Morphological and Statistical Techniques for the Analysis of 3D Images

Enric Meinhardt-Llopis

Tesi Doctoral UPF / 2011

Supervisada pel

Dr Vicent Caselles Costa

Departament de Tecnologies de la Informació i les Comunicacions

A l'Arlet

Acknowledgement

The work described in this thesis is the result of the contribution of many people. I want to express my gratitude to them.

First of all, I want to thank my advisor Vicent Caselles, who gave me the opportunity to join his group and carry out research under his supervision. I thank him for his knowledge, patience, and dedication. He did much more than an advisor is expected to do. I will be forever indebted to him.

Many thanks to the members of the reading committee, Jean-Michel Morel, Alejandro Frangi and Ferran Marqués, as well as their substitutes Coloma Ballester and Lluís Garrido, for their time and interest.

Many thanks to Gregory Randall and his team for the warm welcome that I received during my stage in Montevideo. Special thanks to Pablo Cancela: I am very fortunate for everything that I learned from him.

Many thanks to Ernesto Zacur and to Pau Gargallo. These two people have an interesting ability: when I explain my vague ideas to them, they explain these ideas back to me in a form which I can understand. Many ideas on this thesis have gone through such filtering.

Many thanks to Rafael Grompone and to Jérémie Jakubowicz for the interesting discussions we have had every time we have met.

Many thanks to the former and current staff of my Department, specially to Roser Clavero, Judith Champion and Lydia García.

Many thanks to Gabriele Facciolo for being such a nice and helpful colleague. The third part of this thesis is joint work with him.

Many thanks to Juan Cardelino for his help on the implementation of several algorithms.

I want to thank also my former and current colleagues for the nice time that I spent with them: Dani Martí, Vicenç Gómez, Andreas Kaltenbrunner, Jordi Faro, Oscar Civit, Marc Bernot, Tomàs Winand, Leticia Peres, Héctor Palacios, Rodrigo Palma, Rodrigo Verschae, Laura Igual, Gloria Haro, Adrian Marquas, Javier Preciozzi, Pablo Arias, Edoardo Provenzi, Sira Ferradans, Felipe Calderero, Vanel Lazcano, Rida Sadek and Constantinos Constantinopoulos.

Many thanks to my family and to my friends for still calling me from time to time, even if I rarely call them back.

The highest gratitude goes to my partner Cristina, for her unbelievable ability to create a sweet atmosphere of calmness and tranquility around me. This peaceful environment is what allowed me to complete the present work.

Abstract

This thesis proposes a tree data structure to encode the connected components of level sets of 3D images. This data structure is applied as a main tool in several proposed applications: 3D morphological operators, medical image visualization, analysis of color histograms, object tracking in videos and edge detection. Motivated by the problem of edge linking, the thesis contains also an study of anisotropic total variation denoising as a tool for computing anisotropic Cheeger sets. These anisotropic Cheeger sets can be used to find global optima of a class of edge linking functionals. They are also related to some affine invariant descriptors which are used in object recognition, and this relationship is laid out explicitly.

Resum

Aquesta tesi proposa una estructura de dades per emmagatzemar imatges tridimensionals. L'estructura da dades té forma d'arbre i codifica les components connexes dels conjunts de nivell de la imatge. Aquesta estructura és la eina bàsica per moltes aplicacions proposades: operadors morfològics tridimensionals, visualització d'imatges mèdiques, anàlisi d'histogrames de color, seguiment d'objectes en vídeo i detecció de vores. Motivada pel problema de la completació de vores, la tesi conté un estudi de com l'eliminació de soroll mitjançant variació total anisòtropa es pot fer servir per calcular conjunts de Cheeger en mètriques anisòtropes. Aquests conjunts de Cheeger anisòtrops es poden utilitzar per trobar òptims globals d'alguns funcionals per completar vores. També estan relacionats amb certs invariants afins que s'utilitzen en reconeixement d'objectes, i en la tesi s'explicita aquesta relació.

Preface

Two-dimensional digital images are typically color or gray-level photographs, coming from scanners or digital cameras. The pixels on these images have essentially the same meaning: they measure the amount of light that a captor receives from each direction among a fixed cone of possible directions. Thus, most 2D images are ultimately models of what happens in a retina, and a large part of image processing is built upon this assumption. This assumption can be summarized by saying that the two axes of the image domain have units of distance.

The situation for three-dimensional digital images is quite different, for they can come from a variety of sources. They can be, for example, tomographies, color histograms, video sequences, multispectral or multi-focus 2D images, graphs of gray images, or restricted versions of higher dimensional light fields. Each of these, essentially different, cases of 3D images can be summarized by the units of their three axes. For example, medical images have three spatial dimensions, videos have two spatial and one temporal direction, multispectral images have two spatial and one frequential direction, color histograms have three frequential directions, graphs of gray images (as used in bilateral filtering) have two spatial and one intensity dimension.

Since there are more types of images than in 2D, the analysis of 3D images requires a larger variety of techniques, some of them applicable only to a specific type. Besides this variety of sources, there exist intrinsic difficulties of 3D images, coming from topological properties that differ between the dimensions. Many nice structures of the plane do not transport well to higher dimensions. For example, there is only one notion of *hole* for subsets of the plane, but subsets of the space may have holes and *handles*. Complex numbers do not have a three-dimensional analogue. The Euler-Poincaré characteristic can be computed locally in two dimensions but not in three. This lack of nice properties may result in some further difficulties.

On the other hand, three-dimensional images are in some cases easier. For example, occlusions are a typical circumstance of 2D images, leading to problems when one object occludes another object of the same intensity. In 3D medical images there are no such occlusions: the value of the image at a point is the density of some physical quantity at a corresponding point in space. Thus the objects are, at least in principle, easier to separate by their color (however, 3D medical images tend to be of lower resolution and noisier than 2D photographs).

The work leading to this thesis started as a 3D implementation of a 2D data structure: the Tree of Shapes. It was followed by the generalization of many algorithms using the Tree of Shapes in 2D to the 3D case. Along the way, many of the difficulties outlined above were found and dealt with. One of these difficulties, the linking of partial edges, led to a variational formulation of edge linking whose global optimum gave reasonable results. An unexpected connection with affine invariants used for object recognition was found, and this concluded the development.

Contents

1 Outline

This thesis is divided into three parts.

The **first part** deals mainly with topology, algorithms and data structures for image representation. The tone of this part is formal: we define structures, prove results about them, and propose algorithms to compute them.

What is the technological interest of these constructions? Well, fancy data structures to store digital images are as old as image processing itself. Each data structure provides a representation of the image which is well suited to certain operations on the image. For example, the representation of an image by its Fourier transform is well suited to the application of linear filters, or to the analysis of its smoothness; the representation by a wavelet packet basis lends itself to image approximation or compression. The representation of an image by its tree of shapes is well suited to the application of some morphological operators, to segmentation and to edge detection. Moreover, in the 3D case it provides a great aid to visualization. The tree of shapes of 2D images is a well-known data structure which, since its introduction 10 years ago, and with different names coming from independent rediscoveries, has been applied to many different problems in image processing. The first contribution of this thesis is to extend the algorithm to compute the 2D tree of shapes to 3D images. For that, we have to introduce a new description of the tree of shapes as the fusion of the trees of upper and lower level sets of a function. It turns out that, if the function does not oscillate wildly, this fusion is a combinatorial re-arrangement of some branches of the trees of upper and lower level sets. The detailed explanation of this fusion is the second contribution of this part.

Finally, at the end of this part, we show some simple applications of the tree of shapes: morphological filtering, color histogram analysis, image visualization, and video analysis.

Thus, there are two non-entirely-trivial contributions on the first part: an explanation of some combinatorial properties of the collection of level sets of a function, and an algorithm to compute a data structure to store 3D images, which is based on these combinatorial properties. As an aside, we explain how this data structure is useful for color histogram visualization and object tracking in videos.

The **second part** introduces a new method of edge detection. The tone of this part is heuristic: we define a set of tools that are finally used to build a complex machinery, which just happens to work.

Thus, the second part is focused on a more complex application of the three-dimensional tree of shapes: an edge detector for 3D images. This detector is analogous to a well-known edge detector for 2D images. The main idea of this method is to traverse the level surfaces of the image one by one, and to select the best contrasted parts of each surface, if any. To decide whether a part of a surface is well-contrasted or not, we use a statistical test that compares its contrast with the overall contrast of the image. This statistical test is commonly used in computational gestalt theory, under the name of "Helmholtz principle",

where it is presented as a general method for finding structures without any a-priori model of them (only from a model of noise).

As in the first part, here the main contributions are related to the passage from 2D to 3D images. While the edges of a planar image are pieces of curve, the edges of a volumetric image are pieces of surface, which are more delicate to manage. In the 2D case, all connected pieces of level curves were considered as possible candidates for edges. In our adaptation to 3D, we restrict the space of candidate edges to a only a small class of connected pieces of level surfaces. Namely, to those pieces that arise as segments of a piecewise constant segmentation of the Mumford-Shah functional of a contrast function defined on each level surface. The rationale for this choice is given in detail, but the basic idea is that the desired edges, besides being well-contrasted, must be pieces of surface with a smooth boundary. At this point, the thesis enters into a digression on triangulated surfaces; because we have to explain how to find topologically consistent and precise level surfaces, how to do integral-geometric computations on them, and how to build the Mumford-Shah model on them. Once we have all the necessary constructions on triangulated surfaces, the edge detector itself can be described briefly as a combination of these constructions. At the end of this part, the proposed edge detector is analyzed in detail, both theoretically and practically (by running it on some sample synthetic and real images).

The main contribution of this part is the definition of the edge detector for 3D images (and the description of the path that leads to it).

The **third part** deals mainly with analysis, numerical approximation, and computer vision. The tone of this part is jovial: motivated by the problem of edge linking, we explain how a number of apparently different objects are closely related, and we obtain pleasure from these findings.

This part starts by recalling the relationship between Cheeger sets (subsets of a domain which minimize the perimeter/area ratio) and total variation (a semi-norm on the space of functions). Then we transport this relationship to the context of Finsler manifolds, where Finsler-Cheeger sets are closely related to Finsler total variation. We explain how Finsler-Cheeger sets are found as level sets of stable solutions of a PDE for minimizing Finsler total variation. Then, we explain how to use this fact to devise a numerical scheme for computing Finsler-Cheeger sets. This scheme has two parts: first solve a PDE, and then find the appropriate level surfaces of the solution. At this point, for organizing the search of the appropriate level surface, the tree of shapes comes handy, and it is also used to define various notions of *local* Finsler-Cheeger sets. The adaptation of all these methods from Euclidean space to Finsler manifolds is the first contribution of this part.

Finsler metrics have applications in image processing and in crystallography. In image processing, Finsler metrics are the most general setting for anisotropic diffusion and related techniques. For example, we show that the geodesic active contour model (with an inflating force) produces surfaces which are Finsler-Cheeger sets with an appropriate metric. It turns out that there is a different, apparently unrelated, usage of Finsler-Cheeger sets in image processing, more

concretely in computer vision: the affine invariant descriptors MSER (Maximally Stable Extremal Regions) are a kind of local Finsler-Cheeger sets of the image domain with an appropriate metric. On the last chapter of the thesis we explore this correspondence. After recalling the original definition of MSER, we propose a slight change in the definition, that joins the MSER coming from upper or lower level sets of the image. This change consists in using the tree of shapes instead of the trees of upper or lower level sets. Then, we explain that MSER are precisely the Finsler-Cheeger sets of the image domain with an appropriate metric. Putting MSER into this context clarifies the interpretation of affine invariance and general robustness of MSER and MSER-like descriptors. Thus, the second contribution of this part consists in putting under the same context objects as disparate as: anisotropic total variation, Finsler-Cheeger sets, geodesic active contours, trees of shapes and affine-invariant image descriptors.

Thus, this part presents two contributions. The first contribution is the theory and algorithms for computing local Finsler-Cheeger sets. The second contribution is the application of Finsler-Cheeger sets to the study of affine invariant image descriptors such as MSER.

Part I

The Three Dimensional Tree of Shapes

2 Historical context of trees to represent images

In the image processing literature, several tree-like structures have been used to encode the complexity of images. Here we describe some of them and outline their main uses.

2.1 Trees in Mathematics

In mathematical image processing, grayscale images are often modelled by real-valued functions on a rectangle. Real-valued functions can be very complicated objects. When they are smooth enough, the structure of their singularities can be used to describe that complexity. We begin this report with a historical account of this geometric theory of real-valued functions.

Cayley, Maxwell, and Morse Theory

Although the nature of critical points of smooth functions on the plane was most surely understood since ancient times, the first person to write it down in a modern manner seems to be Arthur Cayley. In 1859 he wrote a memoir [Cay59] titled *On Contour and Slope Lines* where, in the language of topography, he defines the objects of the theory, namely, contour lines and critical points. He notes that in general one can assume that critical points are isolated and happen at different levels, and that there are three types of critical point: maxima, minima, and saddles (he calls them "summits", "immits" and "knots" respectively). His two results state that around a critical point the nonsingular level lines look like a conic (ellipses for extrema, and hyperbolas for saddles), and that there is a relation between the number of maxima, minima, and saddle points.

An independent memoir [Max70], but with similar contents, was presented in 1870 by J. Clerk Maxwell titled *On Hills and Dales*. This memoir still uses the language of topography, for example, defining level sets as those parts of the terrain that would be covered by water as the oceans raised. This is a device to make the exposition more vivid, because the content is purely mathematical. The work of Maxwell improves that of Cayley: he considers arbitrarily degenerate critical points, like multiple saddle points, and discusses also three dimensional functions (described as potentials in space) and their critical points. For our purposes, the most interesting feature of this paper is an unreferenced figure (see fig.2.1) that shows how the lines of slope starting from the saddle points reach the extrema, dividing the terrain in natural districts. While this is not a global hierarchy like a tree, those slope lines indicate which saddles join and split which level sets.

In the twentieth century these ideas of Cayley and Maxwell were refined and generalized to arbitrary manifolds in what is known as Morse theory. Morse theory starts with the observation that for any smooth function on an n-dimensional manifold, around every non-degenerate critical point, one can find a local chart

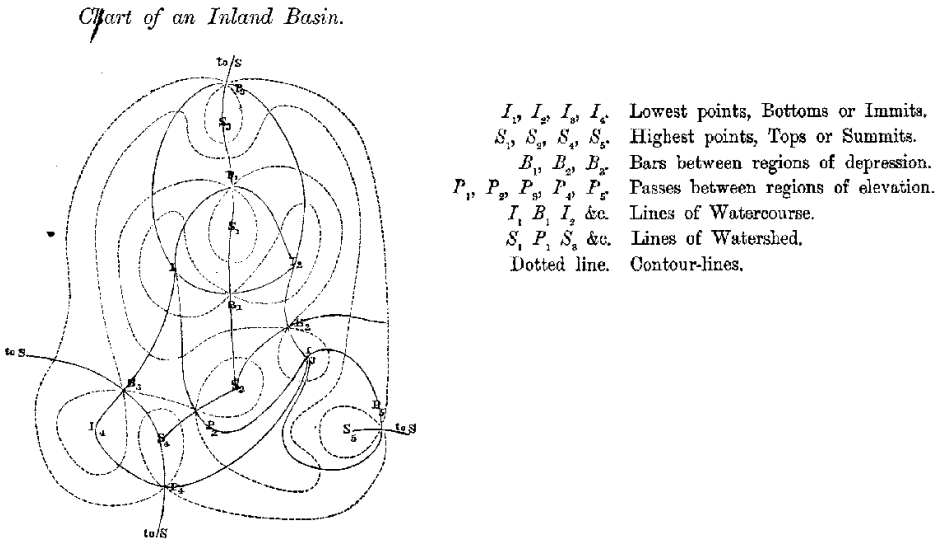


Figure 2.1: Maxwell's figure showing the graph of principal slope lines on a terrain

such that the function is a quadratic polynomial of the form

$$f(x) = c - x_1^2 - \cdots - x_\lambda^2 + x_{\lambda+1}^2 + \cdots + x_n^2.$$

The number λ is called the *index* of the critical point, and is invariant under the choice of local chart. Thus, maxima have index n and minima have index 0, the saddle points having indices in-between. Then, in two dimensions there is only one kind of saddle point, in three dimensions there are two kinds, and so on.

The basic idea of Morse theory is that the topology of a manifold can be understood using only smooth, non-degenerate functions on that manifold. For instance, if we take a single function on a manifold, which has b_i critical points of index i, then

$$\sum_{i=0}^{n} (-1)^i b_i = \chi,$$

where χ is the Euler characteristic of the manifold. In the case of the sphere, $\chi = 2$, and we recover the relations given by Cayley and Maxwell.

Morse theory is a widely used tool in mathematics. Typically, one studies complicated manifolds with the aid of easy functions on them (like heights or distances). In image processing, on the contrary, we have complicated functions on easy manifolds (rectangles), but the same ideas apply. A similar situation is found when studying Riemannian metrics on images (see Figure 10.1 on Part III).

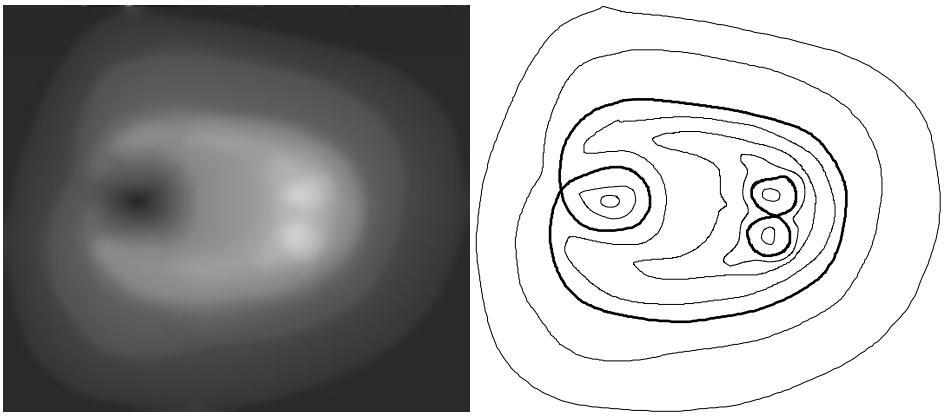


Figure 2.2: A smooth image and some of its level lines

Reeb Graph

The Reeb graph is a one-dimensional topological space that summarizes the topological changes of the level sets of a function. It was defined in 1946 by Georges Reeb in a small note [Ree46]. It is the basis of some image representations as trees. We refer to Reeb's note for the details of its construction, and now we explain briefly how can it be used for images.

For regular enough functions, most of the trees we will treat here can be described as quotients of topological spaces in the following way. Let Ω be a manifold and let $f : \Omega \to \mathbb{R}$ be a smooth function with isolated critical points. The *Reeb graph* of f is defined as the topological quotient of the space Ω by the relation

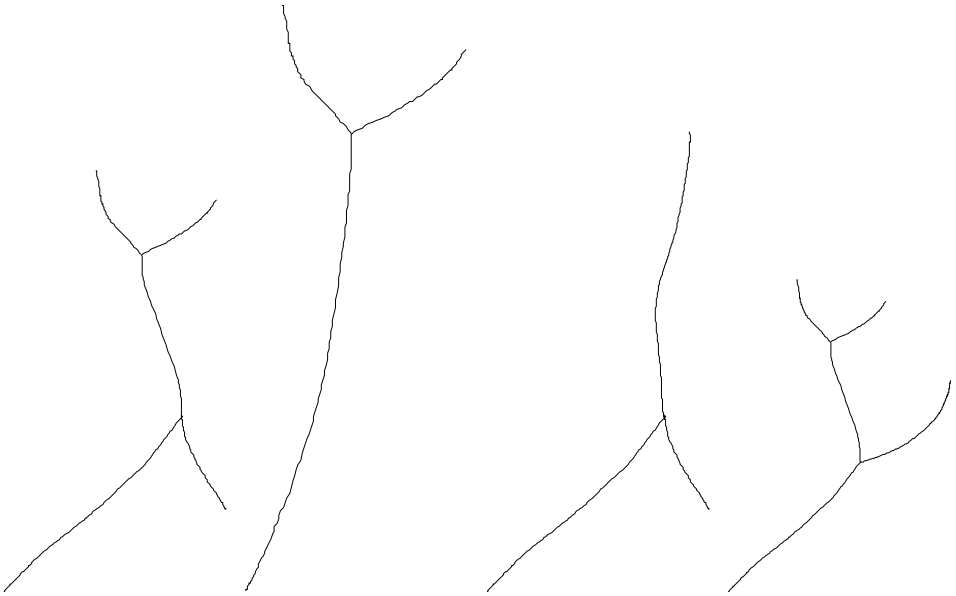
$$x \sim y \quad \Longleftrightarrow \quad f(x) = f(y) \text{ and } cc([f = f(x)], x) = cc([f = f(y)], y).$$

The Reeb graph is a tree, in the sense that as a topological space it does not have any subspace homeomorphic to S^1. The *upper Morse tree* is defined as the quotient of Ω by the relation

$$x \sim y \quad \Longleftrightarrow \quad f(x) = f(y) \text{ and } cc([f \geq f(x)], x) = cc([f \geq f(y)], y).$$

The class of a point where the function reaches its absolute minimum is selected as the root of the upper Morse tree. The lower Morse tree is defined similarly, using lower-level sets. The *tree of shapes* is the same as the Reeb graph, but with one arbitrarily selected point as the root. Notice that when the space Ω is simply connected, the Reeb graph has no loops and it is indeed a tree. The advantage of using a topological quotient is that the resulting tree inherits a topological structure in a natural way, thus we can talk about neighborhoods of level sets being formed by other level sets.

In the "non-degenerate" case, this is the whole story. However, in the computer implementation of these ideas, one faces two problems: 1) the function and



(a) Contour tree (b) Upper Morse tree (c) Lower Morse tree (d) Tree of shapes

Figure 2.3: Trees corresponding to the image in figure 2.2

the space are discrete, and 2) the function is usually degenerate. Thus, one has to adapt these definitions to deal with discrete spaces and degenerate functions. The different trees appearing in the commented literature can be understood as different adaptations to these new scenario of the topological quotients defined above.

Kronrod's Tree

Independently of Reeb's work, and about the same time, Alexander Kronrod defined a similar tree for two-dimensional functions. Kronrod was interested in extending the rich theory of functions of a real variable to functions of two real variables, and used the tree as a tool. His work [Kro50] on functions of two variables is not available in English. This is unfortunate because, as the following excerpt from his bibliography [LY02] suggests, it must be extremely interesting:

> *The linear variation was basically a new object. Kronrod introduced the concept of a monotone function of two variables, a natural generalization of the corresponding concept for a function of a single variable. He proved that the boundedness of the linear variation permits the function to be represented as a difference of two monotone functions. For the linear variation itself, he gave a number of equivalent definitions, of which one is of particular interest. It turns out*

that with a continuous function of two variables one can associate a one-dimensional tree, the elements of which are the components of the level-sets of the function. On them, with the help of the original function, a metric can be defined and, on the tree, a function. The linear variation then turns out to be equal to the usual variation of the function defined on a one-dimensional tree. The boundedness of both the planar and linear variation guarantees the existence almost everywhere of the usual total differential.

The work of Kronrod is rarely referenced in image processing papers, even in those which use constructions like the contour tree. Exceptions include [BCM03], [AC03], [AC04].

2.2 Trees in Image Processing

Since images have been represented as rectangular arrays of pixels, people has noticed that this representation has one major drawback: a pixel is too local a unit of information. Of course, pixels are the simplest representation for images, and lots of interesting operations can be applied directly to pixel-based images. However, some hierarchical representations have been devised to ease tasks like as segmentation and information retrieval. In this section we comment five such representations and their applications. Some of them are the direct application to image processing of the ideas of Reeb and Kronrod, as we saw in the previous section.

Quadtrees and Octtrees

Quadtrees [RK82] were designed to represent binary images with large constant regions. The idea is simple: take a square image, if it is constant then the representation is trivial, otherwise, divide it in four equal sub-squares and represent each one of them recursively. When the image has large constant regions (for instance, when it is sparse in one color), this is a very efficient representation. Note that the smaller squares follow the edges of the image, but the edges themselves are not readily available from the representation.

Nowadays, quadtrees are used in image processing mainly as a storage structure for multi-scale access to very large images. However, the same idea of a recursive subdivision of the space is central in the wavelet representation. There, one decomposes the image into a low-pass component, and three high-pass components representing horizontal, vertical, and diagonal edges. The low-pass version is then represented recursively using the same method. This representation is the basis of one of the best image compression algorithms today, JPEG2000.

The wavelet compression gives impressive results. This is because most of the information it contains is located at the edges of the image. As in quadtrees, however, the edges themselves are not readily available from the representation. This is not important at all when the purpose is compression. But if the purpose

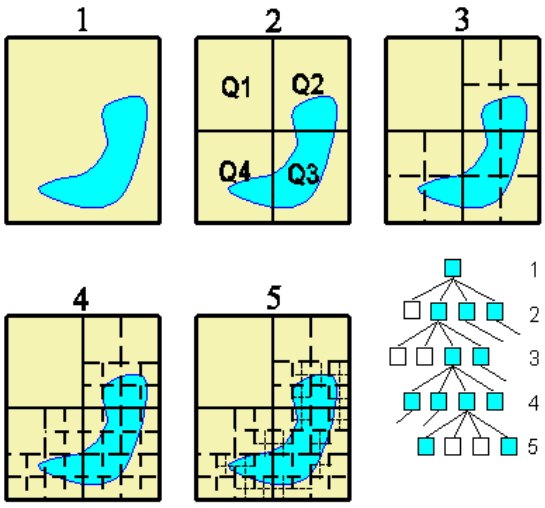


Figure 2.4: First steps in the construction of a quadtree (image taken from [USS00]).

is, for instance, edge extraction, this wavelet representation does not help very much, at least in a direct way.

Octtrees are the 3D version of quadtrees, where a cube is recursively divided into 8 smaller cubes. They are an essential tool for processing medical images interactively, because the multi-scale representation allows an arbitrary speed-up in many operations, at the price of a reduced resolution, which is typically acceptable.

Contour tree

In the search of a representation adapted to the geometry of images, the first idea is to consider isolevel sets: according to mathematical morphology, the edges of an image are often parts of its level lines. Thus, a good starting point is to use data structures based on the set of *all* the level lines of an image. The Reeb graph is such a structure, and when used in image processing, specially for three-dimensional images, it is called the contour tree. The fact that the Reeb graph of an image is, indeed, a tree comes from theorem 6 on Reeb's paper.

The adaptation of critical point theory to a discrete environment was done by Banchoff in 1967 [Ban67]. He built the discrete theory using polyhedra instead of manifolds, and linear functions instead of smooth functions. Then, non-degenerate functions are those whose gradient is not parallel to any edge of the polyhedron.

A slightly different construction was introduced by van Kreveld *et al.* [vvB+97]. These authors wanted a data structure that eased access to the

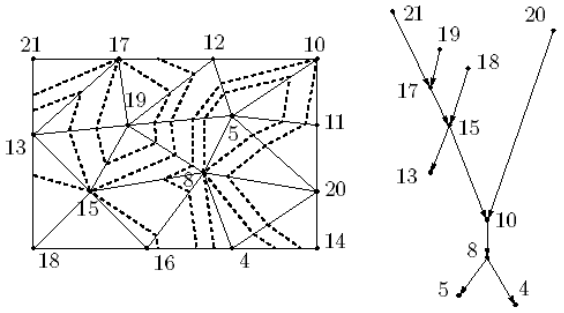


Figure 2.5: Contour tree of a two-dimensional mesh (image taken from [vvB+97])

set of isosurfaces, or contours, of their data. The data was defined on the vertices of a simplicial mesh, which is the same as a flat polyhedron. Then they considered the data interpolated linearly inside each simplex. This gave a continuous function over the whole space, whose isolevel sets could be defined, see figure 2.5. The contour tree is the Reeb graph of this continuous function. The whole construction relies on the hypothesis that the data is "unique", meaning that all vertices of the mesh have different values.

Since the introduction of the contour tree in image processing, several improved algorithms have been presented to compute it [vvB+97], [CSA03], [TV98], [PCM03], [CLLR05]. To summarize the results: there are algorithms to compute contour trees in any number of dimensions with cost $O(n \log n)$, where n is the number of vertices in the mesh. The reference [CLLR05] contains lots of other references and a more detailed history of the algorithms to compute contour trees. All these algorithms rely on the hypothesis of data unicity. While this may be acceptable for functions defined on meshes, it is clearly not true for quantized pixel based-images. One of the first references [CSA03] suggests the use of *simulation of simplicity* [EM90] to overcome that difficulty, but this does not seem to be done anywhere. To-day, the most common workaround is to solve ties of equal-valued vertices in order of their appearance, but this yields trees whose topology depends on arbitrary choices.

Tree of criticalities

Under the name of Digital Morse Theory, Cox and Karron [CKF03] define and propose a method to compute the upper Morse tree of arbitrary 2D and 3D images. The advantage of their construction is that it is based directly on images of pixels, and that it allows for degeneracies in data. For instance, they can treat large regions of equal-valued pixels. Furthermore, as the upper Morse tree is rooted, this organization provides a hierarchical subdivision of the image.

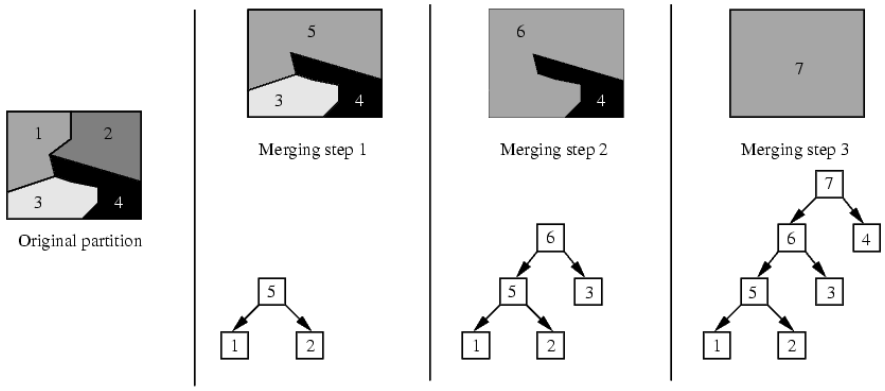


Figure 2.6: Creation of a binary partition tree (image taken from [SG00])

Binary tree

A completely different representation of images by trees is that of Garrido and Salembier [SG00]. Their tree represents how regions in the image were joined according to some joining strategy, coming from a similarity measure among regions; see figure 2.6 to understand the idea. The novelty is that the choice of regions is flexible: the joining strategy is in fact a parameter of the algorithm. They propose several joining strategies, for instance joining the pair of adjacent regions that have the most similar color. Other criteria might be homogeneity (that is, similarity of histograms of both regions) or motion (were the images are frames of a video sequence).

This is a just like the algorithms of region-merging, but where the whole history of the mergings appears in the data structure. The main advantage of this flexible approach is that it can give better results for highly textured images, where the level-lines are too complicated and do not follow the borders of objects.

Mumford-Shah-Koepfler-López-Morel Tree

On [KLM94], the authors introduced a merging criterion to approximate minima of a piece-wise constant version of the Mumford-Shah functional [MS88]. While their emphasis was not on the data structure itself, this work contains the idea of using a hierarchical segmentation of an image that can be pruned later to obtain real segmentations. This hierarchical segmentation can be regarded as multi-scale representation of the image. In some sense, this tree is a particular case of the binary tree above, but discovered earlier, and using a very particular merging criterion that comes from a global segmentation functional.

Monotonic tree

A seemingly independent rediscovery of the tree of shapes was presented in 2002 by Y. Song and A. Zhang [SZ02]. They named it the *monotonic tree*, and presented their construction as an improvement of the contour tree aimed at discrete images. The following quote from that paper explains clearly the different philosophy of contour trees and trees of shapes (or monotonic trees):

> *Contours are only defined for continuous functions. For an image represented by discrete data, a continuous function is first defined as an interpolation of the data. Then the contour tree is defined on this continuous function. In this paper, we introduce a new concept termed monotonic tree, which is directly defined on discrete data.*

Monotonic trees have been used successfully in some applications. Perhaps the most attractive one is an algorithm [SZ03] to segment scenery images with labels such as "sky", "grass", "water", "tree", "building". The algorithm starts computing the monotonic tree, and then joins contiguous regions according to the similarity of their histograms to sample textures in a database. This technique seems to be adaptable to more general segmentation problems.

Trees in mean shift and bilateral filtering

Given a 2D gray-level image, its pixels together with their values can be regarded as points in 3D. Thus, the image is associated in a natural way to a cloud of points. This cloud of points can be associated to a continuous density (e.g., by replacing each point measure by a Gaussian). This density is at the heart of many image processing operations, such as Mean Shift or Bilateral Filtering [Bar02]. The tree of upper level sets of this density, or the associated trees that result by increasing the variances of the Gaussians, are natural objects in image processing that give rise to hierarchical and multi-scale segmentation. They are closely related to the component tree, which is a common tool in statistics for clustering high-dimensional datasets [Wis69, NC06]

2.3 Some uses of the tree of shapes

The tree of shapes has been used successfully in several applications of 2D image processing. The three reasons of this utility seem to be the following:

- It is a representation of images covariant under interesting transformations of the image, such as local contrast changes, and continuous deformations of the domain.
- It allows a quick access to level curves, either in an ordered manner, or starting from a pixel.

- It provides a hierarchical partition of the image into nested regions, and allows the efficient computation and storage of regional measures such as means or histograms.

Now we list some of these applications.

Implementation of morphological operators Some of the most interesting morphological operators can be understood as operators acting on the tree of shapes. Then, if images are represented using their trees, the application of such operators is immediate. Grain filters, for instance, consist in pruning the shapes of the tree with small area. See [CM02] for grain filters and [VKM07] the general theory of morphological operations on trees.

Image Registration In his PhD thesis, Monasse uses the tree of shapes of two images to register them. The registration is done using the largest shapes, while the smallest shapes are allowed to be unregistered. This is useful if the images to be registered represent an object with moving parts [Mon00].

Compression The tree of shapes has been used to compress digital elevation maps [SCSA04, Igu06]. If you remove some level lines of a function, you can recover an approximation to it. If you remove the maximal number of level lines such that the interpolation is close enough to the original function, you get an approximation that can be used for compression purposes.

Edge detection Desolneux et al. [DMM01] proposed a global criterion to select the level lines in an image which are perceptually significant. They call them "meaningful boundaries". Thanks to the tree of shapes, the computation of meaningful boundaries can be done in linear time.

Segmentation An algorithm for global image segmentation based on the tree of shapes has been introduced by Cardelino et al. [CRBC07].

Shape Identification The problem of object matching can be approached by matching of level lines. By computing suitable descriptors of all level lines of an image (or a meaningful subset of them), different images can be compared. Since level lines are robust to contrast changes, this provides a reliable method for shape identification [CLM$^+$08].

Texture Analysis Any descriptor of planar shapes gives rise to texture descriptors via the three of shapes. To describe a texture, first describe each of its shapes and then summarize the results into some kind of histogram. This texture analysis has been successfully used for detecting the orientation of terrain textures in satellite images [XDG09].

Self-dual MSER Maximally Stable Extremal Regions (MSER) [MCUP04] are ubiquitous features for stereo matching, which intuitively are the most salient light blobs on the image. They are formally defined in terms of the tree of upper level sets, but their definition can be straightforwardly adapted to the tree of shapes (see Section 14.2). The advantage of using

the tree of shapes is that it allows light and dark blobs to be processed simultaneously.

Optical Flow Validation Due to their invariance to contrast changes, the level lines are a useful feature for estimating or validating movements between two image which may have suffered a change in illumination. In that setting, the tree of shapes is the natural tool [CGI05].

3 Tree of Shapes: The Theory

The goal of this chapter is to define the tree of shapes as a mathematical concept and explain its main theoretical properties. The chapter is organized into three sections. Section 3.1 introduces all the mathematical definitions which are needed for the construction of the tree of shapes. Section 3.2 briefly defines the tree of shapes using all the previous machinery. Section 3.3 shows how the structure of the tree of shapes can be obtained from the structure of the trees of upper and lower level sets.

3.1 Mathematical Preliminaries

Besides the standard definitions of functions, real numbers and topological spaces, the mathematical preliminaries for this chapter are the following: unicoherent spaces, as a model for the image domains; semicontinuous functions as a model for general images; and weakly oscillating functions, as a model for well-behaved images.

These are mostly technical conditions which can be omitted on a first reading. It is safe to ignore these definitions and skip to Section 3.2. Unicoherent spaces are needed to assure that the external boundary of any connected region is connected. Semicontinuous functions are the minimum requirement for the definition of the tree of shapes. Weakly oscillating functions are needed so that the structure of the tree of shapes is finite.

We also introduce an appropriate language to deal with certain families of subsets of the image domain, namely trees of subsets.

3.1.1 Unicoherent Spaces

Let X be a topological space. As the space X will be used to model the domain of an image (typically, a rectangle), we can require some technical simplifications. In particular, we assume that X is compact and that it is unicoherent.

Definition 1. *A topological space X is said to be* unicoherent *when it is connected and for any closed and connected subsets A, B the following property holds:*

$$A \cup B = X \quad \Longrightarrow \quad A \cap B \text{ is connected.}$$

The notion of unicoherence is closely related to that of simple connectedness, that is, that closed loops are contractible. This condition is needed so that the outer boundary of any region is connected. See figure 3.1 for an illustration.

Taking complements on the definition of unicoherence, we obtain the following property: if A and B are disjoint open sets such that $\Omega \backslash A$ and $\Omega \backslash B$ are connected, then $\Omega \setminus (A \cup B)$ is also connected. Iterating this union to a finite collection of open subsets we obtain the following lemma, which is the form of unicoherence which is most useful in proofs:

Lemma 2. *Let $X_1, X_2, \ldots, X_n$ be disjoint open sets such that for any i, $\Omega \setminus X_i$ is connected. Then $\Omega \setminus \bigcup_i X_i$ is also connected.*

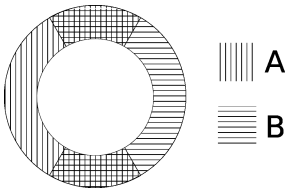


Figure 3.1: An annulus is not unicoherent. Note that in can be covered by two connected sets whose intersection is not connected. This implies that the external boundary of a connected region inside the annulus can have multiple connected components.

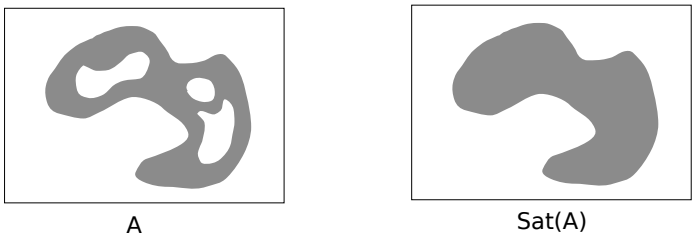


Figure 3.2: A region of the plane and its saturation. The analogous figure for three-dimensional regions is difficult to draw because the holes would be occluded by the external border. However, the definition and the ideas are exactly the same.

Connected components of sets play an important role in our discussion. We introduce the following notation for them. Let A be an open or closed subset of X and $p \in X$.

- $\mathrm{cc}(A)$ is the set of connected components of A within X.
- $\mathrm{cc}(A, p)$ is the connected component of A that contains p, if there is any, and the empty set otherwise.

The tree of shapes is defined in such a way that none of its objects has any hole. In fact, its objects are precisely the objects from the upper and lower trees, but with their holes removed (or, more precisely, covered). To formalize the notion of hole we use the *saturation operator*, see figure 3.2.

The definition of saturation needs the notion of *hole*.

Definition 3. *Let X be a topological space as before, and let $A \subseteq X$. The* holes *of A are the connected components of $X \setminus A$.*

Now we need to differentiate between the *internal* holes of a region A and its *external* hole. In a region such as that of figure 3.2, it is clear that the external hole is that which touches the border of the image. However, for general regions like that of figure 3.3, it is not so clear which is the external hole. In

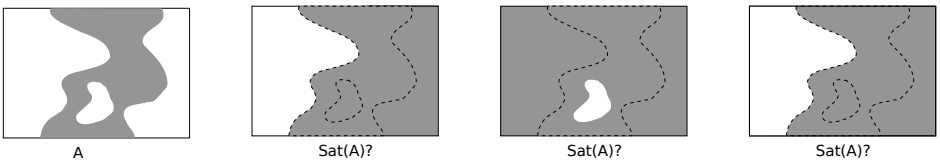


Figure 3.3: A region that touches the border of the image. What has to be its external hole?

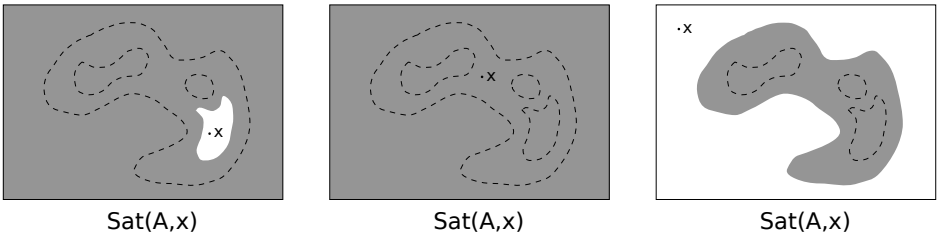


Figure 3.4: A region of the plane and its saturation with respect to some different points.

his PhD thesis [Mon00], Pascal Monasse suggests that the external hole is to be the one that touches the largest part of the border. But this definition is a bit expensive to generalize to 3D, computationalwise. We prefer to fix a point p_∞ which is to belong to the external hole of any regions (except the region formed by the whole domain X).

Definition 4. *Let $p \in X$ and $A \subseteq X$. The* saturation *of A with respect to p is*

$$\mathrm{Sat}(A, p) = X \setminus \mathrm{cc}(X \setminus A, p).$$

Usually, we fix a point $p_\infty \in X$ and denote $\mathrm{Sat}(A) := \mathrm{Sat}(A, p_\infty)$. See figure 3.1.1.

Notice what this definition says: the saturation of A is the complement of the external hole of A, or equivalently, the union of A with all of its internal holes. If $p_\infty \in A$, then the saturation of A is the whole domain X.

Proposition 5. *The saturation operator, acting on subsets of X, has the following properties, all of them proved in [BCM03]:*

1. $A \subseteq \mathrm{Sat}(A)$ *(extensiveness)*
2. $A \subseteq B \implies \mathrm{Sat}(A) \subseteq \mathrm{Sat}(B)$ *(monotony)*
3. $\mathrm{Sat}\,\mathrm{Sat}(A) = \mathrm{Sat}(A)$ *(idempotency)*
4. *If A is connected then* $\mathrm{Sat}(A)$ *is also connected*
5. *If A is open then* $\mathrm{Sat}(A)$ *is also open*

6. *If* A *is closed then* $\mathrm{Sat}(A)$ *is also closed*

7. $\partial\,\mathrm{Sat}(A) \subseteq \partial A$

8. *If* $\mathrm{Sat}(A) \neq X$ *then* $\mathrm{Sat}(A) \subseteq \mathrm{Sat}(\partial A)$

All these properties are quite natural. Properties 1, 2 and 3 say that Sat is well-behaved as an operator of sets. Properties 4, 5 and 6 say that Sat is compatible with the topology of X. Properties 7 and 8 explain that $\mathrm{Sat}(A)$ only depends on the external boundary of A.

3.1.2 Semicontinuous Functions

Given a digital image, defined by its values on a discrete grid, we can produce a semicontinuous function using nearest-neighbor interpolation. The choice of semicontinuous interpolation is not discussed here.

Here we give some equivalent definitions of semicontinuity. This notion is central to our theory because the functions that we use to model images are upper semicontinuous functions. The proofs of all the assertions given here can be found in any textbook on topology, such as [Cho73].

Since most of the following exposition deals with subsets of the image domain, the following notation is useful. Let Ω be a fixed subset of $\mathbb{R}^n$, which serves as the domain for images. When $f : \Omega \to \mathbb{R}$ is any function and P is a predicate on real numbers, we define

$$[P(f)] := \{x \in \Omega \ : \ P(f(x))\}. \tag{3.1}$$

Upper and lower limits Let X be a topological space. If x is a point of X, let $\mathcal{O}(x)$ denote the set of neighborhoods of x. The upper and lower limits of a function $f \in \mathbb{R}^X$ at the point x are defined as follows:

$$\limsup_{y \to x} f(y) = \inf_{U \in \mathcal{O}(x)} \sup_{y \in U} f(y)$$

$$\liminf_{y \to x} f(y) = \sup_{U \in \mathcal{O}(x)} \inf_{y \in U} f(y)$$

Intuitively, the upper limit of f at x is the highest value that the function attains on small neighborhoods of x. A similar interpretation holds for the lower limit.

Notice that, by construction, any function is pointwise bounded by its lower and upper limits:

$$\liminf_{y \to x} f(y) \leq f(x) \leq \limsup_{y \to x} f(y).$$

Local definition of semicontinuity A function is continuous at x when both inequalities of the previous formula are equalities. When the first one is an equality, then the function is *lower semicontinuous at* x. When the second one is an equality, the function is *upper semicontinuous at* x. Thus, a function is

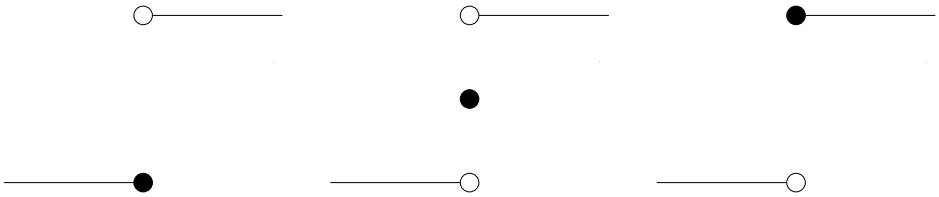


Figure 3.5: Three versions of the step function, a lower semicontinuous, a discontinuous, and an upper semicontinuous.

Figure 3.6: The graph of a function, its epigraph and its hipograph

continuous at a point when it is both upper and lower semicontinuous at that point.

A function is said to be *upper semicontinuous* when it is upper semicontinuous at every point. A function is said to be *lower semicontinuous* when it is lower semicontinuous at every point. Thus, a function is continuous when it is both upper and lower semicontinuous. Intuitively, a discontinuous function will be upper semicontinuous when its value at a discontinuity point is as high as possible. And it will be lower semicontinuous when its value at a discontinuity point is as low as possible.See figure 3.5 for an illustration of this fact for $1D$ functions.

Global definition of semicontinuity Let us introduce the following notation for the levelsets of a function f:

$$[f < \lambda] = \{x \in X : f(x) < \lambda\}$$

$$[f \leq \lambda] = \{x \in X : f(x) \leq \lambda)\}$$

$$[f > \lambda] = \{x \in X : f(x) > \lambda\}$$

$$[f \geq \lambda] = \{x \in X : f(x) \geq \lambda\}$$

and for its epigraph and hipograph (see figure 3.6)

$$\text{epi}(f) = \{(x,t) \in X \times \mathbb{R} : f(x) \leq t\}$$

$$\text{hipo}(f) = \{(x,t) \in X \times \mathbb{R} : f(x) \geq t\}.$$

A function f is lower semicontinuous when any of the three following equivalent conditions holds:

1. $\forall \lambda : [f > \lambda]$ is open
2. $\forall \lambda : [f \leq \lambda]$ is closed
3. epi(f) is closed (in the topological space $X \times \mathbb{R}$).

Similarly, a function f is upper semicontinuous when any of the three following equivalent conditions holds:

1. $\forall \lambda : [f < \lambda]$ is open
2. $\forall \lambda : [f \geq \lambda]$ is closed
3. hipo(f) is closed.

These global definitions will be needed for the construction of the trees on the next section.

3.1.3 Weakly Oscillating Functions

Weakly oscillating functions are our model for well-behaved images. Their interest lies that they have, in a certain sense, a finite structure: the topology of their level sets can be described finitely (however, the level sets themselves may be very complicated).

Definition 6 (regional extrema)**.** *Let $u \in C(\Omega)$ and $M \subseteq \Omega$. We say that M is a regional maximum (resp., minimum) of u at height λ if M is a connected component of $[u = \lambda]$ and, for all $\varepsilon > 0$, the set $[\lambda - \varepsilon < u \leq \lambda]$ (resp., $[\lambda \leq u < \lambda + \varepsilon]$) is a neighborhood of M.*

Definition 7 (weakly oscillating function)**.** *We say that $u \in C(\Omega)$ is weakly oscillating if it has a finite number of regional extrema.*

3.1.4 Trees of Subsets of a Space

The notion of tree can be defined over partially ordered sets. We only need this definition for the partially ordered set of subsets of a given space. In that case, a tree of subsets is a family of subsets such that any two subsets of the family are either disjoint or nested:

Definition 8. *Let Ω be any set (called the space) and let $\mathcal{T} \subsetneq \mathcal{P}(\Omega)$ be a family of subsets of Ω. We say that $\mathcal{T}$ is a tree of subsets of Ω if*

(i) $\mathcal{T}$ contains Ω,

(ii) If $C, D \in \mathcal{T}$, then either $C \cap D \neq \emptyset$, $C \subseteq D$ or $D \subseteq C$. In the last two cases we shall say that C and D are nested.

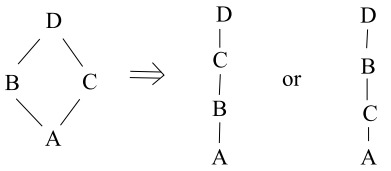


Figure 3.7: A tree of subsets has no cycles: $A \subseteq B_i \subseteq C,\ i = 1, 2 \implies B_1 \subseteq B_2$ or $B_2 \subseteq B_1$

The elements of the tree will be called nodes. This structure is called a tree because it disallows cycles of subsets (see Figure 3.7).

When $\mathcal{T}$ is finite, its structure can be recovered from the relation $\lessdot$, which is the transitive reduction of the relation $\subseteq$

$$A \lessdot B \quad \iff \quad A \subseteq B \text{ and there is no } C \text{ such that } A \subseteq C \subseteq B$$

In that case, $(\mathcal{T}, \lessdot)$ is a tree (e.g., a graph with no cycles). If the tree has n nodes then the relation $\lessdot$ can be defined by listing its $n-1$ pairs. This is the foundation for the efficient representation of trees of subsets explained on section 4.3. In the typical continuous setting, the trees $\mathcal{T}$ are infinite sets. In that case, we have to be more careful to define its structure. For that purpose, the main tool are *intervals* of subsets:

Definition 9 (interval,limit node)**.** *Let $A \subseteq B \subseteq \Omega$. We define $[A, B]$ as the interval of $\mathcal{T}$ between A and B, i.e.,*

$$[A, B] = \{S :\ S \in \mathcal{T},\ A \subseteq S \subseteq B\}\,.$$

We also define

$$\inf[A, B] = \bigcap_{S \in [A,B]} S \quad \textit{and} \quad \sup[A, B] = \bigcup_{S \in [A,B]} S.$$

We say that T is a limit node of $\mathcal{T}$ if it is the infimum or the supremum of a nonempty interval of $\mathcal{T}$.

In using the notation $[A, B]$ we implicitly understand that it is an interval of $\mathcal{T}$. In case that we are considering several trees at the same time, we may use the notation $[A, B]_{\mathcal{T}}$ to stress the fact that we refer to an interval of $\mathcal{T}$. This language is useful but somewhat awkward: notice that limit nodes of $\mathcal{T}$ may or may not be nodes of $\mathcal{T}$.

Observe that, if $A \subseteq B \subseteq \Omega$, then $[A, B] = [\inf[A, B], \sup[A, B]]$. Thus, when considering an interval, we may always assume that its extreme sets are limit nodes.

Definition 10 (leaf)**.** *We call leaf of $\mathcal{T}$, or simply, a leaf, any limit node $L = \inf[A, B]$ that does not contain any other node of $\mathcal{T}$.*

Notice that the definition of leaf inherits the awkwardness of the definition of limit node: a leaf of $\mathcal{T}$ may not actually be a node of $\mathcal{T}$.

Definition 11 (bifurcation)**.** *Let $B \subseteq \Omega$. We say that B contains a bifurcation in $\mathcal{T}$ if there exist $S, T \in \mathcal{T}$ such that $S, T \subseteq B$ and $S \cap T = \emptyset$. Let $A \subseteq B \subseteq \Omega$. We say that there is a bifurcation between A and B if $\inf[A, B] \neq \emptyset$ and there is $S \in \mathcal{T}$ such that $S \subseteq B$ and $S \cap \inf[A, B] = \emptyset$.*

Definition 12 (branch)**.** *Let $A \subseteq B \subseteq \Omega$. We say that $[A, B]$ is a branch of $\mathcal{T}$ if there is no bifurcation between A and B.*

We say that a branch $[A, B]$ contains $x \in \Omega$ if there is a node $S \in [A, B]$ such that $x \in S$.

In [CM10], Proposition 2.24, the following algebraic property of branches is proven:

Proposition 13. *Let $A_1, A_2 \neq \emptyset$, and $[A_1, B_1]$, $[A_2, B_2]$ two branches of $\mathcal{T}$ such that*

$$[A_1, B_1] \cap [A_2, B_2] \neq \emptyset.$$

Without loss of generality we may assume that A_1, A_2, B_1, B_2 are limit nodes. Then $[A_1 \cap A_2, B_1 \cup B_2]$ is a branch.

Proposition 13 permits us to define the maximal branch containing a given node $S \in \mathcal{T}$, since we can take unions of intersecting intervals:

Definition 14 (maximal branch)**.** *Let $S \in \mathcal{T}$. We define $\mathcal{B}_{\mathcal{T}}(S)$, the maximal branch contaning S, as*

$$\mathcal{B}_{\mathcal{T}}(S) = \bigcup \{[A, B] : [A, B] \text{ is a branch of } \mathcal{T} \text{ s.t. } S \in [A, B]\}.$$

3.2 Definitions of the Tree of Shapes

Once we have introduced all the necessary mathematical preliminaries, defining the tree of shapes is simply a matter of putting them together.

3.2.1 Chain of U/L Level sets, ULT, LLT, TOS

Given an image $f : \Omega \to \mathbb{R}$ We define the following sets:

- $U_0(f) := \{[f \geq \lambda] \ : \ \lambda \in \mathbb{R}\}$ (chain of upper level sets of f)
- $L_0(f) := \{[f < \lambda] \ : \ \lambda \in \mathbb{R}\}$ (chain of lower level sets of f)
- $U(f) := \{cc([f \geq \lambda], p) \ : \ \lambda \in \mathbb{R}, \ p \in \Omega\}$ (tree of upper level sets of f)
- $L(f) := \{cc([f < \lambda], p) \ : \ \lambda \in \mathbb{R}, \ p \in \Omega\}$ (tree of lower level sets of f)
- $S(f) := \{Sat(s) \ : \ s \in U(f) \cap L(f)\}$ (tree of shapes of f)

All of these sets are subsets of $\mathcal{P}(\Omega)$, from where they inherit a partial order structure. As partially ordered sets, it is immediate to check that $U_0(f)$ and $L_0(f)$ are chains (totally ordered sets), and that $U(f)$ and $L(f)$ are trees. The fact that $S(f)$ is also a tree is not evident and it requires a proof.

Notice that there are bijections between $U_0(f)$, $L_0(f)$ and $f(\Omega)$, which for continuous functions f is an interval of $\mathbb{R}$. Similarly, there are bijections between branches of $U(f)$ (or of $L(f)$) and intervals of $\mathbb{R}$. Thus, the trees of upper and lower level sets are endowed with a natural topology as unions of intervals of $\mathbb{R}$. The precise way in which these intervals are linked is not as nice: they may be connected or disconnected, depending on whether the branch is open or closed. The goal of the *limit nodes* defined above is precisely to assure that the branches are connected, by completing the branches with new elements when required.

3.2.2 Tree Structure of the TOS

Let us fix a point $p_\infty \in \Omega$. Given an image $f : \Omega \to \mathbb{R}$, we call shapes of inferior (resp. superior) type the sets of the form

$$\mathrm{Sat}(\mathrm{cc}([u < \lambda], p_\infty) \qquad (\text{resp. } \mathrm{Sat}(\mathrm{cc}(u \geq \lambda], p_\infty)))$$

where $\lambda \in \mathbb{R}$. We call shape of f any shape of inferior of superior type. We denote by $S(u)$ the set of all shapes of u.

Note that, by definition, shapes of superior type are closed, while shapes of inferior type are open. Since shapes are connected, the only shapes of both types are $\emptyset$ and Ω.

Theorem 15. *Any two shapes are either disjoint or nested. Hence, $S(u)$ is a tree.*

Proof. See [BCM03] □

The language of definitions 9-14, is adapted to the particular case of trees of shapes ($\mathcal{T} = \mathcal{S}(u)$). Thus, limit nodes of the tree $\mathcal{S}(u)$ are called *limit shapes*, etc. Besides the branches and maximal branches of $\mathcal{S}(u)$, we are interesed in branches all of whose nodes are shapes of the same type (upper or lower).

Definition 16 (monotone section)**.** *Let $A \subseteq B \subseteq \Omega$. We say that $[A, B]$ is a* monotone branch *of $\mathcal{S}(u)$ if $[A, B]$ is a branch with all shapes of the same type.*

Proposition 13 for general branches has an analogous for monotonous branches (proven in [CM10] 2.28):

Proposition 17. *Let $A_1, A_2 \neq \emptyset$, and let $[A_1, B_1]$, $[A_2, B_2]$ be two monotone sections (of $\mathcal{S}(u)$) of the same type such that $[A_1, B_1] \cap [A_2, B_2] \neq \emptyset$. Without loss of generality we may assume that A_1, A_2, B_1, B_2 are limit shapes. Then $[A_1 \cap A_2, B_1 \cup B_2]$ is a monotone section.*

Definition 18 (maximal monotone section)**.** *Let $s \in \mathcal{S}(u)$. We define $B_{\mathcal{S}(u)}(s)$, the* maximal monotone section *containing s, as*

$$\mathcal{B}_{\mathcal{S}(u)}(s) = \bigcup \{[A, B] : [A, B] \text{ is a monotone branch of } \mathcal{T} \text{ s.t. } s \in [A, B]\}.$$

Maximal monotone branches are interesting because they are the basic building blocks of the tree of shapes. As subsets of Ω, they are the largest possible stacks of "parallel" level curves of u that do not contain any singularity.

3.3 Combinatorial properties of the continuous tree

The goal of this section is to show that the monotone branches of the tree of shapes correspond to some branches of the upper and lower trees, and that the connectivity between these monotone branches is given by the holes of limit nodes of the two trees. In some sense, this means that the tree of shapes can be obtained as the *fusion* of the trees of upper and lower level sets. This construction leads to an algorithm to compute the tree of shapes of digital images, by joining the branches of their trees of upper and lower level sets. In the continuous setting, there are some delicate questions to tackle, which are covered in detail in the monography [CM10].

3.3.1 Fine properties of the saturation operator

Let us state three useful properties of the saturation operator with respect to holes. The proofs only use elementary point-set topology and can be found on [CM10], as lemmata 2.12, 2.13 and 3.14.

Lemma 19 (holes are saturations)**.** *Assume that $X \subseteq \Omega$ is open or closed. Let $C \in \mathcal{CC}(X)$, and $x \in \mathrm{Sat}(C) \setminus C$. Then there exists $O \in \mathcal{CC}(\Omega \setminus X)$ such that $x \in \mathrm{Sat}(O) \subseteq \mathrm{Sat}(C)$. Moreover, if X is open and Y is an internal hole of C, then there exists $O \in \mathcal{CC}(\Omega \setminus X)$ such that $Y = \mathrm{Sat}(O)$. The same statement holds if X is closed and has a has a finite number of connected components.*

Lemma 20 (stauration commutes with limits)**.** *(i) Let $(K_n)_{n\in\mathbb{N}}$ be a decreasing sequence of continua, $K = \cap_n K_n$. Then $\mathrm{Sat}(K) = \cap_n \mathrm{Sat}(K_n)$.*
(ii) Let $(O_n)_{n\in\mathbb{N}}$ be an increasing sequence of domains, $O = \cup_n O_n$. Then $\mathrm{Sat}(O) = \cup_n \mathrm{Sat}(O_n)$.

Proposition 21 (hole representation)**.** *If A is a closed set, C a connected component of A and H an internal hole of C, then, for any x in H,*

$$H = \bigcup_{G=\mathrm{Sat}(G'):G'\in\mathcal{CC}(\Omega\setminus A),\, x\in G\subseteq H} G.$$

Thus, any hole of C can be expressed as a countable union of saturations of connected components of $\Omega \setminus A$.

3.3.2 Fine properties of weakly oscillating functions

Let us state three useful properties of weakly oscillating functions. The proofs only use elementary point-set topology and the previous results on the saturation operator. The proofs can be found on [CM10], as lemmata 4.10, 4.11, 4.12 and 4.26.

Lemma 22 (w.o.f. have a finite number of holes)**.** *Let $u \in C(\Omega)$ be a weakly oscillating function. Then for each $\lambda \in \mathbb{R}$, if $X \in \mathcal{CC}([u > \lambda])$ or $X \in \mathcal{CC}([u \geq \lambda])$ is nonempty, then X contains a regional maximum of u. A similar statement holds for lower level sets. Thus, for each $\lambda \in \mathbb{R}$, there is a finite number of connected components of $[u \geq \lambda]$ and each component has a finite number of holes.*

Lemma 23 (holes of level sets are saturations of level sets)**.** *Let $u \in C(\Omega)$ be a weakly oscillating function. Let $X \in \mathcal{CC}([\lambda \leq u \leq \mu])$, $\lambda \leq \mu$, and let H be a hole of X. Then H is the saturation of a connected component either of $[u < \lambda]$ or of $[u > \mu]$.*

Lemma 24 (classification of holes)**.** *Let $u \in C(\Omega)$ be a weakly oscillating function. Let X be a connected component of $[\lambda \leq u \leq \mu]$, $\lambda \leq \mu$, and let L be a hole of X. Then there is some $\eta > 0$ such that either*

i) $\mathrm{Sat}(X, L) = \mathrm{Sat}(\mathrm{cc}([u \geq \lambda], X), L)$*, and $u < \lambda$ on $L_\eta := \{p \in L : d(p, X) < \eta\}$, or*

ii) $\mathrm{Sat}(X, L) = \mathrm{Sat}(\mathrm{cc}([u \leq \mu], X), L)$*, and $u > \mu$ on $L_\eta := \{p \in L : d(p, X) < \eta\}$.*

In the first case of the alternative holds, we say that L is a hole of negative type, in the second case we say that L is a hole of positive type.

Proposition 25 (characterization of limit shapes of w.o.f.)**.** *Let $u \in C(\overline{\Omega})$ be a weakly oscillating function. Then the limit shapes of u are sets of the form* $\mathrm{Sat}(C)$ *where either $C \in \mathcal{C}C([u \geq \lambda])$, or $C \in \mathcal{C}C([u > \lambda])$, or $C \in \mathcal{C}C([u \leq \lambda])$, or $C \in \mathcal{C}C([u < \lambda])$. Conversely, the sets of this form are limit shapes.*

3.3.3 Signatures

Signatures are a formal device to encode which regional extrema are contained within a given shape. They are defined in such a way that all the shapes along a branch have the same signature, which is unique to that branch. Thus, they are useful to identify branches. Let $u \in C(\Omega)$ be weakly oscillating and let $\mathcal{E}$ be the (finite) set of regional extrema of u.

