Probabilistic Multiscale Image Segmentation by the Hyperstack

Koen Vincken

Probabilistic Multiscale Image Segmentation by the Hyperstack

Probabilistische meerschalige beeldsegmentatie met behulp van de hyperstack

(met een samenvatting in het Nederlands)

Proefschrift

Ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de Rector Magnificus, Prof. dr. J.A. van Ginkel, ingevolge het besluit van het College van Decanen in het openbaar te verdedigen op woensdag 22 november 1995 des middags te 13:00 uur

door

Koenraad Lucas Vincken

Geboren op 22 augustus 1965 te Heemskerk

Promotor: Prof. dr. ir. M.A. Viergever
Faculteit der Geneeskunde
Universiteit Utrecht

This work was carried out in the framework of the research program "3D Computer Vision", supported by the Netherlands ministries of Education & Science and Economic Affairs through a SPIN grant, and by the industrial companies Philips Medical Systems, KEMA, and Shell Research.

aan Pauline, Jelle
en mijn ouders

Colofon

The two etches on the cover are made by marquiet Westervaarder. The art work at the front is named *Geworteld II* (by the artist referred to as *'geabstraheerde knotwilg'*), and the art work at the back of the cover is named *Hoge bomen*. All other illustrations are included as PostScript[1] files in this thesis, that has been formatted using the LaTeX2$_\varepsilon$ typesetting language.

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Vincken, Koenraad Lucas

Probabilistic multiscale image segmentation by the hyperstack / Koenraad Lucas Vincken. - Utrecht: Universiteit Utrecht, Faculteit der Geneeskunde
Thesis Universiteit Utrecht. - With ref. - With summary in Dutch.
ISBN 90-393-1261-3
Subject headings: hyperstack image segmentation / multiscale image analysis / partial volume artifact.

[1] PostScript is a registered trademark of Adobe Systems Inc.

Contents

Glossary

Throughout this thesis we will use several technical terms. For the sake of clarity we explain the most important terms in the following list. The terminology is given for the 3D case.

active voxel
: a voxel with at least one link to another (parent or child) voxel.

adultness
: a value between 0 and 1 indicating the suitability of a child to become a root.

affection
: a value between 0 and 1 indicating the suitability of a candidate parent to become an actual parent of a child.

blurring
: the low-pass filtering (smoothing) of an image to remove detail.

boundary problem
: the problem of the missing data for kernels crossing the image boundaries.

building step
: the phases of blurring and linking.

child
: a voxel with at least one parent, *i.e.,* a voxel at the next higher level to which it is connected.

child-parent link(age)
: a link from a voxel in level n (the child) to another voxel in level $n+1$ (the parent).

correctness
: a value between 0 and 1 denoting the part of the segmented image that is regarded correct.

downward projection
the phase in the hyperstack segmentation step where the scale-tree is followed downwards from every root to the ground voxels in order to give those voxels a segment value.

forced root
a voxel in scale space that is forced to become a root because of the amount of outer scale reduction.

ground volume of a voxel
the number of voxels at the ground level connected to this voxel.

heuristic linking
a linking method based on heuristic models.

heuristic OSR
outer scale reduction based on a constant scope factor K_n for each level n.

hyperstack
a stack of blurred replicas of an input image at increasing scale connected by child-parent voxel linkages.

inner scale
the smallest entity of an image containing discernable information.

Kuwahara filter
a nonlinear filter based on comparing variances of sub-windows.

linking
the construction of a link between a child in level n and a parent in level $n+1$.

lowest root level
the lowest level in the hyperstack at which roots may be created.

median filter
a nonlinear filter based on ordering the pixel values contained in the window.

multidimensional hashing
a technique with which the interesting elements of multidimensional data can be stored efficiently without loss of the spatial information (coordinates) per data element.

object distribution
a gold standard for evaluating segmentations.

outer scale
the field of view that an image represents.

outer scale reduction (OSR)
the reduction of the dimensions of the images at larger scales in the hyperstack.

parent
a voxel with at least one child, *i.e.,* a voxel at the next lower level to which it is connected.

partial volume artifact
the effect that a voxel represents more than one object, caused by the limited resolution of the imaging device, and therefore has an 'in-between' value.

passive voxel
a voxel with no parents or children.

post-processing editing costs PPE
the costs that have to be made to improve a segmented image up to a certain correctness by manual editing.

probabilistic linking
a linkage scheme in which a voxel can have more than one parent (also called *multi-parent linking*).

probabilistic segmentation
hyperstack segmentation approach with probabilistic linking; leads to a segmentation in which voxels may belong to more than one segment.

probability map
an image where each intensity corresponds to the highest root probability of that voxel.

pyramid
a multiscale image segmentation method based on coarse down sampling and iterative linking.

root probability
the probability (a value between 0 and 1) that a ground voxel belongs to the corresponding root in the scale-tree.

root (voxel)
a voxel in the hyperstack that represents a single segment at the ground level.

scale space
the set of blurred images derived from an input image by convolution with a kernel of increasing width.

search volume
the volume at level $n+1$ in which candidate parent voxels are sought for a child voxel at level n (often a sphere characterized by its radius).

segment
a collection of voxels in the original image that belong together.

segmentation level
the highest level in the hyperstack that is used in the segmentation step.

segmentation step
the phases of root labeling and downward projection.

sibling voxels
voxels having the same parent.

single-parent linking
a linkage scheme in which a voxel can have at most one parent.

single-parent segmentation
segmentation approach in which each active voxel corresponds to exactly one segment.

strict OSR
outer scale reduction based on the scale space theory, such that the ratio #Voxels/Scale remains constant.

volume rendering
a three-dimensional view of a 3D segmented image.

'Whenever man comes up with a better mousetrap,
nature immediately comes up with a better mouse'

– James Carswell

Chapter 1

Introduction and summary

Images. We see them all day, hang them on the wall, and use them to explain unreadable articles. Images (drawings, photographs, movies) can perfectly be used for inter-human communication, thereby often rousing strong emotional feelings, whether they represent war, art or pornography. Apparently, looking at an image causes a lot of information to be processed and subjectively interpreted. Or, as the trite but true cliché says: an image says more than a thousand words.

Our visual system is continuously busy with processing, analyzing, and interpreting of visual stimuli. Nature combined with nurture provides us with a beautiful system to deal with this visual information. In fact, the statement that no autonomous visual system built by men will outperform its biological equivalent in the foreseeable future is upheld by a lot of researchers in the field—including the author of this thesis.

In the medical community, images from different modalities have found their way to a variety of medical disciplines. Multidimensional images have become indispensable in clinical diagnosis, therapy planning and evaluation Clinicians now have the possibility to 'look' into the anatomy of the entire human body, or to investigate the functional properties of most organs or tissues. It depends on, among others, the nature of the disease and the symptoms of the patient, which of the imaging modalities is called for. Besides the well known two-dimensional X-ray images, some of the most widely known and used image acquisition methods to visualize anatomical structures X-ray Computed Tomography (CT), which is pre-eminently suited to image bone against soft tissues; the Magnetic Resonance Imaging (MRI) scan, useful for imaging the brain and other soft tissues; Ultrasound Imaging, used *inter alia* for fetal imaging; Digital Subtraction Angiography (DSA), CT Angiography (CTA), and Magnetic Resonance Angiography (MRA) for imaging of blood vessels. To visualize the functional properties of tissue, Nuclear Science provides Single Photon Emission CT (SPECT) and Positron Emission Tomography (PET) scans, but also the (non-invasive) MR technique can be used (functional MRI). Examples of four modalities are shown in Fig. 1.1.

Images that are processed by the human visual system are of a considerably higher

Fig. 1.1. *Examples of imaging modalities, from left to right: an X-ray photograph of a knee; an abdominal* CT *scan; an* MRI *scan of the brain; a* SPECT *image of the brain.*

resolution than the digital medical images that are dealt with in this thesis (ranging from 64×64 pixels in the case of a 2D image up to $256 \times 256 \times 256$ pixels for a 3D image). Nonetheless, the latter contain a vast and important amount of information for the physicians. Furthermore, the image acquisition techniques are improving every year, resulting in higher resolution images in all three dimensions. An example of this development is spiral CT recording [48, 45], which allows for fast three-dimensional data acquisition of high resolution images.

The introduction of *scale space* theory by Koenderink [54] and Witkin [131] has been a major breakthrough in image understanding. It was not until then that the notion of 'scale' was coupled to a variety of basic concepts, such as the smallest entity of an image containing discernable information (the *inner scale*), and the regularized calculation of an image derivative. Koenderink proved that the unique linear scale space kernel that satisfies the causality constraint is the Gaussian kernel of various widths. Fig. 1.2 shows the MRI image of Fig. 1.1 at four different scales. If the scale space is sampled at a number of successive scales, a *stack* of images is formed. When the input image is 3D, a four-dimensional stack or *hyperstack* is formed, with scale as the fourth dimension.

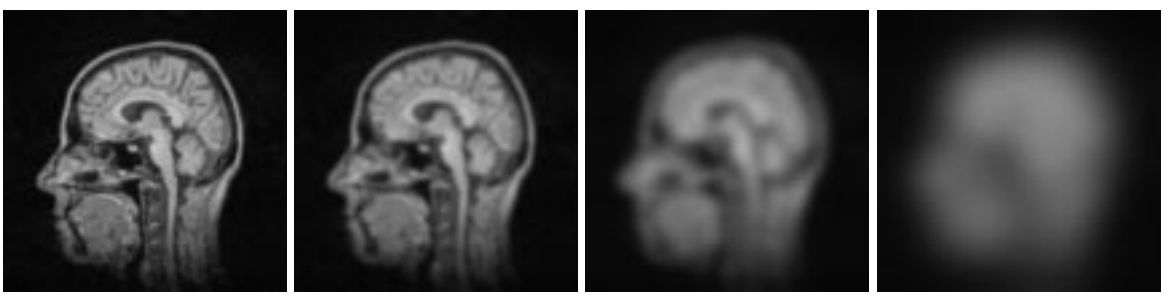


Fig. 1.2. *Four samples of the scale space of the* MRI *brain image of Fig. 1.1*

Scale space theory follows the same constraints as nature. Hence, it is not a coincidence that multiscale image understanding is similar to the human visual system: the Gaussian kernels and their derivatives resemble to a high extent the receptive fields

in the eye that consist of rods and cones [133, 134, 135, 10]. Koenderink's starting point was the *isotropic* diffusion equation, which is in accordance with the absence of information at the stage of early vision (*i.e.,* when no knowledge about the virtual scene is available). The power of the human visual system is that 'observing a scene' (say, looking at a tree) can be followed by a specific, directed action to gather more (other trees) or more specific (branches, leafs) information. In this respect, compare looking at a tree and focusing on a single leaf. Actions like this require some feedback from the brain (that has observed and analyzed the visual scene within milliseconds) to the visual mechanism that controls what we are looking at. Indeed, feedback connections between the visual cortex and the Lateral Geniculate Nucleus (LGN) are physically present, which supports the theory that differential structure in an image (an edge, a corner) can only be seen *after* the image has been analyzed at multiple levels of isotropic scale.

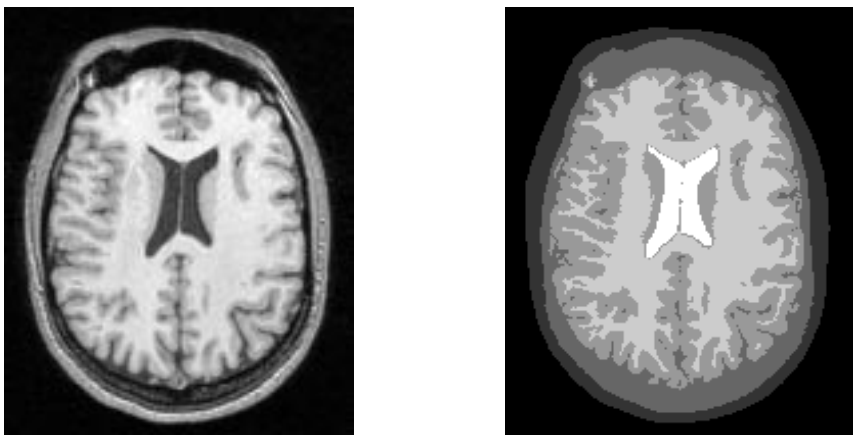


Fig. 1.3. *An* MRI *brain image (left); a segmentation into 6 segments (right). From inside out: ventricles, white brain matter, grey brain matter, liquor & bone, connective tissue & skin, and the background.*

In the pipeline from image acquisition to understanding one of the most important and difficult tasks is the *segmentation* of an image. That is, dividing the picture elements (pixels) in larger, basic structures, such that the grouping results in a meaningful distribution of objects. This is illustrated in Fig. 1.3. Without segmented versions of an image one is virtually not capable to perform quantitative measurements on an image (*e.g.,* calculate the volume of a tumor), or to create a three-dimensional view (*e.g.,* a *volume rendering* of a skull). Fig. 1.4 shows two examples of volume renderings of a segmented 3D MR image. The challenge here is that the biological visual system of higher species performs the extremely difficult task of segmentation with such ease, that segmentation in computer vision should try to reach the same speed and accuracy. In our opinion, a method derived from biological vision—like the hyperstack segmentation method—has the best chances of accomplishing this goal. The availability of 3D imaging creates at least one advantage with respect to biologi-

cal vision. The latter can obtain 2D input images only, which must be reconstructed to a 3D scene in the brain.

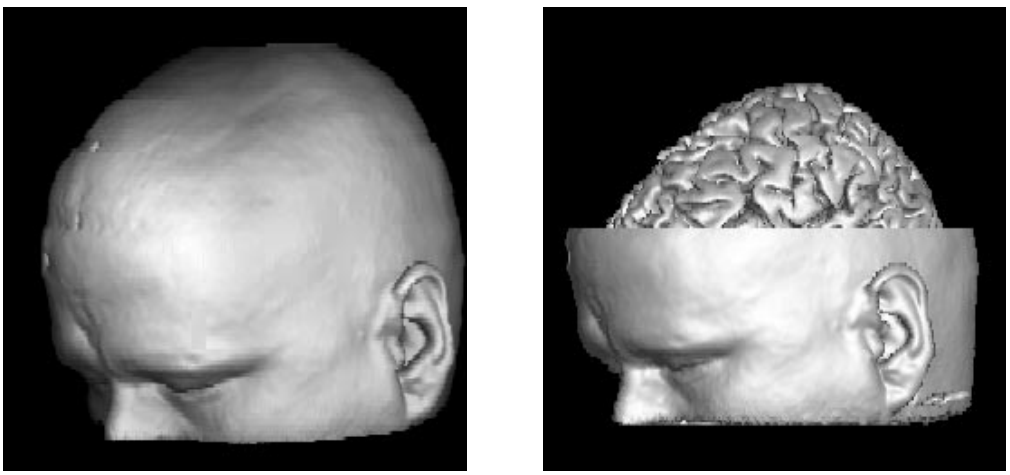


Fig. 1.4. *Volume rendering of a 3D* MRI *brain image (left): if the 3D image is properly segmented, parts of the outmost objects (in this case the skin and the skull) can be easily 'removed' to show hidden objects, such as the cortex of the brain (right).*

Most of the conventional segmentation methods are not based on a scale space representation, but are more locally oriented—like the rule-based systems of Ortendahl [80], Menhardt [74, 75], and Raya [91]—or *ad hoc* implementations for a specific segmentation problem (*e.g.,* Brummer [15]). Schiemann [102] and Pizer [85] have focused on interactive rather than automatic segmentation methods, while Karssemeijer [49] and Vemuri [115] used a multiresolution approach that was not based on the diffusion equation, but on median filtering and the wavelet transform [70], respectively. Finally, promising results have been obtained by using neural networks for image segmentation (*e.g.,* Worth [132]), especially in combination with multiscale techniques (Haring [44]).

Most approaches to multiscale image segmentation involve connecting pixels in adjacent levels of the sampled scale space (Burt, Pizer, De Graaf [16, 86, 25]). By using this multiscale approach the global image information can effectively be included. In this thesis it is shown that the hyperstack performs well on noisy images and is particularly strong in its ability to segment both small and large objects simultaneously.

In Chapter 2 the fundamentals of the hyperstack segmentation method are explained. This includes the blurring process to sample the scale space, and the bottom-up linking process in which the scale space levels are linked to one another pixelwise. The latter results in a tree of linkages in scale space. From each node, the tree can be followed downwards to the pixels of the original (high resolution) image to constitute

a segment. The node from which the downward projection starts is appropriately referred to as the *root* of the segment. A critical step in hyperstack segmentation is to determine which of the scale-tree nodes are likely to represent a segment in the original image. This is the root labeling phase. Chapter 2 also features aspects of the design of the hyperstack method. The data structure has been implemented in the object-oriented programming language C^{++}.

In order to test our segmentation algorithms, it is desirable to have artificial (test) images of which the object distribution is known *a priori*. To this end, an octree-like algorithm has been developed to convert mathematically defined objects to a voxel-based model (a *voxel*, from 'volume element', is a 3D pixel). In essence, the coarse discretization at voxel level can be refined by a subdivision of each voxel that represents part of the edge of an object into eight smaller sub-voxels. This allows for proper representation of the partial volume voxels. The conversion algorithm has been recursively implemented, which makes the method flexible in accuracy. The method is described in detail in Chapter 3.

A major problem with most of the conventional segmentation methods is that they fail to take the partial volume artifact into account. This effect—caused by the limited resolution of the imaging device—is reflected by pixels that have an 'in-between' value (the pixel intensity) with respect to neighboring pixels, that are fully contained in one tissue. Normally, partial volume pixels are forced to choose between either tissue type. By following the isophote structure in the neighborhood of a partial volume pixel, it is possible to extract more detailed information on the different tissue types comprised in that pixel. In the hyperstack, this is effectuated by allowing multiple upward linkages to be established for each pixel (instead of just one as in the conventional linking scheme). In Chapter 4 we explain how the multiscale information (containing the isophote structures) can effectively be used to segment partial volume pixels at sub-pixel level. It is shown that this yields better results with respect to quantitative image analysis (*e.g.,* volumetric measurement), and that the volume renderings of 3D images can be significantly improved.

In Chapter 5 we deal with the sampling of the blurred images in scale space. Since high frequency information vanishes with increasing scale, the images at the larger scales are oversampled. It is shown why a straightforward sampling rate reduction fails in multiscale image analysis. Instead, the decrease of the number of samples should *not* affect the sampling width. This leads to *outer scale reduction* (OSR), which is effectively a decrease of the image boundaries if scale increases. Two different approaches are discussed: strict OSR and heuristic OSR. The profit of applying OSR is twofold: (*i*) the influence of the boundary problem (which is especially apparent at larger scales) is minimized, and (*ii*) the complexity of the linkage structure is decreased. The segmentations of medical images based on hyperstacks with OSR illustrate that the computation can be sped up by a factor of 5, without significant loss in quality.

Finally, in Chapter 6 we deal with a topic that has an increasing interest from the

computer vision world. Since the introduction of linear scale space, researchers have focused on nonlinear variants. As a result, several nonlinear blurring schemes have been proposed, *e.g.,* by Perona [84], Alvarez [2], and Niessen [78]. We have compared the applicability of such schemes with linear scale spaces and two other nonlinear blurring filters: the median filter and the Kuwahara filter. The different blurring strategies are evaluated by the hyperstack segmentation method: the blurring strategies are taken as the first step of the method. On the basis of numerical experiments, it is shown that the multiscale linking model is very robust against the choice of the scale space. The nonlinear algorithms based on the Perona & Malik equation and the Euclidean shortening flow appear to be most promising for research in the near future.

'Study the family as a family, (...)'

– Jan Koenderink
(The structure of images [54])

Chapter 2

Design of the hyperstack

Abstract

In this chapter the design of the hyperstack, a multiscale method for segmentation of multidimensional images, is presented. Hyperstack segmentation is based on the linking of voxels at adjacent levels in scale space, followed by a root selection to find the voxels that represent the segments in the original image. This chapter addresses an advanced linking and root labeling method for the hyperstack.

We present an efficient data structure (based on so-called *containers*) for storing the linkages. Furthermore, we introduce a new technique (*multidimensional hashing*) for the problem of storing sparse multidimensional data structures. A comparison with conventional hashing techniques will be made.

Finally, segmentations obtained from 2D and 3D images are presented and compared to show the surplus value of 3D versus multiple 2D image processing.

Keywords: Image segmentation, scale space, containers, hashing.

2.1 Introduction

Image segmentation—dividing an image into meaningful objects—is a subfield of image processing that is of crucial importance for quantitative analysis and, in the case of 3D images, volume visualization (*e.g.,* see [67, 27, 63, 87, 9, 139, 138]).

We present an approach to segmentation of multidimensional images called the *hyperstack* (in analogy with the name *stack* for its 2D equivalent). The ideas underlying hyperstack segmentation are the scale space representation of images [131, 54] and the hierarchical approach to image segmentation by means of the pyramid [16, 46, 24, 23, 73, 11]. Previous descriptions of scale space based image segmentation include notably the stack [86, 64] for 2D images, and earlier versions of the present work [116, 118, 57]. In this thesis we focus on discrete scale spaces rather than on the continuous case (see [66], *e.g.,* for details).

In the conventional pyramid approach the discrete levels of the multiresolution structure are derived by a repeated averaging of the samples of the input signal, while

the sampling rate is reduced by a factor of two in each dimension. The thus derived levels are generally strongly undersampled—both in space and in scale—owing to this block-wise algorithm.

A major advantage of the pyramid approach is its computing speed. The total number of voxels used for the discrete scale space representation equals approximately $(2^d \cdot N)/(2^d - 1)$, where N denotes the total number of voxels of the d-dimensional input image. The larger d, the more this term approximates N. (For instance, a pyramid of a 3D image requires only an additional $0.14 \cdot N$ voxels.) Pyramids are often used in applications where it is desirable to have a low computational effort at the cost of a less accurate result.

A different approach to scale space was introduced by the *wavelet transform* [70], although up to now wavelets have not been applied to image segmentation. The main reason for this is that it is not clear how the *difference* signal, which is the result of a wavelet decomposition step, can effectively be used to segment an image.

The hyperstack algorithm consists of four main steps, which are discussed in detail in section 2.2. First, a *blurring* step is used to create replicas of the original image at increasingly larger scale. The natural kernel for building such a discrete scale space is the Gaussian [7]. This is the only linear convolution function that does not create image structure when blurring [54] and that provides invariance for position, orientation and scale [41, 30, 31, 33]. Moreover, the biological visual system is known to employ a similar low-pass filtering [56, 134, 137]. The blurred images may be conceived as placed on top of the input image, so as to create a 'stack'. For 3D images we obtain a four-dimensional scale space (hence the name *hyperstack*). See Fig. 2.1 for a schematic of a hyperstack.

The next step is a *linking* step. Voxels in adjacent levels are linked according to some 'suitability' criterion. The connections are called 'parent-child' linkages, where the parent is located at the level of largest scale. In its simplest form, the linking strategy may be based on minimum intensity differences between parent and child. Such a simple strategy, however, has turned out to be insufficient for segmentation of complex images [57, 118, 58]. Similar experiences with the extremum stack [64] showed also that linkages could 'escape' from the isophote umbrella covering all the light and dark blobs in an image. As will be shown, the hyperstack is based on a better and more robust linking strategy.

The tree-like linkage structure which results from the building (blurring and linking) phase is used to produce a segmentation. The segmentation phase also consists of two steps: *root labeling*, followed by a *downward projection*. The root labeling defines which voxels in the discrete scale space represent a single segment in the ground level. Segmented images are then obtained by projection of some *segment value* from every root downwards to the connected voxels in the ground level. Different choices for the segment values will be discussed.

Section 2.3 describes the design of the hyperstack data structure. We explain what objects need to be discriminated and how they are mutually related (section 2.3.1).

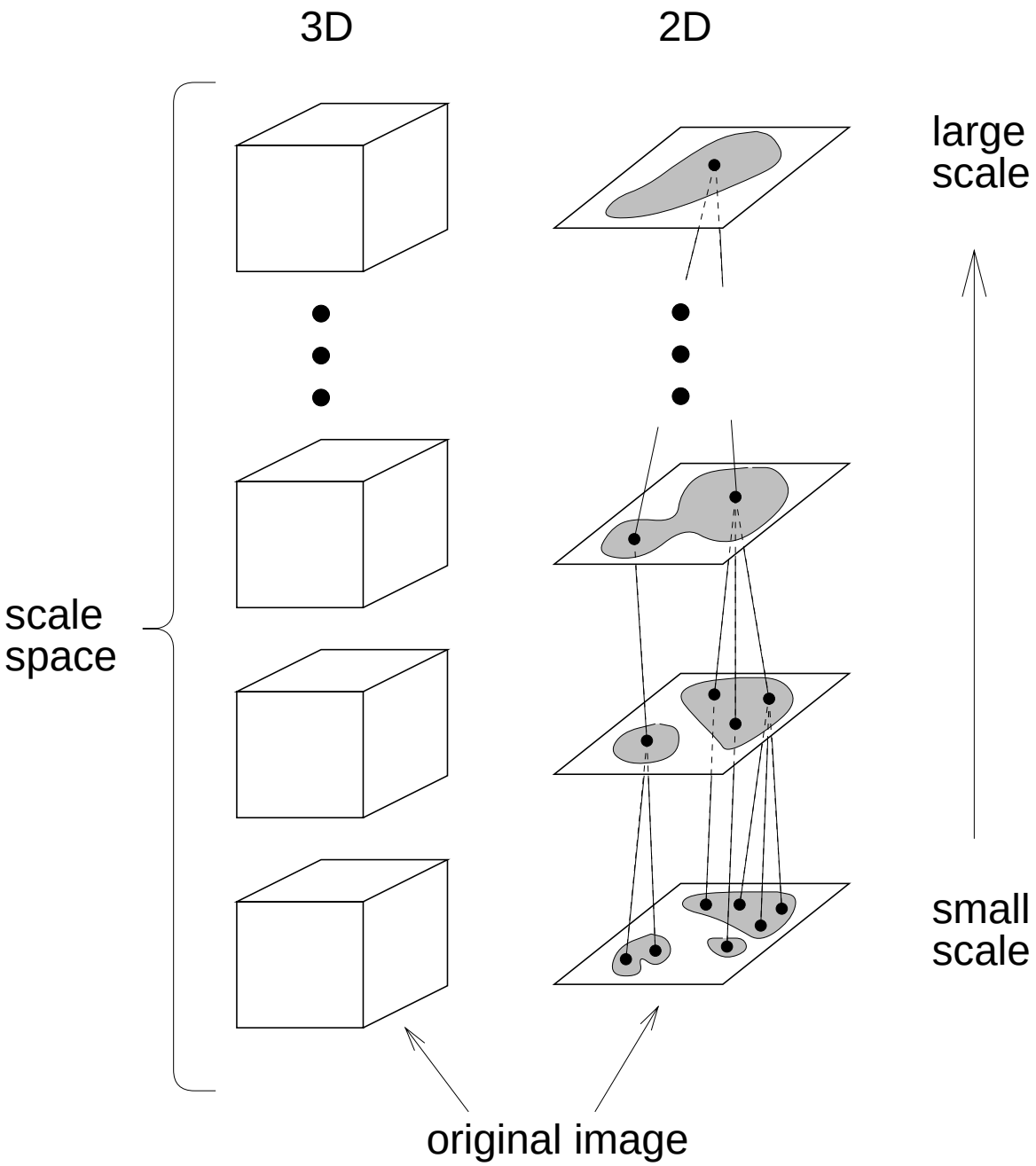


Fig. 2.1. *Schematic of a hyperstack in 3D (left) and 2D (right).*

The design is kept very flexible so as to allow future extensions. In particular, provisions have been made for proper segmentation of partial volume voxels by *probabilistic linking* schemes.

We also introduce the notion of *containers* to store the linkages involved (section 2.3.2).

In section 2.4 we discuss a new storage technique, called *multidimensional hashing*, that can effectively be employed to decrease the amount of memory needed to store large sparse arrays without loss of the multidimensional indexing capability. In the hyperstack this multidimensional hashing technique is used to deal with the problem of the sparseness of the higher levels: the number of interesting (or *active*) voxels is decreasing rapidly because of the convergence of the linking process.

A lot of attention has been paid to full—*i.e.,* three-dimensional—processing of 3D images. In section 2.5.1 we show the surplus value of including the third (spatial) dimension in the segmentation process, rather than applying a series of two-dimensional algorithms (often called $2\frac{1}{2}$D segmentation). We will also show that a hybrid variant that makes full use of the three-dimensional spatial information to generate a 3D scale space, but processes the data on a slice-by-slice basis (the so-called $2\frac{3}{4}$D variant) is not an attractive alternative. Results of an extensive study will be presented here.

Finally, the results of hyperstack segmentation of both a 2D and a 3D real world (medical) image are presented in section 2.5.

2.2 The hyperstack

The essence of the hyperstack image segmentation method is illustrated in Fig. 2.2.

The method consists of four consecutive steps: (*i*) *blurring*, building the image's scale space, (*ii*) *linking*, establishing the connections between voxels at adjacent scale levels, (*iii*) *root labeling*, defining which voxels in scale space represent an entire segment, and (*iv*) *downward projection*, forming the segments in the original image. We will now deal with the separate steps in detail.

2.2.1 Blurring

Applying a blurring algorithm to a (discrete) input image has the effect of a low-pass filter: low frequencies are preserved, while the higher frequencies (the details) are smoothed out. For front-end vision systems, *i.e.,* systems that have no knowledge about the observed scene, it has been shown [54, 33] that the unique linear scale space constructor is the Gaussian function $G(\vec{x};\,\sigma)$:

$$G(\vec{x};\,\sigma) \;=\; \frac{1}{(\sigma\,\sqrt{2\pi})^d}\,\exp\left(-\frac{\|\vec{x}\|^2}{2\,\sigma^2}\right) \quad , \tag{2.1}$$

where d denotes the dimension of the input domain. A blurred replica of the original image is obtained by convolution with $G(\vec{x};\,\sigma)$ for a specific σ. If the intrinsic scale of

the input image is ε (often chosen to be 1), then the scale of the blurred image equals $\sqrt{\sigma^2 + \varepsilon^2}$. The convolution may be effectuated either in the spatial domain or in the frequency domain (by multiplication of the Fourier transforms). Both methods can be used to derive the scale space of an image; they differ mainly in computation time (see below).

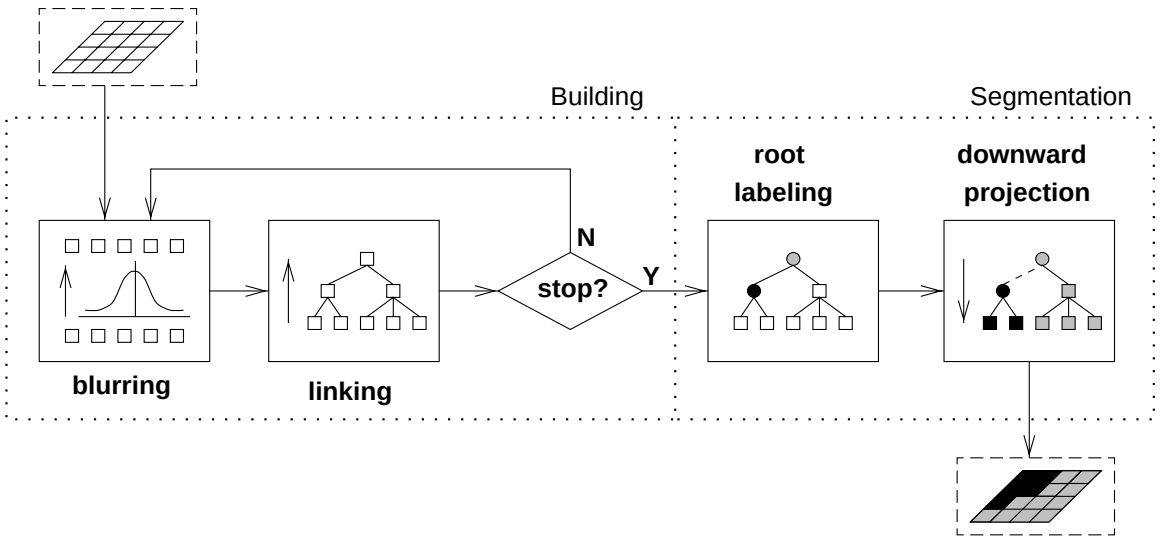


Fig. 2.2. *Schematic overview of the hyperstack image segmentation process.*

In the discrete implementation we make use of the separability of the Gaussian. Hence, a sampled 1D Gaussian $\widehat{G}[n;\sigma]$ can be used:

$$\widehat{G}[n;\sigma] \;=\; \frac{1}{N}\,\exp\left(-\frac{n^2}{2\,\sigma^2}\right) \quad , \quad n \;=\; 0, \pm 1, \pm 2, ..., \pm\lfloor K_n\sigma \rfloor \quad , \tag{2.2}$$

where N is a normalization factor such that

$$\sum_{n\,=\,-\lfloor K_n\sigma \rfloor}^{\lfloor K_n\sigma \rfloor} \widehat{G}[n;\sigma] \;=\; 1 \quad . \tag{2.3}$$

The factor K_n specifies the *accuracy* of the implemented Gaussian kernel: the larger K_n, the larger the scope of the discrete kernel, and hence the accuracy of the calculated blurred values (see also [118]). In practice, K_n is often given the same (constant) value for each level n, although the accuracy of the convolution decreases if the number of dimensions increases. This follows directly from Fig. 2.3, where the percentages of the definite integral of a Gaussian function in d dimensions have been plotted. For instance, for the three-dimensional case spherical coordinates can be used to compute the volume V:

$$V = \frac{1}{\left(\sigma\,\sqrt{2\pi}\right)^3} \int_0^{2\pi} d\phi \int_0^{\pi} \sin\theta\, d\theta \int_0^{K_n\sigma} \rho^2 \exp\left(-\frac{\rho^2}{2\sigma^2}\right) d\rho \quad , \tag{2.4}$$

for which we can write

$$V = \mathrm{erf}\left(\frac{K_n}{\sqrt{2}}\right) \; - \; \sqrt{\frac{2}{\pi}}\, K_n \exp\left(-\frac{K_n^2}{2}\right) \quad . \tag{2.5}$$

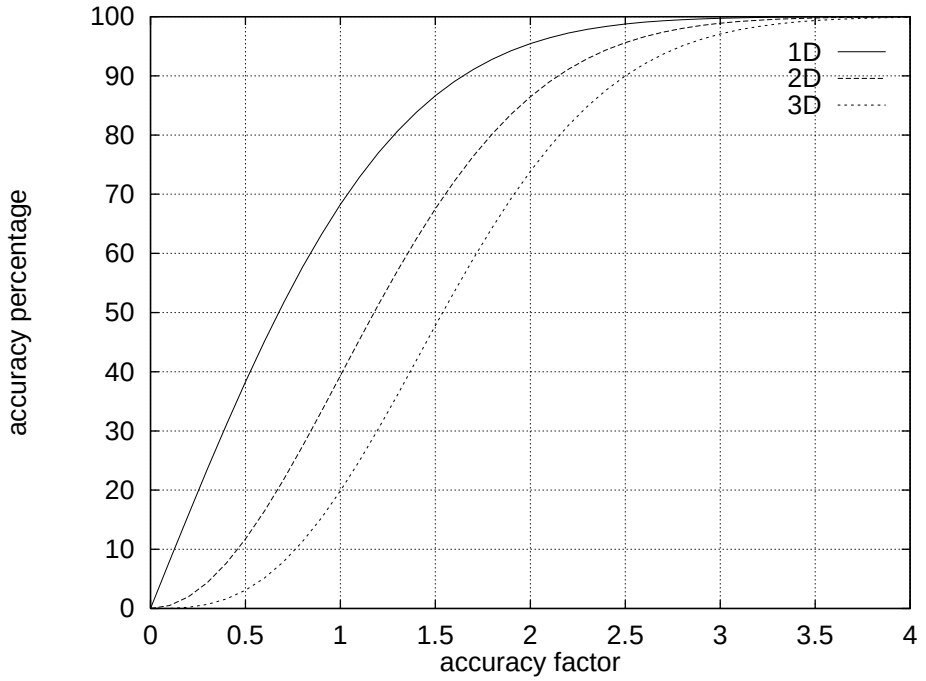


Fig. 2.3. *Accuracy percentages of the convolution of a d-dimensional image with a Gaussian kernel. The scope of the spatial kernel is defined by a radius $K_n\sigma$, where K_n is the accuracy factor.*

The convolution in d dimensions is performed by repeated 1D convolutions of the image with $\widehat{G}[n;\sigma]$ in each spatial direction (see section 6.B.1 for details).

(Note that, from a mathematical point of view, the conventional pyramid kernel is also a discretized Gaussian, albeit with a very limited accuracy. It encompasses two voxels in each spatial direction, and there is no overlap between the domains of the kernels for adjacent pixels or voxels. Because of the normalization of the coefficients of the kernel this amounts to simply averaging.)

In the frequency domain the discrete Fourier transform of the spatial Gaussian kernel of (2.2) is used:

$$\mathcal{G}[\omega;\sigma] \; = \; \exp\left(\frac{-\sigma^2\omega^2}{2}\right) \tag{2.6}$$

Scale space sampling

The sampling in the scale direction of the generated scale space should follow a linear and dimensionless scale parameter $\delta\tau$ [33], which is related to the absolute (or

effective) scale σ_n of level n by:

$$\sigma_n \;=\; \varepsilon\, e^{\tau_0 + n\cdot\delta\tau}\;, n \;\in\; \mathbb{N}\;\;, \tag{2.7}$$

where ε is taken to be the smallest linear grid measure of the imaging device, and τ_0 defines the initial offset in scale. A convenient choice for τ_0 is 0, which implies that the *inner scale* [54] σ_0 of the initial image is taken to be equal to the linear grid measure ε.

2.2.2 Linking

The search volume

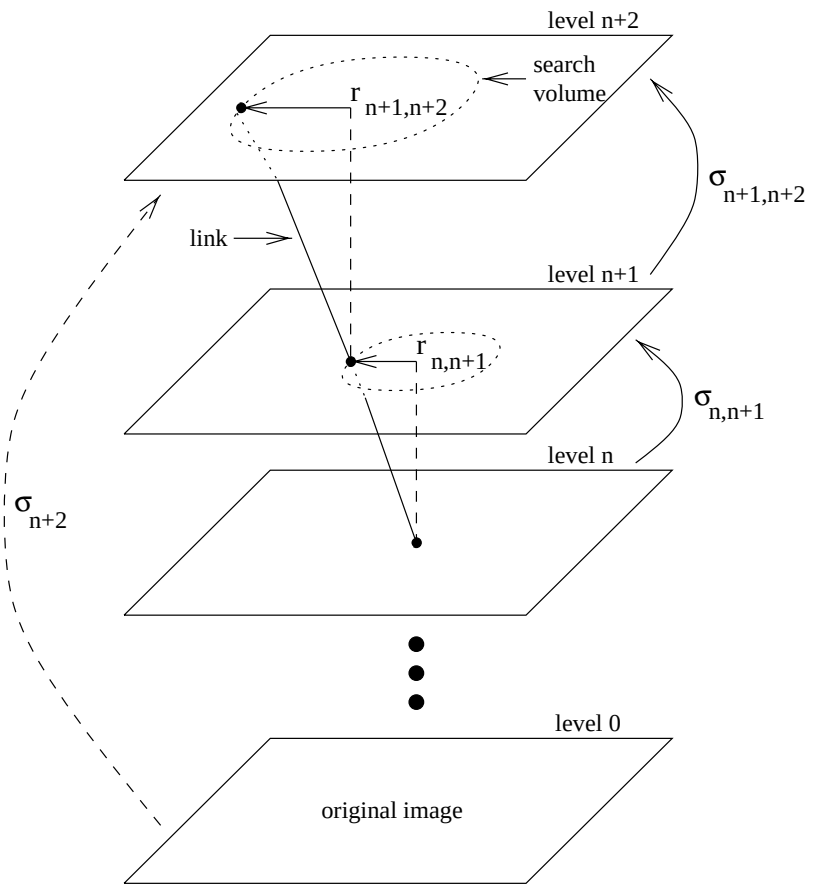


Fig. 2.4. *Search volumes in the linking process. For reasons of simplicity, the schematic is drawn for 2D images.*

The bottom-up linking process creates linkages between voxels in two adjacent levels, say n and $n+1$. For each active voxel at level n a parent is sought in a search volume defined at level $n+1$. (Note that only the *active* voxels—*i.e.,* voxels with at

least one child or parent reference—are linked upwards again. In combination with the blurring, this ensures convergence of the hyperstack linking process; the top level contains exactly one—active—voxel.) Since rotation invariance of the segmentation method with respect to the input image is desirable, the search volume is defined as a spherical volume in 3D, and as a circle in 2D (see Fig. 2.4). The radius $r_{n,n+1}$ of the search volume depends on the difference in scale between the child level n and the parent level $n+1$, as defined by the relative σ between these levels:

$$r_{n,n+1} \;=\; k_n \cdot \sigma_{n,n+1} \;\;,\; k_n \;\geq\; k_{n,min} \;\;. \tag{2.8}$$

For simplicity, k_n is often given a constant value throughout the scale space, *i.e.,* independent of the level n. The reason that k_n has a minimum ($k_{n,min}$) is that we cannot allow a search volume with a radius smaller than the inner scale. The value that is assigned to k_n is a compromise between accuracy and computational complexity. An acceptable value from either point of view has experimentally been found to be $k_n = 1.5$, independent of the image dimension d. See appendix 2.A for more background on the calculation of $k_{n,min}$ and the choice of k_n.

Linkage criteria

For each child a suitable parent is sought in a limited domain. In keeping with the way the images are blurred the attractiveness (or *affection*) decreases with the distance between the potential parent and the child. The dependence of the affection on the child-parent distance is based on the Gaussian shaped function

$$D(d_{c,p}) \;=\; \exp\!\left(-\frac{{d_{c,p}}^2}{2({\sigma_p}^2 - {\sigma_c}^2)}\right) \;\;, \tag{2.9}$$

with $d_{c,p} = ||\vec{x}_p - \vec{x}_c||$, $\vec{x}_c$ and $\vec{x}_p$ being the spatial positions of the child and the parent, and σ_c and σ_p the scales of the child and parent levels.

Every image at each level has an inner scale. Within this inner scale no distinction of similar features is possible. Consequently, the child-parent distance has a minimum, which is set to half the scale of the parent image. We define the distance factor $\mathcal{D}$ as:

$$\mathcal{D} \;=\; \begin{cases} 1 & \text{if } d_{c,p} \leq 0.5\ \sigma_p \\ \frac{D(d_{c,p})}{D(0.5\ \sigma_p)} & \text{if } d_{c,p} > 0.5\ \sigma_p \end{cases} \tag{2.10}$$

The parents are selected on the basis of their affection (♡), or linkage strength, to a given child. The candidate parent with the highest affection value is selected to become the child's parent. This affection is defined as:

$$\heartsuit \;=\; \mathcal{D} \cdot \frac{\sum_{i=1}^{N} w_i \cdot C_i}{\sum_{i=1}^{N} w_i} \;\;,\;\; \text{with } C_i \in [0,1] \;\;. \tag{2.11}$$

The C_i are individual linking components, controlled by weights w_i.

Research into different statistical and heuristic components has shown that a general and robust linking scheme should use three components [58]: (*i*) the traditional intensity difference component $C_{\mathcal{I}}$, (*ii*) the ground volume component $C_{\mathcal{G}}$, and (*iii*) the ground volume mean intensity component $C_{\mathcal{M}}$.

The first affection term is based on intensity proximity and provides intensity following through scale-space. It is defined by:

$$C_{\mathcal{I}} \;=\; 1 \;-\; \frac{|\mathcal{I}_p \;-\; \mathcal{I}_c|}{\Delta\mathcal{I}_{max}} \quad , \tag{2.12}$$

where $\mathcal{I}_p$ and $\mathcal{I}_c$ denote the intensity of the parent and the child, respectively, and $\Delta\mathcal{I}_{max}$ is the maximum intensity difference in the original image.

The second affection term, the *ground volume* of a parent, encourages convergence of the linkages to ever fewer parents. The ground volume is the number of voxels to which a parent is connected at the bottom level, the original image. The ground volume affection term is defined by:

$$C_{\mathcal{G}} \;=\; \frac{\mathcal{G}_p}{\mathcal{G}_{max}} \;, \tag{2.13}$$

with $\mathcal{G}_p$ the ground volume of the parent and $\mathcal{G}_{max}$ the maximum ground volume at the parent's level. Initially, no parents are linked, and hence no ground volumes exist. This problem is solved using an iteration procedure. The linking process is done first without taking the ground volume term into account (by assigning it a weight of 0), followed by repeating the linking process a number of times, each time with a slightly higher weight of $w_{\mathcal{G}}$. In general, no significant changes occur after three or more iterations.

The third affection component is the ground volume mean intensity component $C_{\mathcal{M}}$:

$$C_{\mathcal{M}} \;=\; 1 \;-\; \frac{|\mathcal{M}_p \;-\; \mathcal{M}_c|}{\Delta\mathcal{I}_{max}} \quad , \tag{2.14}$$

with $\mathcal{M}_p$ and $\mathcal{M}_c$ the ground volume mean intensity of the parent and the child, respectively. If the ground volume mean intensity of the child closely resembles the corresponding component of the parent, then it is natural for the child to merge into that parent segment: they both represent (part of) a segment with such an intensity value.

Obviously, the linkages formed heavily depend on the local image structure and the relative weights of the affection components. Experiments with different images showed that general, robust values can be calculated. Typically, the corresponding weight factors $w_{\mathcal{I}}$, $w_{\mathcal{G}}$, and $w_{\mathcal{M}}$ take the values of 1, 10^{-7}, and 1000, respectively. The domain where parents are sought is in effect limited by the distance factor $\mathcal{D}$ of equation (2.10). Given a highest affection $\heartsuit_{max}$ found within a certain radius around

the spatial location of the child, no higher affection can be found at (larger) distances where $\heartsuit_{max}/\mathcal{D} > 1$.

In section 2.5.5 an example will be presented to show that linking based solely on the intensity difference component $C_{\mathcal{I}}$ performs significantly worse than the extended linking scheme presented above.

Probabilistic linking

In the linking process described above every ground voxel has a unique path going up. Hence, every ground voxel is related to exactly one root. The resulting hyperstack is named *single-parent* for this reason; the thus segmented images are called 'single-parent segmentations'.

An important extension to this linking scheme is *probabilistic* or *multi-parent* linking. Here, a child is linked to every candidate parent with a sufficiently high affection value (according to some criterion, see [121] for details). The affection values are normalized by requiring that the sum of all affections of a child equals 1. Accordingly, the linkages to the parents can be regarded as *probabilities*. The sum of all normalized affections between two adjacent levels equals the number of active voxels in the child level, both in single-parent and multi-parent hyperstacks.

Probabilistic linking influences in particular the linking of partial volume voxels and of voxels with a high noise content. These voxels are not forced to choose for one particular segment, but are instead segmented at a sub-voxel level. Furthermore, probabilistic linking improves the quality of three-dimensional image segmentations—especially with respect to visualization of the results [121, 123, 120].

The result of a hyperstack based on multi-parent linking is a list of (normalized) probabilities for each (partial volume) voxel specifying the chance that it belongs to different segments.

Note that at the higher levels of the hyperstack, the profit of multi-parent linking is limited [121]. The interesting partial volume voxels have already been decomposed, so multi-parent linking can be turned off if the building proceeds.

2.2.3 Root labeling

The segmentation phase consists of two steps: root labeling and downward projection. To start with, roots are formed by selecting the *segmentation level*. Linkages from the segmentation level upwards are either not present (in the case that the segmentation level is the top level of the hyperstack), or ignored. Thus, all active voxels in the segmentation level become root. The default is to use the top level as segmentation level.

Additionally, the *lowest root level* may be set. This defines the lowest level in the hyperstack where roots are allowed to be created. The default is set to level 1. The combination 'segmentation level/lowest root level' controls *a priori* the size of the resulting segments in a global sense.

Additional roots are formed by voxels of which the linkages to the parents in the next-higher level are too weak to be followed. The 'weakness' measure, or *adultness*, is modeled in much the same way as the affection:

$$\mathcal{A} \;=\; \frac{\sum_{i=1}^{N} \widehat{w}_i \cdot \widehat{C}_i}{\sum_{i=1}^{N} \widehat{w}_i} \quad , \quad \text{with } \widehat{C}_i \in [0,1] \quad . \tag{2.15}$$

A minimum adultness $\mathcal{A}$ may be set that is needed for a child to be labeled as root, depending on properties like intensity proximity, ground volume, and ground volume intensity variance. A first account of heuristic root labeling criteria is given in [58].

The two components that turn out to be robust root criteria are: (*i*) the ground volume mean intensity difference component $\widehat{C}_{\mathcal{M}}$, and (*ii*) the ground volume component $\widehat{C}_{\mathcal{G}}$. They are defined as:

$$\widehat{C}_{\mathcal{M}} \;=\; \frac{|\mathcal{M}_p - \mathcal{M}_c|}{\Delta\mathcal{I}_{max}} \quad , \tag{2.16}$$

and

$$\widehat{C}_{\mathcal{G}} \;=\; \frac{\mathcal{G}_c}{\mathcal{G}_{max}} \quad , \tag{2.17}$$

respectively. For the $\widehat{C}_{\mathcal{M}}$ component it is plausible that a relatively large value indicates a *catastrophe* in scale space [54]. At such an event, two different objects merge together. Furthermore, large objects are preferred over smaller ones by the $\widehat{C}_{\mathcal{G}}$ component. The relative weight factors $\widehat{w}_{\mathcal{M}}$ and $\widehat{w}_{\mathcal{G}}$ turned out to be both 1.

Instead of using a lower bound for the adultness value, the total number of roots ($\mathcal{R}$) may be prescribed. Then, the best voxels (*i.e.,* those with the largest adultness value) are labeled root, up to the desired number $\mathcal{R}$.

The advantage of the first method is that it is fast: by one thresholding operation all roots will be found (if an adultness is larger than the threshold value, the corresponding child voxel is labeled root). A disadvantage is that it is not clear beforehand how many roots—and thus how many segments—will be found. Statistics on the adultness values may help, but it is likely that at least a few threshold values have to be tried before the number of segments is satisfactory.

The second method does not use a trial and error method, since the number of roots is fixed. Then, the entire linkage structure is scanned $\mathcal{R}$ times. Each time, the child of the child-parent link with the highest adultness value in the hyperstack is labeled root. For efficiency reasons, it may be useful to maintain a (sorted) list of candidate roots during the search to the first root.

2.2.4 Downward projection

Upon having found the appropriate roots, we need to relate them to the ground voxels by following the linkages downwards. Each root represents a single segment in the

ground level. The term 'downward projection' is somewhat misleading, because the process might just as well be implemented as a bottom-up process. The choice depends on how the parent-child linkages are incorporated in the data structure: top-down or bottom-up. In section 2.3.2 we will deal with this issue and compare the alternatives.

The value assigned to each segment is either a unique value per segment—which offers the possibility to discriminate between every two segments—or may be a function of the grey values. In the latter case, the root intensities can be used, or the average value of all ground voxels comprised in that segment. Since the global maximum and minimum tend to converge to each other at increasing scale, it is to be expected that root values also approximate an average value. Obviously, this has a negative effect on the contrast of a segmentation. Thus, average intensities calculated on the basis of original grey values are preferred—although this requires an additional computing step. Both the use of root values and average intensities suffer from the problem that different segments may be assigned the same value. Then, discriminating between them is impossible. Only the use of unique segment values guarantees a unique '(pseudo)color' per segment.

In the case of probabilistic hyperstacks there is not always a unique path from ground voxel to root. Then, the probabilities involved have to be multiplied to obtain the final root probabilities for every ground voxel. Ultimately, the root probabilities of a ground voxel represent the fractions of tissue types comprised in that voxel (partial volume effect).

2.3 The data structure

This section describes how the different objects of the hyperstack (levels, voxels, linkages, roots) are related to each other. Furthermore, the notion of *containers* is

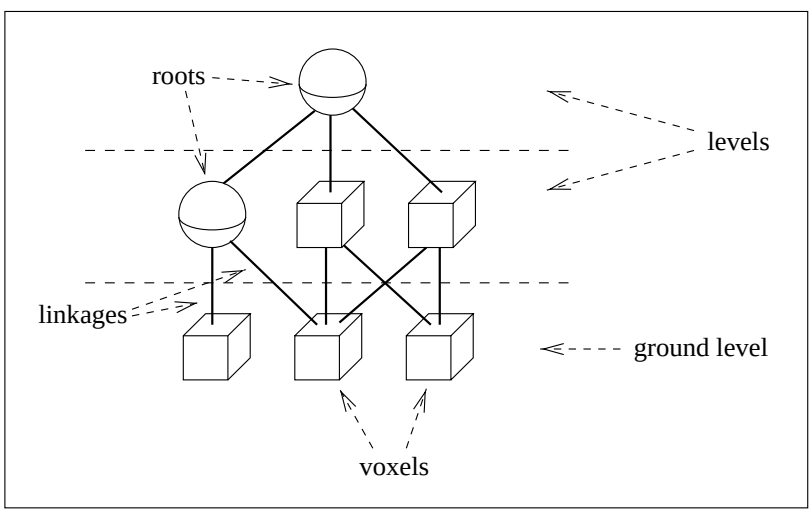


Fig. 2.5. *Schematic picture of the different hyperstack objects and their relationships.*

introduced, and we will compare three different type of linkage structures: top-down, bottom-up, and doubly linked lists.

2.3.1 Hyperstack objects

In Fig. 2.5 a hyperstack and the different objects contained in it are depicted. Each level contains voxels (only active voxels are relevant) and linkages, the latter by means of a *link container* (see 2.3.2) to simplify the maintenance of the linkages. In the case of probabilistic linking, the number of linkages and the number of active voxels may differ per level, hence the objects 'voxels' and 'linkages' can best be separated internally.

Every voxel has a *type* assigned to it. Possible types are *active*, *passive*, and *root*. The relationships of Fig. 2.5 have been translated to *classes* for implementation in the C^{++} programming language [111]. Fig. 2.6 contains an overview of the most important classes.

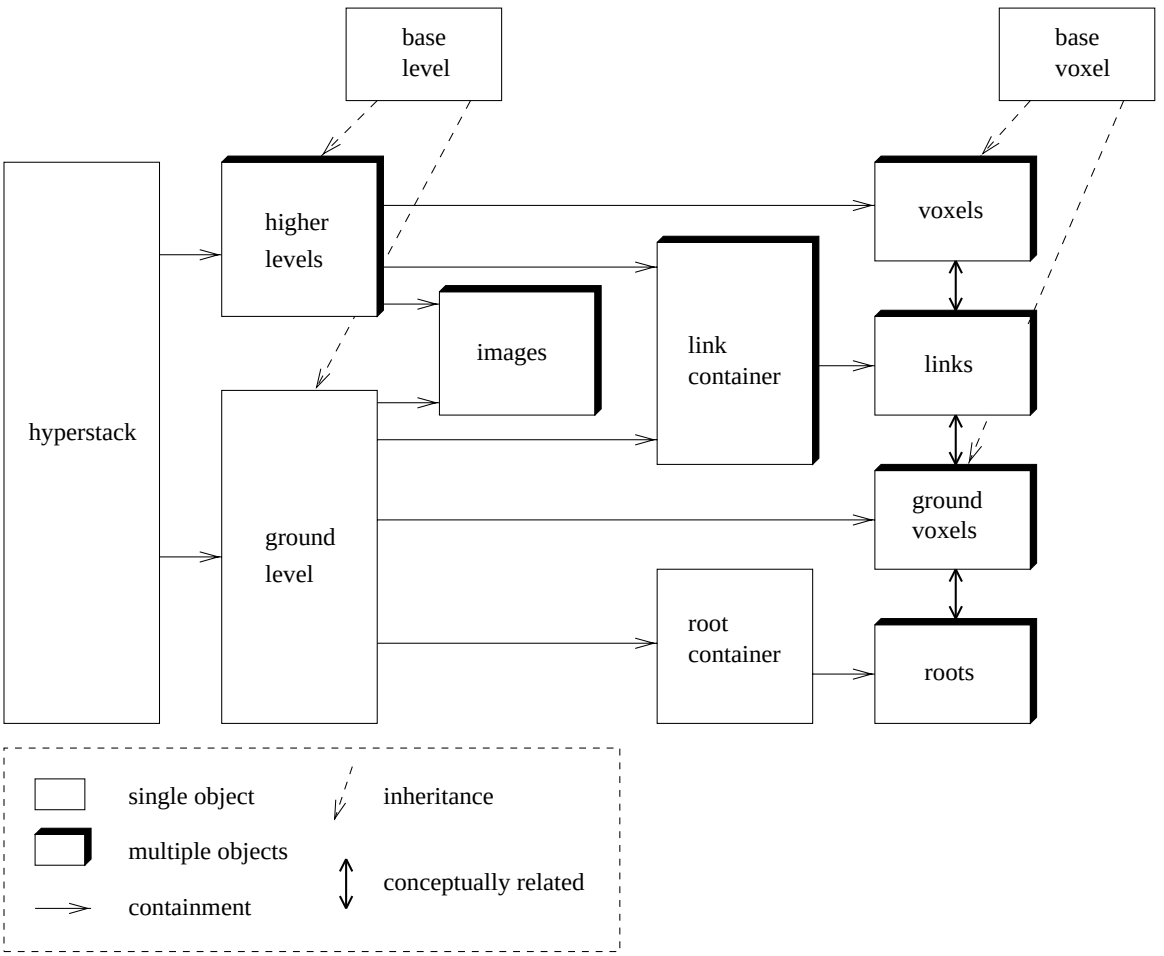


Fig. 2.6. *Class hierarchy of a hyperstack.*

As can be seen from Fig. 2.6, the actual image data (intensities) are separated from the voxels and the linkages. There are several reasons to do so:

- the number of intensities per voxel may be larger than one (multi-component images);
- additional feature images (*e.g.,* containing differential invariant information like gradient magnitude or curvature) may be needed in the linking process;
- the number of active voxels decreases when moving upwards in the hyperstack (convergence); this means that the number of voxels that actually need to be recorded and the number of intensities present in the raw image data at that scale can show a large discrepancy;
- the image data can now be handled separately (reading, writing, analysis of the data, etc.).

