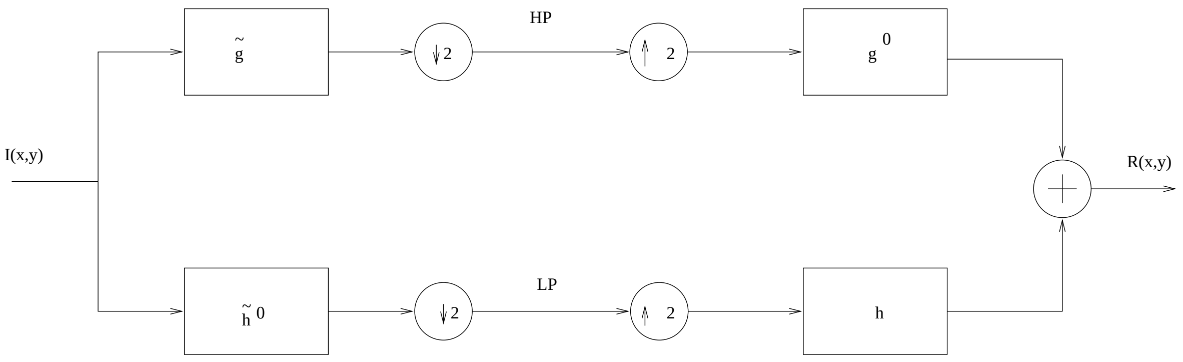


FIGURE A.1 A simple filter bank

A.3.1 A brief introduction to the wavelet transform

A wavelet transform is usually defined as the projection of the signal onto a wavelet for all wavelet positions and all scales.

$$W_f(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \psi(\frac{t-b}{a}) f(t)dt \tag{A.31}$$

The inverse transform is given by,

$$f(t) = \frac{1}{C} \int_{-\infty}^{\infty} \int_{0}^{\infty} \frac{W(a,b)\psi_{ab}(t)}{a^2} dadb \tag{A.32}$$

Where C is given by:

$$C = \int_0^{\infty} \frac{\|\Psi(\Omega)\|^2}{\Omega} d\Omega \tag{A.33}$$

A.3.2 An overview of multi-resolution analysis

Multi-resolution analysis decomposes a signal into into its average and detail components [3]. The process can be repeated for both parts but is usually only done for the average coefficients. The average coefficients are calculated by projecting the signal onto the scaling function and the detail coefficients by projecting the signal onto the wavelet. Because of redundancies in the representation the average and detail coefficients can be subsampled without loss of information [3].

Appendix B

Mathematical Morphology – some definitions

This chapter will briefly define some mathematical morphology terms used in this thesis. The reader is referred to [35, 63, 64, 65, 124, 125] for a more complete introduction to Mathematical Morphology, [35, 64] serves as a good introduction, and [65] shows examples of applying morphology to real world problems. As the term implies, morphology is concerned with shapes of objects within an image. Originally morphology was based on set theory and later extended to greyscale functions. Subsequently only greyscale morphology will be defined.

B.1 Binary morphology

After [124, 125] binary morphology will be defined as below, table B.1,

Operation	Operator	Definition	Properties
Dilation	$\oplus$	$\bigcap_{b \in g} f_{-b}$	Increasing, extensive
Erosion	$\ominus$	$\bigcup_{b \in g} f_{-b}$	Anti-extensive
Opening	$\circ$	$(f \ominus g) \oplus g$	Idempotent, anti-extensive
Closing	$\bullet$	$(f \oplus g) \ominus g$	Idempotent, extensive

Where f_{-b} is the set f now centred on $-b$, and $b \in g$. g is considered to be the structuring element and f an object in a binary image. The dilation of an object, allows for all pixels which, when g is centred on a pixel p, g_p intersects the object. Similarly, the erosion of an object results in the set of pixels which, when g is placed one of these pixels p, g_p is a proper subset of the object (is contained in the object).

The properties attributed to binary morphological operators are defined as follows, with $\phi()$ denoting an operation, and Q denoting an arbitrary binary region,

Extensive $\phi(Q) \supset Q$.

Anti-extensive The above is not true for ϕ, $\phi(Q) \subseteq Q$.

Idempotent Repeating the operator has no further effect, $\phi(Q) = \phi(\phi(Q))$.

Increasing If $R \subseteq Q, \phi(R) \subseteq \phi(Q)$.

Clearly binary morphological filters are translation invariant.

B.2 Greyscale morphology

A greyscale image can be decomposed into a series of level sets. A level set, at level t, is the image thresholded at level t (denoted by χ_t). The level sets can then be filtered and then reconstituted to form the output image. These ideas form the basis of greyscale morphology [124, 125].

After [124, 125] greyscale morphology will be defined as below, table B.2,

Operation	Definition
Dilation	$(f \oplus g)(x, y) = \bigwedge_{(a,b)\in D}\{f(a, b) + g(a - x, b - y)\}$
Erosion	$(f \ominus g(x, y) = \bigvee_{(a,b)\in D}\{f(a, b) - g(a - x, b - y)\}$

Note that function g is mirrored about the axes. D is the domain of the image f. The properties of the greyscale filters are identical to those of the binary operators. The definitions for opening and closing coincide as well.

