THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à MINES ParisTech

Image Characterization by Morphological Hierarchical Representations

École doctorale n°432

SCIENCES ET MÉTIERS DE L'INGÉNIEUR

Spécialité MORPHOLOGIE MATHÉMATIQUE

COMPOSITION DU JURY :

Mme Isabelle BLOCH
Institut Mines-Telecom, Telecom ParisTech,
Présidente et rapporteure

M. Hichem SAHLI
Vrije Universiteit Brussel, Rapporteur

M. Thierry GERAUD
EPITA, Membre du jury

Mme Cristina GOMILA
Technicolor, Membre du jury

M. Fernand MEYER
MINES ParisTech, Directeur de thèse

M. Santiago VELASCO-FORERO
MINES ParisTech, Maître de thèse

Soutenue par **Amin Fehri**
le 25 mai 2018

Dirigée par **Fernand Meyer**
et par **Santiago Velasco-Forero**

MINES ParisTech | PSL UNIVERSITÉ PARIS

Image Characterization by Morphological Hierarchical Representations

Amin Fehri

25 mai 2018

Remerciements

Alors que j'en arrive à conclure ce travail de thèse et à refermer ce chapitre de mon parcours, je réalise et j'apprécie le plaisir que j'y ai pris, ainsi que la chance que j'ai eu de pouvoir évoluer dans un environnement scientifique et humain aussi favorable. Je tiens donc à remercier ici toutes les personnes qui ont rendu cette période aussi agréable et enrichissante, et ont permis le succès de cette thèse.

Tout d'abord, je tiens donc à remercier chaleureusement le jury, à commencer par la rapporteure-présidente Isabelle Bloch et le rapporteur Hichem Sahli pour leurs commentaires constructifs et pertinents sur le manuscrit qui ont permis son amélioration. Je remercie aussi Cristina Gomila et Thierry Géraud pour leurs remarques judicieuses et les discussions très intéressantes qu'ils ont engagées autour de ce travail.

J'ai évidemment également beaucoup de reconnaissance pour Fernand, dont le savoir-vivre, le talent, l'érudition et la hauteur de vue m'inspirent et m'inspireront pour longtemps, ainsi que Santiago pour son aide scientifique et technique bien sûr, rendue possible par l'étendue de ses connaissances et de son ingéniosité, mais aussi pour son soutien moral et les longues discussions passionnantes que nous avons pu avoir. Cela a été un réel plaisir de travailler à vos côtés et d'apprendre de vous.

Le Centre de Morphologie Mathématique (CMM) et le site de Fontainebleau de Mines ParisTech se sont révélé, de façon générale, être un foyer très accueillant. J'ai pris beaucoup de plaisir à y évoluer et à y travailler. Je tiens à remercier le directeur du CMM Michel Bilodeau pour m'avoir permis d'y entrer, ainsi que toute l'équipe, avec qui j'ai eu le plaisir d'échanger et, pour certains, de travailler. Je remercie donc Jesús Angulo, Etienne Decencière, Petr Dokladal, Bruno Figliuzzi, Beatriz Marcotegui, Samy Blusseau, François Willot, Dominique Jeulin, José-Marcio Martins da Cruz, Matthieu Faessel. Je remercie également Catherine et Anne-Marie pour leur accueil bienveillant et leur aide, sans elles le CMM ne serait pas exactement le CMM.
Ces années à Fontainebleau ont aussi été l'occasion de rencontrer de nombreux étudiants et chercheurs, dans une ambiance joyeuse et détendue qui je suis sûr me laisseront nostalgique. Malgré mon éloignement (relatif) parisien, j'ai eu la chance de passer très bons moments avec vous, à discuter de cinéma, de science, à refaire le monde autour d'un café, à jouer au baby-foot,

au tarot, au perudo, au mölkky, et autres joyeusetés. Je remercie donc Gianni, Théo, Jean-Baptiste, Pierre, Marine, Sébastien, Vaïa, Albane, Leonardo, Aurélien, Jean-Charles, Luc, Antoine, Bassam, Anguerrand, Kaiwen, Eric, Elodie, Robin, Mike, Haisheng, Borja, Jean, Irini, Sara, Laure, Jihane, Nicolas, Angélique, Nidhal ...