In order to save memory we discriminate between the ground level and higher levels. (As we will see later there are some clear differences between these, which can be of great advantage with respect to the amount of memory needed to build, store and use a hyperstack.)

2.3.2 The container concept

Conceptually, all information on linkages is handled by so-called *link containers*. A container is an object type designed to hold collections of other objects; the concept is well-known in object-oriented design [20, 96]. There is one link container for each level, although one could also choose for one big container for all the linkages through scale space. Three disadvantages of the latter approach are:

- copying a container (either explicitly by the programmer or implicitly by the compiler) costs more computer time
- indexing the separate linkages requires a larger address space
- the boundaries between linkages of separate levels are difficult to maintain (they may even overlap!)

For these reasons, we have implemented separate link containers per level.

Bottom-up versus top-down linking

Ground voxels need to store information about the roots they are connected to. This information (*i.e.,* a list of roots per ground voxel) concerns the spatial location of each root, the probability value and a root value. Similar to the link container idea that is used to implement the linkages, we use *root containers* to keep track of all interesting

roots. A major advantage of storing the roots separately from the ground voxels is that the roots can directly serve as input to a volume renderer.

As for the linkages three options have been examined (see Fig. 2.7):

1. *bottom-up linked lists.*
2. *top-down linked lists.*
3. *doubly linked lists.*

The doubly linked lists option is simple and straightforward, but very generous with memory. The data structure used in the prototype of the hyperstack [116, 118, 25] was a disguised variant of doubly linked lists. In this prototype we introduced *sibling* linkages (pointing from one child to another) to facilitate the scanning of a hyperstack. This allowed traversing the entire multidimensional tree structure, passing each voxel just once. In the case of probabilistic linking, however, one has to be very careful to prevent ambiguity in the sibling linkages (see [121]).

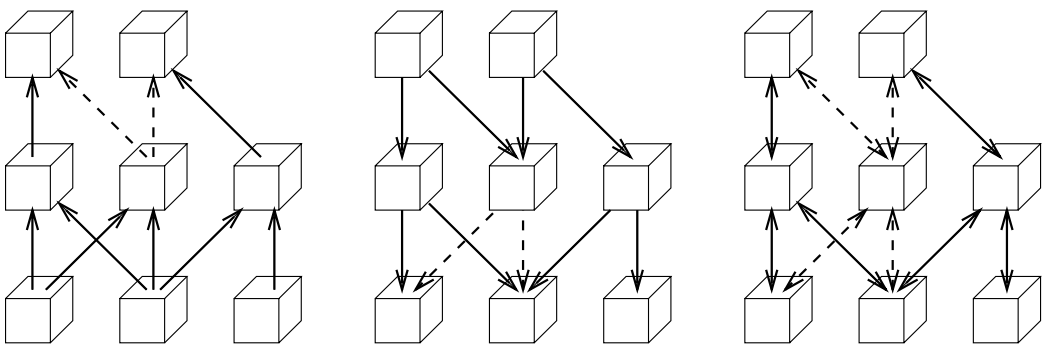


Fig. 2.7. *Three different linkage structures for a probabilistic hyperstack: (a) bottom-up linked lists, (b) top-down linked lists, (c) doubly linked lists. To emphasize the differences between these linkage structures, all linkages starting from the middle voxel have been denoted by dashed arrows.*

Much faster and consequently cheaper in building a hyperstack are the two remaining options, top-down and bottom-up linking. With half the number of references the same structure can be recorded. The only price to pay for this profit is a slightly more complicated scanning function: instead of one single pointer to the current voxel that simply follows child-, sibling- and parent linkages (in that order of priority), we now need an additional reference per level during the scanning process, since there is no way back to where we started scanning a sub-tree. Consequently, singly linked lists are preferable.

The choice between top-down and bottom-up linking is based upon the observation that in the case of top-down linking there is no relation whatsoever between all children

connected to one parent, but in the case of bottom-up linking we have the useful property that the sum of probabilities of all parent linkages starting from a specific child equals one (normalization). Checking and recalculating this important property is more difficult if these linkages originate from different parents downwards. Thus, we opted for singly linked lists based on bottom-up linking.

Implementation

The building phase and the segmentation phase have clearly been separated (see section 2.2). Therefore, if multiple root labeling strategies are to be investigated for one hyperstack, this hyperstack must be stored on disk after the building phase. This calls for an implementation of the container classes that is independent of dynamic memory allocation. To this end, we use conventional indexing methods. In Fig. 2.8 the pointer fields are shown as arrows for sake of clarity, although they have been coded as index fields (every index refers to a parent or a link.)

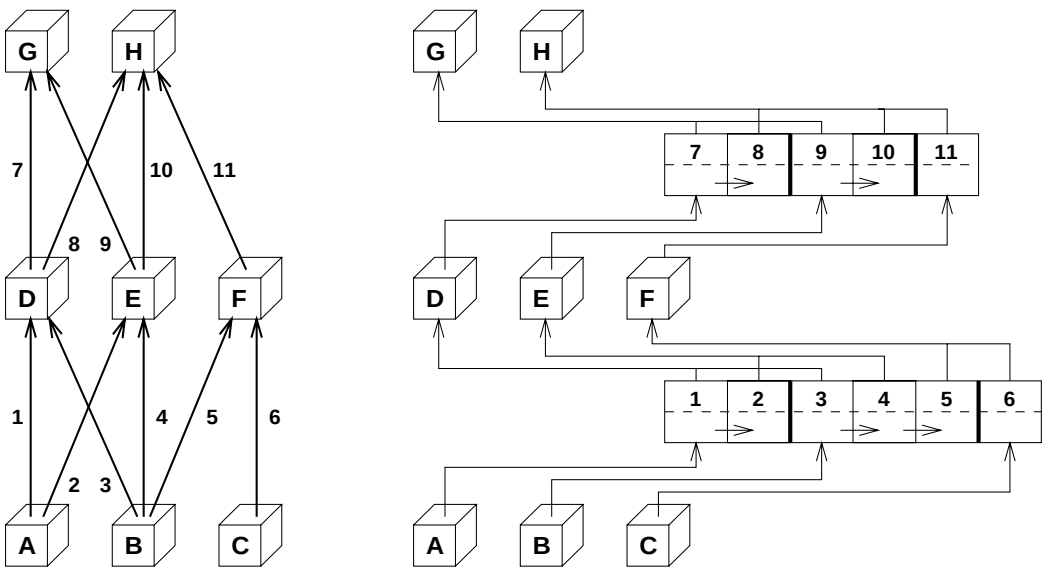


Fig. 2.8. *Implementation of the link container.*

Note that the number of linkages contained in each link container decreases at higher levels, according to the decreasing number of active voxels at larger scales (convergence). Thus, the container that holds the child-parent linkages from level n to level $n+1$ is not initialized until all the linkages from level $n-1$ to level n have been established. At that time, the number of active voxels in level n is known.

2.4 Addressing sparse levels

As the building of a hyperstack proceeds, the number of passive voxels increases. This is caused by the fact that multiple children can link to the same parent. The images at larger scales can be termed *sparse* in the sense that only a limited number of voxels is active in the linking process.

This section deals with aspects affecting the convergence rate, and gives a solution to store the sparse levels efficiently: *multidimensional hashing*.

2.4.1 Convergence rate

In Fig. 2.9 three graphs showing the dependency of the number of active voxels on scale are shown for different parametrizations of a typical hyperstack.

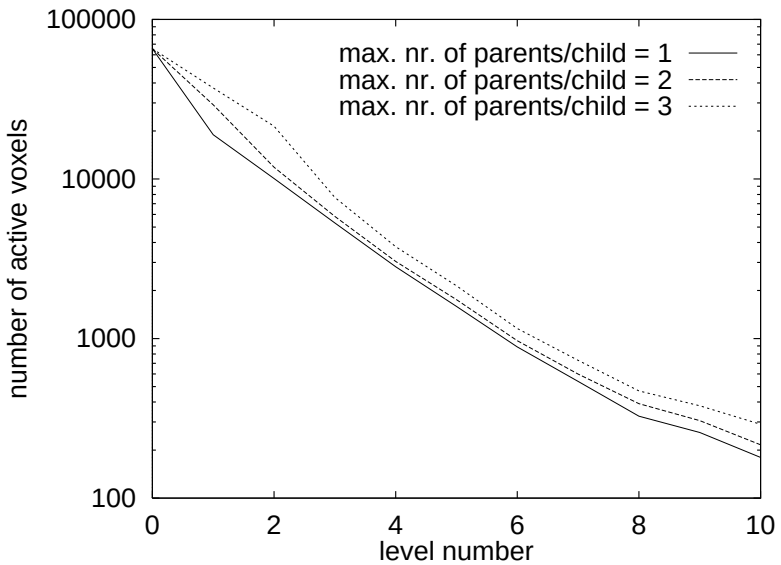


Fig. 2.9. *Dependency (logarithmic) of the number of active voxels on the scale level for three different parametrizations: one single-parent hyperstack (i.e., with a maximum of 1 parent per child), and two probabilistic hyperstacks (with a maximum of two and three parents per child, respectively). The image is two-dimensional of size 256 × 256.*

The convergence of the number of linkages in a hyperstack is influenced by:

- *The maximum number of parents per child.* A large value for this maximum results in a lower convergence rate than a small value. For instance, a single-parent hyperstack has a high convergence speed.

- *The blurring strategy.* A smaller scale space sampling will result in a lower convergence speed, since more levels are needed to assure that root voxels represent

a segment properly. Also, the search volume will be smaller according to equation (2.8), which means that it takes more levels for voxels far apart to link. This has obviously a negative effect on the convergence rate.

- *The linking strategy.* The weighting factors in the affection formula (2.11) influence the convergence in an image-dependent fashion. For instance, a high weight of the ground volume component will result in a high convergence rate if the image at hand contains large homogeneous areas.

- *The contents of the image.* Images with a high local variation in intensity values will normally need more blurring (and hence more levels) to form the different segments. This results in a lower convergence rate.

- *The partial volume effect.* Voxels with a partial volume effect will have difficulty in choosing between candidate parents representing different tissue types, so multiple parents are likely to occur (in probabilistic hyperstacks). This effect diminishes at the higher levels, but the total number of linkages is mainly determined by the linking characteristics in the lower levels, so the overall effect of partial volume voxels on the convergence of the hyperstack is negative.

As can be concluded from Fig. 2.9, the number of passive voxels increases approximately exponentially with increasing scale. So, we need an efficient way to store the active voxels (*i.e.,* the sparse levels in the hyperstack).

2.4.2 Conventional hashing techniques

Most known hashing techniques are based on intelligent mapping formulas, which transform an input index (or *logical* index) into a physical index (called the *hash* index). The used transformation formula is often based on the choice of a prime number.

In the worst case, hashing schemes based on mapping degenerate to index methods with access times comparable to the use of linked lists. In the ideal case, the access time via the hash function approximates that of direct indexing; then, each input index has a unique hash index. This requires two conditions to be fulfilled:

1. The logical indexes are evenly distributed over the logical index space (*e.g.,* see [103]).

2. It is known *beforehand* how sparse the logical index space will be.

Neither condition is satisfied for the hyperstack. The number of active voxels depends on too many factors to predict its incidence within an acceptable accuracy range, while the distribution of voxels over a hyperstack level is too image dependent to ensure a uniform distribution. Furthermore, the existence of multiple levels in the hyperstack is a complicating factor, because every level has its own number of active

voxels (logical indices) with its own distribution characteristics. Every level of the hyperstack thus requires a specific set of input parameters to the mapping formula of the hashing technique. Consequently, conventional hashing techniques are not suited for the hyperstack.

2.4.3 Multidimensional hashing

In this section we introduce *multidimensional hashing*, which is suited to store large sparse arrays, like the levels in a hyperstack, in an efficient way.

We expect that multidimensional hashing—as an alternative for hashing tables—can effectively be applied in several other image processing routines (*e.g.,* image coding).

Multidimensional hashing uses a multidimensional *bitmap*, in which each voxel is assigned one bit (see Fig. 2.10). Active voxels have the value 1, passive voxels the value 0. For sparse levels, the number of 1-bits will be low.

The elements of the cumulative array are used to store the number of 1-bits in the bitmap in a cumulative way. To this end, every cumulative array entry equals the sum of the number of 1-bits in the previous rows of the corresponding bitmap.

The working of the cumulative array in cooperation with the bitmap will now be briefly explained. Suppose $\mathcal{I}$ is a two-dimensional image of size $d_x \times d_y$, with a bitmap $\mathcal{B}$ of the same size, while the cumulative array $\mathcal{C}$ is an array of length d_y (in Fig. 2.11 we have $d_x = 5$, $d_y = 5$). The aim is to find the hash index $\mathcal{H}_{xy}$ for pixel (x, y) in the image $\mathcal{I}$.

If we denote the bits of the bitmap by $\mathcal{B}_{xy}$, then the entries $\mathcal{C}_y$ of the cumulative array take the value:

$$\mathcal{C}_y = \begin{cases} 0 & \text{if } y = 1 \\ \sum_{i=1}^{d_x} \sum_{j=1}^{y-1} \mathcal{B}_{ij} & \text{if } y > 1 \end{cases} , \tag{2.18}$$

for $1 \leq y \leq d_y$. The hash index of the element (x, y), with $1 \leq x \leq d_x$, can now be obtained from:

$$\mathcal{H}_{xy} = \mathcal{C}_y + \sum_{i=1}^{x} \mathcal{B}_{iy} \quad . \tag{2.19}$$

In three dimensions multidimensional hashing extends to a three-dimensional bitmap $\mathcal{B}$ and a two-dimensional cumulative array $\mathcal{C}$. The cumulative array adds the 1-bits similarly to two-dimensional hashing: the y–x order in 2D becomes z–y–x in 3D (meaning: first the x-direction is summed, then the y-direction, and finally the

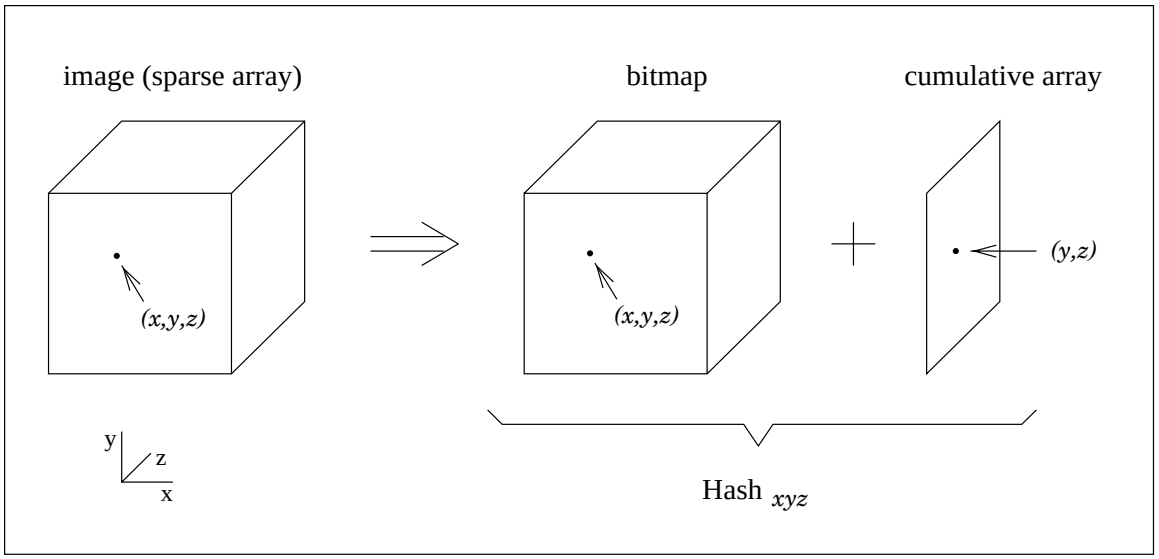


Fig. 2.10. *Schematic representation of the multidimensional hashing concept (3D case). The original image (left) is sparse in the sense that only some voxels need to be stored ('active voxels'), while the remaining elements are superfluous ('passive elements'). If the active voxels are put in a one-dimensional chain (not shown), then this chain can be indexed by using an equally sized bitmap (1-bit per element) and a cumulative array of two dimensions.*

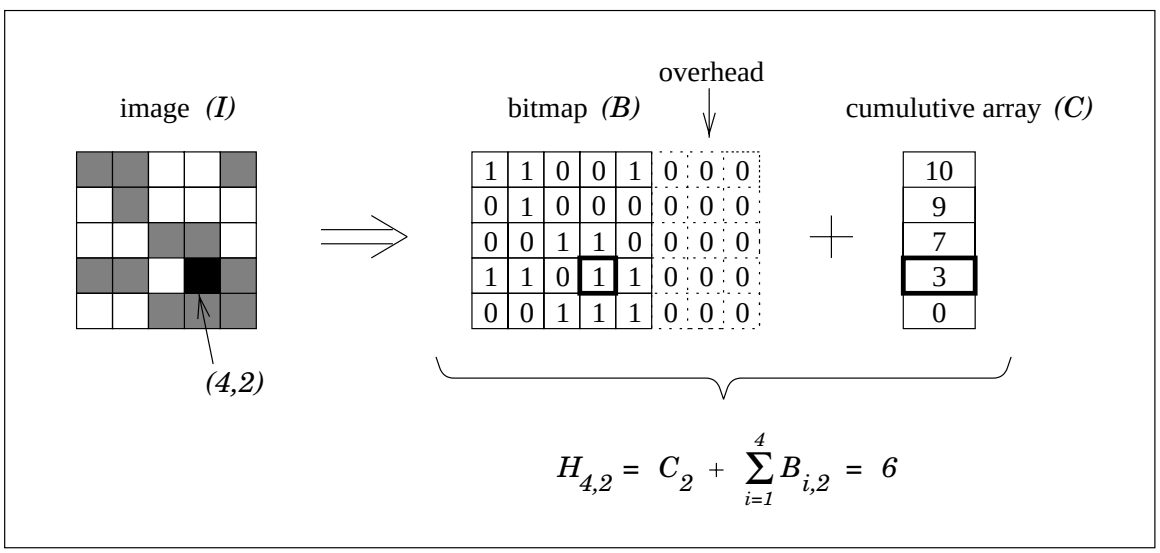


Fig. 2.11. *Two-dimensional example of the use of a cumulative array.* $\mathcal{C}_2 = 3$, $\mathcal{B}_{i,2} = 1$ *for* $i = 1, 2, 4, 5$, *so* $\mathcal{H}_{4,2}$ *equals 6.*

z-direction). Equation (2.18) extends to:

$$\mathcal{C}_{yz} = \begin{cases} 0 & \text{if } y = 1, z = 1 \\ \sum_{i=1}^{d_x} \sum_{j=1}^{y-1} \mathcal{B}_{ijz} & \text{if } y > 1, z = 1 \\ \sum_{i=1}^{d_x} \sum_{j=1}^{d_y} \sum_{k=1}^{z-1} \mathcal{B}_{ijk} & \text{if } y = 1, z > 1 \\ \sum_{i=1}^{d_x} \sum_{j=1}^{y-1} \mathcal{B}_{ijz} + \sum_{i=1}^{d_x} \sum_{j=1}^{d_y} \sum_{k=1}^{z-1} \mathcal{B}_{ijk} & \text{if } y > 1, z > 1 \end{cases} \tag{2.20}$$

for $1 \leq y \leq d_y$, $1 \leq z \leq d_z$. The hash index of (x, y, z) is straightforwardly obtained by:

$$\mathcal{H}_{xyz} = \mathcal{C}_{yz} + \sum_{i=1}^{x} \mathcal{B}_{iyz} \quad . \tag{2.21}$$

Implementation

As regards the implementation of the multidimensional hashing technique, we must pay attention to the overhead as a result of storing the bitmap. Bits are stored in scalar data types, or *units*, to comprise series of bits together. Each unit requires u bytes, typically 1 or 2. Bit operations are not performed directly on the bits, but through these units instead by means of bit shifts and logical operators. In Fig. 2.11 u equals 1, so the overhead for the image of size 5×5 is 3 bits per row (indicated by the dashed lines). If the y-dimension were different, however, then it also matters in which direction the units are chosen. Medical images are often square (and even $2^n \times 2^n$, n integer) in the xy-plane, but have a different size in the z-direction, defined by the number of relevant slices needed for the particular situation. Hence, the x-direction is a plausible choice for the direction of the units. If the size of the original image in the direction of the units equals a multiple of u bytes, there is no overhead.

The overhead $\Omega(n)$ for an image with x-dimension n can be formalized as:

$$\Omega(n) = \begin{cases} 0\% & \text{if } (n \bmod b) = 0 \\ \frac{b - (n \bmod b)}{n} \times 100\% & \text{otherwise} \end{cases} \tag{2.22}$$

where b denotes the number of bits used per unit (*i.e.*, $8u$).

In Fig. 2.12, the overhead $\Omega(n)$ for increasing n is sketched graphically for both $b = 8$ and $b = 16$. From this figure it follows that the overhead is negligible in the case of large sized images ($n > 256$). For most medical images, n equals 128, 256, 512 or 1024. In all of these cases, there is no overhead of the bitmap storage.

The overhead for the cumulative array is comparable to extending the original image with an additional 'slice' in the x-direction. This has only a minor effect on storage requirements.

The summation over the $\mathcal{B}$-values is best implemented by means of a look-up-table. Every possible unit value corresponds to a unique entry in the table, and the pre-calculated number of 1-bits is thus readily returned.

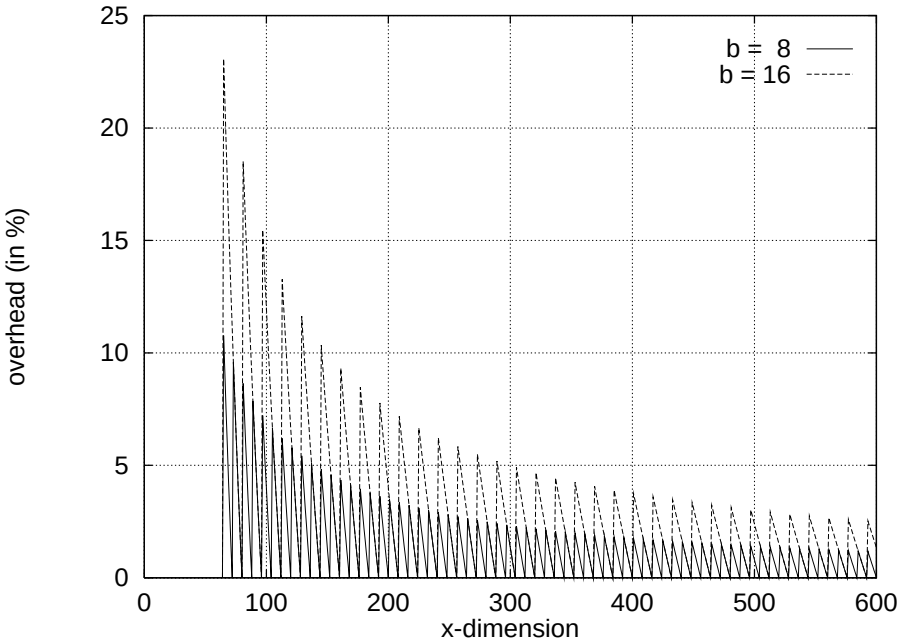


Fig. 2.12. *Overhead for the bitmap used for increasing values of n (the size of the x-dimension of the image), expressed in percentages.*

2.4.4 Discussion

The need for an effective hashing technique is clear, but the question rises at which ratio of active voxels versus total number of voxels cumulative arrays become useful. If we define $\mathcal{N}$ as the number of bytes needed if no hashing is applied, and $\mathcal{H}$ as the same number if cumulative arrays are used, then we require that $\mathcal{H} < \mathcal{N}$.

For simplicity, suppose we have an image of size n^d (d is the number of dimensions). Without any form of hashing we would need to allocate n^d voxel objects, irrespective of whether they are active or not. Thus, if one voxel object occupies v bytes, then $\mathcal{N} = v \cdot n^d$. Next, suppose that m out of n^d voxels are active. This will cost m voxel objects (the absolute minimum), plus a bitmap of n^d bits—we assume no overhead exists—and a cumulative array with n^{d-1} entries. Hence, if one cumulative array entry needs c bytes, then

$$\mathcal{H} = m \cdot v + \frac{n^d}{8} + c \cdot n^{d-1} \quad . \tag{2.23}$$

If we denote m/n^d by p, then $1 - p$ is a measure of the sparseness of the data. If we further define the profit of cumulative arrays as the ratio $(\mathcal{N} - \mathcal{H})/\mathcal{N}$, we arrive at:

$$\text{profit} = 1 - p - \left(\frac{1}{8v} + \frac{c}{nv} \right) \quad . \tag{2.24}$$

Since $8v \gg 1$ (note that not just the original intensities are stored for each ground voxel, but also a number of hyperstack properties) and $nv \gg c$, the profit is almost

proportional to the sparseness $1-p$. Since p has been found to be much smaller than 1 in all of our studies, cumulative arrays have a large advantage over a straightforward implementation (*i.e.,* without hashing), irrespective of the distribution of the active voxels over the image.

2.5 Results

In this section we present some results of hyperstack segmentation. We first discuss 2D versus 3D image analysis by means of an artificial image, and then give some segmentations of real world (medical) images.

2.5.1 2D versus 3D image analysis

The hyperstack segmentation method has been implemented and tested on an artificial test image and on real world (medical) images. For the test image, the results of fully 3D segmentation are compared to slice-by-slice processing (see section 2.5.1). Since test images have a 'gold standard' (*viz.* the originally created image without noise), quantitative validation methods can be used to check the results.

We distinguish between three different dimensionalities of segmentation processes for three-dimensional images:

- **3D.** The full spatial information of the 3D image is taken into account in the hyperstack, which thus becomes a four-dimensional data structure.
- $\mathbf{2\frac{1}{2}D}$. This version works on the individual slices of the 3D data set. The correlation of the image slices in the slice direction is sacrificed for the sake of computational efficiency. The variant is called $2\frac{1}{2}$D because the image itself is processed on a slice-by-slice basis, but the result (a concatenation of the individual 2D results) is a 3D image.
- $\mathbf{2\frac{3}{4}D}$. In this hybrid version the full 3D spatial information is used in the blurring step of the hyperstack. Next, however, the different slices are linked and further processed on a 2D basis by chopping the four-dimensional scale space (three spatial dimensions plus scale) into multiple three-dimensional scale spaces (two spatial dimensions plus scale). After the downward projection step the segmented slices are joined again to generate the three-dimensional segmentation.

These three variants will be compared as regards the quality of the segmentation they produce. As an aside, in appendix 2.B the notion of Morse critical points is discussed in the light of $2\frac{1}{2}$D versus 3D hyperstack analysis.

For the experiments we have used the so-called AIRPLANE picture. (see Fig. 2.13). The image has been modeled with the volumetric modeler THINGS [121]; this package is capable of simulating partial volume artifacts. The fuselage (body) of the airplane

is an elongated ellipsoid, while the wings consist of 2 very flat ellipsoids (the diameter of each ellipsoid in the y-direction equals 1.6 pixels in the middle). The tail is also a small, 2 pixels flat ellipsoid perpendicular to the wing direction, and the propeller is modeled through a torus with a diameter of 1.6 pixels. The three-dimensional airplane has dimension sizes $64 \times 32 \times 64$, which allows fast processing of the image irrespective of the method used ($2\frac{1}{2}$D, $2\frac{3}{4}$D, or 3D). The intensity of the airplane has been set to 2000, the background equals 1000. We added sufficient Gaussian noise (standard deviation of 10%) so that object and background could not be separated via simple thresholding.

The AIRPLANE image has been processed in x, y, and z-direction for both the $2\frac{1}{2}$D and the $2\frac{3}{4}$D method. In both cases the resulting segmentations of the individual slices have been concatenated to produce the desired 3D result. Postprocessing steps—such as pixel connect operations to remove single noise pixels—have deliberately been omitted to emphasize the effect of each segmentation method. The results are presented below.

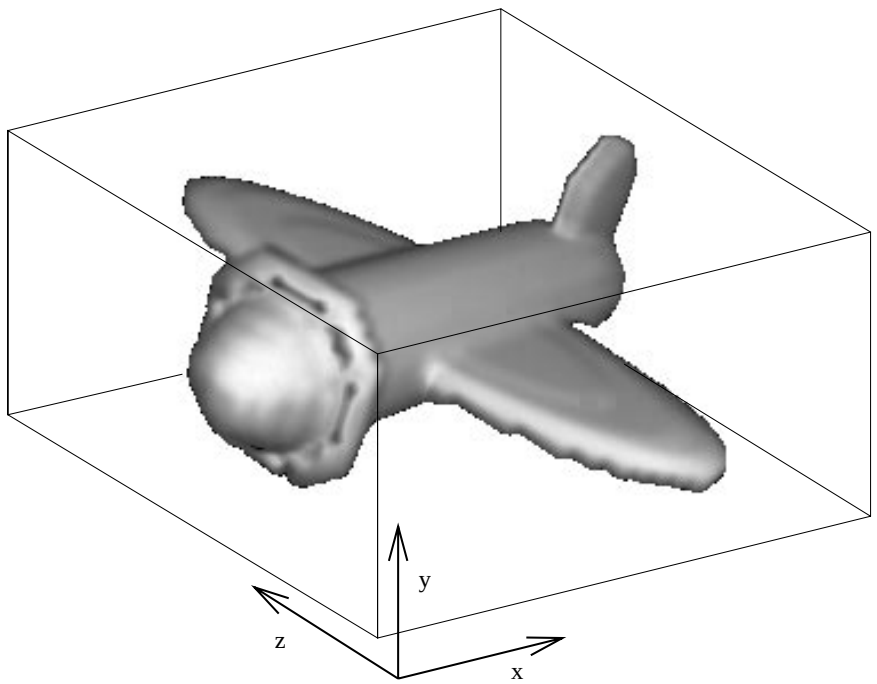


Fig. 2.13. *Rendering of the test image* AIRPLANE *that is used for the* $2\frac{1}{2}D$, $2\frac{3}{4}D$, *and 3D experiments. The slice-by-slice experiments have been performed for all three Euclidean directions.*

2.5.2 $2\frac{1}{2}$D segmentation

Fig. 2.14 shows the results of $2\frac{1}{2}$D image segmentation for all three Euclidean directions (the upper, middle, and lower frames correspond to x, y, and z-direction,

respectively). From left to right the viewpoint changes (one oblique, three perpendicular views) in order to make a fair comparison with the alternative slice-by-slice methods and fully three-dimensional segmentation.

The propeller is only visible in the x-direction. This can be explained by the fact that the propeller is completely contained in two slices in the x-direction: These pixels can not escape to other slices, so this object can be segmented with a reasonable accuracy. The same counts for the wings: if processed in the y-direction the result seems acceptable, otherwise only parts of the wings have been segmented correctly. In the y-direction, the tail has completely been separated from the fuselage.

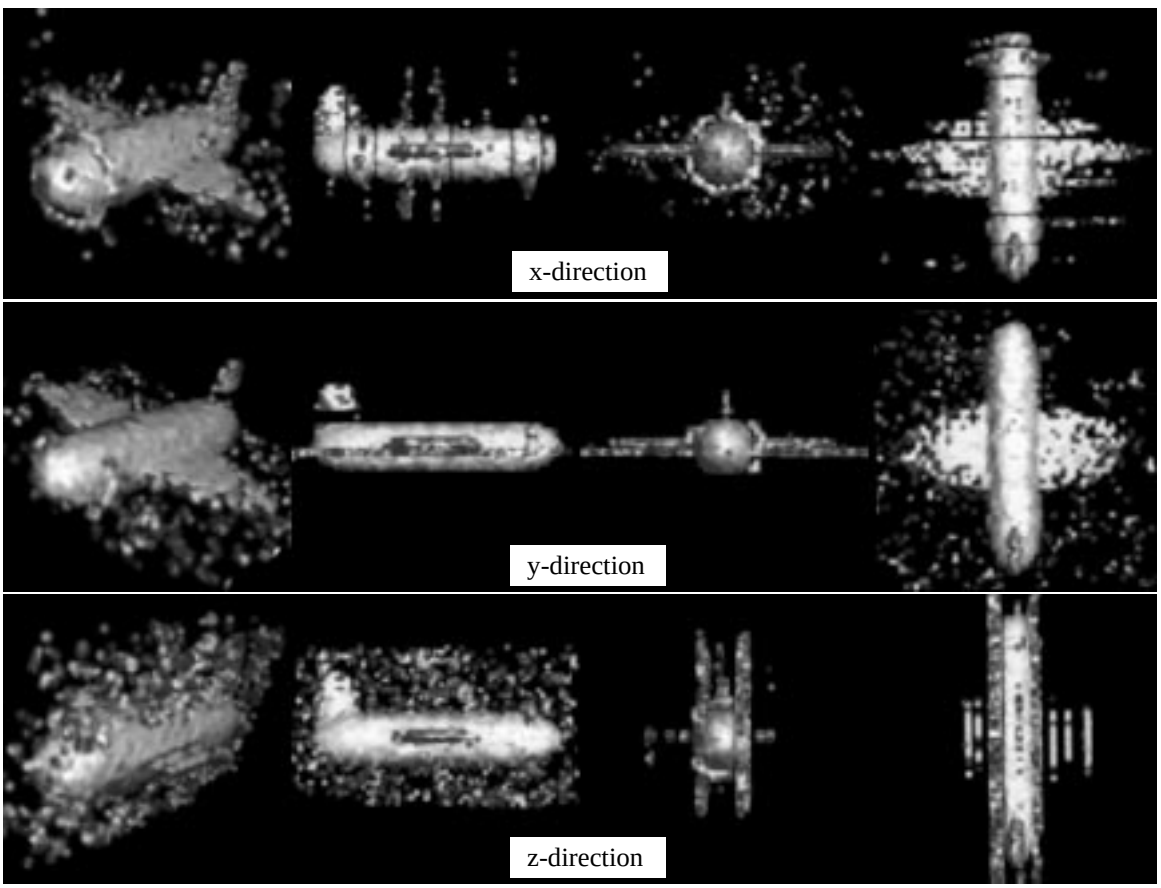


Fig. 2.14. *$2\frac{1}{2}D$ segmentation of the* AIRPLANE *test image of Fig. 2.13. The three horizontal frames correspond to slices in the x, y, and z-direction, respectively.*

2.5.3 $2\frac{3}{4}$D segmentation

Fig. 2.15 shows the results for the $2\frac{3}{4}$D segmentation variant. The viewpoints correspond with those of Fig. 2.14.

The propeller is, again, only completely visible in the x-direction. The y-direction shows a small improvement over $2\frac{1}{2}$D processing. This can be explained by the fact that the three-dimensional scale space earlier reaches the vanishing point (the 'root' of the wings) than the two-dimensional scale space. (The darker background voxels have a destructive effect on the lighter wings.) Hence, in the $2\frac{3}{4}$D case the voxels of the wings have less possibilities to 'escape' from the wings, as does happen in the $2\frac{1}{2}$D variant. The $2\frac{3}{4}$D segmentation in the z-direction produces no wings at all: the voxels representing the wings vanish into the background at the lowest levels of the hyperstack.

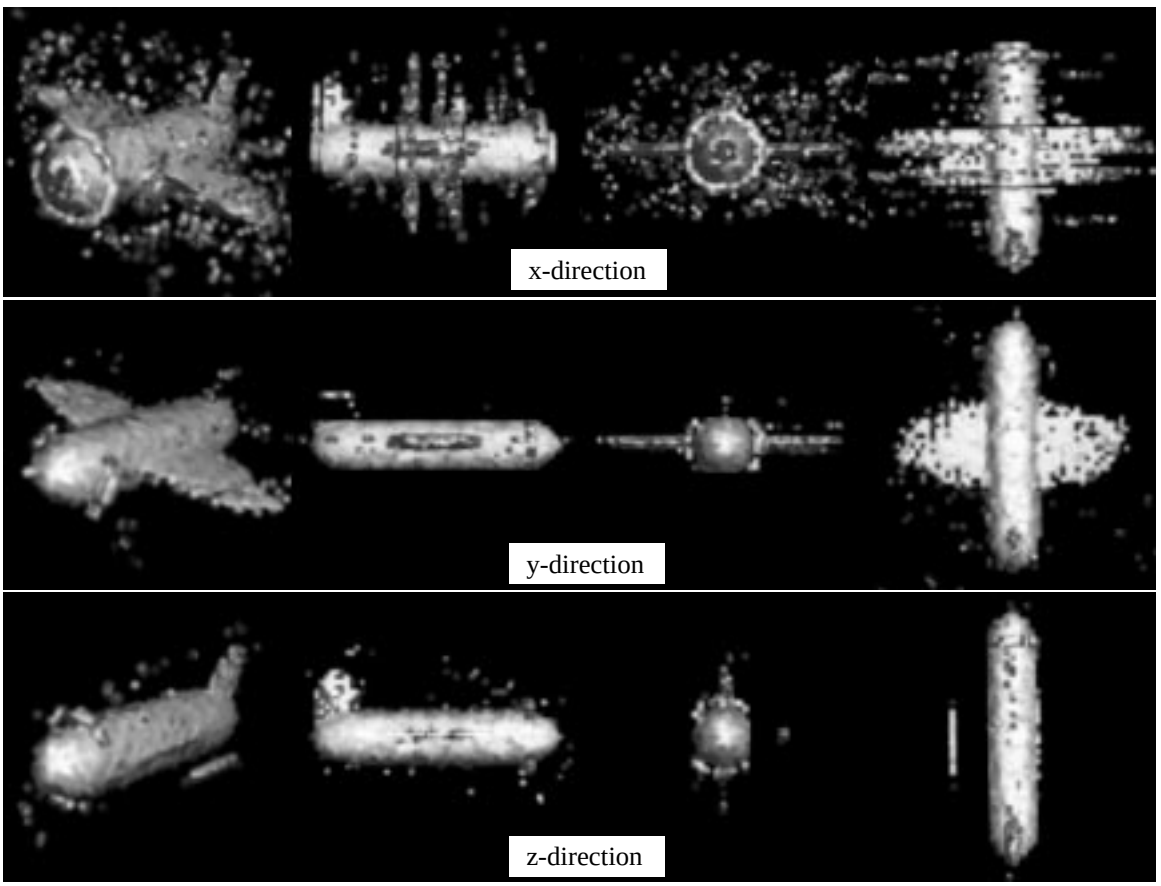


Fig. 2.15. *$2\frac{3}{4}D$ segmentation of the* AIRPLANE *test image of Fig. 2.13. The three horizontal frames correspond to slices in the x, y, and z-direction, respectively.*

2.5.4 3D segmentation

Finally, in Fig. 2.16 the fully 3D segmentation is shown. Clearly can be seen that all parts are present (fuselage, wings, tail and propeller). For the wings of this particular

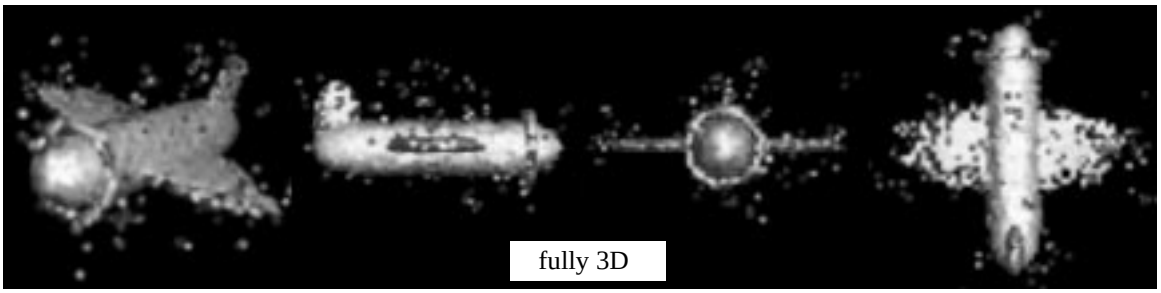


Fig. 2.16. *Fully 3D segmentation of the* AIRPLANE *test image of Fig. 2.13.*

object, only $2\frac{3}{4}$D segmentation in the y-direction outperforms 3D hyperstack segmentation. Thus, this should be regarded a lucky shot, which might be made structural only by incorporating *a priori* knowledge about the object to be segmented.

2.5.5 Segmentation of medical images

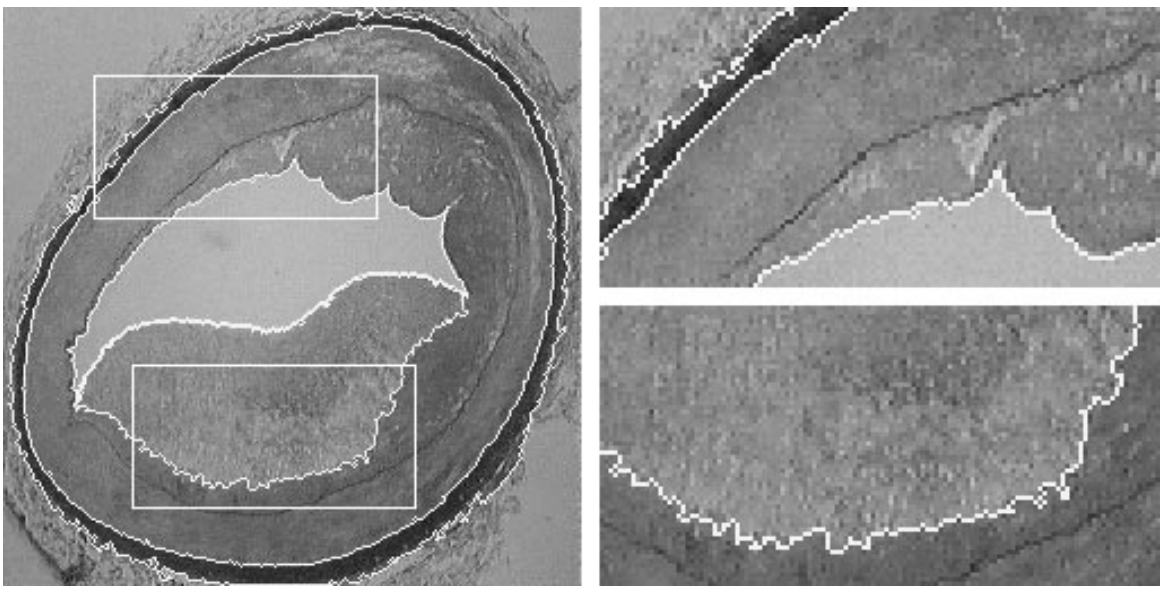


Fig. 2.17. *Single-parent segmentation of the thrombus and the vessel wall of the histological image* VESSEL. *Segment contours have been superimposed in bright white. Enlargements on the right.*

In this section two illustrations of the possibilities of hyperstack segmentation of real world complex images are presented. Fig. 2.17 shows the potential of the hyperstack in segmenting images containing objects of varying size. The image at hand is called VESSEL, a 256^2 histological image in which both the blood vessel wall and the thrombus in the middle are simultaneously segmented using a single-parent

hyperstack. The extended root labeling scheme, where the roots are distributed over a relative large range of levels makes this kind of multi-tasking segmentation possible.

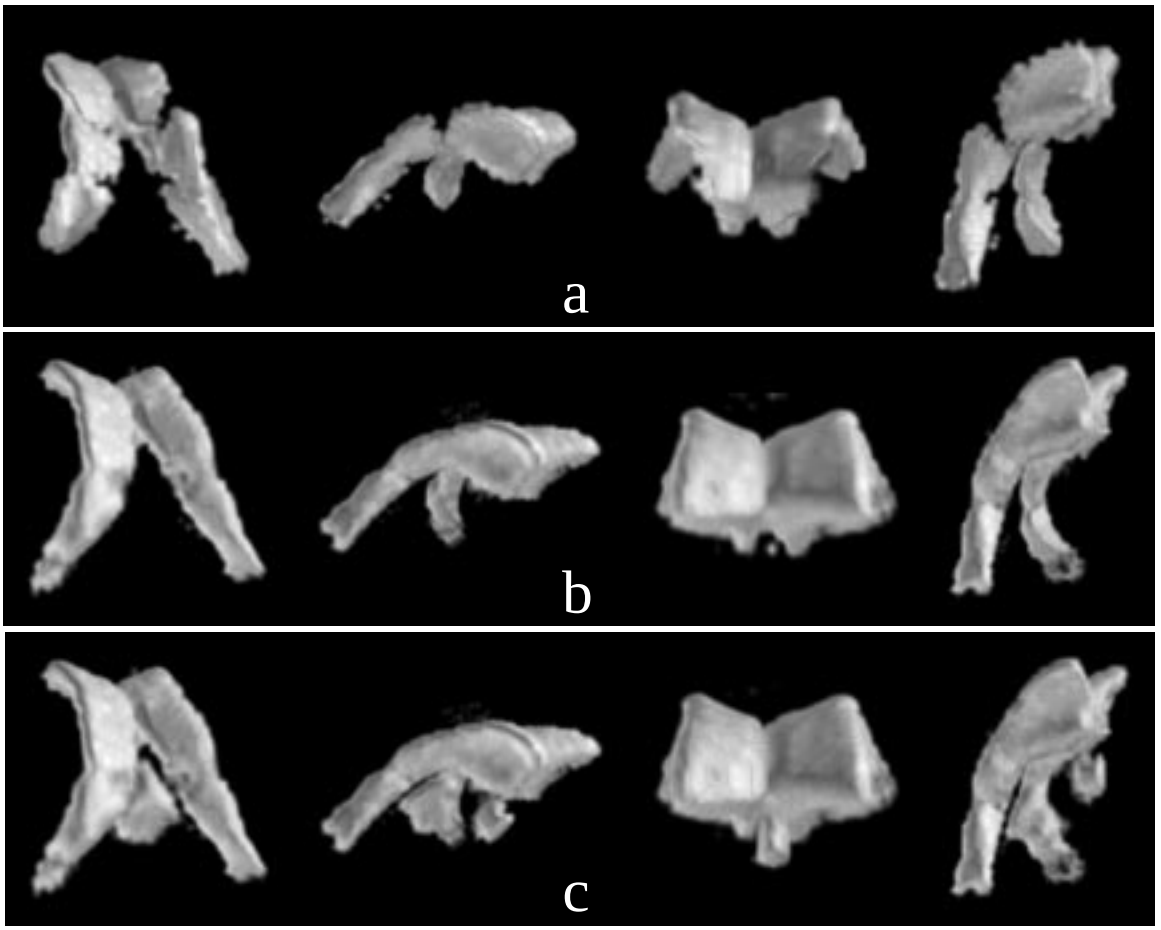


Fig. 2.18. *Renderings of hyperstack segmentations of the ventricle system of the* VENTRICLES *image, a 3D MR image of the brain. Single-parent segmentation based on intensity following only (left) and a multi-parent segmentation based on the extended linking scheme (middle). Adding objects is very simple (right).*

In Fig. 2.18 the surplus value of the extended linking schemes over intensity-only based linking is shown for segmentation of the ventricle system of a 3D MR image of the brain, called the VENTRICLES image. This image contains 32 slices of 128^2 and a grey-level resolution of 12 bits. The simple linking scheme (based on intensity following only) results in a segmentation that has several serious shortcomings, which are not present with the linking scheme of section 2.2.2.

2.6 Conclusions

In this chapter we have presented the hyperstack segmentation method for multi-dimensional images. The method consists of blurring, linking, root labeling, and a downward projection.

We introduced two extensions to the conventional linking scheme based on intensity following: the affection formula, and probabilistic (multi-parent) linking. The adultness formula has shown to be a powerful tool for root labeling schemes.

Furthermore, we have introduced multidimensional hashing as a new technique for storing the elements of a sparse multidimensional data structure without losing the spatial information of each element. As has been shown, the overhead is minimal for square images. More important, multidimensional hashing poses no requirements on the input data, like conventional hashing techniques.

We presented the results of a study where slice-by-slice processing has been compared to fully 3D segmentation. The latter approach is clearly superior.

Finally, we have illustrated the potential of multiscale hyperstack segmentation by applying the method to two complex medical images.

2.A Calculation on the search volume

The absolute, inner scale at level n is given by σ_n (see also Fig. 2.4). Since it does not make sense to link in a search volume with a radius smaller than the inner scale, k_n must have a minimum $k_{n,min}$. Hence, we have the requirement:

$$k_{n,min} \cdot \sigma_{n,n+1} \;=\; \sigma_{n+1} \quad . \tag{2.25}$$

Furthermore, from (2.2) it can easily be derived that the following property holds:

$$\sigma_{n,n+1}^2 \;=\; \sigma_{n+1}^2 \;-\; \sigma_n^2 \quad . \tag{2.26}$$

(2.7), (2.25), and (2.26) yield:

$$k_{n,min} \;=\; \frac{\exp(\delta\tau)}{\sqrt{\exp(2\delta\tau)-1}} \quad . \tag{2.27}$$

A convenient choice for k_n can be found by relating k_n to the influence that a certain pixel P has on the blurred value of its neighboring pixel Q. This influence diminishes if the spatial distance $d(P,Q)$ increases, according to the Gaussian that is used to blur the image. The relation is quantified by the requirement that the influence of P on Q at a spatial distance $d(P,Q) = k_n\sigma$—*i.e.,* $G(k_n\sigma;\sigma)$—*relative* to the influence of pixel P on the blurred value of P itself—*i.e.,* $G(0;\sigma)$—must be at least $\gamma\,\%$. In other words:

$$\frac{G(k_n\sigma;\sigma)}{G(0;\sigma)} \;=\; \frac{\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^d \exp\left(-\frac{k_n^2\sigma^2}{2\sigma^2}\right)}{\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^d} \;\geq \gamma \quad , \tag{2.28}$$

from which follows

$$k_n \leq \sqrt{-2\ln\gamma} \ , \ 0 < \gamma \leq \gamma_{max} \ , \tag{2.29}$$

with $\gamma_{max} = \exp(-k_{n,min}^2/2)$. Note that relation (2.29) is independent of the dimension d. In Fig. 2.19, k_n has been plot as a function of the influence factor γ according to the upper bound $k_n = \sqrt{-2\ln\gamma}$.

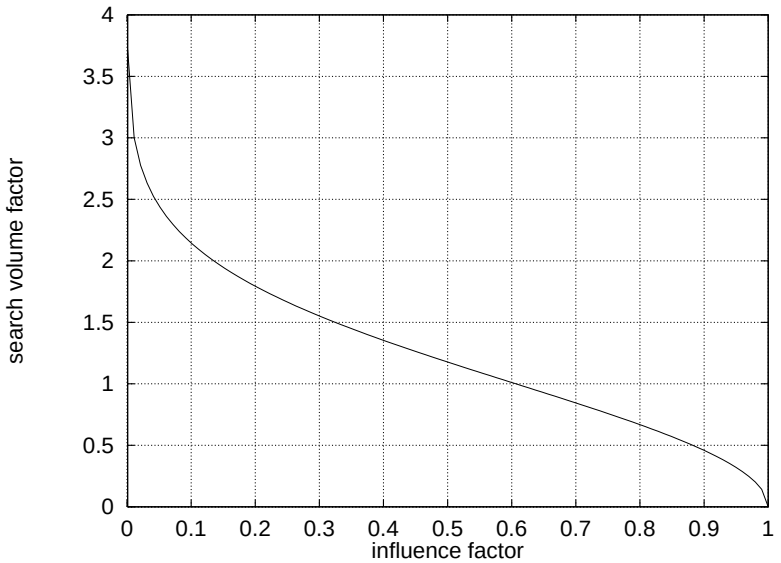


Fig. 2.19. *Plot of different values of k_n (i.e., the multiplication factor for $\sigma_{n,n+1}$ that determines the search volume of the children in level n looking for parents in level $n+1$) as a function of the influence factor γ. See the text for an explanation of the influence factor γ.*

2.B Morse critical points

The most interesting points in an image are those whose topological properties change with increasing scale, the so-called *Morse critical points*. In the 2D case there are three types of Morse critical points: minima, maxima, and saddles. In the 3D case, there are four types of critical points: minima, maxima, and two types of (hyper)saddles [25]: maximum saddles and minimum saddles. We define a maximum saddle as a critical point with a decrease in intensity in two directions and an increase in the third direction of the Euclidean space. At a minimum saddle the intensity increases in two directions, while the third direction shows a decrease in intensity. We note that Morse critical points may be missed entirely using $2\frac{1}{2}$D—or even $2\frac{3}{4}$D—image processing.

As an example, consider the quadric $f(x,y,z) = x^2 + y^2 - z^2$. The origin is obviously a (minimum) saddle, so we may expect to gain an insight into the behavior

of this function by considering the coordinate planes through the origin. These planes are denoted by $f_{x\perp}$, $f_{y\perp}$, and $f_{z\perp}$.

Each cutplane can be displayed separately as a quasi 3D view, with the intensity value as the height of the object (see Fig. 2.20). For the $f_{x\perp}$ and the $f_{y\perp}$ cutplanes this leads to a saddle landscape, corresponding to the equations $f(0, y, z) = y^2 - z^2$ and $f(x, 0, z) = x^2 - z^2$, respectively. The cutplane $f_{z\perp}$ shows the minimum corresponding to $f(x, y, 0) = x^2 + y^2$. The saddle would not have been recognized by $2\frac{1}{2}$D image processing with slices perpendicular to the z-direction.

In real (medical) images a lot of critical points are present. For instance, it is not hard to recognize several saddle points on the surface of a human head. In view of the above, the detection of critical points calls for fully 3D hyperstack analysis.

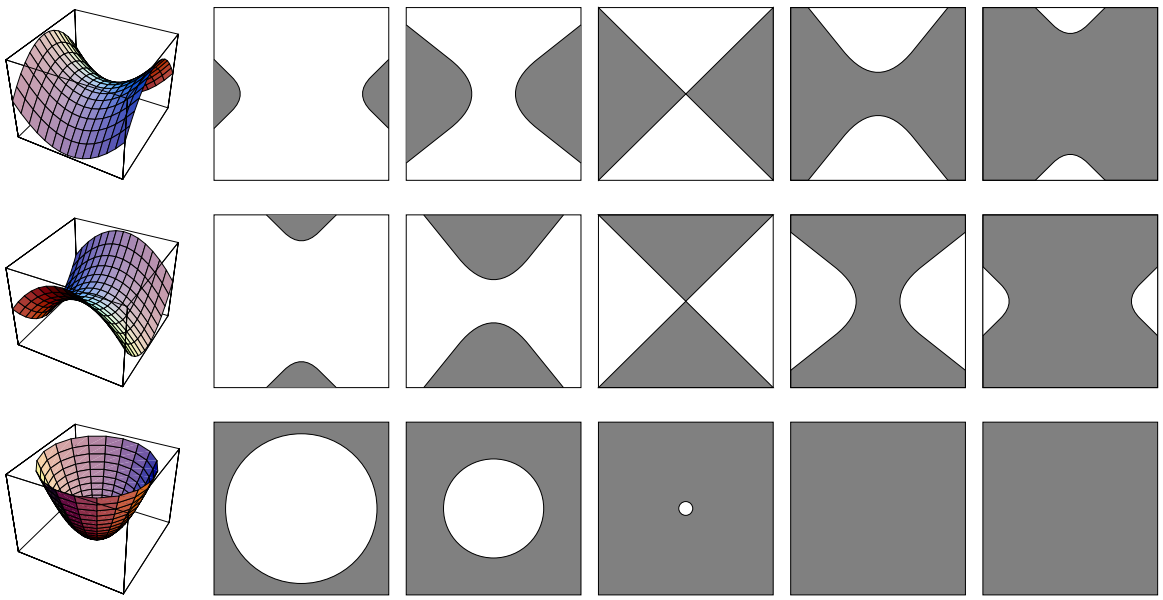


Fig. 2.20. *Quasi three-dimensional views of the cutplanes through the origin of the function* $f(x, y, z) = x^2 + y^2 - z^2$. *From top to bottom, equidistant slices have been included (on the right) to illustrate the difficulty of detecting a saddle point in a solid 3D model by* $2\frac{1}{2}D$ *image processing. The pictures correspond to* $f_{x\perp}$, $f_{y\perp}$, *and* $f_{z\perp}$ *(top to bottom), respectively. The saddle point can clearly be determined from the top and the middle row, but not from the bottom row.*

'Art is goose-pimples or tears'

– Kees van Kooten
(Boekenbal 1995)

Chapter 3

Volumetric modeling

Abstract

In this chapter we present a recursive octree-like algorithm for the conversion from mathematically defined objects to a voxel-based model. The resulting volume represents the objects with an accuracy depending on (*i*) the number of bits used to store each array element, and (*ii*) the depth used in the recursion. This way, the partial volume voxels can be accurately represented by an in-between intensity value.