Morphological operations have one or more of the following properties, where $\phi()$ denotes a greyscale morphological operator, and f denotes a greyscale image.

Extensive $\phi(f) \leq f$.

Anti-extensive $\phi(f) \leq f$.

Idempotent Iterating the morphological filter has no further effect.

Increasing If $g \leq f, \phi(g) \leq \phi(f)$.

B.3 Greyscale reconstruction

Two greyscale reconstruction techniques exist, namely, greyscale reconstruction by dilation, and greyscale reconstruction by erosion [144]. Greyscale reconstruction by dilation is binary conditional dilation extended to greyscale images. Recall that conditional dilation may be defined as [144, 35] [65][pg. 368],

$$I_{\text{out}}^{k+1} = \left(I_{\text{out}}^{k} \oplus \text{sel}\right) \cap I_{\text{mask}} \tag{B.1}$$

where, I_{out}^{0} is initialised to a given marker image. The structuring element used is the unit circle for the sampling grid used. The dilation is repeated until stability. Figures B.1(a) and B.1(b) shows the structuring elements for four and eight connected objects on the rectangular sampling grid.

	1	
1	0	1
	1	

(a) The four connected unit circle.

1	1	1
1	0	1
1	1	1

(b) The eight connected unit circle.

FIGURE B.1 Unit circles for the rectangular sampling grid.

Reconstruction by erosion may be similarly defined,

$$I_{\text{out}}^{k+1} = \left(I_{\text{out}}^{k} \ominus \text{sel}\right) \cup I_{\text{mask}} \tag{B.2}$$

Fast algorithms to perform the above do exist, see, for example, Vincent [144].

B.3.1 A fast greyscale reconstruction algorithm

Beucher and Meyer describe a fast greyscale reconstruction algorithm [13]. Only the greyscale reconstruction by dilation will be described, the modification for reconstruction by erosion is trivial. As it stands, however, the implementation due to Beucher and Meyer is less efficient than it could be. Beucher and Meyer's algorithm will be described, and modifications proposed.

Beucher and Meyer

The greyscale reconstruction by dilation algorithm due to Beucher and Meyer is as follows [13]:

(i) Sort the pixels of the marker image into descending intensity. An ordered Queue is used. An ordered queue associates a queue with each intensity value in the image. All pixels with the same intensity are inserted into the same queue. If interrogated, the ordered queue returns the pixel with the largest intensity first, before removing this pixel from the queue.

(ii) For each pixel in the ordered queue,

(a) For the next unprocessed (unlabelled) pixel p in the ordered queue, and its unprocessed neighbours, q

$$\text{Result}(q) = \text{Minimum}(\text{Mask}(q), \text{Marker}(p)) \tag{B.3}$$

Place q, into the ordered queue with the same priority (intensity) as $\text{Result}(q)$. Label q as being in the ordered queue (label 1).

Note that q is marked as processed, to ensure that a pixel is processed once only.

(b) If the next pixel in the ordered queue has been processed, then assign a new label (label 2) to the pixel.

As a optimisation Beucher and Meyer suggest representing a pixel by its address. This assigns a unique integer to each pixel, in order of occurrence in the raster scan of the image, for example.

Some modifications

The peaks of a marker image in the reconstruction algorithm are important. Recall that greyscale reconstruction by dilation is an iterative flat dilation, using a disc structuring element of radius one (a "plus" shape if four connectivity is assumed on the rectangular grid, a square shape if eight connectivity is assumed). After each flat dilation the pixel-wise maximum of the mask and marker images is taken to produce the new marker image. The dilation is iterated until stability. The marker image then never exceeds the mask image, and, since flat dilation is used, the maximum intensity in the final result image is the maximum intensity in the original marker image.

In principle, given the definitions for conditional dilation, one may perform conditional dilation on the level sets of the mask and marker images. Recall that level sets of an image may be generated by thresholding an image with all thresholds possible [17].

The reconstruction stage examines a pixel once only. The reconstruction stage is then proportional to the number of pixels in the image (or order n, $O(n)$). The sorting stage may not be order n if the image pixels are floating point numbers. It is known that the fastest floating point sorting algorithm, Quicksort, is $O(n \log_2 n)$ [150]. If the pixel intensities are integer valued, however, an order n sorting algorithm, Bucket sort, exists [150].

Bucket sort has been used in Beucher and Meyer's watershed segmentation algorithm [13]. Bucket sort [150], essentially, associates a list with each pixel intensity. As the image is scanned, the pixel under consideration is added to the list corresponding to its intensity.

Some difficulties with Beucher and Meyer's algorithm include,

(i) How are non-integer valued intensity images to be handled? The sorting algorithm used in Beucher and Meyer's algorithm assumes integer valued pixels for fast sorting.

(ii) All pixels in the marker image are placed in the ordered queue, when only pixels on the peak are important.

It should be emphasised that some care is needed in the implementation of the queue, and the representation of the pixels (p and q). The alternative is a slow implementation. It is suggested that a circular integer list be used.