Definition 26 (signature)**.** *For $X \subseteq \Omega$, we note $\mathcal{E}(X)$ the set $\{E \in \mathcal{E} | E \subseteq X\}$. We define the signature of X as $sig(X) = \{\mathcal{E}(C) | C \in \mathcal{CC}(X)\}$. For $\lambda \in \mathbb{R}$, we define the signature of u at level λ the set $sig(\lambda) = sig([u \geq \lambda]) \cup sig([u < \lambda])$.*

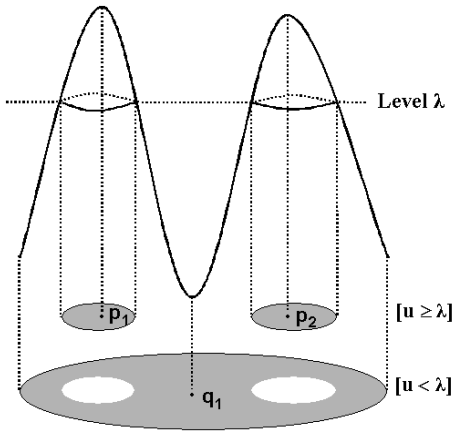


Figure 3.8: A function u and its upper and lower level sets at level λ with its assigned signature. The set $[u \geq \lambda]$ has two connected components depicted in gray and its signature consists of two points $\{p_1, p_2\}$. The set $[u < \lambda]$ has only one connected component with two holes, and is depicted as the circular region below with the two white holes. Its signature consists of the point q_1. (Figure taken from [CM10]).

Notice that $sig(X)$ and $sig(\lambda)$ are in $\mathcal{P}(\mathcal{P}(\mathcal{E}))$. We remark that for any $\lambda \in (\min u, \max u]$, $sig(\lambda)$ is a partition of $\mathcal{E}$. That all elements of $sig(\lambda)$ are nonempty is a consequence of Lemma 22. Moreover, since $\mathcal{CC}([u \geq \lambda]) \cup \mathcal{CC}([u < \lambda])$ is a partition of $\bar{\Omega}$ and each connected component of isolevel is contained in one element of this partition, if $E \in \mathcal{E}$, E is contained in one unique element C of this partition, therefore $E \in sig(C)$, and E belongs to no other element of $sig(\lambda)$.

The definition of signature is illustrated in Fig. 3.8. Note that in the case presented in this figure, $sig([u \geq \lambda]) = \{\{p_1\}, \{p_2\}\}$, $sig([u < \lambda]) = \{\{q_1\}\}$ and $sig(\lambda) = \{\{p_1\}, \{p_2\}, \{q_1\}\}$.

Our next lemma proves that for weakly oscillating functions the signature may only change from above.

Lemma 27 (the signature may only change from above)**.** *Let $u \in C(\overline{\Omega})$ be a weakly oscillating function. Let $\lambda \in \mathbb{R}$. There is $\varepsilon > 0$ such that $sig(\mu)$ is constant for all $\mu \in (\lambda - \varepsilon, \lambda]$.*

Proof. Let $X^{\lambda,i}$, $X_{\lambda,j}$, $i = 1, \ldots, r$, $j = 1, \ldots, s$, be the family of connected components of $[u \geq \lambda]$, resp. $[u < \lambda]$. Let $i \in \{1, \ldots, r\}$. For each $\mu < \lambda$, let $X^{\mu,i}$ be the connected component of $[u \geq \mu]$ containing $X^{\lambda,i}$. Then, obviously, we have

$$X^{\lambda,i} \subseteq \cap_{\mu<\lambda} X^{\mu,i}.$$

Now, since $X^{\mu,i}$ is a decreasing sequence of continua their intersection is also a continuum [Kur66]. Moreover, it is contained in $[u \geq \lambda]$. Therefore,

$$\cap_{\mu<\lambda} X^{\mu,i} \subseteq \mathrm{cc}([u \geq \lambda], p_i) = X^{\lambda,i},$$

and we have the equality of both sets. As a consequence, there is an $\varepsilon > 0$ such that for each $\mu \in (\lambda - \varepsilon, \lambda]$, the sets $X^{\lambda,i}$, $i = 1, \ldots, r$, are contained in different connected components of $[u \geq \mu]$. Moreover, since the number of connected components of each $[u \geq \mu]$ is finite, we may choose $\varepsilon > 0$ such that for each $\mu \in (\lambda - \varepsilon, \lambda]$ the set $[u \geq \mu]$ consists of r connected components, each one of them containing a different component of $[u \geq \lambda]$. Since u is weakly oscillating, for $\epsilon > 0$ small enough, the regional extrema of u in each $X^{\mu,i}$, $i = 1, \ldots, r$, is constant for $\mu \in (\lambda - \epsilon, \lambda]$.

Let $\mu_n \uparrow \lambda$. Again, using that $\cup_n [u < \mu_n] = [u < \lambda]$, for n large enough, we have that $[u < \mu_n] \cap X_{\lambda,j}$, $j = 1, \ldots, s$, are the connected components of $[u < \mu_n]$. As above, we know that the regional extrema of u in each $[u < \mu_n] \cap X_{\lambda,j}$ coincide with the regional extrema in $X_{\lambda,j}$, $j = 1, \ldots, s$, for n large enough. We conclude that there is an $\varepsilon > 0$ such that $sig(\mu)$ is constant for each $\mu \in (\lambda - \varepsilon, \lambda]$. □

Let us explain the phenomena reflected by a change of signature. For simplicity, let us explain them at the discrete level. For that, let us consider two consecutive levels λ and $\lambda + 1$. Let $\cup sig([u \geq \mu]) = \cup_{C \in \mathcal{CC}([u \geq \mu])} sig(C)$, $\mu \in \mathbb{R}$. Notice that $\cup sig([u \geq \lambda+1]) \subseteq \cup sig([u \geq \lambda])$. If $sig(\lambda) \neq sig(\lambda+1)$ several things may happen: (a) $sig([u \geq \lambda]) \neq sig([u \geq \lambda + 1])$ while $\cup sig([u \geq \lambda]) = \cup sig([u \geq \lambda+1])$, (b) $sig([u < \lambda]) \neq sig([u < \lambda+1])$ while $\cup sig([u < \lambda]) = \cup sig([u < \lambda+1])$, (c) $\cup sig([u \geq \lambda]) \neq \cup sig([u \geq \lambda + 1])$ (hence $sig([u \geq \lambda]) \neq sig([u \geq \lambda + 1])$). In this last case we also have that $\cup sig([u < \lambda]) \neq \cup sig([u < \lambda + 1])$. If we are in case (a) there must be two connected components of $[u \geq \lambda+1]$ that have merged at level λ and the corresponding signatures fused. If we are in case (b) there must be two connected components of $[u < \lambda + 1]$ that have split at level λ and the corresponding signatures split. If we are in case (c) then a regional extremum has been transferred from $\cup sig([u < \lambda + 1])$ to $\cup sig([u \geq \lambda])$ be either a regional maximum because a connected component of $[u \geq \lambda]$ appeared which was not present at level $\lambda + 1$, or a regional minimum because a connected component of $[u < \lambda + 1]$ disappeared at level λ. These two last cases could happen combined with merging or splitting of connected components (see Fig. 3.9).

Finally, observe that since the signature of any connected component of $[u \geq \lambda]$ increases (respectively the signature of any connected component of $[u < \lambda]$ decreases) as λ decreases, then there are only finitely many possible changes in $sig(\lambda)$. Thus, if $u \in C(\overline{\Omega})$ is weakly oscillating, then the number of critical values of u is finite. In particular, the signature $sig(\mu)$ is locally constant at each side of a critical value, i.e., if λ is a critical value, then there is $\varepsilon > 0$ such that

$$sig(\mu) = sig(\lambda) \neq sig(\mu') \quad \text{and } sig(\mu') \text{ is constant}$$

for each $\mu < \lambda < \mu'$, $\mu \in (\lambda - \varepsilon, \lambda)$, $\mu' \in (\lambda, \lambda + \varepsilon)$. This implies that the previous description of the changes of the topology of level sets for discrete images also holds in the continuous case.

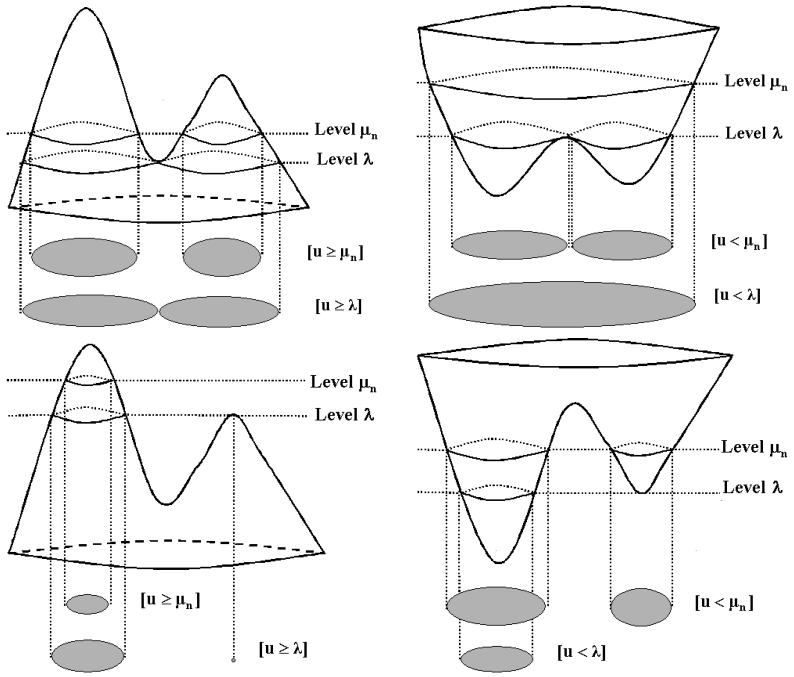


Figure 3.9: The top images display cases of merging and splitting of connected components as λ decreases. The bottom figures display the birth of an upper connected component or the death of a lower one. Those are the changes of signature as λ varies. (Figure taken from [CM10]).

3.3.4 Structure of the Trees

Recall that if $u : \Omega \to \mathbb{R}$ is an upper semicontinuous function, $\mathcal{U}(u)$ and $\mathcal{L}(u)$ denote the trees of connected components of upper and lower level sets of u. To understand the structure of both trees, their leaves and their maximal branches, it is useful to describe first their limit nodes.

Proposition 28 (characterization of limit nodes)**.** *(i) If X is a limit node of $\mathcal{U}(u)$, then either $X \in \mathcal{U}(u)$ or $X \in \mathcal{CC}([u > \lambda])$ for some $\lambda \in \mathbb{R}$.*

(ii) If X is a limit node of $\mathcal{L}(u)$, then either $X \in \mathcal{L}(u)$ or $X \in \mathcal{CC}([u \leq \lambda])$ for some $\lambda \in \mathbb{R}$.

Proof. Being identical, we just sketch the proof of (i). If X is a limit node of $\mathcal{U}(u)$, then X is an inf or a sup of an ordered set of upper connected components which we may assume countable. If $X = \cap_n X_n$ where $X_n \in \mathcal{CC}([u \geq \lambda_n])$ with $\lambda_n \uparrow \lambda$, then $X \in \mathcal{CC}([u \geq \lambda])$. If $X = \cup_n X_n$ where $X_n \in \mathcal{CC}([u \geq \lambda_n])$ with $\lambda_n \downarrow \lambda$, then $X \in \mathcal{CC}([u > \lambda])$. □

Proposition 29 (leaves are regional extrema)**.** *Let $u \in C(\Omega)$ be a weakly oscillating function. Then*

(i) If X is a leaf of $\mathcal{U}(u)$, then X is a regional maximum of u.

(ii) If X is a leaf of $\mathcal{L}(u)$, then X is a regional minimum of u.

Proof. (i) By Proposition 28, if X is a leaf of $\mathcal{U}(u)$, then $X \in \mathcal{CC}([u \geq \lambda])$ for some $\lambda \in \mathbb{R}$. If $u(x) > \lambda$ for some $x \in X$, then the node $\mathrm{cc}([u \geq u(x)], x)$ is nonempty and contained in X. Thus $u = \lambda$ on X and $X \in \mathcal{CC}([u = \lambda])$. If $X = \Omega$, our statement is obviously true. If $X \neq \Omega$, then by Lemma 24 all holes of X must be of negative type. Hence $\mathrm{cc}([u > \lambda - \epsilon], X) = \mathrm{cc}([\lambda - \epsilon < u \leq \lambda], X)$, for any $\epsilon > 0$, and $\mathrm{cc}([u > \lambda - \epsilon], X)$ is an open set containing X.

Being similar to the proof of (i), we skip the proof of (ii). □

Proposition 30 (finite structure of the upper tree)**.** *Assume that $u \in C(\Omega)$ is a weakly oscillating function. The tree $\mathcal{U}(u)$ has a finite number of leaves and a finite number of maximal branches. If $\mathcal{B} = [A, B]$ is a maximal branch of $\mathcal{U}(u)$ with A, B being limit nodes, then*

a) either $B = \Omega$ or $B \in \mathcal{CC}([u > \lambda])$ for some $\lambda \in \mathbb{R}$. In the second case there is no bifurcation between A and B, and if we let $B' = \mathrm{cc}([u \geq \lambda], B)$, then $B' = \inf[B, \Omega]$ and $[A, B']$ contains a bifurcation. Moreover, B cannot be a leaf unless u is constant.

b) $A \in \mathcal{CC}([u \geq \lambda])$ for some $\lambda \in \mathbb{R}$ and either A is a leaf, or for any $X \in \mathcal{U}(u)$, $X \subsetneq A$, $[X, A]$ contains a bifurcation.

Proof. Leaves of $\mathcal{U}(u)$ are regional maxima of u, hence there are finitely many of them. Let $\mathcal{B} = [A, B]$ be a maximal branch in $\mathcal{U}(u)$. By Proposition 28 either $B \in \mathcal{CC}([u \geq \lambda])$, or $B \in \mathcal{CC}([u > \lambda])$ for some $\lambda \in \mathbb{R}$. If $B \in \mathcal{CC}([u \geq \lambda])$ and $\lambda > \inf_{x \in \Omega} u(x)$, then, using Lemma 27, we would be able to extend the branch $\mathcal{B}$ to the right. Hence, $\lambda = \inf_{x \in \Omega} u(x)$, i.e. $B = \Omega$. If $B \in \mathcal{CC}([u > \lambda])$, then $[A, B]$ does not contain a bifurcation. Let $B' = \mathrm{cc}([u \geq \lambda], B)$, then $[A, B']$ must contain a bifurcation, otherwise $\mathcal{B}$ would not be maximal. The argument in Lemma 27 proves that $B' = \inf[B, \Omega]$. The last assertion follows from Proposition 29.

Now, observe that $A \in \mathcal{CC}([u \geq \lambda])$ for some $\lambda \in \mathbb{R}$. Since $\mathcal{B}$ is maximal, if A is not a leaf, then for any $X \in \mathcal{U}(u)$, $X \subsetneq A$, $[X, A]$ contains a bifurcation.

Since u has a finite number of regional maxima, there are finitely many maximal branches in $\mathcal{U}(u)$, since any two of them are disjoint. □

Proposition 31 (finite structure of the lower tree)**.** *Assume that $u \in C(\Omega)$ is a weakly oscillating function. The tree $\mathcal{L}(u)$ has a finite number of leaves and a finite number of maximal branches. If $\mathcal{B} = [A, B]$ is a maximal branch of $\mathcal{L}(u)$ with A, B being limit nodes, then*

a) either $B = \Omega$ or $B \in \mathcal{CC}([u < \lambda])$ for some $\lambda \in \mathbb{R}$. In the second case there is no bifurcation between A and B, and if we let $B' = \mathrm{cc}([u \leq \lambda], B)$, then $B' = \inf(B, \Omega]$ and $[A, B']$ contains a bifurcation. Moreover, B cannot be a leaf unless u is constant.

b) $A \in \mathcal{CC}([u \leq \lambda])$ *for some* $\lambda \in \mathbb{R}$ *and either* A *is a leaf, or for any* $X \in \mathcal{L}(u)$, $X \subsetneq A$, $[X, A]$ *contains a bifurcation.*

We have used the notation $(B, \Omega] = [B, \Omega] \setminus \{B\}$.

Proof. Leaves are regional minima of u, hence there are finitely many of them. Let $\mathcal{B} = [A, B]$ be a maximal branch in $\mathcal{L}(u)$. By Proposition 28 either $B \in \mathcal{CC}([u \leq \lambda])$ or $B \in \mathcal{CC}([u < \lambda])$ for some $\lambda \in \mathbb{R}$. If $B \in \mathcal{CC}([u \leq \lambda])$ and $\lambda < \sup_{x \in \Omega} u(x)$, then, using the arguments in Lemma 27, we would be able to extend the branch $\mathcal{B}$ to the right. Hence, $\lambda = \sup_{x \in \Omega} u(x)$ and $B = \Omega$.

If $B \in \mathcal{CC}([u < \lambda])$, then $[A, B]$ does not contain a bifurcation. Let us prove that $[A, B']$ contains a bifurcation where $B' = \mathrm{cc}([u \leq \lambda], B)$. If $B' = \Omega$ and it does not contain a bifurcation we are in the previous case for B (we could take $B = B'$). Thus, we may assume either that $B' = \Omega$ and $[A, B']$ contains a bifurcation; or $B' \neq \Omega$, that is $\lambda < \sup_{x \in \Omega} u(x)$. We have to consider only the last case. By (the proof of) Lemma 27, there is an $\epsilon > 0$ such that if $\mu \in (\lambda, \lambda + \epsilon)$, then $C := \mathrm{cc}([u < \mu], B')$ does not contain any other connected component of $[u \leq \lambda]$ besides B' and contains the same regional extrema as B'. If there is no bifurcation in $[B, C]$, then we can extend $[A, B]$ to the right. Thus, we may assume that $[B, C]$ contains a bifurcation, i.e., there is $Y \in \mathcal{CC}([u < \alpha])$ with $Y \cap B = \emptyset$ and $Y \subseteq C$. Notice that we have $\alpha \leq \mu$. Let us prove that we may assume that $\alpha \leq \lambda$. If $\lambda < \alpha \leq \mu$ and $Y \cap [u \leq \lambda] \neq \emptyset$, then we take V to be a connected component of $[u \leq \lambda]$ inside Y. Then C contains V and B', but this is not possible in view of our choice of C. Hence $Y \subseteq [\lambda < u \leq \mu]$, and we deduce that Y contains a regional minimum of u not in B', hence also C does it, a contradiction with our choice of C. Thus, we may assume that $\alpha \leq \lambda$. Let $V := \mathrm{cc}([u < \lambda], Y)$. Since $Y \cap B = \emptyset$, we have that $V \cap B = \emptyset$. If $\mathrm{cc}([u \leq \lambda], V)$ is disjoint to B', then C contains two connected components of $[u \leq \lambda]$, again a contradiction with our choice of C. Hence, $B' = \mathrm{cc}([u \leq \lambda], V)$, and in this case $[A, B']$ contains a bifurcation since B and V are disjoint. Finally, the argument in Lemma 27 proves that $B' = \inf(B, \Omega]$. The last assertion in a) follows from Proposition 29.

Since A is a limit node, then $A \in \mathcal{CC}([u \leq \lambda])$ for some $\lambda \in \mathbb{R}$. Let us prove that if A is not a leaf, then for any $X \in \mathcal{L}(u)$, $X \subsetneq A$, $[X, A]$ contains a bifurcation, i.e., there is $Y \in \mathcal{L}(u)$, $Y \subseteq A$ and $Y \cap X = \emptyset$. Fix such an X. Observe that $\inf_A u < \lambda$, otherwise $A \in \mathcal{CC}([u = \lambda])$ and A would be a leaf. If there are more than one connected components of $[u < \lambda]$ in A, our assertion is true. Thus, we may assume that there is only one connected component of $[u < \lambda]$ in A. Let $S \in [A, B]$, $S \in \mathcal{CC}([u < \lambda'])$ with $\lambda < \lambda'$. Observe that $[X, S]$ contains a bifurcation otherwise we could enlarge $[A, B]$ to the left. Let $Y \in \mathcal{L}(u)$ be such that $Y \subseteq S$ and $Y \cap X = \emptyset$. Notice that we may write that $X \in \mathcal{CC}([u < \alpha])$ with $\alpha \leq \lambda$ and $Y \in \mathcal{CC}([u < \mu])$ for some $\mu < \lambda'$ (if $\mu = \lambda'$, then $Y = S$, contradicting the fact that $X \cap Y = \emptyset$). Since there is only one connected component of $[u < \lambda]$ in S (otherwise there would be a bifurcation in $[A, S]$ since A identifies one of them), we have that either $Y \subseteq A$ or $Y \subseteq [\lambda \leq u < \mu]$. In the first case, we have that there is a bifurcation in $[X, A]$. Let us consider the

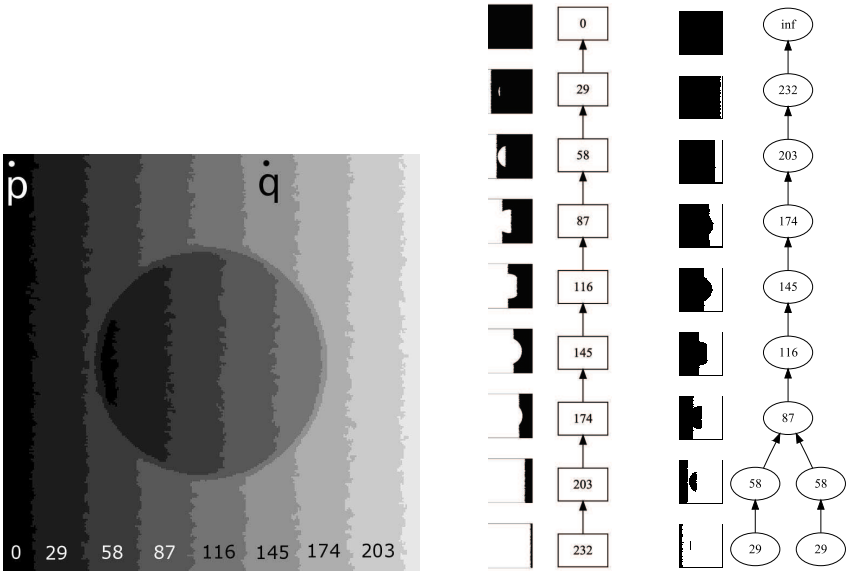


Figure 3.10: An image and the trees of its connected components of upper and lower level sets (component trees). Left: original image. The numbers at the bottom of the image denote the gray levels. The points p and q are two possible points of infinity. Middle: the tree of connected components of upper level sets $\mathcal{U}(u)$. They are denoted as squares with the value of their defining gray level inside. Right: the tree of connected components of lower level sets $\mathcal{L}(u)$. They are denoted as ellipses. In both trees, the level sets are depicted in black at the left side of the nodes.

second case: if $Y \cap A = \emptyset$, then $[A, B]$ contains a bifurcation. Hence $Y \cap A \neq \emptyset$. Since Y is a node and A a limit node, then either $A \subseteq Y$, or $Y \subseteq A$. In the first case, we obtain a contradiction since $X \subseteq A$ and $X \cap Y = \emptyset$. In the second case we have $Y \in \mathcal{CC}([u = \lambda])$, $Y = A$, and A would be a leaf. We conclude that there are more than one connected components of $[u < \lambda]$ in A and $[X, A]$ contains a bifurcation.

Since u has a finite number of regional maxima, there are finitely many maximal branches in $\mathcal{L}(u)$, since any two of them are disjoint. □

3.3.5 TOS as fusion of some branches from ULT and LLT

Let us recall the following result which is essentially a rephrasing of Lemma 19.

Lemma 32 (holes of level sets are saturations of level sets)**.** *Let $u \in C(\Omega)$ be a weakly oscillating function, $\lambda \in \mathbb{R}$. The family of internal holes of all connected components of $[u \geq \lambda]$ coincides with $\{\mathrm{Sat}(O) : O \in \mathrm{cc}([u < \lambda])\}$.*

Proof. By Lemma 19, if $X \in \mathcal{CC}([u \geq \lambda])$ and Y is an internal hole of X, then there is $O \in \mathcal{CC}([u < \lambda])$ such that $Y = \mathrm{Sat}(O)$. Conversely, let $O \in \mathcal{CC}([u < \lambda])$, and observe that $\Omega \setminus \mathrm{Sat}(O)$ is the external hole of O. By taking a point q in $\mathrm{Sat}(O)$ as the point at infinity, we may adapt the proof of Lemma 19 to obtain that there exists $Z \in \mathcal{CC}([u \geq \lambda])$ such that $\Omega \setminus \mathrm{Sat}(O) = \mathrm{Sat}(Z, q)$. Then $\mathrm{Sat}(O)$ is a hole of Z. □

The nodes of the tree $\mathcal{S}(u)$ are the saturations of the nodes of $\mathcal{U}(u)$ and $\mathcal{L}(u)$. Thus, it is intuitive to think that we can construct $\mathcal{S}(u)$ by fusing the information of the upper and lower trees, and this is indeed the case. This operation can de done very simply because of the precise branch structure of both trees described in Propositions 30 and 31.

We denote by $[A, B]_+$ the interval of the upper tree determined by nodes $A, B \in \mathcal{U}(u)$, and $[A, B]_-$ the interval of the lower tree determined by nodes $A, B \in \mathcal{L}(u)$.

To fix ideas, let us assume that p_∞ is a global minimum of u and let $\Lambda = \mathrm{cc}([u = \inf_{x \in \Omega} u(x)],\ p_\infty)$. Observe that Λ is a leaf of $\mathcal{L}(u)$. Let $\mathcal{L}_\Lambda(u) = \mathcal{L}(u) \setminus [\Lambda, \Omega]_-$. All nodes of $[\Lambda, \Omega]_-$ have Ω as saturation. If $C \in \mathcal{L}_\Lambda(u)$, then all nodes previous to it do not contain Λ. Thus, $\mathcal{L}_\Lambda(u)$ is a union of maximal branches of $\mathcal{L}(u)$. Notice that the only node of $\mathcal{U}(u)$ containing p_∞ is Ω.

The trees $\mathcal{U}(u)$ and $\mathcal{L}_\Lambda(u)$ are broken into maximal branches of the form $[A, B]_i$, $i \in \{+, -\}$, where A is either a leaf or a bifurcating node and B may be a bifurcating node or not, in which case it coincides with Ω. Each branch $\mathcal{B} = [A, B]_i$, $i \in \{+, -\}$, determines the set of shapes $\mathrm{Sat}(\mathcal{B}) := \{\mathrm{Sat}(C) : C \in \mathcal{B}\}$ of $\mathcal{S}(u)$. This will be proved in Lemma 33. Then, given $[A, B]_i$, it suffices to describe how to link the upper end of the interval to another interval either of $\mathcal{U}(u)$ or $\mathcal{L}_\Lambda(u)$.

Lemma 33 (branches of shapes are saturations of branches of level sets)**.** *Let $A, B \in \mathcal{U}(u)$ (resp. $\mathcal{L}(u)$) such that $[A, B]_+$ (resp. $[A, B]_-$) is a branch and $p_\infty \notin B$. Then $\mathrm{Sat}([A, B]_+)$ (resp. $\mathrm{Sat}([A, B]_-)$) is an interval of $\mathcal{S}(u)$.*

Proof. The proofs being similar, we show the result when $A, B \in \mathcal{U}(u)$. Obviously it is sufficient to prove it when $[A, B]_+$ is a maximal branch. According to Proposition 30, we can write $A \in \mathcal{CC}([u \geq \lambda])$ and $B \in \mathcal{CC}([u > \mu])$ with $\mu < \lambda$ as $A \subseteq B$.

If $C \in \mathcal{U}(u)$ and $C \notin [A, B]_+$ then clearly $\mathrm{Sat}(C) \notin [\mathrm{Sat}(A), \mathrm{Sat}(B)]$ (interval of $\mathcal{S}(u)$). We therefore assume $C \in \mathcal{CC}([u < \nu])$ and $\mathrm{Sat}(A) \subseteq \mathrm{Sat}(C) \subseteq \mathrm{Sat}(B)$, which will lead to a contradiction.

If $\nu \leq \mu$, then B is in a hole H of C. If H is an internal hole, we get $\mathrm{Sat}(B) \subseteq H \subseteq \mathrm{Sat}(C)$, a contradiction. If H is the external hole of C, we have $\mathrm{Sat}(C) \cap B = \emptyset$ whereas $A \subseteq \mathrm{Sat}(C)$ and $A \subseteq B$, a contradiction.

If $\mu < \nu$, observe first that $\partial\,\mathrm{Sat}(C) \subseteq B$. Indeed, $\partial\,\mathrm{Sat}(C) \subseteq [u = \nu]$ and is connected, so either is in B or in a hole of B. It cannot be in a hole H since we would have that $A \subseteq \mathrm{Sat}(C) \subseteq \mathrm{Sat}(\partial\,\mathrm{Sat}(C)) \subseteq H$, a contradiction with $A \subseteq B$. Let $D = \mathrm{cc}([u \geq \nu], \partial\,\mathrm{Sat}(C))$. Then D is in the external hole of C and

$D \subseteq B$. Since $A \subseteq \mathrm{Sat}(C)$, we have that $A \cap D = \emptyset$. Hence, there is a bifurcation between A and B in $\mathcal{U}(u)$, which is contrary to the hypothesis that $[A, B]_+$ is a branch. □

In Fig. 3.10 we show an image and its trees $\mathcal{U}(u)$ (middle) and $\mathcal{L}(u)$ (right). Next to each node we see a schematic representation of the level set. In Figs. 3.11 and 3.12 we display the fusion of both trees for two different choices of the point at infinity, the point p in the case shown in Fig. 3.11 and the point q in the case of Fig. 3.12. Notice that the point p is a global minimum of u, while the point q is not. The choice of the point at infinity is arbitrary and the structure of the tree does not depend on it, only its orientation. Choosing the point at infinity is equivalent to choosing the root of the tree; indeed, the root shape is the one which contains the point at infinity. In both figures, the left image displays the decomposition of both trees into maximal branches and encircled in a pointed curve we display all nodes of the lower tree containing the point at infinity, hence having Ω as saturation. Both trees are fused according to the rules described below. Notice that the structure of the upper tree is a chain while the lower one is not. Thus, the fusion of the trees illustrates the case of fusion of the maximal branches of $\mathcal{L}(u)$. Since they are similar to the fusion rules for the upper tree, the figures describe the generic situation.

Before explaining precisely the fusion rules for maximal branches, let us expose their principle. If $[A, B]_\pm$ is a maximal branch of $\mathcal{U}(u)$ or $\mathcal{L}(u)$, B is an open set defined by a strict inequality at a level λ. We can then consider the exterior boundary of B, $\partial\,\mathrm{Sat}(B)$ and the two sets $\mathrm{cc}([u \geq \lambda], \partial\,\mathrm{Sat}(B))$ and $\mathrm{cc}([u \leq \lambda], \partial\,\mathrm{Sat}(B))$ (one is designed by B' and the other by N below). They are limit nodes in $\mathcal{U}(u)$ and $\mathcal{L}(u)$ respectively, and their saturations are limit nodes in $\mathcal{S}(u)$. These saturations are therefore nested and we link $\mathrm{Sat}(B)$ to the smaller one, call it $\mathrm{Sat}(C)$. Then the interval $[\mathrm{Sat}(B), \mathrm{Sat}(C)]$ of $\mathcal{S}(u)$ has no other element than $\mathrm{Sat}(B)$ and $\mathrm{Sat}(C)$ themselves.

From now on, when using limit nodes we shall consider them as parts of the corresponding tree.

Fusion rules for maximal branches of $\mathcal{U}(u)$. Let $[A, B]_+$ be a maximal branch of $\mathcal{U}(u)$. By Proposition 30, either $B = \Omega$ or $B \in \mathcal{CC}([u > \lambda])$ for some $\lambda \in \mathbb{R}$. In the first case, there is nothing to say. In the second case there is no bifurcation between A and B, and if we let $B' = \mathrm{cc}([u \geq \lambda], B)$, then $[A, B']_+$ contains a bifurcation. Observe that, by applying Lemma 32 to $-u$, we have that $\mathrm{Sat}(B)$ is a hole of a connected component N on $[u \leq \lambda]$. We compare the two limit shapes $\mathrm{Sat}(B')$ and $\mathrm{Sat}(N)$ (they are limit shapes by Proposition 25). Since they are limit shapes of different type, either they are different or both coincide with Ω. If both coincide with Ω, then $\Lambda \subseteq N$, and we may link $\mathrm{Sat}(B)$ to Ω. Thus we may assume that $\mathrm{Sat}(B') \neq \mathrm{Sat}(N)$. Since $\mathrm{Sat}(N)$ intersects $\partial B \subseteq B'$, then either $\mathrm{Sat}(N) \subsetneq \mathrm{Sat}(B')$ or $\mathrm{Sat}(B') \subsetneq \mathrm{Sat}(N)$. In the first case, we link $\mathrm{Sat}(B)$ to $\mathrm{Sat}(N)$ which is the limit shape generated by N. Notice that N is the lower end of a (not necessarily maximal) branch of $\mathcal{L}(u)$. In the second case, we link $\mathrm{Sat}(B)$ to $\mathrm{Sat}(B')$. In this case B' is the lower end of a maximal branch of $\mathcal{U}(u)$.

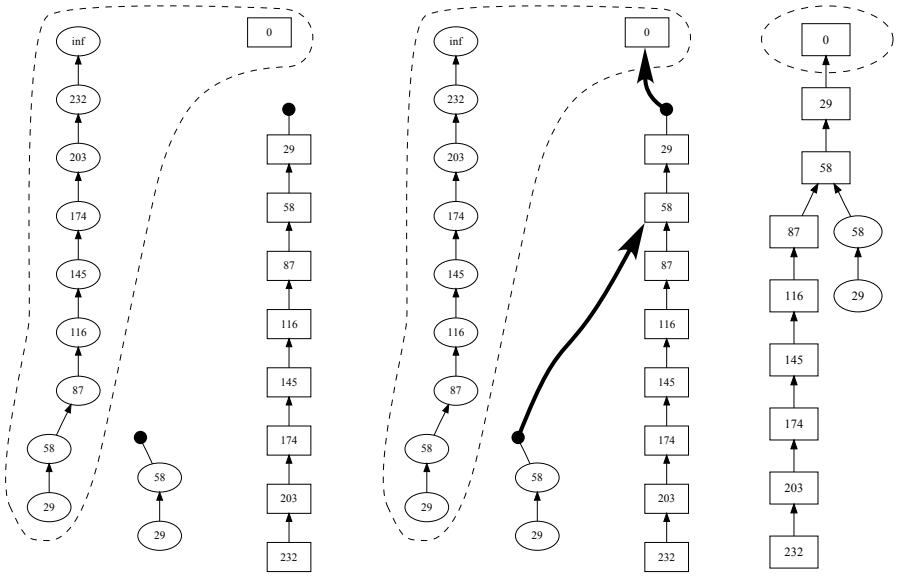


Figure 3.11: Example of the fusion of both trees $\mathcal{U}(u)$ and $\mathcal{L}(u)$ for the image in Fig. 3.10. The point p (at level 0) is taken as the point at infinity. Left: the lower level sets containing p are surrounded by a pointed curve. Their saturation gives the root. We used the value 0 to identify the root. The rest of the trees is decomposed in maximal branches. The nodes that must be connected to the other tree are shown. Middle: the arrows show the connections between the maximal branches of both trees. Right: the tree of shapes obtained by fusion of $\mathcal{U}(u)$ and $\mathcal{L}(u)$. The root is surrounded by a pointed line. In this case, squares and ellipses denote shapes of upper and lower type, respectively.

Fusion rules for maximal branches of $\mathcal{L}(u)$. Let $[A, B]_-$ be a maximal branch of $\mathcal{L}(u)$. By Proposition 31, either $B = \mathrm{cc}([u \leq \lambda]) = \Omega$ or $B \in \mathcal{CC}([u < \lambda])$ for some $\lambda \in \mathbb{R}$. In the first case, there is nothing to say. In the second case there is no bifurcation between A and B, and if we let $B' = \mathrm{cc}([u \leq \lambda], B)$, then $[A, B']_-$ contains a bifurcation. Observe that, by Lemma 32, we have that $\mathrm{Sat}(B)$ is a hole of a connected component N on $[u \geq \lambda]$. Again, we compare the two limit shapes $\mathrm{Sat}(B')$ and $\mathrm{Sat}(N)$. Since they are limit shapes of different type, either they are different or both coincide with Ω. If both coincide with Ω, then $\mathrm{Sat}(N) = \Omega$ and $\lambda = \inf_{x \in \Omega} u(x)$ but this is not the case, due to our choice of B. Thus we may assume that $\mathrm{Sat}(B') \neq \mathrm{Sat}(N)$. Since $\mathrm{Sat}(N)$ intersects $\partial B \subseteq B'$, then either $\mathrm{Sat}(N) \subsetneq \mathrm{Sat}(B')$ or $\mathrm{Sat}(B') \subsetneq \mathrm{Sat}(N)$. In the first case, we link $\mathrm{Sat}(B)$ to $\mathrm{Sat}(N)$ which is the shape generated by N. Notice that N is the lower end of a (not necessarily maximal) branch of $\mathcal{U}(u)$. In the second case, we link $\mathrm{Sat}(B)$ to $\mathrm{Sat}(B')$. In this case B' is the lower end of a maximal branch of $\mathcal{L}(u)$.

Observe that:

(i) the final structure contains the saturations of all connected components of

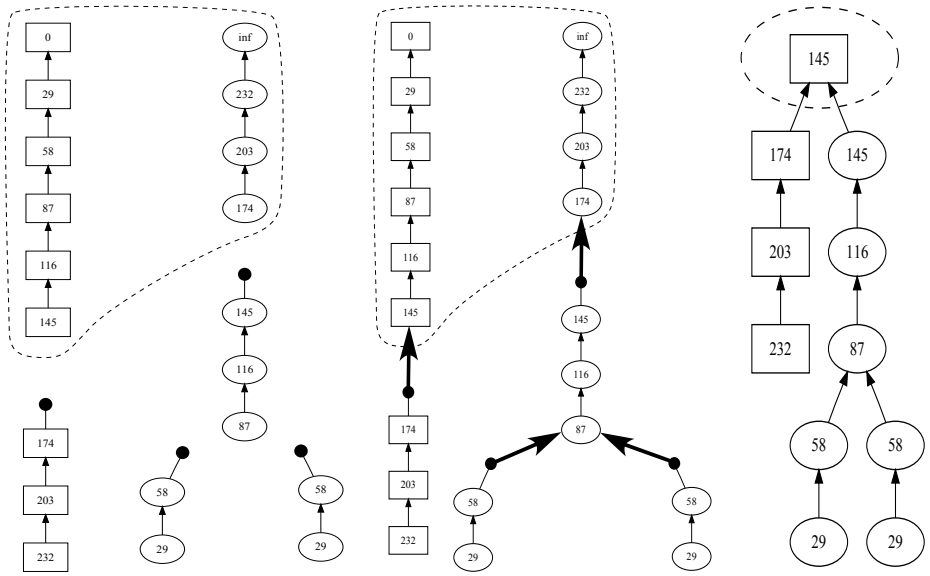


Figure 3.12: Example of the fusion of both trees $\mathcal{U}(u)$ and $\mathcal{L}(u)$ for the image in Fig. 3.10. This time the point q (at level 145) is taken as the point at infinity. Left: the lower level sets containing q are surrounded by a pointed curve. Their saturation gives the root. We used the value 145 to identify the root. The rest of the trees is decomposed in maximal branches. The nodes that must be connected to the other tree are shown. Middle: the arrows show the connections between the maximal branches of both trees. Right: the tree of shapes obtained by fusion of $\mathcal{U}(u)$ and $\mathcal{L}(u)$. The root is surrounded by a pointed line. Again, in this case, squares and ellipses denote shapes of upper and lower type, respectively. Observe that the tree of shapes obtained here is equivalent to the one displayed in Fig. 3.11.

upper and lower level sets,

(ii) If A is a node (or limit node) in $\mathcal{L}(u)$ which is the upper end of an interval, and B a node (or limit node) in $\mathcal{U}(u)$ which is the lower end of an interval, or viceversa, and we have linked $\mathrm{Sat}(A)$ to $\mathrm{Sat}(B)$, we have chosen $\mathrm{Sat}(B)$ to be the minimal limit shape in $\mathcal{U}(u)$ containing A. With this rule, we have not created cycles in the fusion of both trees. Thus, we have constructed a tree.

In Figs. 3.11 and 3.12 we display the decomposition of both trees $\mathcal{U}(u)$ and $\mathcal{L}(u)$ into maximal branches and their fusion. First the nodes whose saturation is Ω have been separated and represented by a single shape, the root of the tree of shapes Ω. Then the rest of both trees are separated into their maximal branches. In Fig. 3.11 we see how a branch of $\mathcal{L}(u)$ is linked to a node of $\mathcal{U}(u)$ according to the rules described above for the branches of $\mathcal{L}(u)$ (of type "$\mathrm{Sat}(B)$ is linked to $\mathrm{Sat}(N)$"). We see also how a maximal branch of $\mathcal{U}(u)$ is linked to Ω (which is the saturation of a node in both trees). In this case, it can be considered as a

linking of type "Sat(B) is linked to Sat(B')" and also of type "Sat(B) is linked to Sat(N)". In Fig. 3.12 we see how a branch of $\mathcal{L}(u)$ is linked to another node of $\mathcal{L}(u)$ according to the rules described above for the branches of $\mathcal{L}(u)$ (of type "Sat(B) is linked to Sat(B')"). We see also how maximal branches of both trees $\mathcal{U}(u)$ and $\mathcal{L}(u)$ are linked to Ω. Observe that the trees of shapes obtained in both figures are equivalent.

Let S, T be two given shapes such that $S \subseteq T$. Let us prove that our construction of the tree of shapes by fusion of $\mathcal{U}(u)$ and $\mathcal{L}(u)$ has connected them. Observe that, since u is weakly oscillating, there are a finite number of leaves and a finite number of singular values. Hence, there is a finite number of singular shapes.

Lemma 34. *Let $S, T \in \mathcal{S}(u)$, $S \subseteq T$. Then there is a partition of $[S, T]$ into maximal monotone sections $\mathcal{M}_i$, $i = 1, \ldots, r$ such that $\mathcal{M}_i$ is at the left of $\mathcal{M}_{i+1}$ for each $i = 1, \ldots, r-1$.*

Proof. Observe that given $Q \in [S, T]$ there is a maximal monotone section containing Q. Since there is a finite number of them, we have $[S, T] = \cup_{i=1}^{r} \mathcal{M}_i$ where $\mathcal{M}_i$ are the maximal monotone sections contained in $[S, T]$. We recall that each $\mathcal{M}_i$ is an interval of shapes which may be open or closed at its ends. Notice that $\mathcal{M}_i \cap \mathcal{M}_j = \emptyset$, for any $i \neq j$. Indeed, if $\mathcal{M}_i$ and $\mathcal{M}_j$ are of the same type, then by Lemma 17 they cannot intersect. Since the only shape of upper an lower type is Ω, if $\mathcal{M}_i$ and $\mathcal{M}_j$ are of different type and they intersect, then Ω is the upper extrema of one of them, say $\mathcal{M}_i$, and the lower extrema of $\mathcal{M}_j$ (in which case $\mathcal{M}_j = \Omega$). In that case, $\mathcal{M}_j$ would not appear since Ω already appears in $\mathcal{M}_i$. Thus we may order them so that $\mathcal{M}_i$, $i = 1, \ldots, r$, constitute a partition of $[S, T]$ into intervals, $\mathcal{M}_i$ is at the left of $\mathcal{M}_{i+1}$ for each $= 1, \ldots, r-1$. □

Let us notice that a monotone section of upper (resp. lower) type of the tree of shapes is composed of saturations of nodes of a branch of $\mathcal{U}(u)$ (resp. of $\mathcal{L}(u)$). Indeed, if $[S, T]$ is a monotone section of $\mathcal{S}(u)$ and $S = \mathrm{Sat}(\mathrm{cc}([u \geq \lambda]))$, $T = \mathrm{Sat}(\mathrm{cc}([u \geq \mu]))$, $\lambda \geq \mu$, then the interval $[\mathrm{cc}([u \geq \lambda]), \mathrm{cc}([u \geq \mu])]_+$ is a branch of $\mathcal{U}(u)$ since it cannot contain any bifurcation of the upper tree. The presence of such a bifurcation would imply the presence of a bifurcation in the monotone section $[S, T]$, a contradiction. The same argument can be done for monotone sections of lower type.

Let us consider the interval $[S, T]$ in $\mathcal{S}(u)$. Let us consider two consecutive maximal monotone sections $\mathcal{M}$ and $\mathcal{M}'$ in $[S, T]$, the second one at the right of the first one. Let us consider four cases:

(*i*) $\mathcal{M}$ and $\mathcal{M}'$ are both of upper type. In that case, $\sup \mathcal{M} = \mathrm{Sat}(B)$ and $\inf \mathcal{M}' = \mathrm{Sat}(B')$ where $B \in \mathcal{CC}([u > \lambda])$ and $B' = \mathrm{cc}([u \geq \lambda]), B)$. Then there is a bifurcation at level λ and we are in the situation described in the fusion rules for $\mathcal{U}(u)$ where we linked Sat(B) to Sat(B').

(*ii*) $\mathcal{M}$ is of upper and $\mathcal{M}'$ of lower type. In that case, $\sup \mathcal{M} = \mathrm{Sat}(B)$ and $\inf \mathcal{M}' = \mathrm{Sat}(N)$ where $B \in \mathcal{CC}([u > \lambda])$ is a hole in $N \in \mathcal{CC}([u \leq \lambda]))$. Then

there is a bifurcation at level λ and we are in the situation described in the fusion rules for $\mathcal{U}(u)$ where we linked $\mathrm{Sat}(B)$ to $\mathrm{Sat}(N)$.

(*iii*) $\mathcal{M}$ and $\mathcal{M}'$ are both of lower type. In that case, $\sup \mathcal{M} = \mathrm{Sat}(B)$ and $\inf \mathcal{M}' = \mathrm{Sat}(B')$ where $B \in \mathcal{CC}([u < \lambda])$ and $B' = \mathrm{cc}([u \leq \lambda]), B)$. Then there is a bifurcation at level λ and we are in the situation described in the fusion rules for $\mathcal{L}(u)$ where we linked $\mathrm{Sat}(B)$ to $\mathrm{Sat}(B')$.

(*iv*) $\mathcal{M}$ is of lower and $\mathcal{M}'$ of upper type. In that case, $\sup \mathcal{M} = \mathrm{Sat}(B)$ and $\inf \mathcal{M}' = \mathrm{Sat}(N)$ where $B \in \mathcal{CC}([u < \lambda])$ is a hole in $N \in \mathcal{CC}([u \geq \lambda]))$. Then there is a bifurcation at level λ and we are in the situation described in the fusion rules for $\mathcal{L}(u)$ where we linked $\mathrm{Sat}(B)$ to $\mathrm{Sat}(N)$.

We have shown that each transition between two consecutive maximal monotone sections of $[S, T]$ has been linked when fusing the upper and lower trees. The path from S to T in $\mathcal{S}(u)$ is reproduced in the fused tree.

Proposition 35. *The fusion of $\mathcal{U}(u)$ and $\mathcal{L}(u)$ according to the fusion rules produces $\mathcal{S}(u)$.*

When $N = 2$ the method just described leads to an algorithm which is less efficient than the algorithm [Mon00] However, this algorithm is the most natural one and can be easily implemented when $N \geq 3$. The resulting algorithm for $N = 3$ is the main contribution of the first part of this thesis.

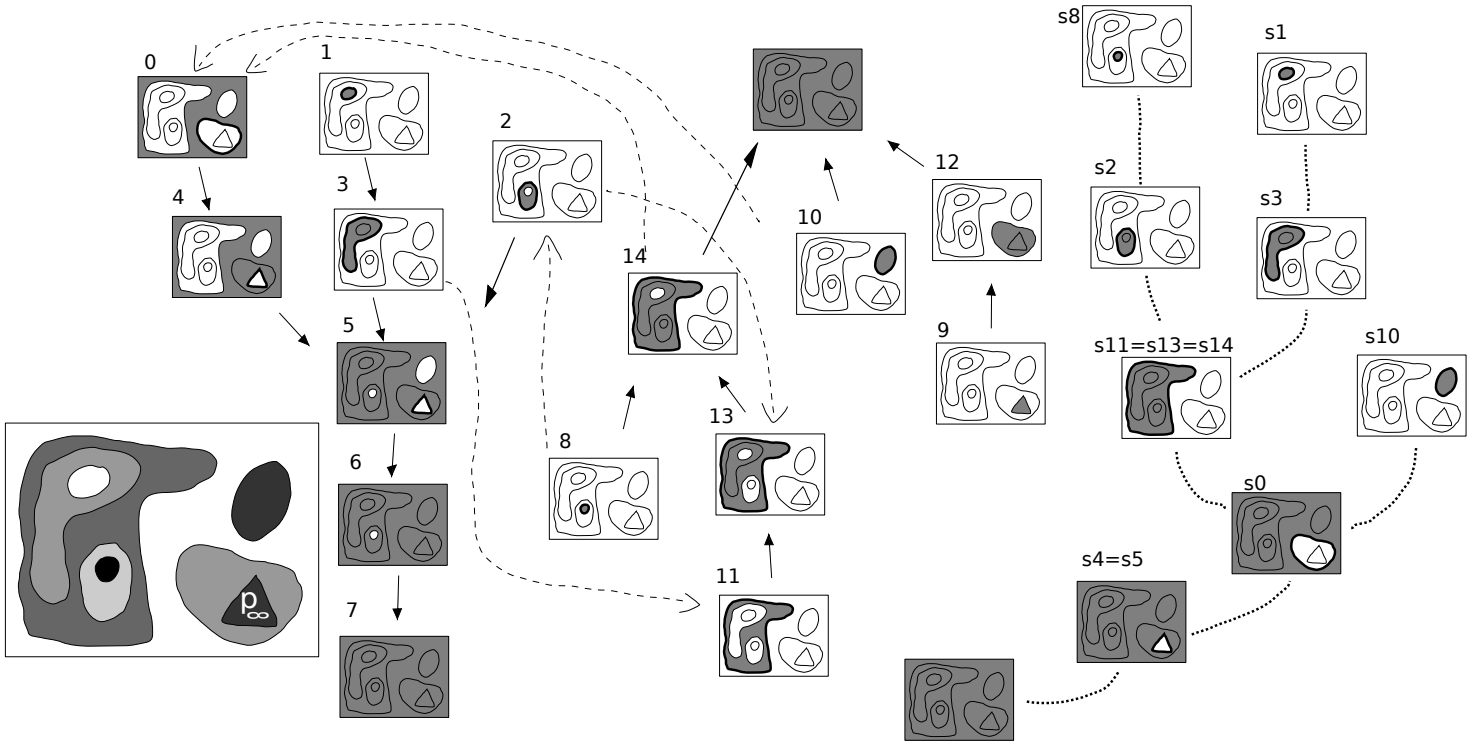


Figure 3.13: From left to right: a synthetic 2D image, its upper tree (with arrows down), its lower tree (with arrows up), and the set of shapes, which are saturations of upper and lower regions. To define the saturation we have used a point p_∞ inside the small triangular region. The saturation of each region is defined by its external boundary, marked in bold. Notice that the shapes have a tree structure, indicated by dotted lines. The external boundary of some orphan regions is shown by dashed arrows. The dotted lines of the tree of shapes come from either arrows of the original tree, or from the dashed arrows that define the external boundaries. (The numbers are only labels so that we can refer to them in the text.)

4 Tree of Shapes: The Implementation

The goal of this chapter is to explain how the mathematical construction given on Chapter 3 can be implemented on a computer. On Sections 4.1 and 4.2 we propose two different discretizations of the image domain: one based on digital geometry, and the other one based on graphs. On Section 4.3 we explain a data structure to store trees of regions, and on Section 4.4 we highlight some algorithms to construct and traverse these data structures. It is important to notice that the data structures and the algorithms are independent to the kind of discretization used.

4.1 First Discrete Approach: Geometry of Digital Images

In this section we talk about the geometry of voxels. This is a fairly straightforward generalization of the geometry of pixel-based images, described in the seminal work [RK82] of Rosenfeld and Kak, and on the more recent book [Her98] by Herman. The algorithm described to follow the border of a region is an adaptation of the algorithm [GU89] of Gordon and Udupa. The main conceptual difference with those other works is that the justification of the types of connectivity arises directly from the use of a semicontinuous interpolation, which is much simpler than the smoothly complicated constructions explained in [LPR93] that try to simulate directly the physical interpretation of figure 4.2.

Most of the content applies equally well to the 2D case. This means that, unless illustrating an idea specific to 3D, all the examples will be two-dimensional images without further commentary.

4.1.1 Voxels, facels, edgels and pointels

Let us suppose that we have an open rectangle of $\mathbb{R}^3$, called the domain, which is tesselated into equal-sized cubes. These cubes are called *voxels*. A voxel is bounded by at most six squares, its faces, called *facels*; by twelve segments, its edges, called *edgels*; and by eight points, its vertices, called *pointels*. The cubes centered at the pointels which have the same size and orientation as the voxels are called *dual voxels*. The vertices of a dual voxel are at the centers of voxels, and vice-versa. Notice that we have defined facels without orientation. Thus, a facel is simply a square, and two voxels in contact can share a facel. When we want to specify an orientation, we will talk about *oriented facels*. Given an oriented facel, we can talk about its *inner voxel* and its *outer voxel*. See figure 4.1 for an illustration of these concepts.

We will use two definitions of neighboring voxels. Given a voxel v, its *6-neighbors* are the voxels that share a facel with v. The *26-neighbors* of v are the voxels that share some pointel with v. Many properties that we will talk about are independent of the notion of neighborhood, and we will talk simply about M-connectedness, where M is either 6 or 26. Thus, hat the number of M-neighbors of a voxel is at most M, and it is exactly M when the voxel does not touch the

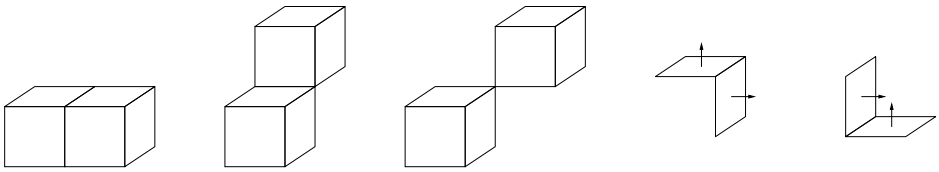


Figure 4.1: From left to right: Two voxels sharing a facel. Two voxels sharing an edgel. Two voxels sharing a pointel. Two oriented facels with the same inner voxel. Two oriented facels with the same outer voxel.

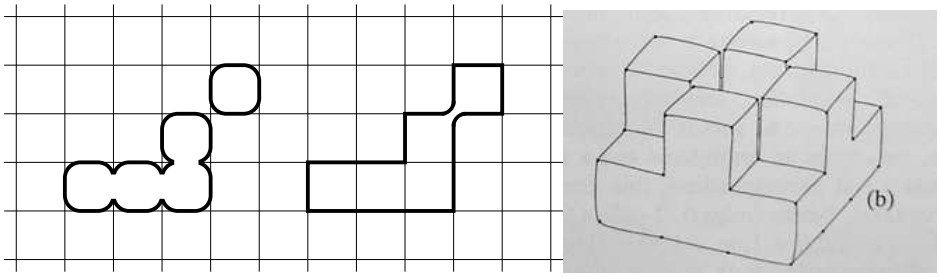


Figure 4.2: A physical interpretation of the two cases of connectivity. Left: a configuration of three voxels in 6-connectivity. Center: the same configuration in 26-connectivity. Right: a 3D configuration of voxels in 6-connectivity. Voxels in 6-connectivity can be understood as cubes with rounded edges, which can only touch each other at whole faces. Voxels in 26-connectivity can be understood as "wet" cubes, covered by a thin layer of fluid, so that when they slightly touch the surface tension of the fluid connects them clearly. (Right figure taken from [LPR93]).

boundary of the domain. See figure 4.2 for an intuitive graphical explanation of the two definitions of connectivity.

The relation of neighborhood permits to define the relation of connectedness (as its transitive closure) . A set V of voxels is called M-connected when for any pair of voxels $u, w \in V$ there is a sequence of voxels $u = v_1, v_2, \ldots, v_n = w$, all of them belonging to V, such that every pair of consecutive voxels v_i, v_{i+1} are M-neighbors. An M-connected set of voxels is called an M-region.

Border and boundary. Let R be a region. The *border* of R is the set of voxels of R that have some neighbor which does not belong to R. The *boundary* of R is the set of oriented facels whose inner voxel belongs to R and whose outer voxel does not. Thus, it coincides with the boundary of R as an oriented simplicial complex. Both the border and the boundary can have several connected components, with the boundary having at least as many components as the border. See figure 4.3 for an illustration.

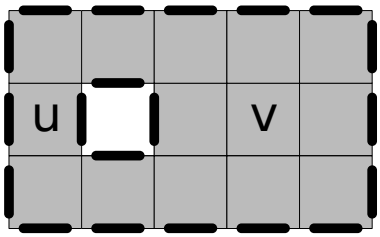


Figure 4.3: A region of voxels marked in gray in a 2D image. The *border* has one connected component and it is the set of all voxels of the region except the voxel marked v. The *boundary* has two connected components, and it is the set of all marked facels. Notice that the voxel marked u has one facel belonging to each component of the boundary. This example works with both types of connectivity.

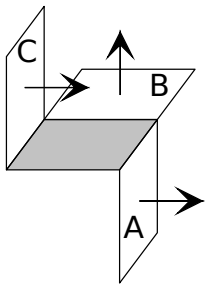


Figure 4.4: An oriented facel (pointing top, shown in gray) and three of its possible neighbors using 6-connectivity. There are altogether 12 of them, but usually only 4 of them will be actually realised as boundarying facels of a region. The labels are used to refer to the facels in the next chapter.

Connectedness of the boundary The definition of the connected components of the boundary is somewhat delicate. Let us treat first the case of 6-connectivity. Note that we have not defined what does it mean for two arbitrary oriented facels to be 6-neighbors. The decision whether or not two facels are connected depends on the voxels of the region, and not on the facels themselves. See figure 4.4 for an illustration of this. In any case, a given oriented facel has at most 12 candidates for neighboring facels. The case of 26-connectivity is subtler, because there is plenty of candidate oriented facels as neighbors. See figure 4.5. When we talk about the implementation of these ideas on section 4.4.2, where we describe an algorithm to follow the boundary of a region, we will explain a trick to avoid most of the complication that results from this.

Border and proper voxels It may be tempting to think that there is some relationship between proper voxels and voxels on the border. After all, when we start with a region D of the tree and add some voxels to it to obtain its mother M,

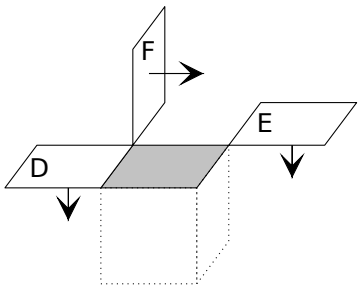


Figure 4.5: An oriented facel (pointing top, shown in gray) and three of its possible neighbors using 26-connectivity. There is a lot of them, but we will not need to compute all of them. The labels are used to refer to the facels in the next chapter.

D D D D D
D D D D D
D D M D D
D D D D D
D D D D D

Figure 4.6: A region M and its daughter D. The labels mark the proper voxels of each region. Note that the border of region M is composed entirely of proper voxels of D. This example works with both types of connectivity.

the region grows. And when a region grows, its seems that its border must also grow. But this is not true. It may happen that the proper voxels of M, the ones that we are adding, cover completely a hole of D and not belong at all to the boundary of M. See figure 4.6 for an example where this situation occurs..

4.1.2 The semicontinuous interpolation of a discrete image

The upper semicontinuous interpolation of an image is simply the nearest-neighbor interpolation, where ties are resolved in favor of the highest possible value. Notice that ties occur only at a set of measure zero[1], so that the interpolation is essentially the same as with any other resolution of ties. The only difference is the mathematical convenience of the model.

[1]The ties occur at places which are equidistant to two or more sampling points; that is, the faces, edges, and vertices of a cubical tessellation

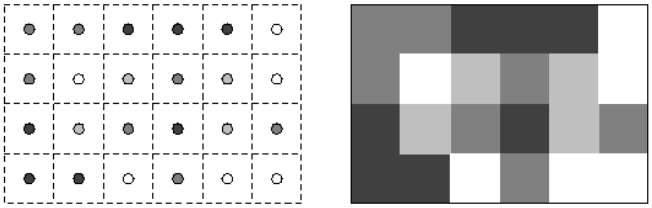


Figure 4.7: Semicontinuous interpolation in 2D. The sampling points have integer coordinates. The interpolated function takes the values of the nearest sampling point. Thus, the interpolated function is constant on squares, whose vertices have half-integer coordinates. (Figure taken from [Mon00].)

In the language of mathematical morphology, the upper semicontinuous interpolation is expressed as a dilatation of the sampling by a structural element which is a closed cube. Namely, let us suppose that we have a discrete image f_d, which is given by a three-dimensional matrix:

$$f_d : X_d = \{1, \ldots, W\} \times \{1, \ldots, H\} \times \{1, \ldots, D\} \to \mathbb{R}$$

and we want to define a semicontinuous function f_c over a rectangle of $\mathbb{R}^3$. We start with a preliminary interpolation

$$\tilde{f}_c : X_c = \left[\tfrac{1}{2},\ W+\tfrac{1}{2}\right] \times \left[\tfrac{1}{2},\ H+\tfrac{1}{2}\right] \times \left[\tfrac{1}{2},\ D+\tfrac{1}{2}\right] \longrightarrow \mathbb{R}$$

$$x \longmapsto f_d(x) = \begin{cases} f_d(x) & \text{if } x \in X_d \\ -\infty & \text{otherwise} \end{cases}$$

and then define f_c as the dilation of $\tilde{f}_c$ by a closed cube $B = \left[-\frac{1}{2}, \frac{1}{2}\right]^3$:

$$f_c(x) = \sup_{y \in x+B} \tilde{f}_c(y).$$

See figure 4.7 for an illustration of the semicontinuous interpolation in 2D.

The semicontinuous interpolation on cubical cells provides a justification of the common choice of connectedness: higher values are connected across diagonals through. This implies that we must use 26-connectivity for upper-level sets, and 6-connectivity for lower-level sets.

4.2 Second Discrete Approach: Topographic Graphs

In this section we study real-valued functions defined on the vertices of arbitrary graphs. Thus, if $G = (V, E)$ is a graph we will consider functions $f : V \to \mathbb{R}$.