Keywords: Volumetric modeling, partial volume artifact, volume rendering.

3.1 Introduction

The use of artificial images in image processing has the advantage that all the geometrical properties of the object(s) are known *a priori*. This makes it possible to test algorithms very thoroughly and in a quantitative manner.

We will explain the conversion of one or more spheres into a volume of grey levels. This in contrast with the well-known geometric modeling method based on octree encoding [72], in which a sphere is represented by a hierarchical structure of voxels and sub-voxels. If the method presented here is used in its simplest and fastest form the result is equivalent to a binary discretization of the sphere [76].

Extension of the method to other objects is straightforward. As an example, the discretization of ellipsoids, cylinders and truncated cones is discussed. The method proposed here can be used to create test images for image processing applications and volume visualization.

3.2 The algorithm

Our recursive algorithm, aimed towards 3D images, can best be explained using a 2D picture (see Fig. 3.1).

In Fig. 3.1 a part of a circle is discretized at three succeeding depths, while 4×3 image pixels are shown. Every pixel is first checked for being entirely inside, entirely outside, or on the edge of the circle. If a pixel turns out to be an edge-pixel *and* the current depth has not yet reached its user-defined maximum value, then the edge-pixel is divided into four sub-pixels, each of which is subjected to the same procedure again. If, on the other hand, the depth may not be increased anymore, then a simple inside/outside test of the midpoint of that sub-pixel stops the recursion. (With a maximum depth of 1 only the midpoint calculation is performed for every pixel.) If this simple test is too rough for a particular application the maximum depth can simply be increased.

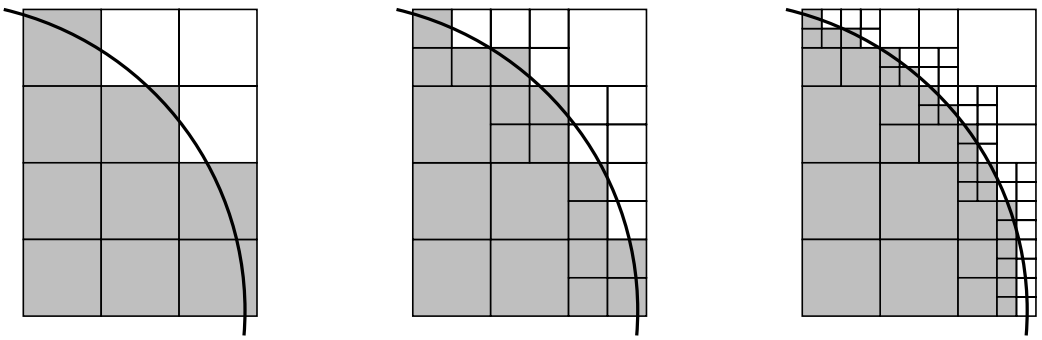


Fig. 3.1. *Discretization of a part of a circle at different depths.*

The final grey value of each edge pixel in the original grid is a weighted sum of the contributions of the sub-pixels comprised in it; the weight factors are proportional to the sub-pixel areas. For example, if the sphere has a grey value of 100 and the background is 0, then the center pixel at the upper row in Fig. 3.1a will have value 0, 25 ($= 100 \cdot 1/4$) and 19 ($\approx 100 \cdot (3/16)$) at depth 1, 2 and 3, respectively. As this simple example illustrates, the pixel values will converge to the actual area of intersection with the circle.

Extension of the above to 3D is straightforward. The weight factors in the sum then are proportional to the sub-voxel volumes.

3.3 Modeling

When defining spheres we have to deal with the possibility that two or more spheres can intersect. We therefore define all spheres in an order-dependent way; that is, the sphere with the highest priority is defined first, followed by the one with the second highest priority, and so on. When checking a voxel upon intersection with the specified spheres, the sphere with the highest priority is checked first. In this way, the priority of each sphere defines the visibility of intersecting parts.

The actual voxel-sphere intersection test is executed by an algorithm of Arvo [6], which has slightly been adapted to be able to discriminate between voxels that are entirely inside or entirely outside a sphere. Using his idea, errors in the intersection test—as described by Shaffer [110]—can be avoided.

With minor adaptations the algorithm can be applied for the generation of hollow instead of solid spheres (by using Arvo's original algorithm).

Note that the spheres considered here do not necessarily have to be origin-centered, since all voxels are simply tested sequentially. In fact, no advantage is taken of any similarity between voxels in the output image.

3.4 Pseudo-code

We now present the pseudo-code of the algorithm for the 3D case. Note that 'Grey' is used to denote the format for the storage of one intensity value. Basic data types and variables are:

```
Grey : integer;                                   format for one intensity

Sphere : record [
    x, y, z : real;                               center of the sphere
    r2 : real;                                    square of the radius
    g : Grey;                                     grey value of the sphere
];

nrOfSpheres : integer;                            number of spheres
spheres : array [1..nrOfSpheres] of Sphere;       sphere specifications
maxDepth : integer;                               maximum depth for recursion
backgroundVoxel, sphereVoxel,
    edgeVoxel : integer;                          the three voxel types
voxelType : integer;                              type of a voxel
background : Grey;                                background grey value
vmin, vmax : array [1..3] of real;                minimum and maximum
                                                  of a voxel for each axis
dimX, dimY, dimZ : integer;                       dimension sizes
image : array [1..dimX] of array [1..dimY]
    of array [1..dimZ] of Grey;                   'image' contains the result
```

The main procedure is *discretize_spheres*. In three straightforward loops the entire image is filled with appropriate intensities, according to the specified depth. The function *get_voxel_type* returns one of the three possible voxel types, which in case of an edge voxel results in a call to the recursive function *split_voxel*. The *vmin* and *vmax* elements keep track of the minimum- and maximum values of the coordinates

of the corners. These are used in the voxel-sphere intersection test (see the function *get_voxel_type*).

```
procedure discretize_spheres();
  Note: lines enclosed in angle brackets < and > should be replaced with the code described.

depth : integer;                                                    current depth

begin
    depth := 1;
    for x := 1 to dimX do
        for y := 1 to dimY do
            for z := 1 to dimZ do
                < fill vmin and vmax >
                if depth < maxDepth then
                    voxelType := get_voxel_type(vmin, vmax, ref sphereNr);
                    if voxelType = backgroundVoxel then
                        image[x][y][z] := background;
                    else
                        if voxelType = sphereVoxel then
                            image[x][y][z] := spheres[sphereNr].g;
                        else
                            image[x][y][z] := split_voxel(x, y, z, depth + 1);
                        fi
                    fi
                else
                    image[x][y][z] := midpoint_value(x, y, z);
                fi
            od
        od
    od
end
```

The function *get_voxel_type* is derived from the *SolidBox_HollowSphere* function of Arvo [6]. The only difference is that *get_voxel_type* returns an integer instead of a boolean, depending on the type of intersection between the two objects.

```
integer function get_voxel_type(vmin, vmax, ref sphereNr);
                                  Note: sphereNr is returned via call-by-reference.

center : array [1..3] of real;                                      midpoint coordinates

begin                                                  Continued on next page
```

```
    for sphereNr := 1 to nrOfSpheres do                    Continued from previous page
        center[0] := spheres[sphereNr].x;
        center[1] := spheres[sphereNr].y;
        center[2] := spheres[sphereNr].z;
        dmax := 0;
        dmin := 0;
        for i := 1 to 3 do
            a := (center[i] − vmin[i])^2;
            b := (center[i] − vmax[i])^2;
            dmax := dmax + max(a, b);
            if center[i] < vmin[i] then
                dmin := dmin + a;
            else
                if center[i] > vmax[i] then
                    dmin := dmin + b;
                fi
            fi
        od
        if dmin ≤ spheres[sphereNr].r2 then
            if spheres[sphereNr].r2 ≤ dmax then
                return [edgeVoxel];
            else
                return [sphereVoxel];
            fi
        fi
    od
    return [backgroundVoxel];
end
```

The recursive function *split_voxel* is similar to the main procedure *discretize_spheres*, except that the grey value is calculated for exactly one (sub-)voxel instead of for all voxels of the image. If the depth of the current recursion level has not yet reached its upper bound, this function will be called recursively in case of an edge voxel. Each sub-voxel is then subjected to the same test again.

```
Grey function split_voxel(midX, midY, midZ, depth);
sum : real;                                          adds the 8 sub-voxel values
weight : real;                                       the weight factor per sub-voxel

begin
    weight := 1/8;                                   const weight factor for 3D
    sum := 0;                                        Continued on next page
```

```
    foreach < sub-voxel > do                          Continued from previous page
        < fill vmin and vmax >
        if depth < maxDepth then
            voxelType := get_voxel_type(vmin, vmax, ref sphereNr);
            if voxelType = backgroundVoxel then
                sum := sum + (weight · background);
            else
                if voxelType = sphereVoxel then
                    sum := sum + (weight · spheres[sphereNr].g);
                else                                                    recursion!
                    sum := sum + (weight · split_voxel(x, y, z, depth + 1);
                fi
            fi
        else
            sum := sum + (weight · midpoint_value(x, y, z);
        fi
    od
    return [sum];
end
```

Finally, the function *midpoint_value* determines if the midpoint of a (sub-)voxel is inside or outside any sphere. The sphere with the highest priority is checked first, and if the midpoint is outside that sphere the next one is checked for, and so on. Thus, the grey value of the sphere with the highest priority that contains the midpoint is returned (or the background value if no such sphere exists).

```
Grey function midpoint_value(midX, midY, midZ);

begin
    for n := 1 to nrOfSpheres do
        if (midX − spheres[n].x)^2 + (midY − spheres[n].y)^2+
            (midZ − spheres[n].z)^2 ≤ spheres[n].r2 then
            return [spheres[n].g];
        fi
    od
    return [background];
end
```

3.5 Accuracy considerations

The number of bits allocated to the Grey data type limits the maximum depth that will still increase the accuracy. This leads to the following equation for the maximum depth d_{max}:

$$d_{max} \;=\; \left\{ \min d \;\middle|\; 2^{(d-1)\cdot D} \;+\; 1 \;\geq\; 2^{b} \right\} \tag{3.1}$$

with D the number of dimensions and b (≥ 1) the number of bits used for the Grey data type. One can easily derive that (for $b > 1$ only) the following—more practical—formula is valid:

$$d_{max} \;=\; \left\lceil \frac{b}{D} \;+\; 1 \right\rceil \;. \tag{3.2}$$

A simple and efficient way to calculate the error made in the discretization is to use one single sphere. Then, the theoretical volume of the sphere can easily be compared with the output image. Suppose the background has grey value 0 and the sphere 100, then the following expression is a measure of the discretization error:

$$\text{error} \;=\; \frac{\left| \frac{sum}{100} \;-\; \frac{4}{3}\,\pi\, r^3 \right|}{\frac{4}{3}\,\pi\, r^3} \;\cdot\; 100\% \tag{3.3}$$

with *sum* equal to the summation of all the image intensities. Typical values for the error-term are less than 0.1% at depth 3. Only very specific applications, such as sophisticated volume renderers or quantitative measurements, may require this kind of accuracy.

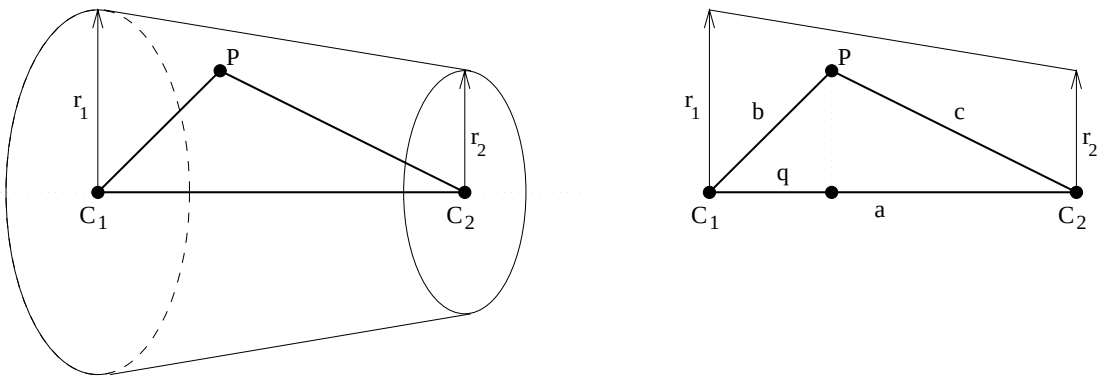
3.6 Extension to other objects

Extension of spheres to ellipsoids is trivial. Arvo [6] already presented pseudo-code for extension of the voxel-sphere intersection test to ellipsoids.

More difficult is the use of truncated cones and cylinders (in fact a special case of a truncated cone). One of the problems is that the used voxel-sphere intersection test does not provide a solution for testing on cones. Here we propose a simple test, based on the checking of all eight corners of a voxel being inside or outside the cone. The caveats as described by Shaffer [110] may still occur, but are minimized if relatively large cones are used. The inside-outside test is illustrated in Fig. 3.2.

Fig. 3.2a contains a truncated cone defined by center point C_1 with radius r_1, and center point C_2 with radius r_2. The point to be tested, representing one of the voxel corners, is denoted by P. The whole calculation focuses on the 2D plane formed by C_1, C_2 and P (see Fig. 3.2b).

First, it is checked whether one of the angles PC_1C_2 or PC_2C_1 is obtuse-angled. If so, P can never lie inside the cone. If not, the perpendicular h_P and the distance q are calculated by means of the edges a, b and c. Finally, a linear interpolation step



(a) (b)

Fig. 3.2. *Schematic representation of the inside-cone test.*

between r_1 and r_2 determines the maximum radius r_P at P, which is compared with the actual value of h_P to find its position with respect to the cone. The pseudo-code of the method follows.

```
boolean function cone_test(C1, C2, P : Point3D; r1, r2 : real);

a, b, c, hP, rP, q, s : real;                                  local variables

begin
    < fill local variables >
    if (b^2 > a^2 + c^2) or (c^2 > a^2 + b^2) then
        return [false];                                            outside
    fi
    q := (a^2 + b^2 - c^2)/2a;
    s := (a + b + c)/2;
    hP := (2/a) · sqrt(s (s − a) (s − b) (s − c));
    rP := r1 + (q/a) · (r2 − r1);                          interpolation step
    if hP > rP then
        return [false];                                            outside
    else
        return [true];                                             inside
    fi
end
```

For the cylinder case (i.e. $r_1 = r_2$) the interpolation step can simply be omitted. Then, r_P equals r_1.

3.7 Examples

In this section we present some examples of test images created by the volumetric modeling program (which is called 'THINGS'). In Fig. 3.3 a series of volume renderings of three basic objects created by THINGS is shown. The depth used is 1, which explains the artifacts at the surfaces of the objects. The figure can be compared with Fig. 3.4, which is created by using a recursive depth of 3. The objects are smooth and contain no artifacts anymore.

Fig. 3.3. *Volume rendering of three simple objects created by* THINGS*, with a recursive depth of 1.*

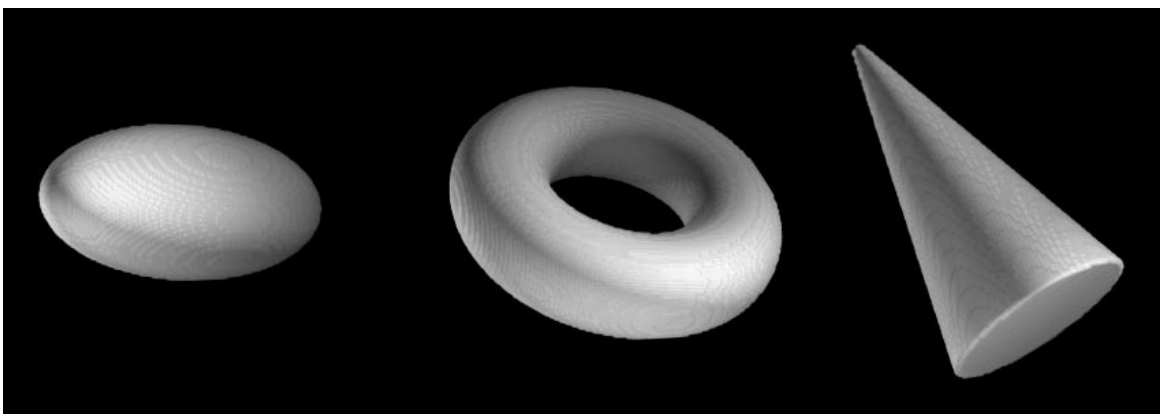


Fig. 3.4. *Volume rendering of three simple objects created by* THINGS*, with a recursive depth of 3.*

By using the priority rule, more complex objects can be created. This is shown in Fig. 3.5. All these objects have been created with a recursive depth of 3, which explains the smooth appearance.

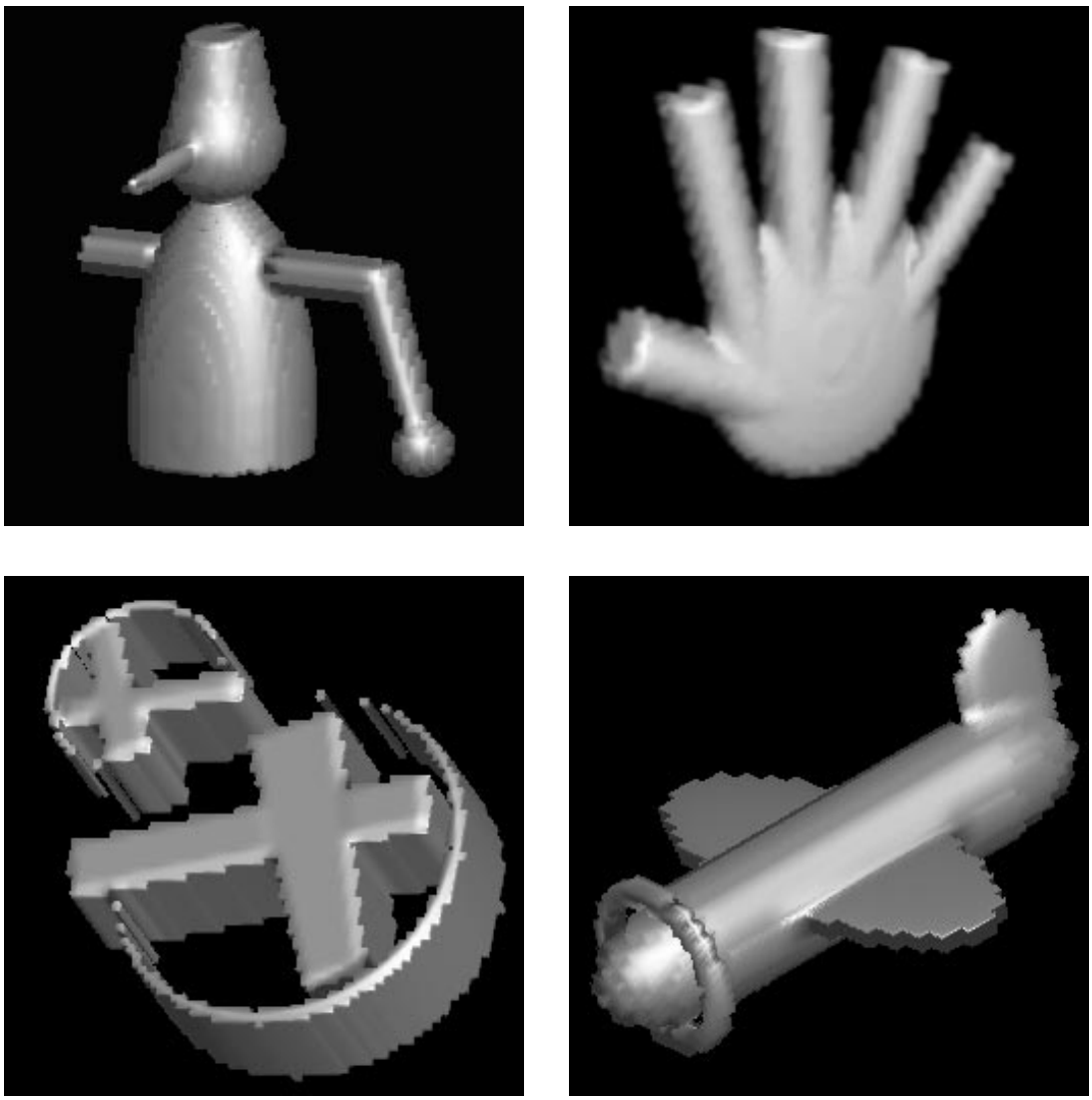


Fig. 3.5. *Volume rendering of four more complex objects, created by* THINGS *using a recursive depth of 3.*

'If you want to study the brain and consciousness, you have to study the brain and consciousness, not computers'

– John Searle
(De Volkskrant, 14 May 1994)

Chapter 4

Probabilistic image segmentation

Abstract

A method is presented to segment multi-dimensional images using a multiscale (hyperstack) approach with probabilistic linking. A hyperstack is a voxel-based multiscale data structure whose levels are constructed by convolving the original image with a Gaussian kernel of increasing width. Between voxels at adjacent scale levels, child-parent linkages are established according to a model-directed linkage scheme. In the resulting tree-like data structure *roots* are formed to indicate the most plausible locations in scale space where segments in the original image are represented by a single voxel. The final segmentation is obtained by tracing back the linkages for all roots.
The present chapter deals with probabilistic or multi-parent linking, *i.e.,* a set-up in which a child voxel can be linked to more than one parent voxel. The multi-parent linkage structure is translated into a list of probabilities that are indicative of which voxels are partial volume voxels and to which extent. The output of the thus constructed hyperstack can be directly related to the opacities used in volume renderers. It is demonstrated by means of artificial images as well as real world (medical) images that probabilistic linking gives a significantly improved segmentation as compared with conventional (single-parent) linking.
Furthermore, *probability maps* are generated to visualize the progress of weak linkages in scale space when going from fine to coarser scale. This is shown to be a valuable tool for the detection of voxels that are difficult to segment properly.

Keywords: Image segmentation, multiscale analysis, scale space, probability maps, partial volume artifact, data structures.

4.1 Introduction

Segmentation of volumetric image data plays a crucial role in image processing, in particular as a preprocessing step for quantitative analysis and volume visualization.

In the last decade, multiscale approaches (like pyramid [16, 24, 23, 73, 11], stack [54, 86, 64] and wavelet [40, 21, 70]) segmentation have gained considerable attention.

Recently, we have developed a flexible data structure—the hyperstack—for multiscale segmentation of two- and three-dimensional (2D, 3D) images [116, 118, 25], which admits of extensions like *outer scale reduction* and *probabilistic linking*. Outer scale reduction—reducing the number of voxels as the scale increases—will speed up the process of building a hyperstack; this subject will be treated in a Chapter 5. Probabilistic linking—the subject of the present chapter—is introduced with the aim of improving the tuning of the segmentation to the subsequent rendering of volumetric structures. In this way the jaggedness of objects which were incorrectly segmented by a binary decision procedure, is sought to be reduced.

We first briefly discuss the design of the conventional (single-parent) hyperstack, which is characterized by the fact that a voxel at some level of the hyperstack is connected to at most one (parent) voxel in the next higher layer. We then discuss probabilistic (multi-parent) hyperstacks and pay special attention to the explosive growth of the number of linkages if no precautions are taken. It will be shown that this problem can be solved without significantly affecting the quality of the segmentation.

Finally, we show how conventional-like segmentations can be derived from the output of probabilistic hyperstacks, and why these segmentations are more robust and of better quality than those obtained by conventional (single-parent) hyperstacks. We will also present some segmentation results of probabilistic hyperstacks to show the surplus value of multi-parent linking over single-parent linking.

4.2 The conventional hyperstack

In the sequel we use the term *level* to denote an image at a specific scale. The original image (or ground level) corresponds with level 0, the top level—representing the most strongly blurred image—is at level $L-1$. A hyperstack thus contains L levels at increasing scale. The terminology in this chapter will generally apply to 3D images, so we will call the elements of the hyperstack *voxels* rather than pixels. The set-up is equally valid for 2D images, however.

Building and employing a hyperstack consists of four different steps (see Fig. 4.1):

1. *Blurring.* Images at increasingly larger scales are obtained by convolving the original image with Gaussians of increasing width.

2. *Linking.* Child-parent linkages are established between voxels at adjacent scale levels.

3. *Root labeling.* Voxels having weak parent linkages *and* all voxels in the top-most level are marked as *roots*.

4. *Downward projection.* The original image is segmented by tracing back the linkages from the roots to the ground level.

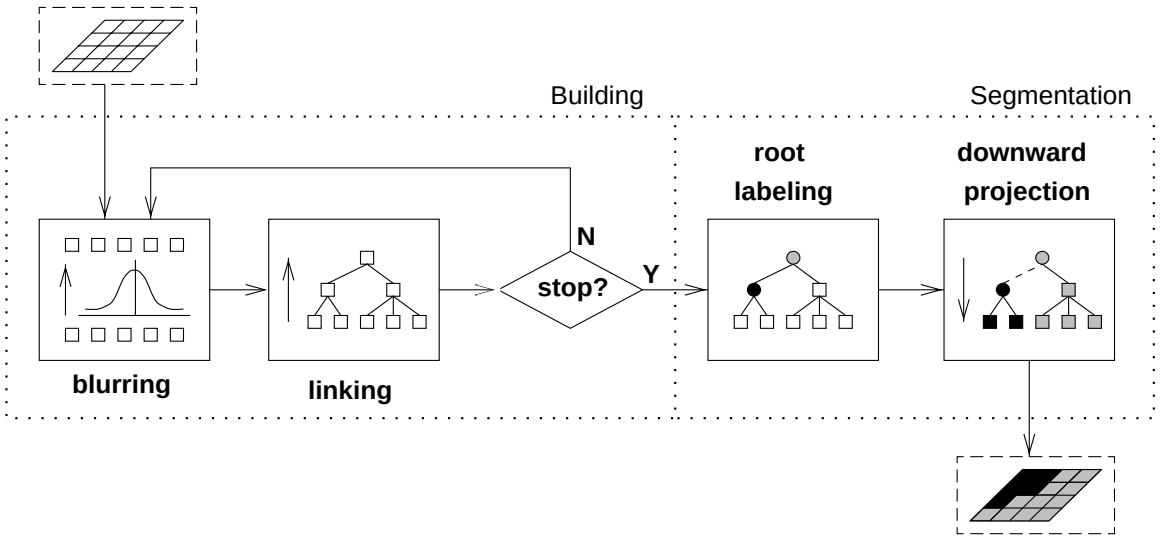


Fig. 4.1. *Schematic overview of the hyperstack image segmentation process.*

The following subsections deal with these steps in detail.

4.2.1 Blurring

The construction of a continuous linear scale space follows a blurring strategy, which is essentially a repeated low-pass filtering using a Gaussian kernel [131, 54]:

$$L(\vec{x}, \sigma_0 \oplus \sigma) \;=\; L_0(\vec{x}, \sigma_0) \;*\; G(\vec{x}, \sigma) \;, \tag{4.1}$$

where $L_0(\vec{x}, \sigma_0)$ is the luminance or intensity of the original image, $G(\vec{x}, \sigma)$ is the Gaussian with width σ of corresponding dimension, and $L(\vec{x}, \sigma_0 \oplus \sigma)$ is the σ-blurred replica of the input image. Note that the additive operator "$\oplus$" does not correspond to ordinary addition, but follows the *semi-group* property: $\sigma_0 \oplus \sigma = \sqrt{\sigma_0^2 + \sigma^2}$.

A discrete scale space is best constructed by convolution of the original image with sampled Gaussian kernels of increasing width. The alternative—building the scale space by convolving level i with a Gaussian to obtain level $n+1$ for $n = 0, ..., L-1$—has the disadvantage that a concatenation of sampled Gaussians is *not* a discrete scale space transformation [66].

As for scale space sampling, an equidistant sampling of the absolute scale σ would violate the important property of scale invariance [33]. Instead, the sampling should follow a linear and dimensionless scale parameter $\delta\tau$, which is related to σ by:

$$\sigma_n \;=\; \varepsilon\, e^{\tau_0 + n \cdot \delta\tau} \;, n \;\in\; \mathbb{N} \;. \tag{4.2}$$

In this equation ε is taken to be the smallest linear grid measure of the imaging device. A convenient choice for τ_0 is 0, which implies that the inner scale σ_0 of the initial image is taken to be equal to the linear grid measure ε.

4.2.2 Linking

In the linking step, voxels may be connected to voxels in the next-higher level (child-parent linking). We discriminate between *active* and *passive* voxels. Passive voxels do not participate in the linking process. At the ground level, passive voxels are voxels without parents; they may be introduced to decrease the number of voxels to be processed, notably to eliminate uninteresting background voxels. At the higher levels passive voxels are generated automatically: all voxels that have not been chosen as a parent of at least one child are considered passive.

In the conventional hyperstack [116, 118, 25], every child is linked to exactly one parent. In the probabilistic hyperstack [121, 123, 124], a child voxel can have more than one parent (see section 4.3.1).

Research has shown that a general and robust linking scheme should consist of three components [58]: (*i*) the intensity difference between a child and its parent, (*ii*) the ground volume of a parent, and (*iii*) the mean ground volume intensity. See section 2.2.2 for details.

4.2.3 Root labeling

Voxels are labeled as roots *after* all child-parent linkages have been established. The main reason for avoiding root labeling during the linking process is that we want to have control over number of roots at the segmentation stage. Reducing the number of roots (which implies increasing the number of linkages) *after* the completion of the hyperstack is not only a laborious task, but also a dubious effort: owing to the ground volume component in the affection formula linkages can not be added without recalculating all the other linkages in that level and higher. Ignoring this fact may very well lead to 'false' links, and hence to a lower accuracy of the segmentations in general. Consequently, every voxel in the hyperstack is linked to a parent, except for the voxels in the top level. After the top level has been appended, the roots can be defined.

For a detailed description of the conventional root labeling process, we refer to section 2.2.3. The process of root labeling will become much more complex in the case of probabilistic linking (see section 4.3.3).

4.2.4 Downward projecting

In the last step, downward projection, intensity values that are characteristic of the roots are projected downwards to the ground level. This requires—besides the choice of a lowest root level—the definition of a level from which to start the projection. All parents in this *segmentation level* are considered roots. A higher segmentation level implies—owing to the fact that the number of parents decreases at larger scales—a smaller lower bound for the minimum number of segments to be found.

The number of segments and their different sizes are thus influenced by the choice of the lowest root level and the segmentation level, and by the actual root labeling.

In the conventional hyperstack there are three possibilities for calculating a segment value (*i.e.,* the actual intensity value given to all the voxels of one segment): (*i*) the root intensity, (*ii*) the mean intensity of the ground voxels, and (*iii*) a unique (pseudo-)value; for the probabilistic hyperstack a fourth possibility (the probability value) exists, which will be discussed in section 4.5.

Root values are blurred intensities and thus will often produce low-contrast segmentations. (Normally, the global intensity extrema of an image converge approximately linearly towards the average intensity of the largest scale, provided the scale space is sampled according to equation (4.2).) The mean intensities of sets of ground voxels are less dependent on the scales at which the roots reside, which results in a higher contrast in the thus produced segmentations (at the cost of a little more computing time).

In some cases it is desirable to know the contours of segments, *e.g.,* in the case where two adjacent segments with the same intensity (after downward projection) need to be distinguished. In such cases, it is not sufficient to use root- or mean intensity values for the down projection, although they give an acceptable output for visual examination of the results. The distinction of segment values may be accomplished by giving each segment a unique number. The use of pseudo-colored overlays can help visualizing the different segments.

4.3 The probabilistic hyperstack

After having discussed the conventional hyperstack, we now turn to probabilistic hyperstacks in which every child voxel is allowed to connect to more than one parent.

4.3.1 Probabilistic linking

Instead of forcing a binary decision to which parent a child should be linked, we simply link a child to all parents that are 'good enough', according to some objective criterion. This minimizes the chance that a crucial link is missed (which may happen if a worse parent is preferred because of noise); higher up in the hyperstack the 'mistake' will be corrected automatically. This is an important feature of probabilistic linking. Note that—since the noise will disappear if the scale is increased—the chance that a crucial link is missed owing to noise will be negligible in the higher levels. We will use this observation when discussing the complexity of the hyperstack in detail (see section 4.4).

Once all interesting linkages have been established, the corresponding probabilities are found simply by normalization of the affection values, since the sum of the linkage probabilities from a child to all of its parents must equal one.

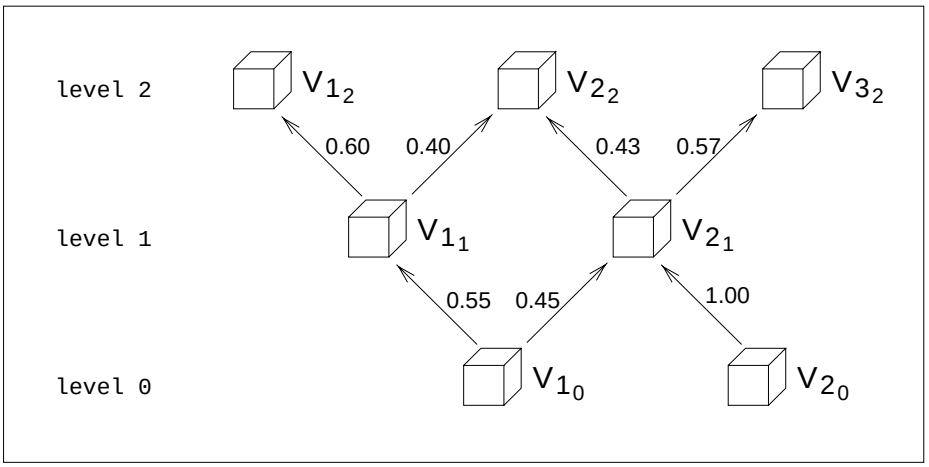


Fig. 4.2. *Example of multi-parent linking in a hyperstack.*

Fig. 4.2 shows an example of multi-parent linking. Here, and in the sequel of this chapter, we denote by $P(V_{i_{l+1}} \mid V_{i_l})$ the conditional probability that voxel i at level $l+1$ is the parent of child voxel i at level l. (Note that a voxel is indexed *both* by the level index (l) and the grid index (i). Consequently, voxel i at level l is not necessarily related to voxel i at level $l+1$.) Child voxel V_{1_0}—meaning: the child with index 1 at level 0—is linked to two parents at level 1: V_{1_1} and V_{2_1}. The corresponding child-parent probabilities are $P(V_{1_1} \mid V_{1_0}) = 0.55$, and $P(V_{2_1} \mid V_{1_0}) = 0.45$. If a child links just to one parent, the child-parent probability takes the value 1.

In appendix 4.A we discuss the consequences for the data structure of extending the hyperstack from single-parent to multi-parent linking, and in appendix 4.B we deal with the implementation of the linkage tree by means of so-called *link containers*.

4.3.2 The ground volume under multi-parent linking

In conventional hyperstacks, the ground volume of a voxel is the number of ground voxels with a route to that voxel. The same definition is not appropriate for probabilistic hyperstacks. Instead, the voxels encompassed in the ground volume are weighted by the probabilities of the linkage paths. For example, the ground volumes of parents V_{1_2}, V_{2_2}, and V_{3_2} in Fig. 4.2 have values 0.33, 0.8435 and 0.8265, respectively.

4.3.3 Root labeling under multi-parent linking

Each of the two paradigms for root labeling—discussed in section 2.2.3 for conventional hyperstacks—must be adapted to multi-parent linking:

- The threshold value on the adultness to determine which voxels have to be labeled as roots must be applied to the *strongest* link of every child.

- A specified number of roots is no longer necessarily identical to the number of segments after downward projection. For instance, if only the highest probability path of a linkage tree is used to segment the image, several roots may end up not having a ground volume. Thus, only an upper bound for the number of segments can be specified with this method.

The eventual *root probabilities* (obtained as a result of the root labeling) represent the chances for a ground voxel to belong to various segments. In an equivalent interpretation these probabilities represent the fractions of segments that are contained in a ground voxel. The result is comparable to the output of fuzzy image subsets (see [94]) in which a degree of membership is associated with every voxel.

The root probabilities are computed by following all the linkages that connect a ground voxel to different roots. The child-parent probabilities are multiplied for each path, and the 'path probabilities' are added to find the final root probability, denoted by $\mathrm{P}(\mathrm{V}_{i_l} \,|\, \mathrm{V}_{i_0})$, where l is the level at which the root is defined.

The root probabilities can be calculated from the recursive relation

$$\mathrm{P}(\mathrm{V}_{i_l} \,|\, \mathrm{V}_{i_0}) \;=\; \sum_{i_{l-1}=1}^{N_{l-1}} \mathrm{P}(\mathrm{V}_{i_l} \,|\, \mathrm{V}_{i_{l-1}}) \cdot \mathrm{P}(\mathrm{V}_{i_{l-1}} \,|\, \mathrm{V}_{i_0}) \;, \tag{4.3}$$

with $2 \le l \le L-1$, $L \ge 3$, and $1 \le i_k \le N_k$ for all $0 \le k \le L-1$; N_k is the number of active voxels at level k. Equation (4.3) can be written as

$$\mathrm{P}(\mathrm{V}_{i_l} \,|\, \mathrm{V}_{i_0}) \;=\; \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \cdots \sum_{i_{l-1}=1}^{N_{l-1}} \left\{ \prod_{j=1}^{l} \mathrm{P}(\mathrm{V}_{i_j} \,|\, \mathrm{V}_{i_{j-1}}) \right\} \;. \tag{4.4}$$

Note that the following expression is also valid for all $1 \le i_{l-1} \le N_{l-1}$ (normalization property):

$$\sum_{i_l=1}^{N_l} \mathrm{P}(\mathrm{V}_{i_l} \,|\, \mathrm{V}_{i_{l-1}}) \;=\; 1 \tag{4.5}$$

The root probabilities between voxels which are *two* levels apart can be found from equation (4.4). Using the affections as given in Fig. 4.2, we find

$$\begin{aligned}
\mathrm{P}(\mathrm{V}_{1_2} \,|\, \mathrm{V}_{1_0}) &= \mathrm{P}(\mathrm{V}_{1_2} \,|\, \mathrm{V}_{1_1}) \cdot \mathrm{P}(\mathrm{V}_{1_1} \,|\, \mathrm{V}_{1_0}) \;=\; 0.33 \\
\mathrm{P}(\mathrm{V}_{2_2} \,|\, \mathrm{V}_{1_0}) &= \mathrm{P}(\mathrm{V}_{2_2} \,|\, \mathrm{V}_{1_1}) \cdot \mathrm{P}(\mathrm{V}_{1_1} \,|\, \mathrm{V}_{1_0}) + \mathrm{P}(\mathrm{V}_{2_2} \,|\, \mathrm{V}_{2_1}) \cdot \mathrm{P}(\mathrm{V}_{2_1} \,|\, \mathrm{V}_{1_0}) \;=\; 0.4135 \\
\mathrm{P}(\mathrm{V}_{3_2} \,|\, \mathrm{V}_{1_0}) &= \mathrm{P}(\mathrm{V}_{3_2} \,|\, \mathrm{V}_{2_1}) \cdot \mathrm{P}(\mathrm{V}_{2_1} \,|\, \mathrm{V}_{1_0}) \;=\; 0.2565
\end{aligned}$$

From Fig. 4.2 it follows that segmentations obtained using the strongest linkage path in multi-parent linking may differ from segmentations obtained with single-parent linking. In the latter, only the parent with the *locally* highest affection is included in the data structure, which would make V_{1_2} the grandparent of V_{1_0}. Multi-parent linking identifies two voxels, V_{1_2} and V_{2_2}, as possible grandparents. If only the strongest linkage path is considered, voxel V_{2_2} is the most probable grand parent of V_{1_0}.

The output of a probabilistic hyperstack is a list of normalized probabilities per (ground) voxel. These probabilities represent the chances that a voxel belongs to the listed segments. To obtain actual segmentations from lists of probabilities, as provided by probabilistic hyperstacks, we have several possibilities. In section 4.5 some methods are discussed and applied to actual images, while also the most important advantages of probabilistic over conventional segmentations will be summarized.

4.3.4 Probability maps

Before actually applying a probabilistic hyperstack, we first want to understand how linkages evolve in scale space. The number of linkages is enormous, which makes it hard to follow and evaluate each of them. Therefore, we developed a way to visualize the progression of weak and strong linkages.

The idea is to produce so-called *probability maps* by displaying for every ground voxel the largest probability that it belongs to a root in a specific level. This simple algorithm first chooses a level l to create a map for, then turns all the active voxels in level l into roots, and finally calculates the highest root probabilities for every ground voxel by scanning the entire hyperstack. A linear remapping of these probabilities onto a range of regular image intensities, produces the desired result: dark voxels represent areas of low probability, whereas brighter voxels correspond to high probabilities.

It might be expected that edge voxels are harder to link than voxels in near-homogeneous volumes, for two reasons:

- *Partial volume effect.* An edge voxel will have, apart from possible noise components, a value proportional to the volumes of the different object types that are represented in that voxel. Thus, statistically the value of an edge voxel will lie in between the values of the surrounding object values. Since the intensity difference in equation (2.12) does not explicitly prefer smaller or larger values, there will not be a preference for either of the objects.

- *Blurring strategy.* The blurred value of a voxel is determined by a weighted sum of the values of the voxel at hand and the values of its neighbors. Thus, edge voxels will be subjected to larger intensity changes in scale space than non-edge voxels.

Hence, it is plausible that voxels near object edges will generally have links to more parent voxels than voxels in homogeneous areas.

In Fig. 4.3 a series of probability maps in scale space is shown for the HEAD image, a sagittal MR image of the brain. One can clearly see the distribution of the largest probabilities evolving in the hyperstack. The edges of the objects of different sizes are best visible at the scale at which they can be represented by a single root. Logically, there is a high correspondence between these levels and the levels found in edge detection algorithms based on scale space [17]. For instance, the edges of the

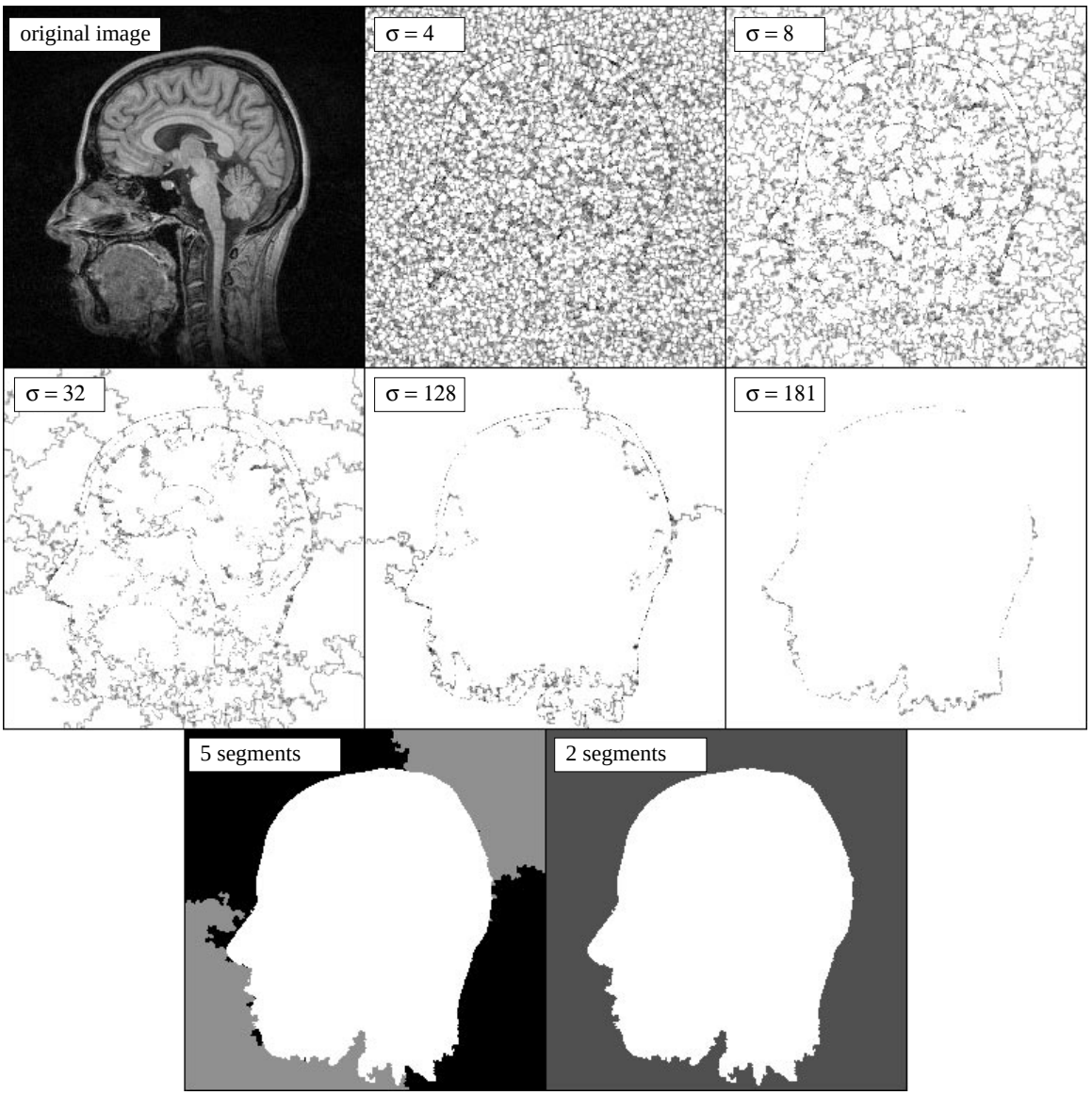


Fig. 4.3. *Probability maps of a hyperstack based on 15 blurred levels of the two-dimensional* HEAD *image. Shown are: the original MR image (with dimension sizes 256 × 256), five probability maps (that correspond with level 4, 6, 10, 14 and 15 of the hyperstack, respectively), and finally two coarse segmentations (containing a predefined number of 5 and 2 segments, respectively). Note the similarity between the last two probability maps and the two segmented images (see text).*

ventricle are noticeable at the middle levels (cf. $\sigma = 32$), but merge with other objects at the higher levels.

At one but the highest level shown ($\sigma = 128$) the probability map contains five separate fragments: one for the head object and four for the background. The reason for the side contours to be on the top, at the bottom, at the left and at the very right of the image is that those pixels have great difficulty choosing between two near background parts. At a higher level still ($\sigma = 181$), the background merges into a single segment. A segmentation made with a predefined number of segments that matches the number of segments in a probability map, bears a close resemblance with this map, as the lower frames of Fig. 4.3 show convincingly.

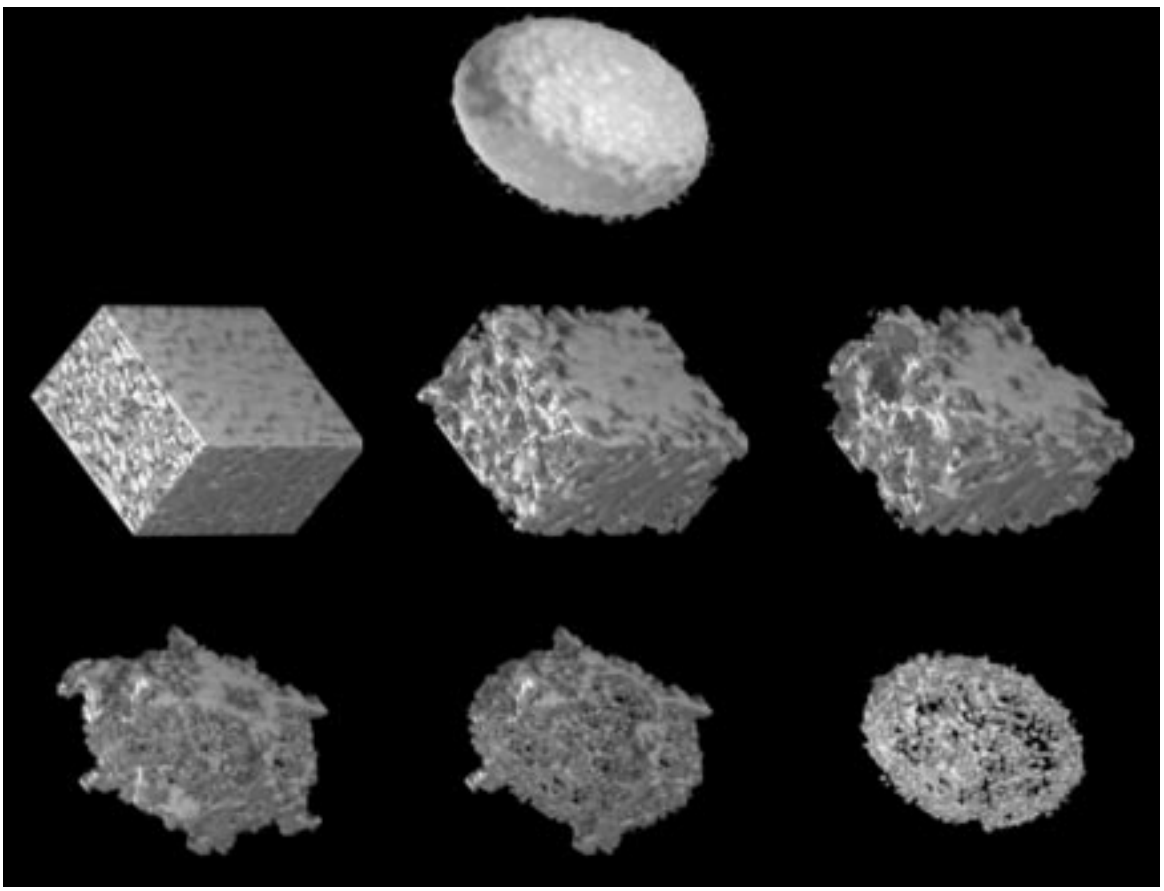


Fig. 4.4. *Volume renderings of 3D probability maps of the artificial* ELLIPSOID *image. Shown are a rendering of the original image (top picture), followed by 6 renderings of inverted probability maps (for an explanation see text).*

In 3D a probability map is generated in the same way as for the 2D case. Visualization of the result in a slice-by-slice manner, however, is not very useful for recognizing edge-like structures. To produce a suitable 3D presentation we invert the

intensities of the probability maps such that high intensities correspond to low root probabilities and vice versa. The result can then be piped to a volume renderer to visualize the low root probabilities at the outer edges of the image structure. In Fig. 4.4 this has been done for the ELLIPSOID image containing one ellipsoid. The successive renderings represent (from left to right and from top to bottom) the probability maps at increasing segmentation levels. To emphasize the weakness of the linkages corresponding to edge voxels, we increased the lowest root level (this forces all voxels to keep linking upwards). Indeed, the last probability map clearly shows the edge of the ellipsoid as formed by the collection of voxels with a low root probability.

4.4 Computational complexity

The complexity of probabilistic hyperstacks is mainly determined by the total number of linkages; if no precautions are taken, this number grows explosively. Thus, in order to keep the number of linkages limited, we need to introduce some constraints.

We have used two limits while building a probabilistic hyperstack:

1. an upper bound for the number of parent linkages per child
2. a lower bound for the affection, below which no linkages are established (provided that at least one link per active voxel is present)

Since either constraint has great influence on the actual number of linkages formed, we will explain them in more detail below. Note that during the linking process, care must be taken that at least one link per child is established, so as to avoid premature creation of roots. Section 4.3 presents some results on the number of linkages established with and without the constraints.

Another factor that influences the computational complexity of a hyperstack, though to a lesser extent, is scale space sampling. Using only a few levels will also keep the amount of linkages low. The minimum scale space sampling rate to avoid deterioration of the quality of the segmentation will be application dependent. Research is due to optimize the sampling rate for classes of applications.

In practice, it turns out that it is not necessary to blur until a scale is reached where only one intensity is left (an average image intensity value). The number of levels used will generally lie between 10 and 20.

4.4.1 A maximum number of parents per child

Owing to noise present at the smaller scales, the maximum number of parent linkages per child should be relatively high when linking the lowest levels, and may decrease when linking levels at larger scales. A similar argument holds for the partial volume voxels in the lower levels (see section 4.3.4 about the probability maps). Thus, we need to allow the lowest levels more links than those higher up in scale space.

If we try to determine the maximum number of parents for every child voxel at the ground level without *a priori* information on the image to be segmented, the number of dimensions plus one seems a reasonable choice. This choice is motivated by the partial volume notion and is based on the fact that it is unlikely that on a two-dimensional map four different areas join together in one point, whereas 3-junctions (*i.e.,* three joining areas) are much more common. Similar considerations lead to a maximum of four in the three-dimensional case. The diminishing influence of both the noise component and the partial volume effect at increasing scale make it acceptable to decrease the maximum number of parents per child with one at every scale step. This decrease is effectuated after level 1 has linked to level 2.

4.4.2 A lower bound for the affection

The linkage strength increases when linking at larger scales, for ground volumes are steadily growing and the intensity differences will diminish. Consequently, the lower bound for the affection should increase accordingly.

When searching for a suitable lower bound for the affection of linkages, it is attractive to define a uniform lower bound per level. A simple calculation—for instance based on the affection range of the first iteration—can then serve as initial estimation for the lower bound. Moreover, the need to recalculate the limit per child, or even per iteration, is absent.

Implementation of this paradigm, however, showed an annoying side-effect: owing to the fact that between any two levels there is a rather high variance in affection values (especially at smaller scales), voxels in areas with a relatively low affection range (edges) are assigned one single link, the minimum amount. But probabilistic linking has been invented to give better segmentation results precisely in those areas! Consequently, we dropped this 'solution'.

Much better results are obtained by defining an adaptive lower bound of the affection, despite the amount of work involved. The idea is to define a parameter, called the *minimum relative affection*, that specifies how much the affection of a child-parent link may differ from the highest affection present for that child. In order to avoid absolute differences here, we choose to set a relative value (*e.g.,* 95%). Intuitively, this leads to the desirable effect that voxels with a strong preference for a specific parent do not need any other links, whereas children *in dubio* are assigned more than one parent.

4.4.3 Results on constrained linking

In order to evaluate the effects of constraining the number of linkages, we built five types of hyperstacks with different parameter settings, and compared the number of linkages evolving in the first 14 levels for:

a. a traditional hyperstack with single-parent linking

b. a multi-parent hyperstack with a minimum relative affection value of 95% and initially a maximum of 3 parents per child

c. a multi-parent hyperstack with a minimum relative affection value of 0% and initially a maximum of 3 parents per child

d. a multi-parent hyperstack with a minimum relative affection value of 95% and continuously a maximum of 3 parents per child

e. a multi-parent hyperstack with a minimum relative affection value of 0% and continuously a maximum of 3 parents per child

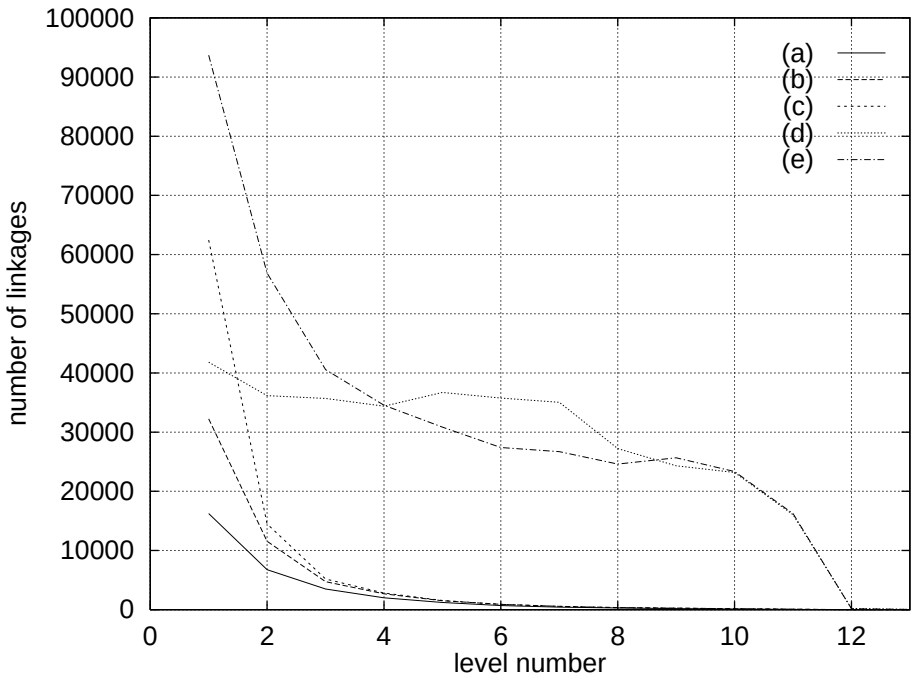


Fig. 4.5. *Scale dependence of the number of links for five different types of hyperstacks based on the* HEAD *image. For an explanation of the different types (a) up to (e) see text.*

The results for the HEAD image of Fig. 4.3 are presented in Fig. 4.5. The hyperstack has been created according to equation (4.2) with a $\delta\tau$ value of $\frac{1}{2}\ln 2$; this results in a hyperstack of 15 levels.

From Fig. 4.5, it follows that the constraints we impose on the linking procedure are effective. Setting the minimum relative affection to 95% (d) limits the number of linkages in the lower levels in the hyperstack. Using no lower bound (e) results in a significantly higher number of linkages.