(i) Non-integer valued pixels imply non-integer valued priorities. The priority queue recommended by Beucher and Meyer may be replaced by a min heap (see [150] for an introduction to min heaps). A min heap is a data structure allowing for fast extraction of the maximum (or minumum) element of a list.

(ii) Scanning along each row and column of an image allows for the local maxima to be extracted. If a pixel is lower in intensity than its predecessor, then the predecessor is a peak pixel. This does not extract all pixels on a local maximum, but will find at least one pixel at the intensity of the local maximum. Only peak pixels need to be inserted in the priority queue or min heap.

(iii) Convergence of the algorithm may be improved by first reconstructing using Vincent's sequential reconstruction algorithm [144]. In the case of greyscale reconstruction by dilation, a new marker image is generated. Pixels on the new marker image at the same intensity as the mask image may be labelled as processed and not considered at all subsequently.

In the instance of froth image analysis, the marker image is generated in some way from the mask image (the original froth image. The mask and marker images are than likely to be similar in shape, and a sequential preprocessing stage is likely to eliminate a large number of pixels from consideration. Non-integer valued pixel intensities are likely given most low-pass filter kernels and filtering techniques like anisotropic diffusion [113]. The results of filtering must then be rounded to integer values and represents another source of noise.

It should be noted that Beucher and Meyer's marker based watershed segmentation algorithm uses a form of sorting to optimise finding, and flooding from, minima in an image. Using a min heap allows extends this algorithm to handle real valued images.

Bibliography

[1] Emile Aarts and Jan Korst. *Simulated Annealing and Boltzman Machines*. Wiley, 2nd edition, 1990.

[2] Janet Aisbett. Optical flow with intensity weighted smoothing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(5):512–522, May 1989.

[3] Ali N. Akansu and Mark J. T. Smith, editors. *Subband and Wavelet Transforms. Design and Applications*. The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 1996.

[4] Luis Alvarez, Pierre-Louis Lions, and Jean-Michel Morel. Image selective smoothing and edge detection by non-linear diffusion, II. *Siam Journal of Numerical Analysis*, 29(3):845 – 866, June 1992.

[5] Luis Alvarez and Luis Mazorra. Signal and image restoration using shock filters and anisotropic diffusion. *Siam Journal of Numerical Analysis*, 31(2):590 – 605, April 1994.

[6] Oscar C. Au, Yiu-Hung Fok, and Ross D. Murch. Novel fast block motion estimation in feature space. In *ICIP 94 Volume III of III*, volume 3 of *ICIP 94*, pages 209–212, 1994.

[7] Jean Babaud, Andrew P. Witkin, Michel Baudin, and Richard O. Duda. Uniqueness of the Gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):26 – 33, January 1986.

[8] Shumeet Baluja and Rich Caruana. Removing the genetics from the standard genetic algorithm. Unpublished paper.

[9] John S. Baras, Sohail Zafar, and Ya-Qin Zhang. Predictive Block-Matching Motion Estimation for TV Coding – Part 1: Inter-Block Prediction. *IEEE Transactions on Broadcasting*, 37(3):97 – 101, September 1991.

[10] J. R. Bergen and R. Hingorani. Hierarchical motion-based frame rate conversion. Technical report, David Sarnoff Research Center, Princeton, April 1990.

[11] Frederik Bergholm and Stefan Carlsson. A theory of optical flow. *Computer Vision Graphics and Image Processing: Image Understanding*, 53(2):171–188, March 1991.

[12] S. Beucher and C. Lantejoul. Use of watersheds in countour detection. In *International Workshop on Image Processing: Real-Time Edge and Motion Detection/Estimation*, Rennes, France, September 17–21 1979. http://cmm.ensmp.fr/ beucher/publi/watershed.pdf.

[13] S. Beucher and F. Meyer. The morphological approach to segmentation: The watershed transformation. In E. R. Dougherty, editor, *Mathematical Morphology in Image Processing*, chapter 12, pages 433 – 481. Marcel Decker, 1993.

[14] M. Bezuidenhout, J. S. J. Van Deventer, and D. W. Moolman. The identification of perturbations in a base metal flotation plant using computer vision of the froth surface. *Minerals Engineering*, 10(10):1057–1073, 1997.

[15] Micheal J. Black, Guillermo Sapiro, David H. Marimont, and David Heeger. Robust anisotropic diffusion. *IEEE Transactions on Image Processing*, 7(3):421–432, March 1998.

[16] C. P. Botha, D. M. Weber, M. vcan Olst, and D. W. Moolman. A practical system for realtime on-plant flotation froth visual parameter extraction. In *Proceedings of IEEE Africon 1999*, September 1999.

[17] Edmond J. Breen and Ronald Jones. Attribute openings, thinnings, and granulometries. *Computer Vision Graphics and Image Processing*, 64(3):377 – 389, November 1996.

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

[19] Francine Catte, Pierre-Louis Lions, Jean-Michel Morel, and Tomeu Coll. Image selective smoothing and edge detection by non-linear diffusion. *Siam Journal of Numerical Analysis*, 29(1):182 – 193, February 1992.