Je remercie également tous mes amis extérieurs à l'environnement du laboratoire, pour leur soutien, leur compréhension de ma présence intermittente et parfois songeuse (le travail de thèse étant à l'occasion plutôt *accaparant*), et surtout la joie qu'ils mettent dans ma vie. Je remercie donc Thomas, Raphaël, Nicolas, Clément, Anne, Amina, Xavier, Guillaume, Laurent, David, Alexandra, Fred, Salwa, Boris, Cannelle, Richard, Alexandre, Thibaud, Lydie, Nithila ...

Enfin, j'ai une pensée toute particulière et émue pour ma famille, mes parents bien sûr pour leur soutien indéfectible et inconditionnel, mais aussi Anis, Inès et Elias, la plus belle fratrie qui soit.

Résumé

Cette thèse porte sur l'extraction de descripteurs hiérarchiques et multi-échelles d'images, en vue de leur interprétation, caractérisation et segmentation. Elle se décompose en deux parties.

La première partie expose des éléments théoriques et méthodologiques sur l'obtention de classifications hiérarchiques des nœuds d'un graphe valué aux arêtes. Ces méthodes sont ensuite appliquées à des graphes représentant des images pour obtenir différentes méthodes de segmentation hiérarchique d'images. De plus, nous introduisons différentes façons de combiner des segmentations hiérarchiques. Nous proposons enfin une méthodologie pour structurer et étudier l'espace des hiérarchies que nous avons construites en utilisant la distance de Gromov-Hausdorff entre elles.

La seconde partie explore plusieurs applications de ces descriptions hiérarchiques d'images. Nous exposons une méthode pour apprendre à extraire de ces hiérarchies une bonne segmentation de façon automatique, étant donnés un type d'images et un score de bonne segmentation. Nous proposons également des descripteurs d'images obtenus par mesure des distances inter-hiérarchies, et exposons leur efficacité sur des données réelles et simulées. Enfin, nous étendons les potentielles applications de ces hiérarchies en introduisant une technique permettant de prendre en compte toute information spatiale a priori durant leur construction.

Mots clés : Traitement d'images, Morphologie mathématique, Segmentation, Segmentation hiérarchique, Classification hiérarchique, Apprentissage statistique, Théorie des graphes.

Abstract

This thesis deals with the extraction of hierarchical and multiscale descriptors on images, in order to interpret, characterize and segment them. It breaks down into two parts.

The first part outlines a theoretical and methodological approach for obtaining hierarchical clusterings of the nodes of an edge-weighted graph. These methods are then applied to graphs representing images and derive different hierarchical segmentation techniques. In addition, we introduce different approaches to combine hierarchical segmentations. Finally, we propose a methodology for structuring and studying the space of hierarchies by using the Gromov-Hausdorff distance as a metric.

The second part explores several applications of these hierarchical descriptions for images. We expose a method to learn how to automatically extract a segmentation of an image, given a type of images and a score of evaluation for a segmentation. We also propose image descriptors obtained by measuring inter-hierarchical distances, and expose their efficiency on real and simulated data. Finally, we extend the potential applications of these hierarchies by introducing a technique to take into account any spatial prior information during their construction.

Keywords: Image Processing, Mathematical Morphology, Segmentation, Hierarchical Segmentation, Hierarchical Clustering, Machine Learning, Graph Theory.

Contents

Introduction

In this thesis, we introduce and study images structurings as series of nested partitions, or hierarchical segmentations, expressing the salient parts of them. Hierarchical segmentation has been one of the major trends in recent years for segmentation and filtering tasks, as it allows to deal with the inherently multi-scale structures and information present in the images. To go further, we take interest in studying such hierarchical representations as images features, that we can combine and adapt for image segmentation, characterization or recognition.

Any method which progressively merges adjacent regions of an initial partition produces such a hierarchy, characterized by the fact that each region of the hierarchy is either a region of the initial partition or results from the fusion of smaller regions of this partition. The resulting hierarchy depends upon the strategy for merging regions.

More specifically, the hierarchical structures that we systematically explore in this work are the so-called *morphological hierarchies*, constructed in the classical framework of mathematical morphology, and thus capture morphological information about local differences in contrast, or regions sizes and shapes. We present three main streams for constructing morphological hierarchies:

— The hierarchies associated to a sequence of levelings. As connected operators, the levelings merge and extend the flat zones, naturally creating a hierarchy of flat zones.

— The hierarchies associated with the watershed. The catchment basins of a topographic surface form a partition, where each region contains a regional minimum of the surface. By flooding the relief, more and more minima are suppressed and the corresponding basins merge or are absorbed by neighboring basins.