Applying the constraint that limits the maximum number of parents per child at increasing scale (c) has even more effect (again compared to (e)): irrespective of the

minimum relative affection used, the hyperstacks quickly converge to single-parent variants (a). Finally, applying both constraints is most useful (b).

4.5 Results

The presentation of the results—lists of tissue probabilities—is hampered by the unavailability of suitable volume rendering software. Yet, we can compare images segmented by multi-parent hyperstacks with conventional segmentations by selecting one segmented image from an ensemble of possible realizations. The natural option is to use the highest object probability of each list. In section 4.3.1 we have indicated why segmentations thus obtained may be expected to outperform segmented images from single-parent hyperstacks.

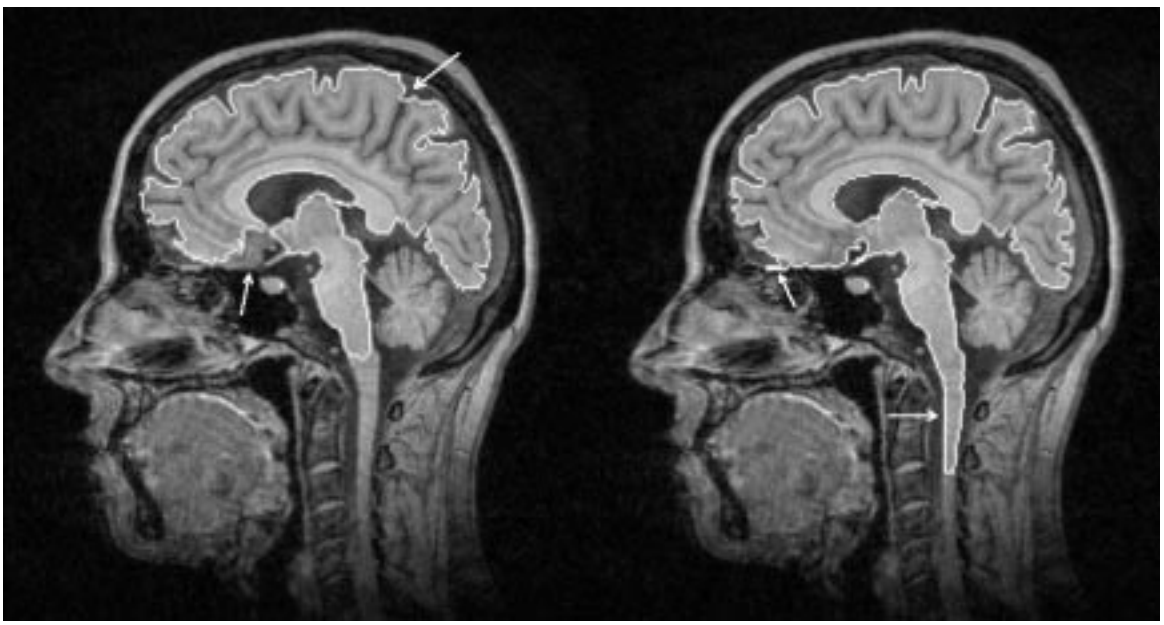


Fig. 4.6. *Two-dimensional segmentation of the sagittal* HEAD *image. Contours have been superimposed (in bright white) on the original MR image. Single-parent segmentation (left), multi-parent segmentation (right).*

Fig. 4.6 shows a comparison of a segmentation using the single-parent hyperstack with a multi-parent segmentation, in which each voxel is represented by the highest object probability of its list. The arrows emphasize the main differences between the two segmentations. The multi-parent hyperstack clearly performs better than the single-parent hyperstack.

The probabilistic hyperstack used contained 17 levels (with constrained linking and a minimum relative affection of 0.90), while the lowest root level was set to 6. The contours have been found by simple thresholding of the segments formed by downward

projection of mean segment values. We emphasize that *no* additional editing has been performed on the segmentations.

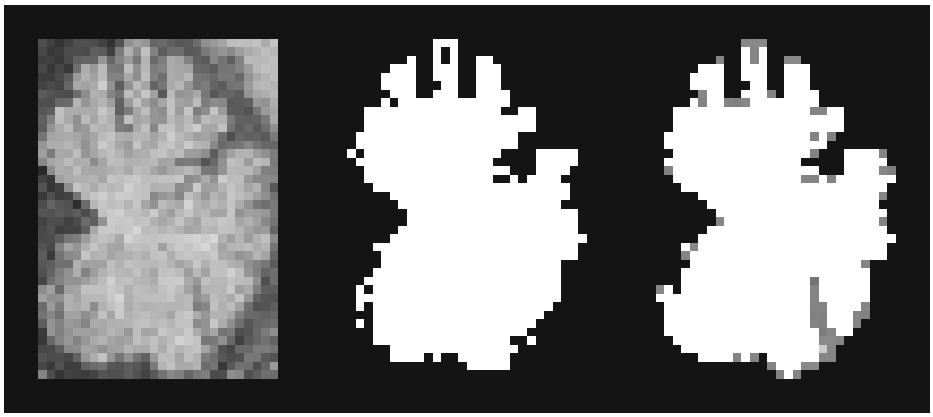


Fig. 4.7. *Segmentation of the cerebellum of the* HEAD *image. Original image (left), single-parent segmentation (middle) and a probabilistic result (right).*

A second way to visualize probabilistic results is to focus on one object (organ or tissue type). The summed probabilities of the paths from a ground voxel to a root are indicative of the probability that this ground voxel belongs to the segment defined by the root. (Note that this value is not equal for all the members of one segment, in contrast with other downward projecting techniques.) The result is an image whose intensities are proportional to the amount of tissue contained in each voxel (see Fig. 4.7). Volumetric compositing methods [27, 63] can be applied to visualize this type of output (see also [87, 9, 139, 138]).

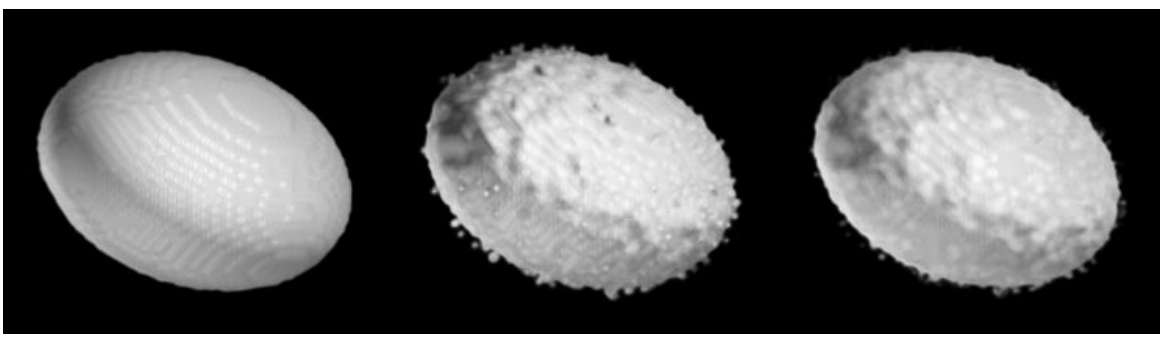


Fig. 4.8. *Renderings of the stylized* ELLIPSOID *image: original image (left), conventional hyperstack segmentation (middle) and probabilistic hyperstack segmentation (right).*

In Fig. 4.8 the same technique has been applied to the ELLIPSOID image. The (noise-free) input image was used to generate the first rendering, in order to show the desired result. The object intensity value is 2000, with a background value of

1000. The second rendering is based on segmentation by a single-parent hyperstack (after having added Gaussian noise with a standard deviation of 10% of the object intensity), and the third rendering is based on a multi-parent hyperstack. Note the notched edges in the middle image, owing to the single-parent segmentation. The probabilistic linking softens this effect (third image).

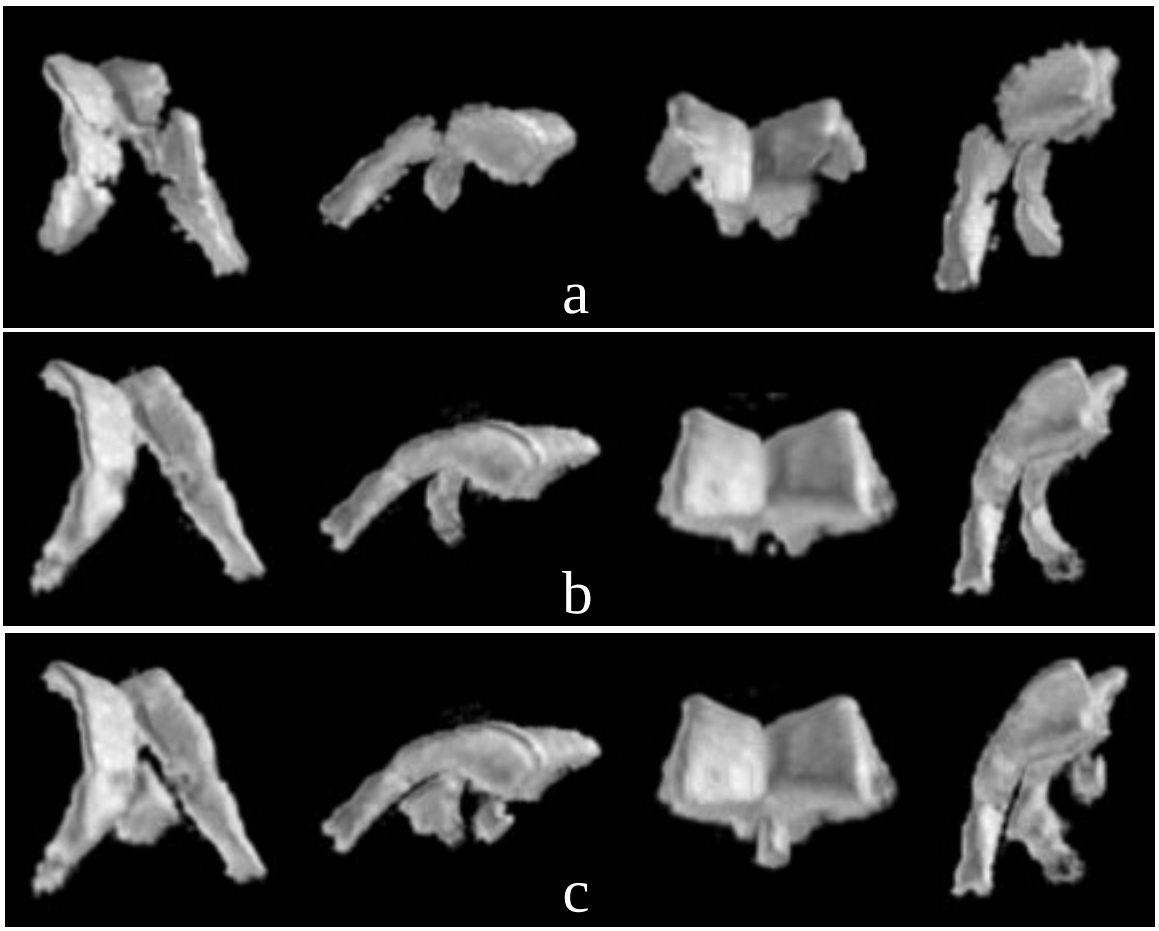


Fig. 4.9. *Renderings of (a) a single-parent and (b) a probabilistic segmentation of the* VENTRICLES *image, a 3D MR image of the brain. Including additional segments is very simple (c).*

Fig. 4.9 shows a series of results of techniques to segment the ventricles from a three-dimensional MR image of the brain (the VENTRICLES image). On the top row, renderings from different viewpoints are shown based on conventional (single-parent) hyperstack segmentations. On the second row, the probabilistic counterparts are shown. The third row also represents a probabilistic hyperstack, but now extended with two small additional segments in the middle of the ventricle. To obtain this result, we only had to increase the desired number of segments with two. Note that the single-parent segmentation has several serious shortcomings.

4.6 Conclusions and discussion

We have presented a method to segment images in a probabilistic way using multiscale based hyperstacks. We showed the surplus value of multi-parent linking over single-parent linking and described a data structure capable of handling such complex child-parent connections.

As regards the complexity of probabilistic hyperstacks, we indicated how the number of linkages involved easily grows prohibitively. Two constraints have been introduced to keep this number limited. Their adequacy has been demonstrated experimentally.

The display of probabilistic images calls for proper visualization software: it seems most appropriate to let the calculation of the opacities of the voxels depend on the knowledge of the partial volume voxels. Until this becomes available we can only visualize two-dimensional segmentations, or simulate intelligent renderers by *ad hoc* postprocessing. Images produced this way do allow us to sufficiently evaluate our segmentation results.

We are currently investigating what other strategies can be pursued to calculate the probability for each link. Interesting features are, *inter alia*, the number of parent/child linkages per voxel, the scale, and the used blurring strategy. It seems promising to extend the affection formula with these features.

We also expect probabilistic segmentations to have advantages over conventional segmentation schemes with respect to quantitative measurements. The accuracy of calculated distances and volumes heavily depends on the accuracy of the segmentation. Probabilistic segmentation introduces voxels that are only *partly* contained in segments, which increases the accuracy of quantitative measures significantly [123, 124].

Experiments to quantify and verify these expectations are currently in progress.

4.A Single-parent data structure

In this appendix we will outline the design of a suitable data structure.

The data structure to store single-parent linked hyperstacks [116, 118, 25] is not suited to be extended to multi-parent linking (see Fig. 4.10). The single-parent data structure is a variant of *doubly linked lists*, with the modification that one parent having multiple children does *not* have separate child references, but each child has a *sibling* reference instead. A sibling linkage points to the next child belonging to that parent, thus creating a chain of children. Together with every parent reference per child, a triangular data structure is formed.

The main problem in applying this data structure to multi-parent linking is that it is impossible to discriminate between multiple children of one parent and multiple parents of one child. Introducing a fourth (sibling-like) type of reference to connect all parents of a child would create ambiguity, because parents can be shared by different children. Besides, the implementation would become unacceptably expensive.

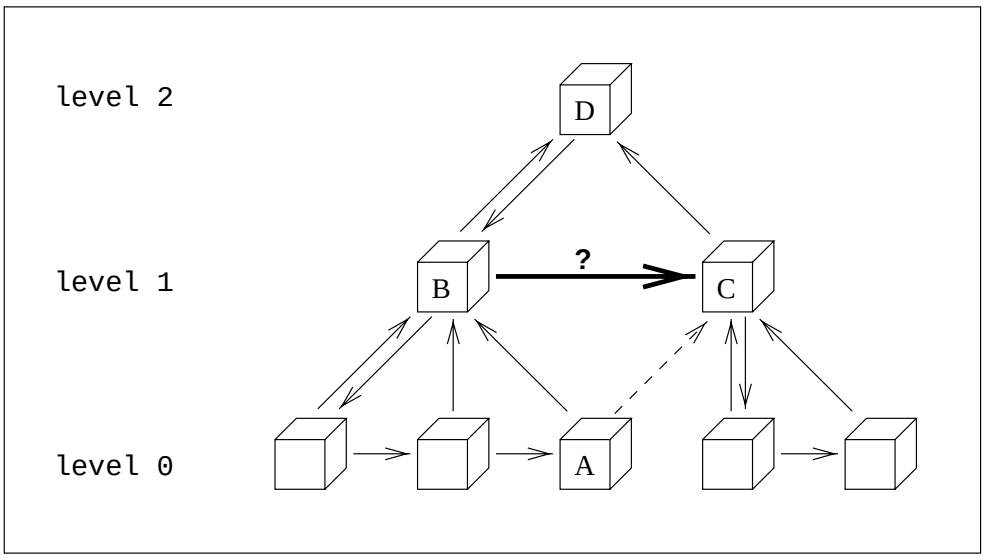


Fig. 4.10. *Single-parent data structure showing the multi-parent problem. Adding a second parent C for child A (dashed arrow) will introduce ambiguity with respect to the sibling link from B to C (fat arrow): all the children of parent B seem to have a second parent C, although this is only true for child A.*

4.B Multi-parent data structure by means of link containers

In this appendix we discuss the implementation of the multi-parent data structure by means of link containers.

The solution to the ambiguity problem of appendix 4.A is effectively a *singly* linked list, implemented with *link containers* (see Fig. 4.11). With the help of the link containers, only parent linkages are stored, keeping all parents that belong to one child together. Each voxel has a unique reference to one link in the link container—indexing the first parent—while all remaining parents are accessed by *subsequent* fields in the container. This makes multi-parent linking possible without introducing ambiguities.

The list of probabilities per ground voxel—denoting the chances that a voxel belongs to different segments—are implemented similarly by means of a *root container* (not shown in the figure).

Using link containers, the downward projecting of root values can easily be implemented as a bottom-up process. Since it can be shown that this segmentation step is equally expensive as the top-down scanning process used in the single-parent data structure, the halving of the number of linkages—owing to the use of singly- instead of doubly linked lists—will result in an appreciable reduction of computing time for the segmentation step.

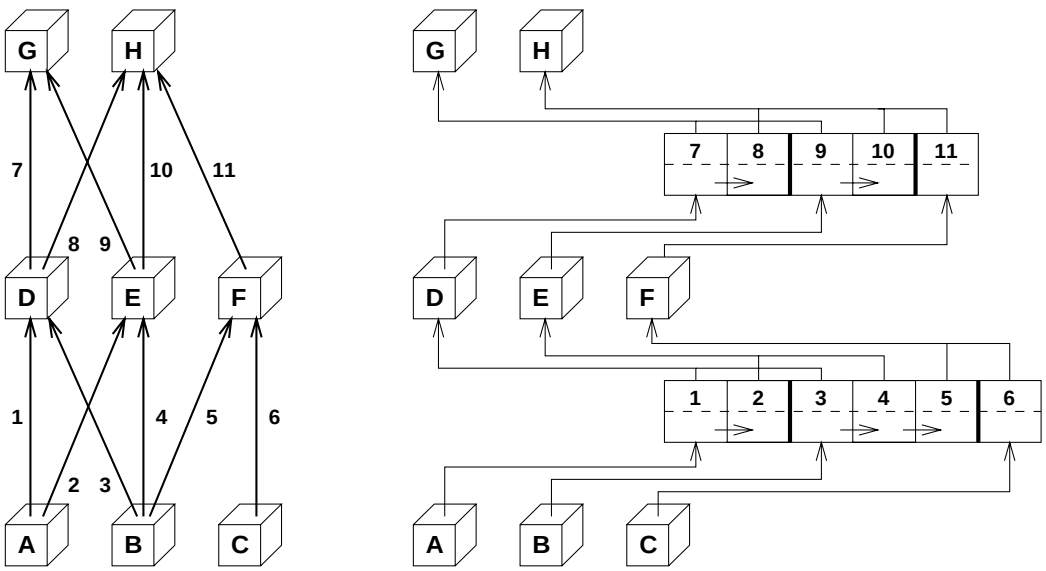


Fig. 4.11. *Schematic of the notion of link containers (right picture). The arrows in the hyperstack (left picture) are implemented as indices, so no pointers are involved.*

'A theory has no more truth than a hammer; what matters is whether you get the nail into the wall'

– Joe Schwartz
(De Volkskrant, 24 December 1993)

Chapter 5

Outer scale reduction in multiscale image analysis

Abstract

In multiscale image analysis a stack of blurred replicas of an input image is obtained by a sequence of filtering steps with an increased smoothing effect. Since high frequency information vanishes at increasing scale, the images at the larger scales are oversampled. In this chapter it is shown how the *number* of samples of the higher levels of the stack can be reduced without affecting the sampling *width*. Effectively, this leads to a reduction of the region of interest, or *outer scale*, with increasing scale. It is argued that the reduction speed should be controlled by the *absolute* scale of each level.

The profit of outer scale reduction in multiscale image analysis is twofold: the influence of the boundary problem is minimized, and the complexity of the scale space application is decreased without significantly affecting the quality of the results.

The latter profit is illustrated by a multiscale image segmentation technique, called the *hyperstack*. No significant loss of quality is found upon introducing outer scale reduction in the hyperstack segmentation process, while the profit in computation time is enormous.

Keywords: Image segmentation, multiscale image analysis, scale space, sampling rate reduction, outer scale.

5.1 Introduction

The by now well-understood and widely accepted framework of linear 'scale space' theory [131, 54] has triggered an increasing number of researchers to develop multiscale image analysis applications. In essence, a scale space representation provides the information contained in an image at multiple levels of scale by deriving blurred replicas of the input image. The thus created stack of images can be used as basis for a multiscale linking model to connect pixels through scale space. The first algorithm

based on this concept was the pyramid (see section 5.3), that was used for image segmentation [16, 24, 23, 60]. In section 5.3 we will deal with the conventional pyramid. Later, algorithms based on other scale space representations, like the extremum stack [64] and the hyperstack [121, 123, 125, 124, 58] (see section 5.4), have proven to be useful for segmentation of complex images.

There are two problems when using scale space theory: (*i*) the extraction of large scale information requires an (explicit or implicit) solution to the boundary problem; and (*ii*) all the levels derived from the original image are equal in size, although the amount of information decreases with increasing scale: the images are oversampled.

In section 5.5 we discuss the former problem. We need knowledge about the pixel data outside the boundaries of the image. Roughly spoken, this problem can be solved in three ways: (*i*) estimating the missing pixel data; (*ii*) adapting the used kernel—by varying its scope—according to its spatial position; (*iii*) applying a non-uniform type of sampling rate reduction (SRR) such that the filter window is limited to the known pixel data. For the first two options the image dimensions remain unaltered. The third option, however, results in a reduction of the field of view (FOV), which manifests itself by a decrease of the dimension sizes of the output image. In multiscale image analysis the FOV (or the 'region of interest') is called *outer scale* [55], since it is scale dependent. Hence, for option (*iii*) the term *outer scale reduction* (OSR) is used. With OSR the number of samples decreases at increasing scale without affecting the sampling width. Hybrids of the three methods are also possible.

The second problem of applying scale space theory is the key point of this chapter. The fact that images at the larger scales are redundant with respect to the number of samples representing the signal at that scale is often ignored. For instance, in [64] the reason for *not* implementing the theoretic reduction was caused by the expected increase in complexity of the involved linking algorithm.

Our main goal is to have a mechanism that controls the sampling of the blurred images such that a maximum profit is achieved (both in storage space and in computing time) without significantly affecting the results of the scale space application. In this chapter we use the hyperstack segmentation method (see section 5.4), which is based on a multiscale linking model, to evaluate the results.

At first sight, the Nyquist criterion offers the proper starting point. It is well-known that the minimum number of samples required for a bandlimited signal to be completely represented follows from the Nyquist criterion (*e.g.,* see [43]). Since high frequencies are removed at increasing blur, it may be expected that the number of samples needed to represent the blurred levels decreases accordingly. In other words, the oversampled data can be uniformly resampled. In section 5.2, however, we will discuss why this criterion is not feasible within our scale space framework.

Fortunately, linear scale space theory provides us with a stable formalism that indicates how the number of samples should decrease at increasing scale [33] (see section 5.6): *strict* OSR. The main differences with the Nyquist criterion are: (*i*) the Nyquist frequency is image dependent, and (*ii*) sampling according to Nyquist leads

to a *sampling rate reduction* (SRR) rather than an OSR. Since decreasing the outer scale reduces the influence of the boundary problem, OSR is preferred to SRR in a multiscale environment.

In the same section we show that applying strict OSR theory in practice leads to a fast decrease of the outer scale. Although the computational profit is enormous, a major disadvantage is that the region of the image that can be segmented by the hyperstack only covers the middle of the input image. A practical but effective solution to this problem is to adapt the amount of OSR. In section 5.7 we will present a solution (called *heuristic* OSR) to achieve this.

Results will be shown by applying the theory to real-world images. The effect on the segmentations of these images will be shown for various forms of OSR.

5.2 Sampling rate reduction according to Nyquist

In this section we discuss sampling rate reduction of multidimensional images using the Nyquist criterion. The theory is presented for the continuous case, but extends straightforwardly to the discrete domain. The term 'sampling rate reduction' (SRR) refers to a rediscretization of a signal that has been sampled by N uniform samples to M (uniform) samples, under the constraint that $M < N$.

If the frequency distribution of a bandlimited signal is known, then the Nyquist criterion determines the *minimum* number of samples that is necessary to represent the signal. More precisely, for a bandlimited signal $L(\vec{x})$ a maximum frequency ω_m exists such that

$$\mathcal{L}(\vec{\omega}) \;=\; 0 \quad for\; |\,\vec{\omega}\,| > \omega_m \quad , \tag{5.1}$$

where $\mathcal{L}(\vec{\omega})$ is the Fourier transform of $L(\vec{x})$. Let T be the sampling period, then $L(\vec{x})$ is uniquely determined by its samples $L(nT)$, $n \in \mathbb{Z}$, if and only if:

$$\omega_s \;>\; 2\,\omega_m \quad , \tag{5.2}$$

where $\omega_s \;=\; 2\pi/T$, the sampling frequency. However, a bandlimited spatial signal has an infinite spatial support (in keeping with the *duality* property). So, real world images cannot be bandlimited. The fact that images have a finite support implies that they extend infinitely in the frequency domain.

A solution to this problem is to estimate the 'missing' spatial data, but this can affect the value of ω_m significantly (*e.g.,* by a 'jump' of the data near the image boundaries). In some cases the assumption that the signal is *periodic* may solve the problem, but this is seldom true for signals containing spatial information. An exception may be the time-series of a periodic signal, such as the beating of a heart.

Even if the missing data problem can be solved (by using an extrapolation method), we still have to determine a practical value of ω_m. That is, find the highest frequency in the Fourier domain that still corresponds to the data. Owing to the presence of

unwanted high-frequency noise, a straightforward calculation of ω_m may lead to a relatively high value, so that the resulting Nyquist sampling rate ω_s will also be high. The problem is that it is very difficult—if not impossible—to define unambiguously the threshold value above which frequencies must be considered (high-frequency) noise.

Consequently, *a priori* information on the image noise or the appropriate signal-to-noise ratio must be added to the sampling scheme for obtaining the maximum profit of SRR. The inclusion of *a priori* information is undesirable for two reasons: (*i*) it makes the SRR method image dependent, and (*ii*) the amount of SRR is unpredictable. Although these disadvantages are not fatal for applying a form of SRR, we prefer a more robust, general reduction method.

Note that for discrete images the situation is less favorable still: if the sampling period T turns out to be a non-integer number, interpolation of the data will be necessary to obtain the resampled image at the maximum reduction. (In fact, to prevent this we have to require $T \in \mathbb{Z}^+$, which leads to a very limited range of discrete values for ω_s.) A major disadvantage of interpolating the data is the simultaneous, unintended introduction of a blurring and/or an aliasing effect. The actual distortion depends on the interpolation method used [83, 47, 112].

To summarize, we opt for a different approach towards reducing the number of samples of a discrete signal. In the sections 5.6 and 5.7 we will explain how we successfully combine the scale information of an image and outer scale reduction, thereby solving the boundary problem. To this end, we first have to go into detail on the concept of *scale*. We do this by explaining the well-known pyramid approach in section 5.3, followed by a description of the *hyperstack* in section 5.4 [121, 123, 125, 124, 120, 126], which is based on a more robust and much improved representation of an image at multiple scales.

5.3 The pyramid

In this section we describe the first multiscale segmentation technique, the pyramid. The pyramid was introduced by Burt *et al.* [16] and further elaborated by various researchers [5, 46, 24, 23, 60, 11, 58].

The conventional d-dimensional pyramid is created by sub-sampling level n to derive level $n+1$, such that an explicit SRR of a factor two is performed in each Euclidean direction. This is done by straight averaging of 2^d pixels (the children) to form one blurred pixel (the parent). Later, a standard option was to use a 50% overlap for the averaging areas of two neighboring pixels. For the Gaussian pyramid the blurring is performed by convolution with a discrete Gaussian kernel.

In the initialization step, all the children that contributed to the blurred parent pixel are linked to that parent. This is done for the entire pyramid structure in a bottom-up manner.

In the iterative linking step, the linkages are reviewed by searching for 'better' parents (*i.e.,* pixels in the next higher level having a smaller intensity difference with

the child at hand). In such cases, the old link is deleted and a new one is established. After a complete revision of the linkage structure, the average parent values are recalculated. The iteration then starts again, until the pyramid does not change anymore. This process is guaranteed to converge [50].

Pyramids have been used in a variety of other applications, such as data compression based on hierarchical interpolation (HINT) [93, 92], and hierarchical chamfer matching [13].

The most striking disadvantages of the pyramid approach are (see also [5, 11]):

- *Coarse scale space sampling.* The resolution in each dimension of the pyramid decreases with a factor two per level. The blurring thus is independent of the image contents: it is as coarse for simple stylized images (for which it may be sufficient) as for complex images. Moreover, the blurring is *ad hoc* since it is based on discretization details rather than on a grid-independent concept.

- *Enhanced discretization effects.* The pyramid model entails that discretization effects are propagated all the way up to the top level of the pyramid. This effect is highly image dependent.

- *Shift/rotation variance.* Translating or rotating the input image may result in a segmentation that is dissimilar with the original segmentation. Thus, reproducibility can not be guaranteed if simple geometrical transformations have been applied to the input image.

- *Inhomogeneous linking.* The linking possibilities for each pixel depend on the spatial location of that pixel. For instance, if the search volumes have the customary 50% overlap, only pixels in the middle of the image can be linked to other quadrants; the other pixels have to remain in their quadrant until in the last pyramid step the final 4 pixels link together.

- *Fixed number of segments.* The fact that the two top levels of the pyramid consist of 4 and 16 pixels (in 2D), respectively, makes it impossible to create a different number of segments. This is highly inflexible. The problem can only be solved by very drastic and *ad hoc* solutions [46].

In our opinion, the major shortcoming of the pyramid is that its SRR is not related to the scale transitions that follow from the contents of the image. These transitions can readily be calculated [33].

The main advantage of the pyramid approach is its computational speed. The total number of pyramid pixels equals approximately $(2^d N)/(2^d - 1)$, where N denotes the number of pixels of the d-dimensional input image. The larger d, the less overhead of the pyramid structure is involved. (For instance, a three-dimensional image requires only an additional $0.14\,N$ pixels.)

Many improvements have been proposed to the conventional pyramid segmentation method. It has been suggested to focus on different blurring methods or on the

used kernel. Examples are the Inca pyramid [114], which has the structure of the conventional pyramid but with a delayed SRR (a 'stretched out' pyramid); the adapted or *dual* pyramid, which makes use of intermediate levels that have been rotated by $\frac{1}{4}\pi$ [60, 61]; the Gaussian pyramid [24, 23], where the averaging has been replaced by a better discrete representation of the Gaussian kernel; the median filter [104], which is a highly nonlinear blurring kernel; and finally the Gaussian kernel in general [54, 7, 33].

5.4 The hyperstack

In this section we briefly describe the hyperstack segmentation method. For an extensive description of this multiscale method we refer to [123, 124, 120].

The hyperstack segmentation method consists of four phases (see Fig. 5.1). In the following subsections we deal with each step separately.

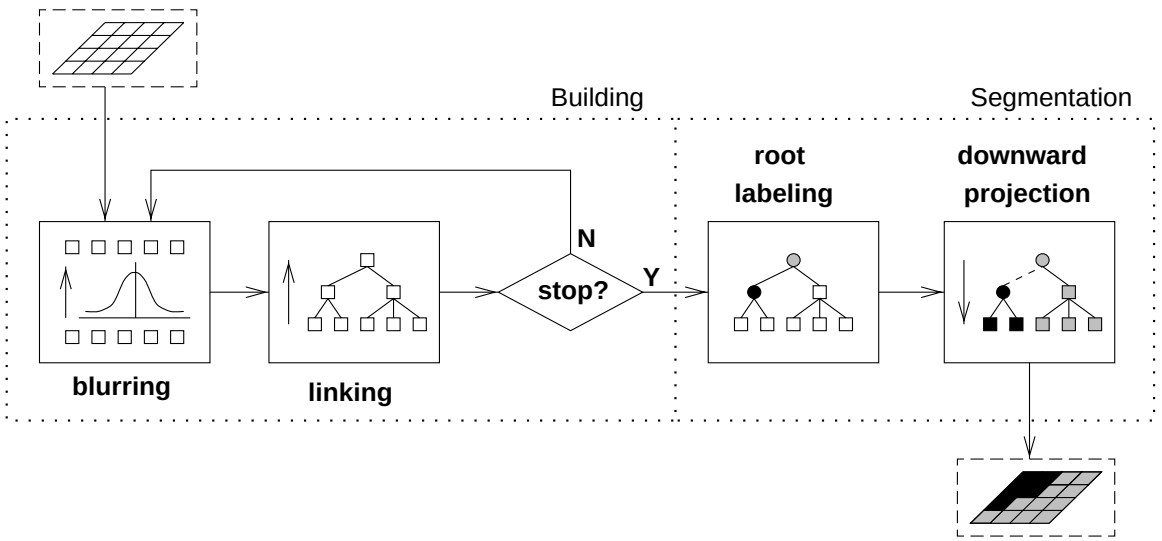


Fig. 5.1. *Schematic of the hyperstack image segmentation process.*

5.4.1 Blurring

A scale space of the image to be analyzed is created by filtering the original image with a Gaussian kernel of increasing width. The Gaussian convolution kernel—originally proposed by Koenderink [54] and Witkin [131]—is denoted by $G(\vec{x}, \sigma)$, where σ is the standard deviation of the corresponding Gaussian function.

The Gaussian is the unique linear scale space generator for front-end vision [7], *i.e.,* if no knowledge about the imaged scene is available. This unicity follows from *postulates* that a front-end vision system should be able to generate 'a complete, homogeneous, isotropic, and scale invariant representation of its input data' [30].

The uniqueness of the Gaussian kernel was derived earlier by Koenderink [54]. He formulated the *causality* requirement that no 'spurious resolution' should be generated with increasing scale. This leads to isosurfaces in scale space that are convex in the direction of increasing scale. Note that this forces the number of local extrema to decrease monotonically with scale only in the 1D case (see also [64, 66, 65]).

According to the property of scale invariance the blurring strategy has to follow an exponential sampling in scale space. Hence, the linear scale parameter $\tau = n\,\delta\tau$, $n \in \mathbb{N}$, is related to the absolute scale σ_n at level n by:

$$\sigma_n = \varepsilon\, e^{\tau_0 + n\,\delta\tau} \quad , \tag{5.3}$$

where τ_0 is the initial scale offset and ε is taken to be the smallest linear grid measure of the imaging device. If we denote level n of the discrete scale space by $\mathcal{S}_n$, then we can write for the *scale space representation* $\mathcal{S}$ of the d-dimensional input image $L(\vec{x})$ (*i.e.,* the collection of d-dimensional images at increasing scales):

$$\mathcal{S}_n = \mathcal{S}(\vec{x}, \sigma_n) = L(\vec{x}) * G(\vec{x}; \sigma_n) \quad , \tag{5.4}$$

with initial conditions $\mathcal{S}(\cdot;\, 0) = L(\cdot)$ and $\tau_0 = 0$. The dimensionality of a scale space $\mathcal{S}$ is $d + 1$.

5.4.2 Linking

During the linking phase pixels in two adjacent levels are connected by so-called *parent-child* linkages. Linking is a bottom-up process in the sense that we start linking the children of level 0 (the ground level) to parents in level 1, then find parents in level 2 for the children in level 1, and so on. Since only pixels that have been selected as parents are considered children in the next linking step, convergence of the scale-tree is assured. The area in which a parent is selected for a specific child is defined by a radius $r_{n,n+1} = k_n\, \sigma_{n,n+1}$, where $\sigma_{n,n+1} = \sqrt{\sigma_{n+1}^2 - \sigma_n^2}$ (the self-similarity property) is the relative σ of the Gaussian kernel that corresponds to the scale transition from level n to level $n + 1$, and k_n is chosen such that only pixels whose intensity has been determined to a significant extent by the child at hand are considered candidate parents (see [120] for details). For simplicity, k_n is often given a constant value throughout the scale space, *i.e.,* independent of the level n. A typical value for k_n is 1.5, which corresponds to an influence factor $\gamma = \exp(-\frac{1}{2}{k_n}^2)$ (see section 2.A) of 0.32 (*i.e.,* only pixels whose intensity is determined by the child pixel at hand for at least 32% are considered candidate parents). We found experimentally that smaller values of the influence factor (*i.e.,* larger values of k_n) did not improve the results. Upon combining the self-similarity property with equation (5.3), we get:

$$\sigma_{n,n+1} = \varepsilon\, e^{\tau_o + n\,\delta\tau} \sqrt{e^{2\,\delta\tau} - 1} \quad . \tag{5.5}$$

Research has shown that a general and robust linking scheme should consist of three components [58]: (*i*) the intensity difference between a child and its parent,

(*ii*) the ground volume of a parent, and (*iii*) the mean ground volume intensity. See section 2.2.2 for details.

5.4.3 Root labeling

The roots—*i.e.,* pixels in the scale-tree that represent segments in the original image—are identified after all levels have been connected through linkages. We have investigated the applicability to root labeling of a feature-based approach (*i.e.,* the *adultness* term) similar to the affection term used in the linking phase. Children having a large adultness are classified as roots. This is done either by setting a threshold on this value, or by prescribing the number of roots.

The two components that turned out to be robust root criteria for the adultness term were [58]: (*i*) the ground volume mean intensity difference component, and (*ii*) the ground volume component. See section 2.2.3 for details.

5.4.4 Downward projection

In the actual segmentation phase root values are projected downwards from every root to the corresponding pixels in the ground level. By using a unique value for each root it is guaranteed that the segments in the original image are distinguishable. This allows for quantitative validation of segmentation results. For visual interpretation of the segmentation, it may be more attractive to display the mean intensity of the pixels within each segment. See section 2.2.4 for a more extensive description of possible values.

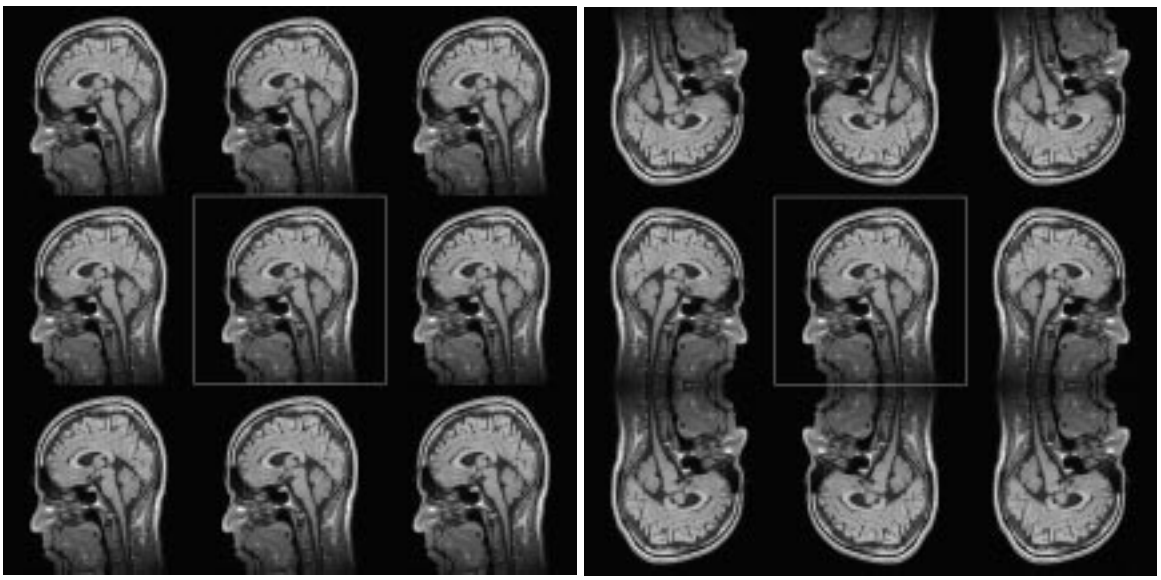
5.5 The boundary problem

The boundary problem occurs when the blurring kernel extends beyond the image borders. Any (linear) kernel with an extent of more than 1 pixel in one of the spatial directions will pose such a problem.

5.5.1 Fourier domain

We first consider the boundary problem in the Fourier domain. Blurring of the input image to build a scale space is often done in the frequency domain, partly because of the nice mathematical *convolution property* that can speed up the filtering process, partly because the boundary problem is implicitly 'solved'. As for the latter, *repetition* and *mirroring* are the two most commonly used variants for 'estimating' the pixels beyond the image boundaries (see Fig. 5.2).

The boundary effects of Fourier filtering are illustrated in Fig. 5.3. The extrapolated pictures of Fig. 5.2 have been blurred (in Fig. 5.3a and Fig. 5.3b, respectively) by a Gaussian kernel with a standard deviation of half the image size, *i.e.,* 128 pixels.



(a) (b)

Fig. 5.2. *Example of data extension used in Fourier filtering: (a) repetition of the signal; (b) mirroring of the signal. The bright white square in the middle denotes the original* BRAIN *image, a two-dimensional* 256^2 *sagittal MR image.*

We did the same for an artificial image, the 'HALF.ELLIPSE' image of size 256×256, where Gaussian noise with a standard deviation of 10% has been added (see Fig. 5.3d and Fig. 5.3e, respectively). (For clarity, we included a coarse iso-intensity contour plot for each picture.)

Evidently, the appearance of bright blobs at the boundaries of the image is caused by the blurring in the Fourier domain. The results also demonstrate that extrapolation by repetition or mirroring produces quite dissimilar artifacts, the nature of which is strongly image dependent.

5.5.2 Spatial domain

In the spatial domain a variety of extrapolation methods can be used. Since an extensive treatment of these methods is beyond the scope of this thesis, we only mention a few popular techniques.

We first dwell upon the notion of 'scope' of a Gaussian kernel. This is the spatial limit imposed on the discrete representation of the kernel. A large scope (measured in natural or scale units) is equivalent to accurately modeling the blurring: the tails of the Gaussian function are well taken into account. For an isotropic kernel of width σ_n, the scope can be defined by a radius $R_n = K_n \sigma_n$, where K_n is related to the d-dimensional integral of the Gaussian function (see [118] for details). Typically, K_n

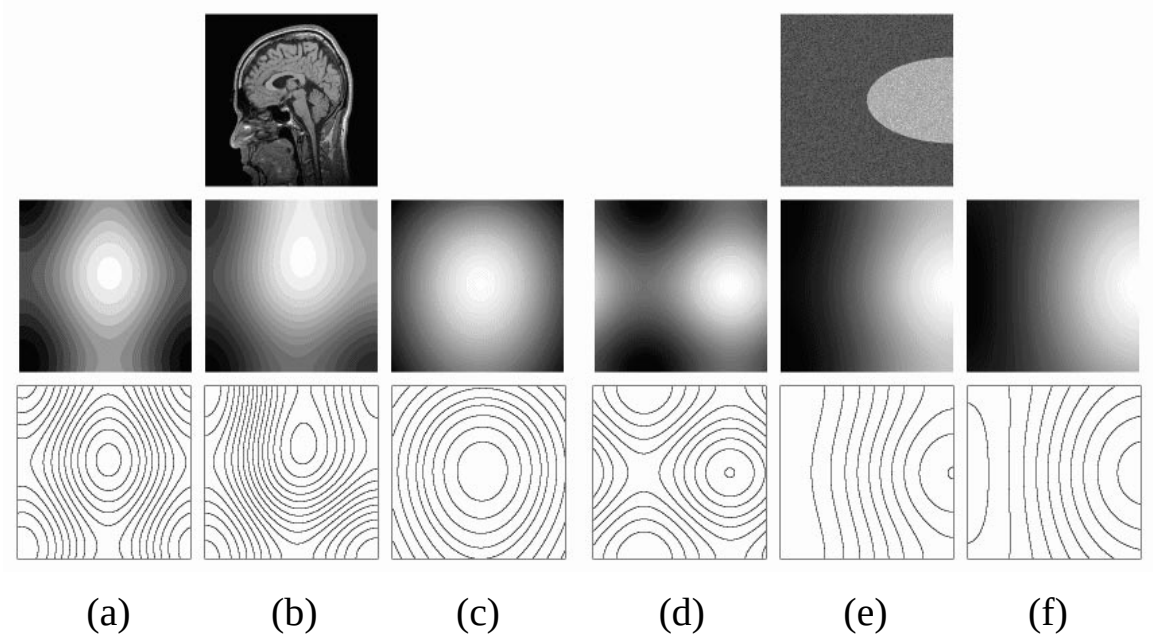


Fig. 5.3. *Illustration of the effect of the used extrapolation method on the blurring for the image of Fig. 5.2 (a, b, and c) and for an artificial image containing half an ellipse (d, e, and f): repetition in the Fourier domain (a and d); mirroring in the Fourier domain (b and e);* 0^{th} *order extrapolation in the spatial domain (c and f). Corresponding iso-intensity plots have been included below. For all images we have used a Gaussian kernel of width* $\sigma = 128$ *pixels.*

takes the value 2.5, which corresponds to a scope encompassing 95.6% of the volume of the 2D Gaussian integral, and to 90.0% in the 3D case.

As for extrapolation techniques in the spatial domain, the most straightforward—and most common—method is 0^{th} order extrapolation, where the outmost pixels of the image are simply expanded on an infinite basis. Alternatively, 1^{st} order (linear) extrapolation can be performed based on the outmost *two* pixels of the image boundary. Higher order extrapolation is generally unattractive, because a large variance of pixel values near the image boundary may result in overflow output (*i.e.,* the aliasing effect increases).

As an alternative, the *average image intensity* value can be used as global extrapolation value. At first sight this may seem a bit odd, but there are two important assets for this option. Firstly, at an extremely large scale, the image must be regarded as a single pixel that is surrounded by an infinite number of other (unknown!) pixels. The least committing estimate for these pixels is the average value of the input image, since this is the value that corresponds to Gaussian blurring at infinite scale. Secondly, it should be prevented that disturbing fluxes enter the image (the image energy should be conserved). The average image value satisfies this requirement, although

small distortions may occur owing to discretization errors.

Note that 0^{th} or 1^{st} order extrapolation may produce results with a substantially different energy, for instance if the image has a region of interest located in the middle and dark background pixels near its borders. The total image energy will then diminish if the blurring step is increased.

To illustrate the spatial extrapolation method, we have applied 0^{th} order extrapolation to both the BRAIN image and the HALF.ELLIPSE image (see Fig. 5.3c and Fig. 5.3f, respectively). It is evident that no bright blobs appear at the image boundaries.

Spatially variant kernels

As a solution to the boundary problem, one could adapt the shape of the kernel while approaching the boundary. Note that this will result in a nonlinear scale space. In general, the calculated scale will be too high at the boundaries, because less pixels are used to compute the blurred value of a pixel at the boundaries than in the middle of the image. The major advantage of spatially variant kernels is that the image needs not be extrapolated.

Three examples of an adaptive kernel are outlined in Fig. 5.4. Note that every such kernel must be implemented in the spatial domain.

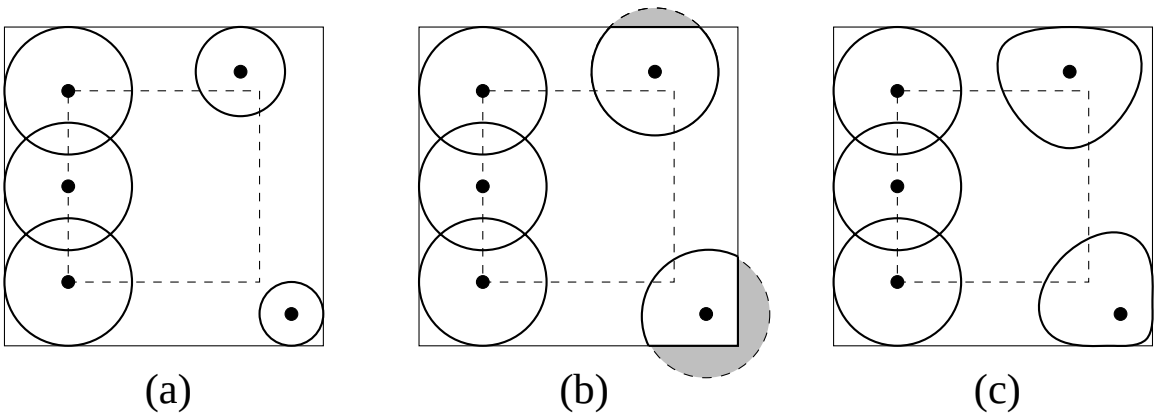


Fig. 5.4. *Schematic of three different spatially varying kernels: (a) decreasing; (b) chopping; (c) deforming. The square in the middle denotes the area in which the boundary problem is absent.*

In its simplest form, the kernel size is constant in the middle of the image and decreases near the boundaries, so that it will never trespass the image border (see Fig. 5.4a). The disadvantages are: (*i*) strong distortion of the linear scale space near the image boundaries (note that the outmost boundary pixels can not be blurred according to this algorithm), and (*ii*) dependency of the amount of distortion on the extent of the discrete representation of the Gaussian kernel. For instance, if the extent is large, the kernel will be reduced already at a large distance from the image boundaries. This may be remedied by adapting the scope in conformity with the

adaptation of the kernel at the cost of a lower accuracy of the convolution at the boundaries. In any case, the boundary pixels will be calculated at a lower accuracy than the middle pixels.

Another option is shown in Fig. 5.4b. Here, the kernel is simply chopped off where it crosses the image boundary. Compared to the previous method it is an advantage that the scope of the blurring kernel no longer determines the amount of scale space distortion. The scale is calculated correctly for each pixel, though again with a lower accuracy at the image borders. Similar to the previous solution, if there is a lot of significant image information near the image boundaries, then ignoring the missing data will influence the convolution process to a large extent.

Finally, Fig. 5.4c shows the most complex variant. The kernel is deformed without changing the number of pixels used per convolution. To achieve this, the kernel has to deform itself, especially at the corners of the image. Although one would expect the scale space deformation to be less compared to the previous two methods, the algorithm is difficult to implement, and hence not used very often. Furthermore, it is far from trivial to determine how the kernel exactly should be deformed in order to obtain optimal results.

5.6 Strict outer scale reduction

In this section we will show how knowledge of the used blurring kernel can aid in solving the boundary problem. The result is a blurring scheme with an explicit outer scale reduction.

5.6.1 Theory

As described by Florack in [33], blurring an image with a (discrete) Gaussian kernel must obey very strict rules. The largest blurring scale $\sigma_{n,max}$ that can be applied is bounded by two components: the *scope* of the kernel (defined by the factor R_n) and the *spatial location* at which the kernel is applied.

The spatial location is expressed by the distance d_p (in pixels) to the closest image boundary. It follows that:

$$\sigma_{n,max} = \frac{d_p}{K_n} \quad . \tag{5.6}$$

This formula clearly shows that it is not feasible to obtain large scale information of an image near its boundaries.

In consequence, a resampling of blurred images should preferably *ignore* the data near the boundaries *without affecting the applied kernel per pixel*. This leads to a fast decrease of the area of pixels that can be blurred correctly if σ increases (see Fig. 5.5). As a result, the boundary problem is avoided, and an explicit OSR is implied.

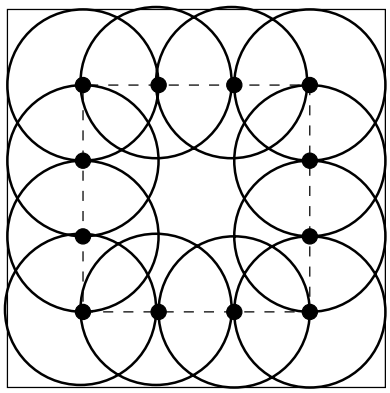


Fig. 5.5. *Strict outer scale reduction: equal-sized kernels are used throughout image space. An explicit outer scale reduction is implied by this technique.*

If we denote the linear image dimension at level n by N_n, then

$$N_n \;=\; N_0 - 2\,R_n \;=\; N_0 - 2\,K_n\,\sigma_n \quad . \tag{5.7}$$

According to equation (5.3) a scale space has to be exponentially sampled in the scale direction, which gives:

$$N_n \;=\; N_0 - 2\,K_n\,\varepsilon\,e^{\tau_0 + n\,\delta\tau} \quad . \tag{5.8}$$

On the other hand, according to Florack [33], the ratio N_n/N_0 has to be inversely proportional to the corresponding σ's:

$$\frac{N_n}{N_0} \;=\; \frac{\sigma_0}{\sigma_n} \quad . \tag{5.9}$$

Jointly with equation (5.3) this yields:

$$N_n \;=\; N_0\,e^{-n\,\delta\tau} \quad , \tag{5.10}$$

which defines the sampling rules according to strict OSR.

Using (5.10) we are now able to calculate $\delta\tau_p$, *i.e.,* the $\delta\tau$ that corresponds to the (Gaussian) pyramid. Pyramids have the property that $N_{n+1} = 2^{-d}\,N_n$, so $N_n = N_0\,2^{-nd}$. It follows that the scale space sampling parameter $\delta\tau_p$ for the pyramid is given by $\delta\tau_p = d\ln 2$. This sampling of scale space is so coarse that the resulting segmentation is not adequate for various purposes [64]. Why from a purely practical point of view a much finer sampling of scale space is required. For practical purposes a value of $\delta\tau = \frac{1}{2}\ln 2$ was found to be reasonable. This gives a (natural) sampling in the scale direction that is '$2d$' as good compared to the pyramid sampling method in d dimensions. Since our images are mostly 2D or 3D, this yields a sampling that is at least four times better than the pyramid sampling. For all cases, we have chosen this value for $\delta\tau$, unless indicated otherwise.

Upon combining equations (5.8) and (5.10), we arrive at:

$$K_n \;=\; \frac{N_0}{2\varepsilon}\, e^{-\tau_0 \,-\, n\,\delta\tau} \left(1 \,-\, e^{-n\,\delta\tau}\right) \quad . \tag{5.11}$$

It is clear that strict OSR implies that K_n is scale dependent (see also equation (5.6)). We will use this observation in section 5.7.

In practice, equation (5.10) can not be used straightforwardly, because the truncated image dimensions have to be natural numbers. Even stronger, if N_0 is even (odd), then N_n has to be even (odd) too. (Otherwise, asymmetric levels have to be linked together.) Therefore, we round N_n to $\widehat{N}_n$ ($\widehat{N}_n \in \mathbb{Z}^+$), which is the nearest integer that is not smaller than N_n and also satisfies the even/odd requirement.

5.6.2 Strict OSR and scale space

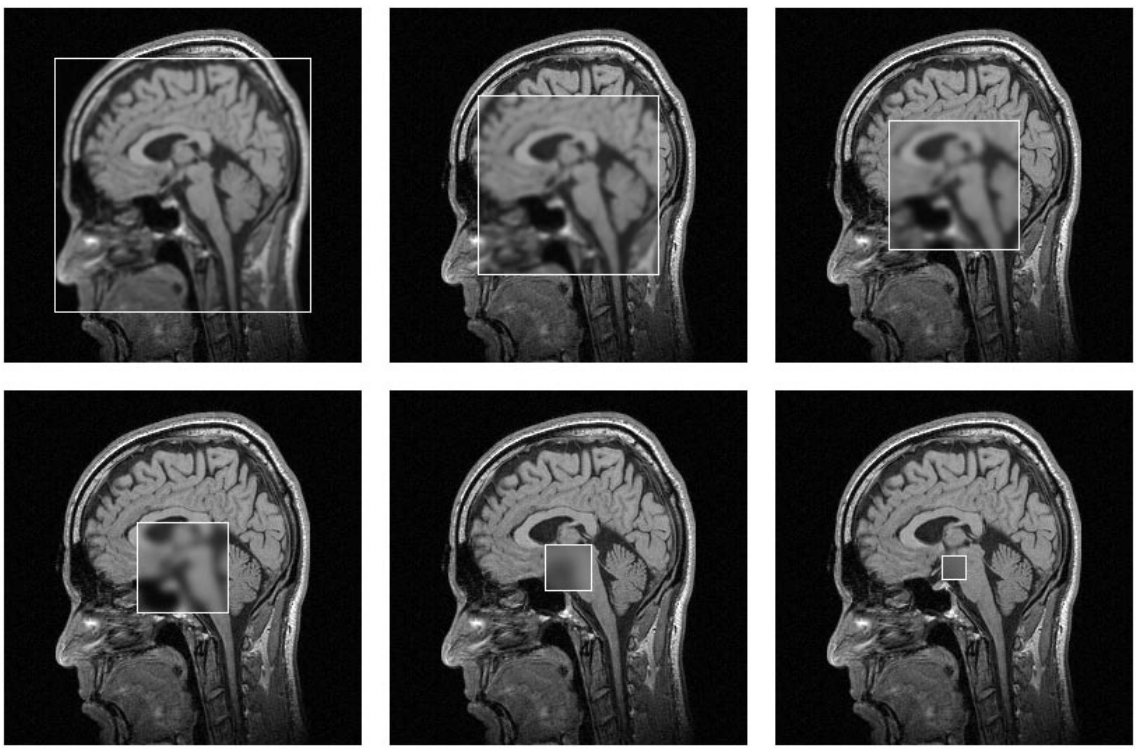


Fig. 5.6. *Effect of strict OSR on a linear Gaussian scale space for the* BRAIN *image of Fig. 5.2. The blurred replicas of the input image at increasing scale are indicated by the bright white borders. Shown are the levels 1, 2, and 3 (upper row), and 4, 6, and 8 (lower row). The corresponding σ's are 1.4, 2.0, 2.8, 4.0, 8.0, and 16.0 pixels.*

Without OSR the dimensions of each level of a scale space remain unaltered. To

show the effect of strict OSR on a scale space, we applied (5.10) to the two-dimensional BRAIN image of Fig. 5.2 (see Fig. 5.6).