[20] Liang-Gee Chen, Wia-Ting Chen, Yeu-Shen Jehng, and Tzi-Dar Chiueh. An efficient parallel motion estimation algorithm for digital image processing. *IEEE Transactions on Circuits and Systems for Video Technology*, 1(4):378 – 385, December 1991.

[21] Liang-Gee Chen, Tzi-Dar Chiueh, and Her-Ming Jong. Accuracy improvement and cost reduction of 3-step search block matching algorithm for video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(1):88 – 90, February 1994.

[22] Mei-Juan Chen, Liang-Gee Chen, and Tzi-Dar Chiueh. One-dimensional full search motion estimation algorithm for video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(5):504 – 509, October 1994.

[23] Tao Chen, Kai-Kuang Ma, and Li-Hui Chen. Tri-state median filter for image denoising. *IEEE Transactions on Image Processing*, 8(12):1834 – 1838, December 1999.

[24] Roland T. Chin and Chia-Lung Yeh. Quantitative evaluation of some edge-preserving noise-smoothing techniques. *Computer Vision Graphics and Image Processing*, 23:67 – 91, 1983.

[25] Kyujin Cho and Peter Meer. Image segmentation from consensus information. *Computer Vision and Image Understanding*, 68(1):72–89, October 1997.

[26] A. Cipriano, M. Guarini, A. Soto, H. Biceno, and D. Mery. Expert supervision of flotation cells using digital image processing. In *Proceedings of the XX IMPC*, pages 281 – 292, Aachen, Germany, 1997.

[27] A. Cipriano, M. Guarini, R. Vidal, A. Soto, C. Sepulveda, D. Mery, and H Briseno. A real time visual sensor for supervision of flotation cells. *Minerals Engineering*, 11(6):489 – 499, 1998.

[28] R Courant. *Differential and Integral Calculus*. Blackie and Son, 1944. Translated J E. McShane.

[29] J. S. J. Van Deventer, D. W. Moolman, and C. Aldritch. Visualisation of plant disturbances using self-organising maps. *Computers in Chemical Engineering, Supplement*, 20:S1095 – S1100, 1996.

[30] Michael B. Dillencourt, Hannah Samet, and Markku Tamminen. A general approach to connected-component labelling for arbitrary image representations. *Journal of the Association for Computing Machinery*, 39(2):253 – 280, April 1992.

[31] Bogdan P. Dobrin, Timo Viero, and Moncef Gabbouj. Fast watershed algorithms: analysis and extensions. In *Nonlinear Image Processing V*, volume 2180, pages 209 – 220. SPIE, 1994.

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

[33] David L. Donoho. De-noising via soft thresholding. Technical Report 409, Department of Statistics, Stanford University, 1992. http://ww-stat.stanford.edu/research/date.html.

[34] Edward R. Dougherty. *An Introduction to Morphological Image Processing*, volume TT9 of *Tutorial Texts in Optical Engineering*. SPIE Optical Engineering Press, 1992.

[35] Edward R. Dougherty. *An Introduction to Morphological Image Processing*. SPIE Optical Engineering Press, 1992.

[36] Edward R. Dougherty and Jaako Astola. *An Introduction to Nonlinear Image Processing*. SPIE Optical Engineering Press, 1994.

[37] Gunther R. Drevin and Adri Reyneke. Determining rock particle roundness using the Lorentz curve. In Z. Fazekas, G. R. J. Cooper, and F. Aghdasi, editors, *Proceedings of the Eleventh Annual Symposium of the Pattern Recognition Association of South Africa*, pages 99 –104, 2000.

[38] Qiang Du, vance Faber, and Max Gunzburger. Centroidal Voronoi teselations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[39] Frederic Dufaux and Fabrice Moscheni. Motion estimation techniques for digital TV: A review and new contribution. *Proceedings of the IEEE*, pages 858 – 876, June 1995.

[40] Bruce Fischl and Eric L. Schwartz. Fast adaptive alternatives to nonlinear diffusion in image enhancement: Green's function approximators and nonlocal filters. In *First International Conference on Scale-Space Theory in Computer Vision*, Utrecht University, Utrecht, the Netherlands, July 1997. http://cns-web.bu.edu/pub/fischl/publications.html.

[41] Bruce Fischl and Eric L. Schwartz. Learning an integral equation approximation to nonlinear anisotropic diffusion in image processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):342 – 352, April 1997. http://cns-web.bu.edu/pub/fischl/publications.html.

[42] Bruce Fischl, Eric L. Schwartz, and Micheal J. Cohen. Real-time anisotropic diffusion using space-variant vision. *International Journal of Computer Vison*, 28(3):199 – 212, July 1998. http://cns-web.bu.edu/pub/fischl/publications.html.

[43] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. The systems programming series. Addison-Wesley, second edition in C edition, 1996.

[44] Luc De Fos and Michael Stegherr. Parameterisable VSLI architectures for the full search block-matching algorithm. *IEEE Transactions on Circuits and Systems*, 36(10):1309 – 1316, October 1989.

[45] J. J. Francis and G de Jager. Watershed segmentation based motion estimation of froth image sequences. In *Proceedings of the Eighth Annual Symposium of the Pattern Recognition Association of South Africa*, 1998.