Topological operations on graphs

Let $G = (V, E)$ be a graph. Any subset of vertices $A \subseteq V$ determines an equivalence relation $\equiv_A$ on A

$$p \equiv_A q \iff \text{there is a path of adjacent vertices of } A \text{ between } p \text{ and } q.$$

The classes of this equivalence relation are called *connected components* of A. The class containing the vertex p is denoted $\mathrm{cc}(A, p)$, and this notation is extended to $\mathrm{cc}(A, p) = \emptyset$ when $p \notin A$. If the equivalence relation has only one class, or equivalently $\mathrm{cc}(A, p) = A$ for any $p \in A$, we say that A is *connected*.

For a fixed $p \in V$, the operator $\mathrm{cc}(\cdot, p)$ has the following properties, which are immediate to prove from the definition:

Proposition 36 (properties of cc)**.** *Let $G = (V, E)$ be a graph, let $p \in V$ and let $A \subseteq V$. The following properties hold:*

1. $\mathrm{cc}(\mathrm{cc}(A, p), p) = \mathrm{cc}(A, p)$
2. $\mathrm{cc}(\mathrm{cc}(A, p), p) \subseteq A$
3. $A \subseteq B \implies \mathrm{cc}(A, p) \subseteq \mathrm{cc}(B, p)$

There are three natural definitions of the boundary of a set of vertices:

Definition 37 (boundaries)**.** *Let $G = (V, E)$ be a graph and let $A \subseteq V$. We define*

$$\partial A := \{(a, b) \in E \ : \ a \in A,\ b \notin A\}$$

$$\partial_{\mathrm{in}A} := \{a \in A \ : \ \exists b \notin A, (a, b) \in E\}$$

$$\partial_{\mathrm{out}A} := \{b \notin A \ : \ \exists a \in A, (a, b) \in E\}$$

We call them, respectively, the boundary, the inner boundary and the outer boundary of A. Notice that the boundary is a set of edges, while the inner and the outer boundaries are sets of vertices.

Now we define the saturation operator. The intuitive idea of this operator is that it fills the inner holes of sets. The holes are the connected components of the complementary, and the inner holes are those that do not contain a previously selected "vertex at infinity" p_∞. Thus, the definition of saturation takes a set and a point of infinity:

Definition 38 (saturation)**.** *Let $G = (V, E)$ be a graph, and let $A \subseteq V$ and $p \in V$. We define*

$$\mathrm{Sat}(A, p) := V \setminus \mathrm{cc}(V \setminus A, p).$$

When $p = p_\infty$ is fixed and clear from the context, we may omit it and write simply $\mathrm{Sat}(A)$.

The following proposition describes the main properties of the saturation operator on graphs. Compare this with 5. The properties are the same as those for the topological saturation operator, but with the notions that only make sense in the continuous case omitted.

Proposition 39. *Let $G = (V, E)$ be a graph such that V is connected, and let us fix $p_\infty \in V$. The following properties of the saturation operator hold:*

1. $\mathrm{Sat}(\mathrm{Sat}(A)) = \mathrm{Sat}(A)$
2. $\mathrm{Sat}(A) \supseteq A$
3. $A \subseteq B \implies \mathrm{Sat}(A) \subseteq \mathrm{Sat}(B)$
4. $\mathrm{Sat}(A) = A \uplus H_1 \uplus \cdots \uplus H_n$, *where H_i are all the connected components of $V \setminus A$ which do not contain p_∞ (the symbol $\uplus$ denotes a disjoint union).*
5. *In the previous decomposition,* $\forall i = 1 \dots n \ \exists a_i \in A \ \exists b_i \in H_i : (a_i, b_i) \in E$.
6. A connected $\implies$ $\mathrm{Sat}(A)$ connected
7. $\partial\,\mathrm{Sat}(A) \subseteq \partial A$
8. $\mathrm{Sat}(A) \neq V \implies \mathrm{Sat}(A) \subseteq \mathrm{Sat}(\partial_{\mathrm{in}} A)$

Proof. Properties 1–3 are immediate from the analogous properties for the cc operator. Property 4 is a re-interpretation of the definition of Sat. Property 5 is a consequence that V is connected. To prove 6 we have to build a path connecting any two points of $\mathrm{Sat}(A)$, using the edges given by property 5. Properties 7 and 8 are best understood by means of a figure (see Figure 4.8). □

Level sets on graphs

Once we have introduced the appropriate language of connected components and saturations, it is immediate to define the trees of level sets of functions defined on graphs.

Definition 40 (level sets on graphs, and their trees)**.** *Let $G = (V, E)$ be a graph, $f : V \to \mathbb{R}$ and $\lambda \in \mathbb{R}$. We define the upper and lower level sets of f, and the associated trees of subsets:*

$$[f < \lambda] := \{v \in V : f(v) < \lambda\}$$

$$[f \geq \lambda] := \{v \in V : f(v) \geq \lambda\}$$

$$\mathrm{LLT}(f) := \{\mathrm{cc}([f < \lambda], p) \ : \ p \in V, \ \lambda \in \mathbb{R}\}$$

$$\mathrm{ULT}(f) := \{\mathrm{cc}([f \geq \lambda], p) \ : \ p \in V, \ \lambda \in \mathbb{R}\}$$

$$\mathrm{TOS}(f, p_\infty) := \{\mathrm{Sat}(s, p_\infty) \ : \ s \in \mathrm{LLT}(f) \text{ or } s \in \mathrm{ULT}(f)\}$$

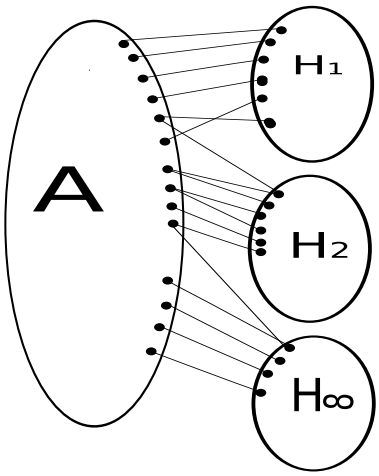


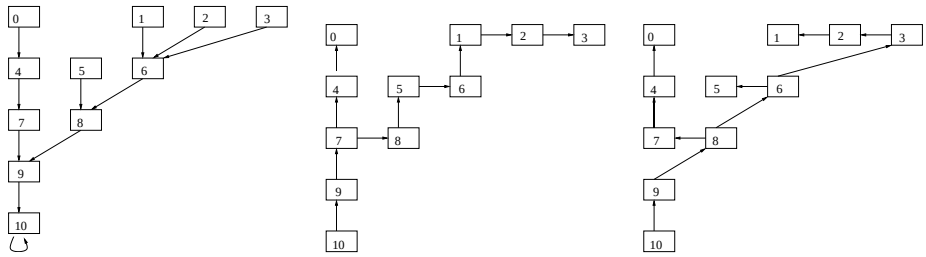
Figure 4.8: A set of vertices A and and the connected components of its complementary. Since all the connectivity information is on the edges of the graph, we can arrange the position of the vertices on the plane so that the connected components are manifest. This simplifies a lot the reasoning of the discrete case.

The main difference between these constructions for graphs and the corresponding ones for digital geometry, as developed on 4.1, is the lack of a nice property of the saturation operator: that the boundary of a saturated set is connected (5). In the case of digital geometry, this arises from the fact that the continuous image domain is unicoherent. Here, since graphs are not usually unicoherent, we do not have this nice property. Indeed, it is difficult to follow the boundary of a saturated set of vertices, because it may have several connected components.

4.3 Data structures for storing trees of subsets

On this section we describe a data structure to store trees of subsets of a finite set, as they have been introduced on definition 8. By *finite*, we mean that the base space Ω is a finite set, so that the tree of subsets $\mathcal{T}$ is a-fortiori finite. This data structure is independent of the kind of discretization (e.g., wether we have used the discretization of section 4.1 or that of section 4.2). In fact, the same data structure is used to store the trees of subsets that arise from Mumford-Shah segmentations (see Section 7.3).

Data structures for storing trees (graphs without cycles) are one of the basic building blocks of computer science [Knu73, Sed01]. They can store a tree of N nodes using $O(N)$ space. However, the data structures that we need, to store *trees of regions* are slightly fancier than that. In our case, each node of the tree of regions is actually a set of pixels. The trivial solution would be to attach to each



(a) Each node to its mother (b) Daughters and sisters (c) Daughters and sisters with different ordering

Figure 4.9: Different ways to represent the same tree

node of the tree a list of the pixels that it contains. However, this would waste a lot of space, because each pixel would appear many times. For example, the root of the tree contains all the pixels, and most of them also belong to other regions. An efficient storage scheme can be obtained by exploiting the fact that the subsets of pixels are nested. For example, if subset A contains subset B, the list of pixels of B can be simply a pointer to the middle of the list of pixels of A, provided that the pixels of A are suitably ordered. At the end, we only need to keep a single list of all the pixels, suitably ordered and all the nodes of the tree point to the appropriate place of this list. The root of the tree points to the beginning of the list. Using this technique, we can store a tree of subsets using $O(N + P)$ space, where N is the number of subsets and P is the number of pixels.

The previous construction allows to store the list of pixels of each node of the tree. Thus, given a node of the tree we can access its pixels immediately. The dual operation, given a pixel access the nodes it belongs to, is not immediate from this data structure. This structure can be augmented with an additional array of the size of the image, which contains the region indexes of the smallest region containing each pixel. The augmented structure is redundant, but provides constant-time access for any desired query. See 4.12 for an illustration of this redundant storage scheme.

4.3.1 Data structures for trees

There are two easy ways to represent a rooted tree on a computer. See figure 4.9 for an illustration. The first way requires one pointer per node, and the second way requires two pointer per node:

1. Each node points to its mother, and the root points to itself.

2. Each node points to a linked list of its daughters.

Each of these representations has some advantages over the other one. We have chosen to use both of them at the same time. This adds a useful but harmless

redundancy to the data structure that facilitates writing the algorithms. For example, we build our trees starting from the leaves, and we end up naturally with a representation of the first kind. To obtain the second representation, we use the following algorithm, that runs in linear time with the number of nodes.

```
foreach(p in nodes) {
        node m = p->mother;
        if (p != m) {
                p->next_sibling = m->daughter;
                m->daughter = p;
        }
}
```

An operation that we can implement easily thanks to this double representation is changing the root of the tree. We need this operation because the root of the tree of shapes depends on an arbitrarily selected point p_∞. If we wanted to use a different point we would need to change the root of the tree to the region that contained properly the new point p_∞. The fact that it is trivial to do so, means that the initial choice of that point is not really important. The following recursive function does the work.

```
void change_root(node r)
{
        node m = r->mother;
        if (r == m)
                return;
        unhook_node(r);
        change_root(m);
        <make m hang from r>;
}
```

4.3.2 Data structures for equivalence relations

Consider the abstract data type "collection of disjoint sets". The collections of disjoint sets can be seen as equivalence relations, because they determine a partition of a universe. We use this data type to represent how the upper and lower level sets merge as we decrease or increase the gray level.

There are three functions needed to work with the elements of the universe:

```
void dsf_make(element x);
```

creates a new set containing only the element `x`.

```
element dsf_find(element x);
```

returns the representative element for the set that contains the element `x`.

```
element dsf_union(element x, element y);
```

(a) Array of pointers (b) Forest

Figure 4.10: The disjoint set forest $\{\{1\}, \{2, 3, 7\}, \{5\}, \{4, 6, 8, 9\}\}$

performs the union of the sets that contain the elements `x` and `y`, and returns the representative element to the resulting set.

Disjoint set forest Note that there are many ways to implement this abstract data type. A specially terse and efficient one is the data structure called "disjoint set forest", which is the one used in our program (see figure 4.10 for a graphical depiction of this data structure).

In a disjoint set forest, each set is represented by a tree, and each of those trees is implemented using the "pointers to the mothers" representation described above (see section 4.3.1). Each node of a tree represents an element of the set. The root of the tree is the representative element of that set. Initially, every singleton is a tree with one single node. If the whole universe is known since the beginning, then the disjoint set forest can be implemented simply as an array of pointers, which is started by making each pointer to point to itself.

For example, if the elements of the universe are indexed by the integers from zero to nine, then we use an array of ten pointers. Initially, all the pointers are null. The function call `dsf_make(3)` makes the pointer at position 3 to point to itself. The function call `dsf_find(x)` follows the pointers starting from `x` until it finds a one pointing to itself which is returned. The function call `dsf_union(x,y)` first finds the representative of `x` and `y`, and then makes one of them point to the other one.

The Tarjan heuristics When equivalence relations are implemented as arrays of pointers, some nasty stuff can happen after many union operations. After a lot of unions, depending on the order in which we performed them, we can end up with long strings of pointers. Those long strings are a problem because they make the function calls to `dsf_find` very expensive. See figure 4.11. On the other side, when the unions are performed in a convenient order, we can make all the pointers to point directly to their representative. This is convenient because then the function `dsf_find` is as cheap as it can be.

The heuristic *union-by-rank* says that, when making the union of two trees, the one which is shallower has to hang from the root of the deeper, and not the other way round. This minimizes the depth of the resulting tree.

(a) Bad order of calls (b) Good order of calls

Figure 4.11: The tree structure depends on the order of the calls to the set-union function.

Another important improvement is called *path-compression*. It consists in the following: every time that a string of pointers is traversed, all the pointers of the string are forced to point to the representative element. That way, successive calls to `dsf_find` keep shortening the depths of trees.

Combining both *union-by-rank* and *path-compression*, we observe that the depth of the trees does not grow very much (never higher than 5). Thus, we can consider that all the operations run in constant time. The analysis of this neat algorithm is described originally at [Tar75] and with full detail at [CLRS01], chapter 21.

Our heuristics A little detail to take into account is that if we use *path-compression*, then an exact accounting of the depth of each tree is no longer possible. What is still possible, however, is to keep an upper bound of the depths, which is already sufficient. In our implementation, we use *path-compression* but we do not use *union-by-rank*. Instead, we use a variant of it that could be called *union-by-graylevel*. Recall that we use disjoint set forests to merge the components of upper and lower level sets as they grow. Then, our heuristic uses the graylevel of each levelset to decide the order of the unions. We are not aware of any formal analysis of this heuristic. The only reference we have seen to it was in a free software library of computer vision [Bru], where it is implemented. Buried inside its source code, we find the following comment:

```
Used to be union by rank with path compression, but
now it just uses path compression as the global
parent index seems to be a faster approximation of
rank in practice.
```

4.3.3 Data structure for trees of regions

So far we have talked about implementing trees as combinatorial structures, now the time has come to talk about implementing trees of regions of voxels (as defined on section 4.1).

The main idea to represent such a tree is to separate the combinatorial information (the nesting of the regions) from the geometrical information (which voxels belong to which region). There are two ways to do that, which are in some

sense dual. Either we list for every region which voxels does it contain, or we list for every voxel to which region does it belong.

(a) Each node of the tree points to an array with the coordinates of its voxels.

(b) Keep an array of pointers to regions, of the same size as the image, indicating the smallest region that contains each voxel.

Each of these representations is complete, meaning that the original image can be completely recovered from either of them. Then, keeping both structures is redundant, but can be nonetheless very convenient, because each representation favors a different kind of query. Spacewise, both representations can also be stored efficiently. The array of pointers has a size proportional to that of the image. The array of voxels also has a size proportional to the image, because nested regions can share part of their arrays. At the end, only the array of voxels of the root is needed, ordered in such a way that the other arrays can be simply pointers to their slice of the big array. As there are linear time algorithms to pass from one representation to the other, there is no need to save both of them. We adopted the following convention: we save only the image of pointer to smallest regions, and compute the list of voxels whenever we need it.

This data structure can be expressed easily in the `C` programming language. We need a `struct` for the nodes of the tree, and another `struct` for the tree itself. See figure 4.12 for a graphical visualization of how to store a simple tree in memory using this data structure.

```
struct region {
        // combinatorial information
        struct region *mother, *daughter, *sister;

        // geometrical information
        int volume, proper_volume;
        int (*voxels)[3];

        // user-supplied data
        float value;
        void *data;
};

struct tree_of_regions {
        int size[3];
        int number_of_regions;
        struct region *the_regions;
        struct region **smallest_region;
        struct region *root; // (points somewhere inside the_regions)
}
```

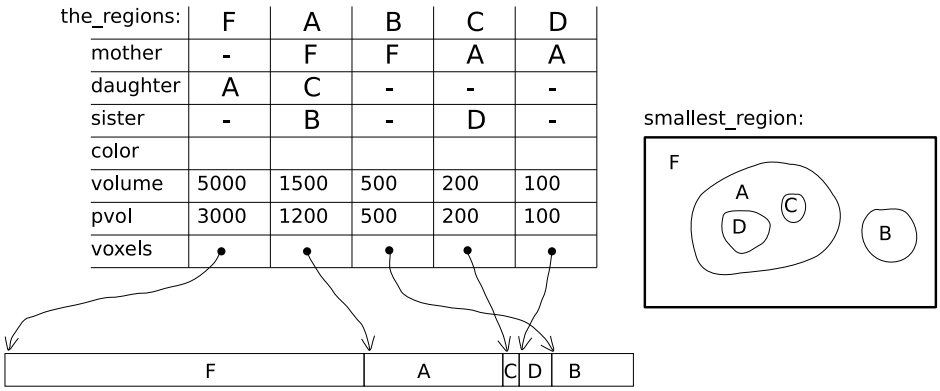


Figure 4.12: Redundant storage of a tree of shapes.

The two structures described above (that is: the tree plus the geometrical information) have a size which is always at least that of the original image. There is a third possibility, that may produce smaller sizes for simpler images, specially those that are synthetic or heavily quantized:

(c) For each region, have a list of the oriented facels of its boundary.

In the worst case, the storage required for this representation is quadratic in the size of the image, so it is not practical in general. We never store all the boundaries of the regions, and we only compute them as required.

Thanks to the redundant representation, the following operations on trees run in time proportional to their output, usually constant time, so they are as fast as possible:

- Given a voxel, return the smallest region it belongs to
- Given a region, list all the voxels it contains
- Decide whether one region is an ancestor of another region
- Decide whether a voxel belongs or not to a region
- Decide whether two regions are disjoint or nested
- Given two nested regions, list all the regions in-between

The first two queries come directly from the representation. The first decision can be implemented by comparing the volumes of the regions with the positions of their voxels in the array of all voxels. The other decisions can be described using the ancestor relation. Only to show that the data structure is very easy to use, let us display the implementation of these functions. See Chapter 5 for examples of real and useful applications.

```
// decide whether region a is an ancestor of region b
bool ancestorP(tree_of_regions *t, region *a, region *b)
{
        int ia = a->voxels - t->root->voxels;
        int ib = b->voxels - t->root->voxels;
        return (ib > ia) && ((ia + a->volume) > ib);
}

// decide whether voxel of index i belongs to region a
bool belongsP(tree_of_regions *t, region *a, int i)
{
        region *s = t->smallest_region[i];
        return (r == s) || ancestorP(t, r, s);
}

// decide whether regions a and b are disjoint
bool disjointP(tree_of_regions *t, region *a, region *b)
{
        return !ancestorP(t, a, b) && !ancestorP(t, b, a);
}
```

4.4 Algorithms

Our main algorithm is the computation of the tree of shapes. There are two strategies to build it. The first strategy, used to construct two-dimensional trees of shapes [Mon00], builds the tree directly from the image data. This is possible using some topological assumptions that do not hold in 3D. Another strategy is to build first the upper and lower trees of the image, and then join into the final tree of shapes, thanks to the way they are related. This second strategy is more general, and it is the one needed to build contour trees; see [CSA03] and [PCM03]. Our choice for the 3D tree of shapes is the general strategy.

4.4.1 Building the upper and lower trees

Here we describe an algorithm to build the upper tree of an image of voxels. We do not talk about building the lower tree because the algorithm is completely analogous. To get the algorithm for the lower tree the comparison between gray levels has to be reversed and 26-connectivity has to be replaced by 6-connectivity.

Let $u : X \to \mathbb{R}$ an image of voxels, and suppose that we can traverse the voxels decreasingly by graylevel (for example, if they are sorted beforehand). The algorithm to build the upper tree starts with the regions of maximum level, and makes them grow as the level decreases. Thus, the overall structure of algorithm is a loop over the different possible values of the voxels, in decreasing order. While the program runs, it keeps track of the new regions that appear and the mergings that occur between already existing regions. Meanwhile a tree is built to keep track of the new regions and the mergings between regions: the leafs of

the tree correspond the new regions, and the nodes of the tree with more than one daughter correspond to the merged regions. At the end, all regions have been merged into a single one, which corresponds to the root of the tree. To keep track of the regions we use the disjoint set forest data structure described in section 4.3.2. This structure allows us to keep a representative voxel of each region, and to merge two regions in (almost) constant time.

Let us write the algorithm in some detail. It needs two data structures, which are interleaved: a tree of regions (see section 4.3.3) and a disjoint set forest, which are both initially empty, and which end up filled with all the voxels of the image domain. It is a loop over all the graylevels g appearing in the image, starting from the highest. Before each iteration, the following invariant holds:

> **Invariant of the loop:** The tree of regions is completely built for all the graylevels in $(g, +\infty)$ and each region of the DSF has a representative which has the lowest gray value of the region it represents.

At each iteration of gray level g, we make two passes through all the voxels of value equal to g. Let $y_1, \ldots, y_n$ be the set of voxels whose value is g. Notice that when we add these voxels, some regions may be merged, some new regions may appear, and some regions may grow at different places (see figure 4.13). We traverse this set of voxels twice, the first time to build the new pieces of levelset, and the second time to connect those pieces to the tree of shapes:

```
1st pass through the y_i:
        add y_i to the DSF
        for each neighbor v of y_i:
                if v is already on the DSF:
                        dsf_union(y_i, v)

2nd pass through the y_i:
        p = dsf_find(y_i)
        if p is already on the DSF:
                r = region of the tree containing p
        else:
                r = new region of the tree with value g
        add the voxel y_i to the region r
        for each neighbor v of y_i
                if v has value higher than y_i :
                        s = largest region containing v
                                hang region s from region r
```

4.4.2 Following a boundary of a connected binary region

In this section we briefly describe an algorithm to traverse the boundary of a binary region. More precisely, the algorithm traverses all the facels of a connected component of the boundary of a connected region. There are two versions of this

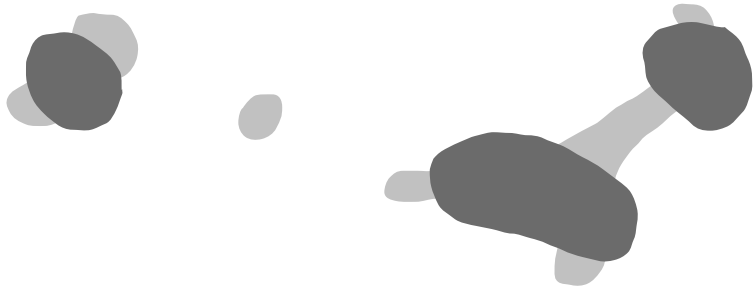


Figure 4.13: A step in the construction of the upper tree. The part which is already constructed is marked in dark gray. The new voxels of value g are marked in light gray. The white background are those voxels not yet considered. Notice that the new voxels may merge existent regions, create new regions, and grow existent regions at different places.

algorithm, depending on whether the binary region is considered using 6 or 26-connectivity. The implementation of both algorithms is identical except at one place where the type of connectivity is used to compute the neighbors of a given facel. They are flood-fill-like algorithms.

Flood-fill is the name that receive those venerable algorithms that change the color of a connected region of pixels of the same color in an image. These algorithms often appear inside painting programs, to paint regions with a new color chosen by the user. Here we need them to traverse all the facels of the border of a region starting at a user-supplied facel. The ideas are the same, but the implementation is a little more cumbersome due to the fact that the facels are not on the same plane.

Flood-fill algorithms are essentially graph-traversing algorithms. What happens is that the graphs they work with are not directly represented in memory, but are implicitly represented by some other data structure, such as an image. The interest of these algorithms lies in the fact that they can operate without building the whole graph. This means that they can run using time and space proportional to the output. For example, in a painting program, the time that it takes to fill a small region does not depend on the size of the whole image, only on the size of the region.

From the point of view of graph theory, the following algorithm is simply a breadth-first search starting from the given facel. However, we do not have the graph explicitly, but implicitly. The graph is built alongside its traversal.

Input

- A choice of connectivity, either 6 or 26.
- A boolean 3D image, which can be given explicitly as an image with values in $\{0, 1\}$ or implicitly, as a boolean function that we can evaluate at each

voxel. We call R the (not necessarily connected) set of voxels with value 1.

- An oriented facel f_0 such that its inner voxel belongs to R and its outer voxel does not.

Data structures

- A hash table to store uniquely the set of facels we have already traversed.
- A FIFO queue to store the facels that we are planning to visit.

Algorithm

1. Put f_0 into the queue.
2. Put f into the hash table.
3. While the queue is not empty:
   a) Pick a facel f from the queue.
   b) Compute the set S of facels neighboring f in the boundary of R.
   c) For each facel $g \in S$: if g is not yet inside the hash table, put it there and into the queue.

Output The set of voxels contained in the hash table.

The only nontrivial step in the algorithm is 3(b), because the definition of "neighboring facel in the boundary" depends on the region R, and the kind of connectivity. To understand that, take a look again at figures 4.4 and 4.5. The case of 6 connectivity is analogous to the case of 4-connectivity in 2D, but instead of having neighboring edgels in two directions, we have neighboring facels in four directions. This means that it is easy to write a function that looks locally at the boolean image and returns the list of the four neighboring facels. In 26-connectivity, besides all the cases coming from 6-connectivity, it is not necessary to consider all the new neighboring facels. It is sufficient to traverse those of the forms labelled D and F in figure 4.5, and there are four of each form. Facels in a position such as E will be visited later as the 6-neighbors of already visited facels.

We use the boundary-following algorithm in three different places:

Boundary and border of a region If we have a tree of regions, we can pick a region of it, a boundarying facel of this region, and produce the set of facels of the c.c. of the boundary that contains the given facel. This is done by applying the boundary-following algorithm to *the boolean image defined by the region*. If we want the set of voxels of the border instead of the facels, these appear as the inner voxels of the set of facels.

Output-sensitive marching cubes The standard implementation of the marching cubes algorithm (see Section 7.1) traverses a whole image to extract an isosurface, which usually has several pieces. The running time is independent of the surface and is proportional to the size of the whole image. If, on the contrary, we are only interested in one single piece of surface, we can run our boundary-following algorithm to *the boolean image defined by the threshold that crosses a given voxel*. For each voxel we traverse, we compute the triangulation of the corresponding cell. The union of all those triangles gives a triangulation of a single piece of isosurface, which is computed in time proportional to its size. That way, we can navigate efficiently through the level surfaces of an arbitrarily large image.

Triangulation of the boundary of a region This third case is a combination of the other two. We traverse a boundary of a region of the tree, but instead of storing the facels, we compute the triangulation of the corresponding cell. At the end, we obtain a triangulated surface that approximates smoothly the boundary of a region, with the correct topology. (Well, at least smoother than the set of facels).

4.4.3 Algorithm for joining the trees

Let $u : X \to \mathbb{R}$ an image of voxels and suppose that we have already computed its upper and lower trees, as described in 4.4.1 (the upper regions considered in 26-connectivity and the lower regions in 6-connectivity). We will describe an algorithm to build the tree of shapes of u from this information. We recall as needed the results from the previous chapters.

Proposition 41. *If a set F of facels is a c.c. of the boundary of an upper region s, then the same set of facels with the opposite orientation is a c.c. of the boundary of some lower region s' (and vice-versa). The regions s and s' are called* adjacent*, or* adjacent through F*.*

Observation 42. *The relation of adjacency through F is not unique; that is, there can be different pairs of regions which are adjacent through the same F. See, for instance, figure 3.13, where the lower regions numbered* 13*,* 14 *and* 15 *are adjacent to the upper regions* 0 *and* 4*, in all possible combinations, through the same boundary F.*

Observation 43. *We have an algorithm to traverse the c.c. of the boundary of a region that contains one given facel of it.*

Now let us describe the algorithm to compute the tree of shapes. We are given as input the upper and lower trees of the image, and a voxel $p_\infty \in X$. This defines a saturation operator over X. The c.c. of the boundary of a region s containing the facel which is nearest to p_∞ is the external boundary of f. The region adjacent to s through this c.c. is called the *outer adjacent* region of s. The algorithm to join the upper and lower trees has two steps: first we **cut** both trees

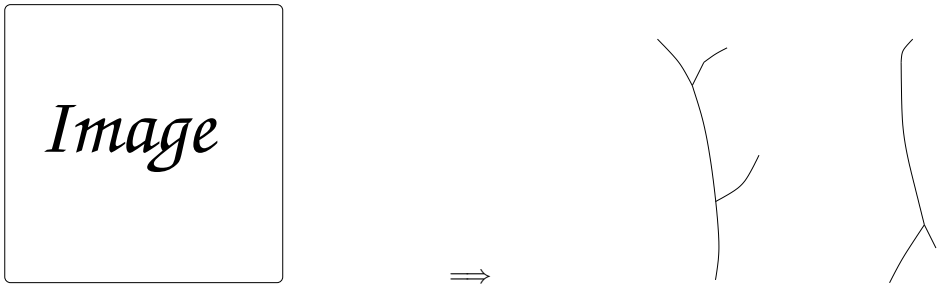


Figure 4.14: First step in the construction of the tree of shapes: we build the upper and lower level set trees of the image.

Figure 4.15: Second step in the construction of the tree of shapes: we identify the background as the set of shapes that contain p_∞ (dotted line) and the orphan regions (black dots) in both trees.

into monotone branches, and then we **paste** the branches altogether to form the tree of shapes.

Cutting into branches The algorithm needs to be able to label some regions with marks, like "*removed*" and "*orphan*". These marks are implemented as flags inside the data structure for a region. The algorithm starts as follows (see figures 4.14 and 4.15):

1. For both trees, mark the region that contains p_∞ and all its ancestors as removed. This removes all the regions of both trees that contain p_∞, that is, all the regions whose saturation is the whole domain.

2. Mark all the regions whose mother has been removed as orphans.

3. Mark all the regions that have any sister as orphans.

These three operations split all the monotone branches of both trees, and leave us with a disconnected set of branches. Note that the oldest region of each branch is marked as orphan. The next step, as suggested by the following proposition, will be to assign a mother to each orphan region.

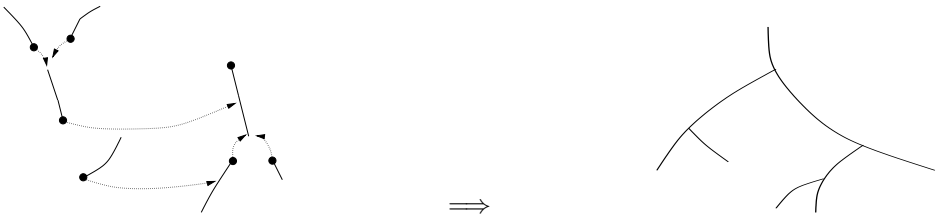


Figure 4.16: Third and last step in the construction of the tree of shapes: we find a mother for each orphan region.

Proposition 44. *The tree of shapes can be obtained from this set of branches hanging each orphan region somewhere, either to another region or to the root (and identifying the equal regions that result after that).*

Proof. This is simply a rephrasing of Proposition 35. □

Joining the branches To complete the construction of the tree, we have to assign a mother to each orphan region. There are two kinds of orphan regions, those that are orphans because their mother was removed, and those that are orphans because they had sisters. In each case, there are two possibilities for the new mother. In the first case, the new mother can be either the outer adjacent or the root. In the second case, the new mother can be either the outer adjacent or the old mother. In both cases, the new mother will be the smallest region of the two possibilities. Figure 4.16 shows an example of the join of two trees where all these possibilities appear.

There are two things to notice:

- To compute the outer adjacent of a region we follow its external boundary using the algorithm described in section 4.4.2. This is only necessary for orphan regions.
- To compare which of two regions is larger we use the constant-time tests of section 4.3.3.

Efficiency of the algorithm We have not performed yet a formal analysis of the theoretical cost of the algorithm to build the tree of shapes. What we can prove easily is that the worst-case cost is at most quadratic in the number of voxels. This comes from the following situation: if all regions of both trees happen to be orphan, then we will have to compute the boundary of all the regions, and the sum of boundary sizes of all the regions is less than the square of the total number of voxels. However, it is not clear whether this extreme possibility can actually happen. In our experience, most of the shapes do not become orphan and the algorithm runs in a reasonable time (always less than a minute for large 2D images, and less than 10 minutes for medical images of sizes up to 100^3). In

synthetic or heavily quantized images, the program runs immediately, because they have a very small number of shapes. The worst case seems to be images with a lot of different values (e.g, floating point values) and with a lot of noise.

5 Tree of Shapes: First applications

This chapter explains four simple, direct and elementary applications of the 3D Tree of Shapes. These applications are merely entry points into much deeper, untreated here, fields of study. The goal of this chapter is not to develop these applications, but to showcase the kind of high-level algorithms that are easy to express using the proposed data structure.

5.1 Self-dual morphological filters

Natural operations with the tree of shapes As we have said in the introduction, each image representation favors some operations. The tree of shapes, being built from the level sets of the image, is well-suited to the application of morphological operators, which are defined over level sets. However, the tree of shapes is by no means necessary, or even the best choice, to implement most morphological operators. For example, median filtering, erosion, dilation and their combinations can be easily computed on an image represented as an array of pixels. There are, still, some filters which are immediate to compute using the tree of shapes, yet very difficult without it. Here we comment two of them: grain filters [CM02] and branch simplification. It is thanks to the 3D tree of shapes that we can easily compute grain filters on 3D images. Besides morphological, these operations are auto-dual, that is, invariant with respect to inversion of contrast.

Grain filtering Grain filtering consists in removing the level sets of an image whose size is smaller than a fixed threshold t. By "size", we refer to area in 2D or volume in 3D. Its effect is similar to the application of an opening followed by a closing using a ball of radius t as structuring element, or a closing following by an opening. These combinations are known as extrema killers. Since opening and closing do not commute, there are two kinds of extrema killers, dual to each other under contrast inversion. An advantage of grain filters over extrema killers is that they are self-dual, and more efficient for large values of t. Grain filters can be also compared to median filtering. A difference, which may be regarded as an advantage or a disadvantage depending on context, is that median filtering changes all the level sets of the image (smoothing their boundaries), whereas grain filters only remove small level sets, leaving others unchanged. See Figures 5.1 and 5.2 for examples of grain-filtering on 2D images, and Figure 5.3 for an example on 3D images.

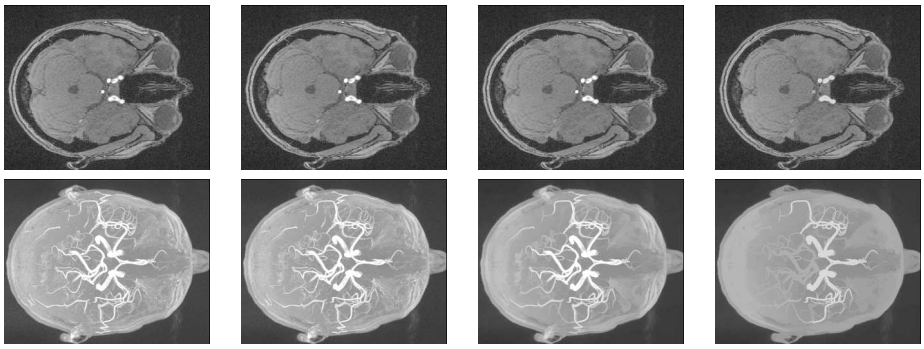
Shape selection Grain filtering is a particular case of *shape selection*, which in turn can be regarded as an extreme case of local contrast changes with step functions. Shape selection is based on an arbitrary criterion to select or reject any shape. The general procedure of shape selection is the following: for each shape of the tree, if it does not satisfy the criterion, mark it to be removed. Then, all non-removed shapes are used to reconstruct the filtered image. Shape selection results in a reduction of the number of level lines, and this it is a simplification of

original image | opening | closing | grain filter

Figure 5.1: From left to right: Original image, opening, closing, grain filter. For this image, a similar effect can be obtained by median filtering or by an extrema killer. The advantage of grain filters is that their speed is independent of the size parameter (unlike the size of structuring elements or windows for the other transforms).

original image 18351 shapes | t=2 11173 (61%) | t=10 4924 (27%) | t=1000 715 (4%)

Figure 5.2: Grain-filtering in 2D, at different grain sizes



original image 295937 shapes | t=2 141005 (47%) | t=100 2414 (0.8%) | t=10000 124 (0.04%)

Figure 5.3: Grain-filtering of a 3D image, at different grain sizes. First row: central slice. Second row: projection.

the image. Notice, however, that the level lines which are kept are not changed, regardless of how complicated they might be. Grain filtering is the particular case of shape selection when the criterion is a threshold on the size of the shape.

Branch simplification Another particular case of shape selection is *branch simplification*. It consists in removing all the shapes of each branch except a representative central one. A naive, but useful, criterion for picking the representative is to choose the shape which is most similar to the neighboring shapes on the branch (this means that the difference of volume between neighboring shapes is the smallest). Whatever the criterion, branch simplification produces a tree where all the branches have length one. The visual effect of this filter on images is that some smooth edges are sharpened, because large branches correspond to stacks of many parallel level sets, which usually occur on smooth edges. Probably, the main interest of branch simplification is not as much image filtering but as automatic selection of interesting shapes. For example, some iterations of grain filtering and branch simplification quickly reduce a 3D medical image of millions of shapes to a representative set of a few shapes. See figure 5.4 for a visual example of branch simplification and grain filtering.

Further development Shape selection can be used as a strategy for image compression, particularly well suited to control the supremum norm. For that, we can use a criterion to select shapes such that the reconstructed image has minimum error and is encoded in a minimum space. This is discussed in detail elsewhere [Igu06, BCIG07].

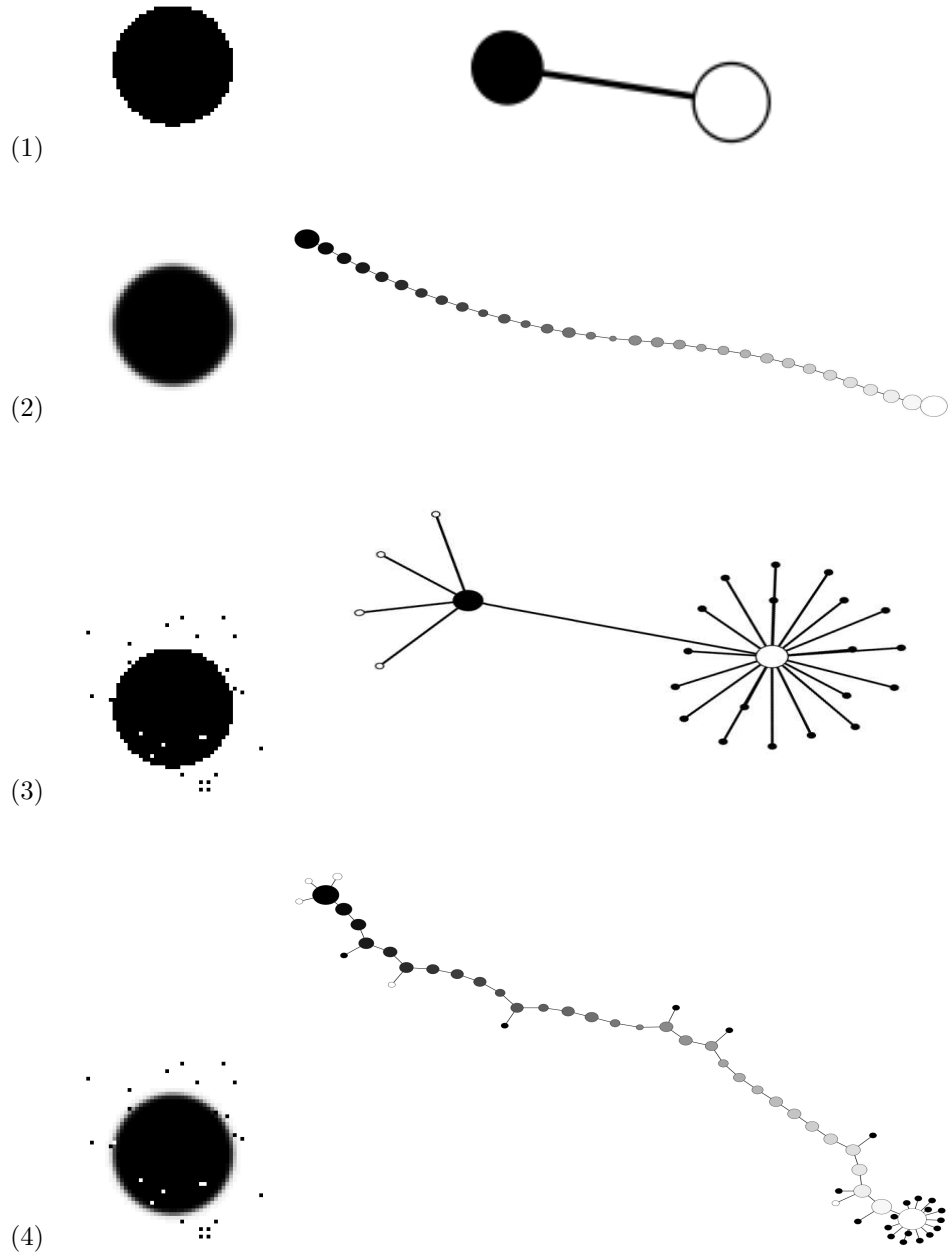


Figure 5.4: Simple images and their trees of shapes. Grain filtering turns (3) into (1) and (4) into (2). Branch simplification turns (2) into (1).

5.2 Visualization of images

The problem of visualization So far, we have not paid much attention to the visualization of 3D images, despite the fact that they can not be directly printed on paper. We displayed slices, or voxels, or isosurfaces, and expected that the content of the image was understandable from the figure. However, practical visualization of 3D images is an extremely interesting and difficult problem *per se*, in striking contrast with the 2D case, were the visualization technique does not usually go further than gamma-adjustment. The goal of this section is to explain how the tree of shapes can be used as an aid to visualization.

Standard methods of visualization The first observation is that the whole information content of a 3D image can not be printed into a 2D paper or screen. This means that, except for very simple images, the visualization is necessarily an interactive process. Thus, there is a contrast between the passive act of viewing a 2D image, where the user merely stares at the data; and the active act of viewing a 3D image, where the user must move some slides to rotate, crop, slice or select isosurfaces from the data. See Figure 5.5 for some examples of these techniques. All these techniques are standard, and they are part of any software that deals with 3D images, such as medical imaging packages, microscopy, molecular chemistry, etc. [ITK, MAYAVI, AMIRA]. Probably, the most convincing of these methods is volume rendering, where the image contents are treated as transparencies on the volume. By editing the mapping from image values to transparency levels (the so-called *transfer function*), and rotating the image domain, volume rendering provides a glimpse of the whole image content which is quite satisfactory even for moderately complex images.

Shape navigation The tree of shapes can be used as an addition to these standard techniques. The most immediate improvement is to enhance global image thresholding by what we call *shape navigation*. Let us explain that. In the standard setting, when the user selects a threshold, an isosurface is produced. By moving the threshold inside its one-dimensional range, the user sees different isosurfaces. Now, any one of these isosurfaces may have many connected components. Each of these connected components is the boundary of a single shape, which corresponds to a node in the tree of shapes. If the image is stored using using its tree of shapes, then this shape can be selected, independently of the other components of the same isosurface. Once a shape is selected, its parent and child shapes can be traversed iteratively, all the way through the branch the original shape belongs to. This is analogous to moving a global threshold, but here the user only sees a single connected component, ignoring the rest of the image. Moreover, when the navigation reaches the bottom of a branch (being stuck in shape that has more than one child), the user is given the choice of which branch to take, by default the one with largest volume. Notice that both thresholding and shape navigation are one-dimensional traversals, but thresholding is linear and shape navigation has ramifications. Besides visualization, shape navigation

can be used to define quickly a region of interest on a large data-set. With some practice and a suitable user-interface (see Figures 5.6 and 5.7), we have found that it can be as fast as a hand-made crop.

Shape editing With the aid of a proper interface, the tree of shapes provides many operations to view and select parts of the image, or do some simple editing to it. All the following operations are available:

- view the tree structure, and traverse it
- display the boundary of the "current" shape
- select a point in the image and find its corresponding shape
- highlight monotone branches
- apply morphological operators on the data
- manually select shapes
- select pieces of shapes, automatically and manually
- display histograms and statistics related to the current shape and current branch (ideally, this data should be displayed inside the tree structure, to easily find shapes with some extremal property inside their branch)
- overlay with common visualizations (volume rendering, projections, thresholds)

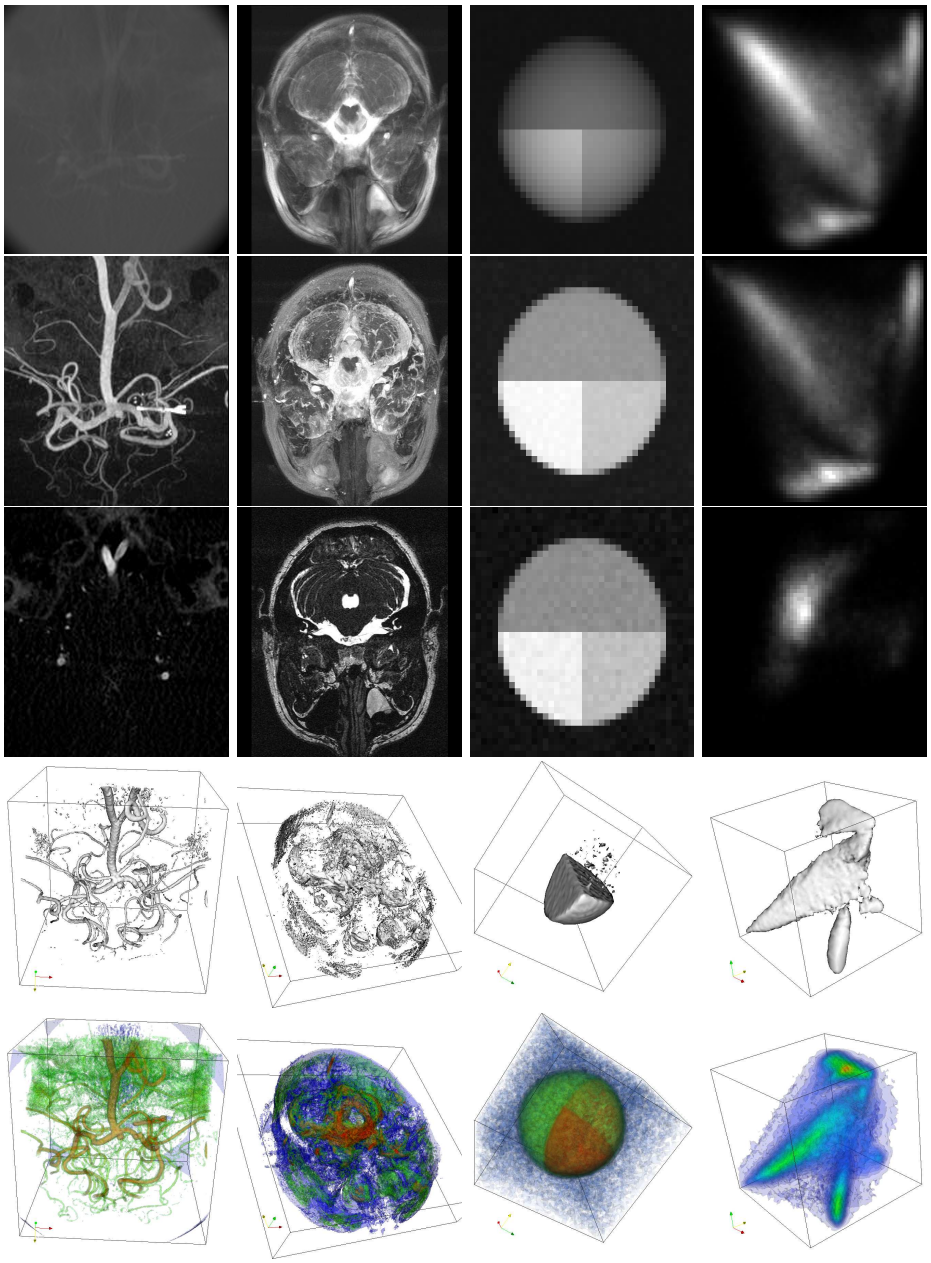


Figure 5.5: Some images of different kinds, as viewed by different methods. *First column:* tomography of inner brain vasculature. *Second column:* magnetic resonance of head tissues. *Third column:* synthetic image. *Fourth column:* density of a color histogram. *First row:* orthographic average projection. *Second row:* orthographic maximum projection. *Third row:* orthographic slices (requires interaction for selecting the slices). *Fourth row:* isosurface (requires interaction for selecting the isosurface, and rotating the domain). *Fifth row:* volume rendering (requires interaction for editing the transfer function, and rotating the domain).

▲	go to the parent shape
▼	go to the largest child
▶	go to the next sister (if any)
⤒	go to the largest shape of the current branch
⤓	go to the smallest shape of the current branch
◀	go to the previous sister (if any)
⏮	go to the largest sister

Figure 5.6: Commands for shape navigation. The first three commands suffice for navigation. The rest of the commands are there for convenience. In a user interface, these commands can be mapped to keys or to mouse movements. With some practice, the commands become rather automatic and the user can browse through huge 3D images to find whatever features she wants to.

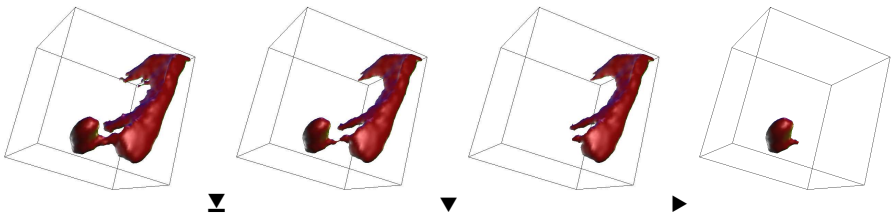


Figure 5.7: Example of shape navigation to explore the blobs of a density function. The navigation starts at the middle of a branch, then moves to the bottom of that branch (notice the singularity), and then traverses the top of the two branches after the singularity.

5.3 Color histogram analysis

Color histograms as 3D images Color histograms are a common source of 3D images. We can use the tree of shapes (or the tree of upper level sets) as a tool to analyze these histograms. Many operations on this tree of shapes correspond to operations on the colors of the original image. We discuss briefly the construction and visualization of useful color histograms, the analysis of these histograms using our data structure, and a simple color segmentation method based on the branches of the tree. The techniques are a direct generalization of the corresponding methods for gray-level images (see Figures 5.8 and 5.9).

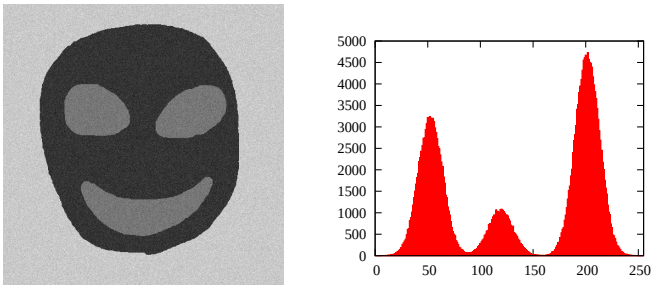


Figure 5.8: A gray image and its 1D histogram. The three main colors of the image appear as three large modes of the histogram.

Figure 5.9: A color image and its 3D histogram, using our proposed visualization based on the three of shapes. The five main colors of the image are clearly displayed as large blobs.

Construction of color histograms Given a standard 24 bit RGB image, its color histogram is defined in a natural way as a gray-level 3D image of size 256^3. The value of the pixel at coordinates (i, j, k) is the number of pixels on the original image whose RGB components are (i, j, k). There are some practical problems with this natural definition. The first problem is the large size of the histogram image, which has 2^{24} voxels, needing about 64MiB to be stored. The second problem is that it is usually very sparse, even for very high resolution input images. This means that its values are not useful as a density in color-space. This problem is often aggravated by artifacts such as JPEG compression, which reduces the number of bits to encode each color channel, so that some colors of the RGB can never appear. The third problem is that the histogram image may have a huge dynamic range, specially for synthetic images, or for real images with a synthetic part such as a black surrounding frame. The first and second problems are related, and can be dealt with by quantizing the RGB color space to, for example 64 values for each component, and smoothing out the resulting histogram by a small kernel (e.g., a cube of size 3^3). This process of smoothing and downsampling results in smooth histograms which can be interpreted as color densities, for most images, except the very small ones, or synthetic ones. To avoid the third problem, concerning the high dynamic range of the histogram, we store the logarithm of the pixel counts. Notice that the operations of smoothing and taking the logarithm are not at all commutative. On Figure 5.10 we display graphically the sparsity and dynamics before and after applying the post-processing of histograms described on this paragraph.

Figure 5.10: Downsampling and smoothing a color histogram to obtain a useful color density. Left: central slice of histogram before processing. Right: central slice of histogram after processing.

Visualization of color histograms, standard methods Besides the general methods for visualizing 3D images (namely: slices, isosurfaces, average projec-

tions, extrema projections), there are some methods which are specific for color histograms. These methods are designed for highlighting the highest values of the image, which correspond to the more frequent colors. For full-color histograms of size 256^3, the simplest method is to display a dot for each nonzero value of the histogram, with the corresponding color. Since color histograms tend to be very sparse, this is usually a small cloud of points inside RGB cube. Rotating the cube, the user can see easily the regions with higher density of points. A variation of this is to draw each dot as a sphere whose radius is proportional to the number of pixels of the corresponding color. This variation enhances the visibility of the large blobs. For smoothed color histograms which can be interpreted as a density, it is useful to look at their level surfaces. The thresholds for the level surfaces can be selected automatically or by hand, to emphasize the blobs of large values. Apart from these methods to view 3D histograms, which in theory display the whole information of the histogram, it is common to display three 1D histograms, one for each color channel. This is clearly less informative, because the combinations of channels are lost. An intermediate option is to display three 2D histograms, one for each pair of color channels. All of these common methods are illustrated on Figure 5.11.

Visualization of color histograms using trees Let us describe a method to display color histograms as 2D color images, whose output seem to be immediately understandable by many people. Once we have a useful color density, as described on the previous paragraph, it makes sense to compute its tree of shapes (or its tree of upper level sets, which almost always coincides, because color densities do usually do not have local minima). The tree of shapes is a planar graph, and it can be drawn by any standard graph-drawing algorithm [TT86, KK89] . By "drawing", we mean assigning to each vertex of the graph a pair of coordinates (x, y) so that if we locate each vertex at its assigned coordinates in the plane, the resulting drawing looks good. Graph drawing is essentially a heuristic process, and for color histograms have chosen the algorithm `neato` from the standard package `Graphviz`. Once graphviz has decided the coordinates of each vertex, we draw each vertex as a coloured disk, using the properties of the color region that corresponds to this vertex. The color of the disk is the average color of the pixels inside that region; and the radius of the disk is the logarithm of the number of pixels inside the region. The resulting 2D images contain, in theory, all the information of the original 3D histogram. Moreover, the user can see at a glimpse which are the main colors of the histogram, by looking at the thickest branches of the tree of shapes. See Figure 5.12 for some examples.

Applications of gray-scale histograms In the case of gray-scale images, the most immediate application of histograms is to help with thresholding and quantization. Histogram-based *thresholding* is a common, if somewhat rude, method to segment gray-scale images. It is consists in partitioning the one-dimensional gray scale into two intervals, and classifying the pixels of the image according to which of these intervals their color belongs. This can be regarded as a par-

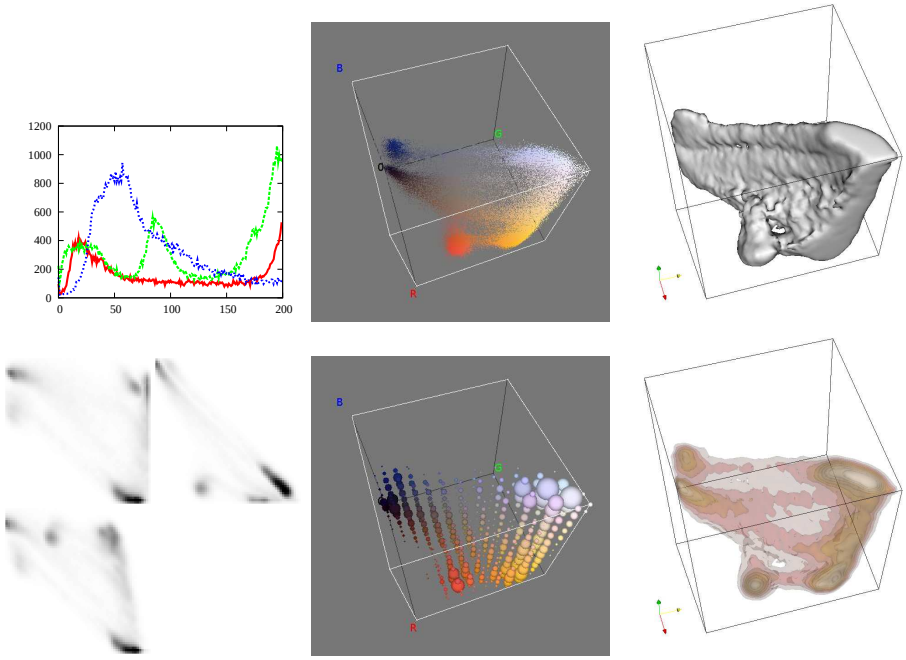


Figure 5.11: Some common ways to display the same color histogram. (a) a 1D histogram for each color channel (b) a 2D histogram for each pair of channels (c) a dot for each color that appears in the image (d) a sphere for each color, with volume proportional to the number of occurrences of that color (e) single level surface (f) multiple level surfaces, with transparency

ticular case of *quantization*, whereby the gray-scale is partitioned into a number of intervals, called quantization steps. There are three natural ways to obtain the quantization steps: 1) a fixed number of equal-lenght intervals, resulting in a blind quantization that does not depend on image contents; 2) a fixed number of equal mass intervals, resulting in an adaptive quantization of the image; 3) one interval for each mode of the histogram, resulting in a different kind of adaptive quantization.

Color segmentation (quantization) by selecting branches The operations of thresholding and quantizing color images are easy to define abstractly, copying the definitions for gray-scale images. However, the actual techniques are necessarily much more complicated, for they entail the segmentation of a 3D volume in color-space instead of a 1D interval in gray-space. Depending on the context, color space segmentation is called *quantization* or *palette building*. The tree of shapes, by representing the density function as a one-dimensional structure, enables the generalization of some thresholding and quantization tech-

niques to color images. The branches of the tree are analogous to the modes of the histogram. This gives rise to a naive method to build a coarse palette: for each terminal branch of the tree, assign to all the colors of this branch the same representative color, such as the most frequent color among them. The palette obtained this way has as many colors as large modes in the 3D histogram. To obtain a finer palette, we can proceed by subdividing the largest branches into two or three sub-intervals. Each sub-interval corresponds to a hollow region on color space of nearly uniform density. This region can be easily subdivided into any number of equal-mass parts to refine the coarse palette.

Further development The color space manipulations described above are arguably very primitive. However, the study of color densities by the tree of shapes is not yet seriously developed. Some of the most poignant issues that have to be solved are the following. First, we have to understand the effects and problems of the subsampling, quantization, smoothing and logarithms used to produce densities out of scattered data. Second, we have to use color spaces more meaningful than RGB, and understand how this affects the outcomes of visualization and palette building. Since color space coordinates are generally continuous, the effect of changing the color space is akin to re-sampling the original colors, but this should not change the topology of the density function, apart from quantization errors. The main effect of changing the color space is that it changes the metric by which we measure the sizes of color densities. For example, a uniform color density in RGB is not at all uniform in HSV. This effect can be seen on Figure 5.13, where the near-black colors are clearly over-represented in HSV. It has been argued that the best color spaces for color histograms are the perceptually uniform spaces [SPK02]. Finally, a third issue that must be understood is how to perform branch selection and partition specially tailored to histograms (as opposed to the branch partition that we could use for general 3D images). See also [VBVV07] for a similar topological approach to the analysis of color histograms based on its local differential descriptors.

Figure 5.12: The tree of shapes of color histograms, as visualized by a standard graph-drawing program (`neato`). These pictures are the graphs of the tree of shapes, suitably *decorated*. Each node is a coloured to the mean color of the corresponding region on colour space, and the radius of the disc is the (logarithm of) the pixel count inside that region.

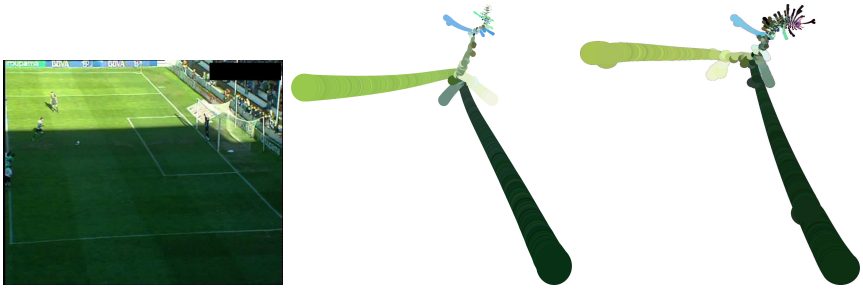


Figure 5.13: Visualization of RGB and HSV histograms. The purpose of this experiment is to asses the robustness of the visualization with respect to the downsampling and smoothing of histograms. Since the RGB-HSV map is continuous, both densities have the same topology, and their trees of shapes should be the same. Any difference must come from quantization and resampling errors. Visually, we see that the difference is small (the main branches of both histograms are equivalent), thus we can conclude that the visualization is quite robust.

5.4 Optical flow analysis

We have seen on Section 4.1 that it is natural to assume 8 and 26-connectivity for upper level sets and 4 and 6-connectivity for lower level sets. This is natural in absence of further knowledge about the desired connectivity. However, sometimes there is further knowledge about the connectivity in the form of a graph that connects pairs of neighboring data points. In that case, it makes sense to extend the data structures and algorithms above to functions defined on the vertices of arbitrary graphs. Video analysis is one of this cases: besides the usual connectivity of pixels on each single frame, we have a natural notion of connectivity between pixels on consecutive frames given by the *optical flow*.

Given a video sequence we can build the appropriate graph in the following way. First, we pre-compute a dense optical flow of the whole video sequence using any of the methods available the literature (we tried some of them with similar final results [BBPW04] [BBM09], [KGC10] [PB10]). This flow assigns a vector on every pixel of each frame but the last. Now, the vertices of the graph are defined as all the pixels of the video, each one assigned its corresponding gray level. The edges of the graph are of two kinds: spatial edges and temporal edges. Spatial edges join each pixel with its 8-neighbors on the same frame. Temporal edges are defined using the pre-computed optical flow: If the flow vector on pixel (x, y, t) is (u, v), then we add to the graph an edge joining pixel (x, y, t) with pixel $(x + [u], y + [v], t + 1)$, where the square brackets denote the nearest integer.