It is clear that the number of remaining pixels decreases fast if scale is increased. From the same equation (5.10) we may conclude that the only way to slow down the reduction process of strict OSR is by using a smaller $\delta\tau$ value (see Fig. 5.7).

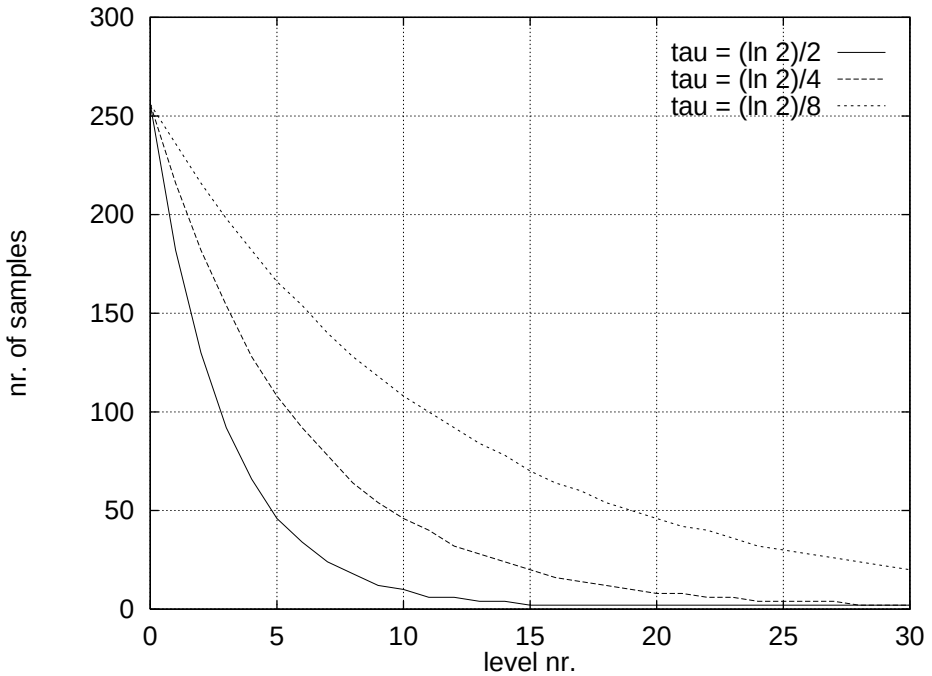


Fig. 5.7. *Effect of decreasing $\delta\tau$ on N_n for strict OSR.*

From this figure, however, it follows that an increasing number of levels will be required to represent the discrete scale space. This is undesirable from the point of view of the number of linkages: the convergence speed of a hyperstack is mainly determined by the scope of the search volume at each level. According to the definition of search volume (see section 2.2.2), this scope depends on the *relative* $\sigma_{n,n+1}$ between the two successive levels n and $n+1$. A smaller $\delta\tau$ results in smaller values for $\sigma_{n,n+1}$, so the hyperstack will have a lower convergence speed (*i.e.,* more linkages are established between adjacent levels). Furthermore, as has been discussed in section 2.2.2, increasing k_n has no qualitative advantage. Thus, strict OSR can not be improved by decreasing $\delta\tau$.

We emphasize that strict OSR is not going 'too fast' from a *theoretical* point of view, and it is entirely correct to use a smaller $\delta\tau$. (In fact, such a scale space approaches the fine discrete representation of the visual system, which is almost a continuous space.) Only computational aspects thwart current research to such fine representations.

5.7 Heuristic OSR

In this section we will present an adaptation to strict OSR, such that the reduction process is slowed down.

5.7.1 Theory

The problem of strict OSR is that, according to equation (5.10) N_n decreases fast if the level n is increased (see Fig. 5.7).

In order to slow down the dimension size reduction process, we make the following observation. The scope of the corresponding blurring steps are given by the K_n values as defined in equation (5.11). In Fig. 5.8 K_n has been plotted as a function of n for three different values of N_0, while the remaining variables have been given fixed values ($\varepsilon = 1$, $\tau_0 = 0$, $\delta\tau = \frac{1}{2}\ln 2$).

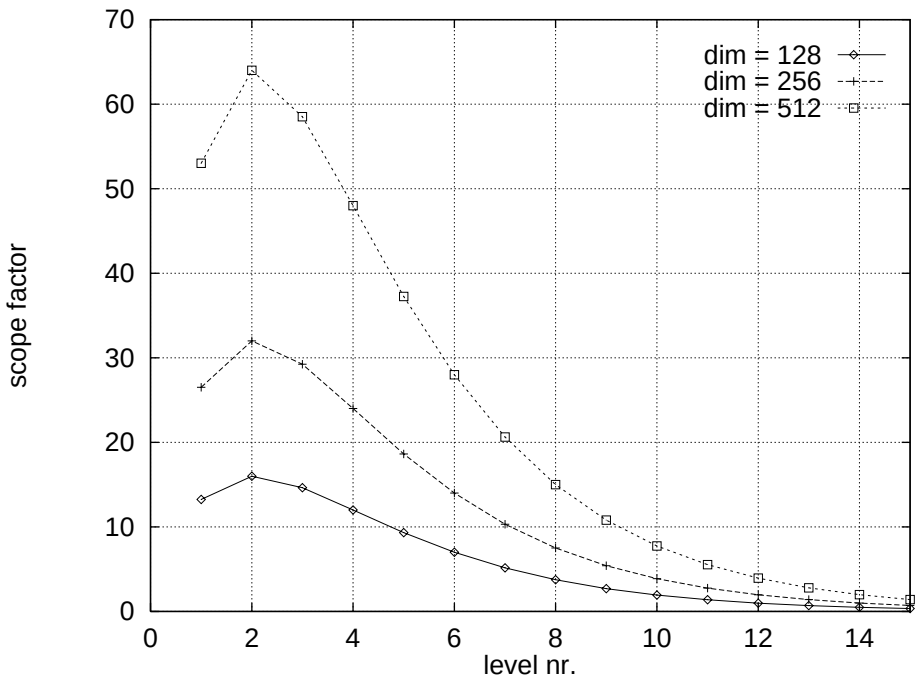


Fig. 5.8. *The scope factor K_n as a function of scale, for three image dimensions.*

It is striking that the scope of the blurring kernel is much higher at smaller scale than at larger scales. This effect is caused by the implication of strict OSR only, according to equation (5.9).

From the blurring point of view it seems odd to change the scope factor for the blurring process from level to level: small scale information should *not* be considered more important than large scale information. Hence, an obvious solution is to make K_n the constant factor and let N_n be dependent of this factor. To this end, we introduce the so-called *reduction factor* ρ: $K_n = \rho, \forall n \in \mathbb{Z}^+$. The term 'reduction

factor' relates to the fact that a different choice for ρ results in a different amount of OSR. Substitution in equation (5.8) yields:

$$N_n \;=\; N_0 - 2\,\rho\,\varepsilon\,e^{\tau_0 + n\,\delta\tau} \quad . \tag{5.12}$$

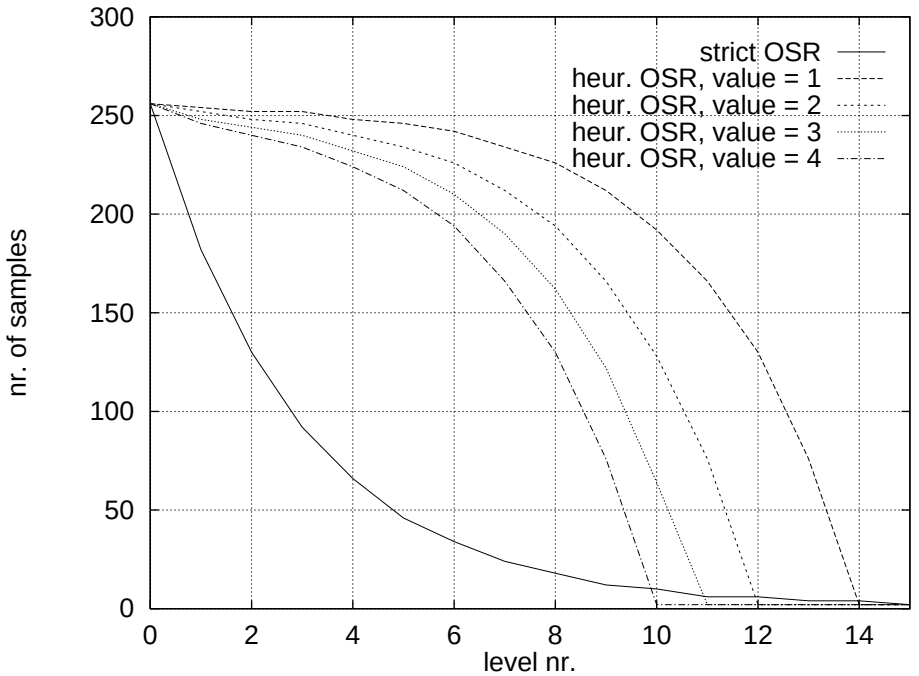


Fig. 5.9. *Decrease of the number of samples of a signal with original dimension $N_0 = 256$. Solid lines: strict OSR; broken lines: heuristic OSR for several reduction factors.*

The resulting N_n values have been plotted in Fig. 5.9 for ρ ranging from 1 to 4, together with the N_n values that correspond to strict OSR, according to (5.10). All N_n values have also been corrected for being integer and even/odd, as stated before. As for the scale space sampling, we have used $\delta\tau = \frac{1}{2}\ln 2$.

5.7.2 Heuristic OSR and scale space

In Fig. 5.10 (heuristic OSR) the outer scale of the BRAIN image has been reduced according to equation (5.12) with $\rho = 3$. In accordance with Fig. 5.9 we note that the reduction process is much shallower than to strict OSR.

At first sight, running a hyperstack on a scale space that is based on heuristic OSR will not give a large computational profit: only the larger levels have a significant decrease of the dimensions. The search volume, however, increases significantly at larger scales. Therefore, heuristic OSR will decrease the computation time especially at the higher levels of the hyperstack. This will be shown for real-world images in section 5.8.3.

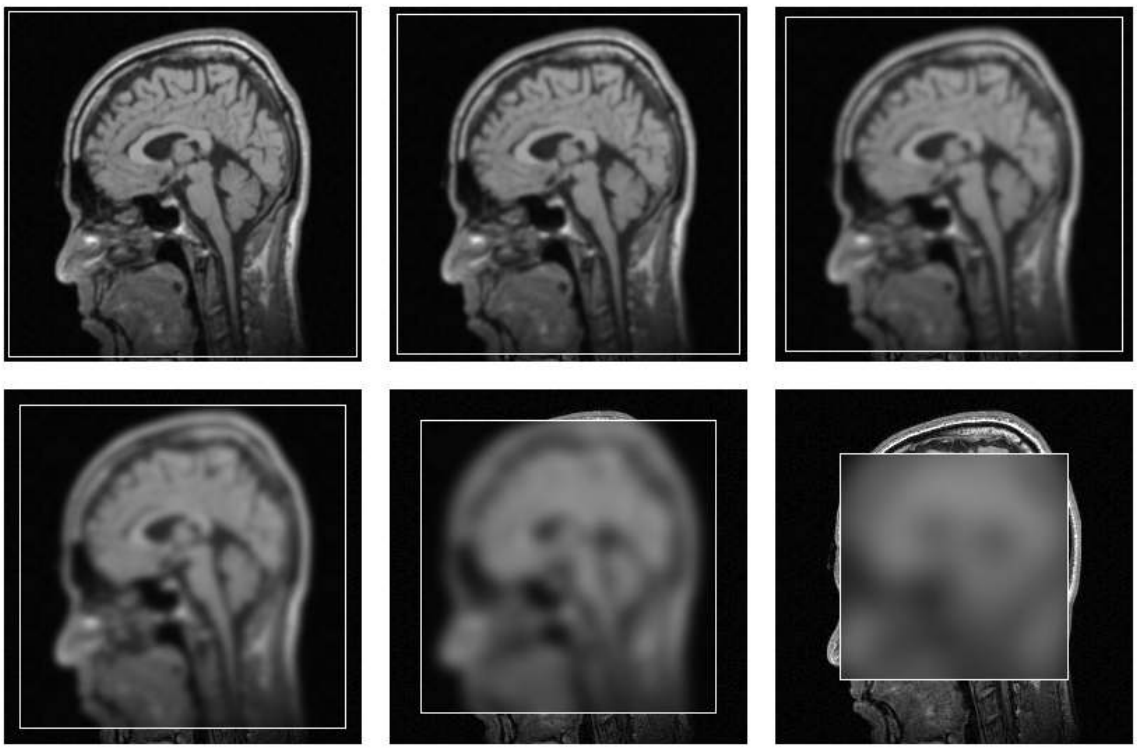


Fig. 5.10. *Effect of heuristic OSR on a linear Gaussian scale space for the* BRAIN *image. As in Fig. 5.6, the blurred levels of decreasing dimension sizes are indicated by the bright white borders. Shown are the levels 1, 2, and 3 (upper row), and 4, 6, and 8 (lower row). Again, the corresponding σ's are 1.4, 2.0, 2.8, 4.0, 8.0, and 16.0 pixels.*

5.8 Results

In this section we quantify the profit in terms of total number of pixels of a scale space, and examine what the effect of (both strict and heuristic) OSR is on the hyperstack multiscale segmentation method. For the latter, we have to investigate the size of the *forced roots* area, which is the area of the input image that can not be segmented properly because OSR is applied.

5.8.1 The profit of OSR

We can calculate the profit that is accomplished by OSR, by computing the total number of pixels of a scale space ranging from level 0 to level l (*i.e.,* the scale-tree volume). To this end, we compare the total number of pixels of a scale space with

and without applying OSR. For the profit P_l at level l we can write:

$$P_l \;=\; \frac{(l+1)\,N_0^d \;-\; \sum\limits_{n=0}^{l} N_n^d}{(l+1)\,N_0^d} \cdot 100\% \quad . \tag{5.13}$$

For the strict OSR we can substitute equation (5.10), which makes the profit $P_{l,str}$ *independent* of the initial image volume N_0:

$$P_{l,str} \;=\; \left(1 \;-\; \frac{1}{l+1} \sum_{n=0}^{l} e^{-d\,n\,\delta\tau}\right) \cdot 100\% \quad . \tag{5.14}$$

In contrast, the profit $P_{l,heur}$ for heuristic OSR is always image size dependent:

$$P_{l,heur} \;=\; \left(1 \;-\; \frac{1}{(l+1)\,N_0^d} \sum_{n=0}^{l} \left(N_0 \;-\; 2\,\rho\,\varepsilon\,e^{\tau_0 + n\,\delta\tau}\right)^d\right) \cdot 100\% \quad . \tag{5.15}$$

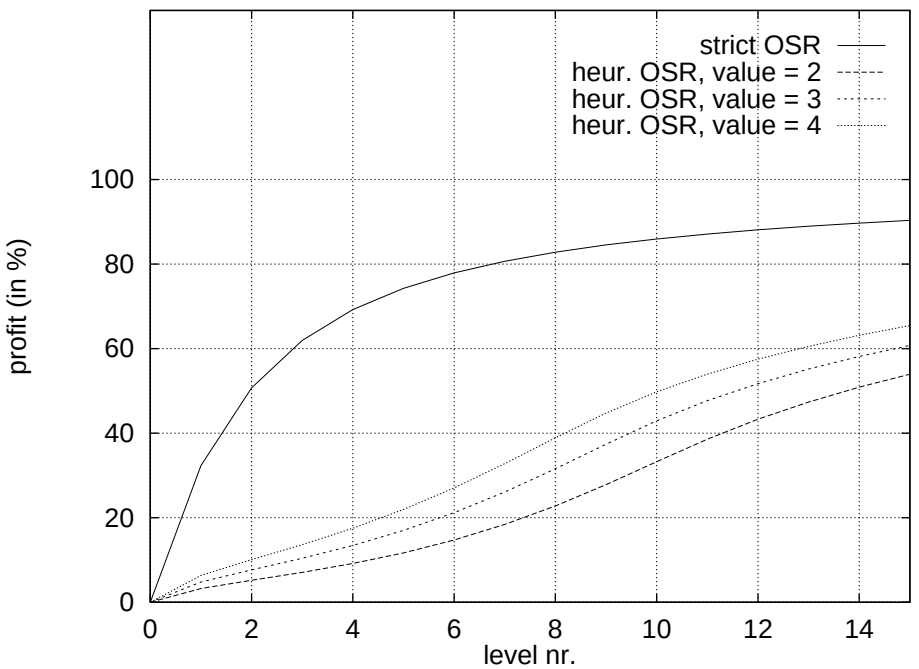


Fig. 5.11. *Profit of OSR expressed as a percentage compared to applying no OSR at all. The total number of voxels for the scale space up to each level have been cumulated from level 0. The input is a three-dimensional image with* $N_0 = 256$.

In Fig. 5.11 the profit has been plotted for a 3D image with $N_0 = 256$, according to equations (5.14) and (5.15). Strict OSR clearly outperforms heuristic OSR, irrespective of the actual value of ρ. Other values for N_0, or two-dimensional images, give

similar results, as can be understood by inspecting equations (5.14) and (5.15) more closely.

Note that the profit of a pyramid-like approach for SRR (not shown in Fig. 5.11) even outperforms strict OSR for $\delta\tau < \ln 2$.

5.8.2 OSR and forced root areas

The fact that N_n decreases for increasing n implies that the search volume narrows down for pixels near the level boundaries. More precisely, if the OSR from level n to $n+1$ is large enough, then the corresponding search volume (defined by the radius $r_{n,n+1}$) might be too small for children in level n to even find one single parent in level $n+1$. These children are termed *forced roots*, since they have to be labeled roots, rather unvoluntary than imposed by a root labeling scheme in the segmentation phase.

In the worst case, a child located at a corner of the image at level n must find a parent in level $n+1$. The very corner point of that level can only be reached if:

$$r_{n,n+1} \;>\; R_{n+1}\sqrt{d} \quad , \tag{5.16}$$

or, equivalently:

$$k_n\,\sigma_{n,n+1} \;>\; K_{n+1}\,\sigma_{n+1}\sqrt{d} \quad . \tag{5.17}$$

For strict OSR this inequality will never be satisfied: the absolute scale σ_{n+1} is always larger than the relative scale $\sigma_{n,n+1}$, and K_{n+1} is much larger than k_n (see Fig. 5.8). Since $\sqrt{d} > 1$ there exist children that can not find a parent in the next higher level.

For heuristic OSR equation (5.17) simplifies to: $k_n > \rho\sqrt{2d}$. Clearly, the effect is much less dramatic.

In Fig. 5.12 the occurrence of forced roots is illustrated. Because the search volume has a spherical character (it is a circle in 2D), the corners are being rounded.

In order to investigate the area of forced roots more precisely, we created an artificial stack where every child links to the parent that is closest to the central axis of the stack (*i.e.,* to the middle of the image). This can be achieved by using (artificially created) levels with the following characteristics: every level has dimensions 256×256 and represents a cone pointing downwards (when viewed as a 3D landscape with image intensity as height feature). The point of the cone coincides with the center of the image, and the maximum intensity difference of each level (*i.e.,* corner point minus center pixel) equals Φ. More precisely, level n of this artificially created 'scale space' contains a reversed cone with intensities ranging from $n\,\Phi$ (midpoint) to $(n+1)\,\Phi$ (corner point). Based on these levels, we run a hyperstack using iso-intensity following only. Children in level n will always try to link to the parent in level $n+1$ that is closest to the center, because that will be the parent pixel closest to the child's intensity. The maximum distance that can be traveled through the scale space is determined by the search volume factor k_n. The pixels of this scale-tree that have been labeled as roots by the hyperstack segmentation process thus represent the minimum number of forced

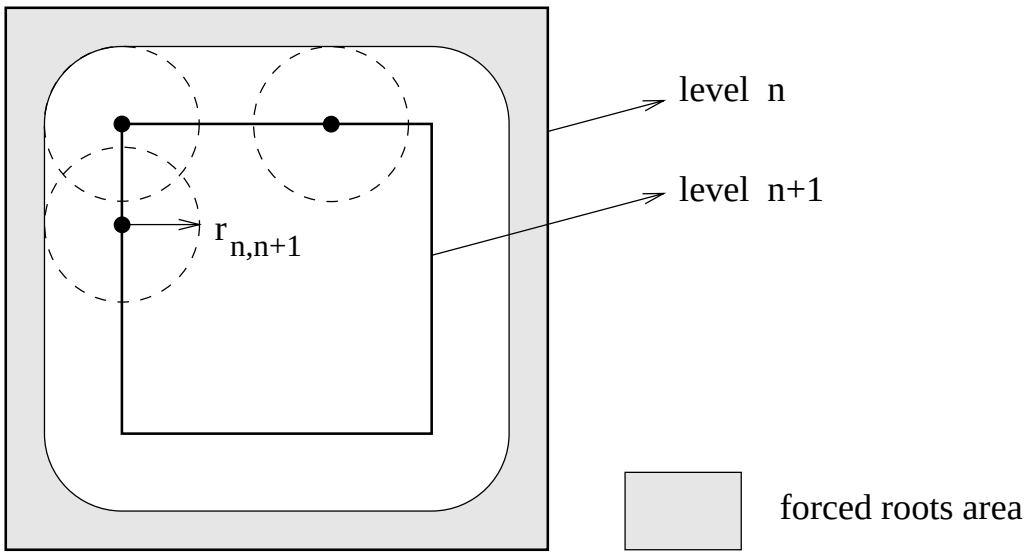


Fig. 5.12. *Schematic of the occurrence of 'forced roots'. If the OSR exceeds the maximum allowable distance—controlled by the search volume factor k_n—then a range of outmost boundary pixels will never find a parent in the next level.*

Fig. 5.13. *Outline of the minimum areas of forced roots (black pixels) obtained by applying strict OSR to an artificially created stack (for explanation see text). The forced root area is shown for (a) level 2, (b) level 4, and (c) level 6 of the corresponding hyperstack (these correspond to a σ of 2.0, 4.0, and 8.0 pixels, respectively).*

roots. For real world images the forced roots area expands if pixels are being linked to parents that in higher levels will become forced roots. This happens most frequently with background pixels that link towards the 'outside direction'.

The result of a hyperstack with strict OSR is shown in Fig. 5.13. To clarify the observation that the number of forced roots increases if more levels are added to the stack, we have shown the interim results of the levels 2, 4, and 6 (Fig. 5.13a, b, and c, respectively). The large area of forced roots indicates that only very specific images can be segmented using this approach. That is, the images must have an ROI located around the spatial center of the image. In particular, large scale information can only be calculated at a high accuracy if a sufficiently wide border surrounds the ROI. Note that small irregularities occur because of rounding errors in the linking process.

(a) (b) (c) (d)

Fig. 5.14. *Minimum areas of forced roots (black pixels) for heuristic OSR applied to the artificial stack (see text for detail). Fully converged hyperstacks have been created. The reduction factor increases from 2 to 5 (from (a) to (d), respectively).*

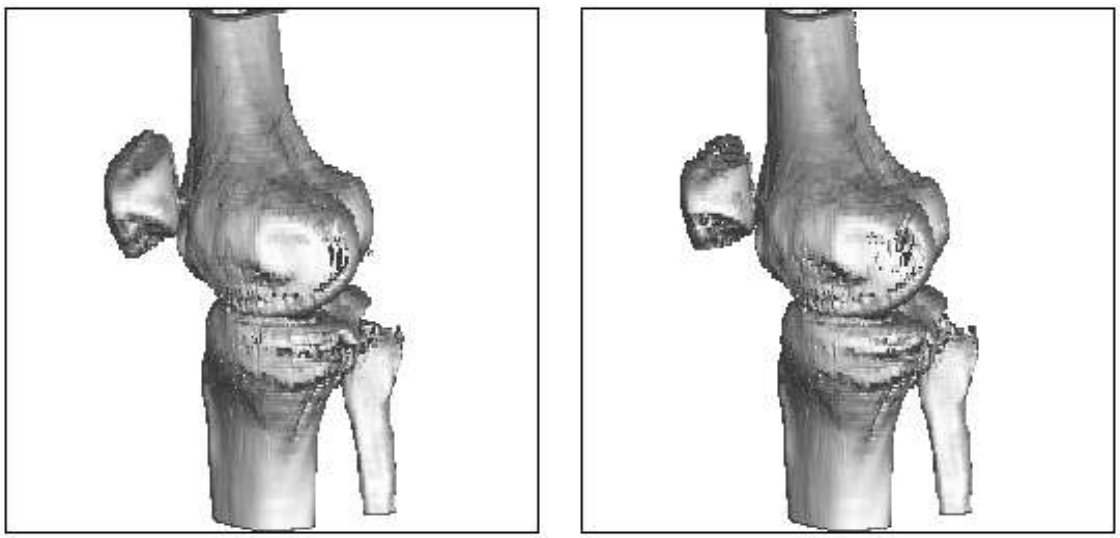
In Fig. 5.14 the forced root area for heuristic OSR is shown. In contrast with Fig. 5.13 (where we have created three small hyperstacks containing just a few levels), we now created four complete hyperstacks, with ρ ranging from 2 to 5. For each stack we needed 12 levels to impose convergence of the linkages. The final minimum areas of forced roots are indicated in black. As can be seen, the expansion drift of the forced roots area is much less dramatic. It is clear that heuristic OSR does not impose such rigorous requirements on the ROI as strict OSR.

5.8.3 The effects of OSR on hyperstack segmentations

From section 5.6.2 and Fig. 5.13 it may be evident that it is difficult to produce useful segmentations by applying strict OSR. Only a smaller $\delta\tau$ in combination with a larger search volume may produce useful results, but even then the number of segments will be too large for subsequent manual selection.

In the case of such a large number of segments, a practical representation of the segmentation results is by means of 'mean values' (see section 5.4.4), followed by a straightforward thresholding step. The influence of noise at pixel level has been

diminished by the very first linkages between the ground level and level 1, but the clustering of the small segments still needs be done manually. In fact, one can conceive the thresholding process as being executed on large, irregular 'pixels' (*i.e.,* the small segments). A hyperstack of only two levels is already an improvement compared to plain thresholding of the input image (the number of active pixels in level 1 is typically more than 30% smaller than in level 0). Note, however, that this does *not* solve the conventional thresholding problems (such as gradient levels in an MR image).



(a) (b)

Fig. 5.15. *Comparison of two renderings of the* KNEE *image, based on the hyperstack segmentation method with heuristic OSR: (a) reduction factor of 2; (b) reduction factor of 3. The qualitative differences are minimal, while the computation time decreases with about 40% for (b) compared to (a).*

In Fig. 5.15 we have compared two volumetric renderings of the KNEE image, a CT image of dimension $256 \times 256 \times 97$, based on two different (hyperstack) segmentations. The segmentation volumes have been produced using heuristic OSR with reduction factors 2 and 3, respectively. The profit in computation time of Fig. 5.15b compared to Fig. 5.15a was about 40%, while no significant differences are apparent. (It takes about 15 hours to process this image with a reduction factor of 3, and 25 hours when a reduction factor of 2 is used. All times have been measured on a HP 9000/755.)

For large images, like the KNEE image of Fig. 5.15, a straightforward segmentation based on a hyperstack without OSR is virtually impossible, owing to the huge amounts of memory and swap space that are needed to deal with the scale space and the linkages. Then, the use of OSR (even with a small reduction factor) can be very helpful.

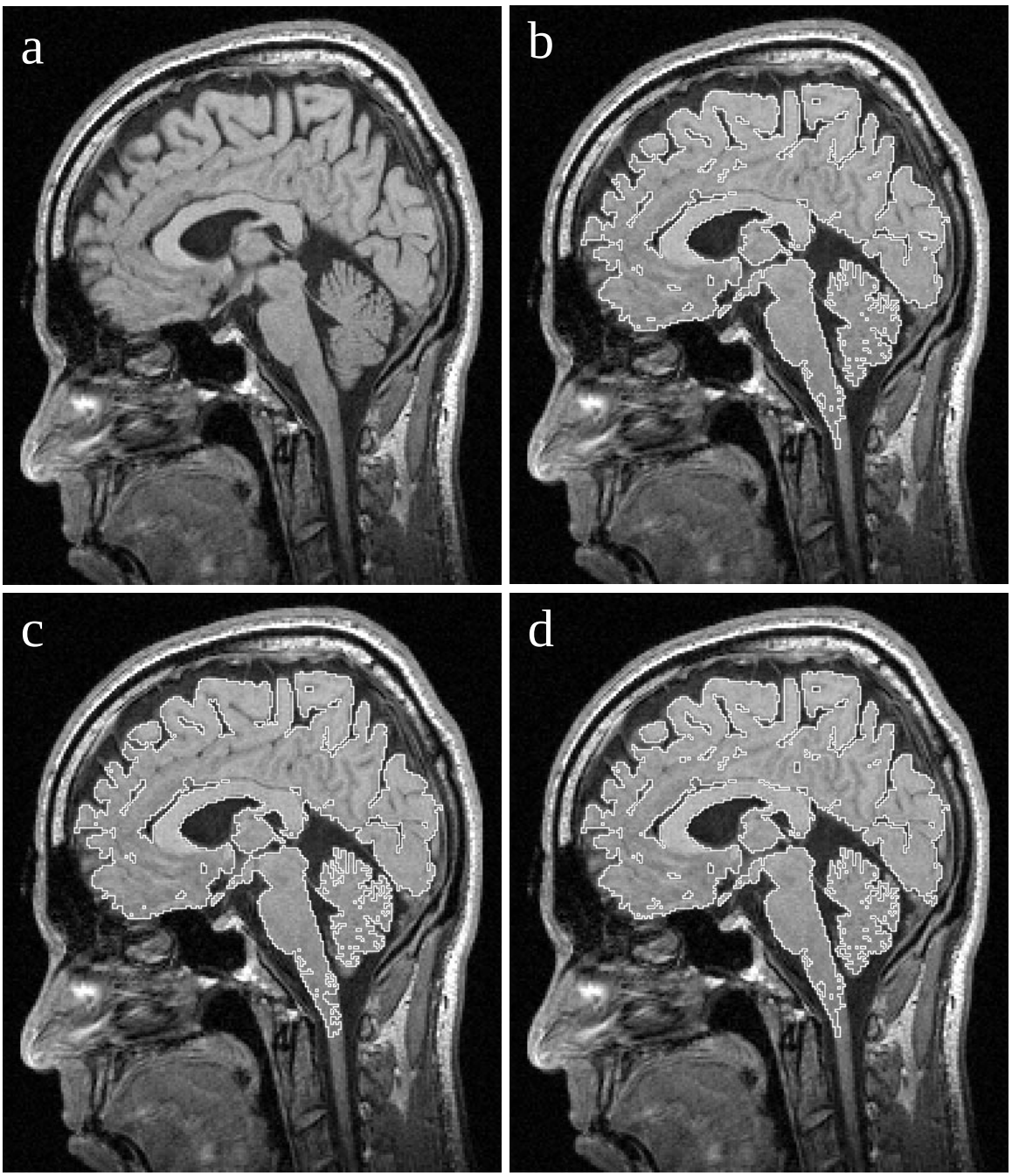


Fig. 5.16. *Comparison of three hyperstack segmentations of the* BRAIN *image. Shown are: (a) original image; (b) full hyperstack segmentation; (c) hyperstack segmentation with heuristic OSR of value 2; (d) idem, with value 3. For all cases, the contours of the segmentation have been superimposed in bright white onto the original image.*

In Fig. 5.16 we show the differences between a straightforward segmentation and heuristic OSR. For this particular BRAIN image we have focused on the white matter of the brain. Fig. 5.16c ($\rho = 2$) and Fig. 5.16d ($\rho = 3$) are very similar to Fig. 5.16b (*i.e.,* the conventional hyperstack segmentation with no OSR), although the computation time decreases from 30 to less than 9 minutes ($\rho = 2$) and to 6 minutes ($\rho = 3$), respectively: a profit of more than 70–80%! For larger values of ρ (not shown in Fig. 5.16) the time gain levels off, and the segmented images slightly begin to 'crumble'. This is a typical effect, caused by the fact that difficult areas (*e.g.,* areas with a lot of partial volume pixels, or thin, elongated objects) tend to keep on linking upwards without significantly extending the current segment size by merging with other segments. (This can be examined by looking at the corresponding ground volumes at each level.) At a certain—relatively high—level in the hyperstack, they are forced to link to 'bad' parents, because convergence is imposed. During the segmentation phase, these pixels are perfect candidates to be labeled as roots, which results in small, crumbled segments at the ground level.

This effect need not necessarily be disastrous for the final segmentation result. In most cases, an increase in the total number of desired segments suffices to obtain useful results, although a smaller ρ value is probably an easier solution.

In Fig. 5.17 a similar comparison has been made for the TRANS image, a 2D transversal MR image of size 256×256. This time, the profit of using heuristic OSR increases to 80% ($\rho = 2$), 85% ($\rho = 3$). Again, using heuristic OSR for $\rho > 3$ produces slightly crumbled segmentations, although the effect is less than for figure Fig. 5.16a. This is caused by the fact that the original TRANS image (Fig. 5.17a) is more blurred in itself than Fig. 5.16a.

The pixels that have difficulty finding a good parent (in this case mainly the partial volume pixels between the white and grey matter) are forced to choose. This leads to an increase in relatively small segments containing the partial volume pixels. Such a segmentation will look 'thin': only the pixels that belong to the white matter with high certainty remain present in the segmentation based on heuristic OSR with $\rho > 3$.

In Fig. 5.18 the effect of OSR on the volumetric rendering of the TUMOR image, a three-dimensional image with dimensions $256 \times 256 \times 128$, is shown. The patient has a tumor in the right frontal lobe (shown as a hole in the grey matter). The hyperstack segmentation without applying OSR could only be made by dividing the large image into several pieces. This requires a lot of administration and computing time. If OSR is applied, a much coarser subdivision is possible, because less voxels and linkages are involved per (sub-)hyperstack.

There is hardly any difference in quality between the full hyperstack segmentation and the one generated by applying heuristic OSR with a ρ-value of 2. The inlets in the grey matter are somewhat deeper for heuristic OSR ($\rho = 2$) compared to the full hyperstack, but the location of the tumor is still clear. Nonetheless the time difference for the two segmentations is significant: approximately 60 hours and 13 hours effective computing time. In Fig. 5.18c small artifacts show up, while the additional profit in

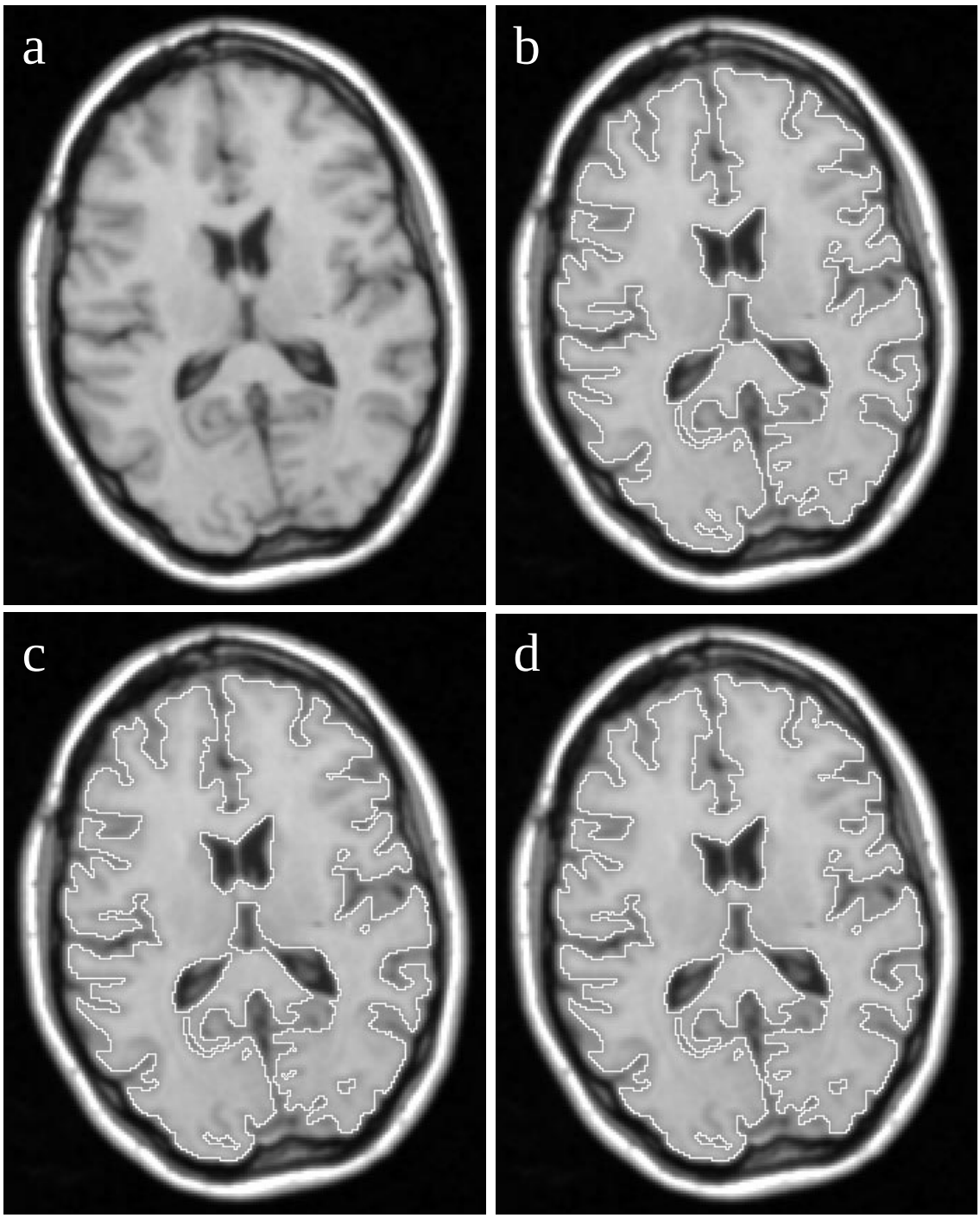


Fig. 5.17. *Comparison of three hyperstack segmentations of the* TRANS *image. Shown are: (a) original image; (b) full hyperstack segmentation; (c) hyperstack segmentation with heuristic OSR of value 2; (d) idem, with value 3. The contours of the segmentation have been superimposed in bright white onto the original image.*

computing time is marginal: 9 hours. Without *a priori* information on the pathology of this patient, the tumor is easily missed. In this case, the loss in quality does not excuse the computational profit.

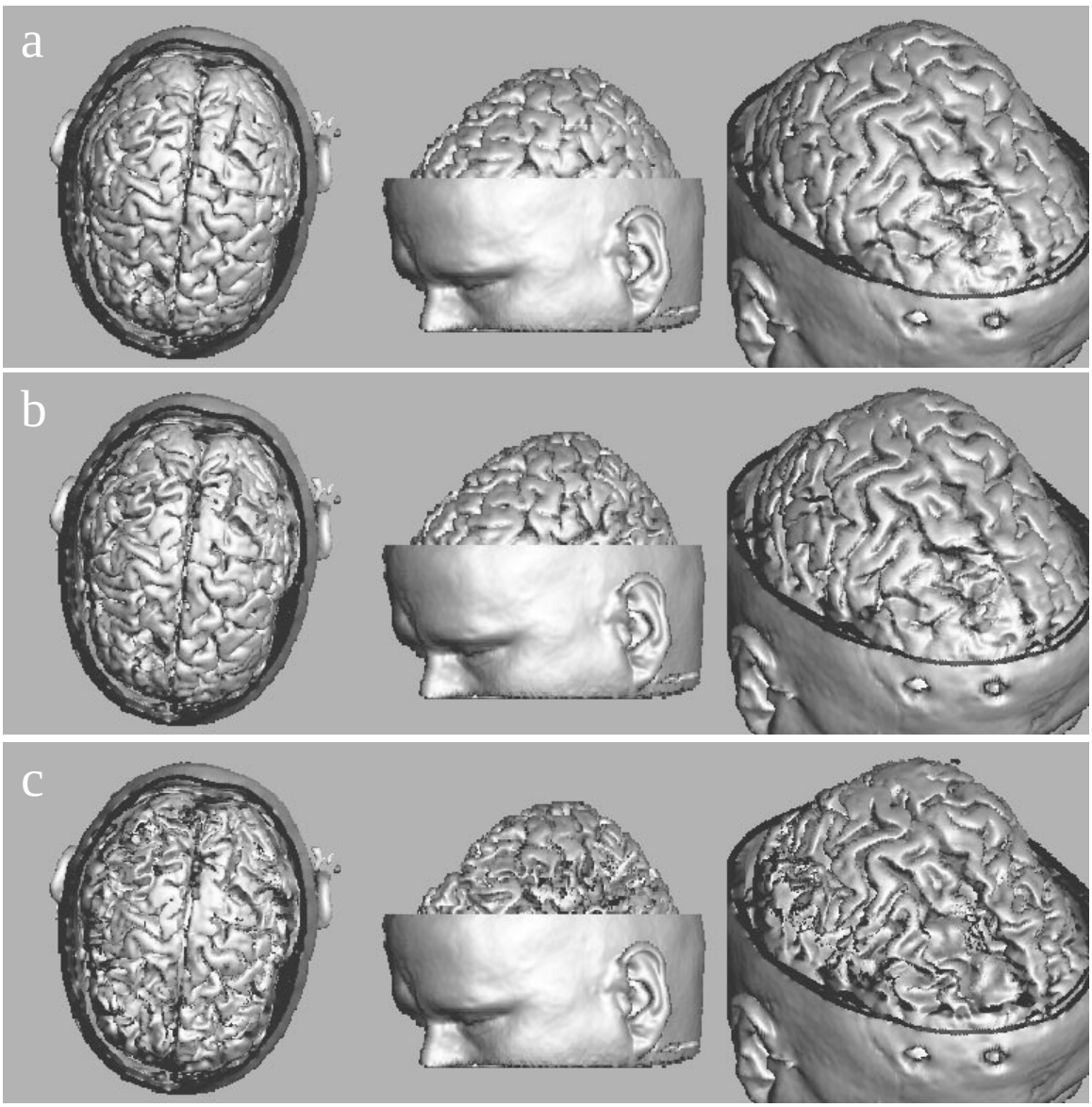


Fig. 5.18. *Comparison of three series of renderings based on hyperstack segmentations of the* TUMOR *image. Shown are: (a) full hyperstack segmentation without OSR; (b) hyperstack segmentation with heuristic OSR,* $\rho = 2$*; (c) idem,* $\rho = 3$.

5.9 Conclusions

In this chapter we have discussed the boundary problem of a multiscale image analysis. We have argued why sampling rate reduction in its conventional form can not easily be applied to real-world images that have been blurred to create the levels at larger scale. Instead, outer scale reduction as non-uniform SRR technique has major advantages over straightforward SRR, in particular with respect to the computation time.

We have explained two different variants of outer scale reduction: strict OSR and heuristic OSR. Their usefulness has been tested by the hyperstack, a multiscale segmentation technique. We have indicated what the effect of OSR is on the generated scale space and on the segmentations.

Based on our research we draw the following conclusions:

- SRR in its original form (*e.g.,* conform the Nyquist criterion) does not seem very promising for multiscale image analysis.
- It is hard to define the requirements for OSR that are *generally* applicable in multiscale image segmentation. Strict OSR requires that the ROI is located in the center of the image. The parts at the image boundaries 'disappear' if scale is increased and hence can not be segmented properly. For heuristic OSR, this requirement is much weaker. Only the pixels close to the boundary of the input image vanish with increasing scale.
- The amount of outer scale reduction is flexible for the heuristic OSR. Although the actual maximum is image dependent, a reduction factor of 2 seems a good default value.
- The profit in computation time by using OSR is enormous. For the hyperstack the use of heuristic OSR (value 2) will result in time savings of at least 70%.
- Strict OSR solves the boundary problem—involved with the blurring kernel extending beyond the image borders—completely, albeit at the cost of a too steep reduction of the volume-of-interest. Heuristic OSR significantly reduces the boundary problem: the 'inaccurately' computed boundary pixels at a large scale are ignored.
- The hyperstack scale-tree converges faster if OSR is applied, because the number of candidate parents decreases. This has the desirable effect that less segments are needed to completely represent an object. If the reduction factor is set too high, the convergence may become too strong. Experiments yield the preliminary conclusion that no significant problems are expected for reduction factors up to 3.

'There is nothing more musical than a sunset'

– Claude Debussy

Chapter 6

Blurring strategies for hyperstack image segmentation

Abstract

Hyperstack image segmentation consists of four consecutive steps: (*i*) blurring, *i.e.,* constructing a scale space by subjecting the original image to a sequence of filtering steps with an increased smoothing effect; (*ii*) linking, *i.e.,* establishing parent-child linkages between pixels of two successive levels in the scale space; (*iii*) root labeling, *i.e.,* determining which of the pixels in the scale-tree represent a segment in the input image; and (*iv*) the downward projection step to obtain an actual segmentation. Up to now, the scale space in hyperstack segmentation has been constructed invariably by means of a Gaussian kernel. This chapter investigates alternatives to Gaussian blurring in hyperstack image segmentation.

We have compared six different scale space generators, *viz.* the Gaussian kernel in both the spatial and in the Fourier domain, variable conductance diffusion according to Perona & Malik, Euclidean shortening flow as proposed by Alvarez, Lions, & Morel, the median filter, and the Kuwahara filter. Hyperstack segmentation is found to be rather insensitive to the choice of the underlying scale space generator. This is shown by evaluating the segmentations through visual inspection and by means of a quantitative algorithm, both for artificial and real-world images with an available gold-standard.

Keywords: Blurring, boundary problem, multiscale image analysis, scale space, nonlinear diffusion, image segmentation, median filter, Kuwahara filter.

6.1 Introduction

The *scale space* [54, 131] of an image provides a framework to process and analyze the image at multiple levels of scale simultaneously. The combination of local and global information has proven to be useful in various applications, *e.g.,* edge detection [17], image segmentation [86, 64, 123, 124], segmentation in combination with neural

networks [44], description of higher order image structure [97, 98], and multimodality matching [68, 69, 35].

There is not one unique scale space for an image, nor a unique way of using the scale space for image analysis. The options comprise the choice of the scale space generator, the number of—discrete—scale space levels used, the sampling in the scale direction, and the sampling in the spatial directions [119]. In this chapter we expand on the choice of the scale space generator.

We focus on applications that use scale spaces as input to a *linking model* [86, 64, 57, 39], which in turn serves as the basis for segmentation of the input image. The linking model establishes parent-child linkages between pixels in adjacent levels of the scale space, thus creating a tree-like structure of pixel connections. The segmentation method we consider is the hyperstack, which has proven to be a promising method for segmenting multidimensional images [121, 122, 123, 124, 120, 126]. We investigate how crucial the choice of the underlying scale space generator is.

Bangham [8] already noted that the well-known linear scale space kernels [131, 54, 66, 30, 29] are not the only viable options to construct a scale space. Therefore, in this chapter we will deal with both linear (section 6.2) and nonlinear scale space filters [34] (sections 6.3, 6.4, and 6.5). The six different scale space constructors that we have compared are:

1. The Gaussian scale space implemented in the spatial domain (section 6.2.1);

2. Ditto, implemented in the Fourier domain (section 6.2.2);

3. The nonlinear scale space based on the Perona & Malik equation (section 6.3.1);

4. Ditto, based on the Euclidean shortening flow of Alvarez, Lions, & Morel (section 6.3.2);

5. The (nonlinear) median filter (section 6.4);

6. The (nonlinear) Kuwahara filter (section 6.5).

In section 6.6 we discuss the main front-end scale space axioms and define the properties that are desirable for a scale space generator. In the same section we outline a method to synchronize the scale parameters between the different scale space constructors, in order to enable a fair comparison by the hyperstack algorithm.

For the experiments we have used both artificial and real-world (medical) images. The results are presented in three ways: (*i*) the scale spaces are compared (section 6.7.1); (*ii*) the segmented images are qualitatively compared (see section 6.7.3); (*iii*) the segmentations are quantitatively compared by an objective evaluation algorithm (see section 6.7.4) of the segmented images that result from the hyperstack segmentation method.

Finally, in section 6.8 we draw some general conclusions.

6.2 Linear scale space

A linear scale space is—according to previous work done by, *e.g.,* Witkin[131], Koenderink[54], and Florack[33]—a one parameter family of images generated by convolution of the input signal with a Gaussian kernel of increasing width σ.

The Gaussian is the unique constructor of the linear scale space as a consequence of four requirements (the so-called *front-end vision postulates*), which express that a front-end system should be able to construct 'a complete, homogeneous, isotropic, and scale invariant representation of its input data' [30].

Koenderink derived the Gaussian kernel from quite a different point of view [54]. In order *not* to generate 'spurious detail' by blurring (the *causality* requirement), he derived that a multiscale representation of an image should be constructed by the linear diffusion equation:

$$\frac{\partial L}{\partial t} = \Delta L \quad , \tag{6.1}$$

where $L \equiv L(\vec{x}, t)$ is a scalar function (*e.g.,* luminance) over the image domain. This also leads uniquely to the Gaussian kernel, which is the Green's function of the linear diffusion equation.

According to the property of scale invariance [30] the blurring strategy has to follow an exponential sampling in scale space. Hence, we introduce a natural scale parameter $\tau = n\,\delta\tau$, $n \in \mathbb{N}$, which is related to the absolute scale σ_n at level n by:

$$\sigma_n \;=\; \varepsilon\, e^{\tau_0 + n\,\delta\tau} \quad , \tag{6.2}$$

where τ_0 is the initial scale offset and ε is taken to be the smallest linear grid measure of the imaging device (the *inner scale*). If we denote level n of the discrete scale space by $\mathcal{S}_n$, then we can write for the *scale space representation* $\mathcal{S}$ of the d-dimensional input image $L(\vec{x})$ (*i.e.,* the collection of d-dimensional images at increasing scales):

$$\mathcal{S}_n \;=\; \mathcal{S}(\vec{x}, \sigma_n) \;=\; L(\vec{x}) \;*\; G(\vec{x}; \sigma_n) \quad , \tag{6.3}$$

with initial conditions $\mathcal{S}(\cdot;\, 0) \;=\; L(\cdot)$ and $\tau_0 = 0$. Hence, the dimensionality of a scale space $\mathcal{S}$ is $d + 1$ for a d-dimensional input image.

The implementation of the Gaussian scale space may be carried out in the spatial domain or in the Fourier domain. Either option has pros and cons, as will be discussed below.

6.2.1 Spatial domain implementation

If we opt for the spatial domain implementation of the Gaussian scale space, we have two problems: (*i*) the discretization of the kernel, and (*ii*) the boundary problem.

The first problem is tackled by using a straightforward discrete implementation of the Gaussian kernel. Gaussian convolution is self-similar: a blurring step by a Gaussian kernel of width σ_1 followed by a second blurring step with width σ_2 equals a single

blurring step by a Gaussian of width $\sqrt{\sigma_1^2 + \sigma_2^2}$. Unfortunately, in the discrete domain the concatenation of sampled Gaussians is *not* a discrete scale space transformation [66]. Hence, a discrete scale space is best constructed by convolution of the *original* image with sampled Gaussian kernels of increasing width.

The second problem is more difficult to deal with. The boundary problem is well-known in filter analysis in general. If we restrict ourselves to multiscale image analysis, then a short overview of commonly used techniques can be found in [119]. In this chapter we will use average value extrapolation of the image to estimate the unknown boundary pixels. That is, the average intensity value of the original image is used as global extrapolation value for kernels that overlap with the image boundary. This method forces the blurring at increasing scales to converge to precisely the same average value.

Note that this imposes the same requirement on the implementation as the discrete Gaussian does: the convolution can *not* be implemented in consecutive steps, because then the flux through the image boundaries is not accounted for. Instead, the original image must be used to derive *every* blurred replica directly. (Deriche [26] has shown that a recursive implementation of the discrete convolutions decreases the computational complexity significantly—even for very large filters.)

For the actual implementation of the discrete convolution—including speed-up techniques—we refer to appendix 6.A.1.

6.2.2 Fourier domain implementation

As an alternative, the blurring can be performed in the Fourier domain. In this case a repetition of the signal is assumed, which leaves two options for dealing with the boundary problem. We can simply repeat the signal in every Euclidean direction, or perform a mirroring step first. In the latter case, the repetition pulse spans a distance that equals twice the length of the original signal (in each direction).

Irrespective of which method is used, the fact that the original data is simply repeated introduces huge artifacts at increasing scales. Generally (*i.e.,* if the brighter areas are in the center of the image), this reflects itself as bright blobs that show up at the boundaries of the images at large scales.

The main advantage of blurring in the Fourier domain is the processing speed at large scales: convolution in the spatial domain equals multiplication in the Fourier domain, hence the computation time is now independent of the width of the used Gaussian kernel.

In this chapter, we have applied straight (*i.e.,* unmirrored) repetition in the Fourier domain implementation.

6.3 Nonlinear scale space

In recent years a number of nonlinear smoothing techniques have been developed. Although there is a wide range of approaches yielding a rich diversity of generating equations, we can roughly distinguish three approaches which cover a wide range of schemes: (*i*) diffusion equations with variable conduction coefficients, (*ii*) geometrically-driven curve or surface evolutions, and (*iii*) morphological schemes. We will briefly discuss the approaches (*i*) and (*ii*) to introduce two equations which are used in the remainder of this chapter to generate a stack of images. The third approach is closely related to approach (*ii*) and models an evolution equation using morphological operations with locally defined (convex) structuring elements [71, 19, 113, 1]. The relation is beyond the scope of this thesis but the interested reader may consult [14] where the authors describe a method to relate functional morphological operators to nonlinear partial differential equations.

6.3.1 Variable conductance diffusion

The first approach is essentially a modification of the linear diffusion (or heat) equation. A general form of a diffusion equation with conduction coefficient c is given by:

$$\frac{\partial L}{\partial t} = \nabla \cdot \Big(c \nabla L \Big) \ . \tag{6.4}$$

In the particular case that c is a constant we have the linear diffusion equation, which has been described in section 6.2.

If the heat conduction coefficient c is not a constant anymore but depends on local image properties, as *e.g.,* proposed by Perona & Malik in [84], the resulting smoothing process is no longer linear. The underlying idea is to choose c such that relevant objects smooth while not affecting one another. Two choices that Perona & Malik proposed for $c(\vec{x}, t) = g(\|\nabla L(\vec{x}, t)\|)$ are:

$$g_1\Big(\|\nabla L\|\Big) = e^{-\left(\frac{\|\nabla L\|}{K}\right)^2} \tag{6.5}$$

$$g_2\Big(\|\nabla L\|\Big) = \frac{1}{1 + \left(\frac{\|\nabla L\|}{K}\right)^2} \tag{6.6}$$

Here K is a free parameter which determines whether a gradient is significant or not. Both choices are decreasing with the magnitude of the gradient. A drawback of equation (6.4) is that it locally behaves like the inverse heat equation if $c(\vec{x}, t)\nabla L$ is decreasing with respect to ∇L for a point $(\vec{x}, t)$ (which happens for $\|\nabla L\| > \frac{1}{2}\sqrt{2}K$ in (6.5) and for $\|\nabla L\| > K$ in (6.6)). Furthermore, the gradient is not calculated at a certain scale which implies that insignificant edges (for example resulting from high frequency noise) will also be kept. Fortunately, a solution to the latter problem also solves the first; a regularization proposed by Catté *et al.* [18] in which the equations

(6.5) and (6.6) are evaluated using the gradient evaluated at a certain scale $DG_\sigma * L$ rather than the gradient at the scale level of the original image (which is physically not possible!) is well-posed. In this chapter we will use $g_1(\|\nabla L\|)$. Extensions towards vector-valued diffusion have been made by Whitaker & Gerig [130].

Although the Perona & Malik equation is often referred to as an anisotropic diffusion equation it in fact is an inhomogeneous diffusion equation . In [128] truly anisotropic diffusion is studied using a heat conduction tensor.

6.3.2 Curve evolution

The second approach to nonlinear scale spaces studies the evolution of curves or surfaces as a function of their geometry. Since the planar case of evolving curves is by now well understood [108, 36, 109, 37, 28, 38, 81, 4, 3] we will briefly outline the main steps in the derivation.

A general evolution of a curve can be written as:

$$\frac{\partial C}{\partial t} = \alpha(p,t)\vec{T} + \beta(p,t)\vec{N} \tag{6.7}$$

where $\vec{T}$ and $\vec{N}$ denote the tangential and normal unit vector to the curve respectively. However, the tangential component only affects the parameterization which implies that we can rewrite $\{\alpha, \beta\}$ as $\{0, \beta'\}$ [28] and if we require β' to be a function of local geometry we find

$$\frac{\partial C}{\partial t} = g(\kappa, \kappa_p, ...)\vec{N} \tag{6.8}$$

where κ denotes isophote curvature and p is the arclength parameter.

In this chapter we will consider the case $g = \kappa$, which leads to the well-known Euclidean shortening flow. The more general choice $g = \alpha\kappa + \beta$ was introduced in computer vision by Kimia *et al.* [52, 51, 53] while Sapiro & Tannenbaum studied extensions towards affine invariant evolutions in which case $g = \kappa^{\frac{1}{3}}$ [100, 99].