[46] I. M. Gelfand and S. V. Fomin. *Calculus of Variations*. Selected Russian Publications in the Mathematical Sciences. Prentice–Hall, 1963.

[47] M. Ghanbari. The cross-search algorithm for motion estimation. *IEEE Transactions on Communications*, 38 No 7:950 – 953, July 1990.

[48] Nathalie Giordana and Wojciech Pieczynski. Estimation of generalized multisensor hidden markov chains and unsupervised image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):465–475, May 1997.

[49] Prem Goel and Brani Vidakovic. Wavelet transformations as diversity enhancers. *Discussion Paper Series, Institute of Statistics and Decision Sciences, Duke University*, ISDS, Duke University(95-04), 1995. http://www.isds.duke.edu/papers/WorkingPapers/95-04.ps.

[50] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company., 1993.

[51] Jaideva C. Goswami and Andrew K. Chan. *Fundamentals of Wavelets. Theory, algorithms and applications.* John Wiley and Sons, 1999.

[52] F. Gourram-Badri, P. Conil, and G. Morizot. Study of parameters which have influence on the froth flotation characteristics. In *Proceedings of the XX IMPC, 21 – 26 September*, pages 95–105, Aachen, Germany, 1997.

[53] Marcelo Guarini, Aldo Cipriano, Alvaro Soto, and Andres Guesalaga. Using image processing techniques to evaluate the quality of mineral flotation process. In *Proceeedings of the International Conference on Signal Processing, Applications and Technology*, pages 1227–1231, Boston, Massachusets, USA, 1995.

[54] Sandeep N. Gupta and Jerry L. Prince. Stochastic models for DIV-CURL optical flow methods. Technical Report MD 21218, The Johns Hopkins University.

[55] Sandeep N. Gupta, Jerry L. Prince, and Stephanos Androutsellis-Theookis. Bandpass optical flow for tagged MRI imaging. Technical report, The Johns Hopkins University, 1997. http://www.ece.jhu.edu/reports/1997.html.

[56] M. W. Hansen and W. E. Higgins. Watershed based maximum homogeneity filtering. *IEEE Transactions on Image Processing*, 8(7):982 – 988, July 1999.

[57] Michael W. Hansen and William E. Higgins. Relaxation methods for supervised image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-19(9):949 – 962, September 1997.

[58] J. M. Hargrave, G. J. Brown, and S. T. Hall. Use of fractal dimensions in describing froth structure. In *Proceedings of the XX IMPC, 21 – 26 September*, pages 87 – 93, Aachen, 1997.

[59] J. M. Hargrave and S. T. Hall. Diagnosis of concentrate grade and mass flowrate in tin flotation from colour and surface texture analysis. *Minerals Engineering*, 10(6):613 – 621, 1997.

[60] J. M. Hargrave, N. J. Miles, and S. T. Hall. The use of grey level measurement in predicting coal flotation performance. *Minerals Engineering*, 9(6):667 – 674, 1996.

[61] J. M. Hargrave, H. H. A. Shirazi, and S. T. Hall. Diagnosis of industrial flotation performance from colour and surface texture analysis. In *Processing of Complex Ores - Mineral Processing*

and the Environment, Proceedings of the 2nd UBC-McGill Bi-Annual Symposium on Fundamentals of Mineral Processing and the Environment, pages 177–184, SubBury, Ontario, CIM, 1997.

[62] D. C. He and L. Wang. Texture Classification using Texture Spectrum. *Pattern Recognition*, 23(8):905–910, 1990.

[63] Henk J. A. M. Heijmans. Theoretical aspects of grey-level morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(6):568 – 582, June 1991.

[64] Henk J. A. M. Heijmans. *Morphological Image Operators*. Number 25 in Advances in electronics and electron physics. Academic Press, 1994.

[65] Henk J. A. M. Heijmans and Jos B. T. M. Roerdink, editors. *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 12 of *Computational Imaging and Vision*. Kluwer Academic Publishers, 1998.

[66] R. C. Hochreiter, D. C. Kennedy, W. Muir, and A. I. Woods. Platinum in South Africa (metal review series no. 3). *Journal of the South African Institute of Mining and Metallurgy*, 85(6):165 – 185, June 1985.

[67] Berthold K. P. Horn and Brian G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185 – 203, 1981.

[68] B.K Horn and B. G. Schunk. *Computer Vision: Fundamentals*, chapter 6: Dynamic Vision, pages 481 – 497. ”Determining optical flow” from Artificial Intelligence 17:185–203, 1981.

[69] Caspar Horne, T. Naveen, and Ali Tabatabai. Study of the characteristics of the MPEG2 4:2:2 profile - application of the MPEG2 in studio environment. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):251 – 272, June 1996.

[70] International Organisation for Standardisation. *Generic coding of moving pictures and associated audio.*, h.262 edition.

[71] P. T. Jackway. Gradient watersheds in morphological scale-space. *IEEE Transactions on Image Processing*, 5(6):913 – 921, June 1996.