On the following subsections we explain how the trees of level sets of this graph are a flexible and useful data structure. This data structure, which we call "the tubes", encodes temporally coherent segmentations of all the objects on the video, which can be used for tracking. The tubes are useful to write higher level algorithms on the video: as an example we provide a simple method for monocular depth estimation. This method can be improved using better segmentations than the bi-level sets used here, and by combining the results with other depth cues. However, the data structure is exactly the same, and here we only want to showcase its power.

5.4.1 A data structure for segmented videos

We have just described how to build a graph from a given video sequence. Since the vertices of this graph are assigned gray-level values, we can compute the tree of level sets of this graph, as explained on Section 4.2. Pruning and simplifying the branches of this graph corresponds to applying temporally coherent morphological operators (cf. Section 5.1) to the video sequence. A selection of regions of this tree can be regarded as a segmentation of the video. For example, by selecting very few regions we obtain a very coarse segmentation of the video into temporally coherent bi-level sets. We call the segments of this segmentation the "tubes" of the video. The intersections of tubes with frames are called "regions". Thus, the regions of a given frame are segmentation of that frame. Alongside the tree, we can store the region-adjacency graph of the segmentations of each frame. That way, we can follow a given region in time, see how its neighbours change in time,

etc. Even if the segmentations are not very good (preferably, they will be over-segmentations), it is much easier to deal with a few hundred regions, having the evolution of their connectivity stored in a small graph, than to deal with millions of unstructured pixels.

Specifically, the data structure contains four lists of objects:

1. list of all the pixels
2. list of all the frames
3. list of all the tubes
4. list of all the regions

The list of pixels and the list of frames are trivially related, because each frame contains the same number of pixels. The relationships among the other lists are the true interest of the data structure, since they hint a combinatorial representation for the video objects. See Figure 5.14 for a diagram illustrating the relative inclusions between these structures.

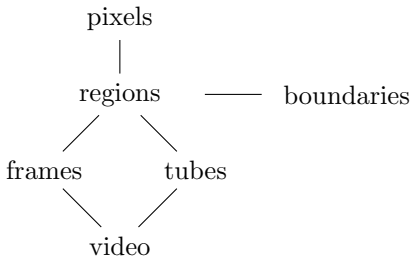


Figure 5.14: Inclusion relationships between the parts of the data structure. A video is divided into frames, and into tubes. The intersection of a tube with a frame is a region. Each region is a set of pixels. Neighboring regions on the same frame are separated by their common boundaries.

5.4.2 Simple computations using the tubes

The mere act of storing a video sequence using "the tubes" lends itself to certain higher level algorithms, which provide raw analysis of the objects that appear in the video. Here we list four of these algorithms. The algorithm for relative depth from motion will be an example of a more complex one.

Tube statistics The simplest thing that we can do with the tubes is to compute statistics of all the regions. For each tube, we can see how its area evolves along frames, its mean motion (to select immediately the fastest moving objects in the video), the evolution of the length of its boundary, etc.

Tube topology The tubes can be classified by their topology, looking how it evolves in time. The simplest case is that of a tube which intersects each frame (from a certain interval of frames) in a single connected region. A different case is that of two objects that merge or split as time passes, for example when one object occludes another one of the same color. In that case, the tube has the shape of the letter Y, with the junction appearing at the frame where the objects merge or split. By single traversal of the data structure, we can build a list of the branched and unbranched tubes, and of the regions that they span along the video.

Optical flow regularization We can user the structure of tubes and regions to improve a given dense optical flow. If we suspect, as often happens, that the optical flow is wrong or imprecise near the boundaries of the objects, we can discard those samples of the dense flow, and extrapolate their values from the inner parts of the region. This is a regularization of the optical flow in a single frame. But we can also smooth the flow along several frames, to enhance its temporal consistency. The connectivity of the regions assures that we will not be mixing flow samples from different layers of movement.

Flow from segmentation As an extreme case of the previous computation, we can construct an optical flow from scratch, just by looking at evolution of the tubes in time. If we find the best match from each region into the next one, we already have a model of the movement of that region. By sampling that motion on the pixels of the region, we produce a dense optical flow. While it is very crude, this method does not depend on the resolution of the video, only on the structure of the tubes. Thus, it can be used as a starting point for more precise algorithms. The quality of the results depends ont the criterion for registering pairs of consecutive regions. A naive criterion that minimizes Hausdorff distance (or that matches the center of mass) will produce incorrect results for occluded objecs, but perfect results for objects which are on top of all the others, and move in a plane perpendicular to the line of view. In some circumstances, this may be useful.

5.4.3 Monocular depth estimation: perceptual principle

Depth perception from a single image is an easy task for the human visual system: people who have lost an eye can lead a normal life, and everybody can easily reconstruct real-world scenes from a single photograph. According to current theories of vision [Kan79, Mar82, YB96] this is achieved by integrating the information of several *depth cues*. There is a rather large list of cues for depth perception, including perspective, texture gradients, distance fog, focus, T-junctions, shading and size. Each of these cues is not sufficient alone, and any single one may lead to incorrect depth perception. However, combining the information from all these cues produces very reliable information. In computer vision, it is easy to obtain information from each of these cues, but difficult to integrate the information from all of them into a single 3D reconstruction.

Depth perception from multiple images adds new cues to this list, thus increasing the reliability of the depth information. The most prominent addition to the list is *parallax*, which can be produced either by binocular perception or by observer movement. A different cue, closely related to our purpose, is *depth from motion*, whereby objects moving towards the observer increase in size, and objects moving away from the observer decrease in size. The brain is very fast and very precise in using this information to compute the crash time of approaching objects. The change of size can be expressed locally by means of the divergence of the optical flow: the optical flow of an approaching object will have positive divergence, and the optical flow of a distancing object will have negative divergence. This idea has been used successfully for collision avoidance in free-moving robots [NA02]. Notice that this is a local criterion which works on the interior of objects, and tells how the objects move in the direction of the observer.

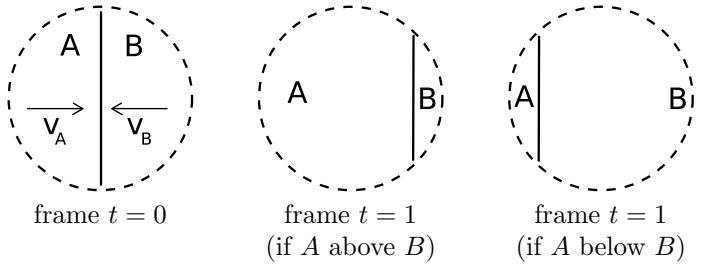
Here, we introduces a new simple cue for depth perception from multiple images. Unlike depth from motion, this cue provides a local criterion which works on the boundaries between objects, looking how the objects move in the direction perpendicular to the observer, and telling how the objects are located in the direction of the observer. The cue is based on the fact that the boundary between two objects moves in the same way as the object which is closer to the observer (because the closest object occludes the other behind that boundary). In terms of divergences, an occlusion boundary produces a band of highly negative divergence around it, and a disocclusion boundary produces a band of highly positive divergence around it. Compared to parallax or depth from motion, the proposed criterion is more general because it does not assume the rigidity of the objects (although our naive implementation does). On the other hand, it only gives a relative ordering of the objects, not a distance.

Now, let us state more formally the promised perceptual principle. Let us assume that we have a perfectly computed dense optical flow and a perfect segmentation of the video frames into objects. In that case, the following criterion provides a relative ordering of neighboring objects: *The boundary between two moving objects in the scene follows the movement of the object which is closest to the camera.* See Figures 5.15 and 5.16 for two examples of this criterion. We assume that the criterion is intuitively true and no further explanation is given beyond these two figures.

Actually, the criterion is not true in full generality. There are some situations where it leads to an incorrect depth ordering (see Figure 5.17). Also, when there are shadows or reflections, most optical flow methods will rarely ignore their movement, as they should. For practical purposes, we will ignore these cases. It is up to the user of the method to decide whether these counterexamples are relevant for his intended application.

5.4.4 Monocular depth estimation: Algorithm

After stating the perceptual criterion for monocular depth estimation, we introduce an algorithm that uses it to compute a relative depth ordering of a video.



frame $t = 0$ | frame $t = 1$ (if A above B) | frame $t = 1$ (if A below B)

Figure 5.15: Local illustration of the criterion, around a point located on the boundary between A and B. Region A moves in the direction v_A, region B moves in the direction v_B. The boundary between the two regions moves in the same direction as the region which is above.

The algorithm is based on the "tubes" data structure described above (Subsection 5.4.1).

The algorithm works by selecting all pairs of neighboring regions, and making a decision on which region of each pair is above or below the other. Suppose that we have two neighboring regions A and B. Let us define the following notation (see also Figure 5.18:

- A_i is the region A on frame $t = i$, for $i = 0, 1$
- B_i is the region B on frame $t = i$, for $i = 0, 1$
- c_i is the boundary between A_i and B_i, for $i = 0, 1$
- R_0 is the model of movement between A_0 and A_1
- S_0 is the model of movement between B_0 and B_1

Notice that if there are no occlusions and the models of movement are correct, then we have $R_0(A_0) = A_1$ and $S_0(B_0) = B_1$. Thus, since the transformations are continuous, it must be that $R_0(c_0) = c_1$ and $S_0(c_0) = c_1$. This implies that $S_0 = R_0$, both movements are parallel to the boundary. When there are occlusions, the movements of A and B differ. The criterion 5.4.3 states that c_1 is the image of c_0 under the movement of the object which is above. Thus, comparing $R_0(c_0)$ and $S_0(c_0)$ to c_1, we can decide which of A or B is above.

There are many choices to be made when implementing this criterion for digital videos. Let us suppose, as above, that we have been given a spatio-temporal segmentation and a dense optical flow. To turn the criterion into an algorithm we must make precise two things: how to move a boundary by a given optical flow, and how to follow that movement. The first thing is easy to define, using the given segmentation. For each region of the segmentation, we build a model of its rigid movement using the given optical flows on its interior. This model can be either projective or affine, using all or a selection of the optical

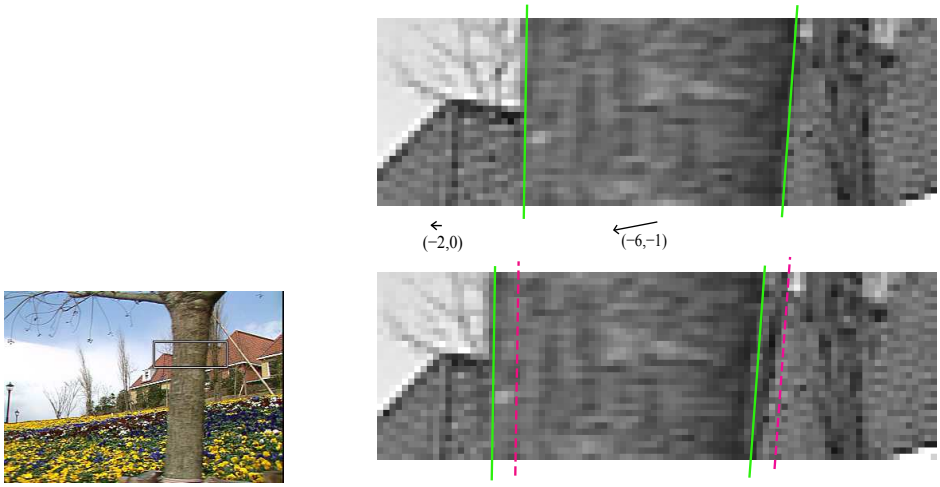


Figure 5.16: Local illustration of the criterion. Top: first frame, with segmentation boundaries in green. Bottom: second frame, with segmentation boundaries in green. The mean flows of each region are shown. The mean flow of the tree correctly moves the boundaries from one frame to the next. The mean flow of the background moves the boundaries to another location (dotted lines). According to the criterion, both boundaries of the tree support the hypothesis that the tree is above the background.

flow vectors within that segment. Since we are using over-segmentations, the segments tend to be small and thus affine models are enough.

The computation of the action of movement over a boundary is more delicate. A first choice is to notice that c_0 is a sampled curve, and so are $R_0(c_0)$ and $S_0(c_0)$. Then, we can compare each of these sampled curves to c_1. This comparison is illustrated on Figure 5.18. This first choice is apparently simple, but cumbersome to implement, because there is no natural choice of curve comparison in that case. A second choice is to measure the overlapping of the displaced regions using Hausdorff distance (area of symmetric difference). Thus, we compare $R_0(A_0)$ to A_1 and $S_0(B_0)$ to B_1. The pair which matches better will correspond to the region which is above. The advantage of this second choice is that it can be implemented very easily in linear time, by moving each pixel in the video according to the movement of its region, and looking whether the image goes to the corresponding region on the next frame, or to a different region. This comparison is illustrated on Figure 5.19. Here follows the pseudo-code of the algorithm:

Input: a spatio-temporal segmentation of a video and a dense optical flow F.

Output: a relative ordering of pairs of neighboring regions of the segmentation.

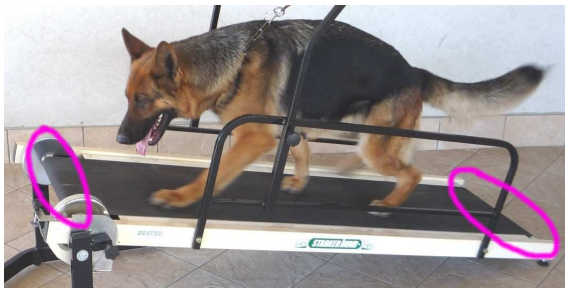


Figure 5.17: Counterexample to the criterion: flat flexible object folding behind a corner. In that video sequence, if the optical flow is correctly computed, the criterion gives a wrong relative ordering on the marked areas.

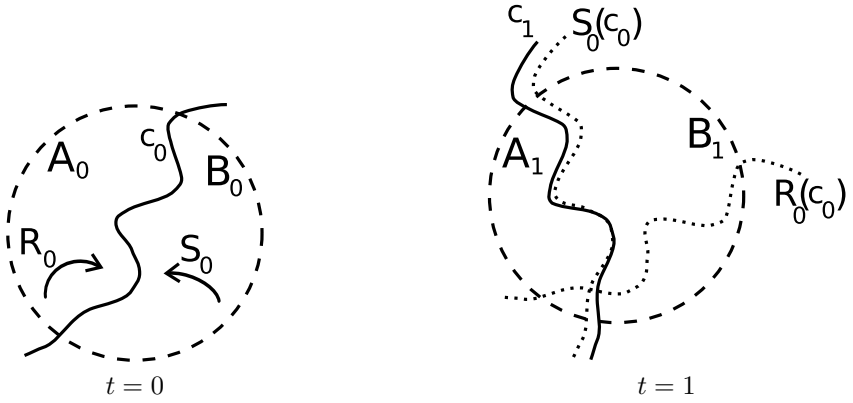


Figure 5.18: Notation used on the description of the criterion. The two figures depict the local situation around a boundary c separating regions A and B. Since c_1 is closer to $S_0(c_0)$ than to $R_0(c_0)$, this means that S_0 is a better model of movement of the boundary c_0, thus that region B is above region A.

Algorithm:

- for each region A_t on frame t:
  - $R_{A_t,A_{t+1}}$:=`movement_model`(A_t, F)
- for each pixel p on frame t:
  - A_t := `region_of_pixel`(p)
  - $q := R_{A_t,A_{t+1}}(p)$
  - B_{t+1} := `region_of_pixel`(q)
  - if $B \neq A$:

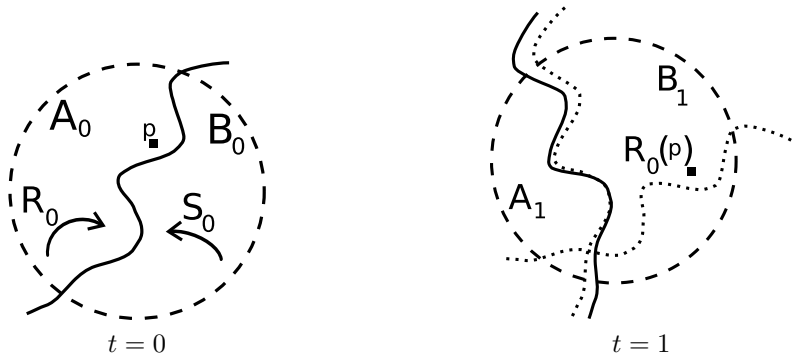


Figure 5.19: Notation used on the description of the algorithm. The two figures depict the local situation around a boundary separating regions A and B. Point p belongs to region A_0, but $R_0(p)$ belongs to B_1. This means that B is occluding A. If A was above, we should have $R_0(p)$ belong to A_1.

$*$ vote $+1$ that B_t is above A_t.

Interpretation of the algorithm. We move each pixel of the video according to the model of movement of the segment A it belongs to. If it falls in a different segment B, that means that B is occluding A, and we record this fact. See Figure 5.19 for a graphical explanation.

First remark on the algorithm. The output of the algorithm is a list of votes for each pair of regions, saying which one is above. By setting a threshold on difference of votes (e.g., 1), we obtain the desired partial ordering.

Second remark on the algorithm. As it is, the algorithm only finds *occlussions*, but not *disocclussions*. To obtain those, we must run it "backwards in time". This can be done by using either a bi-directional optical flow, such as that of [ADPS07], or by naively inverting the movement of each region $R_{A_{t+1},A_t} := R^{-1}_{A_t,A_{t+1}}$.

Third remark on the algorithm. The algorithm gives a relative ordering to *every* pair of neighboring segments. In practice, we work with oversegmented videos, where most segments are parts of the same rigid objects. In that case, most of the information given by the algorithm will be neither meaningful nor useful. We still need some way of distinguishing random output from useful output.

Heuristics

We have started this section whit the hypotheses that we have a perfect segmentation and a perfect optical flow. Then we have stated that in that case the criterion gives a correct ordering between neighboring regions. Of course, the two starting hypothesis are impossible to fulfill in practice. Thus, we ask ourselves how robust is the method to incorrect segmentations and incorrect optical flows.

Before having access to the results of extensive experimentation, we take note of the following tweaks that help to make the computation more robust:

Erosion. The algorithm requires a model for the movement of each region. Since we have a dense optical flow, the simplest model is that each pixel follows its own optical flow. However, dense optical flows tend to be imprecise around occlusion boundaries, which is precisely where we want it to be precise. To improve the optical flow around occlusion boundaries, we do the following. For each region of the segmentation, we take the pixels which are far from the boundaries of that region (i.e., we compute an erosion of that region). The optical flow on these inner pixels is used to build an affine model, which is later extended to the whole region. This produces a model for each region which is not affected by the errors of the dense flow around occlusion boundaries.

Weighting by divergence. We have seen that, around occlusion boundaries, the divergence of the optical flow takes values very far from zero. If the movement of the objects is rigid and perpendicular to the line of sight, the divergence of the optical flow vanishes on the interior of objects. The integral of the divergence on the non-vanishing part along an occlusion boundary is proportional to the relative motion of both regions (regardless of the smoothing of the dense flow). Thus, it makes sense to compute that integral to reinforce the pixel count of the algorithm.

Thresholds. In order to reduce cluttering of the output (i.e., many non-meaningful results), we can apply several post-processing criteria. We have already discussed weighting the result of each boundary by the absolute value of the divergence along that boundary. Many other weightings seem natural at this point: the gradient of the original image, the difference of motion models for the regions on each side of the boundary, the lenght of the boundary, the straightness of the boundary (because spurious boundaries are more likely to be ragged), the area of the smallest region of that boundary (because flow models for small regions are more likely to be noisy), etc.

Experiments

We display the results of our analysis for three sample videos on Figures 5.20, 5.21, 5.22 and 5.23. In each figure we show, from left to right: 1,2) Two consecutive video frames of the sequence. 3) A segmentation of the first frame with arrows indicating the movement of each segment. 4) The computed orientation

of each boundary. 5) The divergence of the optical flow. The orientations of the boundaries are visualized as follows: the light side of the boundary corresponds to the object which is above, the dark side to the object which is below. The divergence of the flow is displayed in order to realize that the interesting occlusion activity happens at places where $|\mathrm{div}(F)|$ is high.

All these results are produced using hand-tuned segmentations The results for general automatic segmentations are more difficult to visualize due to cluttering and to repeated edges.

At this point, the main problem of the automatic results is *clutter*, due to the use of too fine segmentations. If we fine-tune by hand the parameters of the segmentations to avoid over-segmentation, we can suppress most of the clutter, with some effort. The weighting by the divergences explained above can then be used to weight the importance of each boundary, in order to sort them by their meaningfulness. A simple automatic criterion for assessing the meaningfulness of each occlusion boundary is provided by the test explained below on Section 8.2.

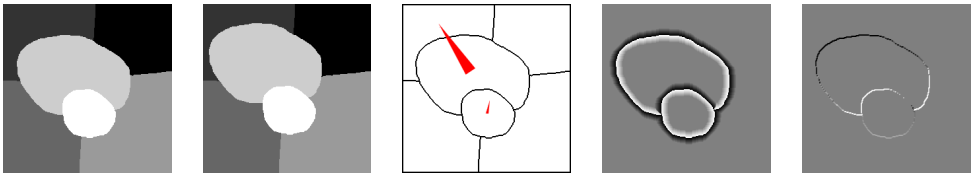


Figure 5.20: Analysis of a synthetic video. The computed interpretation is correct.

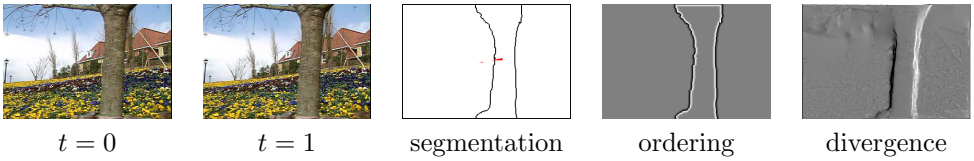


Figure 5.21: Analysis of a real video. The tree is correctly interpreted to be nearer to the camera.

Figure 5.22: Analysis of a real video. The person is correctly interpreted to be in front of the wall.

Figure 5.23: Analysis of the same video as in Figure 5.22, with a finer segmentation. Notice that the relative ordering between the paper and the wall is not meaningful.

Part II

A 3D Edge Detector

6 Historical context of edge detectors

The third part of this thesis is devoted to the description of a new edge detector for 3D images. The goal of this chapter is to motivate the context for the proposed edge detector. On Section 6 we briefly explain the methods and the applications of 2D edge detectors. On Section 6 we state the general statistical principle upon which our edge detection is built. On Section 6 we explain the methods and the applications of 3D edge detectors, highlighting the difficulties that are specific to 3D.

6.1 History of edge detection in 2D

Edge detection is the task of finding the boundaries between the objects that appear in a digital image. Segmentation is a different, but closely related problem, which consists in finding the objects themselves. Both problems have different constraints and applications. As each segmentation gives rise automatically to edges, but not the other way round, edge detection is a strictly more general problem (see, e.g., Figure 6.1). From a mathematical standpoint, edge detection finds the discontinuities of a function and segmentation finds a partition of the domain. Edge detection, being of a lower-level nature than segmentation, is aimed at picking structures all over the image and usually needs no initialization. To put both problems face to face, we can say that edge detection is an extraction of features appearing in the image, while segmentation provides a global interpretation of it.

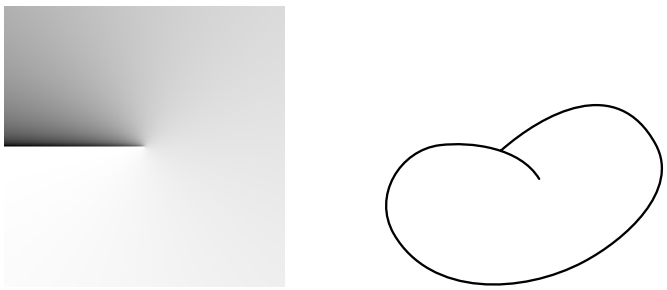


Figure 6.1: Edges are not necessarily boundaries of segmentations. Left: a crack-tip. Right: image of a bean.

People have been using 2D edge detection for years (see [Jul59]) for many tasks. For example, to obtain a visually appealing "primal sketch" [MH80, HWL83] of a picture, to reduce the amount of information present in an image, to get a manageable list of "features" to perform registration [AF87, HLF+97, Bro92] of two images, or shape matching [MAK96, LMMM03]; and finally as a first step towards the segmentation of the image into regions. For an account of its applications to computer vision see [FP03].

Let us briefly review the main approaches to edge detection. A gray level image can be realistically modeled as a real-valued function $u(\mathbf{x})$ where $\mathbf{x}$ represents an arbitrary point of a rectangle Ω in $\mathbb{R}^N$ ($N = 2$ for usual pictures, 3 for medical images and movies for example) and $u(\mathbf{x})$ denotes the gray level at $\mathbf{x}$. In this continuous setting, when an image $u : \Omega \to \mathbb{R}$ is a smooth function, edges are usually defined in terms of a differential operator. Most, if not all, of them are based on one of the following three:

The *norm of the gradient* $|Du| = \sqrt{u_x^2 + u_y^2}$ produces an image which is interpreted as a measure of the "edgeness" at each point of the image domain. Many detectors (e.g., Sobel [Dav75], Prewitt [Pre70], Roberts [Rob65], Kirsch [Kir71] and the morphological gradient [BL79]) can be interpreted as numerical schemes to approximate this norm. The main advantage of these operators is that they are fast and easy to compute. Their main limitations are that their output is difficult to use and blurry edges are not well localized.

The *Laplacian*. According to Marr-Hildreth [MH80], edges can be defined as zero-crossings of the Laplacian $\Delta u = u_{xx} + u_{yy}$. This method has the advantage of being able to directly produce curves which are well-localized, but it may lead to false detections (e.g., at almost flat zones where noise dominates).

Canny's operator $D^2u(Du, Du) = u_x^2 u_{xx} + 2u_x u_y u_{xy} + u_y^2 u_{yy}$ is the second derivative of u along its gradient lines. Its zero-crossings are called Haralick's edges [Har84], and they are better localized than the zero-crossings of the Laplacian [HS85].

Since Canny's operator is the second derivative of u along its gradient lines, it vanishes where the first derivative of u is maximal in the direction of the gradient. Instead of computing second derivatives, these maxima can be found directly by looking at the values of the derivative at the neighboring pixels in the gradient direction and discarding those pixels that have higher contrasted neighbors. This process, called non-maximum suppression, forms the basis of an efficient implementation of Canny's filter. The explicit details of the method (see [Der87] and [Can86]) are somewhat intricate because the choice of previous filtering is critical to ensure the best localization. The result of this non-maximum suppression is then pruned using two threshold parameters in a process called *hysteresis*. Thus, Canny's edge detector, while being able to give very good results, uses three parameters which are usually set by hand, specifically for each image, by visual inspection: The first parameter is the width of the (necessary) initial linear filtering, whose optimal value depends on the overall amount of noise in the original image, and the other two parameters are thresholds, whose optimal value depends on the distribution of the contrast.

Edge detectors usually give as edges a set of pixels, and those have to be connected to produce a set of curves. Active contours or snakes were developed to obtain a boundary segmenting a region of the image (or a set of regions) [KWT88, CKS97, MSV95, CCA92, KKO+96]. They are interfaces (curves in 2D images, surfaces in 3D images) that evolve to minimize an energy functional. The minimization is usually performed using gradient descent starting from a given initialization. The choice of a good initialization is thus critical

because the energy functional may have several local minima. Other particular approaches, based on segmentations, include finding a partition of the image domain that globally minimizes an energy, as in *graph cuts* [BK03]; or finding *watersheds* (i.e., connected components of lower level sets) of the gradient norm [VS91, SSV$^+$97].

6.2 Edge detection in 3D

While traditional edge detection was initially introduced for 2D images, most of the techniques can, at least theoretically, be extended to 3D images. However in three-dimensional images, the boundaries between objects are not curves in the plane but surfaces in space. In that setting, the summarizing property of edges is even more important because these images can not be easily visualized as a whole (whithout resorting to specialized rendering techniques). On the contrary, a set of surfaces in space is easy to visualize, specially if the user can rotate interactively the whole image domain. Even when the surfaces are nested it is useful, because the surfaces can be endowed with transparency. Thus, edge detectors are an invaluable tool in 3D visualization, for they provide an efficient way to glance through the content of whole images. Aside from visualization, 3D edges are used also for other tasks, e.g. registration [HLO99] or landmarking [PSO$^+$01].

Edge detection is often more appealing in 3D than in 2D, because the occlusion phenomenon does not occur in 3D images, and occlusions are one of the two things that make edge detection hard (the other one being textures).

6.3 Edge detection by Helmholtz principe

Desolneux, Moisan and Morel proposed in [DMM01] a new method for edge detection (named DMM, from now on), based on a general theory (see [Low85, GH91, DMM04]) aimed at giving sensible values to perceptual thresholds. It is possible to apply that theory directly to set the hysteresis thresholds for Canny's filter, but DMM is more elaborate in that it finds a separate threshold for each edge, according to its size and its contrast. The main steps of this method are the following:

(i) The family of level lines of the image and the distribution of the modulus of the gradient are computed and stored.

(ii) Then, all arcs of level lines are subsequently tested, one by one, to verify that they are well contrasted. The arcs that pass the statistical test are the output of the algorithm (they are named *meaningful* edges).

This algorithm contains no tunable parameters because the minimum contrast required for a given curve to be meaningful is determined automatically by the statistics of the image contrast. With respect to Canny's detector, it has the advantage of not producing an unstructured set of edge points, but a set of continuous planar curves on the image domain (e.g., with sub-pixel precision).

Let us mention that, in order to refine the computation of the boundary of an object, a meaningful level line may be used as an initialization for a classical active contour model [DMM03]. It is worth noting that [DMM01] introduced two variants of this algorithm, testing either whole level curves or arcs of them, and the respective outputs were named *meaningful boundaries* and *meaningful edges*. Our method is based on the second, more general, variant.

7 Digression on triangulated surfaces

The second part of this thesis proposes a method for 3D edge detection whose output is a finite set of triangulated surfaces in the image domain. On this chapter we explain some general properties and algorithms for triangulated surfaces. The content of this chapter belongs to a field of study which is sometimes disparaged by the image processing community as a part of computer graphics, and by the computer graphics community as a part of image processing. Nevertheless, we find its many details worthy of interest. On Section 7.1 we explain how to obtain level surfaces of a given 3D image with sub-voxelic precision. On Section 7.2 we explain how to compute lenghts of curves defined on surfaces. On Section 7.3 we explain a method to obtain segmentations of functions defined on surfaces.

7.1 Consistent Marching Cubes

The Marching Cubes algorithm takes a 3D grid of gray values and a threshold, and produces a triangulated surface separating the grid points above and below that threshold. The original incarnation of this algorithm had a topological inconsistency problem because the produced triangulations were not always boundaries of polyhedra. This problem was solved on subsequent versions of the algorithm, by carefully modifying the look-up table. Thus, a modern implementation of Marching Cubes will always produce closed surface triangulations.

However, there remains a different topological problem: for many 3D images, the triangulations produced by Marching Cubes at different thresholds may intersect, specially when those levels are close. This phenomenon is specially prevalent at textured parts of the image where many different cases of the Marching Cubes look-up tables are triggered together. However, it can appear even for very regular images, as shown on Figure 7.1

The marching cubes algorithm is used to compute a triangular mesh approximation to the isosurface of a function sampled at points of a 3D orthogonal grid. The idea is to process independently each cubical cell formed by 8 contiguous samples. Linear interpolation along the edges of the grid is then used to compute the vertices of the isosurface aproximation.

The original marching cubes algorithm, as described initially in [LC87], has a well-known consistency problem that gives rise to holes in the generated triangulations (see [vGW94]). These holes can appear only inside cells that contain a singularity. While this hardly poses problem when the surface is used for visualization, it certainly calls for trouble when the surface is used as an object for further computation, as we do.

In the literature one can find several attempts to solve the topological inconsistencies, leading to two successful variants of the original algorithm. One possibility is to require that the polyhedron given by the algorithm has the same topology as the trilinear interpolant of the sample points. A complete case by case study of the singularities of a trilinear interpolant inside a cubical cell appears in [Nie03]. This method has the disadvantage that it gives triangulations

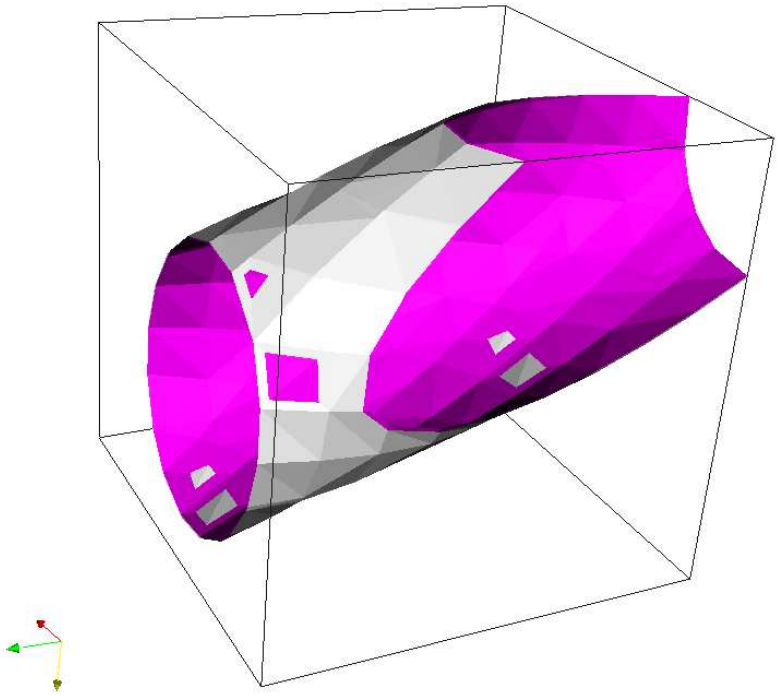


Figure 7.1: Two isosurfaces of the same 3D image at different thresholds, as computed by the VTK library. The two surfaces are drawn with different colors, to emphasize the places where they intersect. The image is obtained by sampling at the grid points the distance to straight line.

that are too complicated and require the introduction of new points outside the edges of the grid. Another, more pragmatic approach, takes the table of cases of the original algorithm and modifies it a bit, with some care, so that the resulting triangulations have no holes. This is the method used in the [va] library. This "combinatorial" approach does not resort to an interpolation of the function, so that it is not clear what it is computing. Again, this is not a problem at all when the purpose of the triangulation is to give nice visualizations, which it does, but we needed something consistent with our tree of shapes.

For that, we designed our own variant of the marching cubes algorithm, see figure 7.2 for its table of cases. Our requirement was that the surfaces should have the same topology as the borders of the shapes. This turns out to be an easy requirement to fulfill, and it gives automatically a consistent table of cases:

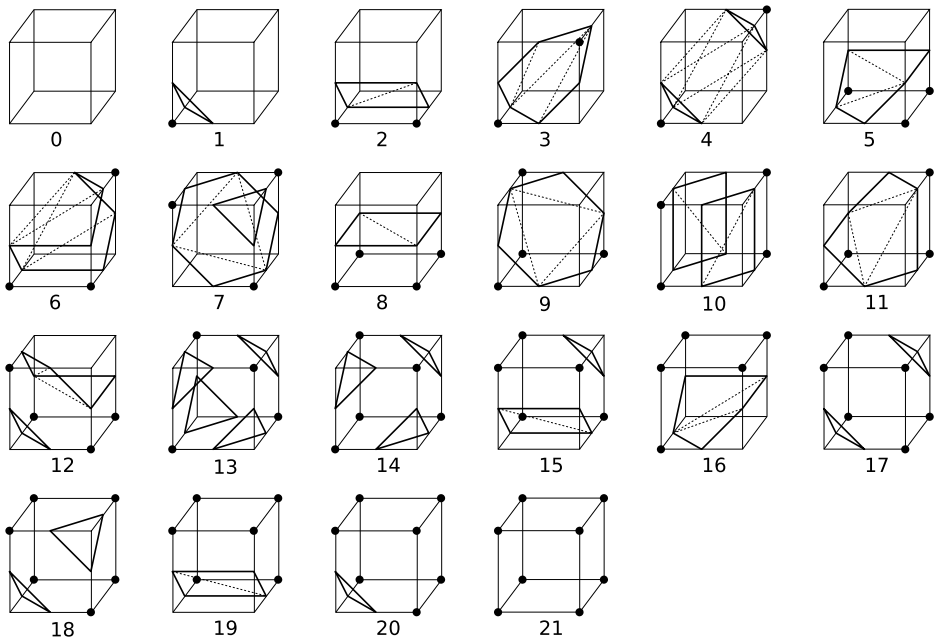


Figure 7.2: Lookup table for our version of the Marching Cubes algorithm. There are 256 cases altogether, which can be obtained as rotations of those shown here plus the mirror image of case 11.

we only have to require that all the vertices whose value is above the threshold (marked as black dots in the table) are not separated by the triangulation.

Let f be the distance function to some straight line in $\mathbb{R}^3$. The isosurfaces of f are cylinders. If we sample the function f at the points of a regular grid we obtain a three dimensional image of voxels. Now, using Marching Cubes we can try to extract the isosurfaces of that image. Popular programs such as `VTK` or `amira` give surfaces that are not convex at all, whereas as cylinders they should be convex; see figure 7.3. Even if the vertices of the triangulation lie nearly on the isosurface, the edges that connect those vertices are far from being correct, resulting in small concavities in some places. This nonconvexity is not an artifact due to the small radius of the cylinder: as we will see, it happens for any radius. It is the result of an indeterminacy in the Marching Cubes algorithm.

Where do the anomalies come from? Well, take a look at case number 8 of the Marching Cubes table in Figure 7.2. There are two possibilities of triangulation, shown in Figure 7.4. Likewise, most of the other cases also allow different possibilities. For example, there are 14 ways to triangulate the hexagonal border in case 9, corresponding to the 14 different triangulations of an hexagon 7.6.

Note that the choice of triangulation does not change the topology of the surface (when viewed as a simplicial cell complex). Classical Marching Cubes

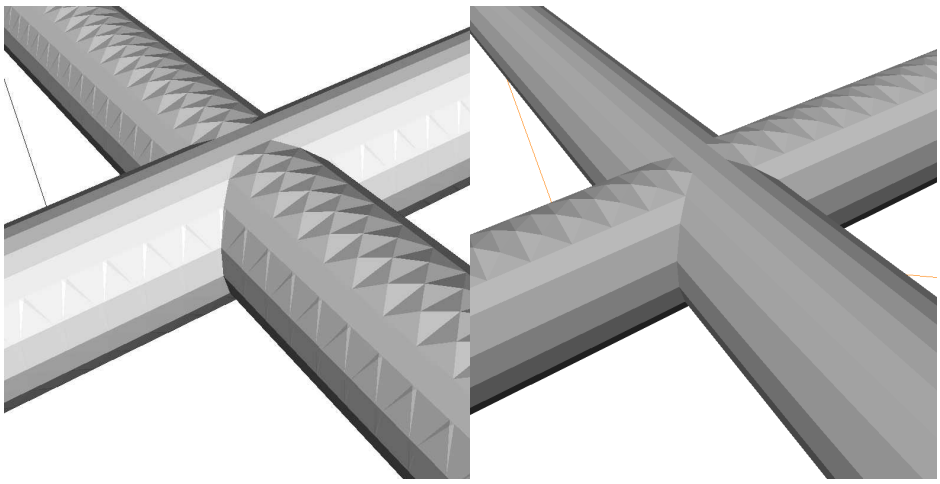


Figure 7.3: Some isosurfaces obtained by VTK (left) and amira (right).

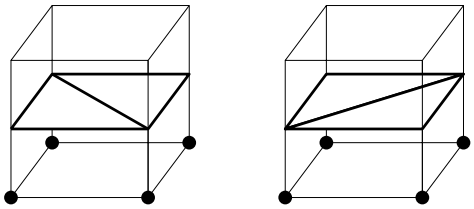


Figure 7.4: Two triangulations for case 8 of Marching Cubes.

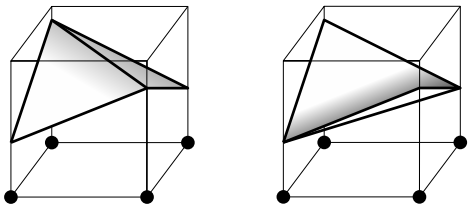


Figure 7.5: The two triangulations give different surfaces when the points on the edges are not coplanar.

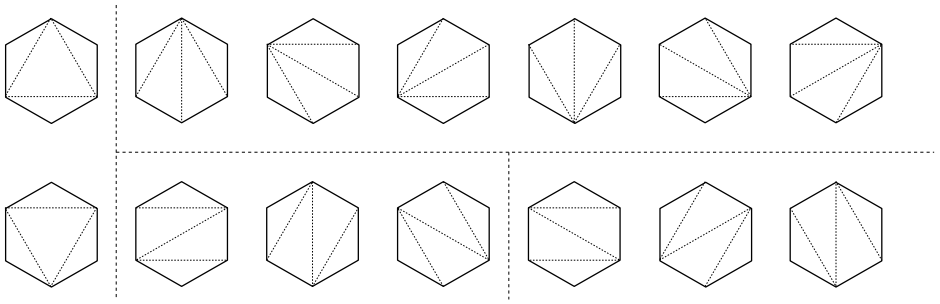


Figure 7.6: The fourteen ways to triangulate an hexagon.

choses once and for all a triangulation for each case and applies it every time this case appears. Another criterion, equally reasonable, would be to choose each time a triangulation at random. The purpose of our study is to show that there are better ways to choose triangulations. Two sensible criteria are proposed.

Before going on further, let us see an example where this choice makes a difference. Of course if in case 8 the four vertices are on the middle of the cell edges, then they lie on the same plane, and both triangulations give rise to the same surface. However, it is customary to locate the vertices e_i on the cell edges at a position corresponding to the linear interpolation of the function along that edge. In that case the two triangulations can give very different surfaces, see figure 7.5.

The example of the cylinders is simple enough to be able to know which is the correct solution. By visual inspection, it is apparent that some edge flips would render the surface convex. The good choice of the edges is the one that best approximates the isosurface. Another way of looking at it, is that the good edges are those most well aligned with the generatrix of the cylinder. This coincides with the direction of minimal curvature.

7.2 Graph cuts on surfaces

Let us address the following problem: given a triangulated surface and a patch of it, compute the length of the boundary of that patch. The patch of the surface is given by a set of vertices of the triangulation. There are two naive, immediate approaches to this problem.

The first approach is based in considering the triangulation to be a polyhedral surface. In that case, we can use the given set of vertices to define a continuous surface patch, by taking the union of the Voronoi cells of the surface at those vertices. Then we compute the length of the boundary of that region, which is a polygonal line. This computation gives an estimation of the boundary length which is consistenly larger than the desired length. This can be seen, for example, when using a planar triangulation of equilateral triangles of side l. Then, we are approximating the desired boundary by hexagons of side $\frac{\sqrt{3}}{2}l = (0.866\ldots)l$.

Notice that if we estimate lengths in that way, the precision can never be increased by refining the triangulation because we are always approximating the boundary by hexagons.

The second approach is based in considering the triangulation to be a (combinatorial) graph. In that case, we simply count the number of edges of the graph such that exactly one of its endpoints belongs to the given set of vertices. The problem with this approach is that it gives always a natural number as a measure of length, so that it is necessarily imprecise. Moreover, this method does not take into account the shapes of the triangles, so that very small triangles are given the same weight as larger ones. This problem is mitigated when using triangulations given by Marching Cubes, since their edges are never larger than $\sqrt{3}$, however, this is still a limitation.

Besides its obvious shortcomings, the second approach is more interesting for three reasons: it is very easy to implement, it can be made more precise by assigning *weights* to the edges, and its precision can be arbitrarily improved by adding more edges to the graph. Let us explain this. First, notice that the same computation of the first approach can be exactly simulated by assigning suitable weights to the edges of the graph, in the following way: the weight of each edge must be the length of the boundary of the Voronoi cell that crosses it (if any). Then, it is natural to ask if these weights provide the best possible results. The answer turns out to be negative. For instance, in the example given above of the regular tesselation of the plane into triangles of side l, the optimal weights are closer to $\frac{1}{2}l$. For example, these weights give asymptotically exact results for the length of straight segments whose slope is a multiple of 60 degrees. For other slopes, the depth estimation given by the cut is slightly larger than the real length.

Now, what is the best way to set the edge weights so that the graph cuts give an optimal approximation of the metric? While the optimal weights for a regular tesselation can be computed manually, there is a general technique for assigning such weights based on Crofton's formula. See [San53], Chapter 3 for the theory in a continuous setting, and [BK03] for a discrete example. In the case of arbitrary triangulations, the optimal weights are given by the following formula

$$\omega_e = \frac{1}{2} l_e \theta_e \tag{7.1}$$

where l_e is the length of the edge and θ_e is the angle "spanned" by this edge, for example the average of the angles of the two neighboring triangles. This formula has an intuitive interpretation in terms of the triangle fans around each vertex: it corresponds to estimating the length of any segment that crosses the triangle as the average length of all possible segments crossing that triangle (at least in the case of small angles θ_e, such that θ_e is a good approximation of $\sin \theta_e$). In the particular case of the regular tesselation commented above, this formula gives all weights equal to $\frac{\pi}{4\sqrt{3}} l = (0.453\ldots)l$. These weights are better than $\frac{1}{2}l$, because they give lenght estimations of straight lines that are slightly above or sligthly below than the exact value, depending on the slope of the line.

Let us recall the result of all this development: we have a method for computing the lenght of boundaries on triangulated surfaces, based on weighted cuts of the graph associated to the triangulation. This method can be made as precise as needed, by refining the triangulation or by adding new edges to the graph and updating their weights according to formula 7.1. We use this method later, in order to compute the lenghts of the boundaries given by Mumford-Shah segmentation of surfaces.

7.3 Mumford-Shah segmentation of surfaces

The Mumford-Shah functional [MS88] is a starting point of various methods for image segmentation ([KLM94], [CV01]). The definition of the functional can be extended immediately to arbitrary manifolds endowed with arbitrary metrics. Since the functional measures the boundaries of a partition of the space, it leads naturally to a discretized algorithm where these boundaries are measured using cut metrics [BK03], possibly weighted locally using the image contents ([KWT88], [CKS97]). For our purposes, we only need this algorithm to segment functions defined over 2D surfaces embedded in 3D space. Nevertheless, on this section we explain the algorithm in the general setting of arbitrary manifolds.

7.3.1 The simplified Mumford-Shah functional on the plane

Let $\Omega \subseteq \mathbb{R}^2$ be a rectangle and let $I : \Omega \to \mathbb{R}$ be a fixed function, called the "image". The Mumford-Shah functional [MS88] assigns a number to each function $u : \Omega \to \mathbb{R}$ which is smooth outside of a set $K \subseteq \Omega$. It has the following form

$$E(u, K) = \alpha \int_\Omega (u - I)^2 + \beta \int_{\Omega \setminus K} \|\nabla u\| + \gamma \mathcal{H}^1(K)$$

where α, β, γ are positive parameters. The first two integrals are taken with respect to the Lebesgue measure on the plane, and $\mathcal{H}^1(K)$ is the one-dimensional Hausdorff measure, that coincides with the length for rectifiable sets. A pair (u, K) that minimizes $E(u, K)$ is called a Mumford-Shah approximation of I with parameters α, β and γ. The three parameters define the trade-off between the fidelity, the smoothness and the simplicity of the approximation. Notice that, since the functional is homogeneous on the three parameters, there are only two independent parameters.

A common simplification of the above functional is to restrict u to piecewise constant functions. Then it is easy to see that the Mumford-Shah approximation is determined by the partition $\{\Omega_i\}$ of Ω that minimizes the following functional:

$$E_{\lambda,I}(\{\Omega_i\}) = \sum_i \text{variance of } I \text{ on } \Omega_i + \lambda \sum_{i,j} \text{length of border between } \Omega_i \text{ and } \Omega_j.$$

We will generalize this second version of the Mumford-Shah functional for spaces Ω more general than a rectangle in the Euclidean plane. There are two

things to do, one for each term of the function: to define a volume measure on the manifold (that allows us to define variances of functions on open sets) and to define a lenght or area measure (that allows us to measure the boundaries of the regions).

7.3.2 Mumford-Shah on manifolds

We start by restricting a bit the space of functions that we consider, to ease the definition of the functional. This is more a stylistic device than a real restriction, because here we are not going to perform a mathematical analysis of the functional.

Definition 45 (space of approximations)**.** *Let Ω be an n-dimensional smooth manifold. The* space of approximations over Ω *is the set of functions $f : \Omega \to \mathbb{R}$ that are constant over finite partitions $\{\Omega_i\}$ of Ω such that for any i, j the set $\partial\Omega_i \cap \partial\Omega_j$ is a finite union of $(n-1)$-dimensional smooth submanifolds of Ω.*

Then we define the Mumford-Shah functional on the space of approximations.

Definition 46 (Mumford-Shah functional)**.** *Let $I : \Omega \to \mathbb{R}$ be a fixed function, and let $\omega \in \bigwedge^n(\Omega)$ and $\zeta \in \bigwedge^{n-1}(\Omega)$ be fixed nondegenerate differential forms of degrees n and $n-1$ respectively. The* Mumford-Shah functional of parameter $\lambda \geq 0$ *assigns the following energy to a partition $\{\Omega_i\}$:*

$$E_{\lambda,I,\omega,\zeta}(\{\Omega_1, \ldots, \Omega_n\}) = \sum_i \int_{\Omega_i} \left| I - \frac{\int_{\Omega_i} I\omega}{\int_{\Omega_i} \omega} \right|^2 \omega + \sum_{i,j} \int_{\partial\Omega_i \cap \partial\Omega_j} \zeta.$$

The first observation is that the choice of the volume form ω allows to produce approximation functions out of partitions of Ω: simply assign the constant equal to the mean value $\frac{\int_{\Omega_i} I\omega}{\int_{\Omega_i} \omega}$ of I to each region Ω_i. Likewise, the volume form allows us to compute the variances. This means that the planar Mumford-Shah functional is a particular case of this (at least over our space of approximations), when ω and ζ are respectively the Euclidean area and length forms.

There are many meaningful ways to choose the forms ω and ζ. The simplest possiblity is when Ω is equiped with a Riemannian metric g. Then there is a natural volume form

$$\omega = \sqrt{|\det g|} dx^1 \wedge \cdots \wedge dx^n = *(1)$$

and a natural $(n-1)$-dimensional "area" form

$$\zeta = \text{natural area form} = *(\text{dl}).$$

An interesting case of the above is when Ω is a surface of $\mathbb{R}^3$ with the metric given by the embedding. Then, this functional can be used over functions defined on that surface.

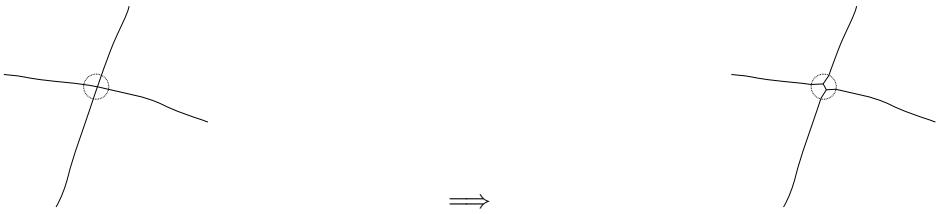


Figure 7.7: Changing the boundaries inside a small disk to obtain only equal-angled triple junctions.

Figure 7.8: Changing the boundaries inside a small disk to obtain only equal-angled triple junctions.

More generally, there is no need that ω and ζ come from one and the same Riemannian metric. For instance, to segment a regular two-dimensional image it makes sense to set ω as the Euclidean area and ζ as an active-contour metric [CKS97] that depends on I such as

$$\zeta = v(|\nabla I|)\, dl \tag{7.2}$$

where dl is the ordinary Euclidean length and v is some decreasing function such as $v(x) = \frac{1}{1+x}$. The net effect of this is that, while in the classical Mumford-Shah approximation all the boundaries are penalized in the same way, in the modified version it is more expensive to cut in the middle regions of low gradient than regions of high gradient.

Notice that the use of non-Euclidean metrics on the functional may solve (or at least mitigate) the well-known "120 degree problem". Minimizers of the classical Mumford-Shah functional only allow for triple junctions, and those in turn can only have three $\frac{2}{3}\pi$ angles. This fact is easy to prove: if we have a non-triple junction or one without $\frac{2}{3}\pi$ angles then we can change the approximation inside a small disc of radius ϵ around the junction, to have only triple junctions of equal angles. This will certainly make the boundaries shorter, and thus the length term will be smaller. If ϵ is small enough, the variance term on the functional can only change proportionally to ϵ^2 and the length term changes proportionally to ϵ. This means that the whole functional can always be improved by transformations as those depicted in figures 7.7 and 7.8. Thus, at the optimum all triple junctions must have equal angles.

This "120 degrees" phenomenon is not usually a problem in practical implementations, because the scale at which those angles occur tends to be smaller than the image discretization. However, it is unsuitable as a theoretical model, for it does not allow T-junctions, which are arguably essential features of images. Our minor generalization of the functional (e.g., when the Euclidean length is replaced by a snakes-like metric) allows any angles on junctions. To see that this is true, simply consider how does an affine transformation of the Euclidean plane affect the functional: optimal partitions transform to optimal partitions, while the 120 angles may transform to any angle. In the more general case, the metric may nearly vanish along some curves which meet at arbitrary angles, and the optima of the functional will strongly prefer those curves.

7.3.3 Discretization and algorithm

Let us describe how to implement the ideas above as a computer program. We propose a discrete setting, where each point of Ω is the vertex of a graph $\mathcal{G}$, whose edges connect some neighboring points. In Section 7.2 we explained how to assign weights to those edges in such a way that cuts of the graph approximate an arbitrary metric. Now, let us explain how to find good enough local minima of the functional by merging regions of this graph, using the method introduced on [KLM94]

The parameter λ in the Mumford-Shah functional controls the trade-off between the fidelity of the approximation to the original image and the simplicity of the approximation. When $\lambda \to \infty$ the only term that counts is the sum of the lengths of the borders and the best approximation is that of the whole image by its mean value. When $\lambda \to 0$ the only term that counts is the variance (or error) and the best approximation is an exact copy of the image (with small enough pieces).

In the discrete setting these "small enough" pieces are precisely the vertices of the graph $\mathcal{G}$, and this is a starting point for our merging algorithm, because it is a global minimizer of the functional. The problem of finding global minima for higher values of λ looks too hard and we will not try to solve it. Instead, we will conform on finding approximations of this minimizer that give reasonable-looking approximationss. Namely, we will look for segmentations with the property that their energy can not be minimized by merging two regions of the partition:

$$\forall k, l \quad E(\{\Omega_1, \ldots, \Omega_n\}) < E(\{\Omega_1, \ldots, \hat{\Omega_k}, \ldots, \hat{\Omega_l}, \ldots, \Omega_n, \Omega_k \cup \Omega_l\}).$$

Partitions $\{\Omega_i\}$ with the property above are called 2-normal partitions. Instead of looking for partitions which are global or local minima of E we will aim only for 2-normal partitions.

Because there is no clear way to choose a single value of λ that gives reasonable segmentations, it is customary to compute at once those approximations for *all values* of λ. In general, the optimal partitions determined by different values of λ need not be related at all. But if we are happy with 2-normal partitions then there is a clever strategy to do it: start with the exact partition for $\lambda = 0$, and

look at the first pair of regions that would make fail the definition of 2-normality as we increase λ. Then merge this pair of regions and keep increasing λ until reaching the trivial partition $\{\Omega\}$.

The procedure that we have just described serves to build a binary tree of mergings that represents a hierarchy of partitions. Each node of the graph represents a region $\Omega_i \subseteq \Omega$. The leafs are the smallest possible pieces, and the root is the whole Ω. Once this tree is built it is easy to prune it to obtain 2-normal segmentations having a pre-defined number of regions. Let us write more formally this procedure:

Input:

- A graph (representing a discretization of the space Ω)
- Values on the vertices on the graph (representing the image I)
- Weights on the vertices of the graph (representing the volume form ω)
- Weights on the edges of the graph (representing the boundary form ζ)

Output:

- A binary tree of mergings (that can be pruned to obtain a collection of 2-normal segmentations for all possible values of λ).

Algorithm:

1. Build a region-adjacency graph out of the original connectivity graph
2. While the region-adjacency graph has more than one vertex:
   - Collapse the pair of adjacent nodes i, j that minimize the following number
$$\frac{|\Omega_i||\Omega_j|}{|\Omega_i| + |\Omega_j|} \frac{\|m_i - m_j\|^2}{l_{ij}}.$$

where $|\Omega_i| = \int_{\Omega_i} \omega$, $m_i = \frac{\int_{\Omega_i} I\omega}{\int_{\Omega_i} \omega}$ and $l_{ij} = \int_{\partial\Omega_i \cap \partial\Omega_j} \zeta$. Now it remains to explain how to compute discrete approximations to these integrals.

We have used the setting above to segment the contrast over each level surface of the image.

As far as we know, this is the first time that the Mumford-Shah functional is used to segment data defined on surfaces.

8 Complete description of the proposed edge detector

We will follow the main steps of the DMM algorithm in order to construct an edge detector for 3D images. Thus, the first step of our edge detector will be to compute the family of its level surfaces. For that, we use the Tree of Shapes described on Part I. Other closely related data structures are those developed by Cox-Karron-Ferdous [CKF03], by Pascucci-Cole-McLaughlin [PCM03], by Carr-Snoeyink-Axen [CSA03], and by Sarioz-Kong-Herman [SKH06].

The proposed edge detector consists of two steps: first we produce a finite family of candidate surfaces and then we select the most contrasted ones (if any) among this family. In Section 8.2 we give a general definition of well-contrasted subsets of an image according to an *a contrario* statistical test. In Section 8.3 we construct a family of surface patches to which we may apply the contrast test. The two sections together form the core of the method.

8.1 Hypotheses of the method

Our method relies on the following assumption: in an ideal case (e.g., a perfect acquisition method giving infinite resolution images without noise) the boundaries of the objects can be obtained by thresholding the image intensity. This assumption holds for a wide class of real world images, like many medical images, namely, X-ray computed tomographies and magnetic resonances, where the acquisition apparatus measures the density of a physical or chemical property of objects in space. In practice, however, *a single threshold does not suffice*, because there are artifacts due to the reconstruction of the image, and the limitations inherent to its finite representation. For example, thin vessels having the same width as a voxel appear in the images much darker than the interior of large vessels, even if the contrast agent concentration or the measured property is the same in both places. See Figure 8.1 for an illustration of this fact. Note that not all 3D images satisfy this assumption. For example, in ultra-sound images the objects are defined mostly by textures, and for these the proposed method will likely not give good results (nor the existing methods described above).

Let us summarize the main assumptions underlying the proposed method of edge detection:

(H1) The edges are formed by *large* pieces of level surfaces of the original gray scale image.

(H2) The edges have as high contrast as possible.

Let us give some arguments in favor of the soundness of these assumptions. First, observe that if an image has no constant regions, any voxel is on a level surface, and therefore *any* arbitrary surface can be trivially approximated as a union of pieces of level surface, indeed, voxels. Thus, the key of point of our first assumption lies on the word "large". Without this word, it is always true, but of no use at all. The assumption is then useful as a heuristic to show that the output of our algorithm will be a small set of large pieces of surface, and not a

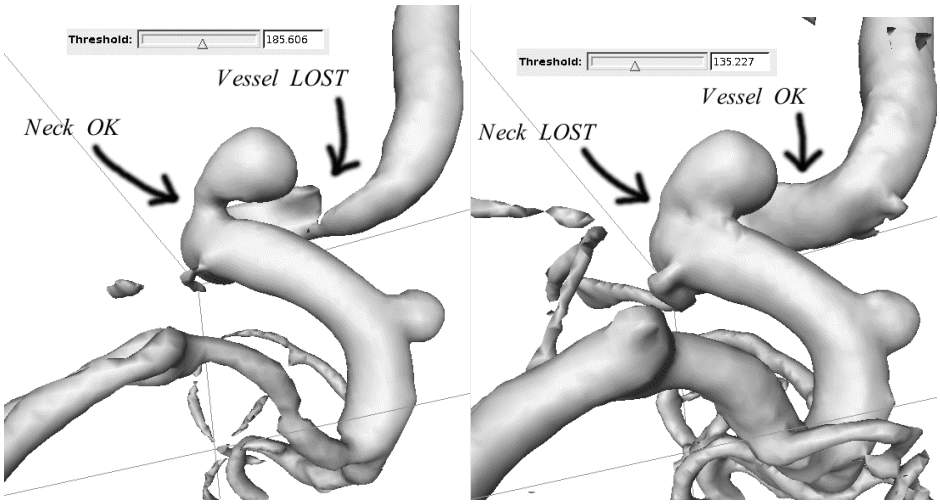


Figure 8.1: Two different isosurfaces of the same medical image. Note that each choice of threshold segments well some part of the image, but no threshold gives a globally correct segmentation.

large set of tiny pieces. Let us discuss to what kind of images this hypothesis applies.

8.2 Selection of meaningful patches from a given collection

Let us define the concept of well-contrasted subset of an image. The definition is a slight generalization of the one given in [DMM01, DMM04] based on an *a-contrario* model: Knowing the distribution function for the image contrast, we would sample the contrast values of the image at a randomly selected set of points, and we would look whether the sample had a distribution with exceptionally high contrast. In that case, this set of samples would be accepted as a well-contrasted set. Numerically, this reduces to selecting the sets which are large and whose minimum contrast is high. For a thorough discussion of the statistical foundation of this method, see [GH91], where it was described using the vivid name "conspiracy of random". For the intuitive idea in our case, see Figure 8.2. Notice that this *a-contrario* model is not based on an image of noise, but on noisy curves over the original image.

The norm of the gradient defines a contrast for every point on the image domain. We regard the values of the contrast at each voxel as *independent and identically distributed random variables, X_i, whose distribution is given by the histogram of the contrast.* This notation will be used throughout this section. This is a good model when a few voxels are chosen randomly over the image domain, but it fails when the voxels are not chosen independently (for instance,

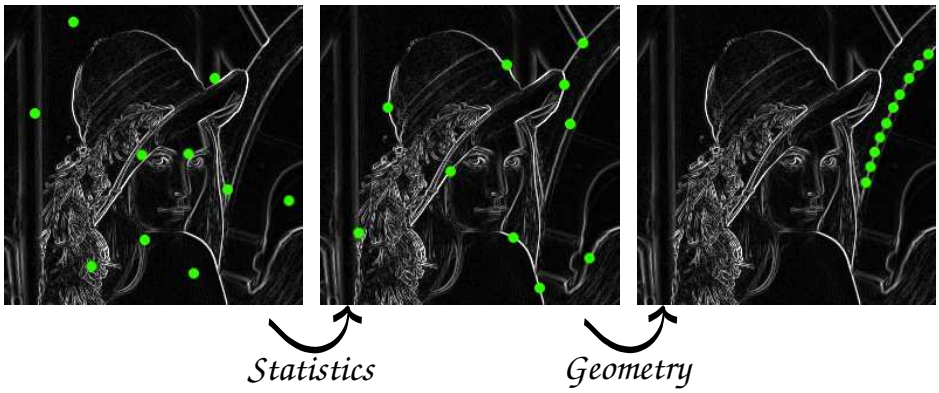


Figure 8.2: *Well-contrasted level curves of an image.* These figures display three different sets of 10 points thrown in the domain of the lena image.
Left: points thrown randomly. *Middle*: points thrown randomly in places where the gradient is high. *Right*: points on a well-contrasted level curve.
The first subset will fail the test described on Section 8.2, and the other two subsets will pass it. However, only the third subset will be presented to the test by the method described on Section 8.3.

if they are specially chosen along the boundary of an object). This failure is precisely what we look for, as the method can be regarded as an hypothesis testing of the independence assumption.