The particular choice we use here has a very intuitive interpretation if we rewrite it in so-called gauge coordinates ((v,w)-gauge, see figure Fig. 6.1).

In this gauge the v direction denotes the local isophote direction while w denotes the normal direction. From $\kappa \stackrel{def}{=} \frac{\partial^2 w}{\partial v^2}$ we find: $\kappa = -\frac{L_{vv}}{L_w}$ which implies the following evolution of the luminance function:

$$\frac{\partial L}{\partial t} = L_{vv} \tag{6.9}$$

which can be interpreted as a diffusion equation which only diffuses in the local isophote direction (note that $\triangle L = L_{vv} + L_{ww}$). Thus, the net result is that no smoothing occurs in the gradient direction (and therefore the diffusion is very anisotropic). This equation has been derived independently by Alvarez *et al.* in [2] because of this

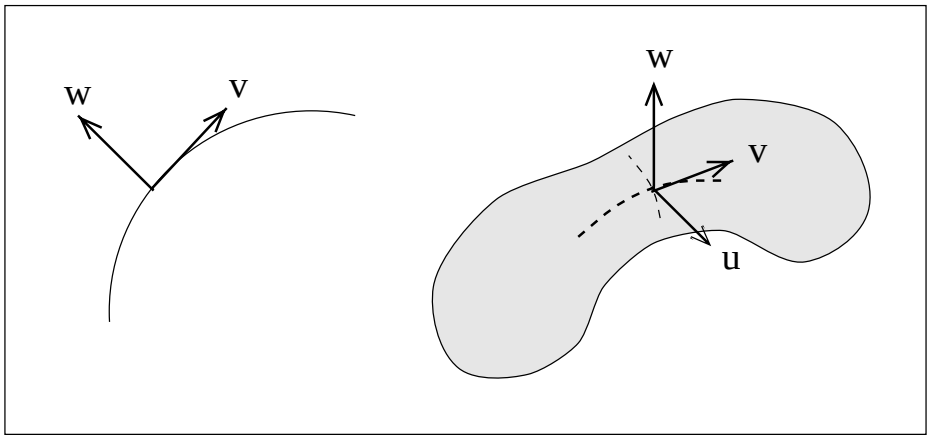


Fig. 6.1. *Local gauge coordinates in 2D and 3D (w points in the normal direction). In 3D u and v denote the principal curvature directions.*

exact property. This derivation does not readily extend to three dimensions, although some approaches have recently appeared in literature [77, 129].

As mentioned before we have to resort to numerical methods to obtain the stack of images generated by the evolution equations that were described in the previous section. In this respect, we have to distinguish between *implicit* and *explicit* schemes [79]. Implicit refers to the fact that the image of the next time step also depends on image information at the next step, whereas in an explicit scheme the image at the next time step is estimated by using only the information that is available at the current time step. We use a simple (explicit) Euler forward scheme. The problem with implicit schemes is that they are difficult to construct if the operators, used to extract the geometrical information, have a large window (if we use Gaussian operators the window is—at least in theory—infinite). We choose to extract the differential operators using scaled Gaussian operators rather than finite differences to build in some of the desired properties from the start. We briefly comment on the implementation in appendix 6.A.2 and refer to [78] for a more extensive treatment.

Free parameters in our scheme are the time-step in the Euler forward scheme, the scale at which the Gaussian derivative operators are calculated, and in the Perona & Malik scheme a weighting factor which is used as a threshold determining whether a gradient is significant or not. A few considerations greatly reduce the parameter space. We want the scale of the derivative operators to be as small as possible and a time-step as large as possible. Lower limits for the scale of the derivative operators are set by the noisiness of the image and the inner scale [42]; aliasing in the Fourier domain is a function of the order of differentiation and the operator scale expressed in inner scale units. We therefore select a scale parameter based on these considerations. Typically, for a second order invariant we select $\sigma \approx 0.8$ pixel units. From [78] follows the maximal time-step allowed.

The parameter K which measures whether a gradient is significant or not is set to

the 90%-value of the cumulative histogram of the image gradient.

6.4 Median filtering

The median filter has originally been designed to remove 'salt-and-pepper' noise from data. Small speckles in the image are simply removed, while the size of the speckle noise is controlled by the size of the median filter.

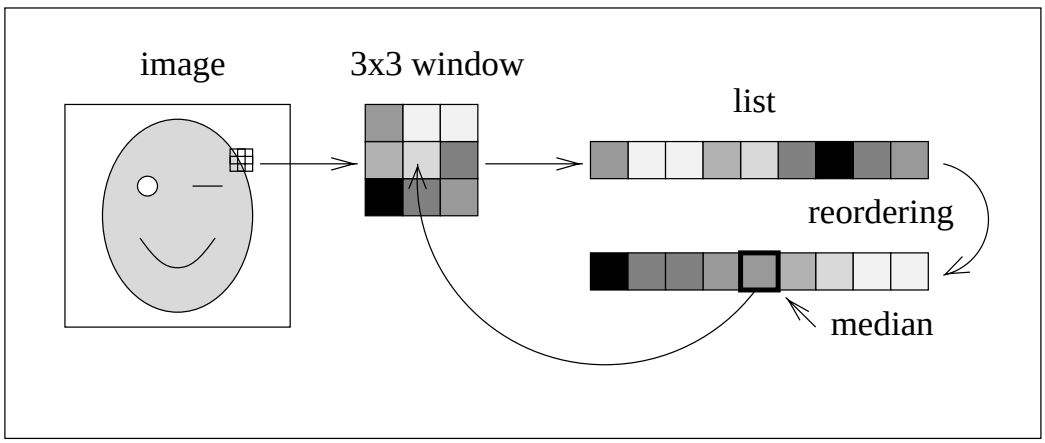


Fig. 6.2. *Schematic of the median filtering of a 2D image by a window of size* 3×3.

The filter design is fairly simple (see Fig. 6.2), although the computation time may become significant—especially if the window size is increased. The basic idea is to shift a squared window of size $m \times m$ (m odd) over the entire image. All the pixels covered by the window are reordered in increasing order. The centered pixel of the window is then replaced by the middle data point of the reordered list (*i.e.,* the 'median'). Since the exceptional pixels (*i.e.,* black and white noise pixels) will find their place at the begin or the end of the reordered list, these noise pixels will be ruled out after median filtering.

Near the image boundary, the window has to be adapted, or an extrapolation step is necessary. In this chapter, we stick to smaller window sizes, although there is no clear preference for either method. As a consequence, it may happen that the total number of elements in the list that need to be reordered is even. In that case, the average value of the two middle data points is taken to be the median.

Several extensions to the basic scheme are possible:

- Use a circular window. This is slightly more difficult to implement, but is more fair from a rotation invariance point of view.

- Use only existing data points as median values. If the total number of elements in the list is odd, this will always be true, but if this number is even (see above),

a new value may be introduced. This is especially annoying at images with a relatively low number of grey levels, such as segmented images. In that case, the one pixel of the two middle pixels that is closest to the original pixel value is taken to be the median. This will only occur at the boundary of the image, since elsewhere m is odd.

- Use the average value of the middle $\widehat{m}$ pixels. For the conventional median filtering, $\widehat{m}$ is taken to be 1. A higher degree of smoothing is achieved by increasing $\widehat{m}$. If $\widehat{m}$ is given its maximum value (*i.e.,* m^2), the median filtering converges to straight averaging over an $m \times m$ window.

- Use the *pseudomedian* filter [89, 105], which is a debilitated variant of the median. The one-dimensional pseudomedian PMED_5 with a filter width of 5 is defined as:

$$\begin{aligned}\mathrm{PMED}_5\Big\{a,b,c,d,e\Big\} &= \frac{1}{2}\max\Big\{\min\{a,b,c\}\,,\,\min\{b,c,d\}\,,\,\min\{c,d,e\}\Big\} \\ &\quad + \frac{1}{2}\min\Big\{\max\{a,b,c\}\,,\,\max\{b,c,d\}\,,\,\max\{c,d,e\}\Big\}\ . \quad (6.10)\end{aligned}$$

  Pratt [88] later introduced the two terms 'maximin' and 'minimax' for the two halves of equation (6.10). Schulze and Pearce [104] extended the definition of the one-dimensional pseudomedian filter to two dimensions. They also proved the equivalence relationship with the morphological *opening* and *closing* operations. More precisely, equation (6.10) can be written as:

$$\mathrm{PMED}\Big\{L(\vec{x});\,\phi\Big\} = \frac{1}{2}\,\mathrm{Open}\Big\{L(\vec{x});\,\phi\Big\} = +\frac{1}{2}\,\mathrm{Close}\Big\{L(\vec{x});\,\phi\Big\}\ , \quad (6.11)$$

  where ϕ equals the morphological structuring element.

It is beyond the scope of this thesis to test all the variants of the conventional median filter, although it is obvious that changing the filter definition may affect the results. For instance, the response of the pseudomedian filter is distinct from the plain median filter on two grounds: (*i*) the amplitude of isolated pixels are halved rather than completely removed, and (*ii*) new pixel values may be introduced because of the averaging character (see also [105]).

In appendix 6.A.3 we expand on the implementation of the median filter.

6.5 Kuwahara filtering

In the original Kuwahara paper [62] the filter is described as a preprocessing step to facilitate the inspection of radioisotope (RI)-angiocardiographic images of the cardiovascular system. At that time there was no suspicion that nonlinear filtering would become so increasingly popular.

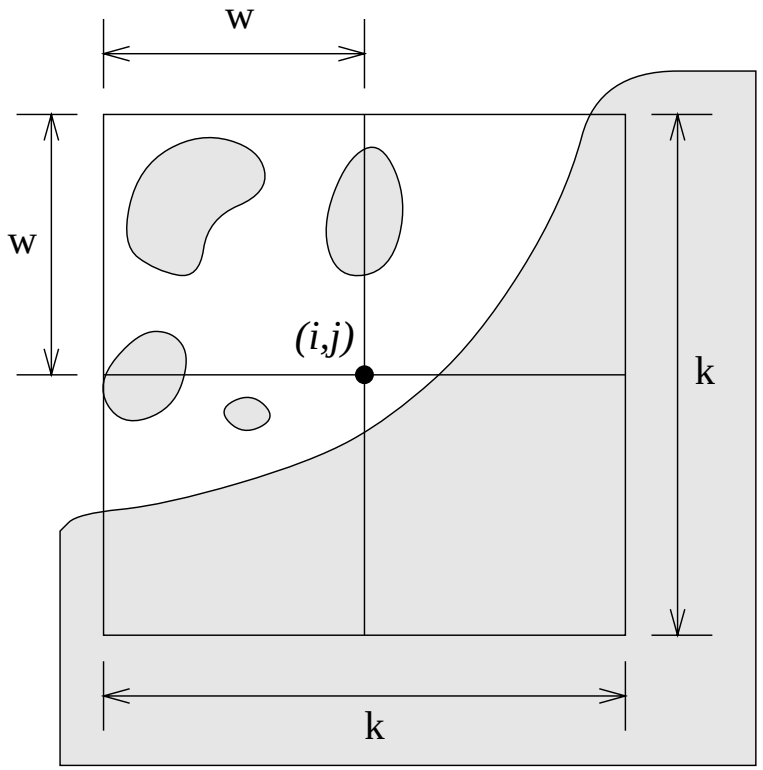


Fig. 6.3. *Schematic of the original Kuwahara filtering of the pixel* (i, j) *in a 2D image by four rectangular neighborhoods (windows) of size* $w \times w$ *each. The total kernel scope equals* $k \times k$*, with* $k = 2w - 1$ *(i.e., the quadrants overlap with one pixel).*

Kuwahara used the sliding window concept, similar to the median filter of the previous section. The original idea was to compare the *variance* in each of the four windows (the 'quadrants' of every pixel), and replace the value of the pixel at hand by the *mean* value of all the pixels in the window with the smallest variance (see Fig. 6.3). This way, it is stimulated that homogeneous areas arise, because windows crossing edges will have a relatively large variance.

Normally, the quadrants of size $w \times w$ ($w > 1$) overlap by a line of one pixel width. For symmetry reasons we require the total kernel scope k per pixel to be odd (*i.e.,* $k = 2w - 1$).

The choice of four rectangular neighborhoods is rather *ad hoc*. Schulze and Pearce [106, 107] suggested to extend the conventional Kuwahara filter to a range of sub-windows, *i.e.,* to use all w^2 possible windows of size $w \times w$ for each pixel. In this way, a substantial amount of overlap between the different windows is achieved, which makes the filter much more robust. A second advantage is that scale transitions are smaller, because the probability that one of the windows has a smaller variance increases if more sub-windows are checked. The paper refers to the filter as a 'Mean-of-Least-Variance' (MLV) filter, but is in essence comparable to the conventional Kuwahara

filter. The d-dimensional MLV filter is formally described by:

$$M_N(\vec{x},k) = \frac{1}{k^d}\sum_{\vec{y}\in N_{\vec{x}}} L(\vec{y}) \quad (6.12)$$

$$V_N(\vec{x},k) = \frac{1}{k^d}\sum_{\vec{y}\in N_{\vec{x}}} \left| L(\vec{y}) - M_N(\vec{x},k) \right|^2 \quad (6.13)$$

$$\mathrm{MLV}_N\left\{\vec{x};k\right\} = M\left(\left\{\vec{x}:\ \vec{x}\in N_{\vec{x}} : v(\vec{x}) = \min\left\{v(\vec{y}) : (\vec{y})\in N_{\vec{x}}\right\}\right\}\right) \quad ,(6.14)$$

where $N_{\vec{x}}$ denotes the window centered at position $\vec{x}$, and $M_N(\vec{x},k)$ and $V_N(\vec{x},k)$ denote the mean and variance over this region, respectively.

For implementation details of the Kuwahara filter we refer to appendix 6.A.4.

6.6 Comparison of the scale space generators

In this section we briefly summarize: (*i*) the main characteristics of the different scale space generators described in the previous sections (section 6.6.1), and (*ii*) the responses of each generator to a series of desirable properties (section 6.6.2). Furthermore, in section 6.6.3 we discuss how to synchronize the scale of the levels corresponding to nonlinear scale spaces with those of the linear scale space. It is shown for all methods that this can be achieved.

Note that we talk about 'blurring' for any scale space generation process, although some constructors do not perform a blurring in the very strict meaning of the word (*i.e.,* smoothing). Moreover, we will talk about 'scale' spaces, even though there is not always an explicit connection between the blurring process and scale.

6.6.1 Front-end scale space axioms

Below we briefly describe the basic properties that a front-end vision system must have. We closely follow the 'front-end vision postulates' as defined by Florack in [29]:

- *Linearity.* The scale space generator is a linear operator.
- *Homogeneity.* The blurring is said to be homogeneous if all the locations within the field of view are considered *a priori* equivalent.
- *Isotropy.* There is no *a priori* preference for any direction of blurring in any point.
- *Self-similarity.* The result of two successive blurring steps b_1 and b_2 can be performed by a single (larger) blurring step $b_3 = b_1 \oplus b_2$, in which '$\oplus$' represents the additive operator. Note that the '$\oplus$' does not necessarily correspond to ordinary addition.

- *Commutativity.* The result of two successive blurring steps should be equivalent if the two steps were reversed, *i.e.,* $b_1 \circ b_2$ equals $b_2 \circ b_1$.

We emphasize that translation and rotation invariance are implied by the assumptions of homogeneity and isotropy (see [29] for details).

For an extensive description of the above-mentioned front-end vision axioms with respect to the six different scale space generators (described in this chapter) we refer to appendix 6.B. Table 6.1 gives an overview. To which degree the six scale space generators satisfy each axiom is indicated by one of the three signs: '+' (good), '□' (reasonable), or '–' (bad).

Axiom	*spatial*	*Fourier*	*Perona & Malik*	*Euclidean short. flow*	*median*	*Kuwahara*
linearity	+	+	–	–	–	–
homogeneity	+	+	–	–	+	+
isotropy	+	+	+	–	□	–
self-similarity	+[1]	+	+[2]	+[2]	–	–
commutativity	+	+	+	+	–	–

Notes:

[1] Only if the original image is the starting point (see also Lindeberg [66]).

[2] Implemented this way (*i.e.,* iteratively).

Table 6.1. *Responses of the different scale space generators to the basic axioms of a front-end vision system.*

6.6.2 Desirable scale space properties

In this section we describe a set of additional properties that a scale space generator should have in order to be 'useful' in any sense. The problem is that it is hard to define the 'complete set' of desirable properties, since this will depend on the application of the scale space. Furthermore, we emphasize that different (and often very logical) desirable properties may be in contradiction with the axioms defined in the previous section. An example will be given below.

Desirable properties are:

Property	*spatial*	*Fourier*	*Perona & Malik*	*Euclidean short. flow*	*median*	*Kuwahara*
adaptive feature preservation	–	–	+	+[3]	□	□
grey value invariance	–	–	–	+	+	–
convergence	+/−[4]	+	–	□	–	+
low noise sensitivity	□	□	+	+	+	□
fast	+	+	□[5]	□[5]	□	□

Notes:

[3] Extensions of this scheme [95, 2, 101, 78] alter the speed of diffusion as a function of the local gradient magnitude.

[4] Depends on the solution to the boundary problem (for instance, + if the average image value is used, – for 0^{th} order extrapolation).

[5] Our implementation uses an explicit scheme, which is prohibitively slow for the higher scale levels.

Table 6.2. *Quality of the different scale space generators with respect to the additional desirable properties.*

- *Adaptive feature preservation.* The blurring should be flexible with respect to features (such as edges, corners, etc.). That is, for some applications it might be useful to preserve the features at that scale and blur only in more or less homogeneous regions. The amount of preservation should be adaptive.

- *Grey value invariance.* The blurring should be invariant under the group of general intensity transformations acting on the image L, given by:

$$\mathcal{T} : \mathbb{R} \to \mathbb{R} : L \mapsto \widehat{L} = \mathcal{T}(L) \quad , \tag{6.15}$$

  where $\widehat{L}$ denotes the image after the transformation $\mathcal{T}$. Note that $\mathcal{T}$ may be any nonlinear transformation, as long as it is a strict monotone function. (See [32] for details.)

- *Convergence.* The blurring should converge to the mathematical mean of the entire image. This should be the only pixel value left at the largest scale that can be computed.

- *Low noise sensitivity.* The influence of noise should be minimal.

- *Fast.* The actual algorithm should be fast *irrespective* of the scale that is calculated.

In appendix 6.B we discuss these items in detail. Note that the property of 'feature preservation' (in fact a nonlinearity preference) heavily conflicts with the axioms 'linearity' and 'isotropy'. Obviously, it depends on the specific task and the used algorithm which property should precede the other one.

Table 6.2 summarizes the results. The '+' , '□', and '–' signs now relate to the 'quality' of each scale space generator with respect to the different properties.

6.6.3 Nonlinear scale spaces and scale

In this section we describe how we relate the scales of the levels of a linear scale space to the levels generated by nonlinear scale space constructors.

The Perona & Malik equation and Euclidean shortening flow

We first consider the two approaches which are modifications of the linear diffusion equation (6.1). Since the Gaussian is the Green's function of this equation, it can be thought of as the aperture function, which generates the one-parameter family of images (known as linear or Gaussian scale space). Hence, there exists a relation between the evolution parameter t in equation (6.1) and the standard deviation of the Gaussian (or scale) σ. This relation is given by:

$$t = \frac{1}{2}\sigma^2 \ . \tag{6.16}$$

For the nonlinear evolution equations discussed in this chapter no analytical solutions are known. This has two major implications. First, we have to resort to numerical approximations to be able to compute the multi-scale representation of an image. Secondly, there is no longer an obvious relation between the evolution parameter and the spatial extent of 'some' blurring filter. However, both the Perona & Malik equation and the Euclidean shortening flow are derived from the linear diffusion equation in the sense that they are limiting the 'normal' linear diffusion. In the former case the linear diffusion is limited in regions with a high gradient while the latter case only allows diffusion in the local isophote direction. Therefore it makes sense to relate the evolution parameter t of the nonlinear evolution equations to the evolution parameter t in the case of the linear diffusion equation. Instead of using an exponential scale sampling as in equation (6.2), we use an exponential 'evolution time sampling t' which corresponds to the scale sampling of the linear diffusion equation, according to equation (6.16).

The median and the Kuwahara filter

In the case of both the median and the Kuwahara filter, the size of the sliding window determines the 'scale' of the resulting level. In particular, if the window size m of the median filter has become large enough so that the entire image of size $N \times N$ will have one and the same value (the global median of the image), we say that the scale of that level equals $\sigma = N$, since at that scale the linear Gaussian kernel will generally result in a (near-)homogeneous image. This will definitely be the case if $m = 2N$, since then the window covers the entire image, irrespective of which pixel is being processed.

Similarly, for the Kuwahara filter, if $w = N$ (or $k \approx 2N$), a homogeneous image is to be expected, which corresponds to $\sigma = N$. (Unlike the median, however, the global image value that corresponds to the largest scale is the mathematical average of the entire original image.)

For the successive values of k_n—corresponding to the Kuwahara kernel size k to create level n from the input image—we follow a similar definition as for the successive values m_n of the median filter (see below).

In accordance with equation (6.2)—defining the exponential character of the linear scale space sampling—successive m-values (corresponding to the scale levels) must have an exponential base too. This is achieved by following the formula:

$$m_n \;=\; \begin{cases} 3 & \text{if } n = 1 \\ \lfloor \psi^j \rfloor & \text{if } n > 1 \end{cases} \qquad j, n \in \mathbb{Z}^+ \;, \tag{6.17}$$

where the base factor ψ regulates the scale space sampling ($\psi \in \mathbb{R}^+$), and j equals the smallest value not smaller than n such that $m_n > m_{n-1}$. This is a simple mechanism to prevent the occurrence of 'double' m_n values; we simply skip these levels. (Note that for $\psi = 1$ no double values will be produced by equation (6.17), *i.e.,* $j = n$, $\forall n \in \mathbb{Z}^+$.) In order to estimate a proper value for ψ, we make the observation that $\psi \approx e^{\delta\tau}$ for a linear scale space with reasonable values for ε and τ_0. Typically, $\varepsilon = 1$, $\tau_0 = 0$, and $\delta\tau = \frac{1}{2}\ln 2$. Hence, $\psi = \sqrt{2}$ seems an appropriate choice for comparing a scale space generator—like the median filter—with the corresponding linear scale space.

A qualitative comparison with the linear Gaussian scale space can be made by comparing the number of 'elliptic patches' at each scale. This requires a second order invariant, called the *umbilicity* of an image. Florack [29] argued that the number of 'generic local features' should decrease exponentially if scale is increased, while the slope of the corresponding log-linear graph is dimension dependent. This linear relationship will only be valid on a scale range fairly within the fundamental upper an lower limits, *i.e.,* the inner and outer scale.

For the two-dimensional case, an elliptic patch is defined as a connected set of pixels in which:

$$\frac{\partial^2 L}{\partial x^2} \cdot \frac{\partial^2 L}{\partial y^2} \;-\; \frac{\partial^2 L}{\partial x\, \partial y} \;>\; 0 \;\;, \tag{6.18}$$

evaluated at the appropriate scale (see Fig. 6.4). Hence, the decrease in the number of elliptic patches corresponding to any scale space generator, should be more or less the same as the decrease for the linear scale space. In this way, we have defined a qualitative measure to synchronize different scale space generators.

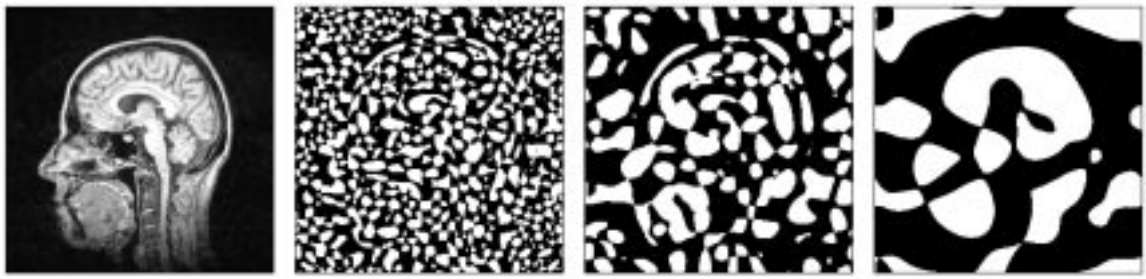


Fig. 6.4. *Elliptic patches of a* 256×256 *MR image at different scale levels. Shown are (from left to right) the original image and the umbilicity images of the levels 3, 5, and 7 (corresponding to* $\sigma = 2.83, 5.66, 11.31$ *pixels, respectively, with* $\delta\tau = \frac{1}{2}\ln 2$*).*

In practice, counting the blob-like structures for a linear scale space is performed by calculating the umbilicity at the corresponding scale. Then we continue scanning the resulting image until all the areas satisfying equation (6.18) have been counted. This is done effectively by a region-growing in each positive valued pixel encountered, after which that region is marked as being 'counted'. The process stops if no more positive valued pixels can be found.

As can be seen in Fig. 6.4, most of the elliptic patches appear to be connected to neighboring patches by a thin path of pixels, or via the 'corners' of two patches. To prevent any coincidental matching of the number of elliptic patches for two different scale space generators, we also compared the decrease in the number of patches after a single erosion step by a morphological filter with a small structuring element (a squared window of size 3×3).

As can be seen from Fig. 6.5, taking $\psi = e^{\delta\tau}$ nicely matches the different scale spaces with respect to the course of the umbilicity feature evaluated at the appropriate scales. The optional erosion step leads to the same conclusion, which emphasizes the robustness of the method of counting elliptic patches. Hence, if $\delta\tau = \frac{1}{2}\ln 2$, we have $\psi = \sqrt{2}$. This is valid for both the median and the Kuwahara filter.

6.7 Results

In this section we describe the results of applying the different scale space generators. This is done in three ways:

- The generated scale spaces are compared (see section 6.7.1).

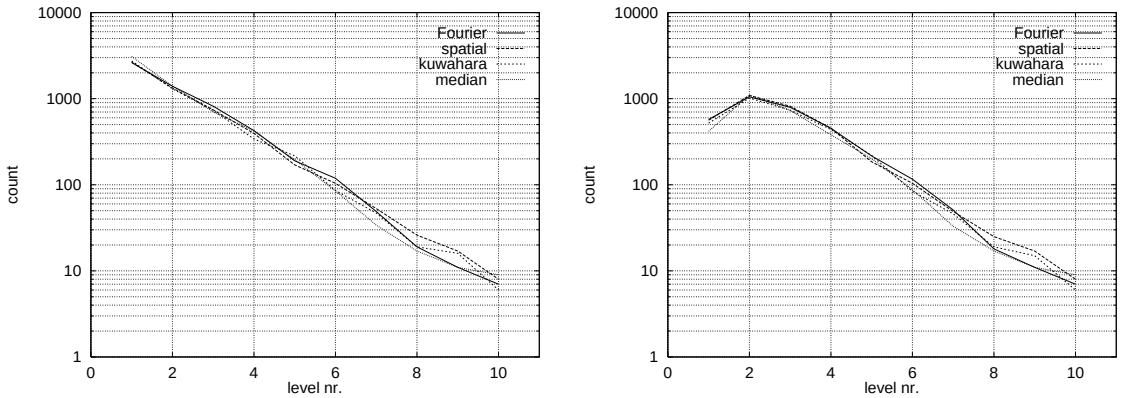


Fig. 6.5. *Exponential decrease of the number of elliptic patches in the first 10 levels for different scale spaces (left). For the linear scale space (both in the Fourier and the spatial domain) we have used* $\delta\tau = \frac{1}{2}\ln 2$. *For the median and the Kuwahara filter we have used* $\psi = \sqrt{2}$. *As expected, this value of* ψ *gives a good qualitative match with the conventional linear scale space. On the right, the elliptic patches have been counted after an additional erosion step (see text).*

- The stacks are used as input to the hyperstack segmentation algorithm (see section 6.7.2). The produced segmentations are qualitatively compared (see section 6.7.3).

- The segmented images are evaluated by an objective measure (see section 6.7.4) that we have developed for quantitative analysis and comparison of segmentation results.

For the experiments we have used four different images. Firstly, we created an artificial 3D image containing a hand by using the volumetric object generating package 'THINGS' [117]. The main feature of this package is the simulation of the partial volume artifact by discretizing the specified objects (*i.e.,* ellipsoids, etc.) at sub-voxel level. The original HAND image contains the pixel values 0 (background), 500 (dumb), 800 (forefinger), 1000 (palm of the hand), 1250 (middle finger and ring-finger), and 1500 (little finger). From this artificial image (containing 16 slices of 64×64 pixels) we derived two different input images by adding two different Gaussian noise levels: one with standard deviation 100 (called the 'HAND.100' image) and one with 200 called the 'HAND.200' image). Furthermore, we have used two real-world 2D MR slices of a human head, which are both originally sized 256×256. The first image (called 'BRAIN.COR') has a coronal viewpoint, a ROI of size 178×179 and contains two objects. The second image (called 'BRAIN.TR') is transversally oriented and has a ROI of size 170×210, in which six objects can be distinguished.

For all images we can check the segmentations against a 'gold standard'. That is, for the HAND.100 and the HAND.200 image (see Fig. 6.6b and Fig. 6.6c, respec-

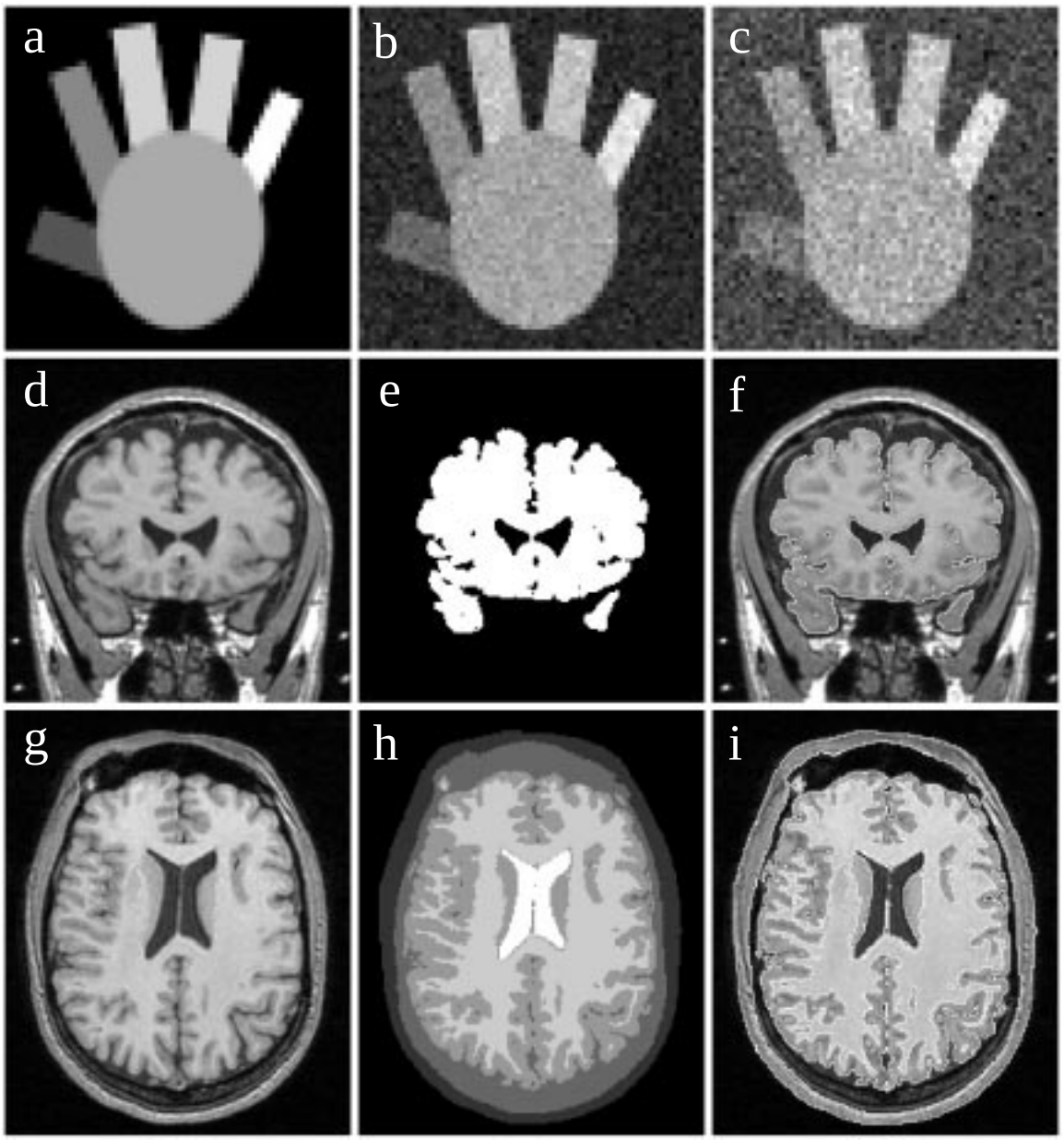


Fig. 6.6. *Input images to the experiments.*
(a) a slice of the original HAND *image (the object distribution);*
(b) the corresponding slice of the noisy HAND.100 *image (s.d. = 100);*
(c) idem, for the HAND.200 *image (s.d. = 200);*
(d) the original BRAIN.COR *image;*
(e) the manual segmentation of the ROI;
(f) the segmentation of (e) superimposed on the original image (d);
(g) the original BRAIN.TR *image;*
(h) the manual segmentation of the ROI;
(i) the segmentation of (h) superimposed on the original image (g).

tively) we can use the original image created by THINGS before noise was added (see Fig. 6.6a). This reflects the 'perfect' object distribution that a segmentation algorithm should produce. For the BRAIN.COR and the BRAIN.TR images we use the manual segmentations of an expert. Within specific accuracy limits these can be considered the gold standards, although other experts would probably define different object contours as 'the best' segmentation possible. This can be seen by examining the manual segmentations superimposed onto the original images (see Fig. 6.6).

6.7.1 The scale spaces

In Fig. 6.7 the different scale spaces—generated by the methods discussed in this chapter—for the HAND image are shown. More precisely, we extracted one slice of the HAND.200 image at some of the levels of the generated scale spaces to show the scaling function. Note that the detail vanishes at different scales, so for clarity we have chosen different series of scales for every scale space generator. For instance, the median and the Kuwahara filters are only interesting at the very first levels of the scale space. The actual level numbers used have been indicated in the subscript of Fig. 6.7.

Note that the scale spaces that correspond to the HAND.100 image have been omitted here; in fact, these are very much like the HAND.200 images shown here. As regards Fig. 6.7 we have the following remarks:

- The boundary effect caused by blurring in the Fourier domain (*i.e.,* data repetition) is much more apparent than in the spatial domain (averaging).

- The Perona & Malik equation preserves the objects' contours best. The thumb disappears first of all fingers, because it has the lowest intensity value. A different (lower) value of K in the equations (6.5) or (6.6) will preserve the thumb better.

- Structure disappears relatively fast at the three-dimensional median and the Kuwahara filter. Nonetheless, at the first levels of the scale space features are better preserved than in the linear case.

- For the Euclidean shortening flow it is striking that the absolute image intensities do not affect the local blurring: the curve evolves at the same speed for points with equal curvature.

In Fig. 6.8 a similar figure is shown for the BRAIN.COR image. The most striking differences are comparable to the HAND.200 image of Fig. 6.7: the nonlinear scale space generators preserve features better than in the linear case. Furthermore, we notice that the squared structure of the Kuwahara filter (at every scale) becomes very apparent at larger scales. Corners seem to be 'introduced' rather than just 'preserved'.

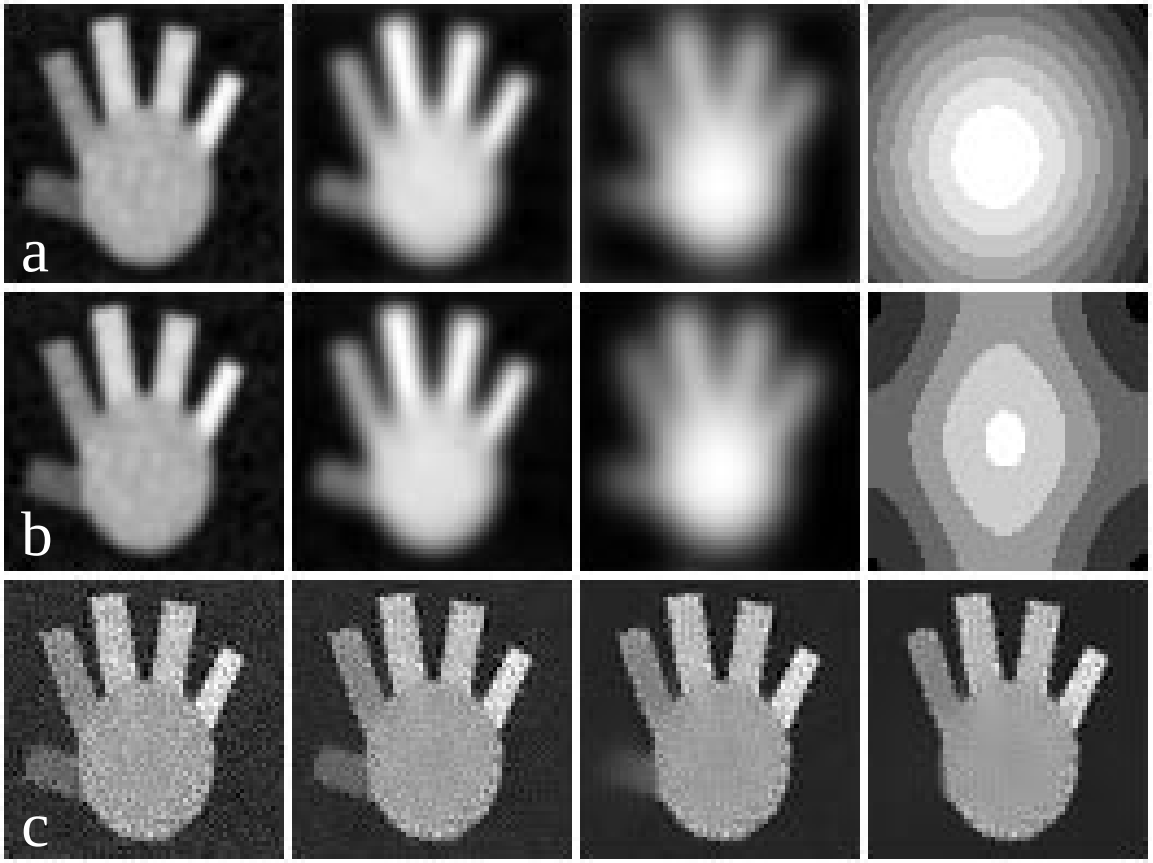


Fig. 6.7. *A single slice of the* HAND.200 *image shown at different levels of scale for the six different scale space generators. Shown are, from top to bottom:*
(a) Linear scale space, spatial domain (levels 1, 3, 5, 11);
(b) Linear scale space, Fourier domain (levels 1, 3, 5, 11);
(c) Perona & Malik filter (levels 2, 4, 6, 8);

In Fig. 6.9 we show the different scale spaces of the BRAIN.TR image. Owing to an evenly distribution of grey values in this (more or less symmetric) image, the Perona & Malik equation and Euclidean shortening flow are much alike. The ventricle system having a considerably lower intensity than its neighboring objects causes variable conductance diffusion to preserve the ventricles at large scales. In contrast, Euclidean shortening flow considers the ventricles to be a relatively small object, hence it disappears before level 10.

6.7.2 The hyperstack

In essence, the hyperstack is a linking model based segmentation technique, originally built upon the linear scale space theory. The basic idea of the hyperstack is to define

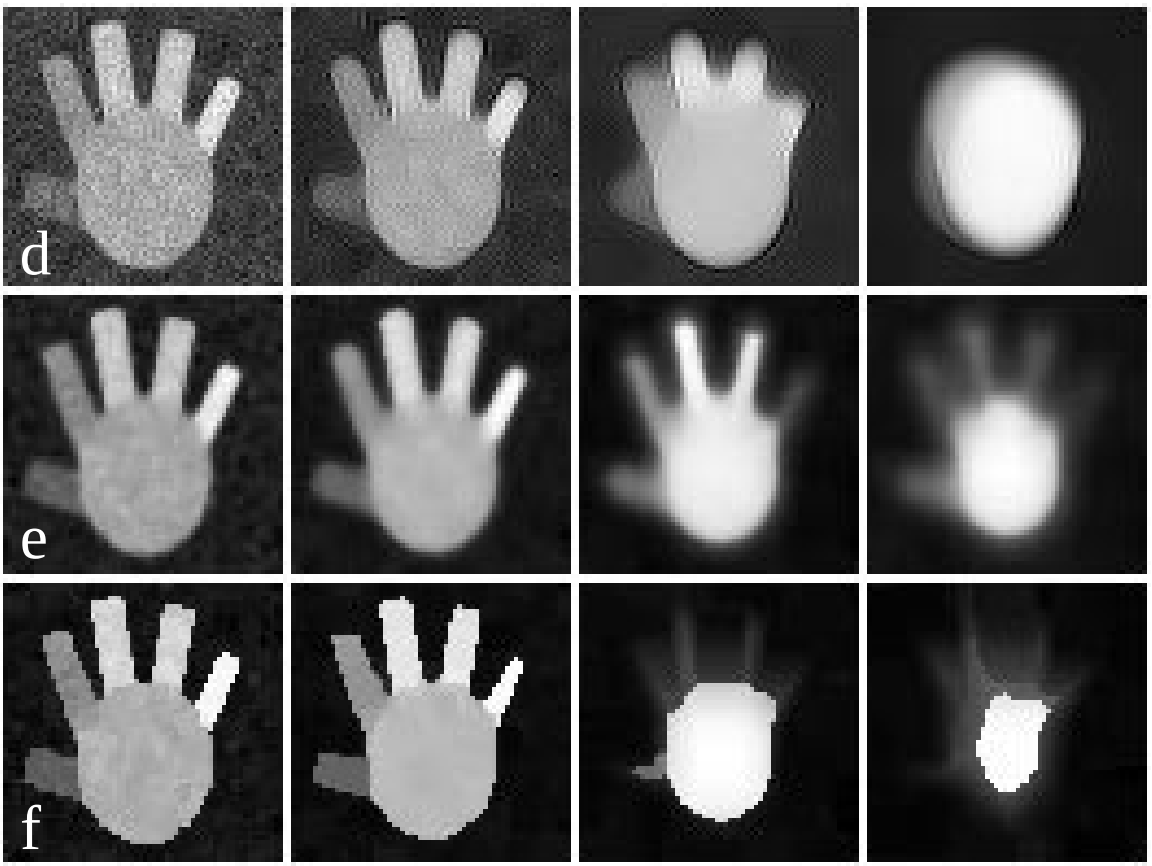


Fig. 6.7 (continued).
(d) Euclidean shortening flow (levels 2, 4, 6, 8);
(e) Median filter (levels 1, 2, 3, 4);
(f) Kuwahara filter (levels 1, 2, 3, 4).

relations between voxels in all pairs of adjacent scale space levels, such that the levels at larger scales—containing the global information—guide the collection of voxels in the original image at the smallest scale (the ground level).

The entire process requires four steps: (*i*) blurring, (*ii*) linking, (*iii*) root labeling, and (*iv*) downward projection.

In the blurring phase the scale space is created. This results in a stack of images at increasing scale. During the linking phase voxels in two adjacent scale levels are connected by so-called child-parent linkages. This is done by searching for suitable parents in level $n+1$ for each child in level n. Each link is awarded a value (or *linkage strength*), based on statistical criteria and heuristic features. The maximum spatial distance that a child-parent link may span (*i.e.,* the search volume of a child) depends on the relative scale step $\sigma_{n,n+1}$ from level n to level $n+1$. The linking is a bottom-up process, such that only parents that have been linked to are considered children in the next linking step. This leads to a tree-like structure of linkages through scale space,

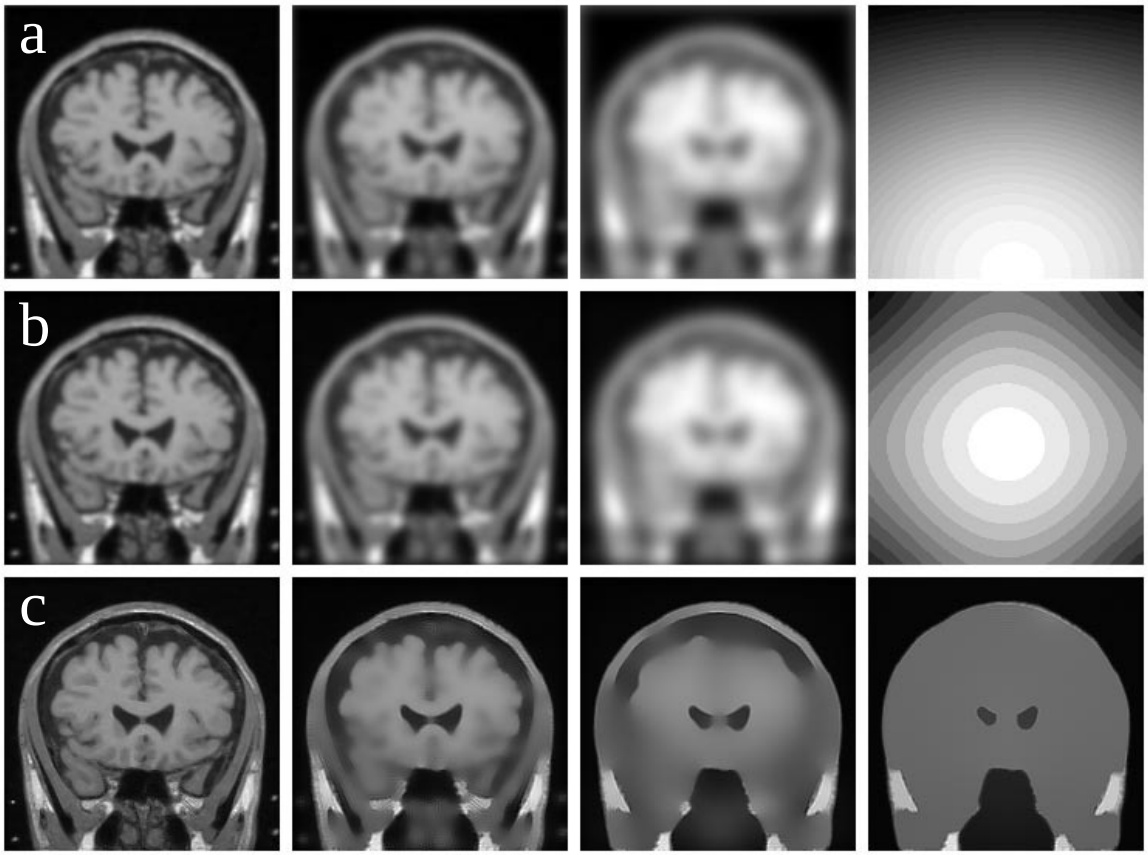


Fig. 6.8. *The* BRAIN.COR *image shown at different levels of scale for the six different scale space generators. Shown are, from top to bottom:*
(a) Linear scale space, spatial domain (levels 2, 4, 6, 15);
(b) Linear scale space, Fourier domain (levels 2, 4, 6, 15);
(c) Perona & Malik filter (levels 2, 5, 8, 11);

called *the hyperstack*.

If the linking has converged—in the sense that only few parents are left in the top level of the hyperstack—the root labeling takes place. Then, the children in the tree with a relatively low child-parent linkage strength are labeled as *roots*, which can be regarded as an 'end point' in the linking of voxels that belong together (*i.e.,* that form a segment). Finally, in the down projection phase the actual segments are formed by a projection of specific segment values from the roots downwards to the ground level. This can readily be executed by following the child-parent linkages backwards (from the parents to their children). Instead of using unique segment values to discriminate between the different segments it may also be useful to apply a 'mean value' segmentation. Then, the mean value of all the pixels belonging to one segment is used for the downward projection of that segment from the corresponding root. This gives usually more attractive segmented images for visual inspection (see

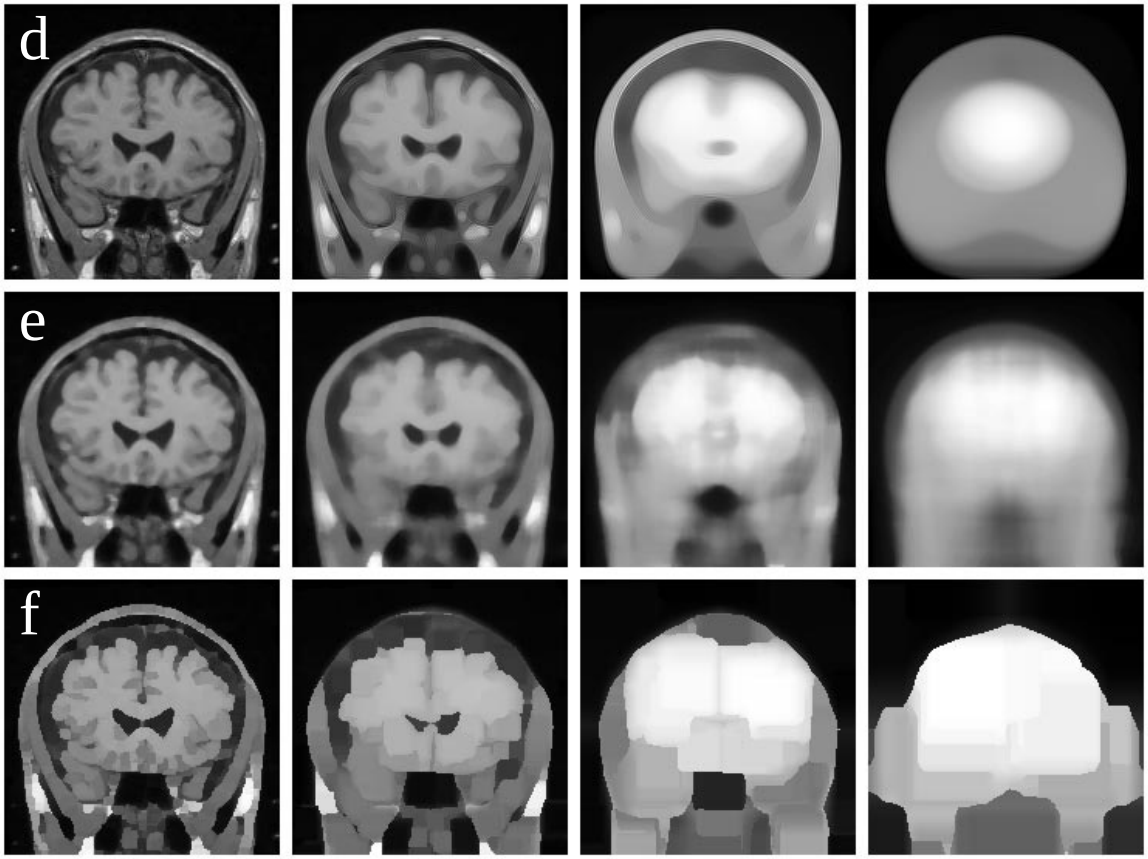


Fig. 6.8 (continued).
(d) Euclidean shortening flow (levels 2, 5, 8, 11);
(e) Median filter (levels 2, 4, 6, 8);
(f) Kuwahara filter (levels 2, 4, 6, 8).

the results for examples of mean value segmentations).

In short, this describes the hyperstack segmentation algorithm. For a more detailed description and various extensions to this basic scheme we refer to different publications on this subject [58, 121, 123, 124, 120, 119, 127, 126], or see section 2.2 of this thesis.

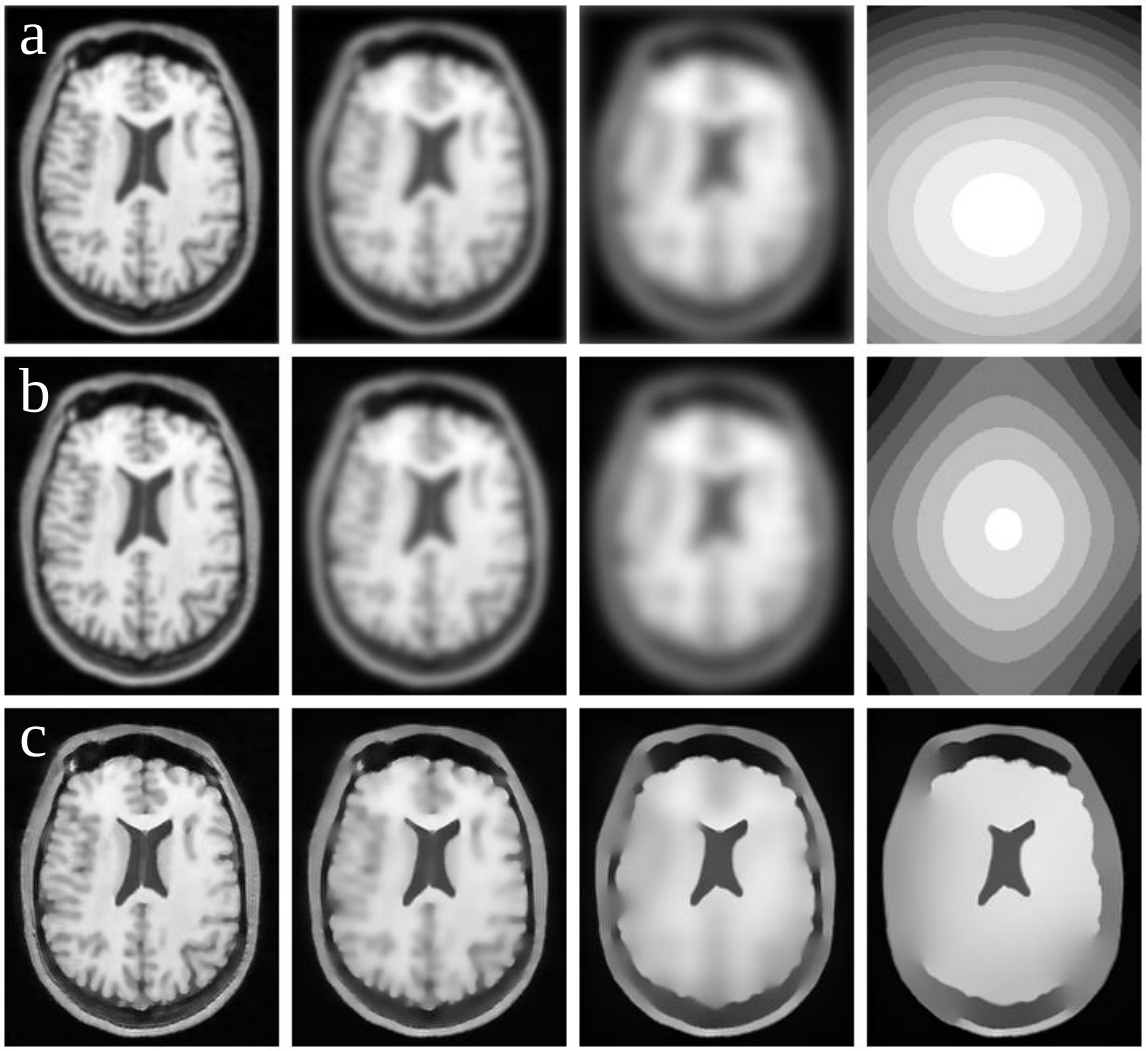


Fig. 6.9. *The* BRAIN.TR *image shown at different levels of scale for the six different scale space generators. Shown are, from top to bottom:*
(a) Linear scale space, spatial domain (levels 2, 4, 6, 15);
(b) Linear scale space, Fourier domain (levels 2, 4, 6, 15);
(c) Perona & Malik filter (levels 2, 4, 7, 10);

6.7.3 The segmentations

In this section we compare the different scale space generators according to the segmentations created by the hyperstack method. Each hyperstack is based on the corresponding scale spaces shown in section 6.7.1, depending on the input image at hand.