[72] Giovanni Jacovitti and Allessandro Neri. Mutiresolution circular harmonic decomposition. *IEEE Transactions on Signal Processing*, 48(11):3242–3247, November 2000.

[73] Bernd Jahne. *Digital Image Processing: Concepts, Algorithms and Scientific Applications*. Springer-Verlag, Berlin Heidelberg, 2 edition, 1993.

[74] Anil K. Jain. *Fundamentals of Digitam Image Processing*. Springer-Verlag, 1989.

[75] Alan Jeffrey. *Linear Algebra and Ordinary Differential Equations.* Blackwell Scientific., 1990.

[76] Ronald Jones. Connected filtering and segmentation using component trees: Efficient implementation algorithms. WWW. http://www.dms.CSIRO.AU/ ronaldj/pseudocode.

[77] Ronald Jones. Connected filtering and segmentation using component trees. *Computer Vision and Image Understanding*, 75(3):215 – 228, September 1999.

[78] S. Kappagantula and K. R. Rao. Motion compensated interframe image prediction. *IEEE Transactions on Communications*, COM-33(9):1011 – 1015, September 1985.

[79] Gabriel Katul and Brani Vidakovic. The partitioning of attached and detached eddy motion in the atmospheric surface layer using lorentz wavelet filtering. *Discussion Paper Series, Institute of Statistic and Decision Sciences, Duke University*, (95-05), 1995. http://www.isds.duke.edu/WorkingPapers/95-05.ps.

[80] James P. Keener. *Principles of applied Mathematics. Transformations and Applications.* Addison-Wesley, 1988.

[81] Benjamin B. Kimia and Kaleem Siddiqi. Geometric heat equation and nonlinear diffusion of shapes and images. *Computer Vision and Image Understanding*, 64(3):305 – 322, November 1996.

[82] Ron Kimmel and Alfred M. Bruckstein. Tracking level sets by level sets: A method for solving the shape from shading problem. *Computer Vision And Image Understanding*, 62(2):47 – 58, July 1995.

[83] Jacek Kordek and Jaroslaw Kulig. The analysis of diffraction patterns of flotation froth as the basis of estimation the content of a useful component. In *Proceedings of the XX IMPC, 21–26 September*, Aachen, 1997.

[84] D. P. Kottke and Ying Sun. Motion estimation via cluster matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(11):1128 – 1132, November 1994.

[85] Liang-Wei Lee, Jau-Yien Lee, and Jhing-Fa Wang. Dynamic search-window adjustment and interlaced search for block-matching algorithm. *IEEE Transactions on circuits and systems for video technology*, 3(1):85 – 87, February 1993.

[86] Renxiang Li, Ming L. Liou, and Bing Zeng. A new three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(4):438 – 442, August 1994.

[87] W. Li and E. Salari. Successive elimination algorithm for motion estimation. *IEEE Transactions On Image Processing*, 4(1):105 – 107, January 1995.

[88] Jean-Hsang Lin, Nirwan Ansari, and Jinhui Li. Nonlinear filtering by threshold decomposition. *IEEE Transactions on Image Processing*, 8(7):925–933, July 1999.

[89] Jia Liu. Flotation cell surface froth analysis. Master's thesis, Department of Electrical Engineering, University of Cape Town, 1995.

[90] Lunrng-Kuo Liu and Ephraim Feig. A block-based gradient descent search algorithm for block motion estimation in video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(4):419 – 422, August 1996.

[91] Ning Lu. *Fractal Imaging*. Academic Press, 1997.

[92] Ravikanth Malladi and James A. Sethian. A unified approach to noise removal, image enhancement, and shape recovery. *IEEE Transactions on Image Processing*, 5(11):1554 – 1568, November 1996.

[93] J. T. McClave and R. L. Schaeffer. *Probability and Statistics for Engineers*. International Thomson Publishers, 1995.

[94] William S. Meisel. *Computer-Oriented Approaches to Pattern Recognition.*, volume 83 of *Mathematics in Science and Engineering*. Academic Press, New York, San Francisco, London, 1972.

[95] Annick Montanvert and Azriel Rosenveld. Hierarchical image analysis using irregular tesselations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):307–316, April 1991.

[96] D. W. Moolman, C. Aldritch, S. J. Van Deventer J, and D. Bradshaw. The interpretation of flotation froth surfaces by using digital image analysis and neural networks. *Chemical Engineering Science*, 50(22):3501 – 3513, 1995.

[97] D. W. Moolman, C. Aldritch, J. S. J. van Deventer, and W. W. Stange. The classification of froth structures in a copper flotation plant by means of a neural net. *International Journal of Minerals Processing*, 43:193 – 208, 1995.

[98] D. W. Moolman, J. J. Eksteen, C. Aldritch, and J. S. J. van Deventer. The significance of flotation froth appearance for machine vision control. *International Journal of Minerals Processing*, 48:135 – 158, 1996.

[99] D. S. Moore and G. P. McCabe. *Introduction to the Practice of Statistics*. W. H. Freeman and Company, New York, 2nd edition, 1993.

[100] H. G. Musmann, P. Pirsh, and H. J. Grallert. Advances in picture coding. *Proceedings of the IEEE*, 73(4):523 – 531, April 1985.