In the following paragraphs we describe a general setting to detect whether a sample from a distribution has abnormally large values. This device can be used to detect exceptionally well-contrasted subsets of an image. *The proposed edge detector is a particular case of this when the subsets are the level surfaces of the image (or their connected parts).*

To measure the contrast of sets of points we use an arbitrary statistic f, which for now is a parameter of the method:

$$\text{contrast}(\{X_i\}) := f(X_1, \ldots, X_n).$$

This statistic serves to summarize the whole contrast distribution of the set of points into a single real number. It may help to think that f is increasing in each of its components, but this is not logically needed for the following propositions to hold. Possible choices of f are the minimum $X_{(1)}$, the mean value $n^{-1}\sum X_i$, the median $X_{(n/2)}$, or some other quantile $X_{(\frac{q}{100}n)}$. From the statistic f we need its distribution functions F_n:

$$F_n(\mu) := \mathbb{P}(f(X_1, \ldots, X_n) \geq \mu)$$

which are decreasing functions of a real variable with values in the $[0, 1]$ interval. Notice that these are the complement of the usual cumulative distribution

function of $f(X_i)$. We also define for every positive integer n a meaningfulness function Sig_n as

$$\mathrm{Sig}_n(x_1, \ldots, x_n; \epsilon, N) := \log \epsilon - \log N - \log F_n(f(x_1, \ldots, x_n))$$

which is a real-valued function of n real variables (and two real parameters ϵ, N). In the following, when the subindex n of both F and Sig can be deduced from context, it will be omitted.

Now we define our statistical test. Suppose that we are going to deal with N sets of samples of the contrast:

$$S_i = \{X_1^i, \ldots, X_{n_i}^i\} \qquad i = 1, \ldots, N$$

Definition 47. *We define the* meaningfulness *of the set S_i as* $\mathrm{Sig}(X_1^i, \ldots, X_{n_i}^i; \epsilon, N)$. *We say that the set S_i is* meaningful *when its meaningfulness is positive. If we want to emphasize the parameters ϵ and f we will talk about ϵ-meaningfulness of S_i in the f-sense, or of whether the set S_i is ϵ-meaningful in the f-sense.*

Notice that the set S_i is meaningful when

$$N \cdot F_n(f(X_1^i, \ldots, X_{n_i}^i)) < \epsilon$$

and this is the usual definition of "meaningfulness" given in [DMM01, DMM04]. The quantity on the left hand side of the previous inequality is then called the Number of False Alarms of the set S_i.

Definition 47 is justified by the following proposition.

Proposition 48. *Under the same statistical model as definition 47, the expectation of the number of ϵ-meaningful sets is smaller than ϵ.*

The proof of the proposition is an easy consequence of this elementary result.

Lemma 49. *Let Y be a random variable and let G be its distribution function $G(y) = \mathbb{P}(Y \geq y)$. Then, for $t \in [0, 1]$*

$$\mathbb{P}(G(Y) < t) \leq t.$$

Proof. Let V be the random variable that counts the number of ϵ-meaningful sets. Notice that V is a function of the variables X_j^i. We want to prove that $\mathbb{E}(V) \leq \epsilon$. Let V_i be the random variable that equals 1 if the set S_i is ϵ-meaningful and 0 otherwise, thus

$$\mathbb{E}(V) = \mathbb{E}(V_1) + \cdots + \mathbb{E}(V_N).$$

Now we have

$$\begin{aligned} \mathbb{E}(V_i) = \mathbb{P}(V_i = 1) &= \mathbb{P}\left(N \cdot F(f(X_1^i, \ldots, X_{n_i}^i)) < \epsilon\right) \\ &= \mathbb{P}\left(F(f(X_1^i, \ldots, X_{n_i}^i)) < \frac{\epsilon}{N}\right) \leq \frac{\epsilon}{N}, \end{aligned}$$

where the last step is the application of the lemma to the random variable $Y = f(X_1^i, \dots, X_{n_i}^i)$, whose distribution function if $G(y) = F(y)$. Substituting this result in the previous formula we get

$$\mathbb{E}(V) = \frac{\epsilon}{N} + \cdots + \frac{\epsilon}{N} = \epsilon$$

□

The above proof is adapted from the proof given in [CMS05].

The original definition of meaningfulness given in [DMM01] used the statistic $f = \min$. There is a reason to allow for different choices of f, that can give more robust detectors; see for example Section 9.2 where it is used with a statistic other than $f = \min$. In the case of the minimum, the function F can be obtained directly from the distribution of the contrast, which is approximated using its histogram:

$$H(\mu) = \mathbb{P}(X \geq \mu) := \frac{\text{number of voxels with } |Du| \geq \mu}{\text{total number of voxels}}.$$

Then we have

$$F_n(\mu) = H(\mu)^n$$

where n is the number of points in the subset (the number of arguments of the function f). This minimum is a special case of the quantiles:

Proposition 50 (Distribution of quantiles)**.** *Let $X_1, \dots, X_n$ be independent and identically distributed random variables with distribution function $H(\mu) = P(X_1 \geq \mu)$ and let $X_{(1)}, \dots, X_{(n)}$ be the outcomes of these variables ordered increasingly. Then $X_{(k)}$ is a random variable whose distribution function is given by a binomial tail of parameter $H(\mu)$:*

$$P(X_{(k)} \geq \mu) = \sum_{i=0}^{k-1} \binom{n}{i} (1 - H(\mu))^i H(\mu)^{n-i}. \tag{8.1}$$

in particular, we have the distribution of the minimum computed before: $P(X_{(1)} \geq \mu) = H(\mu)^n$.

The right hand side of equation (8.1) , denoted $B(n, k, H(\mu))$, is the incomplete beta function (written as $B(n, k, p) = I_p(k, n-k+1)$ in the common notation of [AS65]). This is a well-known special function which can be efficiently computed just as easily as sin or exp. We used the GSL library [G+02] which provides a function call for it.

Remark 1. *In allowing for a choice of statistic f we are motivated by the fact that the minimum contrast is not a robust descriptor: if one single point of the set has very low contrast, the whole set is discarded regardless of the contrast of all other points. We refer to Figure 8.3 for a synthetic image illustrating this phenomenon. In that figure we have a well-contrasted object, surrounded by level*

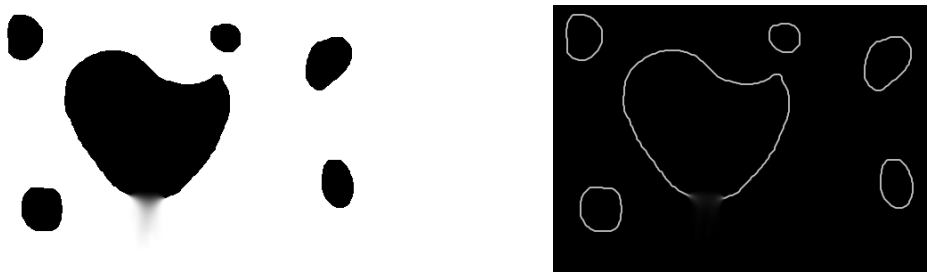


Figure 8.3: An image where level lines selection using $f = \min$ fails to detect a boundary which is visually obvious. Left: the image. Right: its contrast.

curves whose gradient is maximum at almost the totality of their points. However, all the curves cross a blurred region of the image, where the gradient can be made to be arbitrarily low. Then, none of these curves will be detected as meaningful. There are two approaches to deal with this kind of problem: either we work with parts of level curves instead of whole level curves, or we choose a different statistic such $f = 10th$ quantile. The first approach is the one chosen here, but the use of a robust statistic f is much faster and, in some cases, gives similar results.

8.3 Production of candidate patches

This section treats the main difference between the 2D and 3D versions of the edge detector.

The objects of our study are parts of level surfaces. This is the place where our method differs from the 2D case: a connected subset of a surface can be much more complex than a connected subset of a curve (which is determined by its two endpoints). This means that we can not treat all the connected subsets of each level surface, as is done in 2D: the search space would be too large. The first aid in the reduction of this search space comes from the observation that we are not really interested in *all* the subsets of a surface that pass the ϵ-meaningfulness test, but only in those that are "maximal" in the following sense:

Definition 51. *([DMM01]) Let $\mathcal{S}$ be a level surface of the image. A connected subset $S \subseteq \mathcal{S}$ is* maximal meaningful *when it is meaningful and*

- *it does not contain a strictly more meaningful connected subset*
- *it is not contained in a more meaningful connected subset*

Proposition 52. *([DMM01]) Maximal meaningful subsets in the* min*-sense are disjoint inside its level surface.*

Definition 51 was given in [DMM01] when the set $\mathcal{S}$ is a level curve and S an edge curve, that is, a connected subset of S_i. In that case the NFA is given by

the function

$$F(\mu, l) = N \cdot H(\mu)^l \tag{8.2}$$

where N is the number of edge curves of the image. Since edge curves are connected subsets of level curves, N can be computed for a given image. Definition 51 is analogous to the one given in [DMM01]. For the time being, we assume that N is a constant that can be computed. Then, the proof of proposition 52 is the same as in [DMM01] and is based on the observation that, for a fixed value of μ, the function $F(\mu, l)$ is nondecreasing in l.

Proposition 53. *Let $\mathcal{S}$ be a level surface of the digital image u. Maximal meaningful subsets of $\mathcal{S}$ are connected components of upper level sets of the modulus of the gradient restricted to $\mathcal{S}$.*

Observe that there may not be any meaningful connected subset of $\mathcal{S}$, in which case the proposition is vacuously true. In case there is one, then the connected subset with the smallest NFA is maximal meaningful.

Proof. Let S be a maximal meaningful subset of $\mathcal{S}$, and let $\mu = \min_{x \in S} |\nabla u(x)|$. If y is a neighboring point of S with $|\nabla u(y)| \geq \mu$, we could add the point y to S without increasing the NFA, contradicting the fact that S is maximal meaningful. Thus, all neighboring points of S have a modulus of the gradient lower than μ, and the statement of the proposition holds. □

Proposition 53 suggests an strategy for computing the maximal meaningful subsets of $\mathcal{S}$. We can find one of them on the collection of upper level sets of $|\nabla u|$ restricted to $\mathcal{S}$, and then searching recursively into its complementary.

But it turns out that the maximal meaningful subset of a level surface tends to be topologically very complex, with many holes and a complicated boundary. This happens because there is no restriction on the form of a maximal meaningful subset, besides being connected. See Figure 8.4.

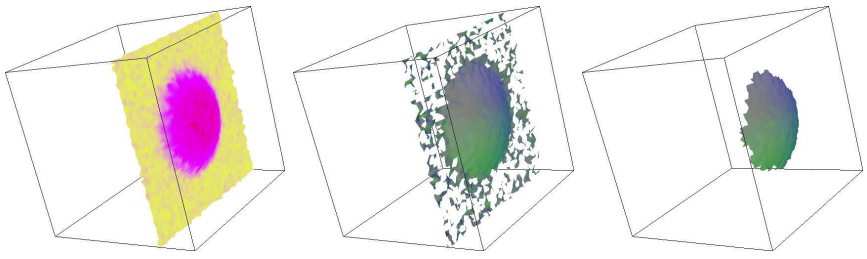


Figure 8.4: The maximal meaningful subsets of a level surface may not coorespond to physical features. Left: a level surface of an image colored by the contrast. Middle: the (connected) maximal meaningful subset of this surface. Right: the node of the Mumford-Shah hierarchy with maximal significativity.

The proposed approximation, suggested by the observation above, is to restrict ourselves to a reduced class of well-behaved subsets. A reasonable way to produce such a class of connected subsets of all sizes is a *hierarchy* of partitions (Figure 8.5).

Definition 54. *A* hierarchy of partitions *over a set M is a family H of subsets of M such that*

- $M \in H$
- *There is a subclass $L \subset H$, whose elements are disjoint and cover M. They are called the* leaves *of the hierarchy.*
- *Any element of H which is not a leaf can be represented as a disjoint union of leaves.*
- *Any pair of elements of H are either disjoint or nested.*

Restricting the family of connected subsets of level surfaces $\mathcal{S}$ to a hierarchy of partitions we define $F(\mu, l)$ as in 8.2, with N equal to the sum of all nodes of the hierarchies associated to all $\mathcal{S}$.

Once we have a hierarchy of partitions for a given level surface, it is easy to select the maximal meaningful objects of this partition in a greedy way (see Figure 8.6). We first compute the meaningfulness of each object, which can be done in linear time in the case $f = \min$. Then we pick the object which is most meaningful. This clears from the search all the descendants and ancestors of this object within the tree of subsets, because we want a set of disjoint patches. Then we pick the most meaningful object in the remaining part of the tree, and we keep doing that iteratively until no more patches can be picked.

8.4 3D Edge Detection Algorithm

On Section 8.2 we have explained how to detect when a set of points from a given family of sets is significantly well-contrasted. On Section 8.3 we have explained how to produce a large but manageable family of subsets to apply this test to. The sets of this family are patches of level surfaces, of all sizes, where the image contrast is as homogeneous as possible. Putting these two ingredients together we obtain the proposed edge detector:

Input:

- Original gray scale image, u
- Sensitivity parameter, $\epsilon > 0$, ($\epsilon = 1$ by default)

Output:

- A set of patches of surface, Γ

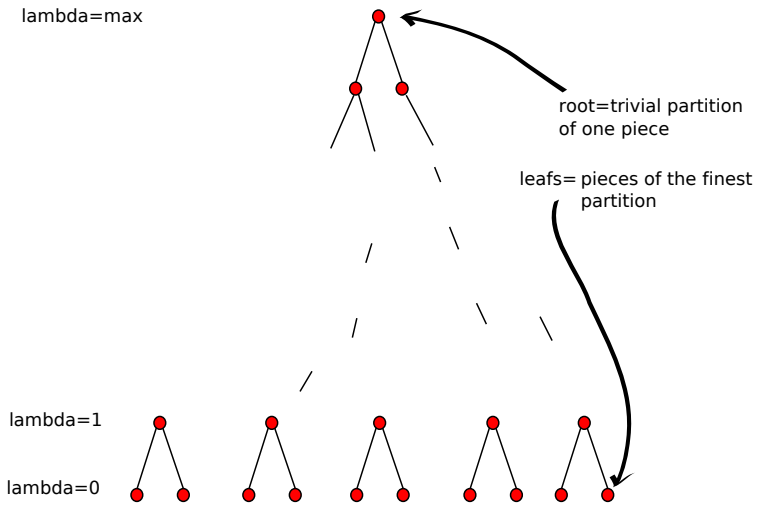


Figure 8.5: A hierarchy of partitions whose depth is indexed by a scale parameter λ. The leaves of the tree represent the points of the discrete surface, and the root of the tree represents the whole surface. Notice that the total number of nodes, being a binary tree, is proportional to the number of leaves (exactly the double minus one).

Algorithm:

1. Compute the image of contrast $g = |\nabla u|$
2. Let N be twice the sum of the surface areas of all the level surfaces. This will be the total number of tests to be done.
3. For each connected component S of each level surface of the gray scale image:
   a) Generate a mesh of triangles to represent S
   b) Interpolate the contrast at the vertices of the triangulated surface S
   c) Compute the Mumford-Shah tree, T, of the contrast function on S
   d) Perform the statistical test with $f = \min$ to all the nodes of T
   e) While there are still nodes in T:
      i. Pick the node q of T that passes the statistical test with highest score
      ii. Output the patch of surface corresponding to q
      iii. Remove from T the node q and all its ancestors and descendants

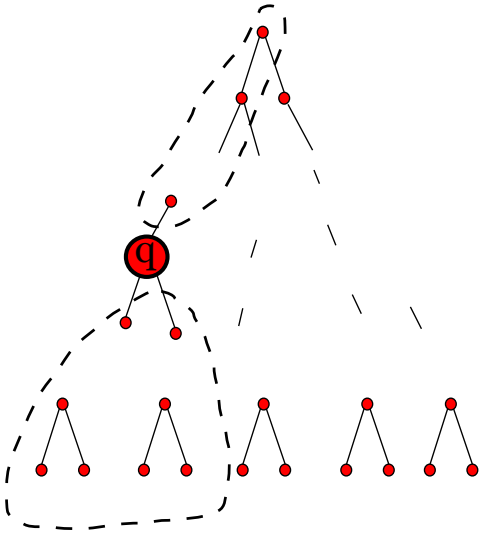
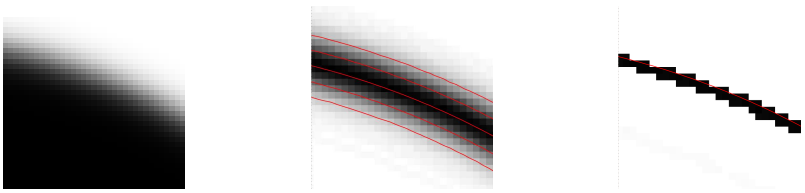


Figure 8.6: Each node of the tree in Figure 8.5 represents a connected patch of surface. Once we have selected the most meaningful node (in this figure, the enlarged one), we can remove all the nodes that are not disjoint with this one. They are all the ancestors and descendants, marked by the dotted line in this figure. Then we are left with the rest of the nodes in the tree.

Notice that this modular design allows us to try some variations of the algorithm. For instance, we can use a different statistical test or a different set of surface patches (for example, all the shapes of the tree). Another variation that gives specially good results consists in setting the contrast g equal to the output of Canny operator, instead of the norm of the gradient. See Figure 8.7 for an example where this makes a difference.



smooth edge gradient norm Canny

Figure 8.7: Combination of our method with Canny's. We can use the output of Canny detector as the contrast for our method, thus enhancing the localisation of the detected features. In this figure we show the detected curves of a synthetic 2D smooth edge in both cases.

9 Further notes about the proposed edge detector

9.1 Exclusion principle

The output of our edge detector (and that of the second version of DMM, based on portions of level lines) is usually highly redundant in the following sense: edges appear represented as bundles of surfaces (or curves). Here we introduce an exclusion principle to reduce the redundancy of the output, by picking the best representative of each bundle. It is based on a similar principle used in a segment detector [DMM00] to reduce output redundancy. As this method works exactly in the same way in 2D and 3D, we only describe here the 2D case and then we can easily support the explanation with figures. For the 3D case, it suffices to replace "curve" by "surface" and "square" by "cube".

The proposed "exclusion principle" works by dividing the image domain into small square regions (e.g. of pixel size, but not necessarily so), and imposing these two requirements on the final set of curves:

(i) Each square belongs to at most one curve

(ii) Each curve passes the statistical test

Note that we say that a square P *belongs* to a curve C when C crosses through P. Of course, the first requirement is not usually fulfilled by the original set of curves. The exclusion method works by removing parts of curves until the first requirement is fulfilled. Then, it removes the remaining pieces of curves that do not pass the test. See figures 9.1 to 9.4 for a graphical explanation.

There are in general non-unique ways to reduce the original set of curves so that the first requirement is true. We propose the following greedy strategy to force uniqueness:

1. Start with the set of all curves
2. While there are still curves that pass the test:
   a) Pick the curve C that passes the test with highest score
   b) The curve C owns all the squares that it crosses
   c) Delete the parts of all the other curves that cross through squares owned by C
   d) Output C and remove it from the set
3. Delete the remaining curves

Remark 2. *In the previous algorithm, the "curves" we speak about are not necessarily connected. For example, when we remove a piece in the middle of a curve, the remaining two pieces are still considered "one curve". This can be seen on the upper curve at Figure 9.4(b).*

Remark 3. *The proposed exclusion principle has a scale parameter, namely the size of the grid. We make the natural proposal to set it to the same size as the voxels of the original image.*

9.2 Size statistics and other heuristics

Let us study the cost of our algorithm. An image of m voxels has $O(m)$ level surfaces (in fact, it has exactly m when all the values are different). A typical level surface has $O(m^{\frac{2}{3}})$ points (this is the area of one side of a cube of volume m). This is a very rough estimate, which happens to underestimate the complexity of the algorithm for real images, for further empirical analysis on this topic see [CDD06] and for a mathematical justification see [AGM99]. Thus, the cost of traversing all the points of all level surfaces is about $O(m^{\frac{5}{3}})$, which is very large, and therefore too slow to scan the surfaces. Recall that a typical size for medical images is $m = 128^3$. We can make the algorithm much faster by discarding from the beginning those level surfaces which we know beforehand that will not produce any useful patches.

Here we discuss three ways to prune the input tree to reduce the number of processed surfaces: pruning very small surfaces, filtering the tree using robust statistics, and pruning the tree using the gray-level values. These are independent steps. The first one does not require any a priori information, but can incorporate it. The other two steps may be applied or not whether we have the required a priori information.

The first pre-processing step we propose is *pruning the smallest shapes of the tree*. When working with the tree of shapes of real images, one notices that usually most of the shapes in the tree have a small volume and belong to the noise that appears inside homogeneous regions (see Figure 9.6, right). We can realize this behaviour by plotting the number of shapes of each volume, as done in Figure 9.7. A good model for the number of small shapes in a textured region of volume M is a power law of the form $p(v) = \frac{M}{6v^{3/2}}$, meaning that there are about $p(v)$ shapes of volume v (see also [AGM99]). This model seems to be independent of the kind of noise and is quite accurate (for the purpose of realising that most shapes in the tree are rather small) for $v < 20$. On Figure 9.7 the number of small shapes for some images is plotted as dots, and the estimated power-law is plotted as a continuous line. The only differences that we observe are due to images with large saturated regions, where there is no texture.

All of these surfaces (say, of volume less than 10 voxels) are too small to pass the statistical test, so they can be discarded from the beginning. This will effectively discard most level surfaces of the image. While there is no study of the computational cost after this optimization, in practice this pruning helps to make the algorithm more tractable: for relatively small images of size 60^3 our implementation on a PC takes between one and five minutes, and we could process images of up to 128^3 voxels in less than one hour.

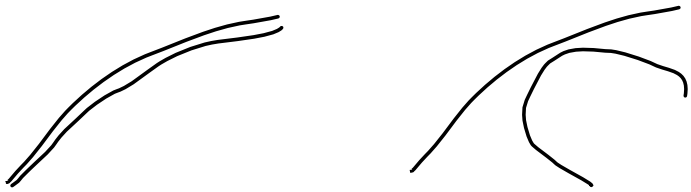


Figure 9.1: The two synthetic cases that we are going to consider below. Left: two curves "covering the same object". Right: two curves "covering different objects".

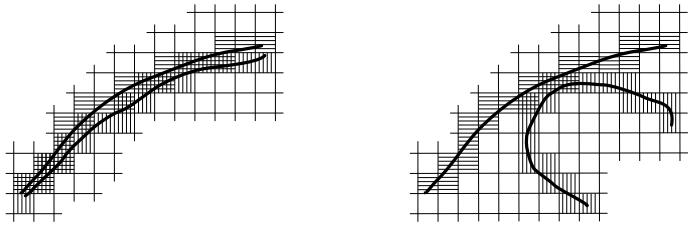


Figure 9.2: Marking the squares according to which curves cross each one.

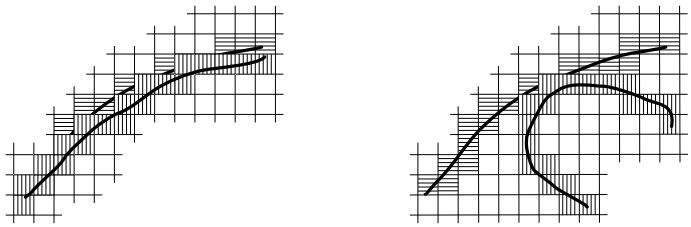


Figure 9.3: Assignment of at most one curve to each square, thus fulfilling the first requirement.

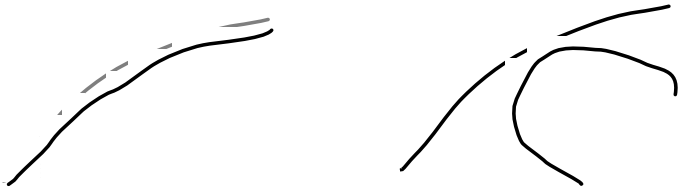


Figure 9.4: Performing the statistical test for the remaining pieces of curve, thus fulfilling the second requirement. Left: only the lower curve passes the statistical test. Right: both curves pass the statistical test.

The number "10" in the previous paragraph is only an example. An appropriate bound can be computed from the image data. For instance, when $f = \min$ and $\epsilon = 1$, a set of size l and minimum contrast μ is meaningful when $NH(\mu)^l < 1$, or equivalently when $l > -\log(N)/\log(H(\mu))$. Thus, if μ_1 is the last-to minimum contrast of the whole discrete image, then $-\log(N)/\log(H(\mu_1))$ is a lower bound for the size of a meaningful subset. This lower bound is not trivial when $NH(\mu_1) > 1$, which is usually the case. When f is another robust statistic, this bound is not easy to compute analytically, but it can nevertheless be obtained by a pre-computed table lookup to find the inverse of $F_l(\mu)$. These bounds on the area of meaningful level surfaces provide equivalent bounds for the minimal *volume* of a meaningful level surface, because for a discrete image the perimeter of a surface is bounded by its volume (each boundarying voxel is also part of the interior).

A second pre-processing step that can be applied consists in *pruning out low contrasted shapes*. As in [DMM01], the statistical tests of section 8.2 can be applied to the set of all level surfaces (without breaking those into pieces). The purpose of this pre-processing is to reduce the number of level surfaces that will be analyzed by the Mumford-Shah hierarchy, hence reducing the computational time. This may be a risk when the discarded surfaces have well-contrasted parts. But it could be justified if we have the a priori information that this is not the case.

The third pre-processing step, which is only useful in some common special cases, is *pruning the tree of shapes using* a-priori *information*. This data structure allows to use easily some *a-priori* information about the intensity ranges of the desired objects that may drastically reduce computational burden. For example, if we know that all gray values in some interval belong to noise, or to structures we are not interested in, we can immediately discard the level surfaces of those values. In X-ray computed tomography images, where the gray values have physical meaning, this means that we can discard most bones and background structures from the beginning and significantly reduce computation time and increase the quality of the output.

9.3 Surface Joining

Surface reconstruction is the problem of finding a closed surface $\Gamma \subseteq \mathbb{R}^3$ from a given set of surface samples $S \subseteq \mathbb{R}^3$. When the samples are themselves small patches of surface, the problem may be called surface joining.

All the following surface reconstructors

- Osher-Zhao: $E_0(\Gamma) = \int_\Gamma d_S$
- "reverse Osher-Zhao": $E_1(\Gamma) = \int_S |d_\Gamma|$
- Adjustment of distance functions: $E_2(\Gamma) = \int_{\mathbf{R}^3} |d_S - |d_\Gamma||$

- Bayesian interpolation: $E_3(\Gamma) = |S| \ln |\Gamma| - \int_S \ln \int_\Gamma p(y|x)\, d\sigma\, dy$ where p is, e.g., a Gaussian
- Combinations of the above ($E_0 + E_1$, etc.)

have trivial global minima. We would like to explore whether it is possible to construct a new functional whose global minimum gives a meaningful surface, at least when the data points are as good as possible. Maybe this is impossible, and then we could try using *hints* that are usually given to the problem, such as surface normals on the points. The purpose of having a functional with a good global minimum is that it will be efficiently implementable using graph cuts and without any user-supplied initialization.

Notice that there are ways to attack this problem which are not based on the minimization of energy functionals (e.g., Mémoli et al.). Here we only want to explore the functional approaches, or those approaches that can be reduced directly to functional minimization, even if they are not stated that way (e.g. Sagawa et al.)

Our edge detector does not produce a segmentation of the image domain into parts, but a set of boundaries. This may be enough for some applications like visualization or detection of structures, but it does not correspond to a segmentation. Let us discuss here a procedure to paste together a set of patches of level surfaces to produce a closed output surface (or a set of them). The method is based on a reconstruction algorithm introduced in [ZOF01].

Let $S \subseteq \Omega$ be a set of edges and let $d_S : \Omega \to \mathbb{R}$ be the distance function to S. Suppose that S covers part of the boundary of an object. A common approach to recover the whole boundary of the object is to search for closed surfaces Γ that are local minima of the following functional

$$E_0(\Gamma) = \int_\Gamma d_S(x) dA,$$

where dA denotes the area element. These minima can be found by starting from an initial guess, a user-supplied closed surface which approximately contours the object, and then letting it evolve by gradient descent of E_0. The Gâteaux derivative of E_0 is

$$\nabla d_S(x) \cdot N + d_S(x)\kappa$$

where N is the outwards unit normal to Γ and κ its mean curvature. In an implicit formulation, where Γ is the zero level set of a function ϕ, the gradient descent of E_0 can be described by the evolution of the following PDE:

$$\frac{\partial \phi}{\partial t} = \left(\nabla d_S \cdot \frac{\nabla \phi}{|\nabla \phi|} + d_S \operatorname{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) \right) |\nabla \phi| \,.$$

Further regularization (mainly for display purposes) can be added to the model if we replace the functional above by adding the term $\lambda \int_S dA$ to the above functional, $\lambda > 0$. For large values of λ the functional approaches the area

times λ, and its optima approach minimal surfaces. For small values of λ, only sharp edges are smoothed out.

This reconstruction method is not definitive (it will fail at junctions where more than three regions meet), but it usually improves the quality of the visualization by smoothing out the ragged appearance of the surface patches. It has two important problems to be used in full generality for this application. The first problem is the choice of the initial surface. Examples of reasonable initial surfaces are the boundary of the image or the most meaningful shape of the tree, but either method can fail when there are multiple objects to be detected, specially when they are nested. The second problem is that the functional E_0 has a global minimum of zero (attained at the empty surface). In practice, this means that the minimization can collapse if the surface "misses" the objects that we want to reconstruct.

9.4 Experimental results

We display and comment the results of our method when applied to a few sample images: two synthetic images, a magnetic resonance, and an X-ray computed tomography. In the synthetic images the task is to find the boundaries that generated them. In both medical images the task is to find the border of a vessel that contains a cerebral aneurysm. We compare the results with those obtained by simple thresholding and by Canny's filter. We also illustrate the difficulties that appear when we try our edge detector for the video segmentation

9.4.1 Experimental results on synthetic images

The first example is a synthetic image built in the following way. A sphere of radius 15 has been drawn at the center of a 40^3 black image, and the interior of the sphere has been colored in three different homogeneous regions. Then, a gaussian noise of variance 10 has been added to the image, to add some texture. Figure 9.10 shows a slice of this image, and the output of the proposed edge detector. Notice that the output is a set of three smooth level surfaces, corresponding to the boundaries of the four large homogeneous regions on the image. This first example is a best case for our method, and serves as a check that the algorithm is working well. Notice that no global threshold can produce all the boundaries of the image, and that Canny's detector misses the junctions.

The second example is a synthetic image built in the following way. A black sphere of radius 19 has been drawn at the center of a 50^3 white image, and then we added to it a ramp function of slope 1. This means that the contrast of both the background and the inside of the sphere are constant (equal to 1) and the borders have a much higher contrast. Then the image has been made more textured by adding a gaussian noise of variance 5 and blurring it with a gaussian of width 3. Two level surfaces of this image are shown in the right part of Figure 9.11. Notice that no level surface can surround the whole sphere, but that many surfaces contain a band touching the sphere. This image is an

example of a worst case for our algorithm (and a best case for Canny's). However, the algorithm manages to find the good parts of all level surfaces, as shown on Figures 9.11 and 9.12.

Remark 4. *The two synthetic images above are intentionally low-resolution to emphasize the sub-pixel accuracy of the output.*

9.4.2 Experimental results on medical images

The first real example we show is the computed tomography image discussed in the introduction (see Figure 8.1). Its size is $180 \times 84 \times 72$. It is a noisy image with several artifacts (*e.g.* dark shadows, radial anisotropic noise), due to the reconstruction algorithm. The proposed edge detector finds the correct boundaries at several difficult places, but still misses some small arteries. See Figure 9.13 for a discussion on the image, and Figure 9.14 for the output of the proposed edge detector, and a comparison with Canny's. After linking the patches via the functional described on Section 9.3, we find a single surface which is better than the best manually-set global threshold.

The second real image is an anatomic MRI image of size $111 \times 65 \times 57$. While this image is not as noisy as the previous one, it has an artifact which produces a problem like that of the second synthetic example. Namely, the image domain is partitioned into three bands, where the gray levels have a different starting point (so that the histograms are displaced). Even if this is an artifact easily tractable in a pre-processing stage, the proposed edge detector produces good results when applied directly on the raw data. See Figure 9.16 for slices and projections, and Figure 9.17 for the result of the proposed edge detection, compared to a manually selected isosurface.

9.4.3 Experimental results on videos

We can apply the proposed edge detector to gray-level videos, when we regard them as 3D images. The resulting output is a set of surfaces, which can be intersected with each frame to produce a set of curves. This can be interpreted as a set of edge curves evolving in time. A similar result can be obtained by applying a 2D edge detector to each frame independently. The advantage of the 3D approach is that it provides *temporal coherence*: we know that this edge on frame $t = 0$ is corresponds to that other edge on frame $t = 10$; the tracking is already done. The disadvantage is the memory constraint, imposed by the fact that we have to keep an uncompressed video into memory together with several additional structures. Thus, only a few seconds can be processed at once.

How good are the results of our edge detector on videos? That depends on how good are the level surfaces of the video, because the result of the edge detector is a selection of them. This is equivalent to the level curves of each frame containing good edge curves; which happens almost always, except for images where the objects are defined by different textures of the same colors.

There is, still, a subtlety concerning the way the gradients are used as a local contrast function: only the spatial part of the gradient is needed. Using the whole spatio-temporal gradient makes no sense and could lead to very bad results. To understand this fact, let us first look at the best possible case for edge detection on video: a smooth black disk slowly moving on a white background, at about one pixel per frame (Figure 9.18). The tree of shapes of the resulting 3D image is a stack of nested "tubes". The central tube has the largest overall contrast and is selected as a single surface by the edge detection algorithm. In this case, the 3D gradient works well. Now imagine that the disk is moving faster, at about 20 pixels per frame (Figure 9.19. The visual effect of the video is very similar, just faster. The structure of the level surfaces is also a single stack of nested tubes. However, the behaviour of the gradient norm is very different. The highest gradients do not occur on the boundaries of the disk at each frame, but on the interior of the disk, over the pixels which on the next frame are not occupied by the disk (corresponding to occlusions and disocclusions). The gradient norm along the boundaries of the disk is lower than that, since the disk is smooth. Thus, the result of the 3D edge detector is a set of lunes, perpendicular to the temporal direction, marking the occlusions and disocclusions of the disk as it moves. This is a wrong result. A correct result can be obtained immediately if we replace the norm of the 3D gradient by the norm of the spatial component of the gradient. With this slight modification, the edge detector can be used to find temporally coherent edge curves in many videos.

We can treat even faster movements, where the objects share no pixels between frames (all the pixels are occlusions and disocclusions). For that, external information from a pre-computed flow-field may be used, and compute the trees of regions using the graphs of optical flow as in Section 5.4, instead of a regular 3D grid.

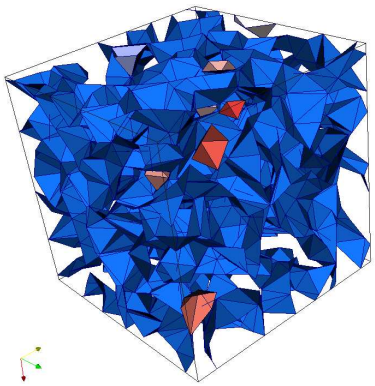


Figure 9.5: Isosurface at level 0 of a gaussian noise image of size 10^3. This surface has 2356 triangles, which is larger than the number of voxels of the image. This is the typical behaviour of the isosurfaces that cross through large regions of the image which have almost constant value (plus some texture).

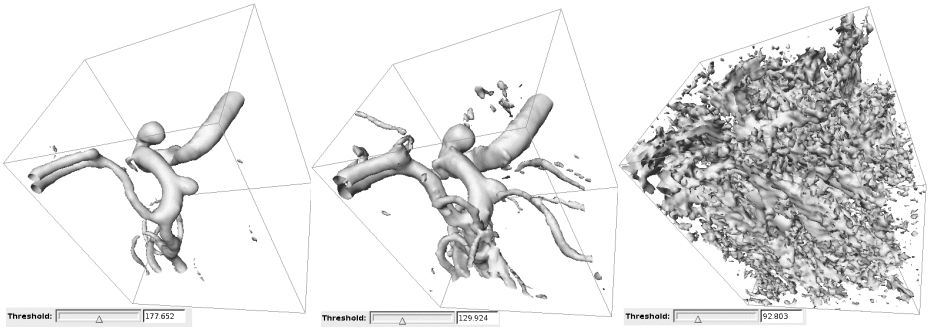


Figure 9.6: Three thresholds of the same image. The first two thresholds show some image content. The third one shows mainly background noise. The level surface on the third image has one large and very convoluted component and many small and almost spherical components.

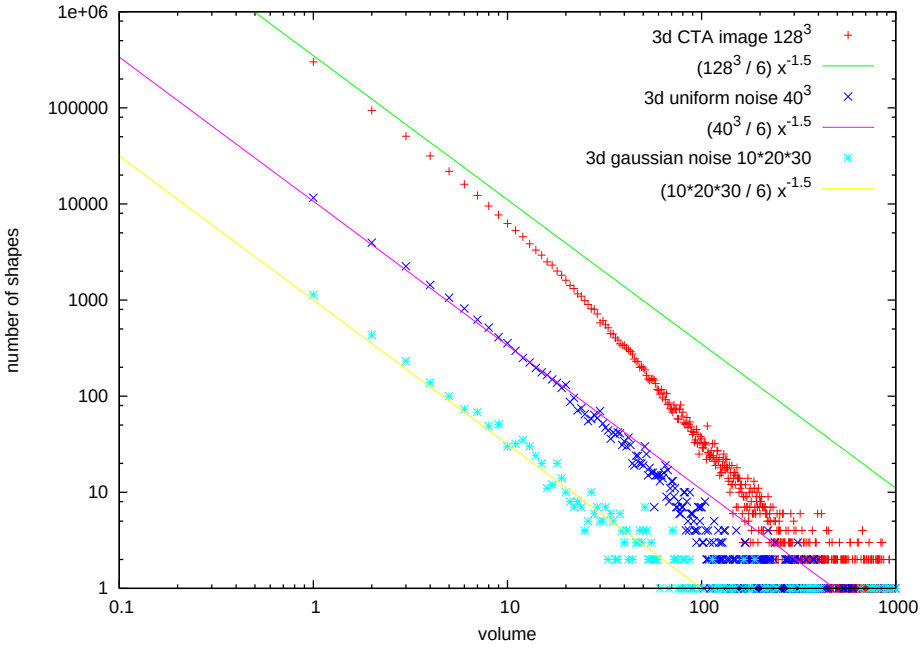


Figure 9.7: Number of shapes of each size, for some images. The interesting pattern is that all the curves start decreasing more or less linearly with slope $-\frac{3}{2}$ (on the $\log - \log$ scale shown here). This means that the number of shapes of each size decreases very fast, as there are $\propto v^{-1.5}$ shapes of volume v, for small v, and these small shapes belong to the texture of noise. This also means that most of the shapes on the tree are small: in a typical tree of shapes, about half of the shapes enclose a single voxel.

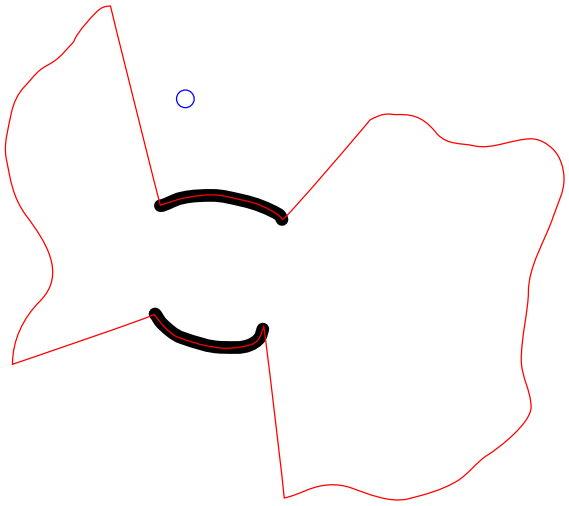


Figure 9.8: Example of bad minima in some of the above functionals. Black: points that we want to approximate as a closed curve. Blue: curve close to a global minimum of E_0. Red: global minimum of E_1.

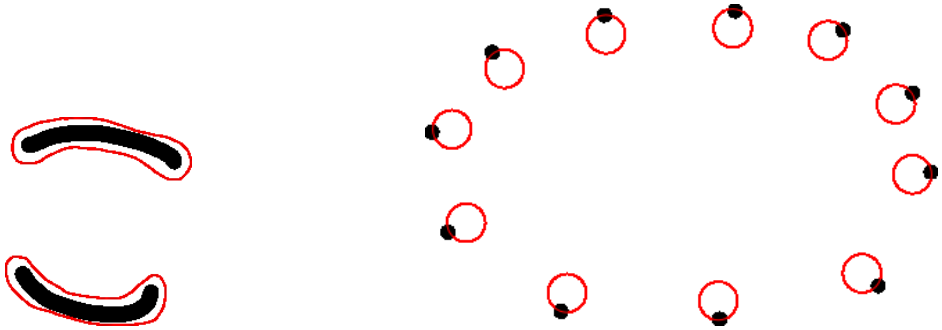


Figure 9.9: Example of data overfit. Black: points that we want to approximate as a closed curve. Red: curves close to a global minimum of the functionals, over-representing the data.

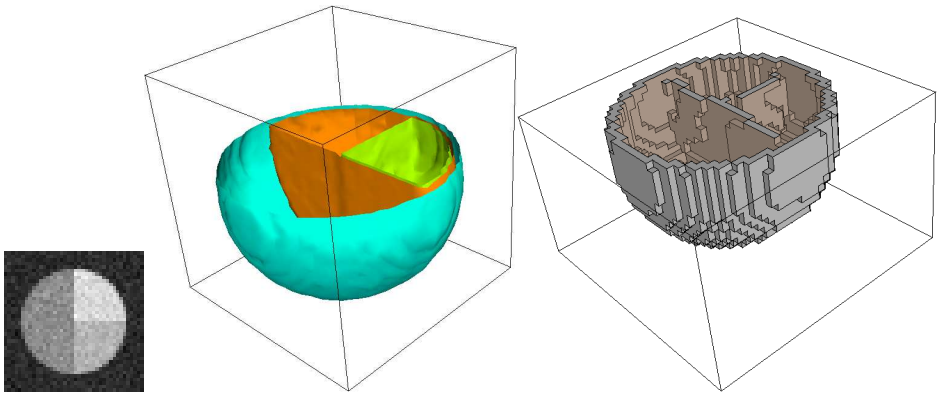


Figure 9.10: Best case for our algorithm. Segmentation of a piecewise constant image with added texture. From left to right: slice of the image, output of the proposed edge detector, output of Canny's edge detector. The 3D images are clipped to show the interior of the object.

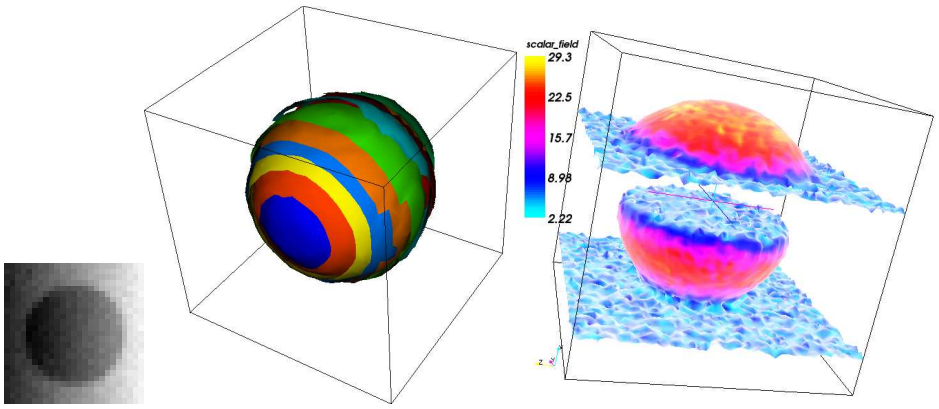


Figure 9.11: Worst case for our algorithm. Segmentation of a piecewise constant image with an added ramp function. From left to right: slice of the image, output of the proposed edge detector, two isosurfaces of the image.

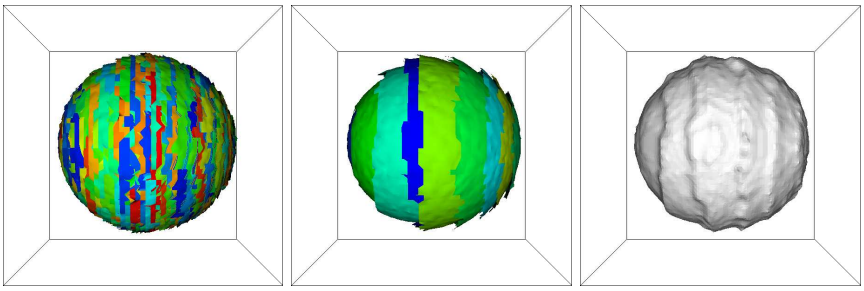


Figure 9.12: Effect of the post-processing pipeline on the synthetic worst-case image (Figure 9.11). Left: before exclusion principle, 207 patches. Middle: after exclusion principle, 9 patches. Right: after edge linking, one single surface patch. In this figure, the edge linking is performed using a higher resolution than the input image and without any smoothing. When using the same resolution as the input image, an almost perfectly spherical surface is obtained.

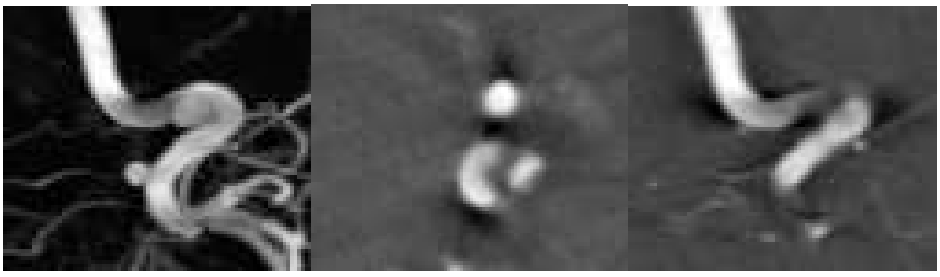


Figure 9.13: A real CT image. From left to right: Maximum intensity projection, vertical slice, horizontal slice. The slices show several artifacts, and the anisotropy of noise.

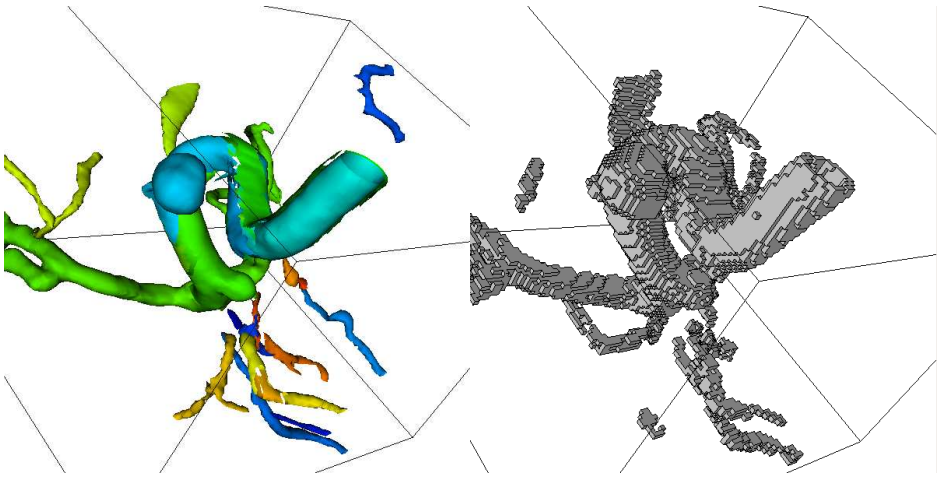


Figure 9.14: Edge detection on the CT image. Left: Output of the proposed edge detector. Right: Output of Canny's edge detector, using hand-tuned parameters to obtain the best result for this image. The main advantage of the proposed method, besides the lack of tunable parameters, is the format of the output: instead of a large set of voxels we have a small set of triangulated surfaces, sampled at sub-voxel precision.

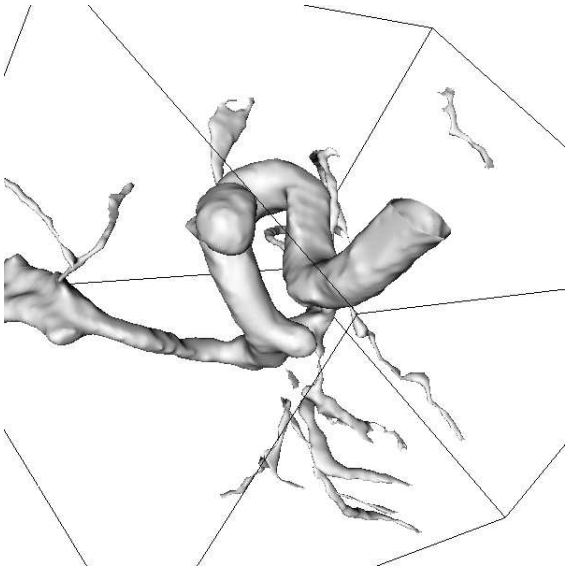


Figure 9.15: Output of the experiment on Figure 9.14, after joning the edges via Osher-Zhao functional. The result is a clean set of five disjoint surface patches.

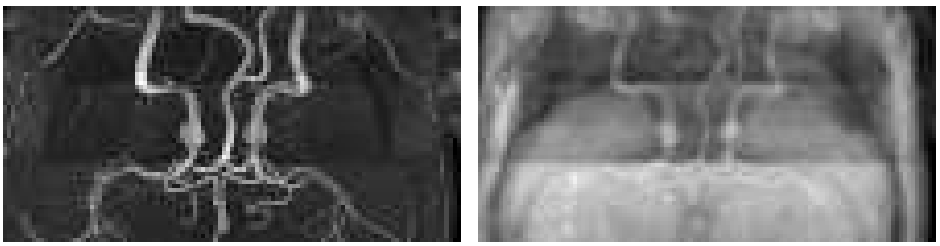


Figure 9.16: A real MRI image. Left: Maximum intensity projection. Right: Average projection (notice the different averages over three bands). The results of our edge detector on this image appear on Figure 9.17

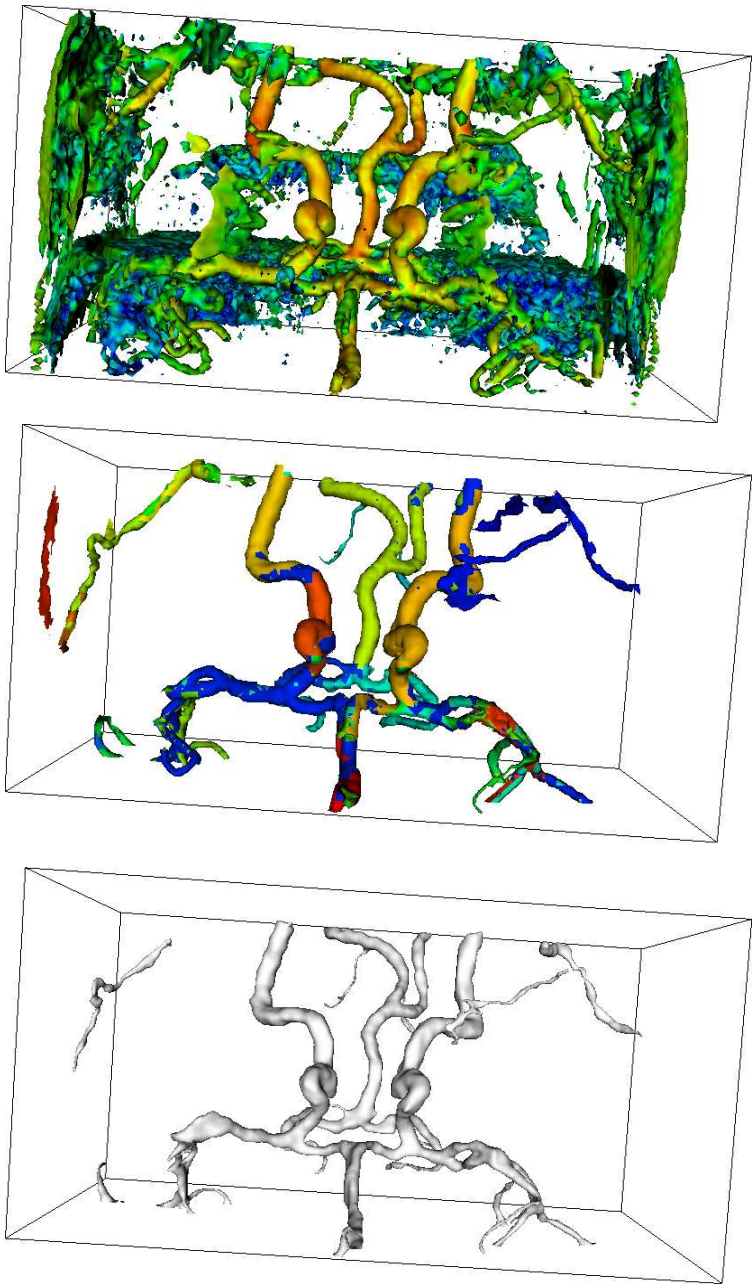
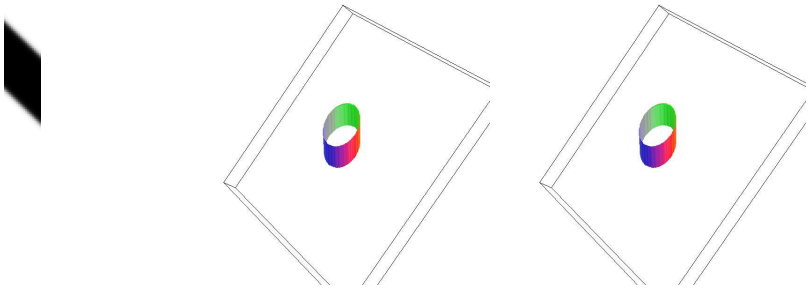
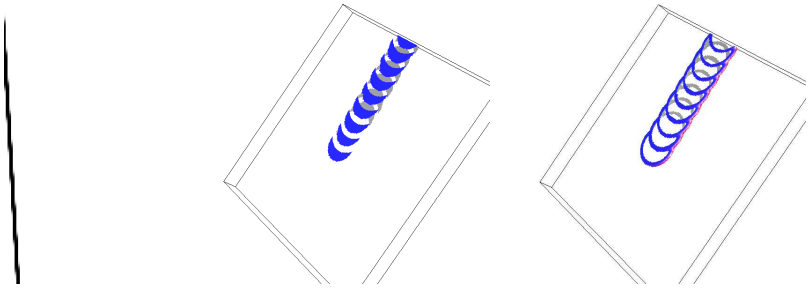


Figure 9.17: Processing of the real MRI image from Figure 9.16. Top: manually selected isosurface that segments well the upper part of the image. Middle: output of the proposed edge detector, 14 surface patches. Notice that the two structures which are not vessels (they are parts of bones) can be easily removed manually. Bottom: result of edge linking via Osher's functional.



(a) Temporal slice closeup (b) Result (3D gradient) (c) Result (2D gradient)

Figure 9.18: Video of a slowly moving smooth disk, and results of our 3D edge detector using different contrast measures. Since the movement is slower than a pixel per frame, there is no difference on the results.



(a) Temporal slice closeup (b) Result (3D gradient) (c) Result (2D gradient)

Figure 9.19: Video of a fast moving smooth disk, and results of our 3D edge detector using different contrast measures. Only the result using 2D gradients makes sense.

Part III

Cheeger sets and affine invariants

10 Historical context of Finsler-Cheeger sets

This part of the thesis was originally motivated by its application to edge linking. We formulated the problem of edge linking in terms of the computation of an anisotropic Cheeger set. Incidentally, this formalism encompasses also other image processing tasks, like the computation of active contours or the problem of colorization by diffusion of color samples along the level lines of a gray-level image. We discovered other connections with the generation of affine invariants associated to level sets and level lines and gave and interpretation of the MSER affine invariant as a kind of Cheeger sets. Thus, the main themes of this chapter are: Finsler metrics, Cheeger sets, total variation, partial differential equations and affine invariants.

10.1 Historical context of Finsler metrics

After more than a thousand years trying to prove Euclid's fifth postulate, people finally realized that Euclidean geometry was not the only geometry possible. The first complete description of a non-Euclidean geometry, by Bolyai and Lobachevsky, was done in an intrinsic manner, by giving a list of axioms for points and lines. In modern terms, this was the definition of a space of constant negative curvature. Soon, it was clear how to build actual models of this and of many other non-Euclidean geometries. It fact, since ancient times these models of non-Euclidean geometries had been in plain view: surfaces. Gauss' theory of surfaces clarified the notion of intrinsic geometry, by identifying the properties of a surface which do not depend on the way this surface is embedded in space (curvature). However, the metric on the surface was still inherited from the Euclidean metric of space. It was Riemann, in 1854 [Rie54], who made the huge leap of abstraction and defined metric as an arbitrary field of quadratic forms on a manifold (he also defined the notion of n-dimensional manifold at the same time). All the previous geometries, Euclidean and non-Euclidean, turn out to be particular cases of Riemannian geometry.

Let us recall the formal definition of Riemannian metric, adapted to our purposes. Let M be an open rectangle in $\mathbb{R}^n$. A function which assigns to each point $x \in M$ a symmetric, positive definite, matrix G_x is called a Riemannian metric on M. The length of a vector ξ emanating from point x is defined as $\sqrt{\xi^t G_x \xi}$. This definition assures that the set of vectors emanating from a single point (called the tangent space at that point) is a normed vector space. Standard constructions define the length of arbitrary curves on M, and the geodesic distance between pairs of points of M. See equations 10.1 and 10.2 for these constructions in a slightly more general setting.

Traditionally, the definition of Riemannian manifold is given in a more general setting than in the previous paragraph, as a metric structure on an abstract manifold, not necessarily a rectangle of $\mathbb{R}^n$. There are two interpretations of these metric structures: extrinsic, or coming from an embedding of the manifold into a higher-dimensional Euclidean space; or intrinsic, coming from an arbitrary

metric field defined on a given Euclidean space. These two interpretations are equivalent via parametrizations on one sense, and (locally) via Nash embedding in the opposite sense. However, they have a very different *flavour* and they lead to different intuitions about the same objects. See Figure 10.1 for an example of these two views. Of course, for the kind of metrics used in image processing it is more natural to think of Riemannian manifolds as metric fields on a fixed flat space.

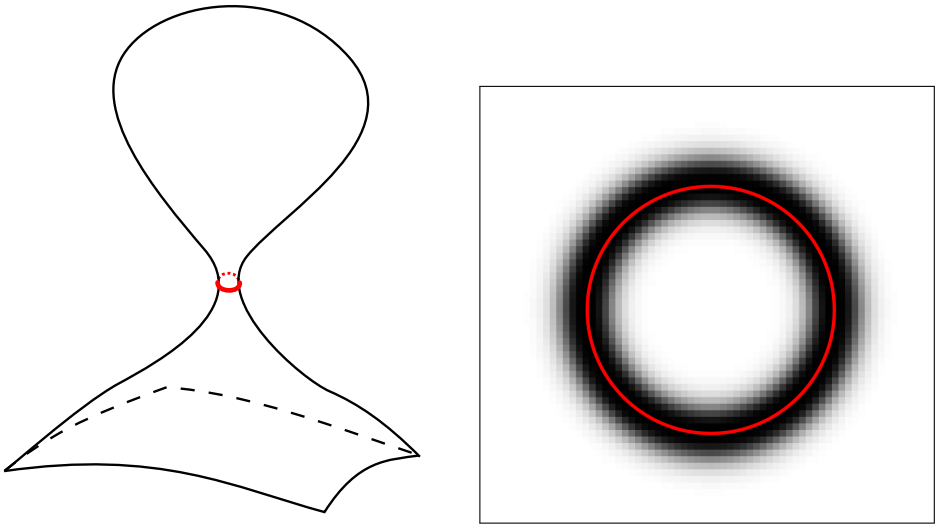


Figure 10.1: Two views of the same Riemannian manifold: a complicatedly curved submanifold of a flat space, or a flat manifold with complicated metric field defined on it. In both cases, a closed geodesic is shown.

Today, Riemannian metrics are a standard tool in many scientific disciplines. Perhaps the most well-known is general relativity, where space-time is modelled as a Riemannian manifold, and matter is defined from the curvature of the metric. For our purposes, we will focus on the applications of Riemannian geometry to image processing. In that field, many ideas and methods are derived from physical analogies. For example, to blur an image corresponds to apply the heat equation using the image data as initial condition. Thus, it is often quite natural to replace the Euclidean metric of the simplest models with Riemannian metrics, in order to obtain more powerful models. In that case, the metric is usually derived from the image contents. The first use of Riemannian metrics for image analysis can probably be traced back to the *snakes* of Kass-Witkin-Terzopoulos [KWT88]. These snakes are curves which evolve to minimize a length, locally weighted by the image contents in some way. For example, if the length is weighted by a decreasing function of the image gradient, the minimal curves tend to approach sharp edges of the image. Although the inventors of snakes did not use a geometric language, the snakes were later identified as geodesics of a suitable Riemannian

manifold by Caselles-Kimmel-Sapiro [CKS97]. An independent usage of Riemannian metrics in image processing is the introduction of *anisotropic diffusion* by Perona-Malik [PM90], which is the diffusion equation on a suitable Riemannian manifold[1]. This is useful because it allows to enhance the images by allowing diffusion on flat zones, removing noise, while disallowing diffusion along sharp edges, preserving their sharpness. See [Wei98] and [KMS00] for reviews of different metrics used for image diffusion.

Despite its wide usage, Riemannian metrics are not the most general metric structures possible. For example, when studying crystals, it is found that the speed of light inside a crystal depends on the direction. This variation of speed is not well-modeled with a Riemannian metric, because the "unit ball" is not an ellipse, but a convex polyhedron. Actually, it is the dual polyhedron of the shape of the (perfect) crystal [Tay78], [PT04]. Thus, an appropriate generalization of Riemannian metrics is obtained by dropping from the definition the requirement that they be given by a quadratic form (or equivalently, that the local unit balls are ellipses), and allowing arbitrary convex functions (or equivalently, that the local unit balls are arbitrary convex sets). This generalization of Riemannian metrics is precisely Finsler metrics.