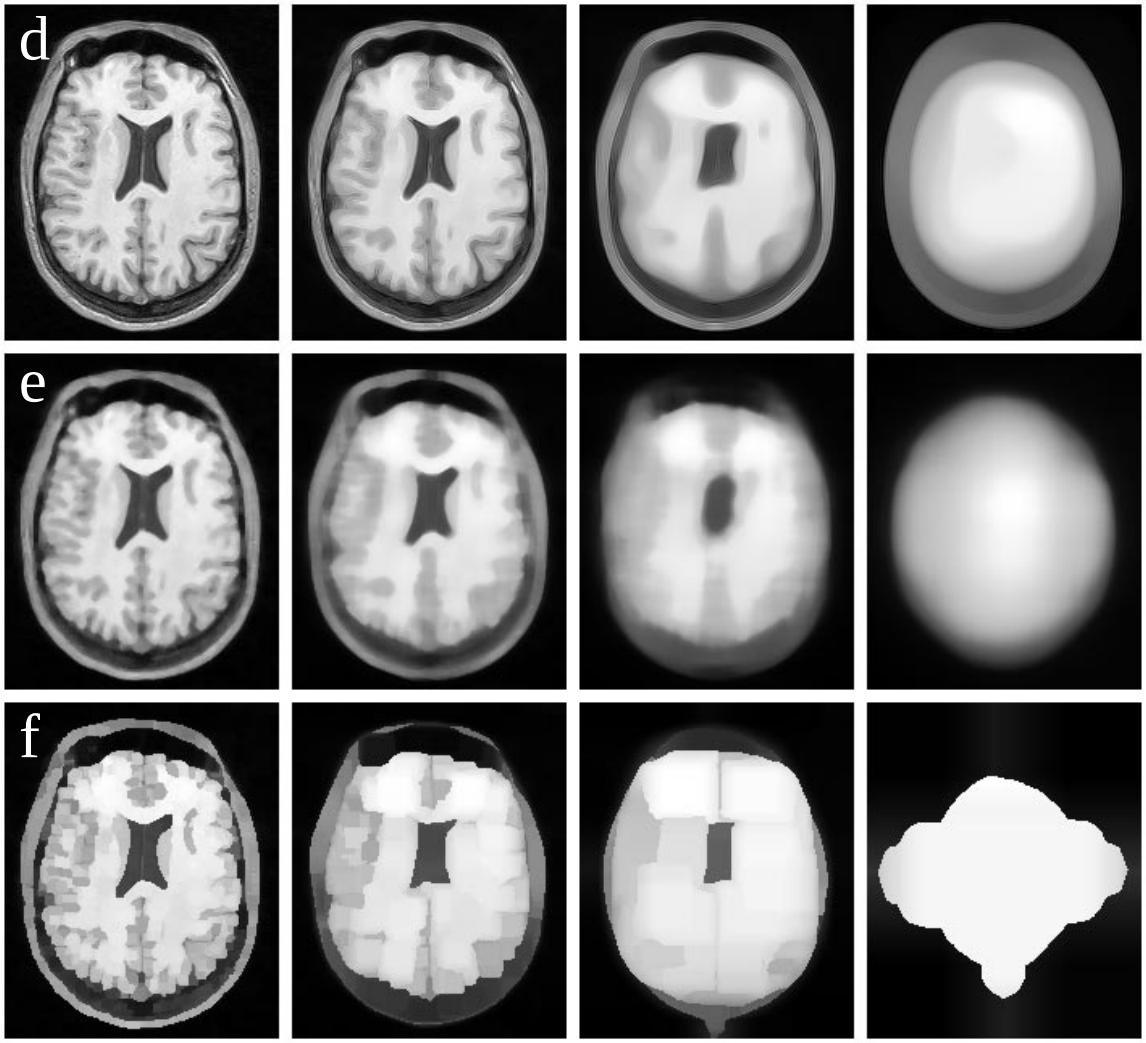


Fig. 6.9 (continued).
(d) Euclidean shortening flow (levels 2, 4, 7, 10);
(e) Median filter (levels 2, 4, 6, 10);
(f) Kuwahara filter (levels 2, 4, 6, 10).

Convergence of the linkages in a hyperstack is enforced by using a sufficient number of discrete scale space levels, so that only a few top parents remain in the highest level of the stack. (A hyperstack typically needs 15 levels to converge.) Note that this also requires that all the detail disappears at increasing scale; otherwise, groups of linkages may continue linking upwards without linking to each other. For instance, the stacks generated by the Perona & Malik equation suffer from this problem, while the median and the Kuwahara stacks will converge relatively fast in terms of diffusion time. See section 6.B.8 for details and solutions.

In Fig. 6.10 five pairs of volume renderings of HAND images are shown. Each pair—corresponding to the front and back side of the hand—relates to a segmentation of the hyperstack based on one of the six scale space generators. In order to enable a visual comparison, the front and back view renderings of the original image are presented as well.

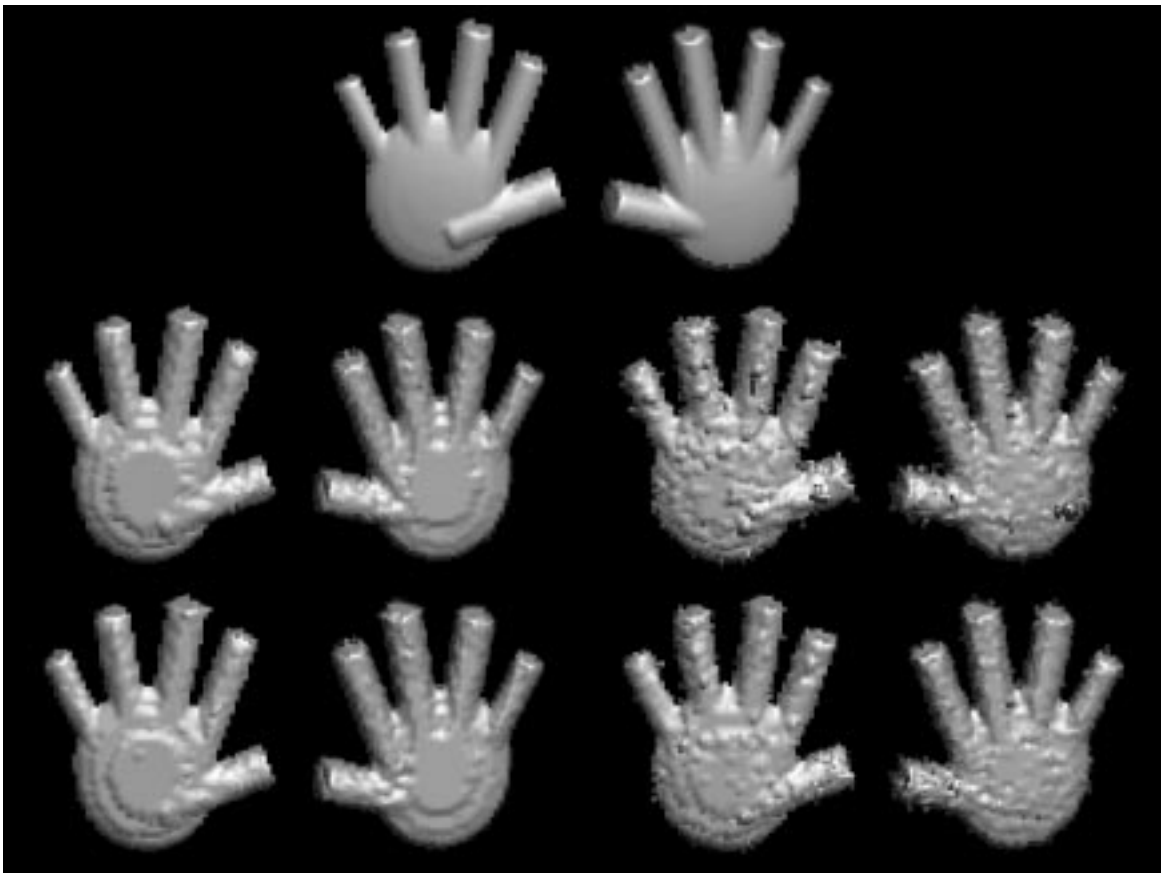


Fig. 6.10. *Volume renderings of segmentations of the* HAND.100 *and the* HAND.200 *image. The results have been rendered twice to show the front and back sides of the hand. The pairs of hands correspond to the original image (top row), the* HAND.100 *image (left pairs), and the* HAND.200 *image (right pairs). The middle row contains the 'worst' segmentations of both the* HAND.100 *and the* HAND.200 *image according to the objective evaluation measure (see section 6.7.4); the bottom row contains the 'best' segmentations according to the same criterion.*

For both the HAND.100 and the HAND.200 image the worst and best results are shown. This is based on a criterion that will be explained in section 6.7.4, where a quantitative evaluation method for segmentation results is outlined. In any case, comparing the middle row (worst results) with the bottom row (best results), it is hard to detect clear differences. This is a first indication for the fact that the hyperstack

segmentations do not depend greatly on the different scale spaces used. (Note that the low resolution of the HAND images—$64 \times 64 \times 16$—leads to relatively poor renderings, where single pixels look 'blown up'.)

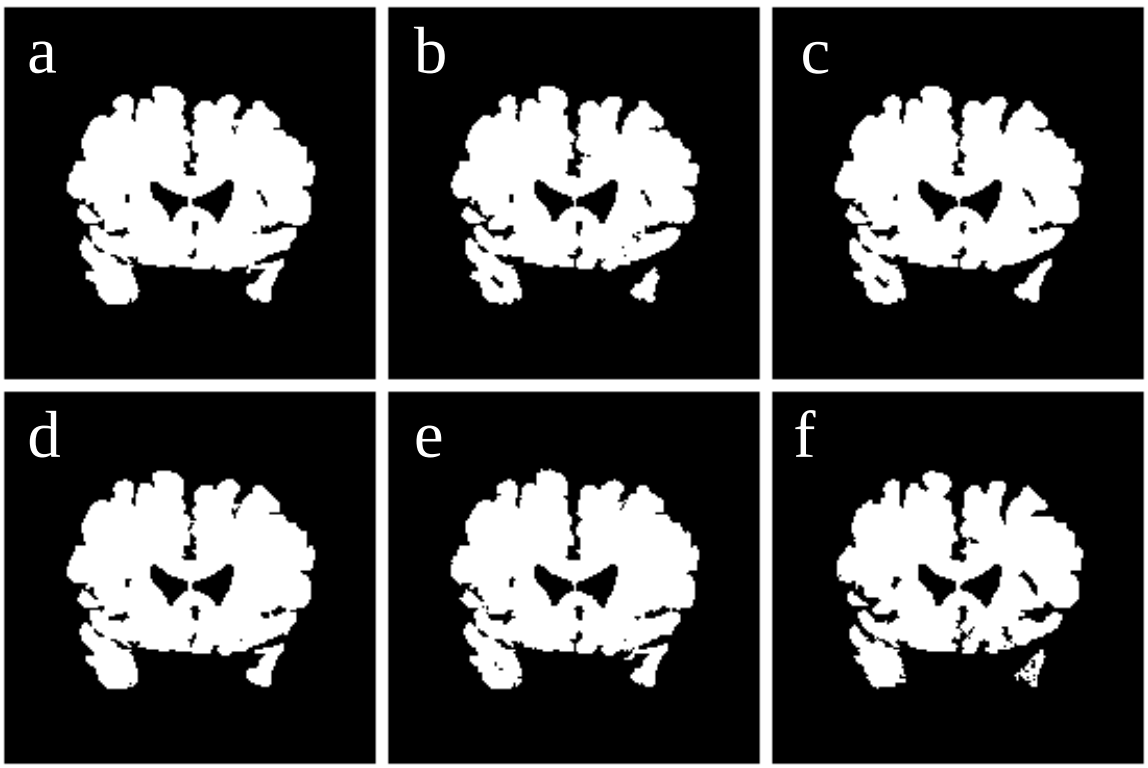


Fig. 6.11. *Segmentations of the* BRAIN.COR *image based on the six different scale space generators. The segmentations correspond to:*
(a) Linear scale space implemented in the spatial domain;
(b) Linear scale space implemented in the Fourier domain;
(c) Perona & Malik equation;
(d) Euclidean shortening flow;
(e) Median filter;
(f) Kuwahara filter.

In Fig. 6.11 six segmentations of the BRAIN.COR image are presented. The results are shown in binary form, so as to make them comparable to the manual segmentation of Fig. 6.6e. Again, the differences are small and mainly concern the small, disconnected lobe on the right part of the picture. Indeed, small segments and deep inlets are the most difficult parts to segment.

To emphasize the differences we use so-called *error* pictures (see Fig. 6.12). These are constructed by subtracting each (binary) segmentation of Fig. 6.11 from the manual segmentation (Fig. 6.6e). The pixels that are in agreement with each other are colored grey, the differences are colored white. Note that the color white may denote

either 'too little' or 'too much' segment.

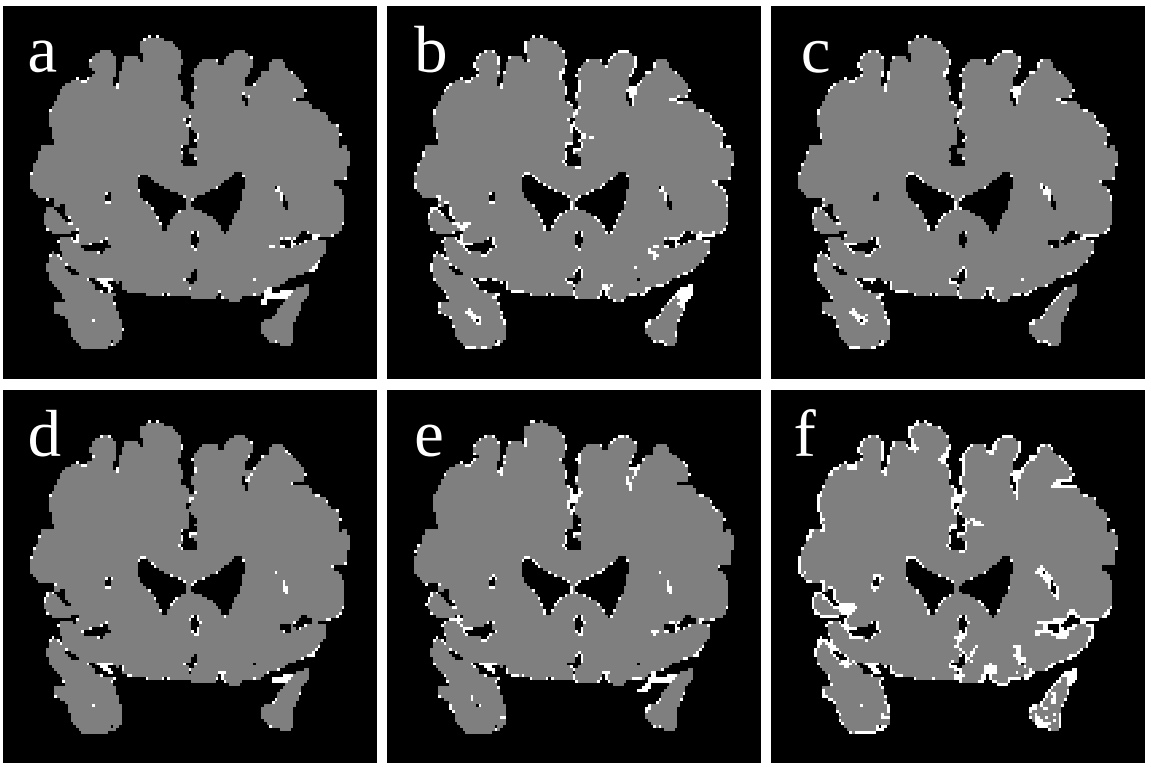


Fig. 6.12. *Errors of the segmentations of the* BRAIN.COR *image based on the six different scale space generators. The grey colored areas have been segmented correctly (according to the gold standard), the white colored areas correspond to erroneously segmented pixels. The figures correspond to:*
(a) Linear scale space implemented in the spatial domain;
(b) Linear scale space implemented in the Fourier domain;
(c) Perona & Malik equation;
(d) Euclidean shortening flow;
(e) Median filter;
(f) Kuwahara filter.

It is clear that the Kuwahara filter performs worse than the other five. However, we have to be careful with interpreting these results. A manual segmentation will never be 100% correct, and in addition the segmentation task will generally not require a 100% correct result. Therefore, not all the white pixels will have to be corrected to obtain a useful result. See section 6.7.4 for a more detailed consideration on evaluating segmentation results.

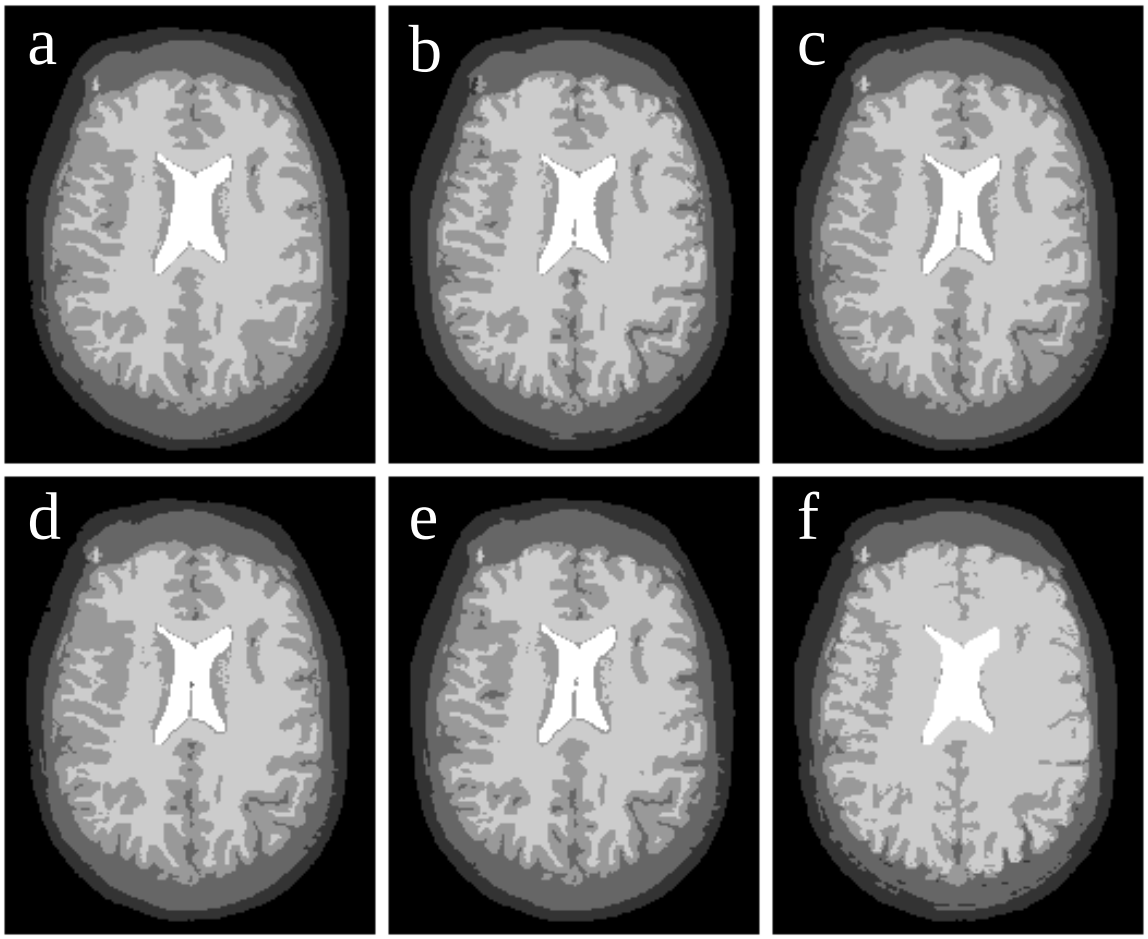


Fig. 6.13. *Segmentations of the* BRAIN.TR *image based on the six different scale space generators. The segmentations correspond to:*
(a) Linear scale space implemented in the spatial domain (initially 58 segments);
(b) Linear scale space implemented in the Fourier domain (38 segments);
(c) Perona & Malik equation (16 segments);
(d) Euclidean shortening flow(34 segments);
(e) Median filter (50 segments);
(f) Kuwahara filter (54 segments).

In Fig. 6.13 the object distribution obtained from the six hyperstack segmentations of the BRAIN.TR image is shown. Each segmentation produces a series of sub-segments that have to be collected to end up with a distribution similar to Fig. 6.6h. The number of sub-segments that resulted from the hyperstack segmentation step is indicated below the figures in Fig. 6.13.

Finally, in Fig. 6.14 error images corresponding to Fig. 6.13 are shown. We have focused on the white matter of the brain, because this is the segment with the largest

and most difficult curving elements. Again the Kuwahara filter performs relatively bad, characterized by the large white parts at the edges of the contours. The nonlinear methods based on the Perona & Malik equation and the Euclidean shortening flow perform well.

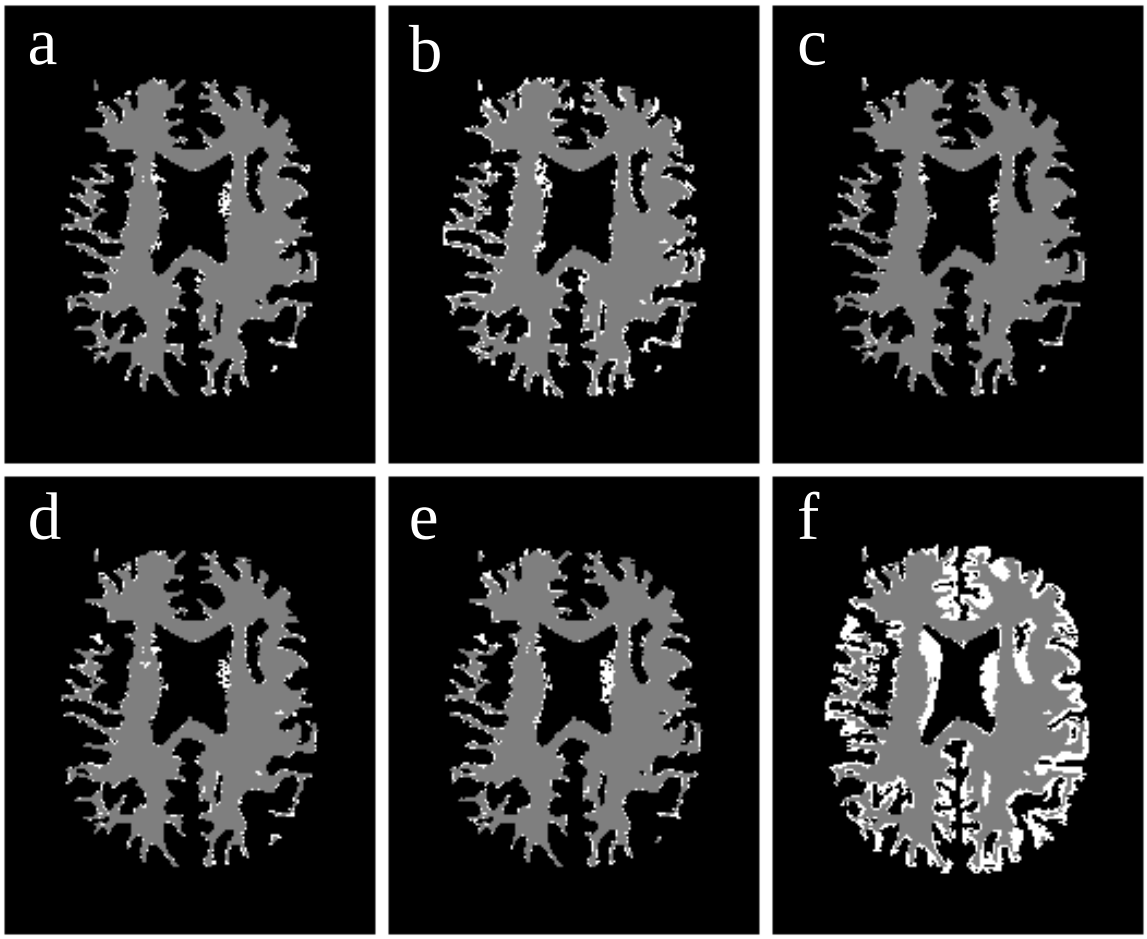


Fig. 6.14. *Errors of the segmentations of the white matter of the* BRAIN.TR *image based on the six different scale space generators. The grey colored areas have been segmented correctly (according to the gold standard), the white colored areas correspond to erroneously segmented pixels. The figures correspond to:*
(a) Linear scale space implemented in the spatial domain;
(b) Linear scale space implemented in the Fourier domain;
(c) Perona & Malik equation;
(d) Euclidean shortening flow;
(e) Median filter;
(f) Kuwahara filter.

6.7.4 Evaluation

In order to be able to make an objective and quantitative comparison between the performance of the different scale space generators with respect to the hyperstack segmentations, we have developed a task-driven evaluation method [59, 136, 22]. The task is defined as to minimize the effort of manually editing the segmented image—*post-processing editing* (PPE)—until a result of satisfactory quality is obtained.

Manual editing is modeled by an editing scenario that consists of a series of two basic actions: (*i*) the merging of two segments into one segment, and (*ii*) splitting a segment along a borderline (2D) or surface (3D). These actions are labeled with costs. A merge counts for 2 cost units, and a split for $1 + 0.5 \cdot \mathcal{C}$, where $\mathcal{C}$ denotes the number of contour pixels/voxels involved in the split operation. All the editing of a segmented image can be expressed in these two basic actions, resulting in a certain amount of 'editing costs'.

The quality of a segmented image is determined by mapping each segment of a segmentation to the gold standard (or rather *object distribution* in this respect) in a one-to-one relation. The *correctness* of a segmentation is defined by the number of voxels that could be mapped to objects relative to the total number of voxels in the image. This results in correctness values ranging from 0 to 1, in which '1' corresponds to a segmentation that is completely identical to the object distribution.

Segmenting an image correctly up to the very last pixel may involve excessive editing costs. In most cases, this is not needed. (Note further that the use of a manual segmentation as gold standard also introduces errors, because no two manual segmentation are the same.) Hence, the required correctness of a segmentation is not 1, but a value slightly less than 1. In our experiments we allow a fifth of the number of object border pixels to be segmented incorrectly. In this way, the complexity of objects in the image is acknowledged by allowing a tolerance for small segmentation errors.

Some of the values used in the editing scenario—*viz.* the basic costs for a split action, the 'one fifth' of the border pixels, etc.—may look rather *ad hoc*. However, the final costs results should not be interpreted as absolute judgements with respect to the quality of the segmentations, but more as a relative measure for *comparing* different segmentations for one and the same input image (with a corresponding object distribution). Indeed, we have experimentally found out that tuning the editing parameters only changes the absolute costs. Mutual relations ('method A is cheaper than method B') generally remain valid.

The PPE costs do not have an absolute meaning for the same reason. Rather, they should be compared *relatively* to the costs needed for a complete manual segmentation. The manual costs are calculated by using an entirely homogeneous image as 'segmented image'. Then, the editing costs to change the image to the gold standard correspond precisely to drawing the contours of all the segments. The relative PPE costs refer to the ratio of the costs of an automatic segmentation and the costs of a 100% manual segmentation.

Table 6.3 gives an overview of the PPE costs for the four images and for the six scale space generators considered. The nonlinear diffusion paradigms (Perona & Malik, Euclidean shortening flow) perform best.

Image		*spatial*	*Fourier*	*Perona & Malik*	*Euclidean short. flow*	*median*	*Kuwahara*
HAND.100	#segs	26	22	34	12	51	36
	costs	1.4%	1.7%	1.5%	1.2%	2.8%	1.7%
HAND.200	#segs	26	24	26	26	54	32
	costs	1.2%	1.6%	0.9%	1.7%	3.9%	1.5%
BRAIN.COR	#segs	24	16	34	18	24	46
	costs	8.8%	16.0%	5.6%	7.1%	8.6%	20.6%
BRAIN.TR	#segs	58	38	16	34	50	54
	costs	7.9%	5.1%	2.3%	3.8%	7.5%	31.7%

Table 6.3. *Relative PPE costs plus corresponding number of segments that result from a hyperstack segmentation for six different scale space generators.*

6.8 Conclusions

In this chapter we have investigated the sensitivity of hyperstack segmentation to the strategy of scale space blurring. We have compared six different scale space generators, and judged the corresponding segmentations both quantitatively and qualitatively. Based on the experiments described with artificial and medical images, we draw the following conclusions:

- In general, hyperstack segmentation is not very sensitive to the underlying scale space. Apparently, using a linking model this way (*i.e.,* based on multiple features) is a robust system that nicely 'follows' the distortion of the scale space caused by a nonlinear generator.
- The nonlinear scale space generators based on the Perona & Malik equation and the Euclidean shortening flow perform best on the test images.
- For the Perona & Malik equation more research has to be done to find the criteria that determine at which scale linear blurring should take over the nonlinear

blurring. Our preliminary conclusion is that using nonlinearly blurred levels prior to a linear scale space will yield an improvement in all cases. A large diversity in sizes of objects that need to be segmented requires a longer nonlinear blurring in terms of diffusion time.

- The Euclidean shortening flow requires a proper 3D implementation for a principled comparison with the other (truly 3D) algorithms. However, the presented slice-by-slice approach already performs remarkably well, so the results may be expected to improve only slightly.

- The explicit implementations of both the Perona & Malik equation and the Euclidean shortening flow are computationally slow. An implicit implementation—which is currently being implemented—will speed up the calculation of all levels considerably. Note, however, that if only a couple of nonlinearly blurred levels are needed for the stack the speed issue can easily be dropped: the first levels can be created sufficiently fast for most purposes, even by explicit implementations.

- The Kuwahara filter has a low quality performance for real-world images, whereas it performs satisfactory for the artificial images. This can be explained by the fact that the HAND images are geometrically simple. Then, the Kuwahara filter nicely preserves some image features. For medical images, though, the larger scale levels have a lot of spurious corners, introduced by the square form of the filter. To a lesser extent, the same counts for the median filter. (But we emphasize that these two filters were never designed to operate at such extremely large scales.)

6.A Implementation details

In this appendix we discuss the implementation of the following scale space generators: the discrete Gaussian kernel in the spatial domain (section 6.A.1), the Perona & Malik equation and Euclidean shortening flow (section 6.A.2), the median filter (section 6.A.3), and the Kuwahara filter (section 6.A.4).

6.A.1 The discrete Gaussian convolution

In this appendix we will deal with the implementation of the convolution of a d-dimensional image—containing N^d pixels—with a Gaussian kernel of size k^d in the spatial domain ($k > 2$ and odd). The symbols '$\mathcal{M}$' and '$\mathcal{A}$' are used to denote the total number of multiplications and additions needed for a complete convolution of the image, respectively.

As explained in section 6.2.1, the best option to generate level n is to perform the blurring by convolution of the *original* image with a discretized Gaussian kernel of width σ_n.

A straightforward implementation of the discrete convolution in the spatial domain (*i.e.,* use the kernel as a sliding window to compute the convolution of every pixel in the image) requires $\mathcal{M} = k^d N^d$ multiplications and approximately the same amount of additions: $\mathcal{A} = (k^d - 1)N^d$.

In order to speed up the algorithm, we make use of two observations:

- the Gaussian kernel is separable in the Euclidean directions;
- the Gaussian kernel is symmetric.

The first observation simplifies the implementation significantly: instead of one complex d-dimensional computation we only have to perform d successive one-dimensional convolutions. (In fact, the Gaussian kernel is the *only* convolution kernel with this property [29]) This reduces $\mathcal{M}$ to $k \cdot d \cdot N^d$ and $\mathcal{A}$ to $(k-1)dN^d$.

The second observation may also lead to significant profits in computing time, provided that it is used properly. We will explain this according to the following definitions. Set h to half the kernel size k, such that it covers the top of the discrete Gaussian kernel and one of the tails, *i.e.,* $h = (k+1)/2$. Let $G[1..h]$ represent half the discrete Gaussian kernel in one dimension. This array is filled according to:

$$G[i] \;=\; \frac{1}{\gamma} \exp\left(-\frac{i^2}{2\,\sigma_n^2}\right) \quad , \quad i \in [0..h] \quad , \tag{6.19}$$

where γ is a normalization factor such that

$$G[1] \;+\; 2 \cdot \sum_{i=2}^{h} G[i] \;=\; 1 \quad . \tag{6.20}$$

Let $L[1..N]$ represent the pixel values of the 1D image line that is currently being processed. Now a 2D temporary array $T[1..N, 1..h]$ is filled according to:

$$T[i,j] \;=\; L[i] \cdot G[j] \quad . \tag{6.21}$$

Hence, the T-array contains all the possible multiplications of every pixel with every kernel index.

Next, notice that a straightforward convolution of a blurred pixel value $L'[i] = (L * G)[i]$ is calculated as:

$$\begin{aligned} L'[i] \;\;=\;\; & L[i-h+1]\,G[h] \;+\; ... \;+\; L[i-1]\,G[2] \;+\; L[i]\,G[1] \\ & L[i+1]\,G[2] \;+\; ... \;+\; L[i+h-1]\,G[h] \;, \end{aligned} \tag{6.22}$$

which can readily be rewritten as:

$$L'[i] \;=\; T[i,1] \;+\; \sum_{j=1}^{h-1} \Big(T[i-j, j+1] \;+\; T[i+j, j+1] \Big) \quad . \tag{6.23}$$

In this way, $\mathcal{M}$ has further been reduced to hdN^d, which is computationally cheaper than the kdN^d of the straightforward implementation, since $h \approx \frac{1}{2}k$. The number of additions is not further reduced by using the symmetry property of Gaussian kernels (*i.e.,* $\mathcal{A} = (k-1)dN^d$).

Note that the T-array indices as calculated in equation (6.23) may cross the array bounds. In those cases, the border problem becomes apparent. Therefore, an additional check is necessary to detect border errors and to calculate the correct value (the actual border handling, *e.g.,* 0^{th} order extrapolation). This can readily be achieved by using a macro in the implementation.

Optimization used	*factor per pixel*	
	$\mathcal{M}$	$\mathcal{A}$
none	k^d	$k^d - 1$
separability	$k\,d$	$(k-1)\,d$
symmetry	$\left(\frac{k}{2}\right)^d$	$k^d - 1$
separability & symmetry	$\frac{k+1}{2}\,d$	$(k-1)\,d$

Table 6.4. *Number of multiplications ($\mathcal{M}$) and additions ($\mathcal{A}$) needed per pixel of a d-dimensional image to calculate the convolution of that pixel with a kernel of size k^d. The numbers have been indicated for (*i*) conventional convolution, (*ii*) convolution based on the separability property, (*iii*) convolution based on the symmetry of Gaussian kernels, and (*iv*) convolution based on both the separability and the symmetry property.*

Finally, Table 6.4 gives a summary of the computing speed factors for the different optimization methods. The table indicates the $\mathcal{M}$ and $\mathcal{A}$ values per pixel, so all factors must be multiplied by N^d to obtain the number of operations necessary for the entire image.

When considering the symmetry optimization only (the third entry in the table), we have assumed that $1/2^d$ part of the kernel contains unique (different) values.

6.A.2 The Perona & Malik equation and Euclidean shortening flow

The image at time $t_0 + \triangle\tau$ can be written as:

$$L(\vec{x}, t_0 + \triangle\tau) = \sum_{n=0}^{\infty} \frac{(\triangle\tau)^n}{n!} \partial t^n L(\vec{x}, t_0) \tag{6.24}$$

If we include terms up to $n = 1$ we obtain the Euler forward scheme:

$$L(\vec{x}, t_0 + \triangle\tau) - L(\vec{x}, t_0) = \partial_t L \ \triangle\tau = F(L_i, L_{ij}, ...)\triangle\tau \tag{6.25}$$

Higher order terms can be included to improve the accuracy. We have to be aware whether these operators are still correctly represented at that scale.

$F(L_i, L_{ij}, ...)$ at the desired scale σ can be obtained by a convolution of the original image with the equivalent derivative of the Gaussian at scale σ. Convolutions are conveniently calculated in the Fourier domain because the Fourier transform $\mathcal{F}$ of a convolution in the spatial domain is equal to the product of the individual Fourier transforms in the Fourier domain. We need to compute both the Fourier transform of the original image and of the desired (combination) of derivatives of the Gaussian kernel. The Fourier transform of the n-th order derivative of a Gaussian kernel is obtained by an n-times multiplication with $-i\omega$:

$$\mathcal{F}\{\frac{\partial^n}{\partial\vec{x}^n} G(\vec{x}, \sigma)\} = (-i\omega)^n \mathcal{F}\{G(\vec{x}, \sigma)\} \tag{6.26}$$

After multiplication with the Fourier transform of $L_0(\vec{x})$, the inverse Fourier transform $\mathcal{F}^{-1}$ yields the desired derivative.

6.A.3 The median filter

Fast implementations of the median filter, such as in [82], are based on fixed values of m and d ($m = 3$, d=2). In this case, a lot of preprocessing work can be done before applying the filter to an actual image. This is not a suitable solution for generating scale spaces, however, since m varies from level to level and for different values of ψ.

The algorithm described in Numerical Recipes [90] is designed to read data in a sequential pass, for instance from an external tape. The trick is to limit the total number of passes, while the total number of interesting data points decreases at each pass. The basic idea is to keep track of the number of data points that are larger and smaller than the currently estimated median. In general, it takes $\log N^2$ passes through the data to find the median.

A rather simple and effective improvement on this algorithm is based on the fact that a better initial guess of the median can be made, because our data points do not have a sequential character; instead, they are simply present in memory. The initial guess is based on a small number (typically 7) of random pixels of the list. The algorithm then continues in an iterative way by scanning the list, while keeping track of the number of values that are larger and smaller than the 'best' value found so far. If the list has been scanned completely and the median value has not been found (*i.e.,* 'larger' and 'smaller' are not equal), then the pixel value closest to the previous estimate (this decision depends on the values of 'larger' and 'smaller' again) is considered to be the median in the next iteration step. The process continues until

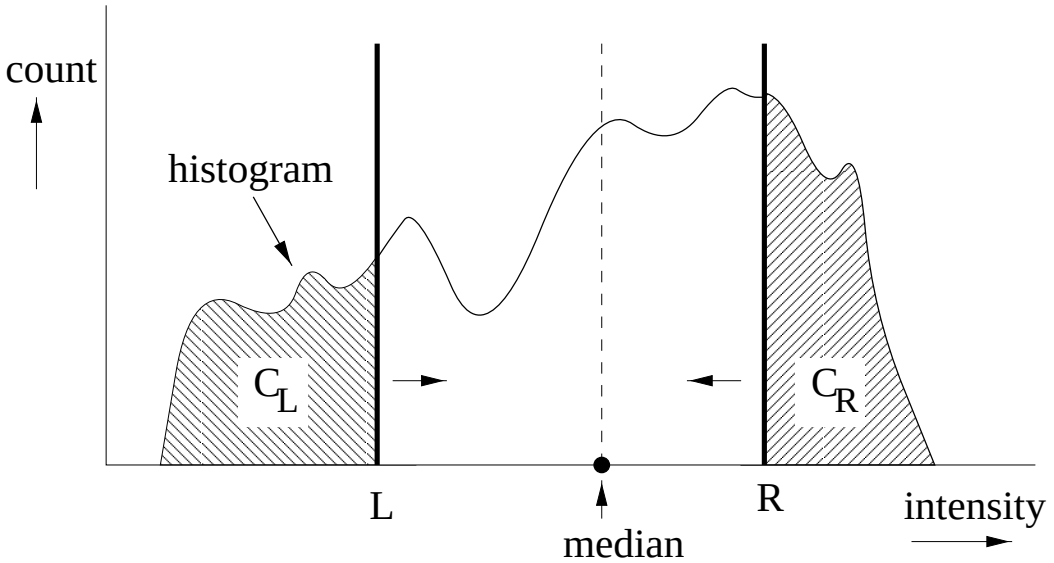


Fig. 6.15. *Schematic of the median filtering process for relatively large kernels. The points L and R approach each other such that the size of the areas C_L and C_R are kept as closely as possible. If L and R touch each other, the median has been found.*

the median has been found. We found that this technique is faster than $\log N^2$ passes, although the actual computation time depends strongly on the image data.

This improvement fails for larger kernel sizes (needed to create the levels of largest scale), since the computation time is still enormous. Therefore, we have also implemented the median filter based on the histogram distribution of an image (see Fig. 6.15). The idea is to detect the median by finding the histogram entry that divides the histogram in two parts that are equal in area. This can readily be effectuated by using two 'pointers', one starting at the most left histogram entry (L), and one at the right (R). Two cumulative sums (C_L and C_R) keep track of the total area covered by L and R so far. The smallest sum index (L or R) is shifted towards the middle of the histogram, and the cumulative sum is updated. This process continues until L and R are next to each other. The median then corresponds to the index with the largest C_i value.

An advantage of this method is that the sliding window can be used to further optimize the algorithm. Each time the kernel is moved to the next pixel, one row of pixels is shifted out and one row is shifted into the window. The corresponding data points of these rows are used to update the histogram, after which the next median can be found. The major advantage of this method is that it is fast even for very large values of m_n.

The only drawback of the method is that it can not be applied directly on floating point data. Then, a quantification step is necessary to construct a histogram, which slightly decreases the accuracy of the method.

6.A.4 The Kuwahara filter

As regards the implementation we start with the input image `Orig`, and calculate `OrigSqr`, containing the squared pixel values of `Orig`. Next, we perform the following four steps for every kernel width k that generates one level:

1. Calculate the 'mean' image `Mean(k)` according to equation (6.12). This image contains for every pixel the mean value of that pixel in a $k \times k$ neighborhood. Use 0^{th} order extrapolation at the borders to fill in the missing values.

2. Calculate `MeanSqr(k)`, which is the $k \times k$ mean image based on `OrigSqr`.

3. Calculate the 'variance' image `Var(k)`, which equals `MeanSqr(k) - Mean(k)`^{2}. Each pixel in `Var(k)` then contains the variance in a $k \times k$ neighborhood. (Note that the normalization factor of equation (6.13) can be omitted without changing the results of the MLV filtering.)

4. Finally, calculate the result image `Kuwa(k)`, which is done by finding the pixel in a $k \times k$ neighborhood with the smallest local variance value (as stored in `Var(k)`), and replace each pixel with the corresponding pixel value in `Mean(k)`.

With respect to the calculation of `Mean(k)` and `MeanSqr(k)`, note that a significant profit in time can be achieved by reusing the overlap between two succeeding windows in the shifting process.

6.B Scale space axioms and properties

We are aware that judging the axioms and selecting desirable properties for a scale space generator is a rather arbitrary process. Therefore, for each axiom/property we explain the methodology used, followed by a conclusion for each scale space generator.

6.B.1 Linearity

We have examined the linearity of the operator Ψ, operating on an input image L, according to the well-known linearity property:

$$\Psi\Big(aL_1(\vec{x}) \;+\; L_2(\vec{x})\Big) \;\stackrel{?}{=}\; a\Psi L_1(\vec{x}) \;+\; \Psi L_2(\vec{x}) \quad , \tag{6.27}$$

where $L_1 \neq L_2$ and a is a constant. The operator Ψ represents the scale space generator, *e.g.,* convolution with a Gaussian kernel for the linear scale space.

The convolution of an image with a Gaussian kernel is a linear algorithm. This can easily be seen by inspecting the multiplication and addition process for a single pixel (see appendix A). Hence, both the spatial and Fourier implementation of the linear scale space are linear by definition.

Not surprisingly, the nonlinear scale space generators Perona & Malik and the curve evolution process turn out to be highly nonlinear. This can be checked by expanding equations (6.4) and (6.9) according to the linearity equation (6.27). Much easier, however, is finding a counter example of two images L_1 and L_2 that fail the linearity test. The latter also works perfectly for the median and the Kuwahara filter.

6.B.2 Homogeneity

This property can be judged by looking at the blurring process itself. The process can be called 'homogeneous' if the region of neighboring pixels (centered around that pixel) that contribute to the blurred value of the pixel at hand has a constant volume and shape for every pixel in the image. (Note that this does *not* require that all the pixels equally contribute to the blurring.)

In the case that a kernel is used, the kernel itself can be inspected. This accounts for the Gaussian kernel (perfectly homogeneous), and the median and the Kuwahara filter (both homogeneous). The last two kernels are homogeneous because all the sub-windows contribute equally to the *choice* which sub-window is chosen. Note that in the case of a median filter, the size of such a sub-window is 1, and for the Kuwahara filter larger than 1.

Perona & Malik and curve evolution both depend on local image features, and can therefore not be called homogeneous.

6.B.3 Isotropy

The blurring process can be called 'isotropic' if for every pixel in the image the neighboring pixels contribute to the blurred value of the pixel at hand according to their Euclidean distance. This relation (*i.e.,* distance versus contribution) need not be linear.

The symmetry of the Gaussian kernel makes the linear scale space generator (spatial/Fourier domain) isotropic. The median filter can be called 'reasonably isotropic', because it is symmetric in every Euclidean direction. The kernel not being circular, however, introduces artifacts—especially at larger scales.

The Perona & Malik equation, although often called 'anisotropic', actually is isotropic. The confusion is caused by the fact that the influence of all the neighboring pixels is locally varying.

The Kuwahara filter uses only one of the surrounding windows, while in the Euclidean shortening flow only diffusion in the isophote direction takes place. Hence, both methods are not isotropic.

6.B.4 Self-similarity

For both the linear and the nonlinear scale space generators, it is not hard to see that they are self-similar. In fact, they are often implemented this way (*i.e.,* iteratively), although this may introduce discretization artifacts at larger scales.

If this property can not easily be derived analytically, a simple test is the following: for two successive blurring steps b_1 and b_2 try to find a single (larger) blurring kernel b_3 that matches the overall result $b_1 \oplus b_2$. To make a fair and robust comparison, use two different combinations of b_1 and b_2: (i) a large blurring step with a small one, and (ii) two blurring steps that are closely together. For the median and the Kuwahara filter it can readily be tested that such a blurring kernel b_3 can seldom be found.

6.B.5 Commutativity

This criterion can easily be tested by comparing two different blurring results $b_1 \circ b_2$ and $b_2 \circ b_1$. As before, two different experiments have been done.

Except for the median and the Kuwahara filter, all scale space generators are commutative. That is, Gaussian convolution is commutative (linear scale spaces), and the iterative character of both the Perona & Malik and the curve evolution filter indicates that it makes no difference whether b_1 precedes b_2 or the other way around: the effective result is the same.

6.B.6 Adaptive feature preservation

One of the characteristics of a linear scale space is that it does not take *a priori* information into account. Hence, there is no possibility to preserve any specific feature information whatsoever. On the other hand, the scale spaces obtained with the Perona & Malik equation and Euclidean shortening flow give specific control over features. In the original Perona & Malik paper the equations (6.5) and (6.6) are presented as edge-preserving control functions. One can easily extend these formulas such that other features, such as corners, are effectively preserved. The Euclidean shortening flow preserves features by not diffusing in the gradient direction. From the figures in section 6.7.1 it is obvious that the Perona & Malik equation tends to preserve the features 'better' (*i.e.,* longer in terms of diffusion) than the Euclidean shortening flow. For the latter, an arbitrary contour will always converge to a circle (in 2D) and subsequentially shrink to a point [36, 38].

Because the design of both the median and the Kuwahara filter is rather strict, no high adaptive feature preservation may be expected. However, they both perform remarkably well with respect to preserving edges, especially at the smaller scales. Other researchers have shown that corners easily disappear by median filtering, but are preserved by Kuwahara filtering [106]. At larger scales, both filters fail dramatically to preserve any image information, but then again, they were never designed to be used this way.

6.B.7 Grey value invariance

The evolution is invariant under grey value transformations if equation (6.15) is satisfied. It is readily shown that the linear diffusion equation—in which the luminance is the potential for the diffusive current—and the Perona & Malik equation are not grey value invariant. Since the Euclidean shortening flow evolves curves as a function of their curvature, it is grey value invariant by definition (it is a pure geometric evolution). The median filter is also grey value invariant, since the sequence of values is not altered by a strict monotone grey value transformation of the image intensities. Finally, the Kuwahara filter is not grey value invariant, since the variances in the respective sub-windows may change after a transformation. This also affects the choice of the sub-window that is chosen, and hence the mean value that makes up the new pixel value.

6.B.8 Convergence

Every implementation of a linear scale space should converge to the average image intensity. Indeed, Fourier domain implementations nicely converge to this value. For spatial domain implementations of the linear scale space, convergence is only guaranteed if the boundary problem is tackled by using the average image intensity to fill in the missing data. Frequently used techniques like 0^{th} order extrapolation do not force the image to converge to the mean image value. Rather, the extreme values asymptotically approach each other if the scale is increased, but do not merge together until the scale is extremely large. This may prevent linking models from converging to a single node.

A property of nonlinear scale space generators is that they preserve features, as contrasted with the linear scale space (see above). If this property is still present at the larger scales, then the linking may have difficulties to converge. It is unlikely that dark and bright blobs will merge together into a single link (a *catastrophe*), because the isophote structure of the underlying scale space does not forces this event to happen.

However, if we take a closer look at features in scale space (*e.g.,* edges), then it is undesirable that they remain present at every scale level. We rather would like to use the feature preserving character of the scale space generator to facilitate the linking of pixels close to these edges (*i.e.,* the difficult pixels), followed by a fast and converging linking step to collect the sub-segments at a larger scale. Therefore, from a pure practical point of view it makes sense to combine feature preserved blurring at the first levels of scale and straight linear blurring at larger scales.

The nonlinear scale space generators according to the Perona & Malik equation and the Euclidean shortening flow behave differently in this respect. Euclidean shortening flow does not require measures to impose convergence. For the Perona & Malik stacks, however, we have to add a series of linearly blurred levels on top of the stack to force convergence of the scale space to the average image intensity. We found that

it depends on the image at which level the nonlinear scale space is best continued by a linear scale space (although the actual differences in PPE costs are minimal). In our experiments we have tried to use 1 up to 10 levels for the nonlinear part of the scale space, added with linearly blurred levels. For the results presented in this chapter we have used 2 nonlinearly blurred levels for the BRAIN.cor image, and 5 for the BRAIN.tr image. These numbers corresond to the minimum PPE costs for the segmentations of those images.

As regards the median and the Kuwahara filter, the convergence property is clear. As soon as a (sub-)window covers the entire image at every pixel in the image, the resulting level will be homogeneous. For the Kuwahara filter ($k \approx 2N$), the homogeneous value will be the average image value. For the median ($m = 2N$), however, this end value is strongly image dependent (although the image will be homogeneous). If a hyperstack requires more levels for complete convergence than the series generated by the filter process, the final (homogeneous) level is simply used to extend the stack with more levels.

6.B.9 Low noise sensitivity

From a pure theoretical point of view, all the six scale space generators described in this chapter remove noise if the scale is increased. The low-pass filtering character varies from method to method (*e.g.,* locally versus globally), but normally the noise does not survive the first levels of the scale space.

The main difference is determined by the behavior of each filter for *different* noise levels. In this respect, the median filter performs excellent: if a noise peak is increased, this has almost no influence on the result. The same effect can be observed for the Kuwahara filter, although to a lesser extent. Here, the selected sub-window (*i.e.,* the one with smallest variance) will often be the same, but the mean value of the sub-window will depend slightly on the noise level.

The propagation of noise in linear scale space has been studied extensively by Blom [12]. He shows that the propagation of noise variance is always substantially reduced when scale is increased. For the computation of higher order derivatives noise is usually enhanced, owing to its high pass nature. However, in the same paper [12] is shown that determining derivatives is stable if we use the scale at which they are evaluated properly. Therefore, the two nonlinear equations according to Perona & Malik and Euclidean shortening flow are also robust against different noise levels. Features can be extracted nicely irrespective of the noise level by using the scale space concept. Hence, the gradient information that is used by the Perona & Malik filter does not loose its influence at increasing noise. If the noise level is increased, the curve evolution will only be disturbed locally, but at larger scales the curve still converges to the circular shape.

6.B.10 Computational speed

We have compared the computing times for the different scale space generators necessary to create the entire scale space. There are two different criteria to be checked: (*i*) the algorithm should be fast (*i.e.,* measured in absolute time units); (*ii*) the speed of the calculation should preferably be scale independent.

Obviously, we can only judge our scale space generators based on the implementations that we have available. Nevertheless, we believe that we can make a fair comparison, apart from a few side notes.

The linear scale space generators are the absolute winners. The spatial domain implementation (see appendix A) is extremely fast at the smallest scale levels—even for large images. If scale increases, the algorithm slows down, but within proportions. The Fourier domain implementation is slower than the spatial domain counterpart at small scales, but the major advantage is that the computation time does not alter if scale is increased.

Explicit implementations of the Perona & Malik equation and Euclidean shortening flow are slow for large scale levels, since only a small time step is allowed. In this case, implicit methods are preferred, which allow larger time steps.

The Kuwahara filter has the largest decay in performance. The computation time increases exponentially with increasing window size, although it is very fast for relatively small kernels.

Finally, for the median filter we have used two implementations. The straightforward method—based on the counting the number of pixels that are smaller/larger than the best median found so far—for the first levels of the scale space, and the histogram division for the larger scale levels. The former behaves similar to the Kuwahara filter: fast for small values of m, extremely slow for large values of m. The computing times of the latter are relatively independent of the scale level (and fast enough to yield manageable results).

Level nr.	*spatial*	*Fourier*	*Perona & Malik*	*Euclidean short. flow*	*median*	*Kuwahara*
1	0.3	3.4	14.8	5.9	16.0	0.9
5	0.5	3.5	237.2	95.1	20.5	3.2
10	2.0	3.6	7589.7	3041.9	31.3	50.4
1–10	6.5	23.8	7589.7	3041.9	220.0	113.0

Table 6.5. *Computation times (measured in seconds) for the six different scale space generators. Indicated are the times needed to created levels 1, 5, 10, and the total time to create the levels 1 up to 10. Times are measured on a HP 9000/755 series (user times).*

References

[1] L. Alvarez, F. Guichard, P. L. Lions, and J. M. Morel. Axioms and fundamental equations of image processing. *Arch. for Rational Mechanics*, 123(3):199–257, September 1993.

[2] L. Alvarez, P.-L. Lions, and J.-M. Morel. Image selective smoothing and edge detection by nonlinear diffusion. II. *SIAM J. Num. Anal.*, 29(3):845–866, June 1992.

[3] S. Angenent. On the formation of singularities in the curve shortening flow. *J. Differential Geometry*, 33:601–633, 1991.

[4] S. Angenent. Parabolic equations for curves on surfaces, part II. intersections, blowup, and generalized solutions. *Annals of Mathematics*, 133:171–215, 1991.

[5] H. J. Antonisse. Image segmentation in pyramids. *Computer Graphics and Image Processing*, 19:367–383, 1982.

[6] J. Arvo. A simple method for box-sphere intersection testing. In A. S. Glassner, editor, *Graphics gems*, chapter 5: 3D Geometry, pages 335–339. Academic Press, 1990.

[7] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda. Uniqueness of the gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):26–33, 1986.

[8] J. A. Bangham, P. Ling, and R. Harvey. Scale-space from nonlinear filters. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 163–168. IEEE Computer Society Press, 1995.

[9] C. Barillot, F. Lachmann, B. Gibaud, and J. M. Scarabin. 3D display of MRI data in neurosurgery: segmentation and rendering aspects. In *Proc. SPIE 1445 Image Processing*, pages 54–65. 1991.

[10] P. Bijl. *Aspects of Visual Contrast Detection.* PhD thesis, Utrecht University, The Netherlands, 1991.

[11] M. Bister, J. Cornelis, and A. Rosenfeld. A critical view of pyramid segmentation algorithms. *Pattern Recognition Letters*, 11:605–617, 1990.

[12] J. Blom, B. M. ter Haar Romeny, A. Bel, and J. J. Koenderink. Spatial derivatives and the propagation of noise in Gaussian scale-space. *Journal of Visual Communications and Image Representations*, 4(1):1–13, March 1993.

[13] G. Borgefors. Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, 1988.

[14] R. W. Brockett and P. Maragos. Evolutions equations for continuous-scale morphology. In *International Conference on Acoustics, Speech, Signal Processing*. IEEE, 1992.

[15] M. E. Brummer, R. M. Mersereau, R. L. Eisner, and R. R. J. Lewine. Automatic detection of brain contours in MRI data sets. In A. C. F. Colchester and D. J. Hawkes, editors, *IPMI, Proc. of the 12th conference*, pages 188–204. Springer-Verlag, Berlin, July 1991.

[16] P. J. Burt, T. H. Hong, and A. Rosenfeld. Segmentation and estimation of image region properties through cooperative hierarchical computation. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(12):802–809, 1981.

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

[18] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM J. Num. Anal.*, 29(1):182–193, 1992.

[19] M. Chen and P. Yan. A multiscale approach based on morphological filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):694–700, 1989.

[20] J. O. Coplien. *Advanced C++ – Programming Styles and Idioms*. Addison-Wesley, Reading, Massachusetts, 1991.

[21] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Commun. Pure Appl. Math.*, 41:909–996, 1988.

[22] C. N. de Graaf, A. S. E. Koster, K. L. Vincken, and M. A. Viergever. A methodology for the validation of image segmentation methods. In J. N. Brown and P. Santiago, editors, *Computer-Based Medical Systems*, pages 17–24. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[23] C. N. de Graaf, S. M. Pizer, A. Toet, J. J. Koenderink, P. Zuidema, and P. P. Van Rijk. Pyramid segmentation of medical 3D images. In M. L. Rhodes, editor, *Proc. International Symposium on Computer Graphics*, pages 71–77. Innsbruck, 1984.

[24] C. N. de Graaf, A. Toet, J. J. Koenderink, P. Zuidema, and P. P. Van Rijk. Some applications of hierarchical image processing algorithms. In F. Deconinck, editor, *Information Processing in Medical Imaging*, pages 343–369. Martinus Nijhoff, The Hague, 1984.

[25] C. N. de Graaf, K. L. Vincken, M. A. Viergever, J. J. Koenderink, F. J. R. Appelman, and O Ying-Lie. A hyperstack for the segmentation of 3D images. In D. A. Ortendahl and J. Llacer, editors, *Information Processing in Medical Imaging*, pages 399–413. Wiley-Liss, New York, NY, 1991.

[26] R. Deriche. Recursively implementing the gaussian and its derivatives. In V. Srinivasan, Ong Sim Heng, and Ang Yew Hock, editors, *Proceedings of the 2nd Singapore International Conference on Image Processing*, pages 263–267. World Scientific Publishing Co. Pte. Ltd., Singapore, 1992. September 7–11.

[27] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *Computer Graphics*, 22(4):65–74, 1988.

[28] C. L. Epstein and M. Gage. *Wave Motion: Theory, Modeling, and Computation*, chapter The Curve Shortening Flow. Springer-Verlag, New York, 1987. A. Chorin, A. Majda, (eds.).