[101] Makoto Nagao and Takashi Matsuyama. Edge preserving smoothing. *Computer Graphics and Image Processing*, 9:394 – 407, 1979.

[102] Balas Natarajan, Konstantinos Konstantinides, and Cormac Herley. Occam filters for stochastic sources with applications to digital images. *IEEE Transactions on Signal Processing*, 46(5):1434 – 1438, May 1998.

[103] P. Nesi, A. Del Bimbo, and D. Ben-Tzvi. A robust algorithm for optical flow estimation. *Computer Vision and Image Understanding*, 62(1):59–68, July 1995.

[104] A. N. Netravali and J. D. Robbins. Motion-compensated television coding: Part 1. *The BELL System technical Journal*, 58(3):631 – 669, March 1979.

[105] Hong Hai Nguyen and Paul Cohen. Gibbs random fields, fuzzy clustering, and the unsupervised segmentation of textured images. *Graphical Models and Image Processing*, 55(1):1–19, January 1993.

[106] K. K. Nguyen and A. J. Thornton. The application of texture based image analysis techniques in froth flotation. In *Proceedings of the DICTA-95, the 3rd Conference on Digital Imaging Computimng Techniques and Applications*, pages 371 – 376, Brisbane, Australia, 1995. Australian Pattern Recognition Society.

[107] Khoi Ke Nguyen and Peter Holtham. The application of pixel tracing techniques in the flotation process. In *Proceedings of the first joint Australian and New Zealand biennial conference on Digital Imaging and Vision Computing and Applications*, pages 207 – 212, 1997.

[108] Wiro Niessen, Koen L. Vinken, Joachim A. Wieckert, and Max A. Viergever. Nonlinear multiscale representations for image segmentation. *Computer Vision and Image Understanding*, 66(2):233 – 245, May 1997.

[109] Mark Nitzberg and Takahiro Shiota. Nonlinear image filtering with edge and corner enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-14(8):826 – 832, August 1992.

[110] Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. *Spatial Tesselations, Concepts and Applications of Voronoi Diagrams*. John Wilety and Sons, 199.

[111] Stephen Orr. Digital video spearheads TV and video conferencing applications. *Computer Design*, pages 59 – 70, December 1994.

[112] Stanley Osher and Leonid Rudin. Feature-oriented image enhancement using shock filters. *Siam Journal Numerical Analysis*, 27(4):919 – 940, August 1990.

[113] P. Perona and J. Malik. Scale-space filtering and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(7):629 – 639, July 1990.

[114] Lai-Man Po and Wing-Chung Ma. A novel four-step search algorithm for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):313 – 317, June 1996.

[115] William H. Press. *Numerical Recipes in C: the art of scientific computation*. Cambridge University Press, Cambridge, 2nd edition, 1992.

[116] Natan Pterfreund. The velocity snake: Deformable contour for tracking in spatio-velocity space. *Computer Vision and Image Understanding*, 73(3):346 – 356, March 1999.

[117] Subrate Rakshit and C. H. Anderson. Computation of optical flow using basis functions. *IEEE Transactions on Image Processing*, 6(9):1246 – 1254, September 1997.

[118] Steven K. Rogers and Matthew Kabrisky. *An Introduction to Biological and Artificial Neural Networks for Pattern recognition*. SPIE Optical Engineering Press, 1991.

[119] John C. Russ. *The Image Processing Handbook*. CRC Press, 1995.

[120] N. Sadr-kazemi and J. J. Cilliers. An image processing algorithm for measurement of flotation froth bubble size and shape distributions. *Minerals Engineering*, 10(10):1075 – 1083, 1997.

[121] A Said and W. A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transaction on Circuits and Systems for Video Technology*, 6, June 1996.

[122] Guillermo Sapiro and Vincent Caselles. Contrast enhancement via image evolution flows. *Graphical Models and Image Processing*, 59(6):407 – 416, November 1997.

[123] Elisa Sayrol, Antoni Gasull, and Javier R. Fonollosa. Motion estimation using higher order statistics. *IEEE Transactions on Image Processing*, 5(6):1077 – 1084, June 1996.

[124] P. Maragos R. W. Schaefer. Morphological filters - part I: Their set-theoretic analysis and relations to linear shift invariant filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(8):1153 – 1169, August 1987.

[125] P. Maragos R. W. Schaefer. Morphological filters - part II: Their relations to median, order statistic, and stack filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(8):1170 – 1184, August 1987.

[126] Philip A. Schweitzer. *Handbook of chemical separation techniques for chemical engineers*. McGraw-Hill, New York, 3rd edition, 1997.

[127] Jean Serra. *Image Analysis and Mathematical Morphology*, volume 1. Academic Press, 1982.

[128] P. De Smet and D. De Vleeschauwer. Motion-based segmentation using a thresholded merging strategy on watershed segments. In *Proceedings of the ICIP*, 1997.

[129] Ram Srinivasan and K. R. Rao. Predictive coding based on efficient motion estimation. *IEEE Transactions on Communications*, COM-33(8):888 – 896, August 1985.