Finsler metrics are the most general metric structures on manifolds. Given a manifold M, a Finsler metric φ assigns a length $\varphi(x,\xi)$ to any vector ξ emanating from each point $x \in M$. Such a function, called a *metric integrand* can be used to define the length of any directed piecewise smooth curve $\gamma : [0,1] \to M$ as

$$\text{length}(\gamma) := \int_0^1 \varphi(\gamma(t), \gamma'(t))dt. \tag{10.1}$$

To assure that the lengths of curves are independent of their parametrizations, the only requirement on the function φ is to be positively homogeneous on the second argument: $\varphi(x, \lambda\xi) = \lambda\varphi(x,\xi)$ for any $\lambda > 0$. The *distance* between two points of M can be defined as the infimum of the lengths of all curves joining them:

$$d(p,q) := \inf\{\text{length}(\gamma) \;\mid\; \gamma : [0,1] \to M,\ \gamma(0) = p,\ \gamma(1) = q\} \tag{10.2}$$

This distance is a positive function on pairs, which vanishes only when $p = q$, and satisfies the triangle inequality. In general it is not symmetric, $d(p,q) \neq d(q,p)$. The symmetry of d corresponds to the symmetry of the Finsler metric: $\varphi(x,-\xi) = \varphi(x,\xi)$. Together with homogeneity, the symmetry property can be written as $\varphi(x,\lambda\xi) = |\lambda|\varphi(x,\xi)$, for $\lambda \in \mathbb{R}$.

Besides symmetry, there are other conditions on φ which give interesting particular cases of Finsler metrics:

1. symmetric: $\varphi(x,-\xi) = \varphi(x,\xi)$
2. uniform: $\varphi(x,\xi) = f(\xi)$

[1] Actually, the authors of [PM90] proposed a more complicated nonlinear model where the metric evolves alongside the image

3. isotropic: $\varphi(x, \xi) = g(x)|\xi|$

4. riemannian: $\varphi(x, \xi) = |A(x) \cdot \xi|$, where $A(x)$ is a non-singular linear map.

Each of these particular cases has different applications. For example, isotropic Finsler metrics are widely used in image processing (sometimes, under the confusing name of "anisotropic", e.g. in [PM90]), to model image-dependent lengths of curves, and as a background for image-dependent diffusion. Uniform Finsler metrics are used in crystallography, as a model for the speed of growth of a crystal along each spatial direction. A diagram of many possible particular cases is shown on Figure 10.2.

For technical reasons, namely that $\xi \to \varphi(x, \xi)$ is a norm, it is also required that φ be convex on its second argument. Other technical conditions, such as linear growth or coercivity, will be introduced when they are needed.

10.2 Historical context of Cheeger sets

Cheeger constants, and their associated Cheeger sets, are powerful tools in the study of global properties of spaces. In the case of compact Riemannian manifolds, as originally introduced by Cheeger [Che70], they provide a bound for the first eigenvalue of the Laplacian. This continuous theory has a discrete analogue in graph theory, where the Cheeger constant of graphs, often called the conductance (e.g. [Bol98], p.221), opens the door to the rich world of spectral graph theory [Chu97]. There are many related definitions of "Cheeger constant", both in the continuous and discrete setting.

The continuous, and original, definition of the Cheeger constant applies to compact manifolds, with or without boundary. Given an n-dimensional compact manifold with boundary Ω, its Cheeger constant is defined as

$$h(\Omega) := \inf_{A \subseteq \Omega} \frac{|\partial A|}{|A|}. \tag{10.3}$$

In the previous formula, $|\partial A|$ is the $(n-1)$-dimensional measure of the boundary of A, and $|A|$ is the n-dimensional measure of A. The intuitive interpretation is that a set A that minimizes the "Cheeger ratio" $\min \frac{|\partial A|}{|A|}$ wants to have a volume large as possible, while having a boundary as small as possible. If Ω is a closed domain of Euclidean space, a dimensionality argument proves that minimizers of the Cheeger ratio will touch the boundary Ω (otherwise they could be scaled by a positive constant to obtain a smaller ratio). A Cheeger set of Ω is any set for which the infimum in 10.3 is attained. They are known to be unique for the case of convex or callibrable sets [AC09].

A related definition, which applies to manifolds without boundary, is

$$h(\Omega) := \inf_{\substack{A \subseteq \Omega \\ 0 < |A|}} \frac{|\partial A|}{\min\{|A|, |\Omega \setminus A|\}}. \tag{10.4}$$

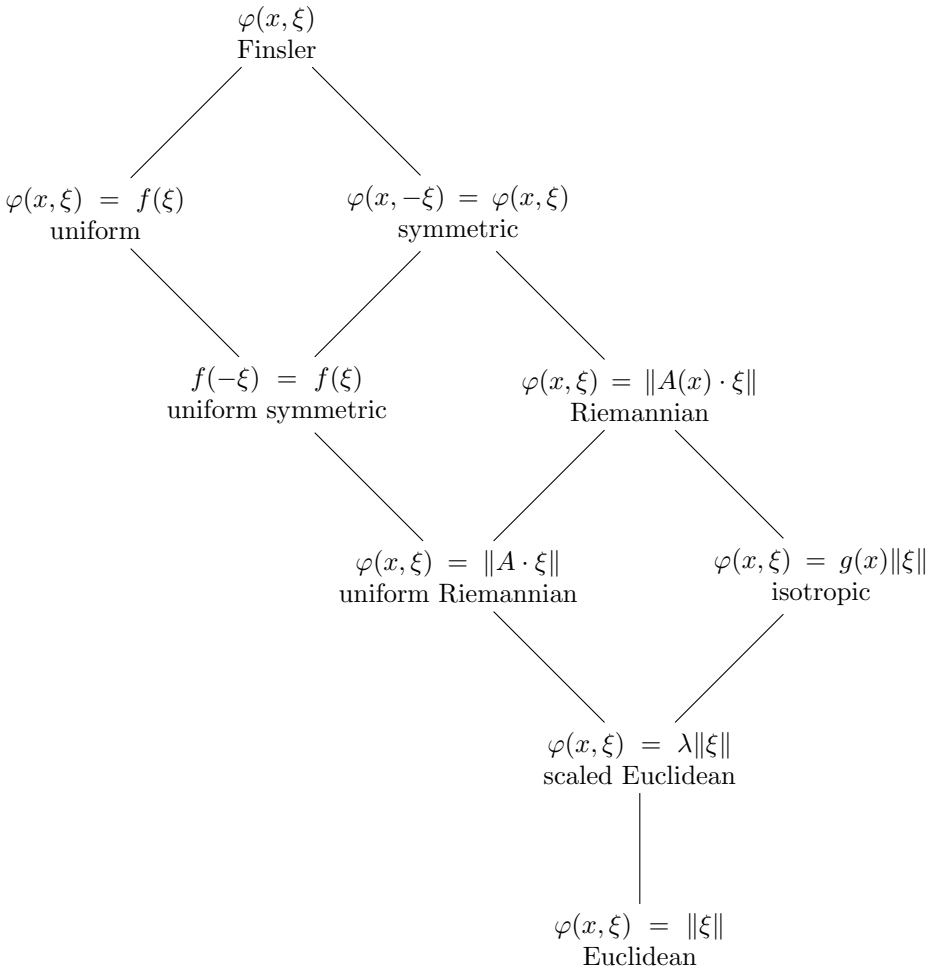


Figure 10.2: Particular cases of Finsler metrics. There also many technical conditions not displayed on this table, such as convexity, linear growth and coercivity.

In that case, the interpretation is different. Here the Cheeger constant measures how thin is the thinnest "bottleneck" on Ω, with respect to the volume of Ω.

The discrete definition applies to graphs. It is based on the continuous definition for manifolds without boundary. Given a graph $G = (V, E)$, the *Cheeger constant* of G is defined as

$$h(G) := \min_{\substack{A \subseteq V \\ 0 < |A| < \frac{1}{2}|V|}} \frac{|\partial A|}{|A|}.$$

The value of the Cheeger constant $h(G)$ measures the thinnest "bottleneck" of

G. It is strictly positive if an only if G is connected. The fact that $h(G)$ is very small means that there are two sets of vertices which are connected by a very few edges (a bottleneck). The fact that $h(G)$ is large means that any partition of V in two subsets has many edges between the two subsets. Notice that an equivalent definition is

$$h(G) := \min_{\substack{A \subseteq V \\ 0<|A|}} \frac{|\partial A|}{\min\{|A|, |V \setminus A|\}}$$

The relationship between the discrete and the continuous definitions is explained in the review paper by Brooks [Bro93], and fully analyzed for the particular case of random geometric graphs on a recent article by Arias-Castro [PPAC10].

Cheeger constants and Cheeger sets have been seldom used in image processing, sometimes for segmentation [SM02], but mainly as a technique for partitioning the high dimensional graphs that appear in clustering of large image datasets [CM07], [JJH06]. Other works use structures which are closely related to Cheeger sets in anisotropic metrics, without saying explicitly so [MCUP04], [SMS02].

10.3 Historical context of variational problems and PDE

Variational problems are optimization problems in a continuous setting, often with infinite degrees of freedom. These degrees of freedom are typically represented either by a subset of some space, or by a mapping between spaces. The definition of Cheeger constant given above is an example of variational problem defined on sets, where the objective function to be minimized is the perimeter/area ratio. The cases of sets and of real-valued functions are closely related, because a subset of a space Ω can be represented by its $\{0, 1\}$-valued indicator function or, more generally, as the level set of some function. The process of transforming a variational problem from sets to functions can be formalized as a relaxation, or convexification, and is very useful, since the tools of functional analysis can be applied to the transformed problem. This is a very useful step, because the collection of all appropriate functions, being a topological vector space, has a richer structure than the collection of all subsets.

A general method for solving many variational problems is the *direct method*. Let L be a space of functions and let $F : L \to \mathbb{R}$ be a functional which we want to minimize, i.e., find a function $u \in L$ where the infimum

$$\inf_{u \in L} F(u) \tag{10.5}$$

is attained. The scheme of the direct method is as follows: by definition of infimum there exists a sequence $u_n \in L$ such that $\lim_{n\to\infty} F(u_n) = \inf_{u\in L} F(u)$. In general, the sequence u_n is not convergent in the space L. However, under suitable compactness conditions, we can extract a subsequence of u_n which is convergent in L to a certain function u. If F is lower-semicontinuous (a very general and easy to meet condition), then the function u is a minimizer of the

variational problem. This proves existence of the solution. Uniqueness is a more delicate issue which must be dealt with using other methods; typically variations of the maximum principle, or by approximating the given functional F with a sequence of functionals F_n with better properties.

In image processing, variational problems are extremely common tools [MS95]. Many problems in image analysis are directly stated as variational problems. For example image segmentation consists in finding the "best" partition of the image domain into two parts, or into an arbitrary number of parts. Each definition of "best" is usually described by a functional that evaluates the partition, which has to be minimized. This functional is often an integral on the boundary of the partition (such is the case, for example, for Active Contours [KWT88, CKS97]), an integral on one part of the partition, or a weighted sum of two kinds of terms ([MS88]). Sometimes, these methods are presented by giving a computational solution to a variational problem that is not explicitly stated. Another example of variational problem arises in non-rigid registration [PCA99], where two images have to be matched by an arbitrary deformation, and there are many functionals imposing smoothness of the deformation, and measuring the correctness of the match. A general form of many variational problems for image analysis has two terms: a regularity term and a data attachment term. This common case has a probabilistic interpretation, where the regularity term corresponds to a prior of the model, and the data attachment corresponds to the likelihood [Mum94].

Partial differential equations are closely related to variational problems, because a problem such as 10.5 can be often be rewritten, at least symbolically, as $F'(u) = 0$. Many PDE do not have classical solutions, that is, smooth functions satisfying the expression of the equation. Even when they have a classical solution, it is very difficult to prove its existence directly. Instead, it is better to use generalized solutions. Thus, a standard method of "solving" a PDE involves three steps: (1) define a space V of suitable functions; (2) define what does it mean for a function of V to satisfy the PDE; (3) prove existence and uniqueness. The definitions on (1) and (2) are generally ad-hoc, so that the proof of existence and uniqueness can be constructed. In practice, one starts with a standard proof of existence and uniqueness, and fine-tunes the definitions of the functional space and the solution until the proof is right. The proof usually involves re-writing the PDE as a variational problem, which is then solved by a direct method (find a minimizing sequence, etc.). As a last step, one may (4) recover a classical solution from a generalized solution by checking its regularity once it has been constructed.

Just like variational problems, PDE are used extensively in image processing [Sap01]. First, they arise as methods to solve variational problems, where an arbitrary initial function is evolved towards the minimum of the desired functional. But they also have interest in themselves, specially diffusion and related equations, as a method to produce a scale-space out of a given image [Wit83], [FtHRKV92], [AGLM93].

10.4 Historical context of total variation

The total variation of a single-variable function $f : [a, b] \to \mathbb{R}$ is defined as

$$TV(f) := \sup_{a=x_0<\cdots<x_n=b} \sum_{i=1}^{n} |f(x_i) - f(x_{i-1})| \tag{10.6}$$

where the supremum runs over all partitions of the interval $[a, b]$ into a finite number of sub-intervals $[x_i, x_{i+1}]$. The space of functions of bounded variation $BV([a, b])$ is defined as the set of functions f such that $TV(f) < \infty$. There are many elementary results describing the nice properties of single-variable functions of bounded variation. For example, the space $BV([a, b])$ can be characterized as the set of functions which can be expressed as the difference of two monotone functions. As a more interesting example, when $f : [a, b] \to \mathbb{R}$ is differentiable, we have that

$$TV(f) = \int_a^b |f'(x)| \mathrm{d}x, \tag{10.7}$$

but this formula is less general than the definition, which works for arbitrarily discontinuous functions.

For higher dimensional functions, the situation is more complicated: even the definition of total variation is difficult to write. If Ω is a domain of $\mathbb{R}^n$, the definition given by 10.6 can not be generalized directly to functions $f : \Omega \to \mathbb{R}$. However, formula 10.7 has a direct analogue:

$$TV(f) = \int_\Omega |\nabla f(x)| \mathrm{d}x. \tag{10.8}$$

This formula provides a definition for the total variation of smooth functions on Ω. This definition can be generalized to arbitrary functions by using derivatives in the sense of distributions. Namely, if u is a differentiable function, the function $|\nabla u|$ equals the point-wise supremum of $\nabla u \cdot z$, for all vector fields $z : \Omega \to \mathbb{R}^n$ such that $|z(x)| \leq 1$. The same supremum is achieved almost everywhere even if we restrict z to differentiable vector fields compactly supported within Ω. Thus, swapping the supremum and the integral we have

$$TV(u) = \int_\Omega |\nabla u| = \sup_{z \in C_0^\infty(\Omega, \mathbb{R}^n) : |z(x)| \leq 1} \int_\Omega \nabla u \cdot z \tag{10.9}$$

and, integrating by parts,

$$TV(u) = \sup_{z \in C_0^\infty(\Omega, \mathbb{R}^n) : |z(x)| \leq 1} - \int_\Omega u \operatorname{div} z. \tag{10.10}$$

Now, this coincides with 10.8 when u is smooth, but it can be applied to arbitrary (measurable) functions u. This is indeed the generalized definition of total variation for functions of several variables. More formally, measure theory

provides the appropriate framework to talk about total variation, whence any function $u \in L^1(U)$ gives rise to a (vector-valued) Radon measure Du, which is its gradient in the sense of distributions. The total variation of this vector-valued Radon measure is precisely $TV(u)$. Two textbooks on the kind of measure theory that we need here are [EG92a] and [AFP00].

Total variation is closely related to Cheeger sets, through a chain of ideas that links the eigenvalues of the Laplacian, isoperimetric constants (similar to the Cheeger constant, but dimension-less), and dual variational problems. See the original article of Cheeger [Che70], and [Gri06] for a very readable overview. From our point of view, this relationship can be stated succintly in the following way. Let us consider the minimization problem

$$\min_{u \geq 0} \frac{TV(u)}{\|u\|_{L^1}}. \tag{10.11}$$

Now, using the coarea formula, this problem can be restricted to functions u which are characteristic functions, leading directly to the Cheeger ratio. Thus, algorithms for minimizing the total variation lead to methods of calculation of Cheeger sets [Cha04], [BCC07].

Independently of its relationship to Cheeger sets, total variation has seen many fruitful applications to image processing. Its main interest comes from the fact that it provides regularity or data attachment terms which do not favor smooth boundaries (as opposed to the commonly used L^2 norm). See [ROF92], [CE05], [ABCM00].

10.5 Affine invariants in object recognition

The last chapter of this thesis will deal with affine invariants and their use in object recognition. Affine invariants are related to Finsler-Cheeger sets in an unexpected and beautiful way (equation 14.5). Now, let us introduce briefly the need for affine invariants in object recognition.

Object recognition can be succinctly defined as the task of deciding whether two images are photographs of the same object. In an ideal world, object recognition is a trivial task: to decide whether two images represent the same object, compare the pixels of both images, one by one. If they all coincide, then the object is the same and we have a match. Of course, in the real world this will never work. Even two images of the same object taken with the same camera under the same conditions will have different pixels. A large part of the research in object recognition consists in identifying and reversing the distortions that appear between different images of the same object. Since reversing unknown distortions is hard, a clever idea consists in computing quantities from the images which are robust (or even *invariant*) to these distortions. Then, instead of comparing the original pixel values, compare these new, invariant, quantities (called descriptors or features). Let us make a list of the different kinds of image distortions that we may encounter. They are summarized on the following table, where $I : \mathbb{R}^2 \to \mathbb{R}$ denotes an image that is distorted.

image distortion	formula	interesting special cases
contrast change	$I \mapsto g \circ I$ $g : \mathbb{R} \to \mathbb{R}$ increasing	$g(x) = ax + b$ $g(\mathbb{R})$ finite (quantization)
deformation	$I \mapsto I \circ T$ $T : \mathbb{R}^2 \to \mathbb{R}^2$ continuous	T affine
linear operator	$I \mapsto k * I$ $k : \mathbb{R}^2 \to \mathbb{R}$ convolution kernel	k positive (blur)
noise	$I \mapsto I + n$ $n : \mathbb{R}^2 \to \mathbb{R}$ random function	n white gaussian noise
occlusion	$I \mapsto \chi_M J + (1 - \chi_M) I$ $M \subset \mathbb{R}^2,\ J : \mathbb{R}^2 \to \mathbb{R}$ other image	M convex (new object) M concave (new background)

Let us discuss briefly these distortions and how they are tackled by the use of invariants. Occlusion and noise are hard distortions with loss of information. The other distortions on the table are easier to study because they are the result of some small group acting on the set of all possible images.

Occlusion corresponds to a basic operation in image formation, thus it is the most important distortion of all. However, it is very difficult to model occlusion directly, so it is generally ignored as a source of invariants in the context of detection or recognition. The possibility to detect partially occluded objects is recovered later by using local features that describe small parts of objects. Thus, we can say that occlusion invariance is achieved by using only local features.

Noise is an inevitable artifact of image acquisition. It is due mainly to physical effects which are not taken into account, and to the sampling and quantization of pixels. Noise invariance is, by definition, impossible to achieve. The best we can aim for is noise robustness. This is achieved by comparing the feature descriptors using a suitable descriptor metric, instead of comparing them by equality. This requires a threshold on this descriptor metric, or some other criterion to accept to reject matched features.

Image **blur** is the result of many things: out of focus camera, camera movement, aperture diffraction, pixel sampling and interpolation, etc. Although it is in theory an invertible operation, computing the inverse is an ill-posed problem. A typical way to obtain features invariant to blur is to use a multi-scale representation of the images. When the blur kernel has a small support, then at some scale the effect of the blur will disappear.

Contrast changes arise from from illumination changes, white-point correction, gamma correction, etc. Exact invariance to contrast changes is assured by using only morphological operations to construct the features and their descriptors. In practice, full-contrast invariance is not desired (in the end, we see high-contrasted objects better than low-contrasted ones) and morphological operations are often complemented by a weighting of the results by a measure of local contrast, such as the norm of the image gradient.

Global domain deformations come mainly from defects of the lens curvature (differences with respect to the pinhole camera model). These deformations can be corrected by camera calibration and re-interpolation, if needed. More interesting are the **local domain deformations**, due to the position of flat objects

in three-dimensional space. These deformations are, strictly speaking, plane homographies (projective transformations). However, since we are interested only in local features, these transformations are locally approximated by plane affinities. This happens as well for any smooth transformation of the domain; indeed, the definition of a smooth map is precisely that which can be locally approximated by affinities. This is the reason for the interest in affine invariant features and descriptors, the subject of this part of the thesis. Affine invariants were introduced early in image processing, and have been thoroughly used and compared since then [LW88], [BBG96], [MS04].

Once the need for affine invariants has been clarified, the next question is: how to obtain affine invariants from images? This is a very general question, and has been answered by very different techniques, operating at different levels. For example, affine invariance can be achieved by computing affine invariant descriptors on arbitrary points [Low99], or by computing arbitrary descriptors on features selected in an affine invariant way [MCUP04]. These are two possible ways of achieving an invariance. But there are other ways: even starting with non-invariant points and non invariant descriptors, we can still build an affine invariant detector by means of an affine orbit of images [OFL07], [MY09].

Most techniques of computing affine invariant descriptors on an arbitrary point boil down to the same idea: taking histograms of gradient orientations around the given point. The precise way in which these histograms are computed and stored gives a plethora of different methods: such as SIFT descriptors [Low99], SURF [BTVG06], HOG [DT05], GLOG [MS05], LESH [SH08] and spin images [JH02]. On the other hand, finding affine invariant features seems a more elusive task, and only MSER [MCUP04], based on the selection of certain level lines, is commonly used.

11 Finsler total variation and Cheeger sets

The goal of this Chapter is to study the computation of Cheeger sets with respect to anisotropic perimeters and develop some of its applications to image processing, in particular to the problem of edge linking. For that, we first study solution of the anisotropic total variation denoising problem that will permit us to compute anisotropic Cheeger sets.

Let us briefly describe the structure and contents of this Chapter.

On Sections 11.1.1-11.1.6 we lay out the preliminary definitions needed to work with Finsler-Cheeger sets: First we recall the definition of Cheeger sets (11.1.1) and functions of bounded variation (11.1.2) on Euclidean spaces. Then, we transport many of these constructions to the case of Finslerian spaces. Namely, we recall the definition of Finsler total variation (11.1.4), Finsler coarea formula (Equation 11.10), the total variation relaxed functional (11.1.4), the pairing of a Finsler BV-function and a bounded field (Equation 11.18), Finsler traces (Propositions 63 and 64), the subdifferential of Finsler total variation (11.1.6) and finally a Finsler version of Green Formula (Theorem 65).

Once we have all the necessary tools, we proceed to introduce a variatonal problem on functions and its associated PDE (Equations 11.30 and 11.31). Then, we give the proof of existence and unicity of this variational problem (Theorem 68) and state several inequalities satisfied by the solutions (proposition 69). Then we introduce a new variational problem defined on sets, which has Finsler-Cheeger sets as a particular case (Lemma 72). Finally, using the previous inequalities we state the relationship between the variational problem on functions and the variational problem on sets (Proposition 75). As it turns out, Finsler-Cheeger sets are found among the level sets of the solution of the PDE. At the end, we introduce a notion of *local* Finsler-Cheeger sets (11.3).

11.1 Mathematical preliminaries

11.1.1 Definition of Cheeger sets in the Euclidean case

Given an nonempty open bounded subset Ω of $\mathbb{R}^N$, we call Cheeger constant of Ω the quantity

$$\mathcal{C}_\Omega := \min_{F\subseteq\Omega} \frac{P(F)}{|F|}. \tag{11.1}$$

Here $|F|$ denotes de N-dimensional volume of F and $P(F)$ denotes the perimeter of F. The minimum in (11.1) is taken over all nonempty sets of finite perimeter contained in Ω. A Cheeger set of Ω is any set $G \subseteq \Omega$ which minimizes (11.1). Observe that G is a Cheeger set of Ω if and only if $|G| > 0$ and G minimizes

$$\min_{F\subseteq\Omega} P(F) - \mathcal{C}_\Omega |F|. \tag{11.2}$$

Existence of Cheeger sets follows directly from the direct methods of calculus of variations. Uniqueness of Cheeger sets is a more delicate issue and is not true in general (a counterexample is given in [KLR06] when Ω is not convex), though

it has recently been proved that it is generically true [CCN09] (that is, true modulo a small perturbation of the domain Ω). However, uniqueness of Cheeger sets inside convex bodies of $\mathbb{R}^N$ was proved in [CCN07] when the convex body is uniformly convex and of class C^2 and in [AC09] in the general case. The case of convex bodies of $\mathbb{R}^2$ was studied in [ACC05b, KLR06].

11.1.2 BV functions and sets of finite perimeter

Let Ω be an open subset of $\mathbb{R}^N$. A function $u \in L^1(\Omega)$ whose gradient Du in the sense of distributions is a (vector valued) Radon measure with finite total variation in Ω is called a function of bounded variation. The class of such functions will be denoted by $BV(\Omega)$. The total variation of Du on Ω turns out to be

$$\sup\left\{\int_\Omega u\ \mathrm{div} z\ dx : z \in C_0^\infty(\Omega;\mathbb{R}^N), \|z\|_{L^\infty(\Omega;\mathbb{R}^N)} := \operatorname*{ess\,sup}_{x\in\Omega} |z(x)| \le 1\right\}, \tag{11.3}$$

(where for a vector $v = (v_1, \dots, v_N) \in \mathbb{R}^N$ we set $|v|^2 := \sum_{i=1}^N v_i^2$) and will be denoted by $|Du|(\Omega)$ or by $\int_\Omega |Du|$. It turns out that the map $u \to |Du|(\Omega)$ is $L^1_{\mathrm{loc}}(\Omega)$-lower semicontinuous. The total variation of u on a Borel set $B \subseteq \Omega$ is defined as $\inf\{|Du|(A) : A \text{ open }, B \subseteq A \subseteq \Omega\}$. For more results and information on functions of bounded variation we refer to [AFP00, EG92b].

A measurable set $E \subseteq \mathbb{R}^N$ is said to be of finite perimeter in Ω if (11.3) is finite when u is substituted with the characteristic function χ_E of E. The perimeter of E in Ω is defined as $P(E,\Omega) := |D\chi_E|(\Omega)$, and $P(E,\bar\Omega) = P(\mathbb{R}^N \setminus E, \Omega)$. We shall use the notation $P(E) := P(E, \mathbb{R}^N)$. For sets of finite perimeter E one can define the essential boundary $\partial^* E$, which is countably $(N-1)$-rectifiable with finite $\mathcal{H}^{N-1}$ measure, and compute the outer unit normal $\nu^E(x)$ at $\mathcal{H}^{N-1}$ almost all points x of $\partial^* E$, where $\mathcal{H}^{N-1}$ is the $(N-1)$-dimensional Hausdorff measure. Moreover, $|D\chi_E|$ coincides with the restriction of $\mathcal{H}^{N-1}$ to $\partial^* E$ [AFP00, EG92b].

Throughout the text we will use the notation $\{u \ge s\}$ to denote $\{x \in \Omega : u(x) \ge s\}$, $s \in \mathbb{R}$. Also, when we write a.e. without specifying the measure we refer to the Lebesgue measure.

11.1.3 A generalized Green's formula

Let Ω be an open subset of $\mathbb{R}^N$. Following [Anz83], let

$$X_p(\Omega) := \{z \in L^\infty(\Omega;\mathbb{R}^N) : \mathrm{div}\ z \in L^p(\Omega)\}.$$

If $z \in X_p(\Omega)$ and $w \in L^q(\Omega) \cap BV(\Omega)$, $p^{-1} + q^{-1} = 1$, we define the functional $z \cdot Dw : C_0^\infty(\Omega) \to \mathbb{R}$ by the formula

$$\langle z \cdot Dw, \varphi\rangle := -\int_\Omega w\,\varphi\,\mathrm{div}\ z\,dx - \int_\Omega w\, z\cdot\nabla\varphi\,dx. \tag{11.4}$$

Then $z \cdot Dw$ is a Radon measure in Ω, $\int_\Omega z\cdot Dw\,\varphi = \int_\Omega z\cdot\nabla w\varphi\,dx$ for all $\varphi \in C_c^\infty(\Omega)$, $w \in L^q(\Omega)\cap W^{1,1}(\Omega)$, and

$$\left|\int_B z\cdot Dw\right| \le \int_B |z\cdot Dw| \le \|z\|_{L^\infty(\Omega;\mathbb{R}^N)} \int_B |Dw| \qquad \forall B \text{ Borel set } \subseteq \Omega.$$

We recall the following result proved in [Anz83].

Theorem 55. *Let $\Omega \subset \mathbb{R}^N$ be a bounded open set with Lipschitz boundary and $z \in X_p(\Omega)$. Then there exists a function $[z \cdot \nu^\Omega] \in L^\infty(\partial\Omega)$ such that $\|[z \cdot \nu^\Omega]\|_{L^\infty(\partial\Omega)} \leq \|z\|_{L^\infty(\Omega;\mathbb{R}^N)}$, and, for any $u \in BV(\Omega) \cap L^q(\Omega)$ we have*

$$\int_\Omega u \ \mathrm{div} z \ dx + \int_\Omega (z \cdot Du) = \int_{\partial\Omega} [z \cdot \nu^\Omega] u \ d\mathcal{H}^{N-1}.$$

Remark 5. *Let $\Omega_1, \Omega \subset \mathbb{R}^N$ be two bounded Lipschitz open sets with $\Omega_1 \subset\subset \Omega$, $\Omega_2 = \Omega \setminus \overline{\Omega_1}$, and $z_1 \in X_p(\Omega_1)$, $z_2 \in X_p(\Omega_2)$. Assume that*

$$[z_1 \cdot \nu^{\Omega_1}](x) = -[z_2 \cdot \nu^{\Omega_2}](x) \qquad \text{for } \mathcal{H}^{N-1}\text{-a.e } x \in \partial\Omega_1.$$

Then if we define $z := z_1$ on Ω_1 and $z := z_2$ on Ω_2, we have $z \in X_p(\Omega)$.

11.1.4 Finsler total variation

Let us define the general notion of total variation with respect to an arbitrary metric integrand. Following [BBF99] we say that a function $\phi : \Omega \times \mathbb{R}^N \to [0, \infty)$ is a *metric integrand*, or a Finsler metric, if ϕ is a Borel function satisfying the conditions:

$$\text{for a.e. } x \in \Omega, \text{ the map } \xi \in \mathbb{R}^N \to \phi(x, \xi) \text{ is convex,} \tag{11.5}$$

$$\phi(x, t\xi) = |t|\phi(x, \xi) \qquad \forall x \in \Omega, \quad \forall \xi \in \mathbb{R}^N, \quad \forall t \in \mathbb{R}, \tag{11.6}$$

and there exists a constant $\Lambda > 0$ such that

$$0 \leq \phi(x, \xi) \leq \Lambda \|\xi\| \qquad \forall x \in \Omega, \quad \forall \xi \in \mathbb{R}^N. \tag{11.7}$$

We could be more precise and use the term symmetric metric integrand, but for simplicity we use the term metric integrand. Recall that the polar function $\phi^0 : \Omega \times \mathbb{R}^N \to \mathbb{R}$ of ϕ is defined by

$$\phi^0(x, \xi^*) = \sup\{\langle \xi^*, \xi \rangle : \xi \in \mathbb{R}^N, \ \phi(x, \xi) \leq 1\}. \tag{11.8}$$

The function $\phi^0(x, \cdot)$ is convex and lower semicontinuous.

For any $p \in [1, \infty]$, let

$$\mathcal{K}^p_\phi(\Omega) := \{\sigma \in X_p(\Omega) : \phi^0(x, \sigma(x)) \leq 1 \text{ for a.e.} x \in \Omega, \ [\sigma \cdot \nu^\Omega] = 0\}.$$

This is the space of bounded vector fields σ, which vanish on the boundary of Ω, whose divergence is in L^p, and whose length (as measured by ϕ^0) is pointwise bounded by 1.

Definition 56. *Let $u \in L^1(\Omega)$. We define the ϕ-total variation of u in Ω as*

$$\int_\Omega |Du|_\phi := \sup\left\{ \int_\Omega u \ \mathrm{div}\, \sigma \ dx : \sigma \in \mathcal{K}^\infty_\phi(\Omega) \right\}, \tag{11.9}$$

We set $BV_\phi(\Omega) := \{u \in L^1(\Omega) : \int_\Omega |Du|_\phi < \infty\}$ which is a Banach space when endowed with the norm $|u|_{BV_\phi(\Omega)} := \int_\Omega |u| dx + \int_\Omega |Du|_\phi$.

We say that $E \subseteq \mathbb{R}^N$ has finite ϕ-perimeter in Ω if $\chi_E \in BV_\phi(\Omega)$. We set

$$P_\phi(E,\Omega) := \int_\Omega |D\chi_E|_\phi.$$

If $\Omega = \mathbb{R}^N$, we denote $P_\phi(E) := P_\phi(E,\mathbb{R}^N)$. By assumption (11.7), if $E \subseteq \mathbb{R}^N$ has finite perimeter in Ω it has also finite ϕ-perimeter in Ω.

The coarea formula for the ϕ-total variation was proved in [BBF99] (see also [AB94] in a slightly different formulation):

$$\int_\Omega |Du|_\phi = \int_{\mathbb{R}} P_\phi(\{u > s\},\Omega)\, ds \qquad \forall u \in BV_\phi(\Omega). \tag{11.10}$$

Moreover, if T is a Lipschitz function and $u \in BV_\phi(\Omega)$, then $T(u) \in BV_\phi(\Omega)$ [BBF99].

If $u \in BV_\phi(\Omega)$, then u determines a Radon measure in Ω. Indeed, for each open set $U \subseteq \Omega$ we define

$$|Du|_\phi(U) := \sup\left\{\int_U u \operatorname{div}\sigma\ dx : \sigma \in \mathcal{K}_\phi^{\infty,c}(U)\right\}. \tag{11.11}$$

Notice that $|Du|_\phi(U) \le \int_U |Du|_\phi$ for any open set $U \subseteq \Omega$ with Lipschitz boundary. We have that $|Du|_\phi(U)$ is an inner content (see [Hal74] for the definition and the appendix of [CFM09] for the proof). Let

$$\mu^*(E) := \inf\{|Du|_\phi(U) : U \text{ is an open set in } \Omega,\ E \subseteq U\}$$

be the outer measure induced by $|Du|_\phi$. Then for any Borel set F we define $\mu(F) = \mu^*(F)$. Then μ is a regular Borel measure [Hal74], p. 235. We shall write $|Du|_\phi(E)$ instead of $\mu(E)$.

Definition 57. *Let $\phi : \Omega \times \mathbb{R}^N \to \mathbb{R}$ be a metric integrand, $B \subseteq \Omega$. We say that ϕ is coercive in B if there exist $\beta \ge \alpha > 0$ such that*

$$\alpha\|\xi\| \le \phi(x,\xi) \le \beta\|\xi\| \qquad \forall x \in B, \quad \forall \xi \in \mathbb{R}^N. \tag{11.12}$$

We say that ϕ is continuous in B if ϕ restricted to $B \times \mathbb{R}^N$ is a continuous function of (x,ξ). If $B = \Omega$ we just say that ϕ is coercive (resp. continuous). We say that ϕ is coercive (resp. continuous) near $\partial\Omega$ if there exists $\Omega_1 \subset\subset \Omega$ an open bounded set with Lipschitz boundary such that ϕ is coercive (resp. continuous) in a neighborhood of $\Omega \setminus \Omega_1$.

Example. An interesting case occurs when $g : \Omega \to [0,\infty)$ is a bounded Borel function. Let $\phi(x,\xi) = g(x)|\xi|$. Then [AB94]

$$\phi^0(x,\xi^*) := \begin{cases} 0 & \text{if} \quad g(x) = 0, \xi^* = 0 \\ +\infty & \text{if} \quad g(x) = 0, \xi^* \neq 0 \\ \frac{|\xi^*|}{g(x)} & \text{if} \quad \xi^* \in \mathbb{R}^N, g(x) > 0. \end{cases} \tag{11.13}$$

If $\sigma \in X_\infty(\Omega)$ and $\phi^0(x,\sigma(x)) \le 1$, then we may write $\sigma(x) = g(x)z(x)$ where $z \in L^\infty(\Omega;\mathbb{R}^N)$ is such that $|z(x)| \le 1$ for a.e. $x \in \Omega$, and

$$\int_\Omega g|Du| := \sup\left\{ \int_\Omega u \operatorname{div}(gz)\, dx : gz \in X_\infty(\Omega),\ \ |z(x)| \le 1 \ \text{ for a.e. } x \in \Omega \right\}.$$

Finsler TV relaxed functional. From the definition, it follows that $u \in L^1(\Omega) \to \int_\Omega |Du|_\phi$ and $E \to P_\phi(E,\Omega)$ are lower-semicontinuous with respect to the L^1 convergence. The following result was proved in [BBF99] when $\Omega = \mathbb{R}^N$. The proof adapts easily.

Proposition 58. *Assume that $\phi : \Omega \times \mathbb{R}^N \to [0,\infty)$ is a metric integrand. Let*

$$J(u) := \begin{cases} \displaystyle\int_\Omega \phi(x,\nabla u)\, dx & \text{if} \qquad u \in W^{1,1}(\Omega) \\ \\ +\infty & \text{if} \quad u \in L^1(\Omega) \setminus W^{1,1}(\Omega). \end{cases}$$

Let $\overline{J}$ be the relaxed functional, that is,

$$\overline{J}(u) := \inf\{\liminf_n J(u_n) : u_n \to u \ \text{ in } L^1(\Omega),\ u_n \in W^{1,1}(\Omega)\}.$$

Then for every $u \in BV_\phi(\Omega)$, we have $\overline{J}(u) = \int_\Omega |Du|_\phi$. Hence, for any $u \in BV_\phi(\Omega)$, there exists a sequence $u_n \in W^{1,1}(\Omega)$ such that $\int_\Omega \phi(x,\nabla u_n) \to \int_\Omega |Du|_\phi$. In particular, $BV_\phi(\Omega)$ is the finiteness domain of $\overline{J}$.

The following result was proved on [CFM09]:

Theorem 59. *Let Ω, Q be open bounded sets in $\mathbb{R}^N$ with Lipschitz boundary such that $\Omega \subset\subset Q$. Let $\phi : \Omega \times \mathbb{R}^N \to \mathbb{R}$ be a metric integrand which admits an extension to $Q \times \mathbb{R}^N$ such that ϕ is continuous and coercive in a neighborhood of $Q \setminus \Omega$. Let $u \in BV_\phi(\Omega)$, $\varphi \in L^1(\partial\Omega)$. Let $\varphi \in L^1(\partial\Omega)$,*

$$J_{\phi,\varphi}(u) := \begin{cases} \displaystyle\int_\Omega \phi(x,\nabla u) & \text{if } u \in W^{1,1}(\Omega) \text{ and } u = \varphi \text{ on } \partial\Omega \\ \\ +\infty & \text{otherwise.} \end{cases} \tag{11.14}$$

and

$$\mathcal{J}_{\phi,\varphi}(u) := \begin{cases} \displaystyle\int_\Omega |Du|_\phi + \int_{\partial\Omega} \phi(x,\nu^\Omega)|u-\varphi|\, d\mathcal{H}^{N-1} & \text{if } u \in BV_\phi(\Omega) \\ \\ +\infty & \text{otherwise.} \end{cases} \tag{11.15}$$

Then the functional $\mathcal{J}_{\phi,\varphi}(u)$ equals the relaxed functional of $J_{\phi,\varphi}(u)$.

The following technical lemma was also proved in [CFM09]. It allows to extend functions of ϕ-bounded variation out of the boundary of their domain.

Lemma 60. *Assume that $\phi : \Omega\times\mathbb{R}^N \to \mathbb{R}$ is a metric integrand which admits an extension as a metric integrand to an open bounded set Q with Lispchitz boundary such that $\Omega \subset\subset Q$. Assume that the extension is continuous and coercive in a neighborhood of $Q \setminus \Omega$. Let $u \in BV_\phi(\Omega)$, $\tilde\varphi \in W^{1,1}(Q \setminus \Omega)$ be such that $\tilde\varphi|_{\partial\Omega} = \varphi \in L^1(\partial\Omega)$. Let*

$$\tilde u(x) := \begin{cases} u(x) & \text{if} \quad x \in \Omega \\ \\ \tilde\varphi & \text{if} \quad x \in Q \setminus \Omega. \end{cases} \tag{11.16}$$

Then $u \in BV_\phi(Q)$ and

$$\int_Q |D\tilde u|_\phi = \int_\Omega |Du|_\phi + \int_{Q\setminus\overline{\Omega}} |\nabla\tilde\varphi|_\phi + \int_{\partial\Omega} \phi(x,\nu^\Omega)|u-\varphi|\, d\mathcal{H}^{N-1}. \tag{11.17}$$

11.1.5 An extension of Green formulas

In all this Section we assume that Ω is a bounded open set with Lipschitz boundary. We assume also that $\phi : \Omega \times \mathbb{R}^N \to \mathbb{R}$ is a metric integrand. We just give an overview of the main results and refer to the appendixes of [CFM09] for the technical proofs.

The Measure $z\cdot Du$. Recall the definition of $z\cdot Du$ in Section 11.1.3, for $q = 1$.

Let $u \in BV_\phi(\Omega) \cap L^p(\Omega)$ and $z \in X_q(\Omega)$, $p, q \in [1,\infty]$, $p^{-1} + q^{-1} = 1$. We define the functional $z \cdot Du : \mathcal{D}(\Omega) \to \mathbb{R}$ as

$$\langle z\cdot Dw, \varphi\rangle := -\int_\Omega w\,\varphi \,\mathrm{div}\; z\, dx - \int_\Omega w\, z\cdot\nabla\varphi\, dx. \tag{11.18}$$

Although this Section could be developed in this general functional setting, we shall restrict to the case $p = 1$. we shall restrict to this case.

Let us write

$$\mathcal{A}_\infty(\Omega) := \{z \in X_\infty(\Omega) : \|\phi^0(x, z(x))\|_{L^\infty(\Omega)} < \infty\}. \tag{11.19}$$

To develop the theory we shall assume from now on that $z \in \mathcal{A}_\infty(\Omega)$.

Proposition 61. *For any open set $U \subset \Omega$ and for any function $\varphi \in \mathcal{D}(U)$, one has*

$$|\langle z\cdot Du, \varphi\rangle| \le \|\varphi\|_\infty \|\phi^0(x,z)\|_{L^\infty(U)} |Du|_\phi(U). \tag{11.20}$$

In particular, $z\cdot Du$ is a Radon measure in Ω.

Lemma 62. *Let $u \in BV_\phi(\Omega)$, $\sigma \in \mathcal{A}_\infty(\Omega)$ with $\|\phi^0(x, z(x))\|_{L^\infty(\Omega)} \le 1$. Assume that $\int_\Omega \sigma\cdot Du = \int_\Omega |Du|_\phi$. Then for any $b \in \mathbb{R}$ we have*

$$\int_\Omega \sigma\cdot D(u-b)^+ = \int_\Omega |D(u-b)^+|_\phi \quad \text{and} \quad \int_\Omega \sigma\cdot D(u\wedge b) = \int_\Omega |D(u\wedge b)|_\phi$$

where $(u-b)^+ = \max(u-b, 0)$ and $u\wedge b = \inf(u, b)$.

Proof. By the observation following (11.10) we know that $(u-b)^+, u\wedge b \in BV_\phi(\Omega)$. Then we have

$$\begin{aligned}\int_\Omega |Du|_\phi &= \int_\Omega \sigma\cdot Du = \int_\Omega \sigma\cdot D(u-b)^+ + \int_\Omega \sigma\cdot D(u\wedge b)\\ &\le \int_\Omega |D(u-b)^+|_\phi + \int_\Omega |D(u\wedge b)|_\phi = \int_\Omega |Du|_\phi.\end{aligned}$$

where the inequality follows from Proposition 61 and the last equality follows from the coarea formula (11.10). The Lemma follows. □

Traces. The following result can be proved as in [Anz83] (see also [AVCM04]).

Proposition 63. *Assume that ϕ is continuous and coercive at $\partial\Omega$. There exists a bilinear map $\langle z,u\rangle_{\partial\Omega} : \mathcal{A}_\infty(\Omega)\times BV_\phi(\Omega)\to\mathbb{R}$ such that*

$$\langle z,u\rangle_{\partial\Omega} = \int_{\partial\Omega} u(x)z(x)\cdot\nu(x)\, d\mathcal{H}^{N-1} \qquad \text{if } \ z\in C^1(\Omega,\mathbb{R}^N)\cap C(\overline{\Omega},\mathbb{R}^N) \tag{11.21}$$

$$|\langle z,u\rangle_{\partial\Omega}| \le \|z\|_{L^\infty(\Omega;\mathbb{R}^N)} \int_{\partial\Omega} |u(x)|\, d\mathcal{H}^{N-1} \qquad \text{forall } \ z,u. \tag{11.22}$$

Proposition 64. *Assume that ϕ is continuous and coercive at $\partial\Omega$. Then there exists a linear operator $\gamma : \mathcal{A}_\infty(\Omega)\to L^\infty(\partial\Omega)$ such that*

$$\|\gamma(z)\|_{L^\infty(\partial\Omega)} \le \|z\|_{L^\infty(\Omega;\mathbb{R}^N)} \tag{11.23}$$

$$\langle z,u\rangle_{\partial\Omega} = \int_{\partial\Omega} \gamma(z)(x)u(x)\, d\mathcal{H}^{N-1} \qquad \text{forall } \ u\in BV_\phi(\Omega), \tag{11.24}$$

$$\gamma(z)(x) = z(x)\cdot\nu(x) \qquad \text{forall } \ x\in\partial\Omega \ \text{ if } \ z\in C^1(\overline{\Omega},\mathbb{R}^N). \tag{11.25}$$

The function $\gamma(z)$ is a weakly defined trace on $\partial\Omega$ of the normal component of z. We shall denote $\gamma(z)$ by $[z\cdot\nu]$.

Proof. Fix $z\in\mathcal{A}_\infty(\Omega)$, $u\in BV_\phi(\Omega)$. Consider the functional $F : L^\infty(\partial\Omega)\to\mathbb{R}$ defined by $F(u) := \langle z,w\rangle_{\partial\Omega}$, where $w\in BV_\phi(\Omega)$ is such that $w|_{\partial\Omega} = u|_{\partial\Omega}$. By estimate (11.22), we have $|F(u)| \le \|z\|_{L^\infty(\Omega;\mathbb{R}^N)} \int_{\partial\Omega}|u|$. Hence there exists a function $\gamma(z)\in L^\infty(\partial\Omega)$ such that

$$F(u) = \int_{\partial\Omega} \gamma(z)(x)u(x)\, d\mathcal{H}^{N-1}$$

and the result follows. □

Green's formula. We give now the expected anisotropic *Green's formula* relating the function $[z\cdot\nu]$ and the measure $z\cdot Du$. Its proof was given in [CFM09], and is an immediate consequence of having introduced the appropriate definitions.

Theorem 65. *Assume that ϕ is continuous and coercive at $\partial\Omega$. Let $z\in\mathcal{A}_\infty(\Omega)$, $u\in BV_\phi(\Omega)$. Then*

$$\int_\Omega u\,\mathrm{div}(z)\, dx + \int_\Omega z\cdot Du = \int_{\partial\Omega} [z\cdot\nu]u\, d\mathcal{H}^{N-1}. \tag{11.26}$$

11.1.6 Subdifferential of Finsler total variation

In this Subsection *we assume that* $\phi : \Omega \times \mathbb{R}^N \to [0,\infty)$ *is a continuous and coercive metric integrand in* Ω. Notice that in this case $BV_\phi(\Omega) = BV(\Omega)$. Let us define the functional

$$\psi_\phi(u) := \begin{cases} \displaystyle\int_\Omega \phi(x,\nabla u) & \text{if } u \in L^2(\Omega)\cap W^{1,1}(\Omega) \text{ and } u = 0 \text{ on } \partial\Omega \\ +\infty & \text{otherwise.} \end{cases} \tag{11.27}$$

According to [Mol05, AB94], the functional $\Psi_\phi : L^2(\Omega) \to (-\infty,+\infty]$ defined by

$$\Psi_\phi(u) := \begin{cases} \displaystyle\int_\Omega |Du|_\phi + \int_{\partial\Omega} \phi(x,\nu^\Omega)|u|\, d\mathcal{H}^{N-1} & \text{if } u \in L^2(\Omega)\cap BV(\Omega) \\ +\infty & \text{otherwise,} \end{cases} \tag{11.28}$$

is the lower-semicontinuous relaxation of ψ_ϕ. Moreover $\Psi_\phi(u)$ is lower-semicontinuous with respect to convergence in $L^1(\Omega)$ [Mol05]. Since Ψ_ϕ is convex and lower semicontinuous in $L^2(\Omega)$, we have that $\partial\Psi_\phi$ is a maximal monotone operator in $L^2(\Omega)$ (see [Bré73]). Next lemma gives the characterization of $\partial\Psi_\phi$ (see [Mol05] for a proof).

Theorem 66. *The following conditions are equivalent*

(i) $v \in \partial\Psi_\phi(u)$.

(ii) $u, v \in L^2(\Omega)$, $u \in BV(\Omega)$ *and there exists* $\sigma \in X_2(\Omega)$ *with* $\phi^0(x,\sigma(x)) \le 1$ *a.e. in* Ω, $v = -\mathrm{div}(\sigma)$ *in* $\mathcal{D}'(\Omega)$ *such that* $\sigma(x) \in \partial_\xi\phi(x,\nabla u(x))$ *a.e. in* Ω, $\sigma\cdot Du = |Du|_\phi$ *and* $[\sigma\cdot\nu^\Omega] \in \mathrm{sign}(-u)\phi(x,\nu^\Omega(x))$ $\mathcal{H}^{N-1}$ *a.e. on* $\partial\Omega$.

(iii) $u, v \in L^2(\Omega)$, $u \in BV(\Omega)$ *and there exists* $\sigma \in X_2(\Omega)$ *with* $\phi^0(x,\sigma(x)) \le 1$ *a.e. in* Ω, $v = -\mathrm{div}(\sigma)$ *in* $\mathcal{D}'(\Omega)$ *such that*

$$\int_\Omega (w-u)v \le \int_\Omega \sigma\cdot Dw - \int_\Omega |Du|_\phi - \int_{\partial\Omega}[\sigma\cdot\nu^\Omega]w - \int_{\partial\Omega}\phi(x,\nu^\Omega(x))|u|,$$

$\forall w \in BV(\Omega)\cap L^2(\Omega)$.

(iv) $u, v \in L^2(\Omega)$, $u \in BV(\Omega)$ *and there exists* $\sigma \in X_2(\Omega)$ *with* $\phi^0(x,\sigma(x)) \le 1$ *a.e. in* Ω, $v = -\mathrm{div}(\sigma)$ *in* $\mathcal{D}'(\Omega)$ *such that*

$$\int_\Omega (w-u)v \le \int_\Omega \sigma\cdot Dw - \int_\Omega |Du|_\phi + \int_{\partial\Omega}\phi(x,\nu^\Omega(x))|w| - \int_{\partial\Omega}\phi(x,\nu^\Omega(x))|u|,$$

$\forall w \in BV(\Omega)\cap L^2(\Omega)$.

When we used in the previous statement the expression a.e. in Ω we mean a.e. with respect to the Lebesgue measure in Ω. The identity $\sigma\cdot Du = |Du|_\phi$ means that both Radon measures coincide.

From now on we shall write $v = -\mathrm{div}\,\partial_\xi\phi(x,\nabla u)$ instead of $v \in \partial\Psi_\phi(u)$.

11.2 A PDE that produces Finsler-Cheeger sets

The maximal ϕ-Cheeger set inside Ω Let Ω, Q be open bounded sets in $\mathbb{R}^N$ with Lipschitz boundary such that $\Omega \subset\subset Q$. Let $\phi : \Omega \times \mathbb{R}^N \to \mathbb{R}$ be a metric integrand with an extension to $Q \times \mathbb{R}^N$ such that ϕ *is continuous and coercive in a neighborhood of* $Q \setminus \Omega$. *For the rest of the chapter we assume that this property holds.* Let $h \in L^\infty(\Omega)$, $h(x) > 0$ a.e. in Ω, such that

$$\int_\Omega \frac{1}{h(x)}\, dx < \infty. \tag{11.29}$$

We denote by $L^2(\Omega, hdx)$ the set of measurable functions $u : \Omega \to \mathbb{R}$ such that $\int_\Omega u^2\, h\, dx < \infty$. From (11.29) we have that $L^2(\Omega) \subseteq L^2(\Omega, hdx)$. For $f \in L^2(\Omega, hdx)$, $\lambda > 0$, let us consider the energy functional

$$\mathcal{E}_{\phi,h,\lambda}(u) := \int_\Omega |Du|_\phi + \frac{\lambda}{2}\int_\Omega (u-f)^2 h\, dx + \int_{\partial\Omega} \phi(x, \nu^\Omega)|u|\, d\mathcal{H}^{N-1}. \tag{11.30}$$

Although for ϕ-Cheeger sets we need only the case $f = 1$, the general case where $f \neq 1$ is of interest in Section 13.3, where we discuss the application to anisotropic diffusion.

Let us consider the partial differential equation formally related to (11.30):

$$hu - \lambda^{-1}\mathrm{div}\,(\partial_\xi \phi(x, Du)) = hf \tag{11.31}$$

with Dirichlet boundary conditions. Notice that this is a symbolic notation. There is also a slight abuse of notation in writing (11.31) as an equality. Since the subdifferential of the ϕ-total variation is multivalued, (11.31) should be better written as $hf \in hu - \lambda^{-1}\mathrm{div}\,(\partial_\xi \phi(x, \nabla u))$. In spite of this we will write the equation as (11.31), understanding that the equality holds for an element of the subdifferential.

Definition 67 (Solution of the PDE)**.** *Let $f \in L^\infty(\Omega)$. We say that $u \in L^2(\Omega, h\,dx)$ is a solution of (11.31) if $u \in BV_\phi(\Omega) \cap L^\infty(\Omega)$, and there is a vector field $\sigma \in \mathcal{A}_\infty(\Omega)$ such that*

(i) $hu - \lambda^{-1}\mathrm{div}\,(\sigma) = hf$ *in* $\mathcal{D}'(\Omega)$,

(ii) $\phi^0(x, \sigma(x)) \leq 1$ *a.e.,*

(iii) $\displaystyle\int_\Omega \sigma \cdot Du = \int_\Omega |Du|_\phi$,

(iv) $[\sigma \cdot \nu^\Omega] \in \mathrm{sign}(-u)\phi(x, \nu^\Omega(x))$, $\mathcal{H}^{N-1}$*-a.e.* $x \in \partial\Omega$.

We could have given a more general definition but the present case is sufficient for our purposes.

Theorem 68 (Existence and uniqueness for the variational problem)**.** (*i*) *Let $f \in L^2(\Omega, h dx)$. Then there is a unique solution of the problem*

$$(Q)_\lambda : \quad \min_{u \in BV_\phi(\Omega) \cap L^2(\Omega, h\,dx)} \quad \mathcal{E}_{\phi,h,\lambda}(u). \tag{11.32}$$

(*ii*) *Assume that $f \in L^\infty(\Omega)$. Then there is a unique solution $u \in L^2(\Omega, h\,dx)$ of (11.31). Moreover the solution $u \in L^\infty(\Omega)$ and it minimizes (11.32).*

Proof. (*i*) Let u_n be a minimizing sequence for (11.32). Then u_n is bounded in $L^2(\Omega, h\,dx)$. Assume that $u_n \to u$ weakly in $L^2(\Omega, h\,dx)$. Observe that $u \in L^1(\Omega)$ since

$$\int_\Omega |u|\,dx \le \left(\int_\Omega u^2\, h\,dx\right)^{1/2} \left(\int_\Omega \frac{1}{h}\,dx\right)^{1/2}, \tag{11.33}$$

and $u_n \to u$ weakly in $L^1(\Omega)$. Indeed if $\varphi \in L^\infty(\Omega)$, then $\frac{\varphi}{h} \in L^2(\Omega, h\,dx)$ and

$$\int_\Omega (u_n - u)\varphi\,dx = \int_\Omega (u_n - u)\frac{\varphi}{h}\,h\,dx \to 0.$$

For any given function $v \in L^1(\Omega)$, let $\tilde v$ denote its extension by 0 in $\mathbb{R}^N$. Since $v \to \int_\Omega |D\tilde v|_\phi$ is convex and lower semicontinuous with respect to the convergence in $L^1(\Omega)$, then it is also lower-semicontinuous with respect to the weak convergence in $L^1(\Omega)$. Hence by Lemma 60 we have

$$\begin{aligned}\int_\Omega |Du|_\phi + \int_{\partial\Omega} \phi(x,\nu^\Omega)|u| &= \int_{\mathbb{R}^N} |D\tilde u|_\phi \le \liminf_n \int_{\mathbb{R}^N} |D\tilde u_n|_\phi \\ &= \liminf_n \int_\Omega |Du_n|_\phi + \int_{\partial\Omega} \phi(x,\nu^\Omega)|u_n|.\end{aligned}$$

Then also

$$\mathcal{E}_{\phi,h,\lambda}(u) \le \liminf_n \mathcal{E}_{\phi,h,\lambda}(u_n),$$

and $u \in BV_\phi(\Omega) \cap L^2(\Omega, h\,dx)$ is a minimum of $\mathcal{E}_{\phi,h,\lambda}(u)$. Since the functional is strictly convex, the solution is unique.

(*ii*) We divide the proof in three steps.

Step 1. Existence and uniqueness of solutions of an approximating problem. Let $h_n = h + \frac{1}{n}$, $\phi_n(x,\xi) = \phi(x,\xi) + \frac{1}{n}\Xi(\xi)$ where $\Xi(\xi) = |\xi|$, $x \in \Omega$, $\xi \in \mathbb{R}^N$. As in (*i*) there is a unique minimizer u_n of $\mathcal{E}_{\phi_n,h_n,\lambda}(u)$ which is in $BV_{\phi_n}(\Omega) \cap L^2(\Omega, h_n\,dx)$. Since ϕ_n is coercive and $h_n \ge \frac{1}{n}$, then $u_n \in BV(\Omega) \cap L^2(\Omega)$. As a consequence we have that

$$\lambda(f - u_n)h_n \in \partial\Psi_{\phi_n}(u_n), \tag{11.34}$$

the subdifferential $\partial\Psi_{\phi_n}(u_n)$ being taken in $L^2(\Omega)$. Now, since ϕ_n is continuous and coercive, by the characterization of the subdifferential $\partial\Psi_{\phi_n}(u_n)$ given in Theorem 66, u_n is satisfies the equation

$$h_n u_n - \lambda^{-1}\mathrm{div}\,(\partial\phi_n(x, \nabla u_n)) = h_n f \tag{11.35}$$

in the sense of Definition 67. That is, there exists $z_n \in \partial\phi_n(x, \nabla u_n)$, $z_n \in X_2(\Omega)$, such that

$$h_n u_n - \lambda^{-1}\mathrm{div}\, z_n = h_n f \qquad \text{in } \mathcal{D}'(\Omega), \tag{11.36}$$

$$\int_\Omega z_n \cdot Du_n = \int_\Omega |Du_n|_{\phi_n}, \tag{11.37}$$

$$[z_n \cdot \nu^\Omega] \in \mathrm{sign}(-u_n)\phi_n(x, \nu^\Omega(x)), \qquad \mathcal{H}^{N-1}\text{-a.e. } x \in \partial\Omega. \tag{11.38}$$

Conversely, using again Theorem 66, if $u_n \in L^2(\Omega)$ is a solution of (11.35), then it is also a solution (11.34). Now, since Ψ_{ϕ_n} is a maximal monotone operator in $L^2(\Omega)$, the uniqueness of solutions of (11.35) follows immediately by standard results [Bré73].

Since $\partial\phi_n = \partial\phi + \frac{1}{n}\partial\Xi$, then we may write $z_n = \sigma_n + \frac{1}{n}\eta_n$ where $\sigma_n \in \partial\phi(x, \nabla u_n)$ and $\eta_n \in \partial\Xi(\nabla u_n)$.