[29] L. M. J. Florack. *The Syntactical Structure of Scalar Images*. PhD thesis, Utrecht University, Utrecht, The Netherlands, 1993.

[30] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. Scale-space and the differential structure of images. *Image and Vision Computing*, 10:376–388, 1992.

[31] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. Cartesian differential invariants in scale-space. *Journal of Mathematical Imaging and Vision*, 3(4):327–348, 1993.

[32] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. General intensity transformations and differential invariants. *Journal of Mathematical Imaging and Vision*, 4(2):171–187, May 1994.

[33] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. Linear scale-space. *Journal of Mathematical Imaging and Vision*, 4(4):325–351, 1994.

[34] L. M. J. Florack, A. H. Salden, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. Nonlinear scale-space. *Image and Vision Computing*, 13(4):279–294, May 1995.

[35] D. S. Fritsch, S. M. Pizer, B. Morse, D. H. Eberly, and A. Liu. The multi-scale medial axis and its applications in image registration. *Pattern Recognition Letters*, 15(5):445–452, 1994.

[36] M. Gage. Curve shortening makes convex curves circular. *Invent. Math.*, 76:357–364, 1984.

[37] M. Gage and R. S. Hamilton. The heat equation shrinking convex plane curves. *J. Differential Geometry*, 23:69–96, 1986.

[38] M. Grayson. The heat equation shrinks embedded plane curves to round points. *Journal of Differential geometry*, 26:285–314, 1987.

[39] L. D. Griffin, A. C. F. Colchester, and G. P. Robinson. Scale and segmentation of grey-level images using maximum gradient paths. *Image and Vision Computing*, 10:389–402, 1992.

[40] A. Grossmann and J. Morlet. Decomposition of Hardy functions into square integrable wavelets of constant shape. *SIAM J. Math.*, 15:723–736, 1984.

[41] B. M. ter Haar Romeny, L. M. J. Florack, J. J. Koenderink, and M. A. Viergever. Scale-space: its natural operators and differential invariants. In A. C. F. Colchester and D. J. Hawkes, editors, *IPMI, Proc. of the 12th conference*, pages 239–255. Springer-Verlag, Berlin, July 1991.

[42] B. M. ter Haar Romeny, W. J. Niessen, J. Wilting, and L. M. J. Florack. Differential structure of images: Accuracy of representation. In *Proc. First IEEE Internat. Conf. on Image Processing*, pages 21–25, Austin, TX, 1994. IEEE. IEEE Catalog number 94CH35708.

[43] R. W. Hamming. *Digital filters.* Prentice-Hall, London, 1989.

[44] S. Haring, M. A. Viergever, and J. N. Kok. A multiscale approach to image segmentation using Kohonen networks. In A. Gmitro and H. H. Barrett, editors, *IPMI, Proc. of the 13th conference*, pages 212–224. Springer-Verlag, Berlin, 1993.

[45] J. P. Heiken, J. A. Bricken, and M. W. Vannier. Spiral (helical) CT. *Radiology*, 189:647–656, 1993.

[46] T. H. Hong, K. A. Narayanan, S. Peleg, A. Rosenfeld, and T. Silberberg. Image smoothing and segmentation by multiresolution pixel linking: further experiments and extensions. *IEEE Transactions on Systems, Man, and Cybernetics*, 12(5):611–622, 1982.

[47] A. K. Jain. *Fundamentals of Digital Image Processing*, chapter 4: Image Sampling and Quantization, pages 80–99. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1989.

[48] W. A. Kalender, W. Seissler, and E. Klotz. Spiral volumetric CT with single-breath-hold technique continuous transport, and continuous scanner rotation. *Radiology*, 176:181–183, 1990.

[49] N. Karssemeijer, L. J. Th. O. van Erning, and E. G. J. Eijkman. Recognition of organs in CT-image sequences: a model guided approach. *Computers and Biomedical Research*, 21:434–438, 1988.

[50] S. Kasif and A. Rosenfeld. Pyramid linking is a special case of ISODATA. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(1):84–85, 1983.

[51] B. B. Kimia and K. Siddiqi. Geometric heat equation and nonlinear diffusion of shapes and images. In *Proc. IEEE CVPR*, pages 113–120, Seattle, WA, 1994.

[52] B. B. Kimia, A. Tannenbaum, and S. W. Zucker. Towards a computational theory of shape, an overview. In *Proc. First European Conference on Computer Vision*, volume 427, pages 402–407, New York, 1990. Springer-Verlag. Lecture Notes in Computer Science.

[53] B. B. Kimia, A. Tannenbaum, and S. W. Zucker. Shapes, shocks, and deformations, I. *International Journal of Computer Vision*, 15(3):189–224, 1994.

[54] J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[55] J. J. Koenderink. *Brain Theory, spatio-temporal aspects of brain function*, chapter Embodiments of geometry, pages 3–28. Elsevier, Amsterdam, The Netherlands, 1993.

[56] J. J. Koenderink and A. J. van Doorn. Visual detection of spatial contrast; influence of location in the visual field, target extent and illuminance level. *Biological Cybernetics*, 30:157–167, 1978.

[57] A. S. E. Koster. A linking strategy for multi-scale image segmentation. Report 3DCV 90-05, Utrecht University, 1990.

[58] A. S. E. Koster, K. L. Vincken, C. N. De Graaf, , O. C. Zander, and M. A. Viergever. Heuristic linking models in multiscale image segmentation. *Computer Vision and Image Understanding*, 65(3):382–402, 1997.

[59] A. S. E. Koster, O. C. Zander, K. L. Vincken, and M. A. Viergever. Model-based evaluation of image segmentation methods. Technical report, Utrecht University. Submitted for publication.

[60] W. G. Kropatsch. A pyramid that grows by powers of 2. volume 3, pages 315–322, 1985.

[61] W. G. Kropatsch. Preserving contours in dual pyramids. In *Proc. of the 9th Conference on Pattern Recognition*, IEEE Catalog number 88CH2614-6, pages 563–565. IEEE, 1988.

[62] M. Kuwahara. Processing of ri-angiocardiographic images. In K. Preston and M. Onoe, editors, *Digital Processing of Biomedical Images*, pages 187–202. Plenum Press, New York, 1976.

[63] M. Levoy. Rendering of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):28–37, 1988.

[64] L. M. Lifshitz and S. M. Pizer. A multiresolution hierarchical approach to image segmentation based on intensity extrema. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):529–541, 1990.

[65] T. Lindeberg. *Discrete Scale-Space Theory and the Scale-Space Primal Sketch.* PhD thesis, Royal Institute of Technology, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, S-100 44 Stockholm, Sweden, 1991.

[66] T. P. Lindeberg. Scale-space for discrete signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):234–254, 1990.

[67] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–170, 1987.

[68] J. B. A. Maintz, P. A. van den Elsen, and M. A Viergever. Comparison of feature-based matching of CT and MR brain images. In N. Ayache, editor, *CVRMed*, volume 905 of *Lecture notes in computer science*, pages 219–228, Berlin, 1995. Springer-Verlag.

[69] J. B. A. Maintz, P. A. van den Elsen, and M. A Viergever. Evaluation of ridge seeking operators for multimodality medical image matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):353–365, 1996.

[70] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–694, 1989.

[71] P. Maragos. Tutorial on advances in morphological image processing and analysis. *Optical Engineering*, 26:623–632, 1987.

[72] D. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.

[73] P. Meer, S. N. Jiang, E. S. Baugher, and A. Rosenfeld. Robustness of image pyramids under structural pertubations. *Computer Vision, Graphics, and Image Processing*, 44:307–331, 1988.

[74] W. Menhardt. Image analysis using iconic fuzzy sets. In Y. Kodratoff, editor, *Proc. of European Conference on Artificial Intelligence*, pages 672–674. Pitman Publishing, London, 1988.

[75] W. Menhardt. Iconic fuzzy sets for MR image segmentation. In A. E. Todd-Pokropek and M. A. Viergever, editors, *Medical Images: Formation, Handling and Evaluation, NATO ASI Series F98*, pages 579–591. Springer-Verlag, Berlin, 1992.

[76] C. Montani and R. Scopigno. Spheres-to-voxels conversion. In A. S. Glassner, editor, *Graphics gems*, chapter 5: 3D Geometry, pages 327–334. Academic Press, 1990.

[77] P. Neskovich and B. B. Kimia. Three-dimensional shape representation from curvature dependent surface evolution. Technical Report LEMS-128, Division of Engineering, Brown University, 1993.

[78] W. J. Niessen, B. M. ter Haar Romeny, L. M. J. Florack, and M. A. Viergever. A general framework for geometry-driven evolution equations. *International Journal of Computer Vision*, 21(3):187–205, 1997.

[79] W. J. Niessen, B. M. ter Haar Romeny, and M. A. Viergever. Numerical analysis of geometry-driven diffusion equations. In B. M. ter Haar Romeny, editor, *Geometry-Driven Diffusion in Computer Vision*, Computational Imaging and Vision, pages 393–410. Kluwer Academic Publishers, 1994.

[80] D. A. Ortendahl and J. W. Carlson. Segmentation of magnetic resonance images using fuzzy clustering. In C. N. de Graaf and M. A. Viergever, editors, *IPMI, Proc. of the 10th conference*, pages 91–106. Plenum Press, New York, 1988.

[81] S. Osher and S. Sethian. Fronts propagating with curvature dependent speed: algorithms based on the Hamilton-Jacobi formalism. *J. Computational Physics*, 79:12–49, 1988.

[82] A. W. Paeth. Median finding on a 3×3 grid. In A. S. Glassner, editor, *Graphics gems*, chapter 3: Image Processing, pages 171–175. Academic Press, 1990.

[83] J. A. Parker, R. V. Kenyon, and D. E. Troxel. Comparison of interpolating methods for image resampling. *IEEE Transactions on Medical Imaging*, MI-2(1):31–39, March 1983.

[84] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[85] S. M. Pizer, T. J. Cullip, and R. E. Fredericksen. Toward interactive object definition in 3D scalar images. In K. H. Hohne, H. Fuchs, and S. M. Pizer, editors, *3D Imaging in medicine*, pages 83–105. Springer-Verlag, Berlin, 1990.

[86] S. M. Pizer, J. J. Koenderink, L. M. Lifshitz, L. Helmink, and A. D. J. Kaasjager. An image description for object definition, based on extremal regions in the stack. In S. L. Bacharach, editor, *Information Processing in Medical Imaging*, pages 24–37. Martinus Nijhoff, Dordrecht, 1986.

[87] A. Pommert, U. Tiede, G. Wiebecke, and K. Höhne. Surface shading in tomographic volume visualization: a comparative study. In *Proc. First Conference on Visualization in Biomedical Computing*, pages 19–26. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[88] W. K. Pratt. *Digital Image Processing, 2nd ed.* Wiley, New York, 1991.

[89] W. K. Pratt, T. J. Cooper, and I. Kabir. Pseudomedian filter. In F. J. Corbett, editor, *Architectures and Algorithms for Digital Image Processing II*, pages 34–43. Proc. SPIE 534, 1985.

[90] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*, chapter 13: Statistical description of data, pages 476–479. Cambridge University Press, New York, 1988.

[91] S. P. Raya. Low-level segmentation of 3-D magnetic resonance brain images–a rule-based system. *IEEE Transactions on Medical Imaging*, 9(3):327–337, 1990.

[92] P. Roos. *Reversible compression of medical images.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 1991.

[93] P. Roos, M. A. Viergever, M. C. A. van Dijke, and J. H. Peters. Reversible intraframe compression of medical images. *IEEE Transactions on Medical Imaging*, 7:328–336, 1988.

[94] A. Rosenfeld. The fuzzy geometry of image subsets. *Pattern Recognition Letters*, 2:311–317, 1984.

[95] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[96] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[97] A. H. Salden, L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. Multi-scale analysis and description of image structure. *Nieuw Archief voor Wiskunde*, 10(3):309–326, 1992.

[98] A. H. Salden, B. M. ter Haar Romeny, L. M. J. Florack, J. J. Koenderink, and M. A. Viergever. A complete and irreducible set of local orthogonally invariant features of 2-dimensional images. In I. T. Young, editor, *Proc. of the 11th Conference on Pattern Recognition*, volume III: Image, Speech and Signal Analysis, pages 180–184, The Hague, the Netherlands, August 30–September 3 1992. IEEE Computer Society Press, Los Alamitos.

[99] G. Sapiro and A. Tannenbaum. Affine invariant scale-space. *International Journal of Computer Vision*, 11:25–44, 1993.

[100] G. Sapiro and A. Tannenbaum. On affine plane curve evolution. *Journal of Functional Analysis*, 119(1):79–120, January 1994.

[101] G. Sapiro, A. Tannenbaum, Y. You, and M. Kaveh. Experiments on geometric enhancement. In *International Conference on Image Processing*, pages 472–475. IEEE, 1994.

[102] T. Schiemann, M. Bomans, U. Tiede, and K. H. Höhne. Interactive 3D-segmentation. In R. A. Robb, editor, *Visualization in Biomedical Computing 1992*, pages 376–383. Proc. SPIE 1808, 1992.

[103] H. Schildt. *C: The Complete Reference*. Osborne McGraw-Hill, Berkeley, CA, 1987.

[104] M. A. Schulze and J. A. Pearce. Some properties of the two-dimensional pseudo-median filter. In E. R. Dougherty, J. Astola, and C. G. Boncelet, Jr., editors, *Nonlinear Image Processing III*, pages 48–57. Proc. SPIE 1451, 1991.

[105] M. A. Schulze and J. A. Pearce. Linear combinations of morphological operators: the midrange, pseudomedian, and LOCO filters. In *Proc. IEEE Int. Conf. Acoust, Speech, Signal Process, Vol. V*, pages 57–60, 1993.

[106] M. A. Schulze and J. A. Pearce. Value-and-criterion filters: a new filter structure based upon morphological opening and closing. In E. R. Dougherty, J. Astola, and H. Longbotham, editors, *Nonlinear Image Processing IV*, pages 106–115. Proc. SPIE 1902, 1993.

[107] M. A. Schulze and J. A. Pearce. A morphology-based filter structure for edge-enhancing smoothing. In *Proc. IEEE Int. Conf. on Image Processing, Vol. II*, pages 530–534, 1994.

[108] J. A. Sethian. *An Analysis of Flow Propagation.* PhD thesis, University of California, 1982.

[109] J. A. Sethian. Curvature and the evolution of fronts. *Comm. Math. Phys.*, 101:487–499, 1985.

[110] C. A. Shaffer. Fast circle-rectangle intersection checking. In A. S. Glassner, editor, *Graphics gems*, chapter 1: 2D Geometry, pages 51–53. Academic Press, 1990.

[111] B. Stroustrup. *The C++ programming language (second edition).* Addison-Wesley, Cambridge, MA, 1991.

[112] G. R. Timmens. Resampling of multidimensional image data. Master's thesis, Delft, University of Technology, the Netherlands, 1991. Report 3DCV 91-27.

[113] R. van den Boomgaard. *Mathematical Morphology: Extensions towards Computer Vision.* PhD thesis, University of Amsterdam, 1992.

[114] R. Van Os. The inca-pyramid algorithm. Report VMFLF 30-84, Institute for Nuclear Medicine, University Hospital Utrecht, 1984.

[115] B. C. Vemuri, A. Radisavljevic, and C. M. Leonard. Multi-resolution stochastic 3D shape models for image segmentation. In A. Gmitro and H. H. Barrett, editors, *IPMI, Proc. of the 13th conference*, pages 62–76. Springer-Verlag, Berlin, 1993.

[116] K. L. Vincken. The hyperstack, a multiresolution technique for the segmentation of 3D images. Report 3DCV 89-04, Delft Univ. of Technology, 1989.

[117] K. L. Vincken and F. J. R. Appelman. Accurate conversion of geometrical objects to voxel-based images. Report 3DCV 91-20, Utrecht University, 1991.

[118] K. L. Vincken, C. N. de Graaf, A. S. E. Koster, M. A. Viergever, F. J. R. Appelman, and G. R. Timmens. Multiresolution segmentation of 3D images by the hyperstack. In *Proc. First Conference on Visualization in Biomedical Computing*, pages 115–122. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[119] K. L. Vincken, L. M. J. Florack, A. S. E. Koster, W. J. Niessen, and M. A. Viergever. Outer scale reduction in multiscale image analysis. Submitted for publication.

[120] K. L. Vincken, A. S. E. Koster, C. N. de Graaf, F. J. R. Appelman, and M. A. Viergever. Hyperstack segmentation of multidimensional images. Report ICU-IS 95-01, Utrecht University. Submitted for publication.

[121] K. L. Vincken, A. S. E. Koster, and M. A. Viergever. Probabilistic multiscale image segmentation – set-up and first results. In R. A. Robb, editor, *Visualization in Biomedical Computing 1992*, pages 63–77. Proc. SPIE 1808, 1992.

[122] K. L. Vincken, A. S. E. Koster, and M. A. Viergever. PROMISE, a system for probabilistic multiscale image segmentation. In M. A. Viergever, editor, *Volume Image Processing '93*, pages 5–8. SCVR, Utrecht, 1993.

[123] K. L. Vincken, A. S. E. Koster, and M. A. Viergever. Probabilistic segmentation of partial volume voxels. *Pattern Recognition Letters*, 15(5):477–484, 1994.

[124] K. L. Vincken, A. S. E. Koster, and M. A. Viergever. Probabilistic hyperstack segmentation of MR brain data. In N. Ayache, editor, *Computer Vision, Virtual Reality and Robotics in Medicine, Proc. CVRMed'95*, volume 905 of *Lecture Notes in Computer Science*, pages 351–357. Springer-Verlag, Berlin, 1995.

[125] K. L. Vincken, A. S. E. Koster, and M. A. Viergever. Probabilistic hyperstack segmentation of MR brain data. In L. Beolchi and M. H. Kuhn, editors, *Medical Imaging, Analysis of Multimodality 2D/3D Images*, pages 145–157. IOS Press, Amsterdam, 1995.

[126] K. L. Vincken, A. S. E. Koster, and M. A. Viergever. Probabilistic multiscale image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):109–120, 1997.

[127] K. L. Vincken, W. J. Niessen, and M. A. Viergever. Blurring strategies for image segmentation using a multiscale linking model. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR'96*, pages 21–26, San Francisco, CA, 1996. IEEE Computer Society Press.

[128] J. Weickert. Scale-space properties of nonlinear diffusion filtering with a diffusion tensor. Technical Report 110, Laboratory of Technomathematics, 1994.

[129] R. Whitaker. Algorithms for implicit deformable models. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 822–827. IEEE Computer Society Press, 1995.

[130] R. Whitaker and G. Gerig. Vector-valued diffusion. In B. M. ter Haar Romeny, editor, *Geometry-Driven Diffusion in Computer Vision*, Computational Imaging and Vision, pages 393–410. Kluwer Academic Publishers, 1994.

[131] A. P. Witkin. Scale space filtering. In *Proc. International Joint Conference on Artificial Intelligence*, pages 1019–1023. Karlsruhe, W. Germany, 1983.

[132] A. J. Worth and D. N. Kennedy. Segmentation of magnetic resonance brain images using analog constraint satisfaction neural networks. In A. Gmitro and H. H. Barrett, editors, *IPMI, Proc. of the 13th conference*, pages 225–243. Springer-Verlag, Berlin, 1993.

[133] R. A. Young. The Gaussian derivative theory of spatial vision: Analysis of cortical cell receptive field line-weighting profiles. Publication GMR-4920, General Motors Research Labs, Computer Science Dept., 30500 Mound Road, Box 9055, Warren, Michigan 48090-9055, May 28 1985.

[134] R. A. Young. The Gaussian derivative model for machine vision: Visual cortex simulation. *Journal of the Optical Society of America*, July 1986.

[135] R. A. Young. Simulation of human retinal function with the Gaussian derivative model. In *Proc. IEEE CVPR CH2290-5*, pages 564–569, Miami, Fla., 1986.

[136] O. C. Zander. Costq: Evaluating segmentations of two- and three dimensional medical images. Master's thesis, Delft, University of Technology, The Netherlands, 1995. Report 3DCV 91-27.

[137] S. W. Zucker and R. A. Hummel. Receptive fields and the representation of visual information. In *Proc. of the 7th Conference on Pattern Recognition (Montreal, Canada, July 30-August 2, 1984)*, IEEE Catalog number 84CH2046-1, pages 515–517, 1984.

[138] K. J. Zuiderveld. *Visualization of multimodality medical volume data using object-oriented methods*. PhD thesis, Utrecht University, The Netherlands, 1995.

[139] K. J. Zuiderveld, A. H. J. Koning, and M. A. Viergever. A flexible software architecture for visualization of multi-modality volume data. Report 3DCV 92-12, Utrecht University, 1992.

Publications

(Note: this chapter has been adapted to be up-to-date in March 1998)

Articles in international journals

- K.L. Vincken, A.S.E. Koster and M.A. Viergever (1994), "Probabilistic segmentation of partial volume voxels", *Pattern Recognition Letters*, vol. 15, no. 5, 477–484.

- K.L. Vincken, A.S.E. Koster and M.A. Viergever (1997), "Probabilistic multiscale image segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, 109–120.

- A.S.E. Koster, K.L. Vincken, C.N. de Graaf, O. Zander and M.A. Viergever (1997), "Heuristic linking models in multi-scale image segmentation", *Computer Vision and Image Understanding*, vol. 65, no. 3, 382–402.

Refereed articles in conference proceedings/books

- K.L. Vincken, C.N. de Graaf, A.S.E. Koster, M.A. Viergever, F.J.R. Appelman and G.R. Timmens (1990), "Multiresolution segmentation of 3D images by the hyperstack", in: *First Conference on Visualization in Biomedical Computing, Proceedings VBC '90*, IEEE Computer Society Press, Los Alamitos, CA, 115–122.

- C.N. de Graaf, K.L. Vincken, M.A. Viergever, J.J. Koenderink, F.J.R. Appelman and O Ying Lie (1990), "A Hyperstack for the Segmentation of 3D Images", in: *Information Processing in Medical Imaging*, D.A Ortendahl and J. Llacer (eds.), Wiley-Liss, New York, 399–413.

- K.L. Vincken, A.S.E. Koster and M.A. Viergever (1992), "Probabilistic Multiscale Image Segmentation – set-up and first results", in: *Visualization in Biomedical Computing 1992, Proceedings SPIE 1808*, R.A. Robb (ed.), SPIE press, Bellingham, 63–77.

- C.N. de Graaf, A.S.E. Koster, K.L. Vincken and M.A. Viergever (1992), "A methodology for the validation of image segmentation methods", in: *IEEE Computer-Based Medical Systems*, J.N. Brown, P. Santago (eds.), IEEE Computer Society Press, Los Alamitos, 17–24.

- C.N. de Graaf, A.S.E. Koster, K.L. Vincken and M.A. Viergever (1992), "Task-directed evaluation of image segmentation methods", in: *Proceedings 11th IAPR International Conference on Pattern Recognition, Vol III: Image, Speech, and Signal Analysis*, I.T. Young (ed.), IEEE Computer Society Press, Los Alamitos, 219–222.

- K.L. Vincken, A.S.E. Koster and M.A. Viergever (1994), "Probabilistic hyperstack segmentation of MR brain data", in: *Medical Imaging, Analysis of Multimodality 2D/3D Images, Studies in Health Technology and Informatics*, L. Beolchi and M.H. Kuhn (eds.), vol. 19, IOS Press, Amsterdam, 145–157.

- K.L. Vincken, A.S.E. Koster and M.A. Viergever (1995), "Probabilistic hyperstack segmentation of MR brain data", in: *Computer Vision, Virtual Reality and Robotics in Medicine, Lecture Notes in Computer Science 905, Proceedings CVRMed'95*, N. Ayache (ed.), Springer-Verlag, Berlin, 351–357.

- K.L. Vincken, W.J. Niessen and M.A. Viergever (1996), "Blurring strategies for image segmentation using a multiscale linking model", in: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR'96*, IEEE Computer Society Press, San Francisco, 21–26.

- W.J. Niessen, K.L. Vincken, A.S.E. Koster and M.A. Viergever (1996), "A Comparison of Multiscale Image Representations for Image Segmentation", in: *Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, A.A. Amini and F.L. Bookstein (eds.), San Francisco, 263–272.

- W.J. Niessen, K.L. Vincken, J. Weickert and M.A. Viergever (1998), "Nonlinear Multiscale Representations for Image Segmentation", in: *Computer Vision and Image Understanding*, vol. 66, no. 2, 233–245.

- K.L. Vincken, W.J. Niessen and M.A. Viergever (1998), "Subvoxel Segmentation of MR Images: Necessity and Approach", in: *Proceedings of the International Society for Magnetic Resonance in Medicine*, Sixth scientific meeting and exhibition, in press.

- K.L. Vincken, A.S.E. Koster, O.C. Zander, C.N. de Graaf and M.A. Viergever (1998), "Quantitative Evaluation of Image Segmentation Methods", in: *Proceedings of the International Society for Magnetic Resonance in Medicine*, Sixth scientific meeting and exhibition, in press.

- R. Stokking, K.L. Vincken and M.A. Viergever (1998), "Fully Automatic Brain Segmentation from MRI-T1 Data", in: *Proceedings of the International Society for Magnetic Resonance in Medicine*, Sixth scientific meeting and exhibition, in press.

Other publications

- K.L. Vincken and F.J.R. Appelman (1991), "Accurate conversion of geometrical objects to voxel-based images", Internal 3DCV-report, no. 91-20, Utrecht University Hospital.

- K.L. Vincken, A.S.E. Koster and M.A. Viergever (1993), "PROMISE, a system for probabilistic multiscale image segmentation", in: *Volume Image Processing '93*, M.A. Viergever (ed.), SCVR, Utrecht, 5–8.

- H.G. Schnack, H.E. Hulshoff Pol, W. Staal, K.L. Vincken, W.J. Niessen, M.A. Viergever and R.S. Kahn (1997), "Application of a nearly automatic image scale space linking scheme to segment gray and white matter and intracranial volumes on MR images", *Schizophrenia Research, Proceedings of the VI-th International Congress on Schizophrenia Research*, vol. 24, no. 1/2, Colorado Springs, 155.

Patents

- K.L. Vincken (1994). "Image store with length map", a new multidimensional hashing technique. Described in section 2.2.4.
  Patent number 94203690.6 (Europe/USA).

Samenvatting

Beelden. Wij worden er dagelijks mee geconfronteerd, hangen ze aan de muur en gebruiken ze om onleesbare artikelen te verduidelijken. Beelden (tekeningen, foto's, films) zijn uitstekend geschikt voor communicatie tussen mensen, hetgeen vaak heftige emoties opwekt, ongeacht of het oorlog, kunst of pornografie betreft. Blijkbaar gaat het kijken naar een beeld gepaard met het verwerken van een enorme hoeveelheid informatie. Of, zoals het afgezaagde maar toepasselijke cliché luidt: een beeld zegt meer dan duizend woorden.

Ons visuele systeem is voortdurend bezig met het verwerken, analyseren en interpreteren van visuele prikkels. De natuur, in combinatie met wat wij leren in de eerste maanden van ons leven, biedt ons een uitmuntend systeem om met deze visuele informatie om te gaan. De stelling dat in de nabije toekomst geen enkel door mensen gemaakt autonoom visueel systeem het biologische equivalent zal overtreffen, wordt dan ook onderschreven door veel wetenschappers op dit gebied—inclusief de auteur van dit proefschrift.

Beelden van verschillende modaliteiten hebben hun weg gevonden naar diverse medische disciplines. Multidimensionale beelden zijn onmisbaar geworden voor klinische diagnose, en voor planning en evaluatie van therapie. Artsen hebben nu de mogelijkheid de anatomie van het hele menselijk lichaam te 'bekijken', of de functionele eigenschappen van bepaalde organen en weefsels te onderzoeken. Welke van de beeldmodaliteiten relevant zijn, is onder meer afhankelijk van de aard van de ziekte en de symptomen die een patiënt heeft. Naast de overbekende twee-dimensionale röntgenfoto is eveneens op röntgenstraling gebaseerde Computed Tomography (CT) ontwikkeld, die met name geschikt is om bot te onderscheiden van zachte weefsels. Magnetic Resonance Imaging (MRI) is zeer bruikbaar voor het afbeelden van de hersenen en andere zachte weefsels, terwijl onder meer voor het zichtbaar maken van foetussen Ultrasound Imaging gebruikt kan worden. Digital Subtraction Angiography (DSA), CT Angiography (CTA) en Magnetic Resonance Angiography (MRA) worden toegepast voor het in beeld brengen van bloedvaten. Om functionele eigenschappen van weefsels af te beelden is vanuit de nucleaire geneeskunde Single Photon Emission CT (SPECT) en Positron Emission Tomography (PET) beschikbaar, maar tevens kan de (niet-invasieve) MR techniek worden gebruikt (functionele MRI). Fig. 1 bevat vier voorbeelden van verschillende modaliteiten.

Beelden zoals die door het menselijk visueel systeem worden verwerkt, zijn van

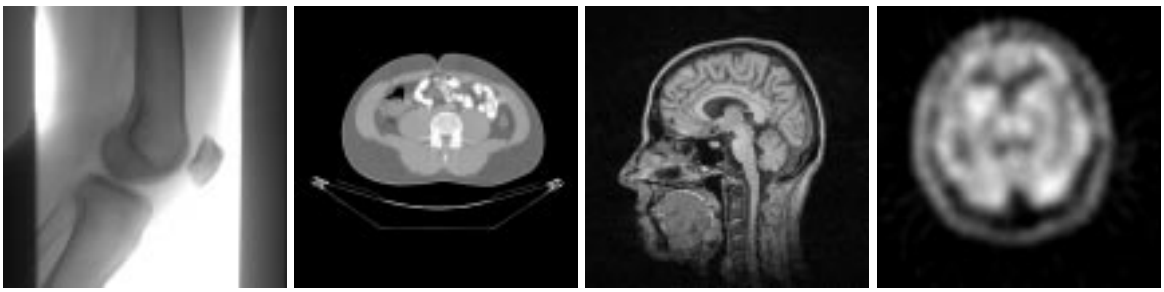


Fig. 1. *Voorbeelden van vier verschillende beeldmodaliteiten. Van links naar rechts: een röntgenfoto van een knie, een* CT *opname van de buik, een* MRI *opname van het hoofd, een* SPECT *beeld van de hersenen.*

een aanmerkelijk hogere resolutie dan de digitale medische beelden die aangehaald worden in dit proefschrift (te weten, variërend van 64×64 pixels (beeldpunten) bij 2D beelden, tot $256 \times 256 \times 256$ pixels in het geval van 3D beelden). Desalniettemin bevatten deze digitale beelden voor artsen een schat aan belangrijke informatie. Doordat de opnametechnieken voortdurend verbeterd worden, neemt de resolutie in alle drie dimensies toe. Een voorbeeld van deze ontwikkeling is de spiraalsgewijze CT opname [48, 45], die snelle drie-dimensionale data-acquisitie van hoge resolutie beelden mogelijk maakt.

De introductie van de *schaalruimte* (*scale space*) theorie door Koenderink [54] en Witkin [131] is een belangrijke doorbraak geweest voor het begrijpen van beelden. Voor het eerst werd het begrip 'schaal' gekoppeld aan een aantal basisconcepten, zoals de kleinste eenheid die in een beeld nog onderscheidbare informatie bevat (de *inner scale*) en de berekening van de afgeleide van een beeld. Koenderink bewees dat een Gaussisch filter van verschillende breedtes het enige filter is dat een (lineaire) schaalruimte onder de voorwaarde van het causaliteitscriterium[1] genereert. Fig. 2 laat het MRI beeld van Fig. 1 zien op vier verschillende schalen. Het bemonsteren van de schaalruimte op een aantal opeenvolgende schalen resulteert in een stapel (*stack*) van beelden. Wanneer het invoerbeeld 3D is, geeft dit een vier-dimensionale stack, oftewel een *hyperstack*. Hierbij is de schaal de vierde dimensie.

De schaalruimte-theorie volgt dezelfde randvoorwaarden als de natuur. Het is dus geen toeval dat het concept van meerschalige beelden lijkt op die van het menselijk visueel systeem: de Gaussische filters en hun afgeleiden komen in hoge mate overeen met de receptieve velden die worden gevormd door de staafjes en kegeltjes in het oog [133, 134, 135, 10]. Koenderink is dan ook uitgegaan van de *isotrope* diffusievergelijking, hetgeen in overeenstemming is met de afwezigheid van informatie in de fase van 'early vision' (d.w.z. dat geen subjectieve kennis over een bekeken scène beschikbaar is). De kracht van het menselijk visueel systeem ligt in het feit dat het 'het observeren van een scène' (zoals het kijken naar een boom) gevolgd kan

[1] Het causaliteitscriterium houdt in dat op grotere schalen geen details mogen onstaan die er op kleinere schalen niet zijn.

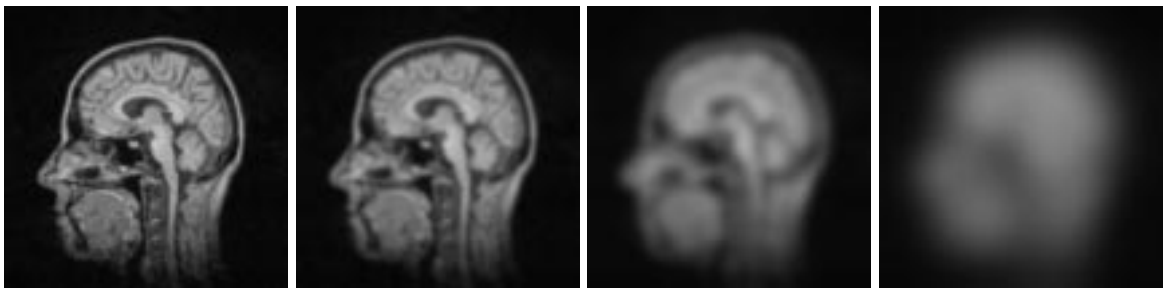


Fig. 2. *De* MRI *opname van het hoofd van Fig. 1 afgebeeld op vier verschillende schalen*

worden door een specifieke, gerichte actie om meer informatie (andere bomen) of meer specifieke informatie (takken, bladeren) te verzamelen. Vergelijk in dit verband het kijken naar een boom en het vestigen van de aandacht op een enkel blaadje. Dit soort acties vereist terugkoppeling van de hersenen (die binnen enkele milliseconden een scène hebben geobserveerd en geanalyseerd) naar het visuele mechanisme dat bepaalt *waarnaar* we kijken. Inderdaad zijn er voor zulke terugkoppelingen fysieke verbindingen gevonden tussen de visuele cortex en de Laterale Geniculate Nucleus (LGN), hetgeen de theorie ondersteunt dat differentiaalstructuren in een beeld (een rand, een hoek) alleen waargenomen kunnen worden *nadat* het beeld is geanalyseerd op meerdere isotrope schaalniveaus.

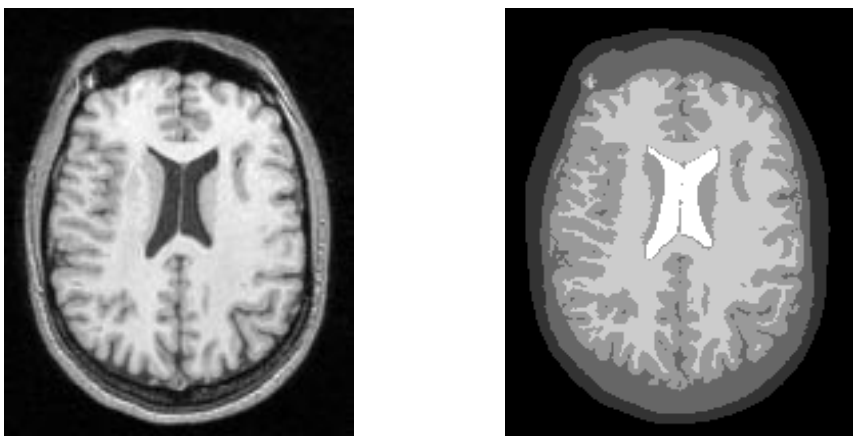


Fig. 3. *Een* MRI *opname van de hersenen (links) en een corresponderende segmentatie in 6 segmenten (rechts). Van binnen naar buiten: ventrikels, witte hersenstof, grijze hersenstof, liquor & bot, bindweefsel & huid, en de achtergrond.*

In de pijplijn van beeldacquisitie naar interpretatie is het *segmenteren* van een beeld één van de belangrijkste en moeilijkste taken. Segmenteren is het groeperen van pixels tot grotere structuren, zodanig dat dit resulteert in een betekenisvolle verdeling van objecten. Dit is geïllustreerd in Fig. 3. Zonder een gesegmenteerde versie van een

beeld is het vrijwel onmogelijk kwantitatieve metingen aan dat beeld te doen (zoals het bepalen van het volume van een tumor), of een drie-dimensionaal aanzicht te genereren (zoals de *volume rendering* van een schedel). Fig. 4 bevat twee voorbeelden van volume renderings gebaseerd op de segmentatie van een 3D beeld. De uitdaging is hier niet zozeer de visualisatie zelf, als wel het segmenteren van de hersenschors. Het doel is dit te realiseren met een snelheid en nauwkeurig die vergelijkbaar is met die waarmee het biologisch visueel systeem van hogere diersoorten deze extreem moeilijke taak uitvoert. Naar onze mening heeft een methode die gebaseerd is op het biologisch visueel systeem—zoals de hyperstack segmentatiemethode—de beste papieren om dit doel te bereiken. De beschikbaarheid van 3D afbeeldingen geeft een computationeel systeem in ieder geval één voordeel ten opzichte van het biologisch visueel systeem: voor deze laatste zijn alleen 2D invoerbeelden beschikbaar, waarvan in de hersenen een 3D voorstelling gemaakt moet worden.

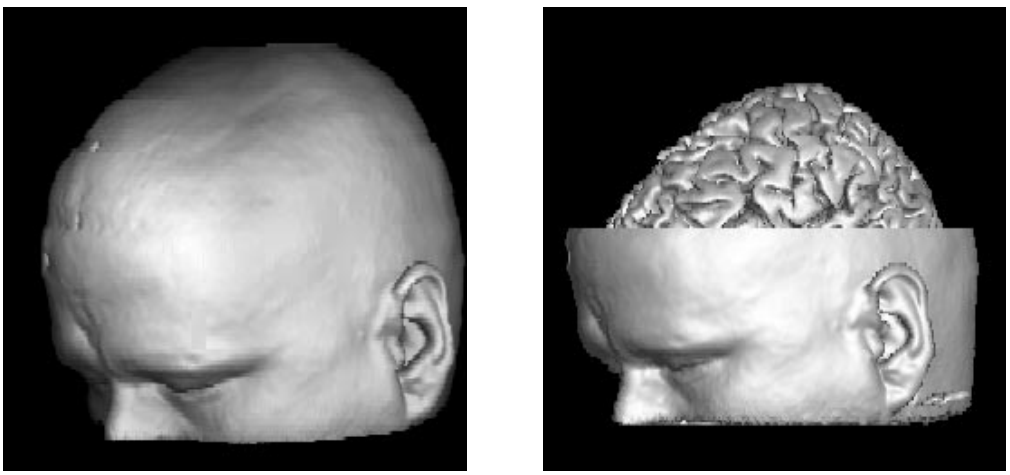


Fig. 4. *Volume rendering van een 3D* MRI *hersenopname (links). Indien het beeld correct gesegmenteerd is kunnen de objecten aan de buitenkant eenvoudig worden 'verwijderd' zodat verborgen objecten zichtbaar worden. In dit geval zijn een deel vande huid en de schedel verwijderd, waardoor de cortex zichtbaar wordt (rechts).*

De meeste conventionele segmentatiemethoden zijn niet gebaseerd op het schaalruimte concept, maar zijn meer lokaal georiënteerd—zoals de rule-based systemen van Ortendahl [80], Menhardt [74, 75], en Raya [91]—of *ad hoc* implementaties voor een specifiek probleem (bijv. Brummer [15]). Schiemann [102] en Pizer [85] hebben zich met name toegelegd op interactieve segmentatiemethoden in tegenstelling tot automatische methoden, terwijl Karssemeijer [49] en Vemuri [115] een multiresolutie aanpak gebruikten die niet gebaseerd is op de diffusievergelijking, maar op respectievelijk het mediaanfilter en de wavelet-transformatie [70]. Tenslotte zijn er voor beeldsegmentatie veelbelovende resultaten geboekt met het gebruik van neurale netwerken (Worth [132]), met name in combinatie met meerschalige technieken (Haring [44]).

In de meeste meerschalige beeldsegmentatiemethodes is er sprake van het verbinden van pixels tussen aangrenzende schaalniveaus (Burt, Pizer, De Graaf [16, 86, 25]). Door de meerschalige benadering kan globale informatie in het beeld effectief worden gebruikt. In dit proefschrift wordt aangetoond dat de hyperstack met ruisige beelden goed presteert en met name sterk is in het tegelijkertijd segmenteren van zowel kleine als grote objecten.

In hoofdstuk 2 wordt de hyperstackmethode uiteengezet. Dit omvat het filteren om de schaalruimte te bemonsteren, en het 'bottom-up' linking proces waarin de schaalniveaus pixelgewijs met elkaar verbonden worden. Dit laatste levert een boom van verbindingen in de schaalruimte op. Vanaf elk knooppunt kan de boom naar beneden worden gevolgd tot aan de pixels in het originele (hoge resolutie) beeld. Deze pixels vormen dan een segment. Het knooppunt waar de projectie begint heet de wortel (*root*) van het segment. Een kritische fase in een hyperstack segmentatie is het bepalen welke knooppunten in de boom door de schaalruimte de beste vertegenwoordigers zijn van segmenten in het originele beeld. Dit is de 'root labeling' fase. Hoofdstuk 2 behandelt tevens ontwerpaspecten van de hyperstackmethode. De datastructuur is geïmplementeerd in de object-georiënteerde programmeertaal C^{++}.

Voor het testen van onze segmentatie-algoritmen is het wenselijk om de beschikking te hebben over (test)plaatjes waarvan de objectverdeling *a priori* bekend is. Hiertoe is een *octree*-achtig algoritme ontwikkeld om objecten die op een wiskundige manier zijn gedefinieerd, om te zetten naar een voxel-model (een *voxel*, afgeleid van *volume element*, is een 3D pixel). De essentie van deze methode is dat de grove discretisatie op voxelniveau verfijnd kan worden door elk voxel dat een deel van de rand van een object representeert op te delen in acht kleinere sub-voxels. Hiermee kunnen de zgn. *partial volume voxels* nauwkeurig weergegeven worden. Het conversie algoritme is op een recursieve manier geïmplementeerd, waardoor de methode een instelbare nauwkeurigheid heeft. De methode wordt in detail beschreven in hoofdstuk 3.

Een groot probleem van de meeste conventionele segmentatiemethoden is dat ze geen rekening houden met het *partial volume effect*. Dit effect—dat veroorzaakt wordt door de beperkte resolutie van het afbeeldingssysteem—uit zich in voxels met een 'tussenwaarde' (de intensiteit van het voxel) van de buurvoxels, die wel volledig in één weefseltype zitten. Normaal gesproken moeten partial volume voxels kiezen tussen deze weefseltypen. Door de isofoten in de directe omgeving van een partial volume voxel te volgen, is het mogelijk meer gedetailleerde informatie van de verschillende weefseltypen (die in dat voxel zitten) te verkrijgen. Het equivalent met de hyperstack is dat het toegestaan is meerdere verbindingen naar de bovenliggende laag aan te leggen voor ieder voxel (in plaats van slechts één zoals in het conventionele linking schema). In hoofdstuk 4 wordt uitgelegd hoe de meerschalige informatie (die de structuur van de isofoten bevat) effectief gebruikt kan worden om partial volume voxels op sub-voxel niveau te segmenteren. Er wordt aangetoond dat dit betere resultaten oplevert met betrekking tot kwantitatieve beeldanalyse (bijv. volume metingen) en dat de volume renderings van 3D beelden aanzienlijk in kwaliteit vooruit gaan.

In hoofdstuk 5 wordt de bemonstering van de gefilterde beelden in de schaalruimte behandeld. Omdat de hoogfrequente informatie verdwijnt als de schaal toeneemt zijn de beelden op de hogere schaalniveaus overbemonsterd. Er wordt aangetoond waarom het rechttoe-rechtaan verminderen van het aantal bemonsteringen, ook wel *sampling rate reduction* genoemd, faalt voor de meerschalige beeldverwerking. In plaats daarvan zou de afname van het aantal bemonsteringen *onafhankelijk* moeten zijn van de afstand tussen twee bemonsteringen (de *sampling width*). Dit leidt tot een methode waarbij de *outer scale*[2] kleiner wordt, ook wel *outer scale reduction* (OSR) genoemd; het netto resultaat is dat de beeldgrenzen kleiner worden bij toenemende schaal. Er worden twee verschillende benaderingen besproken: stricte en heuristische OSR. Het voordeel van het toepassen van OSR is tweeledig: (*i*) de invloed van het randprobleem (vooral aanwezig op grotere schaalniveaus) wordt geminimaliseerd, en (*ii*) de complexiteit van de verbindingsstructuur gaat omlaag. Segmentaties van medische beelden gebaseerd op hyperstacks met OSR laten zien dat de rekensnelheid met een factor 5 opgeschroefd kan worden, zonder dat er een significant verlies in kwaliteit optreedt.

In hoofdstuk 6 tenslotte wordt een onderwerp aangehaald dat een groeiende interesse geniet van de computer vision wereld. Sinds de introductie van lineaire schaalruimte hebben onderzoekers zich gericht op niet-lineaire varianten. Als gevolg hiervan werden er diverse niet-lineaire filters voorgesteld om een schaalruimte mee te bouwen, bijv. door Perona [84], Alvarez [2] en Niessen [78]. We hebben de bruikbaarheid van dergelijke filters vergeleken met de lineaire schaalruimte en twee andere niet-lineaire filters: het mediaanfilter en het Kuwahara filter. De verschillende filter strategieën worden geëvalueerd aan de hand van de hyperstack segmentatiemethode: de filters worden gebruikt om de schaalruimte te creëren (de eerste stap van de hyperstack segmentatiemethode). Op basis van numerieke experimenten wordt aangetoond dat het meerschalige linking model zeer robuust is voor wat betreft de keuze van de onderliggende schaalruimte. De niet-lineaire filters gebaseerd op de Perona & Malik vergelijking en 'curve evolutie' lijken het meest veelbelovend voor onderzoek in de nabije toekomst.

[2]Met *outer scale* wordt de Field Of View (FOV) bedoeld: de totale reikwijdte van een beeld.

Dankwoord

Het schrijven van een proefschrift is in meerdere opzichten een extreme ervaring. Aan de ene kant is het waarschijnlijk het mooiste en meest innoverende dat je ooit gemaakt hebt, aan de andere kant is het een (vrijwillige) maandenlange inbreuk op je privéleven zoals je nog nooit eerder hebt meegemaakt. Bijna niemand kan van te voren in schatten hoe groot de werkdruk in de eindfase zal zijn...

Voor mij persoonlijk heb jij, Pauline, mijn vrouw en vriendin, daar het meest onder geleden. De afgelopen maanden is het meer dan eens voorgekomen dat we 's ochtends afscheid namen, waarna ik je om 01:45 uur de volgende dag weer met een 'hallo, ik ben Koen' begroette, om vervolgens zo diep in slaap te vallen dat ik onze zoon Jelle de hele nacht niet meer zou horen. Die blik van herkenning van jou, lekker ventje van me, maakte overigens wel weer heel veel goed na een dag hard werken ('Agie, aboe'). Bovendien ben je een uitstekend communicatiemedium gebleken voor je ouders ('Je vader is de afwas vergeten te doen, hè Jelle?').

Max, jou ben ik veel dank verschuldigd voor je humor, je geduld, je afkortingen, je makkelijk zijn, en bovenal je openhartigheid. Ondanks het feit dat de subsidieaanvragen altijd eergisteren binnen moesten zijn, stond je altijd voor me klaar (als je er was...) en had je een meer dan positieve houding tegenover mijn 'heb je nog een minuutje?' (antwoord steevast: 'VK?'[1]). Onze gezamenlijke terugreis uit Amerika in 1992, onder begeleiding van Whoopi Goldberg, staat mij (en de stewardessen) nog helder voor de geest. Met meer flessen wijn op vrijdag kan onze komende samenwerking alleen nog maar beter worden. Desnoods sleep ik je auto nóg een keer terug naar Soest (het blijft toch je baas), als ik maar geen adressen-labeltjes meer hoef te plakken met een Pritt-stift.

Ik wil de commissie bedanken voor hun geduld en strakke planning bij het lezen en beoordelen van dit proefschrift. Er is een grote mate van objectiviteit nodig om ieder boekje op zich te beoordelen, zeker gezien de variëteit aan onderwerpen (en kwaliteit) van proefschriften die gelezen moeten worden.

En dan André. Tja, André. Als kamer- en promotiegenoot ben je vooral te herkennen aan het feit dat je 'rechts' en 'links' consequent door elkaar haalt. Bovendien heb je een hele nieuwe dimensie aan het begrip 'homo sapiens' gegeven... Jou ben ik gewoon eigenlijk[2] ontzettend dankbaar voor het feit dat onze boekjes er zo uitzien

[1] Verse Koffie

[2] Op zijn Alfons Saldens.

zoals ze er nu uitzien. Het is dan wel geen op zijn kop gebundelde eenheid geworden, maar ze horen overduidelijk bij elkaar. Ik denk dat we de afgelopen jaren allebei veel profijt hebben gehad van onze samenwerking, niet alleen op segmentatieniveau. Onze discussies betroffen een breed scala aan onderwerpen: van ergernissen over buslijn 12 tot discussies over de Nederlandse taal op niveau[3] en over ingewikkelde koffiebeheersystemen. De laatste maanden konden we goed samenwerken, mits jij je schoenen gewoon aanhield en ik niet zeurde over teveel melk in mijn koffie. En ochtendmensen zijn we allebei niet...

Ger–2 seconden–Timmens, ex-collega en vriend, jij bent een onmisbare steun in de rug gebleken. Je kennis van allerhande grapjes en grolletjes, van LaTeX tot Postscript, van Linux tot vi, van 'Kugelschreiber' tot ons polynoom-programmaatje. Onze nachtenlange sessies hebben zeker vruchten afgeworpen, was het niet voor een student die af moest studeren, dan wel voor onze vriendschap.

En dan Wiro Niessen, de dorpsgek van 3DCV. Na alle hyper-nerds die Max binnen had gehaald, was jij een knappe vent. Je humor staat op het niveau van Maurits–*tadadadadam*–Konings, een prestatie van formaat. Ik dank je vooral voor je vermogen om het serieuze werk (zoals het schrijven van mijn artikelen) te combineren met 'leuk doen', en tussendoor nog even te roddelen over de hele groep. En een langere paranimf kon ik niet vinden.

Verder wil ik Rik–standje-2–Stokking bedanken voor zijn werktijden, zodat ik 's avonds niet altijd alleen (of met Ger) hoefde te werken, en de meiden Carolien Bouma ('this is the animal') en Manon–gammafi(e)t(s)–Kluytmans voor hun niet-mannelijke kijk op het groepsgebeuren. Een verademing!

De beheerders Fred Appelman en Bart–jˆh–Muyzer bedank ik voor het (meestal) draaiende houden van al die computers met een te lang ethernet. Fred, als ex-kamergenoot heb ik veel van (en over) je geleerd, maar ik bedank je vooral voor je levensreddende klap op mijn rug toen ik bijna leek te stikken in een hapje biefstuk. Onze reis naar Madrid was er één om nooit te vergeten! Bartm, bedankt dat ik je altijd mocht bellen, ook al moest het eigenlijk per e-mail.

Sandra Boeijink ('ik ben géén secretaresse!') bedank ik voor alle administratieve rompslomp die wij niet begrijpen en voor de (nu al legendarische) barbecue, en Margo Agterberg voor het aanhoren van mijn file-problematiek en de gratis koffie.

Bart ter Haar Romeny, nooit ben ik iemand tegengekomen die zich meer gedreven in zijn vakgebied (en daarbuiten) beweegt dan jij. Je bent onbetwist een grote stimulus voor velen, met je brede kennis, je vriendelijkheid en je reisverhalen. Lunchen met jou kost een uurtje, maar het is het waard.

Verder loopt (of liep) er nog een variëteit aan mensen rond op 3DCV die ik eigenlijk niet op een hoop mag gooien, maar ik sta al als 'te wollig' bekend, dus vooruit maar. Bas Haring (onze Kunstmatige Intelligent), Karel–O-O–Zuiderveld, Freek–5/6/94-of-6/5/94-?–Beekman, Twan Maintz voor het niet met mij in één bed willen slapen in Brussel en in Nice, Romhild Hoogeveen ('ik k'm uut 't oost'n, 'n da wet'n ze

[3] Voor André: nivo.

nog njet wat uun piex'l ies'), Robert–de-*Pèhp*–Maas, Chris Bakker, Ruud–heup-doet-leven–Geraets, Georg Steinfelder, Evert-Jan Vonken, Frank Bastin, Janita Wilting, Ilse Klinkenberg, Alfons Salden gewoon eigenlijk voor zijn triviale uiteenzettingen, en Erwin van Soest, die als eerste buitenstaander de code van de hyperstack heeft kunnen doorgronden. Proficiat!

Aan de beginperiode heb ik veel goede herinneringen overgehouden. Kees–dat-heb-ik-vijf-jaar-geleden-al-in-FORTRAN-gedaan–de Graaf, Alex–blockout–Hulzebosch, Paul Roos, Petra van den Elsen, Emiel Polman, Analies Schipperheijn, Luc Florack, Anton Koning alias Mr. Unsupp en Sander Wendel, jullie waren prima collega's. Het is jammer dat ik jullie nu nog zo weinig zie.

Marcel Metselaar en Jan de Groot bedankt voor jullie uitstekende service voor wat betreft ons foto- en diamateriaal, Peter Anema bedankt omdat je een aangename, integere kamergenoot was (en voor het snellere scan-werk). Hilleke Hulshoff Pol en Wim Baaré ben ik zeer erkentelijk voor hun introductie van mij bij Psychiatrie. Ik hoop een waardevolle collega voor jullie te kunnen zijn in de nabije toekomst.

Ik wil ook de studenten bedanken, met name Birgit Arkesteijn, Chris Kamphuis, Olaf Zander, Nee Hè en Niet Weer.

I also would like to thank my friend Richard Holloway for knowing you and your family. We don't meet very often, but you are always welcome when you are in Holland; I'm sure we'll meet again some day, wíth our wives and children! A special thanks to Steve Pizer, who always showed a lot of interest in my work during visits. I also want to thank Graham Gash, Tim Cullip, Mark Schulze, Lewis Griffin, and Jorge Llacer for being so helpful, in person or by e-mail. Mads–popcorn–Nielsen, I think I like you, but I don't know why...

Naast het werk heb ik ook veel steun, liefde en gezelligheid mogen ontvangen van mijn naaste familie, schoonfamilie en mijn vrienden. Jullie zorgden voor broodnodige ontspanning en heerlijke vakanties en weekenden. Mijn grote broer Bart ben ik dankbaar voor het 'broer-zijn'.

Tenslotte wil ik mijn ouders bedanken voor alles wat ze in de afgelopen dertig jaar voor mij gedaan hebben. Zonder jullie zou ik zeker niet geworden zijn wat ik nu ben, maar vooral ook hóe ik nu ben. Ik bedank jullie dat ik mij heb kunnen ontplooien, van pianist tot wollige (nét z'n vader) wetenschapper, van zoon tot jonge vader.

Curriculum Vitae

Koen Vincken werd op 22 augustus 1965 in Heemskerk geboren. Zijn middelbare schooltijd bracht hij door op het Pius X College (tegenwoordig het Augustinus College) te Beverwijk, waar hij in 1983 het Gymnasium β diploma behaalde. In datzelfde jaar begon hij met de studie informatica aan de Technische Universiteit te Delft. Zijn afstudeeronderzoek verrichtte hij bij de groep Beeldverwerking (de *Computer Vision Research Group*), gevestigd in het Academisch Ziekenhuis Utrecht (AZU), en betrof het ontwerpen en implementeren van een prototype van de *hyperstack*, een methode waarmee (medische) beelden semi-automatisch gesegmenteerd kunnen worden. De onderzoeksgroep maakt deel uit van de vakgroep Radiodiagnostiek & Nucleaire Geneeskunde van de Faculteit der Geneeskunde aan de Universiteit Utrecht.

Na zijn afstuderen aan de TU Delft bleef hij aan de Universiteit Utrecht verbonden via een aanstelling als Assistent In Opleiding (AIO) per 1 juli 1989. Het onderzoek van zijn promotie betrof het verder ontwikkelen van het prototype van de hyperstack. Dit proefschrift is daar het resultaat van.
In mei 1992 werd zijn AIO aanstelling omgezet in een functie als Toegevoegd Onderzoeker. In die hoedanigheid heeft hij—naast zijn promotie-onderzoek—onder meer meegewerkt aan de ontwikkeling van een nieuwe beeldverwerkingsbibliotheek.

Per 1 oktober 1995 is hij als Toegevoegd Onderzoeker Computational Neuroimaging (in het kader van het zgn. Schizofrenie project) voor 50% verbonden aan de afdeling Psychiatrie van het AZU en voor 50% aan de afdeling Beeldverwerking.
Koen Vincken is getrouwd met Pauline Plantinga en ze hebben sinds 1 juni 1995 een zoon, Jelle.