[130] F. G. Stremler. *Introduction to communications systems.* Addison-Wesley, 1990.

[131] Kokichi Sugihara and Masao IRI. Construction of the Voronoi Diagram for ”one million” generators in single-precision arithmetic. *Proceedings of the IEEE*, 80(9):1471–1484, September 1992.

[132] D. Suter. Motion estimation and vector splines. In *Proceedings of the CVPR94, (Seattle)*, pages 939 – 942, 1994.

[133] J. W. De Swart, R. E. Van Vliet, and R. R. Krishna. Size, structure and dynamics of large bubbles in a two-dimensional slurry bubble column. *Chemical Engineering Science*, 51(20):4619 – 4629, 1996.

[134] Earl W. Swokowski. *Calculus with Analytic Geometry.*, volume Second Alternate Edition. PWS-KENT, 1988.

[135] Paul Symmonds. The investigation of the characterisation of flotation froth and design of a machine vision system for the monitoring the operation of a flotation cell ore concentrator. Master’s thesis, Department of Electrical Engineering, University of Cape Town, 1992.

[136] P. J. Symonds and G. de Jager. A technique for automatically segmenting images of the surface froth structures that are prevalent in industrial flotation cells. In *Proceedings of the 1992 South African Symposium on Communications and Signal Processing*, pages 111–115, University of Cape Town, Rondebosch South Africa, September 1992.

[137] Richard Szeliski. Prediction error as a a quality metric for motion and stereo. In *Seventh International Conference on Computer Vision*, pages 781–788, Kerkyra, Greece, 20–27 September 1999.

[138] Massimo Tistarelli. Multiple constraints to compute optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1243 – 1250, December 1996.

[139] Martin Vetterli and Jelena Kovacevic. *Wavelets and Subband Coding*. Prentice Hall, 1995.

[140] L. Vincent. Grayscale area openings and closings, their efficient implementation and applications. In J. Serra and P. Salembier, editors, *Mathematical Morphology and its applications to Signal Processing*, pages 22 – 27. UPC Publications, May 1992.

[141] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583 – 598, 1991.

[142] Luc Vincent. Morphological Area Openings and Closings for Greyscale Images. In *Proceedings of the NATO Shape in Picture Workshop*, pages 197 – 208, Driebergen, The Netherlands, 1992. Springer-Verlag. http://www.vincent-net.com/luc/papers/92shape_areaopen.djvu.

[143] Luc Vincent. Grayscale area openings and closings, their efficient implementations and applications. In *Proceedings of the EURASIP Workshop on Mathematical Morphology and its Applications to Signal Processing*, pages 22–27, 1993. http://www.vincent-net.com/luc/papers/93barcelona_areaopen.djvu.

[144] Luc Vincent. Morphological grayscale reconstruction in image analysis: applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2):176 – 201, April 1993.

[145] Gregory K. Wallace. *The JPEG Still Picture Compression Standard*, volume 38. International Standards Organisation, February 1992.

[146] Jia-Ping Wang. Stochastic relaxation on partitions with connected components and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-20(6):619–636, June 1998.

[147] W. X. Wang, O. Stephansson, and S. C. Wang. On-line syetem setup in a cellar of a flotation plant. In *15th International Conference on Pattern Recognition*, volume 4 of *Applications, Robotic Systems and Architectures*, pages 791 – 794, Barcelona, 2000.

[148] Xin Wang. Optimal edge-preserving hybrid filters. *IEEE Transactions on Image Processing*, 3(6):862 – 865, November 1994.

[149] Harry Wechsler. *Computational Vision*. Academic Press, 1990.

[150] Mark Allen Weis. *Data Structures and Algorithm Analysis in C++*. Benjamin Cummings, 1994.

[151] Peter D. Wendt, Edward J. Coyle, and Jr Neal C. Gallager. Stack filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-34(4):898–911, August 1986.

[152] D. V. Widder. *The Heat Equation*. Academic Press Inc, 1975.

[153] Allan S. Willsky. *Digital signal processing and control and estimation theory: points of tangency, areas of intersection, and parallel directions*. Number 2 in MIT Press series in optimisation and control. Cambridge MIT Press, 1979.

[154] L Wixson. Detecting salient motion by accumulating directionally-consistent flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):774–780, August 2000.

[155] E. T. Woodburn, L. G. Austin, and J. B. Stockton. A froth based flotation kinetic model. *Transactions of the Institute on Chemical Engineers*, 72(Part A):211 – 226, March 1994.

[156] Benedict Wright. Real time segmentation of flotation froth images. Technical report, University of Cape Town, 1997. Undergraduate thesis. Dept. of Electrical Engineering.

[157] Benedict Wright. The development of a vision-based froth analysis system. Master's thesis, University of Cape Town, 1999. Dept of Electrical Engineering.

[158] *Proceedings ICIP-94*, 1994.

[159] Sohail Zafar and Ya-Qin Zhang. Predictive block matching motion estimation for TV coding – part2: Inter-frame prediction. *IEEE Transactions on Broadcasting*, 37(3):102 – 105, September 1991.

[160] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-21(8):690 – 705, August 1999.