Step 2. Basic estimates and passage to the limit. Assume that $a \le f \le b$. Let us prove that $a \le u_n \le b$, $a, b \in \mathbb{R}$. First we observe that multiplying (11.35) by u_n and integrating by parts, we obtain

$$\int_\Omega u_n^2\, h_n\, dx + \lambda^{-1}\int_\Omega |Du_n|_{\phi_n} + \lambda^{-1}\int_{\partial\Omega} \phi_n(x, \nu^\Omega)|u_n| = \int_\Omega f\, h_n\, u_n\, dx, \tag{11.39}$$

which implies that u_n is uniformly bounded in $BV_\phi(\Omega)$. To prove that $u_n \le b$, we multiply $h_n(u_n - b) - \lambda^{-1}\mathrm{div}\, z_n = h_n(f - b)$ by $(u_n - b)^+$ and integrating by parts we obtain

$$\begin{aligned}\int_\Omega ((u_n - b)^+)^2\, h_n\, dx \quad &+ \quad \lambda^{-1}\int_\Omega z_n \cdot D(u_n - b)^+ + \lambda^{-1}\int_{\partial\Omega} \phi_n(x, \nu^\Omega)(u_n - b)^+\, d\mathcal{H}^{N-1} \\ &= \quad \int_\Omega (f - b)(u_n - b)^+ h_n\, dx \le 0.\end{aligned}$$

Using Lemma 62 we have that $\int_\Omega z_n \cdot D(u_n - b)^+ \ge 0$. Since the third term on the left hand side is also ≥ 0, we have that $(u_n - b)^+ = 0$, i.e. $u_n \le b$ a.e.. In a similar way we prove that $u_n \ge a$ a.e.. Modulo a subsequence, we may assume that $u_n \to u$ weakly in $L^2(\Omega)$ for some $u \in BV_\phi(\Omega) \cap L^\infty(\Omega)$. Since $h_n \to h$ uniformly in Ω also

$$u_n h_n \to uh \quad \text{ weakly in } L^2(\Omega) \text{ as } n \to \infty. \tag{11.40}$$

Finally, since z_n, η_n are bounded in $L^\infty(\Omega)$, by extracting a further subsequence, we may assume that $z_n, \sigma_n \to \sigma$ weakly* in $L^\infty(\Omega)$. Now, since $\phi^0(x, \sigma_n(x)) \le 1$ a.e., and this condition is stable under weak* convergence in $L^\infty(\Omega)$, we have $\phi^0(x, \sigma(x)) \le 1$ a.e.. Now, since $\mathrm{div}\, z_n$ is bounded in $L^\infty(\Omega)$ we have that, by Banach-Alaouglu's Theorem,

$$\mathrm{div}\, z_n \to \mathrm{div}\, \sigma \qquad \text{weakly in } L^2(\Omega),$$

and $\sigma \in \mathcal{A}_\infty(\Omega)$. Letting $n \to \infty$ in (11.36) we obtain that

$$hu - \lambda^{-1}\mathrm{div}\, \sigma = hf \qquad \text{in } \mathcal{D}'(\Omega). \tag{11.41}$$

Step 3. Final step. Let us prove that

$$\int_\Omega \sigma \cdot Du = \int_\Omega |Du|_\phi, \tag{11.42}$$

$$[\sigma \cdot \nu^\Omega] \in \operatorname{sign}(-u)\phi(x, \nu^\Omega) \quad \text{a.e. on } \partial\Omega. \tag{11.43}$$

Let $\varphi \in C^1(\Omega) \cap C(\overline{\Omega})$. Since $|[z_n \cdot \nu^\Omega]| \le \phi_n(x, \nu^\Omega(x))$ a.e. in $\partial\Omega$, we may assume that $[z_n \cdot \nu^\Omega] \to \beta(x)$ weakly* in $L^\infty(\partial\Omega)$, and letting $n \to \infty$ in

$$\int_\Omega z_n \cdot \nabla\varphi \, dx + \int_{\partial\Omega} [z_n \cdot \nu^\Omega]\varphi \, d\mathcal{H}^{N-1} = -\int_\Omega \operatorname{div} z_n \, \varphi \, dx,$$

we obtain

$$\int_\Omega \sigma\cdot\nabla\varphi \, dx + \int_{\partial\Omega} \beta\varphi \, d\mathcal{H}^{N-1} = -\int_\Omega \operatorname{div}\sigma \, \varphi \, dx = \int_\Omega \sigma\cdot\nabla\varphi \, dx + \int_{\partial\Omega} [\sigma\cdot\nu^\Omega]\varphi \, d\mathcal{H}^{N-1}.$$

Hence

$$\int_{\partial\Omega} \beta\varphi \, d\mathcal{H}^{N-1} = \int_{\partial\Omega} [\sigma \cdot \nu^\Omega]\varphi \, d\mathcal{H}^{N-1}$$

holds for any $\varphi \in C^1(\Omega) \cap C(\overline{\Omega})$ and we obtain that $\beta = [\sigma \cdot \nu^\Omega]$. In particular

$$|[\sigma \cdot \nu^\Omega]| \le \phi(x, \nu^\Omega(x)) \quad \text{a.e. in } \partial\Omega. \tag{11.44}$$

Now, using (11.40) in the fifth line of the following computations, and using Corollary A.1 of Appendix 2 of [CFM09] (which says that the measure $z \cdot Du$ is absolutely continuous with respect to the measure $|Du|_\phi$) together with $\|\phi^0(x, \sigma(x))\|_\infty \le 1$ and (11.44) in the last inequality below, we have

$$\begin{aligned}
&\int_\Omega u^2 \, h \, dx + \lambda^{-1} \int_\Omega |Du|_\phi + \lambda^{-1} \int_{\partial\Omega} \phi(x, \nu^\Omega)|u| \, d\mathcal{H}^{N-1} \\
\le\ & \liminf_n \int_\Omega u_n^2 \, h_n \, dx + \liminf_n \left\{ \lambda^{-1} \int_\Omega |Du_n|_\phi + \lambda^{-1} \int_{\partial\Omega} \phi(x, \nu^\Omega)|u_n| \, d\mathcal{H}^{N-1} \right\} \\
\le\ & \liminf_n \int_\Omega u_n^2 \, h_n \, dx + \liminf_n \left\{ \lambda^{-1} \int_\Omega |Du_n|_{\phi_n} + \lambda^{-1} \int_{\partial\Omega} \phi_n(x, \nu^\Omega)|u_n| \, d\mathcal{H}^{N-1} \right\} \\
\le\ & \liminf_n \left\{ \int_\Omega u_n^2 \, h_n \, dx + \lambda^{-1} \int_\Omega |Du_n|_{\phi_n} + \lambda^{-1} \int_{\partial\Omega} \phi_n(x, \nu^\Omega)|u_n| \, d\mathcal{H}^{N-1} \right\} \\
=\ & \liminf_n \int_\Omega f u_n \, h_n \, dx = \int_\Omega f u \, h \, dx = \int_\Omega u^2 \, h \, dx - \lambda^{-1} \int_\Omega u \operatorname{div} \sigma \, dx \\
=\ & \int_\Omega u^2 \, h \, dx + \lambda^{-1} \int_\Omega \sigma \cdot Du - \lambda^{-1} \int_{\partial\Omega} [\sigma \cdot \nu^\Omega] u \, d\mathcal{H}^{N-1} \\
\le\ & \int_\Omega u^2 \, h \, dx + \lambda^{-1} \int_\Omega |Du|_\phi + \lambda^{-1} \int_{\partial\Omega} \phi(x, \nu^\Omega)|u| \, d\mathcal{H}^{N-1}.
\end{aligned}$$

In particular, we obtain (11.42) and

$$-\int_{\partial\Omega} [\sigma \cdot \nu^\Omega] u \, d\mathcal{H}^{N-1} = \int_{\partial\Omega} \phi(x, \nu^\Omega)|u| \, d\mathcal{H}^{N-1}.$$

The last identity implies that $-[\sigma \cdot \nu^\Omega]u = \phi(x, \nu^\Omega)|u|$, hence (11.43) follows. Thus, conditions $(i) - (ii) - (iii) - (iv)$ of Definition 67 are satisfied and u is a solution (11.31).

The proof of uniqueness follows in a standard way (see [AVCM04], Chapter 2). Finally, since the energy is convex, the solution of (11.31) is a minimizer of (11.32). □

Proposition 69 (Inequalities satisfied by solutions of the PDE)**.** *Let $u \in BV_\phi(\Omega) \cap L^2(\Omega, h\,dx)$ be the solution of the variational problem (11.32) with $f = 1$. Then $0 \le u \le 1$. Let $E_s := \{u \ge s\}$, $s \in (0, 1]$. Then for any $s \in (0, 1]$ we have*

$$P_\phi(E_s) - \lambda(1-s)|E_s|_h \le P_\phi(F) - \lambda(1-s)|F|_h \tag{11.45}$$

for any $F \subseteq \Omega$.

We denote $|F|_h = \int_F h(x)\,dx$ for any measurable subset $F \subseteq \Omega$.

Proof. Recall that u satisfies the following partial differential equation

$$hu - \lambda^{-1}\mathrm{div}\,\sigma = h \quad \text{in } \Omega, \tag{11.46}$$

where $\sigma(x) \in \partial_\xi \phi(x, \nabla u(x))$ a.e.. As in Step 2 of the proof of Theorem 68.(ii) we deduce that $0 \le u \le 1$.

Let us prove that for almost any $s \in (0, \infty)$ we have

$$P_\phi(E_s) = \int_\Omega \sigma \cdot D\chi_{E_s} + \int_{\partial\Omega} \phi(x, \nu^\Omega(x))\chi_{E_s}(x)\,d\mathcal{H}^{N-1}.$$

Indeed, multiplying (11.46) by u and integrating by parts we obtain

$$\int_\Omega \sigma \cdot Du + \int_{\partial\Omega} \phi(x, \nu^\Omega)|u|\,d\mathcal{H}^{N-1} = \lambda \int_\Omega (1-u)uh\,dx. \tag{11.47}$$

Now, multiplying (11.46) by χ_{E_s} and integrating by parts we obtain

$$\int_\Omega \sigma \cdot D\chi_{E_s} - \int_{\partial\Omega} [\sigma \cdot \nu^\Omega]\chi_{E_s}\,d\mathcal{H}^{N-1} = \lambda \int_\Omega (1-u)h\chi_{E_s}\,dx. \tag{11.48}$$

Notice that this relation proves that $\int_\Omega \sigma \cdot D\chi_{E_s} - \int_{\partial\Omega}[\sigma \cdot \nu^\Omega]\chi_{E_s}\,d\mathcal{H}^{N-1}$ is a measurable function of s. Integrating (11.48) with respect to s we obtain

$$\begin{aligned}
\int_0^\infty \int_\Omega \sigma \cdot D\chi_{E_s}\,ds \;&-\; \int_0^\infty \int_{\partial\Omega} [\sigma \cdot \nu^\Omega]\chi_{E_s}\,d\mathcal{H}^{N-1}\,ds = \lambda \int_0^\infty \int_\Omega (1-u)h\chi_{E_s}\,dx\,ds \\
&= \lambda \int_\Omega (1-u)h \int_0^\infty \chi_{E_s}\,ds dx = \lambda \int_\Omega (1-u)uh\,dx \\
&= \int_\Omega \sigma \cdot Du + \int_{\partial\Omega} \phi(x, \nu^\Omega)|u|\,d\mathcal{H}^{N-1} \\
&= \int_\Omega |Du|_\phi + \int_{\partial\Omega} \phi(x, \nu^\Omega)|u|\,d\mathcal{H}^{N-1}.
\end{aligned}$$

Now, using Lemma 60 and the coarea formula (11.10), we have

$$\begin{aligned}\int_\Omega |Du|_\phi + \int_{\partial\Omega} \phi(x,\nu^\Omega)|u|\,d\mathcal{H}^{N-1} &= \int_{\mathbb{R}^N} |D\tilde{u}|_\phi = \int_0^\infty P_\phi(\{\tilde{u} > s\})\,ds \\ &= \int_0^\infty P_\phi(\{u > s\})\,ds,\end{aligned}$$

where $\tilde{u}$ denotes the extension of u by zero outside Ω. Hence

$$\int_0^\infty \int_\Omega \sigma \cdot D\chi_{E_s}\,ds - \int_0^\infty \int_{\partial\Omega} [\sigma\cdot\nu^\Omega]\chi_{E_s}\,d\mathcal{H}^{N-1}\,ds = \int_0^\infty P_\phi(\{u > s\})\,ds. \tag{11.49}$$

Let $s > 0$. Now, using Proposition 61 and the fact that $[\sigma \cdot \nu^\Omega] \in \operatorname{sign}(-u)\phi(x,\nu^\Omega(x))$, we have

$$\left|\int_\Omega \sigma\cdot D\chi_{E_s} - \int_{\partial\Omega}[\sigma\cdot\nu^\Omega]\chi_{E_s}\,d\mathcal{H}^{N-1}\right| \leq \int_\Omega |D\chi_{E_s}|_\phi + \int_{\partial\Omega}\phi(x,\nu^\Omega(x))\chi_{E_s}\,d\mathcal{H}^{N-1},$$

and using again Lemma 60 and the coarea formula, we may continue the equalities and obtain

$$\int_\Omega |D\chi_{E_s}|_\phi + \int_{\partial\Omega}\phi(x,\nu^\Omega(x))\chi_{E_s}\,d\mathcal{H}^{N-1} = \int_\Omega |D\tilde{\chi}_{E_s}|_\phi = \int_\Omega |D\chi_{E_s}|_\phi = P_\phi(E_s),$$

where $\tilde{\chi}_{E_s}$ is the extension of χ_{E_s} by zero outside Ω. Combining this inequality with (11.49), we obtain

$$\int_\Omega \sigma\cdot D\chi_{E_s} - \int_{\partial\Omega}[\sigma\cdot\nu^\Omega]\chi_{E_s}\,d\mathcal{H}^{N-1} = P_\phi(E_s) \qquad \text{a.e. } s>0. \tag{11.50}$$

Let $F \subseteq \Omega$ be a set of ϕ-finite perimeter. For $s \in (0,1]$, we have

$$\begin{aligned}-\int_\Omega \operatorname{div}\sigma\,(\chi_F - \chi_{E_s})\,dx &= \int_\Omega (\sigma, D\chi_F) - \int_\Omega \sigma\cdot D\chi_{E_s} - \int_{\partial\Omega}[\sigma\cdot\nu^\Omega](\chi_F-\chi_{E_s})\,d\mathcal{H}^{N-1} \\ &= \int_\Omega (\sigma, D\chi_F) - \int_{\partial\Omega}[\sigma\cdot\nu^\Omega]\chi_F - P_\phi(E_s) \leq P_\phi(F) - P_\phi(E_s),\end{aligned}$$

and we deduce

$$P_\phi(F) - P_\phi(E_s) \geq \lambda\int_\Omega (1-u)h(\chi_F-\chi_{E_s}) = \lambda\int_\Omega ((1-s)+(s-u))h(\chi_F-\chi_{E_s}).$$

Since $(s-u)(\chi_F-\chi_{E_s}) \geq 0$, we have

$$P_\phi(F) - P_\phi(E_s) \geq \lambda\int_\Omega (1-s)h(\chi_F-\chi_{E_s}) = \lambda(1-s)(|F|_h - |E_s|_h).$$

Since all sets E_s are contained in Ω, the perimeter is lower semicontinuous, and the area is continuous for increasing or decreasing families of sets contained in Ω, we deduce that (11.45) holds for any $s \in (0,1]$. □

Remark 6. *Let us observe that (11.50) holds for any $s > 0$. Indeed, by lower semicontinuity we have that $P_\phi(E_s) < \infty$ for all $s > 0$. If $s > 0$, we may approximate it by s_n such that (11.50) holds for s_n. Since*

$$P_\phi(E_{s_n}) = \int_\Omega \sigma \cdot D\chi_{E_{s_n}} - \int_{\partial\Omega} [\sigma \cdot \nu^\Omega]\chi_{E_{s_n}}\, d\mathcal{H}^{N-1} = -\int_\Omega \operatorname{div} \sigma\, \chi_{E_{s_n}},$$

we have $P_\phi(E_s) \le \liminf_n P_\phi(E_{s_n}) = -\int_\Omega \operatorname{div}\sigma\, \chi_{E_s} = \int_\Omega \sigma \cdot D\chi_{E_s} - \int_{\partial\Omega}[\sigma \cdot \nu^\Omega]\chi_{E_s}\, d\mathcal{H}^{N-1} \le P_\phi(E_s)$. The identity (11.50) holds for any $s > 0$.

Lemma 70 (Nonvanishing solutions of the variational problem)**.** *Let u_λ be the solution of $(Q)_\lambda$. If*

$$\frac{1}{\lambda} < \|h\chi_\Omega\|_* := \sup\left\{ \left| \int_\Omega hu\, dx \right| : u \in L^2(\Omega) \cap BV_\phi(\Omega), \right.$$
$$\left. \int_\Omega |Du|_\phi + \int_{\partial\Omega} \phi(x, \nu^\Omega)|u|\, d\mathcal{H}^{N-1} \le 1 \right\},$$

then $u_\lambda \neq 0$.

Proof. Notice that u_λ is characterized as the solution of (11.31). If $u_\lambda = 0$, then there exists a vector field $\sigma \in X_\infty(\Omega)$ with $\phi^0(x, \sigma(x)) \le 1$ a.e., $|[\sigma \cdot \nu^\Omega]| \le \phi(x, \nu^\Omega(x))$ $\mathcal{H}^{N-1}$-a.e. $x \in \partial\Omega$, and $-\lambda^{-1}\operatorname{div}\sigma = h\chi_\Omega$ in $\mathcal{D}'(\Omega)$. Multiplying by $u \in BV_\phi(\Omega)$ and integrating by parts, we obtain that $\|h\chi_\Omega\|_* \le \frac{1}{\lambda}$. □

If $\phi : \Omega \times \mathbb{R}^N \to \mathbb{R}$ is a metric integrand and ϕ_1, ϕ_2 are two extensions of ϕ to open sets Ω', Ω'', respectively, such that $\Omega \subset\subset \Omega', \Omega''$, and they coincide in $\Omega' \cap \Omega''$, then

$$P_{\phi_1}(E, \Omega') = P_{\phi_2}(E, \Omega'') = P_{\phi_i}(E, \Omega' \cap \Omega'') \qquad i = 1, 2,\ E \subseteq \Omega.$$

In particular, if ϕ has an extension to $\mathbb{R}^N$, denoted again by ϕ, then the above perimeters are also equal to $P_\phi(E)$. Thus, if ϕ has an extension to an open set Q containing $\overline{\Omega}$ we shall denote $P_\phi(E)$ instead of $P_\phi(E, Q)$ for any set $E \subseteq \Omega$ with finite ϕ-perimeter. Notice that if ϕ is continuous and coercive in a neighborhood of $Q \setminus \Omega$, then

$$P_\phi(E) = P_\phi(E, \Omega) + \int_{\partial\Omega} \phi(x, \nu^\Omega)\, \chi_E(x)\, d\mathcal{H}^{N-1} \qquad E \subseteq \Omega.$$

Notice that in this case, $P_\phi(E)$ depends only on $\phi : \overline{\Omega} \times \mathbb{R}^N \to \mathbb{R}$.

Lemma 71 (Perimeter of intersecting sets)**.** *Let Ω, Q be open bounded sets Lipschitz boundary such that $\Omega \subset\subset Q$. Let $\phi : Q \times \mathbb{R}^N \to \mathbb{R}$ be a metric integrand which is continuous and coercive in a neighborhood of $Q \setminus \Omega$. If E, F are two sets of finite ϕ-perimeter, then*

$$P_\phi(E \cup F) + P_\phi(E \cap F) \le P_\phi(E) + P_\phi(F). \tag{11.51}$$

Using the results of Subsection 11.1.4, the proof is exactly the same as in [Giu84] for the case of the Euclidean perimeter and we omit the details.

The following Lemma can be proved as in [ACC05a] and we also omit the details.

Lemma 72 (The variational problem on subsets)**.** *For any $\lambda > 0$, let us consider the problem*

$$(P)_\lambda : \quad \min_{F\subseteq\Omega} P_\phi(F) - \lambda|F|_h. \tag{11.52}$$

Then

(i) Let C_λ, C_μ be minimizers of $(P)_\lambda$, and $(P)_\mu$ respectively. If $\lambda < \mu$, then $C_\lambda \subseteq C_\mu$.

(ii) Let $\mu > \lambda$. Assume that Ω is a solution of $(P)_\lambda$. Then Ω is a solution of $(P)_\mu$.

(iii) Let $\lambda_n \uparrow \lambda$. Then $C_\lambda^+ := \cup_n C_{\lambda_n}$ is a minimizer of $(P)_\lambda$. Moreover $P_\phi(C_{\lambda_n}) \to P_\phi(C_\lambda^+)$. Similarly, if $\lambda_n \downarrow \lambda$, then $C_\lambda^- := \cap_n C_{\lambda_n}$ is a minimizer of $(P)_\lambda$, and $P_\phi(C_{\lambda_n}) \to P_\phi(C_\lambda^-)$.

(iv) If $\lambda \to \infty$ and C_λ is a minimizer of $(P)_\lambda$, then $C_\lambda \to \Omega$.

Remark 7. In Proposition 69 we have proved that for any $s \in (0,1]$, the level set $\{u_\lambda \geq s\}$ is a minimizer of $(P)_{\lambda(1-s)}$. Moreover, by Lemma 72, the sets $\{u_\lambda \geq s\}^+ := \cup_{\epsilon>0}\{u_\lambda \geq s+\epsilon\}$, $s \in [0,1)$, and $\{u_\lambda \geq s\}^- := \cap_{\epsilon>0}\{u_\lambda \geq s-\epsilon\}$, $s \in (0,1]$, are also minimizers of $(P)_{\lambda(1-s)}$ (obviously $\{u_\lambda \geq 1\}^+ = \emptyset$ is also a minimizer of $(P)_0$). Notice that, except on countably many values of s, they coincide with $[u_\lambda \geq s]$.

As a consequence of Lemma 72.(ii), we obtain:

Corollary 73. *Then for almost any λ, $(P)_\lambda$ has a unique solution.*

From Proposition 73 and Lemma 72.(*iii*) we deduce the following consequence.

Proposition 74 (Equivalence of solutions of the variational problem)**.** *Let $\alpha, \beta > \frac{1}{\|h\chi_\Omega\|_*}$. Then $\alpha(1-\|u_\alpha\|_\infty) = \beta(1-\|u_\beta\|_\infty)$.*

Proof. Assume that these two numbers are not equal. Without loss of generality, we may assume that

$$\alpha(1-\|u_\alpha\|_\infty) < \beta(1-\|u_\beta\|_\infty).$$

Let us take λ such that such that the solution of $(P)_\lambda$ is unique and $\alpha(1-\|u_\alpha\|_\infty) < \lambda < \beta(1-\|u_\beta\|_\infty)$. Let us write $\lambda = \alpha(1-s) = \beta(1-t)$ for some values $s < \|u_\alpha\|_\infty$, and $t > \|u_\beta\|_\infty$. Since $\{u_\beta \geq t\} = \emptyset$, and the solution of $(P)_\lambda$ is unique, being $\{u_\alpha \geq s\}$ a solution of $(P)_\lambda$, we deduce that $\{u_\alpha \geq s\} = \emptyset$, a contradiction. This proves our proposition. □

Let λ^* be the unique value of $\alpha(1-\|u_\alpha\|_\infty)$ determined by the above proposition.

Proposition 75 (Cheeger sets are level sets of the solution of the PDE)**.** *Let* $\alpha,\beta > \frac{1}{\|h\chi_\Omega\|_*}$. *Then* $\{u_\alpha \geq \|u_\alpha\|_\infty\} = \{u_\beta \geq \|u_\beta\|_\infty\}$, *and*

$$\lambda^* = \frac{P(\{u_\alpha \geq \|u_\alpha\|_\infty\})}{|\{u_\alpha \geq \|u_\alpha\|_\infty\}|_h}. \tag{11.53}$$

The set $\{u_\alpha \geq \|u_\alpha\|_\infty\}$ *is the maximal* ϕ*-Cheeger set of* Ω.

Proof. Let $\delta_n \to 0+$ be such that $(P)_{\lambda^*+\delta_n}$ has a unique solution for each n. Since $\{u_\alpha \geq \|u_\alpha\|_\infty - \frac{\delta_n}{\alpha}\}$, $\{u_\beta \geq \|u_\beta\|_\infty - \frac{\delta_n}{\beta}\}$ are both solutions of $(P)_{\lambda^*+\delta_n}$, we have that

$$\{u_\alpha \geq \|u_\alpha\|_\infty - \frac{\delta_n}{\alpha}\} = \{u_\beta \geq \|u_\beta\|_\infty - \frac{\delta_n}{\beta}\}.$$

Since $\{u_\alpha \geq \|u_\alpha\|_\infty\} = \cap_n\{u_\alpha \geq \|u_\alpha\|_\infty - \frac{\delta_n}{\alpha}\}$, and $\{u_\beta \geq \|u_\beta\|_\infty\} = \cap_n\{u_\beta \geq \|u_\beta\|_\infty - \frac{\delta_n}{\beta}\}$ we deduce that $\{u_\alpha \geq \|u_\alpha\|_\infty\} = \{u_\beta \geq \|u_\beta\|_\infty\}$, and this set minimizes $(P)_{\lambda^*}$.

Now, since $\{u_\alpha \geq \|u_\alpha\|_\infty + \epsilon\} = \emptyset$ is a solution of $(P)_{\lambda^*-\lambda\epsilon}$, for all $\epsilon > 0$, by Lemma 72.(*iii*), we have that $\emptyset$ is also a solution of $(P)_{\lambda^*}$. Then

$$P_\phi(\{u_\alpha \geq \|u_\alpha\|_\infty\}) - \lambda^*|\{u_\alpha \geq \|u_\alpha\|_\infty\}|_h = P_\phi(\emptyset) - \lambda^*|\emptyset|_h = 0,$$

and (11.53) follows. Since $\{u_\alpha \geq \|u_\alpha\|_\infty\}$ is a minimizer of $(P)_{\lambda^*}$ we deduce that

$$0 = P_\phi(\{u_\alpha \geq \|u_\alpha\|_\infty\}) - \lambda^*|\{u_\alpha \geq \|u_\alpha\|_\infty\}|_h \leq P_\phi(F) - \lambda^*|F|_h$$

for any set $F \subseteq \Omega$ of finite perimeter. Then

$$\frac{P_\phi(\{u_\alpha \geq \|u_\alpha\|_\infty\})}{|\{u_\alpha \geq \|u_\alpha\|_\infty\}|_h} \leq \frac{P_\phi(F)}{|F|_h}$$

for any set $F \subseteq \Omega$ of finite perimeter. Thus, the set $\{u_\alpha \geq \|u_\alpha\|_\infty\}$ is a ϕ-Cheeger set of Ω. Now, if C is any other ϕ-Cheeger set in Ω, then C is a solution of $(P)_{\lambda^*}$. Then $C \subseteq \{u_\alpha \geq \|u_\alpha\|_\infty - \frac{\delta_n}{\alpha}\}$ for all n. Then $C \subseteq \{u_\alpha \geq \|u_\alpha\|_\infty\}$. We conclude that $\{u_\alpha \geq \|u_\alpha\|_\infty\}$ is the maximal ϕ-Cheeger set of Ω. □

11.3 Local Finsler-Cheeger sets

In this Section we assume that ϕ *is continuous and coercive in* Ω. Let $E \subseteq \mathbb{R}^N$ be a set of finite perimeter. We say that E is decomposable if there exists a partition (A,B) of E such that $P_\phi(E) = P_\phi(A) + P_\phi(B)$ and both $|A|$ and $|B|$ are strictly positive. We say that E is indecomposable if it is not decomposable; notice that the properties of being decomposable or indecomposable are invariant modulo Lebesgue null sets and that, according to our definition, any Lebesgue negligible set is indecomposable.

The following result was proved in [ACMM01] for the Euclidean perimeter. The proof easily extends to cover the case where ϕ is continuous and coercive in Ω, but it also follows from the Euclidean case since the assumptions on ϕ imply that

$$P_\phi(E) = \int_{\partial^* E} \phi(x, \nu^E(x))\, d\mathcal{H}^{N-1}$$

for any set $E \subseteq \mathbb{R}^N$ with finite perimeter.

Theorem 76 (Decomposition of sets)**.** *Let E be a set with finite perimeter in $\mathbb{R}^N$. Then there exists a unique finite or countable family of pairwise disjoint indecomposable sets $\{E_i\}_{i\in I}$ such that $|E_i| > 0$ and $P_\phi(E) = \sum_i P_\phi(E_i)$. Moreover the sets E_i's are maximal indecomposable sets, i.e. any indecomposable set $F \subseteq E$ is contained modulo a Lebesgue null set in some set E_i.*

In view of the previous Theorem, we call the sets E_i the ϕ-connected components of E.

Proposition 77 (Local Cheeger sets)**.** *Assume that ϕ is continuous and coercive in Ω. Let $u \in BV_\phi(\Omega) \cap L^2(\Omega, h\,dx)$ be the solution of (11.32). Let $t \in (0,1]$ and $E_t := \{u \geq t\}$. Let E'_t be a ϕ-connected component of E_t and $F_s = \{u \geq s\} \cap E'_t$, $s \geq t$. Then for any $s \in (0,1]$ we have*

$$P_\phi(F_s) - \lambda(1-s)|F_s|_h \leq P_\phi(F) - \lambda(1-s)|F|_h \tag{11.54}$$

for any $F \subseteq E'_t$. If $s = \max_{x\in E'_t} u(x)$, then F_s is a maximal ϕ-Cheeger set in E'_t.

The sets F_s will be called local ϕ-Cheeger sets.

Proof. Let $\{E^i_s\}_{i\in I}$ be the ϕ-connected components of E_s. Since $\chi_{E_s} = \sum_i \chi_{E^i_s}$, $|\chi_{E_s}| = \sum_i |\chi_{E^i_s}|$, $\sigma \cdot D\chi_{E_s} = \sum_i \sigma \cdot D\chi_{E^i_s}$, and

$$\int_\Omega \sigma\cdot D\chi_{E^i_s} - \int_{\partial\Omega} [\sigma\cdot\nu^\Omega]\chi_{E^i_s}\, d\mathcal{H}^{N-1} \leq P_\phi(E^i_s, \Omega) + \int_{\partial\Omega} \phi(x,\nu^\Omega)\chi_{E^i_s}\, d\mathcal{H}^{N-1} = P_\phi(E^i_s),$$

from the extension of (11.50) given in Remark 6 we have that

$$\int_\Omega \sigma \cdot D\chi_{E^i_s} - \int_{\partial\Omega} [\sigma\cdot\nu^\Omega]\chi_{E^i_s}\, d\mathcal{H}^{N-1} = P_\phi(E^i_s),$$

for any $i \in I$. Now, we can proceed as in the proof of Proposition 69 to get that (11.54) holds. The last assertion follows as in Propositions 74 and 75. □

Recall that, when ϕ is coercive, by the isoperimetric inequality there is a constant $\alpha > 0$ (depending on the domain) such that any ϕ-Cheeger set has measure $\geq \alpha$. Moreover, the union and intersection of ϕ-Cheeger sets are ϕ-Cheeger [CCN09]. In particular, there are minimal ϕ-Cheeger sets and there are finitely many of them [CCN09].

12 Numerical computation of Finsler-Cheeger sets

In this chapter we explain some numerical methods for computing Finsler-Cheeger sets. On Section 12.1 we describe a finite difference scheme to solve the PDE (11.31). On Section 12.2 we explain how to compute the level-sets of (11.53) and their perimeters to the required sub-pixel precision.

12.1 Minimization of the dual problem by finite differences

In this Section we present an adaptation of Chambolle's algorithm [Cha04] that permits to solve a discrete version of (11.31) for some particular instances of $\phi(x,\xi)$. For simplicity, the formulas with discrete indices are restricted to the 2D case, but they are immediately generalizable to higher dimensions. Let us give some notations that we use in the sequel, keeping in mind that, for simplicity, we will denote the discrete functions we use like their continuous counterparts.

Let us consider the discrete domain $\hat{\Omega} = \{0,1,\dots,N-1\}^2$ (more generally, we could assume that $\hat{\Omega} \subseteq \{0,1,\dots,N-1\}^2$). For convenience, let us denote $\hat{\Omega}^e$ the extended domain $\{-1,0,\dots,N\}^2$. We denote by U the Euclidean space $\mathbb{R}^{(N+2)\times(N+2)}$. Let us give the definition of the discrete gradient which is adapted to problem (11.31) (which considers Dirichlet boundary conditions). In Section 12.1.2 we shall use Neumann boundary conditions with the definition of the gradient and divergence taken as in [Cha04]. Given $u \in U$ its discrete gradient ∇u will be a vector in $V := U \times U$ given by $(\nabla u)_{i,j} = \big((\nabla u)^1_{i,j}, (\nabla u)^2_{i,j}\big)$, $(i,j) \in \hat{\Omega}^e$, where

$$(\nabla u)^1_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j} & \text{if (i+1,j),(i,j)} \in \hat{\Omega} \\ -u_{i,j} & \text{if (i+1,j)} \notin \hat{\Omega},\ \text{(i,j)} \in \hat{\Omega} \\ u_{i+1,j} & \text{if (i+1,j)} \in \hat{\Omega},\ \text{(i,j)} \notin \hat{\Omega} \\ 0 & \text{if (i+1,j),(i,j)} \notin \hat{\Omega}, \end{cases} \tag{12.1}$$

$$(\nabla u)^2_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{if (i,j+1),(i,j)} \in \hat{\Omega} \\ -u_{i,j} & \text{if (i,j+1)} \notin \hat{\Omega},\ \text{(i,j)} \in \hat{\Omega} \\ u_{i,j+1} & \text{if (i,j+1)} \in \hat{\Omega},\ \text{(i,j)} \notin \hat{\Omega} \\ 0 & \text{if (i,j+1),(i,j)} \notin \hat{\Omega}. \end{cases} \tag{12.2}$$

The above case amounts to say that $u_{i,j} = 0$ when the indexes are in $\hat{\Omega}^e \setminus \hat{\Omega}$. These definitions of gradient embody the Dirichlet boundary conditions. The extension of the anisotropy ϕ to $\hat{\Omega}^e$ will be made precise in the examples below. The scalar product and the norm in U are defined as usual and denoted by $\langle\cdot,\cdot\rangle_U$ and $\|\cdot\|_U$, but in absence of ambiguities the subindex will be omitted. In V the scalar product is denoted $\langle p,q\rangle_V = \sum_{i,j\in\hat{\Omega}} p^T_{i,j}\, q_{i,j}$ and the norm $\|p\|_V = \langle p,p\rangle_V$. Finally, the divergence is defined so that it verifies $\langle p, \nabla u\rangle_V = -\langle \operatorname{div} p, u\rangle_U$

$$(\operatorname{div} p)_{i,j} = \begin{cases} p^1_{i,j} - p^1_{i-1,j} & \text{if (i,j)} \in \hat{\Omega} \\ 0 & \text{if (i,j)} \notin \hat{\Omega}, \end{cases} + \begin{cases} p^2_{i,j} - p^2_{i,j-1} & \text{if (i,j)} \in \hat{\Omega} \\ 0 & \text{if (i,j)} \notin \hat{\Omega}. \end{cases} \tag{12.3}$$

12.1.1 Example 1: geodesic active contour type models

Let us consider the following generalization of the problem studied by Chambolle in [Cha04]

$$\min_{u\in U, u=0 \text{ in } \hat{\Omega}^e\setminus\hat{\Omega}} \frac{\|(u-f)h^{1/2}\|_U^2}{2}+\lambda^{-1}J_g(u), \quad \text{where} \quad J_g(u)=\sum_{(i,j)\in\hat{\Omega}^e} g_{i,j}|(\nabla u)_{i,j}|, \tag{12.4}$$

$f, g, h \in U$ and $h_{i,j} > 0$ for all $(i,j) \in \hat{\Omega}^e$. We consider these functions defined originally in $\hat{\Omega}$ and extended to $\hat{\Omega}^e$ by specular symmetry. Observe that the Euler equation of (12.4) is a discretization of (11.31) where $\phi(x,\nabla u) = g(x)|\nabla u|$ (discrete case), and where we write $\partial_\xi \phi(x,\nabla u) = g\frac{\nabla u}{|\nabla u|}$,

$$hu - \lambda^{-1}\text{div}\left(g\frac{\nabla u}{|\nabla u|}\right) \ni hf. \tag{12.5}$$

As in [Cha04] let us derive the dual formulation for (12.4) by re-writing (12.5) as $h\lambda(f-u) \in \partial J_g(u)$ which is equivalent to $u \in \partial J_g^*(h\lambda(f-u))$ where J_g^* is the Legendre-Fenchel transform of J_g. Writing $w = \lambda(f-u)$ we have

$$0 \in (w-\lambda f)h + h\lambda \partial J_g^*(hw) \tag{12.6}$$

which is the minimizer of the *dual* problem

$$\min_{w\in U} \frac{\|h^{1/2}w - b\|_U^2}{2} + \lambda J_g^*(hw) \quad \text{with} \quad b = h^{1/2}\lambda f. \tag{12.7}$$

Since J_g is homogeneous, then J_g^* is the indicator function of a convex set $\mathcal{K}_g$ given by

$$J_g^*(w) = \begin{cases} 0 & \text{if } w \in \mathcal{K}_g \\ +\infty & \text{otherwise} \end{cases} \text{ with } \mathcal{K}_g = \left\{-\text{div}\,\xi : \xi \in V, |\xi_{i,j}| \le g_{i,j}\ \forall (i,j)\in\hat{\Omega}^e\right\}. \tag{12.8}$$

Therefore we may write (12.7) as

$$\min_{hw\in\mathcal{K}_g} \|h^{-1/2}h\,w - b\|_U^2. \tag{12.9}$$

Note that any solution $hw \in \mathcal{K}_g$, must satisfy $h_{i,j}w_{i,j} = -\text{div}\,(g_{i,j}p_{i,j})$ with $|p_{i,j}| \le 1$. Hence we may write (12.9) as

$$\begin{aligned} &\min_{p\in V} \|h^{-1/2}\text{div}\,(gp) + b\|_U^2 \\ &\text{s.t.} \quad |p_{i,j}|^2 - 1 \le 0 \quad \forall (i,j)\in\hat{\Omega}^e \end{aligned} \tag{12.10}$$

and introducing the Lagrange multipliers $\alpha_{i,j}$ for the constraint we obtain the functional

$$\mathcal{F}(p,\alpha) = \sum_{(i,j)\in\hat{\Omega}^e} |h_{i,j}^{-1/2}\text{div}\,(gp)_{i,j} + b_{i,j}|^2 + \sum_{(i,j)\in\hat{\Omega}^e} \alpha_{i,j}(|p_{i,j}|^2 - 1), \quad \alpha \in U,\ p \in V.$$

Proceeding as in [Cha04] the solution of (12.10) satisfies:

$$-\left[g\nabla(h^{-1}\mathrm{div}\,(gp)+\lambda f)\right]_{i,j}+\alpha_{i,j}p_{i,j}=0 \quad \forall (i,j)\in\hat{\Omega}^e. \tag{12.11}$$

The Karush-Khun-Tucker's Theorem yields the existence of the Lagrange multipliers $\alpha^*_{i,j}\geq 0$ for the constraints in (12.11), which are either $\alpha^*_{i,j}>0$ if $|p_{i,j}|=1$ or $\alpha^*_{i,j}=0$ if $|p_{i,j}|<1$ but in this case also $[g\nabla(h^{-1}\mathrm{div}\,(gp)+\lambda f)]_{i,j}=0$. In any case $\alpha^*_{i,j}=|[g\nabla(h^{-1}\mathrm{div}\,(gp)+\lambda f)]_{i,j}|$, and substituting it into (12.11) and using a gradient descent we arrive to the following fixed-point algorithm

$$p^{n+1}=\frac{p^n+\tau\left\{g\nabla[h^{-1}\mathrm{div}\,(gp^n)+\lambda f]\right\}}{1+\tau|g\nabla[h^{-1}\mathrm{div}\,(gp^n)+\lambda f]|}, \tag{12.12}$$

where the maximum $\tau>0$ will depend on the chosen discretization. For the present scheme, with a straightforward computation [Cha04, ACHR06], one can show that the method converges if $\tau<\frac{1}{8}\frac{1}{\max|g|^2}\frac{1}{\max|h^{-1/2}|^2}$. At convergence, the solution is obtained using the formula $u=f+\lambda^{-1}h^{-1}\mathrm{div}\,(gp)$.

Let us summarize the steps of the algorithm:

Chambolle's algorithm with Dirichlet boundary conditions:

1. Initialize $p^0=0\in V$, $q^0=0\in U$ and $t=0$
2. Iterate until convergence:
   a) Compute: $p^{t+1}\leftarrow\frac{p^t+\tau g\nabla q^t}{1+\tau|g\nabla q^t|}$
   b) Compute: $q^{t+1}\leftarrow h^{-1}\mathrm{div}\,(gp^{t+1})+\lambda f$
3. Recover the solution $u=\lambda^{-1}q^{t+1}$

In Step 2(a) p^t is updated for $(i,j)\in\hat{\Omega}^e$, while in Step 2(b) q^t may be updated only for $(i,j)\in\hat{\Omega}$ and $q^t_{i,j}=0\ \forall(i,j)\in\hat{\Omega}^e\setminus\hat{\Omega}$.

Remark 8. *Let us check that the solution u obtained from the fixed point of (12.12) verifies a discrete version of the boundary conditions of Definition 67. That is, the field σ satisfies (a discrete version of) $[\sigma,\nu^\Omega]\in\mathrm{sign}(-u(x))\phi(x,\nu^\Omega(x))$ where ν^Ω denotes the outer unit normal to the boundary, and $\phi(x,\nu^\Omega(x))=g(x)$. The outer unit normals at the points of the discrete boundaries take only four possible values $\nu^{\hat{\Omega}}\in\left\{\binom{1}{0},\binom{-1}{0},\binom{0}{1},\binom{0}{-1}\right\}$, we describe here just one direction (i.e. the left boundary of the domain).*

Thus, let us assume that $i=-1$ so that we are at the left boundary side. We check that $[gp,\nu^{\hat{\Omega}}](-1,j)\in\mathrm{sign}(-u_{0,j})\phi((0,j),(-1,0)^t)=\mathrm{sign}(-u_{0,j})g_{0,j}$, which is a discrete way of imposing the boundary condition. Notice that the fixed point solution p of (12.12) satisfies $g_{-1,j}p_{-1,j}\nabla q_{-1,j}=g_{-1,j}|\nabla q_{-1,j}|$ where $q=h^{-1}div\,(gp)+\lambda f$ (note as well that $\sigma=gp$). At the left side of the boundary we have $\nabla q_{-1,j}=\binom{q_{0,j}}{q_{-1,j+1}-q_{-1,j}}$ (with $q_{-1,j+1}-q_{-1,j}=0$ since both pixels are in $\hat{\Omega}^e\setminus\hat{\Omega}$). If $u_{0,j}>0$, then $q_{0,j}>0$. Therefore $[gp,\nu^{\hat{\Omega}}](-1,j)=-g_{-1,j}\,q_{0,j}/|q_{0,j}|=$

$g_{0,j}\,\mathrm{sign}(-u_{0,j})$. *If* $u_{0,j} = 0$, *then* $[gp, \nu^{\hat{\Omega}}](-1, j) \in g_{0,j}\,\mathrm{sign}(-u_{0,j})$, *since* $|[gp, \nu^{\hat{\Omega}}](-1, j)| \leq g_{0,j}$. *The computation for the other tree sides of the boundary can be done in a similar way.*

12.1.2 Example 2: An anisotropic diffusion type problem

In this example we consider an anisotropic diffusion problem with Neumann boundary conditions. The discretization of the gradient and divergence are the same as in [Cha04]. Let us consider the anisotropic total variation with $\phi(x, \xi) = |A_x\xi|$, $\forall x \in \Omega$, where A_x is a symmetric and positive definite (hence, invertible) matrix. As before, the solution of the minimization problem

$$\min_{u\in U} \frac{\|h^{1/2}(u-f)\|_U^2}{2} + \lambda^{-1} J_\phi(u) \quad \text{with} \quad J_\phi(u) = \sum_{0\leq i,j\leq N-1} \phi((i,j), \nabla u(i,j))$$

is obtained via its dual formulation

$$\min_{w\in U} \frac{\|h^{1/2}w - h^{1/2}\lambda f\|_U^2}{2} + \lambda J_\phi^*(hw) \quad \text{with} \quad w = (f-u)\lambda. \tag{12.13}$$

Since J_ϕ is homogeneous, then J_ϕ^* is the characteristic function of a set $\mathcal{K}_\phi$ which we will characterize next. Following [AB94] we have $\mathcal{K}_\phi = \{-\mathrm{div}\,\xi^* : \phi^0(x, \xi^*) \leq 1\}$, where

$$\phi^0(x,\xi^*) = \begin{cases} 0, & \text{if } \xi^* = 0 \\ +\infty, & \text{if } \xi^* \notin Z_x^\perp \\ \sup\limits_{\xi:\phi(x,\xi)\leq 1} \langle \xi, \xi^* \rangle, & \text{if } \xi^* \in Z_x^\perp \setminus \{0\}, \end{cases}$$

with $Z_x = \{\xi : \phi(x,\xi) = 0\} = \{\xi : |A_x\xi| = 0\}$. Since A_x is symmetric and invertible, then $Z_x = \{0\}$ and $Z_x^\perp = \mathbb{R}^n$. Since the second condition is empty, and $\sup_{\xi:|A_x\xi|\leq 1} \langle A_x^{-1}A_x\xi, \xi^* \rangle = |A_x^{-1}\xi^*|$, then $\phi^0(x,\xi^*) \leq 1$ if and only if $\xi^* = A_x p$ with $|p| \leq 1$. We get that

$$\mathcal{K}_\phi = \{-\mathrm{div}\,\xi^*(x) : \xi^*(x) = A_x p(x), |p(x)| \leq 1, \forall x \in \Omega\}.$$

This allows to write the problem (12.13) as

$$\min_{p(x):|p(x)|\leq 1} \|h^{-1/2}\mathrm{div}\,(A_x p(x)) + h^{1/2}\lambda f\|_U^2 \tag{12.14}$$

and derive the following fixed point algorithm

$$p^{n+1} = \frac{p^n + \tau\left\{A_x \nabla[h^{-1}\mathrm{div}\,(A_x p^n) + \lambda f]\right\}}{1 + \tau|A_x \nabla[h^{-1}\mathrm{div}\,(A_x p^n) + \lambda f]|}. \tag{12.15}$$

At convergence, the solution is obtained using the formula $u = f + \lambda^{-1}h^{-1}\mathrm{div}\,(A_x p)$. For applications to image diffusion it is more adequate the use of Neumann boundary conditions, which are imposed by adapting the definitions of the gradient and divergence as in [Cha04].

12.2 Numerical computation of Finsler-Cheeger sets

The numerical scheme described above produces a function u which is a solution of the PDE (11.31). By Proposition 69, the level sets of u are global minima of the anisotropic ϕ-perimeter with an inflating force. And the regional maxima of u are local ϕ-Cheeger sets in a suitable domain containing them.

In this Section we describe a method for finding these local ϕ-Cheeger sets. We want to define local extrema of a function $P_\phi(\cdot)/|\cdot|$ which is defined on the set of all connected components of upper level sets of an image u. To fix ideas, let us assume that $N = 2$. Then we have to examine the connected components of the upper level sets $\{u > t\}$, $t \in (0, 1]$, of the solution u of (11.31). The ϕ-Cheeger set is defined by $\{u = \|u\|_\infty\}$ but due to the floating point operations we cannot proceed to a direct computation of this set. Instead we take the ϕ-Cheeger set as the minimum of $t \to P_\phi(\{u > t\})/|\{u > t\}|$ with a suitable discretization of the variable t. Similarly, to compute the local ϕ-Cheeger sets we use the tree of connected components of upper level sets of the image (see [MG00, CM10] and the first part of this thesis) and look for the local minima of $P_\phi(cc\{u > t\})/|cc\{u > t\}|$ where $cc\{u > t\}$ denotes a connected component of $\{u > t\}$. Thanks to the topological structure of the tree of connected components of upper level sets, we can speak of local extrema of functions defined on that set. Intuitively, a neighborhood of $\Gamma_t = cc\{u > t\}$ consists in those connected components of upper level sets whose levels are slightly above or below the level of Γ_t.

Let us explain how to compute the weigthed perimeter and volume of a given level set. Then we will show how to use this computation to obtain an efficient algorithm to find the connected components of the upper level sets which are local minimizers of the ϕ-Cheeger ratio. When ϕ is of the form $\phi(x, \xi) = g(x)|\xi|$, we will use the expression g-Cheeger set.

12.2.1 Sub-pixel computation of weigthed perimeters and areas

Notice that it is not trivial to compute the perimeter of a set which is defined by pixels or voxels. The naive approach of counting the voxels which touch the boundary of the region does not work, mainly because this quantity is not invariant by rotations. There are two common solutions to this problem: approximate the perimeter using integral geometric measure techniques as in graph cuts [BK03] or approximate the ragged boundary of the set by a smoother surface and compute its perimeter. We found the second option best suited to our needs because, as the goal is to compute perimeters of level sets, we can produce high-resolution approximations of their boundaries by methods such as Marching Cubes or Marching Squares [LC87]. Once we have a triangulated surface, we can compute its weighted perimeter by adding the areas of all triangles, each one multiplied by the weight ϕ interpolated at the barycenter of each triangle.

To test the consistency and precision of this scheme, let us consider a spherical image $u(x) = f(|x - x_0|)$ whose profile f is an increasing function from $[0, +\infty)$

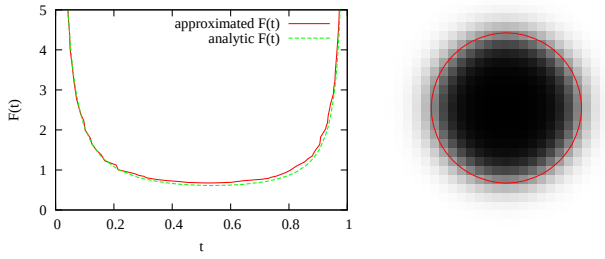


Figure 12.1: Numerical evaluation of $\frac{1}{|\nabla I|}$-Cheeger ratios for a synthetic image where they can also be computed analytically. The image is given by the model in (12.16) with $\kappa = 8$ and $s = 1$. Left: graphs of the exact and computed $F(t)$ for that image. Right: interpolated level curve at which the minimum of $F(t)$ is attained, overlaid on the original image. This example has a very low resolution (the original image has dimensions 31×31. For higher resolutions the approximation is nearly perfect and the two curves $F(t)$ are visually identical.

to $[0, 1)$. For each t in $(0, 1)$, the level set of value t is a sphere of radius $r(t) = f^{-1}(t)$ centered at x_0, and this surface is weighted by $\frac{1}{|\nabla u|} = \frac{1}{f'(r(t))} = r'(t)$. The $\frac{1}{|\nabla u|}$-Cheeger ratio is then

$$F(t) = \frac{NV_N r^{N-1} r'}{V_N r^N} = N\frac{r'(t)}{r(t)},$$

where V_N is the volume of the unit ball of $\mathbb{R}^N$. This function $F(t)$ is a real-valued function whose minimum can be evaluated numerically, or even analytically in some easy cases.

We can set, for example,

$$f(x) = \frac{1}{1 + \exp\frac{\kappa - x}{s}}, \tag{12.16}$$

where κ and s are parameters, such as $\kappa = 8$ and $s = 0.1$. Intuitively, the desired segmentation of this image is a circle of radius $\kappa \pm s$, or equivalently some level set of value near $t = \frac{1}{2}$. On Figure 12.1, we compare the graphs of the $\frac{1}{|\nabla u|}$-Cheeger ratio over t as computed analytically and with the numerical methods described above. The minima is in both cases attained very near $t = \frac{1}{2}$ which agrees with our intuition and suggests that the numerical approximations we use are consistent.

As another numerical test, we computed the Euclidean Cheeger set of a square and a cube (using the Euclidean metric). See Figures 12.2 and 12.3 for the plausible result obtained.

The discrete images u are obtained by an iterative numerical method, and they have floating point values. Most of the values are concentrated around 1, with

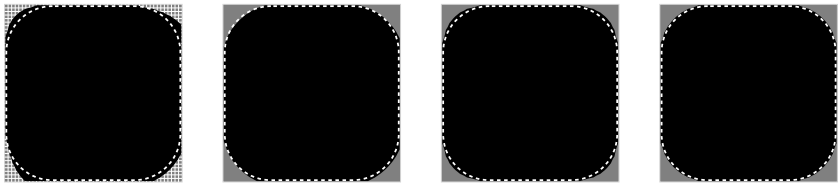


Figure 12.2: Computation of the Euclidean Cheeger sets (black regions) for a square of 50 pixels. For the square the analytical Cheeger set is known (dashed line) and its constant is 1/13.253 (for readability we display the inverse of the Cheeger constant). The first column shows the results obtained with the Chambolle's numerical scheme [Cha04], observe that it is asymmetric. The second solution is obtained by increasing the sampling (as proposed by Chambolle), this improves considerably the solution, but the asymmetry persists. The third column shows a symmetric solution obtained by averaging all the numerical schemes (4 in 2D). The fourth solution is obtained using the *consensus algorithm*. All the results where obtained after 20000 iterations of (12.12).

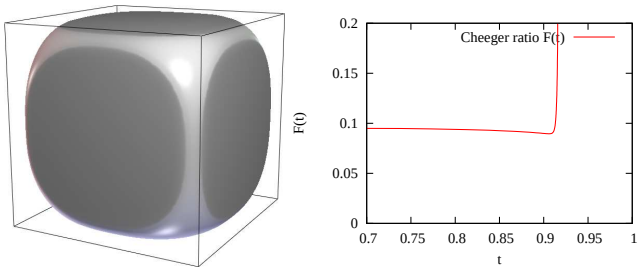


Figure 12.3: Cheeger set of a cube. This was computed by Euclidean total variation minimization of the characteristic function of the cube, followed by the selection of the level surface that minimized the Cheeger ratio. On the left, graph of $F(t)$. Notice that for large values of t, the level sets collapses to a central point of the domain, so that its Cheeger ratio is much larger than the optimum.

the interesting part of the range being often contained in the interval $[0.99, 1]$. Thus, it is important to conserve their floating-point values. This implies that there are as many different level surfaces as pixels, one for each different floating-point value. But it is not necessary to compute the ϕ-Cheeger ratio of all of these surfaces: via dichotomic search we can efficiently locate the minimum.

Remark 9. On the symmetry of the numerical scheme. *In Figure 12.2 we show an example of Cheeger set determined with the method described above. Observe that the first solutions are asymmetric, this effect is particularly clear (and annoying) for small images (the first square is 50 pixels high) and it is due to the forward/backward scheme adopted to discretize the gradient* (12.1)*. In [Cha04] the author remarks that the finite difference scheme converges to the continuous formulation as the number of samples $N \to \infty$, but when applied to volumetric images increasing the sampling is not an affordable option. To maintain the symmetry of the solutions, we propose the* consensus algorithm*, which computes the mean solution of all the finite difference schemes (4 schemes in 2D, and 8 in 3D) at each iteration. The consensus algorithm outperforms the considered schemes while keeping the symmetry but it is 4 times slower than the standard finite differences scheme (or 8 times slower for 3D images).*

12.2.2 Computing the minimum of the geodesic active contour model with an inflating force

Given an image $I : \Omega \to \mathbb{R}$, let us consider the following formulation of geodesic active contour (GAC) with an inflating force

$$\min_{E \subseteq \Omega} P_g(E) - \mu |E|_h \qquad \mu > 0, \tag{12.17}$$

where $P_g(E)$ is a weighted perimeter with weights $g(x) = (\sqrt{1 + |\nabla (G * I)|^2})^{-1}$, $|E|_h = \int_E h(x)\, dx$ is the weighted area, and μ is a parameter that controls the balloon force. In Proposition 69 we have shown that if u is the solution of the problem (11.32), then $E_s := \{u \geq s\}, s \in (0, 1]$, is a global minimum of (12.17) with $\mu = \lambda(1 - s)$. In particular, if $\lambda > \mathcal{C}_\Omega$ (the g-Cheeger constant) then the set $\{u = \|u\|_\infty\}$ is the g-Cheeger set of Ω. Therefore to compute the solutions of (12.17) it suffices to solve the problem (11.32) for some $\lambda > \mu$, or, equivalently, to find u by solving

$$hu - \lambda^{-1} \mathrm{div} \left(g \frac{Du}{|Du|} \right) = h. \tag{12.18}$$

The solution of (12.18) is computed using the scheme proposed by Chambolle [Cha04] and described in Section 12.1.2 (with $f = \chi_\Omega$). For λ big enough, for all values of $\mu > 0$, the solutions of (12.17) can be found as the level sets of u. In practice we select the g-Cheeger set as the upper level set of u that minimizes

$$\min_{\Gamma \subseteq \Omega} \frac{P_g(\Gamma)}{|\Gamma|_h},$$

where the minimum is taken over the upper level sets of u.

12.2.3 Three ways to approximate the Cheeger ratio.

Given an image of pixels (or voxels), there are three natural ways to compute the $\frac{1}{|\nabla I|}$-perimeters of its level sets. In increasing order of precision, they are:

1. Substract the discrete areas of two nearby level sets, and divide by the gray-level increment (according to equation 14.3 of next chapter).
2. Substract the sub-pixelic areas as found by a triangulation of the level sets.
3. Compute perimeter of the triangulated boundary, pointwise weighted by $\frac{1}{|\nabla I|}$.

The first method is straightforward to implement, and does not even need computing gradients: simply counting pixels of bi-level sets. However, it is very imprecise due to the heavy discretization (Figure 12.4). This method is useful when local minimizers of the Cheeger ratio are used as affine invariants for object detection, as described on the next chapter. The third method is the most precise (Figure 12.5) and is useful when an exact smooth Cheeger set is required. It is also useful to increase the precision and robustness of the affine invariants, at the cost of a slight increase of computation time.

See Figures 12.4 and 14.2 for a comparison of these methods and their results.

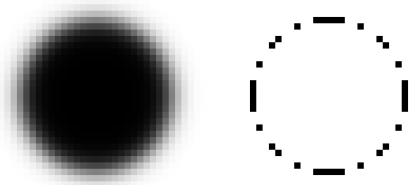


Figure 12.4: A synthetic image, and the pixels belonging to a central bi-level set. The count of these pixels is a rough estimate of the area between two level lines.

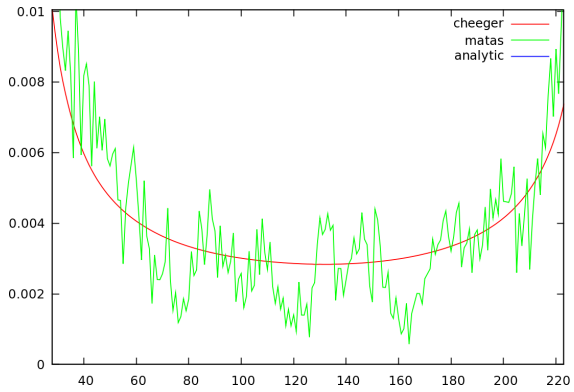


Figure 12.5: Comparison of Cheeger vs Matas approximations for the synthetic image on figure 14.2.

13 Applications of Finsler-Cheeger sets

On this chapter we expose several experiments that make use of the numerical methods for computing Finsler-Cheeger sets.

13.1 Framework for the applications

We have used the theory above in two different ways, corresponding to different choices of a metric integrand g. The first choice is $g(x) = (\sqrt{\epsilon + |\nabla(G * I)|^2})^{-1}$, where ϵ is a scale parameter and G is a small convolution kernel. The second choice is the distance function to the set of edge points detected by a preprocessing of the image, that is $g = d_S$, where S is the set of edges of u. We label these two cases respectively $\frac{1}{|\nabla I|}$-*Cheeger sets* and d_S-*Cheeger sets*. We observe that the convergence of the iterative scheme to solve the PDE is much faster for d_S-Cheeger sets, and the result is less likely to miss parts of the image. On the other hand, the computation of $\frac{1}{|\nabla I|}$-Cheeger sets gives smooth results after a long time, and sometimes misses parts of the desired objects, or fails to break at holes. The choice of a subdomain $B \subseteq \Omega$ allows for some flexibility: we can enforce hard restrictions on the result by removing from the domain some points that we do not want to be enclosed by the output surfaces.

Notice that, for a given choice of g, we actually find many *local* g-Cheeger sets, disjoint from the global minimum, that appear as local minima of the g-Cheeger ratio on the tree of connected components of upper level sets. The computation of those sets is partially justified by Proposition 77. Notice that the assumptions in it do not cover the case where g vanishes. These are the sets which we show on the following experiments.

13.2 Segmentation and edge linking

2D images On Figure 13.1, we display some local g-Cheeger sets of 2D images for different choices of metric g. These experiments are equivalent to applying the model (12.17) to edge linking problems. As in [Coh91] the inflating force allows to link the pieces of the boundaries of the objects. We display in Figure 13.1 some 2D linking experiments, that show how the d_S-Cheeger set indeed links the edges. Let us remark here a limitation of this approach, that can be observed in the last subfigure. Even if this linking is produced, the presence of a bottleneck (bottom right subfigure) makes the d_S-Cheeger set to be a set with large volume. This limitation can be circumvented by adding barriers in the domain Ω.

Synthetic 3D image The first 3D example is a synthetic image built in the following way. We have taken the characteristic function of a slanted torus plus a linear function and then added some blurring and gaussian noise to the result. Some slices and a level surface of this image are shown on the left subfigure of Figure 13.2. The first experiment with this synthetic image has been to segment

it using the $\frac{1}{|\nabla I|}$-Cheeger set of the image domain. This gives a reasonable segmentation of the object, as shown on Figure 13.2. The second experiment with this synthetic image has been to perform edge linking. We have taken the output of an edge detector [DMM01, MZFC09] to it, and used the distance function to the set of edges as a metric. The d_S-Cheeger set of the image domain is a surface that correctly interpolates the given patches. We can observe that the result of the edge linking has a ragged appearance. On Figure 13.3 we display the input edges, the corresponding metric and the final result.

Real 3D CT image The first real 3D example is based on a Computed Tomography of cerebral arteria containing an aneurysm. We have tried both $\frac{1}{|\nabla I|}$ and d_S metrics (where S is computed, as before, by an edge detector). The results are visually similar. Noticing that both methods give an incorrect segmentation on a small part of the image (at the neck of the aneurysm), we have forced a correct segmentation by manually marking some voxels, as in the rightmost column on Figure 13.1. Thus, instead of computing the ϕ-Cheeger set of the image domain, we have computed the ϕ-Cheeger set of the image domain minus some manually selected voxels. On Figure 13.5 we display the results.

Real 3D MR image The second real 3D example is an edge linking experiment coming from a Magnetic Resonance image. This is a very low-resolution image, where the thin vessels have a width of one voxel. An edge detector correctly finds most of the vessels (in several different connected components). We show the best six local d_S-Cheeger sets of this image on Figure 13.7.

We have used the theory above in two different ways to segment some images, they amount to different choices of a metric integrand g. The first choice is the inverse of the gradient of the original image $g = \frac{1}{|\nabla I|}$ and the second choice is the distance function to the set of edge points deteced by a preprocessing of the image, that is $g = d_S$, where S is the set of edges of u. We label these two cases respectively *gradient Cheegers* and *distance Cheegers*. We observe that the evolution of PDE is much faster for distance Cheegers, and the result is less likely to miss parts of the image. On the other hand, gradient Cheegers are give smooth results after a long time of computation, and sometimes miss parts of the desired objects, or fail to break at holes.

13.3 Diffusion and colorization

Consider the anisotropic diffusion problem formulated as

$$\min_{u \in X} \frac{\|\pi_Z u - f\|^2}{2} + \lambda^{-1} J_\phi(u), \tag{13.1}$$

where π_Z is the orthogonal projection onto a set $Z \subset X$, and $f \in Z$. The regularizer $J_\phi(u) = \int_\Omega \phi(x, \nabla u(x))$ is defined so that the diffusion is constrained to the geometry (given by the level lines) extracted from a reference image I. We

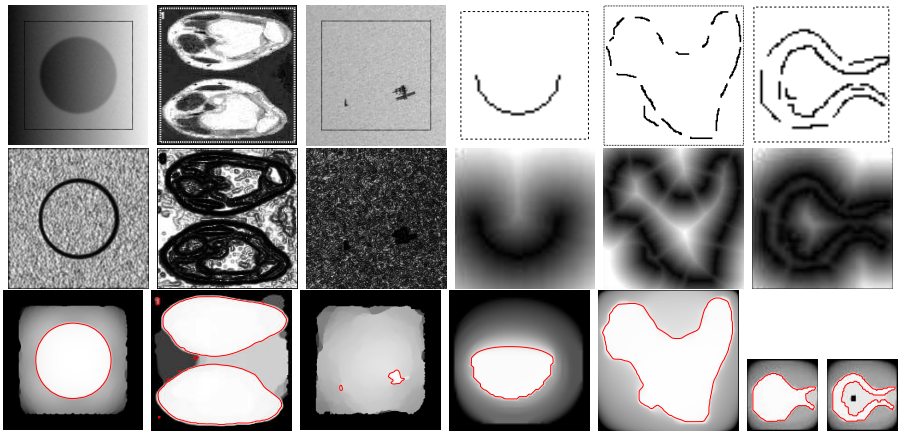


Figure 13.1: Geodesic active contours as g-Cheeger minimizers. The first row shows the images I to be processed. The second row shows the weights g used for each experiment (white is 1, black is 0), in the first two cases $g = (\sqrt{1+|\nabla(G*I)|^2})^{-1}$, for the third $g = 0.37(\sqrt{0.1+|\nabla(G*I)|^2})^{-1}$ and for the linking experiments $g = d_S$, the scaled distance function to the given edges. The third row shows the disjoint minimum g-Cheeger sets extracted from u (shown in the background), there are 1,7,2,1,1 and 1 sets respectively. The last linking experiment illustrates the effect of introducing a barrier in the initial domain (black square).

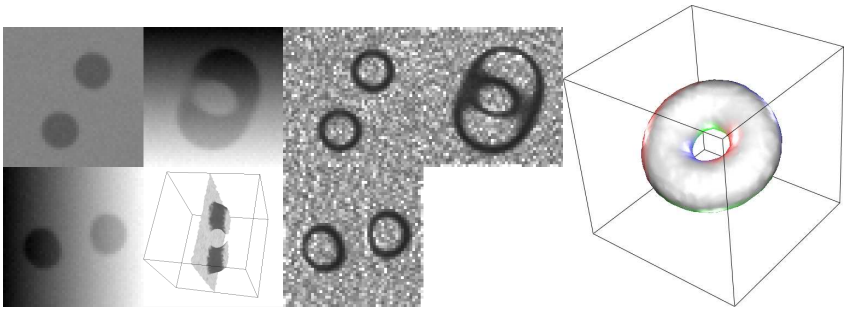


Figure 13.2: Pipeline for computing $\frac{1}{|\nabla I|}$-Cheeger sets, applied to a synthetic 3D image. From left to right: slices of the original image I, slices of the metric $g = \frac{1}{|\nabla I|}$, and $\frac{1}{|\nabla I|}$-Cheeger set of the image domain.

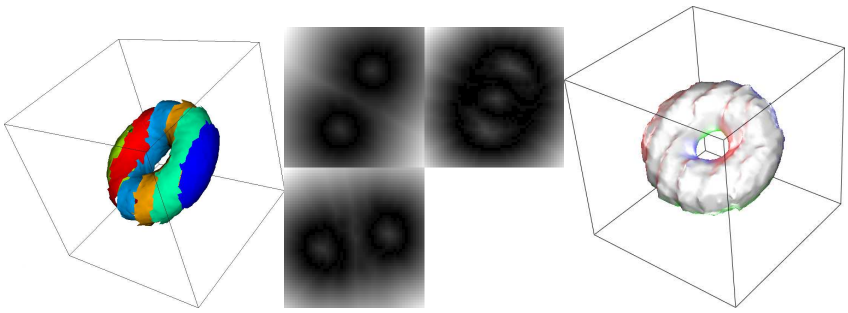


Figure 13.3: Pipeline for computing d_S-Cheeger sets, applied to the same synthetic image as in Figure 13.2. From left to right: detected 3D edges S, slices of the metric $g = d_S$, and d_S-Cheeger set of the image domain.

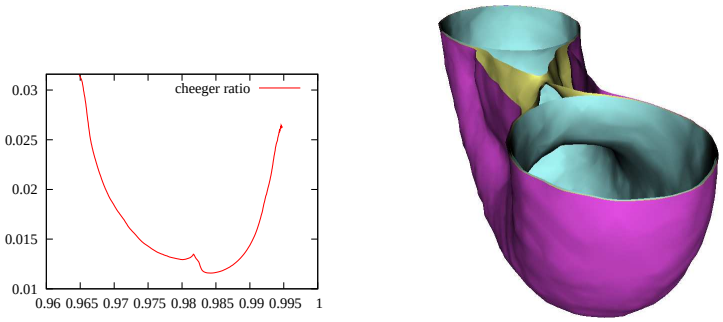


Figure 13.4: Left: Graph of the $\frac{1}{|\nabla I|}$-Cheeger ratio $F(t)$ for the input image "torus". See Right: Superposition of the three level sets shown, corresponding to the interesting points of $F(t)$ (the two minima and the cusp). To see the inner surfaces, the display is clipped near the central singularity. Notice that these level surfaces are all local minima of the classical geodesic snakes functional with an inflating force, for different weights of the inflating force.

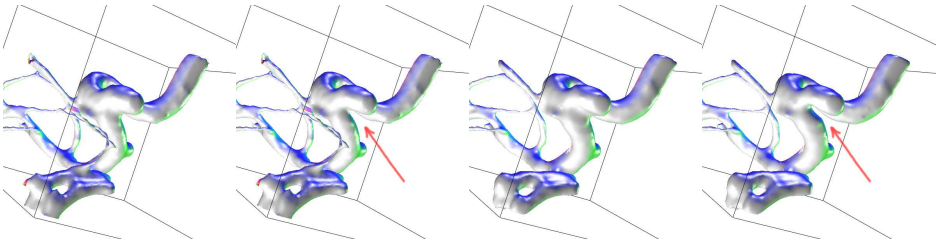


Figure 13.5: Computation of $\frac{1}{|\nabla I|}$-Cheeger sets of the CT image. From left to right: 1. d_S-Cheeger set of the whole image domain. 2. d_S-Cheeger set of the image domain minus some manually selected voxels at the neck of the aneurysm. 3. $\frac{1}{|\nabla I|}$-Cheeger set of the whole image domain. 4. $\frac{1}{|\nabla I|}$-Cheeger set of the image domain minus some manually selected voxels at the neck of the aneurysm.

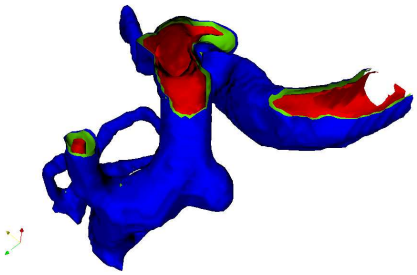


Figure 13.6: Three level surfaces of the solution u of $\frac{1}{|\nabla I|}$-total variation minimization. The central surface is the $\frac{1}{|\nabla I|}$-Cheeger set of the image domain according to this active-contours-like metric. The other two surfaces have a higher $\frac{1}{|\nabla I|}$-Cheeger ratio and appear as local extrema of an active contour with appropriate inflating force. In this figure, the image domain is split so that the innermost surfaces can be seen. Notice that the inner surface, having a higher level t, is farther than the others. This indicates the concentration of values around the maximum 1.

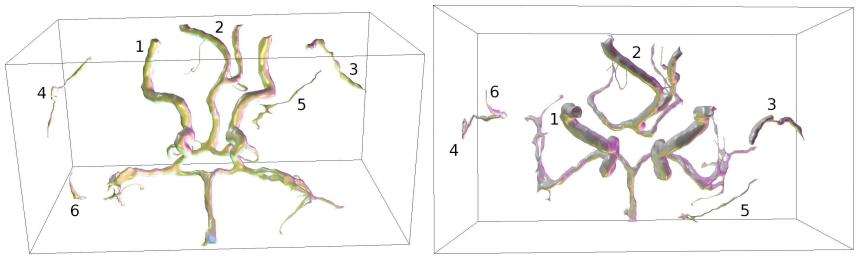


Figure 13.7: These two Figures display the best six local d_S-Cheeger sets of the MR image, labelled, and from different points of view.

define for example $\phi(x,\xi) = |A_x\xi|$ where A_x is a matrix that embodies knowledge about the boundaries of the objects in I. A common example in 2D corresponds to $A_x = V(x)^\perp \otimes V(x)^\perp$ with $V(x) = \frac{\nabla I(x)}{\sqrt{1+|\nabla I(x)|^2}}$. This example favors the diffusion along to the level lines of I. In low gradient (flat) zones the previous definition can be relaxed to allow diffusion across the level lines (as depicted in Figure 13.8) in a way inversely proportional to the modulus of the gradient. In that case, we may take $A_x = V(x)^\perp \otimes V(x)^\perp + \frac{1}{\sqrt{1+|\nabla I(x)|^2}} V(x) \otimes V(x)$, where $V(x)^\perp$ denotes the counterclockwise rotation of $V(x)$ of angle $\frac{\pi}{2}$. Notice that by the structure of A_x we could also take the clockwise rotation.

We will solve (13.1) by adapting the zoom algorithm proposed in [Cha04]. Observing that $\|\pi_Z u - f\| = \min_{w\in Z^\perp} \|u-(f+w)\|$, then (13.1) can be reformulated as

$$\min_{u\in X, w\in Z^\perp} \frac{\|u-f-w\|^2}{2} + \lambda^{-1} J_\phi(u), \tag{13.2}$$

which is solved by alternate minimization with respect to u and w. The first minimization is done by the algorithm described in Section 12.1.2: $u_n = (f - w_n) - \pi_{\mathcal{K}_\phi}(f - w_n)$, and the second one consists in a projection over $Z^\perp$: $w_{n+1} = \pi_{Z^\perp}(u_n - f)$.

13.3.1 Experiments (diffusion)

The scheme presented earlier for solving (13.2) can be applied in a variety of diffusion problems like: image colorization [LLW04], or to the interpolation of sparse height data in a digital elevation model [FLA+06]. In any of those cases, there are better algorithms to perform the task. For us they serve as an illustration of the proposed methods.

In the case of colorization and interpolation, Z is defined as $Z = \{\chi_\Gamma f : f \in X\}$ where $\Gamma \subseteq \Omega$ is a subdomain of the image where the values are known, the reference image $I : \Omega \to \mathbb{R}$ is used to compute the field $V(x)$ to guide the diffusion of these values. For the colorization experiment shown in Figure 13.9 the result is computed in YUV color space, where Y is the input luminance channel, and

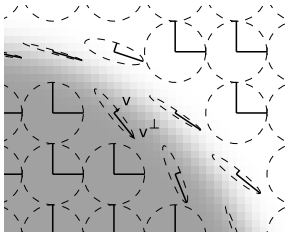


Figure 13.8: Anisotropic diffusion directions for a synthetic image (background). The plotted ellipses correspond to the tensor field $A_x = V(x)^\perp \otimes V(x)^\perp + \frac{1}{\sqrt{1+|\nabla I(x)|^2}} V(x) \otimes V(x)$ with $V(x) = \frac{\nabla I(x)}{\sqrt{1+|\nabla I(x)|^2}}$, the orientation and widths of the ellipses reflect the preferred direction of the diffusion.

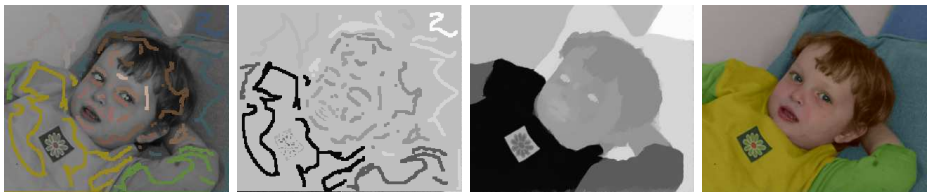


Figure 13.9: Colorization. From left to right, scribbled image, the known values of the U channel (20% of the image), diffusion in the U channel, and colorized result using the reference Y channel (not shown).

the chromatic channels U and V are interpolated with (13.1), where the field $V(x) = \frac{\nabla I(x)}{\sqrt{1+|\nabla I(x)|^2}}$ restricts the diffusion to the geometry of I.

The last example concerns the interpolation of urban digital elevation models (see Figure 13.10). In this case the datum f is known only at sparse locations, and it is provided by a stereo sub-pixel correlation algorithm [SAM08] (which also provides an estimation of the measure's variance Err). The reference image of the stereo pair is used as geometric constraint for the interpolation, and the variability Err is used to normalize the data fitting by adapting the spatial metric $h(i,j)^{1/2} = 1/Err(i,j)$. In Figure 13.10 we compare this method with the anisotropic minimal surface interpolation described in [FLA+06].

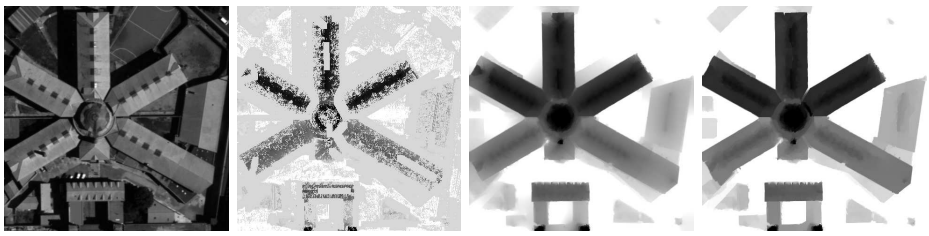


Figure 13.10: Disparity interpolation in an urban digital elevation model. From left to right, the reference image of the stereo pair, and the incomplete data set computed with [SAM08] (30% of the image) where each point's graylevel represents the height (darker is higher, mid gray is unknown). The interpolation obtained with the minimal surface interpolation [FLA$^+$06] (RMSE 0.239 when compared with the ground truth) and with the proposed algorithm (RMSE 0.190). The minimal surface model [FLA$^+$06] recovers the slanted surfaces better than total variation, however the latter is better at approximating the geometry near jumps.

14 Appropriate setting for Maximally Stable Extremal Regions

This Chapter contains an interpretation of MSER in terms of Finsler-Cheeger sets, when the Cheeger ratio is minimized restricted to the sets contained in the tree of shapes of the image. The metric is taken from the image contents. In the end, we prove affine invariance of the MSER features using elementary linear algebra.

14.1 Overview

Maximally Stable Extremal Regions of images were introduced in [MCUP04] in a graph-theoretic context. They are defined for *discrete and quantized* gray-scale images. If $Q_1 \subseteq \cdots \subseteq Q_n$ is a sequence of consecutive[1] upper or lower level sets of the image, then Q_i is called an *stable* region when $F(i) < \min(F(i-1), F(i+1))$, where

$$F(i) := \frac{|Q_{i+\delta}| - |Q_{i-\delta}|}{|Q_i|} \tag{14.1}$$

and δ is a fixed, arbitrary, small positive integer. This original definition was proposed on the grounds that it gives curves which are both meaningful and robust, but without further mathematical explanation. Here we give an alternative interpretation of these curves in a continuous setting, where they turn out to be the Cheeger sets of the image domain with an appropriate metric, restricted to the tree of shapes of the image. Then we recover the original definition of Matas et al. as a discrete approximation of the continuous version.

This approach overcomes three theoretical shortcomings of the original definition of stable regions. The first problem is an ambiguity in the definition, because the set of Q_i need not be a chain (that is, it may happen that $Q_1 \subseteq Q_3$ and $Q_2 \subseteq Q_3$ while $Q_1 \cup Q_2 = \emptyset$). The second problem is that the distance between the gray levels of the Q_i is not taken into account[2], so that the method is strictly contrast-invariant, even under highly non-linear contrast changes which create or remove visually obvious edges. The third problem is the seemingly arbitrary choice between upper or lower level sets. We will explain why each of these three issues does not arise as practical problems of the original implementation, which is quite robust.

14.2 Definition of MSER over an arbitrary tree

14.2.1 Trees of regions and shapes.

We want to define local extrema of a function which is defined on the set of level curves. Thus, we have to turn this set into a topological space. There

[1] that is, whose levels are contiguous in the quantized set of values

[2] this problem arises as a confusion on whether the i inside Q_i indexes gray levels or level sets within an inclusion chain

are three ways to do that, depending on the precise definition we use of "level curves": as boundaries of upper level sets, as boundaries of lower level sets or as *outer* boundaries of upper and lower level sets, where the orientation is taken with respect to an arbitrary fixed point $p_\infty \in \Omega$. We will not give the precise definition of these topological spaces, but we will describe its points and explain their main properties.

Definition 78. *If $A \subseteq \mathbb{R}^N$ and $x \in \mathbb{R}^N$, let us denote by $\mathrm{cc}(A, x)$ the connected component of A that contains x, if any, or the empty set otherwise. Let Ω be an N-dimensional rectangle and let $u : \Omega \to \mathbb{R}$. The* upper tree *of u is the set of connected components of upper level sets of u:*

$$\mathrm{ULT}(u) := \{\mathrm{cc}([u \geq \lambda], x) \ : \ x \in \Omega, \ \lambda \in \mathbb{R}\}$$

The lower tree *of u is the set of connected components of lower level sets of u:*

$$\mathrm{LLT}(u) := \{\mathrm{cc}([u < \lambda], x) \ : \ x \in \Omega, \ \lambda \in \mathbb{R}\}$$

The tree of shapes *of u is the set of saturations of the upper and lower trees:*

$$\mathrm{TOS}(u) := \{\mathrm{Sat}(S, p_\infty) \ : \ S \in \mathrm{ULT}(u) \cup \mathrm{LLT}(u)\}$$

where $\mathrm{Sat}(A, x) := \Omega \setminus \mathrm{cc}(\Omega \setminus A, x)$ and p_∞ is an arbitrary fixed point of Ω.

Each one of these three trees can be naturally endowed with a topological structure that turns it into a cell complex of dimension 1 without cycles, that is, a *tree*. This means two things: First, that the neighborhoods of these topological spaces are homeomorphic either to an open interval of $\mathbb{R}$ or to a disjoint union of $n \neq 2$ intervals of the form $[0, \epsilon)$ with the zeros glued. Second, that they have no sub-space homeomorphic to S^1. See Figures 14.1 and 14.2. The topologies of $\mathrm{ULT}(u)$ and $\mathrm{LLT}(u)$ are defined from that of Ω by taking the quotient by an equivalence relation. The topology of $\mathrm{TOS}(u)$ is defined by projection from $\mathrm{ULT}(u)$ and $\mathrm{LLT}(u)$ through Sat. In the smooth case when u is a Morse function, the (unrooted) Tree of Shapes is also called the Reeb Graph of Ω, and its topology can be defined as the quotient of Ω bye the equivalence relation $x \equiv y \iff u(x) = u(y)$. The full details of these constructions are described elsewhere [Ree46, BCM03], or on Part I of this thesis, Chapter 3.

The advantage of using TOS over either ULT or LLT is that the elements of TOS have no holes. This means that their boundaries can be represented by connected curves or surfaces. On the contrary, the boundaries of elements of ULT and LLT may have several connected components. For that, we believe that the tree of shapes is the appropriate framework for generalizing Matas' stable regions.

14.2.2 Continuous generalization of Stable Regions

Now we can define the continuous generalization of Stable Regions. We give three versions of the definition, one for each of the trees defined above.

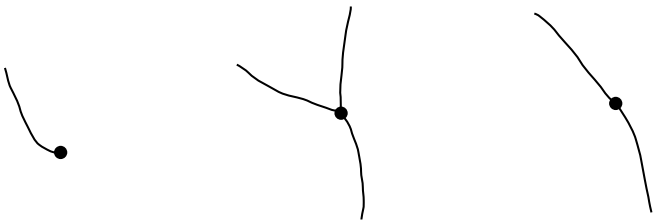


Figure 14.1: Some possible neighborhoods of the tree of shapes as a topological space. Each point of this space is a level set. The highlighted points represent, from left to right, a local extremum, a level curve through a saddle point, and the general case: a nonsingular level curve.

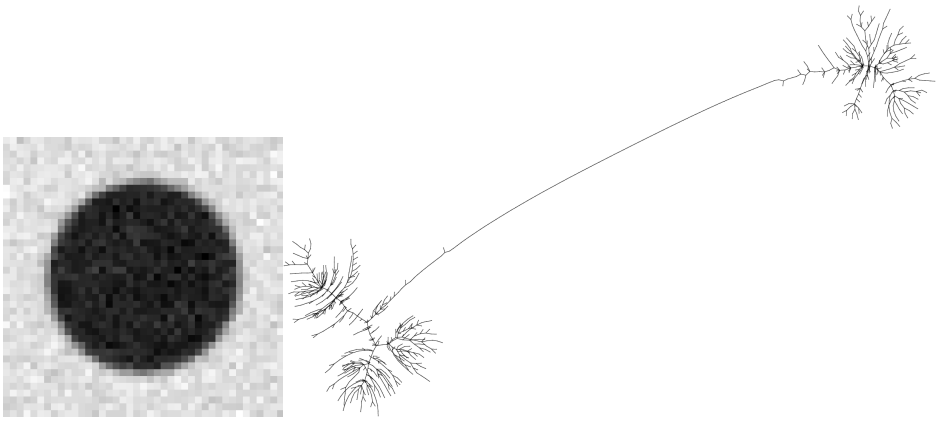


Figure 14.2: A synthetic image and its tree of shapes. The points on the large branch correspond to the level curves that surround the disk on the image. The two clusters of small branches correspond to the noise inside and outside the disk.

Definition 79. *Let $u : \Omega \to \mathbb{R}$ be an image and let $\mathcal{T}$ be any of the spaces* $\mathrm{ULT}(u)$, $\mathrm{LLT}(u)$, $\mathrm{TOS}(u)$. *We define the* Cheeger ratio *associated to this data as the following real-valued function on $\mathcal{T}$:*

$$F(E) = \frac{\int_{\partial E} g \, \mathrm{d}\mathcal{H}^{N-1}}{\int_E f \, \mathrm{d}\mathcal{H}^n}$$

where f and g are nonnegative real-valued functions on Ω. The local Cheeger level sets *of u are the local minimizers of $F(E)$.*

Remark 10. *The definition above makes sense because $\mathcal{T}$ is a topological space on which we can take neighborhoods of its points E. Furthermore, each E is a subset of Ω which has well-defined boundary ∂E, so that the integrals above are well-defined.*

Remark 11. *If* $E \in \mathrm{ULT}(u)$ *or* $E \in \mathrm{LLT}(u)$ *then* ∂E *may have several connected components. This case was not discussed on [MCUP04] and its treatment is ambiguous. On the other side, the construction of* $\mathrm{TOS}(u)$ *guarantees that the boundaries of its elements are connected.*

Figure 14.3: Comparison of local Cheeger sets restricted to ULT, LLT and TOS for a real 2D image.

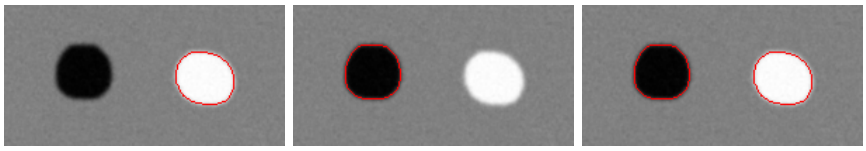


Figure 14.4: Comparison of local Cheeger sets restricted to ULT, LLT and TOS for a synthetic 2D image.

14.3 MSER are Finsler-Cheeger sets

14.3.1 Recall of the co-area formula.

Subtracting the areas enclosed by two contiguous level curves gives an approximation of their boundary length (see Figure 14.5). More precisely, it is an approximation of the length multiplied by the separation between the curves, which is $1/|\nabla u|$. This intuition is formalized by the following special case of the co-area formula [EG92a], for smooth $u \in \mathcal{C}^\infty$:

$$|u < b| - |u < a| = \int_a^b \left[\int_{\{u=\lambda\}} \frac{1}{|\nabla u|} \mathrm{d}\mathcal{H}^{N-1} \right] \mathrm{d}\lambda \tag{14.2}$$

which after the application of the mean-value Theorem becomes

$$|u < b| - |u < a| = (b-a) \int_{\{u=\lambda_m\}} \frac{1}{|\nabla u|} \mathrm{d}\mathcal{H}^{N-1} \tag{14.3}$$

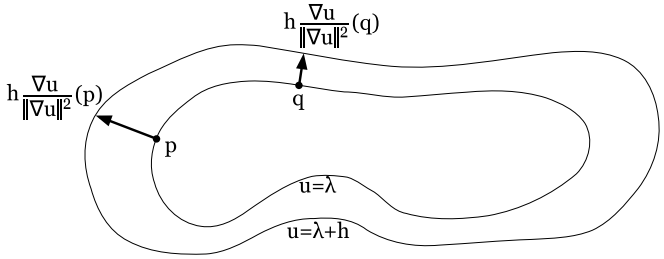


Figure 14.5: The separation between two nearby level curves of u is approximated to first order by $\frac{1}{|\nabla u|}$. The co-area formula (14.3) states that the area enclosed between them equals their length weighted by this separation.

for some $\lambda_m \in [a, b]$. The same relation holds for upper level sets, changing the order of a and b.

Differentiating equation (14.2) with respect to b we get

$$\frac{\partial}{\partial \lambda}|u < \lambda| = \int_{\{u=\lambda\}} \frac{1}{|\nabla u|} \mathrm{d}\mathcal{H}^{N-1} \tag{14.4}$$

which is an interesting relation. For instance, it allows to succinctly characterize local minimizers of the $\frac{1}{|\nabla I|}$-Cheeger ratio by the relation $\frac{\partial^2}{\partial \lambda^2}|u < \lambda| = 0$.

14.3.2 MSER are Cheegers

Equation (14.3) above bears a strong resemblance to the numerator on definition (14.1). Indeed, both expressions involve differences of areas enclosed by nearby level lines. We can put them into exact correspondence, by noting that the gray level λ is a monotone function $\lambda(i)$ of the indices i on (14.1). Thus, $|Q_i| = |u < \lambda(i)|$ and the Matas ratio can be written as

$$F(i) = (\lambda(i+\delta) - \lambda(i-\delta)) \, \frac{\int_{\{u=\lambda_m\}} \frac{1}{|\nabla u|} \mathrm{d}\mathcal{H}^{N-1}}{|u < \lambda(i)|} \tag{14.5}$$

for some $\lambda_m \in [\lambda(i-\delta), \lambda(i+\delta)]$.

This last expression could be used to approximate the Matas ratio $F(i)$, but we will not do that. Instead, we start from the $\frac{1}{|\nabla u|}$-weigthed Cheeger ratio $F(u; \lambda)$:

$$F(u; \lambda) = \frac{\int_{\{u=\lambda\}} \frac{1}{|\nabla u|} \mathrm{d}\mathcal{H}^{N-1}}{|u < \lambda|}. \tag{14.6}$$

and afterwards we observe that (14.5) can be understood as an approximation of (14.6) (and not the other way round).

In what sense is $F(i)$ an approximation of $F(u; \lambda(i))$? Well, notice that in an ideal case, all the possible gray levels appear and then $\lambda(i) = i$, and $\lambda(i +$

$\delta) - \lambda(i - \delta) = 2\delta$ is a constant factor that does not matter in the minimization. Then, taking te value of δ to be small enough, we can use the 14.4 to obtain the desired approximation.

This approximation is affected by two different types of error: the quantization of the gray levels and the discretization of the areas. On the figures we show how good (or bad) is this approximation, depending on these parameters. On the other side, notice that gray-level quantization plays an essential role in the approximation: in the common[3] extreme case when all the gray levels are different, we have that $|Q_{i+\delta}| - |Q_{i-\delta}| = 2\delta$, so that

$$F(i) = 2\delta \frac{\lambda(i+\delta) - \lambda(i-\delta)}{|Q_i|}.$$

This suggests that, from our point of view, a *better* definition of stable regions would involve weighting the Matas ratio by the difference of gray levels. In practice, it would make little difference for quantized images on the range $\{0, 255\}$. But nevertheless it is easy to construct synthetic images where the two methods would give different results.

14.4 Affine invariance

We will show that the Matas and Cheeger ratios given above are affine invariant. This means that for any invertible affine map A, we have $F(u \circ A; \lambda) = F(u; \lambda)$ for all λ. The denominators of these ratios are areas of level sets, which are always affine covariant:

$$\begin{aligned} |u \circ A < \lambda| &= |\{x \ : \ u(Ax) < \lambda\}| \\ &= \left|\left\{A^{-1}y \ : \ u(y) < \lambda\right\}\right| \\ &= \left|A^{-1} \ \cdot \ \{y \ : \ u(y) < \lambda\}\right| \\ &= |A|^{-1} \ \cdot \ |u < \lambda| \end{aligned}$$

Now it remains to prove the covariance of the numerators.

The proof for the case of stable regions is immediate and was given by Matas et al., thus motivating the use of these objects for stereo matching. Here we give two proofs for the affine invariance of $\frac{1}{|\nabla u|}$-weighted Cheeger ratios. The first proof is a direct proof using parametrized curves, which is cumbersome to extend to higher dimensions. The second proof is an informal argument using the Dirac delta, which works for all dimensions. The different proofs provide different insights to the property of affine invariance.

Proposition 80. *MSER are affine invariant.*

Proof. The numerator on equation (14.1) is a difference of areas of level sets, which are affine covariant. □

[3] for example, with floating-point images

Proposition 81. *Let us define*

$$P(u,\lambda) = \int_{\{u=\lambda\}} \frac{1}{|\nabla u|} \mathrm{d}\mathcal{H}^{N-1}$$

for functions $u : \mathbb{R}^N \to \mathbb{R}$. Then, for any invertible $N \times N$ matrix A we have $P(u \circ A, \lambda) = |A|^{-1} P(u,\lambda)$.

We will prove this proposition first for the case when $N = 2$ and the curve $\{u = \lambda\}$ is smooth. Afterwards, we will give a sketch of the proof for the smooth N-dimensional case, and an informal argument for the general case. The two-dimensional case is immediate after some linear algebra computation developed in the appendix.

Proof of proposition 81 (using parametrizations, 2D-only). With appropriate smoothness, the integral

$$P(u,\lambda) = \int_{\{u=\lambda\}} \frac{1}{|\nabla u|} \mathrm{d}\mathcal{H}^{1}$$

is a line integral which can be evaluated using a parametrization $\gamma : I \to \mathbb{R}^2$ of the level curve $\{u = \lambda\}$,

$$P(u,\lambda) = \int_I \frac{|\gamma'(t)|}{|\nabla u(\gamma(t))|} \mathrm{d}t.$$

To evaluate $P(u \circ A, \lambda)$ we can use the parametrization $\eta = A^{-1} \circ \gamma$ of the level curve $\{u \circ A = \lambda\}$ and apply the chain rule:

$$P(u \circ A, \lambda) = \int_I \frac{|\eta'|}{|\nabla u \circ A|} \mathrm{d}t = \int_I \frac{\left|A^{-1}\gamma'\right|}{|A^{\mathsf{T}} \nabla u|} \mathrm{d}t.$$

Now, Lemma 83 can be used on this last expression to obtain the desired result. □

Proof of proposition 81 (sketch, using parametrizations, ND). To compute the weighted perimeter of the hypersurface $\{u = \lambda\}$ we use a parametrization

$$\gamma : I^{N-1} \to \mathbb{R}^N$$

such that $u \circ \gamma = \lambda$. Then

$$P(u,\lambda) = \int_{I^{N-1}} \frac{|K(\gamma_1, \ldots, \gamma_{N-1})|}{|\nabla u|} \mathrm{d}t_1 \cdots \mathrm{d}t_{N-1}$$

where the γ_i are the tangent vectors to the parametrization and the notation K is explained in the appendix. Now, to evaluate $P(u \circ A, \lambda)$ we can use the parametrization $\eta = A^{-1} \circ \gamma$ of the level surface $\{u \circ A = \lambda\}$ and apply the chain rule:

$$P(u \circ A, \lambda) = \int_{I^{N-1}} \frac{\left|K(A^{-1}\gamma_1, \ldots, A^{-1}\gamma_{N-1})\right|}{|A^{\mathsf{T}} \nabla u|} \mathrm{d}t_1 \cdots \mathrm{d}t_{N-1}$$

Now, Lemma 84 can be used on this last expression to obtain the desired result. □

The following proof offers additional insight.

Proof of proposition 81 (using deltas). It suffices to write the weighted perimeters as an integral over the whole space using the delta function

$$P(u;\lambda) = \int_{\{u=\lambda\}} \frac{1}{|\nabla u|} \mathrm{d}l = \int_{\mathbb{R}^N} \delta\left(u(x) - \lambda\right) \mathrm{d}x \tag{14.7}$$

and verify that $P(u \circ A; \lambda) = |A|^{-1} P(u;\lambda)$ applying the change of variables $Ax = y \implies |A|\mathrm{d}x = \mathrm{d}y$. Note that the choice of metric $g = \frac{1}{|\nabla u|}$ is special: general metrics will not work. In particular, Euclidean Cheeger sets are not affine invariant. □

Remark 12. *The representation given by equation* (14.7) *is the higher-dimensional version of the usual formula for one-dimensional deltas:*

$$\delta(f(x)) = \sum_{z \in f^{-1}(0)} \frac{\delta(x-z)}{f'(z)}.$$

14.4.1 Linear algebra computations

Lemma 82. *Let* $J = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$. *Then for any invertible* 2×2 *matrix* A

$$A^{-1} = \frac{1}{|A|} J^{\mathsf{T}} A^{\mathsf{T}} J$$

where the T *denotes transposition.*

Proof.

$$\left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)\left(\begin{smallmatrix} a & c \\ b & d \end{smallmatrix}\right)\left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)\left(\begin{smallmatrix} c & -a \\ d & -b \end{smallmatrix}\right) = \left(\begin{smallmatrix} d & -b \\ -c & a \end{smallmatrix}\right)$$

and this last matrix is the adjoint of $A = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, which is the inverse times the determinant. □

Lemma 83. *Let* x *and* $p \neq 0$ *be orthogonal vectors in the plane. Then for any invertible* 2×2 *matrix* A

$$\frac{\left|A^{-1}x\right|}{|A^{\mathsf{T}}p|} = \frac{1}{|A|}\frac{|x|}{|p|}.$$

Proof. The map J is an isometry and the vectors Jx and p are parallel, thus

$$Jx = \pm \frac{|x|}{|p|} p.$$

Now, using the previous Lemma

$$\left|A^{-1}x\right| = \frac{1}{|A|}\left|J^{\mathsf{T}}A^{\mathsf{T}}Jx\right| = \frac{1}{|A|}\left|A^{\mathsf{T}}Jx\right| = \frac{1}{|A|}\frac{|x|}{|p|}\left|A^{\mathsf{T}}p\right|$$

and the result follows. □

On the proof of affine invariance above we use a notation that generalizes the 90° rotation in $\mathbb{R}^2$ and the vector cross product in $\mathbb{R}^3$. If $x_1, \ldots, x_{n-1} \in \mathbb{R}^n$, let us denote by $K(x_1, \ldots, x_{n-1})$ the vector uniquely defined by the following three properties. Its modulus is the $(n-1)$-dimensional volume of the parallelepiped spanned by the vectors $x_1, \ldots, x_{n-1}$; it is perpendicular to each of these vectors; its sign is such that when the arguments are linearly independent, the tuple $(x_1, \ldots, x_{n-1}, K(x_1, \ldots, x_{n-1}))$ is an oriented basis of $\mathbb{R}^n$. The following properties are readily checked:

1. $\langle K(x_1, \ldots, x_{n-1}), x_n \rangle = \det(x_1, \ldots, x_n)$
2. $\det(Ax_1, \ldots, Ax_n) = |A| \det(x_1, \ldots, x_n)$, this says that $|AX| = |A||X|$
3. $\left\langle A^\mathsf{T} K(Ax_1, \ldots, Ax_{n-1}), v \right\rangle = |A| \langle K(x_1, \ldots, x_{n-1}), v \rangle$, combining properties 1 and 2.
4. $A^\mathsf{T} K(Ax_1, \ldots, Ax_{n-1}) = |A| K(x_1, \ldots, x_{n-1})$, because the previous formula holds for all values of v.

As a consequence of this last property, we obtain the n-dimensional generalization of Lemma 83:

Lemma 84. *Let $p \in \mathbb{R}^n \setminus \{0\}$ be orthogonal to the vectors $x_1, \ldots, x_{n-1}$. Then for any invertible $n \times n$ matrix A*

$$\frac{\left|K(A^{-1}x_1, \ldots, A^{-1}x_{n-1})\right|}{|A^\mathsf{T} p|} = \frac{1}{|A|} \frac{|K(x_1, \ldots, x_{n-1})|}{|p|}.$$

Proof. Writing relation 4 above for A^{-1} we get

$$K(A^{-1}x_1, \ldots, A^{-1}x_{n-1}) = \frac{1}{|A|} A^\mathsf{T} K(x_1, \ldots, x_{n-1})$$

and taking norms,

$$\frac{\left|K(A^{-1}x_1, \ldots, A^{-1}x_{n-1})\right|}{|A^\mathsf{T} K(x_1, \ldots, x_{n-1})|} = \frac{1}{|A|}. \tag{14.8}$$

Now, since p is parallel to $K(x_1, \ldots, x_{n-1})$, we have

$$K(x_1, \ldots, x_{n-1}) = \pm \frac{|K(x_1, \ldots, x_{n-1})|}{|p|} p$$

which, subtituted into equation 14.8, gives the desired result. □

Part IV

Appendixes

15 Conclusion

15.1 Overview of proposed contributions

The contributions proposed in this thesis have been freely interleaved with previous work by others. Here we recall the original contributions of this thesis, pointing to the place where each contribution is exposed.

- Joining upper and lower trees in general (Chapter 3)
- Algorithm for construction of discrete 3D tree of shapes (Chapter 4)
- Implementation of 3D grain filters (Section 5.1)
- Implementation of a 3D image visualization method (Section 5.2)
- A tool for RGB histogram analysis (Section 5.3)

- Semicontinuously and topologically consistent Marching Cubes (section 7.1)
- Mumford-Shah segmentation of functions defined on surfaces (Section 7.3)
- Robust 3D edge detector (Chapter 8)
- A generalization of edge detection by Helmholtz principle (Section 8.2)
- Tool for 3D image visualization and surface editing (Section 5.2)
- Exclusion principle for edge detection (Section 9.1)

- Relationship of Finsler-Cheeger sets with anisotropic TV (Chapter 11)
- Numerical computation of Finsler-Cheeger sets (Chapter 12)
- Application of Finsler-Cheeger sets to segmentation, surface joining and colorization (Chapter 13)
- Maximal stable extremal regions are local Finsler-Cheeger sets (Section 14.3
- Algebraic proof of affine invariance (section 14.4)

15.2 Future work

Some themes of this thesis are open issues worth of further study. Some of this research is already being developed in the form of submissions to conferences.

- tree of shapes of a video conditioned by optical flow
- quantitative comparison of edge detectors
- quantitative comparison of surface joiners
- definition and implementation of multi-scale tree of shapes
- study of topological persistence of structures
- further applications of the tree of shapes to color densities
- development of monocular depth estimation
- development of object detection based on the new invariants

16 Published Work

Most of the research reported above has been published in peer-reviewed journals and conferences.

- **Constructing the Tree of Shapes of an Image by Fusion of the Trees of Connected Components of Upper and Lower Level Sets**
  V. Caselles, E. Meinhardt and P. Monasse
  Positivity, Vol. 12, Num. 1, pp. 55–73, 2008

- **Edge Detection by Selection of Pieces of Level Lines**
  E. Meinhardt
  International Conference on Image Processing 2008, October 2008, San Diego

- **3D Edge Detection By Selection of Level Surface Patches**
  E. Meinhardt, E. Zacur, A.F. Frangi and V. Caselles
  Journal of Mathematical Imaging and Vision,
  Vol. 32, Num. 1, pp. 1–16, 2009

- **Anisotropic Cheeger Sets and Applications**
  V. Caselles, G. Facciolo and E. Meinhardt
  SIAM Journal on Imaging Sciences,
  Vol. 2, Num. 4, pp. 1211–1254, 2009

Other work has been already submitted and is under review:

- **Relative Depth Estimation from Monocular Video**
  E. Meinhardt, V .Caselles
  Submitted to *International Conference on Image Processing 2011*

- **A Robust Pipeline for Logo Detection**
  E. Meinhardt, C. Constantinopoulos, V. Caselles
  Submitted to *International Conference on Multimedia and Expo 2011*

- **On Affine Invariant Descriptors Related to SIFT**
  R. Sadek, C. Constantinopoulos, E. Meinhardt, C. Ballester, V. Caselles
  Submitted to *SIAM Journal on Imaging Sciences* on 2010

Bibliography

[AB94] M. Amar and G. Belletini. A notion of total variation depending on a metric with discontinuous coefficients. In *Annales de l'Institut Henri Poincaré. Analyse non linéaire*, volume 11, pages 91–133. Elsevier, 1994.

[ABCM00] F. Andreu, C. Ballester, V. Caselles, and J. Mazon. Minimizing total variation flow. *Comptes Rendus de l'Academie des Sciences-Serie I-Mathematique*, 331(11):867–872, 2000.

[AC03] P. Arbeláez and L. Cohen. Path variation and image segmentation. *Proc. EMMCVPR-03*, 2003.

[AC04] P. Arbeláez and L. Cohen. Energy partitions and image segmentation. *J. Math. Imaging Vis.*, 2004.

[AC09] F. Alter and V. Caselles. Uniqueness of the Cheeger set of a convex body. *Nonlinear Analysis: Theory, Methods & Applications*, 70(1):32–44, 2009.

[ACC05a] F. Alter, V. Caselles, and A. Chambolle. A characterization of convex calibrable sets in. *Mathematische Annalen*, 332(2):329–366, 2005.

[ACC05b] F. Alter, V. Caselles, and A. Chambolle. Evolution of characteristic functions of convex sets in the plane by the minimizing total variation flow. *Interfaces Free Bound*, 7(1):29–53, 2005.

[ACHR06] A. Almansa, V. Caselles, G. Haro, and B. Rougé. Restoration and zoom of irregularly sampled, blurred, and noisy images by accurate total variation minimization with local constraints. *Multiscale Modeling & Simulation*, 5(1):235–272, 2006.

[ACMM01] L. Ambrosio, V. Caselles, S. Masnou, and J.-M. Morel. Connected components of sets of finite perimeter and applications to image processing. *Journal of the European Mathematical Society*, 3:39–92, 2001. 10.1007/PL00011302.

[ADPS07] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sánchez. Symmetrical dense optical flow estimation with occlusions detection. *International Journal of Computer Vision*, 75(3):371–385, 2007.

[AF87] N. Ayache and B. Faverjon. Efficient registration of stereo images by matching graph descriptions of edge segments. *Int. J. Comp. Vis.*, 1(2):107–131, 1987.

[AFP00] L. Ambrosio, N. Fusco, and D. Pallara. *Functions of bounded variation and free discontinuity problems*. Oxford University Press, USA, 2000.

[AGLM93] L. Alvarez, F. Guichard, P. Lions, and J. Morel. Axioms and fundamental equations of image processing. *Archive for Rational Mechanics and Analysis*, 123(3):199–257, 1993.

[AGM99] L. Alvarez, Y. Gousseau, and J.-M. Morel. The size of objects in natural and artificial images. *Adv. Imaging Electron. Phys.*, 111:167–242, 1999.

[Anz83] G. Anzellotti. Pairings between measures and bounded functions and compensated compactness. *Annali di Matematica Pura ed Applicata*, 135(1):293–318, 1983.

[AS65] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*. Dover, 1965.

[AVCM04] F. Andreu-Vaillo, V. Caselles, and J. Mazón. *Parabolic quasilinear equations minimizing linear growth functionals*. Birkhäuser, 2004.

[Ban67] T. Banchoff. Critical points and curvature for embedded polyhedra. *Journal of Differential Geometry*, 1:245–256, 1967.

[Bar02] D. Barash. Fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(6):844–847, 2002.

[BBF99] G. Bellettini, G. Bouchitté, and I. Fragalà. BV Functions with Respect to a Measure and Relaxation of Metric Integral Functionals. *Journal of convex analysis*, 6(2):349–366, 1999.

[BBG96] S. Bose, K. Biswas, and S. Gupta. Model based object recognition: the role of affine invariants. *Artificial intelligence in engineering*, 10(3):227–234, 1996.

[BBM09] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *Computer Vision and Pattern Recognition*, pages 41–48. IEEE, 2009.

[BBPW04] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *Computer Vision-ECCV 2004*, pages 25–36, 2004.

[BCC07] G. Buttazzo, G. Carlier, and M. Comte. On the selection of maximal Cheeger sets. *Differential and integral equations*, 20(9):991–1004, 2007.

[BCIG07] C. Ballester, V. Caselles, L. Igual, and L. Garrido. Level lines selection with variational models for segmentation and encoding. *Journal of Mathematical Imaging and Vision*, 27(1):5–27, 2007.

[BCM03] C. Ballester, V. Caselles, and P. Monasse. The tree of shapes of an image. *ESAIM: Control, Optim. Calc. Variations*, 9:1–18, 2003.

[BK03] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. *Proc. 9th Int. Conf. Comp. Vis.*, pages 26–33, 2003.

[BL79] S. Beucher and C. Lantuejoul. Use of watersheds in contour detection. *Proc. Int. Workshop Image Process.*, 1979.

[Bol98] B. Bollobás. *Modern graph theory*. Springer Verlag, 1998.

[Bré73] H. Brézis. *Opérateurs maximaux monotones*. North-Holland Amsterdam, 1973.

[Bro92] L. Brown. A Survey of Image Registration Techniques. *ACM Computing Surveys*, 24:325–376, 1992.

[Bro93] R. Brooks. Spectral Geometry and the Cheeger constant. In *Expanding graphs: proceedings of a DIMACS workshop, May 11-14, 1992*, page 5. Amer Mathematical Society, 1993.

[Bru] J. Bruce. CMVision software library.

[BTVG06] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proc. of the European Conf. on Computer Vision*, pages 404–417, 2006.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Machine Intellicence*, 8(6):679–698, 1986.

[Cay59] A. Cayley. On contour and slope lines. *The Philosophical magazine*, 1859.

[CCA92] I. Cohen, L. Cohen, and N. Ayache. Using deformable surfaces to segment 3-D images and infer differential structures. *CVGIP: Image Understanding*, 56(2):242–263, 1992.

[CCN07] V. Caselles, A. Chambolle, and M. Novaga. Uniqueness of the Cheeger set of a convex body. *Pacific J. Math.*, 232(1):77–90, 2007.

[CCN09] V. Caselles, A. Chambolle, and M. Novaga. Some remarks on uniqueness and regularity of Cheeger sets. *Rend. Sem. Mat. Univ. Padova*, 2009.

[CDD06] H. Carr, B. Duffy, and B. Denby. On Histograms and Isosurface Statistics. *IEEE Trans. Vis. Comp. Graph.*, 12(5):1259–1266, 2006.

[CE05] T. Chan and S. Esedoglu. Aspects of total variation regularized L1 function approximation. *Siam J. Appl. Math*, 65(5):1817–1837, 2005.

[CFM09] V. Caselles, G. Facciolo, and E. Meinhardt. Anisotropic Cheeger sets and applications. *SIAM Journal on Imaging Sciences*, 2:1211, 2009.

[CGI05] V. Caselles, L. Garrido, and L. Igual. A contrast invariant approach to motion estimation. *Scale Space and PDE Methods in Computer Vision*, pages 242–253, 2005.

[Cha04] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1):89–97, 2004.

[Che70] J. Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. *Problems in analysis*, pages 195–199, 1970.

[Cho73] G. Choquet. *Cours de Topologie*. Dunod, 1973.

[Chu97] F. Chung. *Spectral graph theory*. American Mathematical Society, 1997.

[CKF03] J. Cox, D. B. Karron, and N. Ferdous. Topological zone organization of scalar volume data. *J. Math. Imaging Vis.*, 18(2):95–117, 2003.

[CKS97] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic Active Contours. *Int. J. Comp. Vis.*, 22(1):61–79, 1997.

[CLLR05] Y. Chiang, T. Lenz, X. Lu, and G. Rote. Simple and optimal output-sensitive construction of contour trees using monotone paths. *Computational Geometry*, 30(2):165–195, 2005.

[CLM+08] F. Cao, J. Lisani, J. Morel, P. Musé, and F. Sur. A theory of shape identification, volume 1948 of Lecture Notes in Mathematics. *Springer*, 2(4):7, 2008.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT, 2nd edition, 2001.

[CM02] V. Caselles and P. Monasse. Grain Filters. *Journal of Mathematical Imaging and Vision*, 17(3):249–270, 2002.

[CM07] H. Chang and J. Moura. Classification by cheeger constant regularization. In *ICIP 2007*, volume 2. IEEE, 2007.

[CM10] V. Caselles and P. Monasse. *Geometric Description of Images as Topographic Maps*. Lecture Notes in Mathematics. Springer, 2010.

[CMS05] F. Cao, P. Musé, and F. Sur. Extracting Meaningful Curves from Images. *J. Math. Imaging Vis.*, 22(2):159–181, 2005.

[Coh91] L. Cohen. On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218, 1991.

[CRBC07] J. Cardelino, G. Randall, M. Bertalmio, and V. Caselles. Region based segmentation using the tree of shapes. In *Image Processing, 2006 IEEE International Conference on*, pages 2421–2424. IEEE, 2007.

[CSA03] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Comput. Geom.: Theory Appl.*, 24(2):75–94, 2003.

[CV01] T. Chan and L. Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001.

[Dav75] L. Davis. A Survey of Edge Detection Techniques. *Comp. Grah. Image Process.*, 4:248–270, 1975.

[Der87] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *Int. J. Comp. Vis.*, 1(2):167–187, 1987.

[DMM00] A. Desolneux, L. Moisan, and J. Morel. Meaningful Alignments. *Int. J. Comp. Vis.*, 40(1):7–23, 2000.

[DMM01] A. Desolneux, L. Moisan, and J.-M. Morel. Edge Detection by Helmholtz Principle. *J. Math. Imaging Vis.*, 14(3):271–284, 2001.

[DMM03] A. Desolneux, L. Moisan, and J. Morel. Variational snake theory. *Geometric Level Set Methods in Imaging, Vision, and Graphics. Springer Verlag*, 2003.

[DMM04] A. Desolneux, L. Moisan, and J.-M. Morel. *Seeing, Thinking and Knowing*, chapter Gestalt theory and computer vision, pages 71–101. Springer, 2004.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[EG92a] L. Evans and R. Gariepy. *Measure theory and fine properties of functions*. CRC, 1992.

[EG92b] L. Evans and R. Gariepy. Measure theory and fine properties of functions. *Studies in Advanced Mathematics, CRC Press, Boca Raton*, 1992.

[EM90] H. Edelsbrunner and E. Muecke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 1990.

[FLA+06] G. Facciolo, F. Lecumberry, A. Almansa, A. Pardo, V. Caselles, and B. Rougé. Constrained anisotropic diffusion and some applications. In *British Machine Vision Conference*, volume 3, pages 1049–1058, Edinburgh, Scotland, September 2006.

[FP03] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[FtHRKV92] L. Florack, B. ter Haar Romeny, J. Koenderink, and M. Viergever. Scale and the differential structure of images. *Image and Vision Computing*, 10(6):376–388, 1992.

[G+02] M. Galassi et al. *GNU scientific library*. Network Theory Ltd., 2002.

[GH91] W. Grimson and D. Huttenlocher. On the verification of hypothesized matches in model-based recognition. *IEEE Trans. Pattern Anal. Machine Intellicence*, 13(12):1201–1213, 1991.

[Giu84] E. Giusti. *Minimal surfaces and functions of bounded variation*. Birkhauser, 1984.

[Gri06] D. Grieser. The first eigenvalue of the Laplacian, isoperimetric constants, and the max flow min cut theorem. *Archiv der Mathematik*, 87(1):75–85, 2006.

[GU89] D. Gordon and J. Udupa. Fast surface tracking in three-dimensional binary images. *Comp. Vis., Graph, Image Process.*, 45:196–214, 1989.

[Hal74] P. Halmos. *Measure theory*. Springer, 1974.

[Har84] R. Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Trans. Pattern Anal. Machine Intellicence*, 6(1):58–68, 1984.

[Her98] G. T. Herman. *Geometry of Digital Spaces*. Birkhauser, 1998.

[HLF+97] J. Hsieh, H. Liao, K. Fan, M. Ko, and Y. Hung. Image registration using a new edge-based approach. *Computer Vision and Image Understanding*, 67(2):112–130, 1997.

[HLO99] L. Hsu, M. Loew, and J. Ostuni. Automated registration of brain images using edge and surfacefeatures. *IEEE Eng. Med. Biol. Mag.*, 18(6):40–47, 1999.

[HS85] R. Haralick and L. Shapiro. Image segmentation techniques. *Comp. Vis., Graph, Image Process.*, 29:100–132, 1985.

[HWL83] R. Haralick, L. Watson, and T. Laffey. The topographic primal sketch. *Int. J. Robot. Res.*, 2(1):50–72, 1983.

[Igu06] L. Igual. *Image segmentation and compression using the tree of shapes of an image. Motion estimation.* PhD thesis, Ph. D. Thesis, Universitat Pompeu Fabra, 2006.

[JH02] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, 2002.

[JJH06] B. Jeon, Y. Jung, and K. Hong. Image segmentation by unsupervised sparse clustering. *Pattern Recognition Letters*, 27(14):1650–1664, 2006.

[Jul59] B. Julesz. A method of coding TV signals based on edge detection. *Bell Sys. Technol.*, 38(4):1001–1020, 1959.

[Kan79] G. Kanizsa. *Organization in vision: Essays on Gestalt perception.* Praeger Publishers, 1979.

[KGC10] M. Kalmoun, L. Garrido, and V. Caselles. Multilevel optimization as computational methods for dense optical flow. *preprint*, 2010.

[Kir71] R. Kirsch. Computer determination of the constituent structure of biological images. *Comp. Biomed. Res.*, 4(3):315–28, 1971.

[KK89] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(12):7–15, 1989.

[KKO+96] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi. Conformal curvature flows: From phase transitions to active vision. *Arch. Rat. Mech. Anal.*, 134(3):275–301, 1996.

[KLM94] G. Koepfler, C. Lopez, and J.-M. Morel. A Multiscale Algorithm for Image Segmentation by Variational Method. *SIAM J. Numer. Anal.*, 31(1):282–299, 1994.

[KLR06] B. Kawohl and T. Lachand Robert. Characterization of Cheeger sets for convex subsets of the plane. *Pacific journal of mathematics*, 225(1):103, 2006.

[KMS00] R. Kimmel, R. Malladi, and N. Sochen. Images as embedded maps and minimal surfaces: movies, color, texture, and volumetric medical images. *International Journal of Computer Vision*, 39(2):111–129, 2000.

[Knu73] D. Knuth. *Fundamental algorithms*, volume 1. Addison-Wesley Reading, MA, 1973.

[Kro50] A. Kronrod. On functions of two variables. *Uspehi Mathematical Sciences*, 1950.

[Kur66] K. Kuratowski. Topology, I, II. *New York*, 1966.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. Comp. Vis.*, 1(4):321–331, 1988.

[LC87] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Proc. 14th Annu. Conf. Comp. Graph. Interact. Tech.*, pages 163–169, 1987.

[LLW04] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 689–694, New York, NY, USA, 2004. ACM.

[LMMM03] J. Lisani, L. Moisan, P. Monasse, and J.-M. Morel. On the Theory of Planar Shape. *Multiscale Model. Simul.*, 1:1, 2003.

[Low85] D. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, 1985.

[Low99] D. Lowe. Object recognition from local scale-invariant features. In *iccv*, page 1150. Published by the IEEE Computer Society, 1999.

[LPR93] C.-N. Lee, T. Poston, and A. Rosenfeld. Holes and genus of 2d and 3d digital images. *CVGIP: Graph. Models Image Process.*, 55(1):20–47, 1993.

[LW88] Y. Lamdan and H. Wolfson. Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In *Second International Conference on Computer Vision*, page 238. IEEE Computer Society Press, 1988.

[LY02] E. Landis and I. Yaglom. Remembering a.s. kronrod. *available online*, 2002.

[MAK96] F. Mokhtarian, S. Abbasi, and J. Kittler. Robust and efficient shape indexing through curvature scale space. *Proc. Br. Mach. Vis. Conf.*, pages 53–62, 1996.

[Mar82] D. Marr. *Vision*. W. H. Freeman San Francisco, 1982.

[Max70] J. C. Maxwell. On hills and dales. *The Philosophical magazine*, 1870.

[MCUP04] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[MG00] P. Monasse and F. Guichard. Fast computation of a contrast-invariant image representation. *IEEE Trans. Image Process.*, 9:860–872, 2000.

[MH80] D. Marr and E. Hildreth. Theory of Edge Detection. *Proc. R. Soc. Lond.*, 207(1167):187–217, 1980.

[Mol05] J. Moll. The anisotropic total variation flow. *Mathematische Annalen*, 332(1):177–218, 2005.

[Mon00] P. Monasse. *Contrast Invariant Representation of Digital Images and Application to Registration*. PhD thesis, Université Paris IX-Dauphine, jun 2000.

[MS88] D. Mumford and J. Shah. Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems. *Commun. Pure App. Math.*, 42(5):577–685, 1988.

[MS95] J. Morel and S. Solimini. *Variational methods in image segmentation*. Birkhauser, 1995.

[MS04] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[MS05] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[MSV95] R. Malladi, J. Sethian, and B. Vemuri. Shape Modeling with Front Propagation: A Level Set Approach. *IEEE Trans. Pattern Anal. Machine Intellicence*, page 158175, 1995.

[Mum94] D. Mumford. Bayesian rationale for energy functionals. In *In Geometry-driven diffusion in Computer Vision*. Citeseer, 1994.

[MY09] J. Morel and G. Yu. ASIFT: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[MZFC09] E. Meinhardt, E. Zacur, A. Frangi, and V. Caselles. 3d edge detection by selection of level surface patches. *Journal of Mathematical Imaging and Vision*, 34(1):1–16, 2009.

[NA02] R. Nelson and J. Aloimonos. Obstacle Avoidance Using Flow Field Divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:1102–1106, 2002.

[NC06] L. Najman and M. Couprie. Building the component tree in quasi-linear time. *Image Processing, IEEE Transactions on*, 15(11):3531–3539, 2006.

[Nie03] G. Nielson. On marching cubes. *Visualization and Computer Graphics, IEEE Transactions on*, 9(3):283–297, 2003.

[OFL07] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[PB10] N. Papadakis and A. Bugeau. Tracking with occlusions via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[PCA99] X. Pennec, P. Cachier, and N. Ayache. Understanding the demon's algorithm: 3D non-rigid registration by gradient descent. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI'99*, pages 597–605. Springer, 1999.

[PCM03] V. Pascucci and K. Cole-McLaughlin. Parallel Computation of the Topology of Level Sets. *Algorithmica*, 38(2):249–268, 2003.

[PM90] 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.

[PPAC10] B. Pelletier, P. Pudlo, and E. Arias-Castro. The Cheeger Constant: from Discrete to Continuous. *Arxiv preprint arXiv:1004.5485*, 2010.

[Pre70] J. Prewitt. *Picture Processing and Psychopictorics*, chapter Object enhancement and extraction, pages 75–149. Academic Press, 1970.

[PSO+01] R. Pielot, M. Scholz, K. Obermayer, E. Gundelfinger, and A. Hess. 3D edge detection to define landmarks for point-based warping in brain imaging. *Proc. Int. Workshop Image Process.*, 2, 2001.

[PT04] J. Paiva and A. Thompson. Volumes on normed and Finsler spaces. *A sampler of Riemann-Finsler geometry*, page 1, 2004.

[Ree46] G. Reeb. Sur les points singuliers d'une forme de pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus Acad. Sciences*, 222:847–849, 1946.

[Rie54] B. Riemann. *On the hypotheses which lie at the foundations of geometry.* 1854.

[RK82] A. Rosenfeld and A. C. Kak. *Digital Picture Processing.* Computer Science and Applied Mathematics. Academic Press, 2nd edition, 1982.

[Rob65] L. Roberts. Machine perception of 3D solids. *Opt. Electro-Opt. Inf. Process.*, pages 159–197, 1965.

[ROF92] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[SAM08] N. Sabater, A. Almansa, and J.-M. Morel. Rejecting wrong matches in stereovision. CMLA Preprint 2008-28, ENS Cachan, CNRS, Cachan, France., 2008.

[San53] L. Santaló. *Introduction to integral geometry*. Hermann, 1953.

[Sap01] G. Sapiro. *Geometric partial differential equations and image analysis*. Cambridge University Press, 2001.

[SCSA04] A. Sole, V. Caselles, G. Sapiro, and F. Arandiga. Morse description and geometric encoding of digital elevation maps. *Image Processing, IEEE Transactions on*, 13(9):1245–1262, 2004.

[Sed01] R. Sedgewick. *Algorithms in C*. Addison-Wesley Professional, 2001.

[SG00] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Trans. Image Process.*, 2000.

[SH08] M. Sarfraz and O. Hellwich. Head pose estimation in face recognition across pose scenarios. In *International Conference on Computer Vision Theory and Applications*, pages 235–242, 2008.

[SKH06] D. Sarioz, T. Kong, and G. Herman. History Trees as Descriptors of Macromolecular Structures. *Lect. Notes Comp. Sci.*, 4291:263, 2006.

[SM02] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2002.

[SMS02] A. Sarti, R. Malladi, and J. Sethian. Subjective surfaces: a geometric model for boundary completion. *International Journal of Computer Vision*, 46(3):201–221, 2002.

[SPK02] L. Shafarenko, H. Petrou, and J. Kittler. Histogram-based segmentation in a perceptually uniform color space. *Image Processing, IEEE Transactions on*, 7(9):1354–1358, 2002.

[SSV+97] J. Sijbers, P. Scheunders, M. Verhoye, A. Van der Linden, D. Van Dyck, and E. Raman. Watershed-Based Segmentation of 3D MR Data for Volume Quantization. *Magnetic Resonance Imaging*, 15:679–688, 1997.

[SZ02] Y. Song and A. Zhang. Monotonic tree. *Proc. 10th Int. Conf. Discrete Geom. Comp. Imagery*, 2002.

[SZ03] Y. Song and A. Zhang. Analyzing scenery images by monotonic tree. *Multimedia Systems*, 8(6):495–511, 2003.

[Tar75] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, 1975.

[Tay78] J. Taylor. Crystalline variational problems. *Bulletin of the American Mathematical Society*, 84(4):568–588, 1978.

[TT86] R. Tamassia and I. Tollis. A unified approach to visibility representations of planar graphs. *Discrete and Computational Geometry*, 1(1):321–341, 1986.

[TV98] S. P. Tarasov and M. N. Vyalyi. Construction of contour trees in 3d in O(n log n) steps. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 68–75. ACM Press, 1998.

[USS00] C. Uhlenküken, B. Schmidt, and U. Streit. Visual exploration of high-dimensional spatial data: requirements and deficits* 1. *Computers & Geosciences*, 26(1):77–85, 2000.

[va] various authors. Vtk, the visualization toolkit.

[VBVV07] E. Vazquez, R. Baldrich, J. Vazquez, and M. Vanrell. Topological histogram reduction towards colour segmentation. *Pattern Recognition and Image Analysis*, pages 55–62, 2007.

[vGW94] A. van Gelder and J. Wilhelms. Topological considerations in isosurface generation. *ACM Transactions on Graphics (TOG)*, 13(4):337–375, 1994.

[VKM07] A. Vichik, R. Keshet, and D. Malah. Self-dual morphology on tree semilattices and applications. *Mathematical Morphology and its Applications to Image and Signal Processing, Proc. of ISMM*, 2007:49–60, 2007.

[VS91] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based onimmersion simulations. *IEEE Trans. Pattern Anal. Machine Intellicence*, 13(6):583–598, 1991.

[vvB+97] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 212–220. ACM Press, 1997.

[Wei98] J. Weickert. *Anisotropic diffusion in image processing*, volume 256. Citeseer, 1998.

[Wis69] D. Wishart. Mode analysis: A generalization of nearest neighbor which reduces chaining effects. In *Numerical taxonomy: proceedings of the Colloquium in Numerical Taxonomy held in the University of St. Andrews, September 1968*, page 282. academic press, 1969.

[Wit83] A. Witkin. Scale-space filtering. In *Proceedings of the Eighth international joint conference on Artificial intelligence*, volume 2, pages 1019–1022. Morgan Kaufmann Publishers Inc., 1983.

[XDG09] G. Xia, J. Delon, and Y. Gousseau. Locally invariant texture analysis from the topographic map. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2009.

[YB96] A. Yuille and H. Bülthoff. Bayesian decision theory and psychophysics. *Perception as Bayesian inference*, pages 123–161, 1996.

[ZOF01] H. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. *1st IEEE Workshop Var. Lev. Set Methods*, 80(3):194–202, 2001.