— The last type of methods explored in this thesis for constructing hierarchies is the stochastic watershed. The stochastic watershed produces a hierarchy, by estimating the probability that a particular region of the hierarchy is produced by a watershed segmentation in which random germs are used as markers.

Despite the fact that the methods we use for constructing hierarchies are diverse, the hierarchies are represented in the unified framework of weighted graphs. First of all, one constructs a fine partition to which is associated a region adjacency graph or RAG. Each node of the RAG represents a region of the fine partition. Two nodes are linked by an edge if the

corresponding regions are neighbors in the fine partition. Each edge is weighted by a weight expressing the dissimilarity between the regions represented by its extremities.

This representation paves the way for introducing various distances between the nodes of the graph. One of them will play a fundamental role in this thesis, namely the ultrametric distance. A series of nodes, in which two successive nodes are linked by an edge is called a path. Each path then gets a weight, equal to the weight of the highest edge along this path. The ultrametric distance between two nodes is by definition the lowest weight of the paths connecting these nodes. Other distances, like lexicographic distances may be used, refining the ultrametric distance.

Various auxiliary representations may then be derived, like the minimum spanning tree (MST) of the weighted graph. The MST completely encodes the ultrametric distance, as the ultrametric distance between any pair of nodes of the graph may be derived from it. The regions of the hierarchy appear to be the closed balls of the ultrametric distance. An order relation between regions is defined expressing the level in which the regions have merged. A dendrogram based on this order relation structures the interrelations between the regions of the hierarchy.

Additionally, an order relation between hierarchies structures them as a lattice, in which several operators like the supremum and infimum may be defined.

Finally, as the hierarchies provided with an ultrametric distance become metric spaces, one may use the Gromov-Hausdorf distance between hierarchies in order to study the geometry of the space of hierarchies.

We have presented three main families of morphological hierarchies. For each of them we explain how to derive the associated weighted graph, minimum spanning tree and dendrograms.

These families of hierarchies may be further enriched, by combining hierarchies. The sequential chaining of hierarchies transforms an input hierarchy into an output hierarchy by assigning new weights to its minimum spanning tree. The parallel combinations takes advantage of the fact that all hierarchies based on the same fine partition share by construction the same minimum spanning tree. Thanks to this property, the many parallel combinations of hierarchies become in addition extremely fast to compute.

Building on these theoretical and methodological elements, several applications of these hierarchical features are then presented for image segmentation and classification.

This thesis will thus be decomposed in two parts:

1. A methodological part, in which will be introduced in details the different notions we will use regarding graphs, hierarchies, images modelizations as graphs and multiple ways to generate and combine hierarchical representations of images. Note that the chosen representation is very general, so that even if we apply it to images, it may be applied to other types of data modeled as graphs.
2. Provided with these tools, we address in the second part several applications examples. We expose a framework for learning-based image segmentation for a given task. We also see how to do efficient image classification using hierarchical features. Finally, we present a

simple and versatile way of introducing prior information for hierarchical segmentation and image representation.

More precisely, the rest of the work presented herein is structured as follows:

— **Chapter 1** introduces notations, definitions and properties regarding graphs and trees, and especially about the minimum spanning tree that has an important role in the methods introduced throughout this thesis. It also makes a reminder on the taxonomy of clusterings on graphs methods, and expose non-hierarchical techniques to do so.

— **Chapter 2** focuses specifically on hierarchical clustering. It introduces definitions, notations and properties concerning hierarchies, their relation to ultrametric distances, and their representation as dendrograms. In the case of a graph, it then shows how the structure of its minimum spanning tree inherently induces a hierarchical clustering of its nodes, which takes the form of a minimum spanning forest. Finally, a lattice structure on the hierarchical space is introduced, with the existence of an order relation, possible combinations and a distance between hierarchies.

— **Chapter 3** exposes some image hierarchical segmentations algorithms implemented within the graphs-based hierarchical clustering framework described in the previous chapter. They effects are illustrated from a qualitative point of view and some insights regarding the choices we made to implement them are provided.

— **Chapter 4** proposes several methods to combine hierarchies and thus obtain derived hierarchies that capture complex features of the image. In the specific case where we combine hierarchies obtained from the same minimum spanning tree, an interesting result is demonstrated that simplifies the combination of these hierarchies. A possible structuring of the space of hierarchies is also proposed by using the Gromov-Hausdorff distance, which is difficult to compute in the general case but is more easily obtained between two hierarchies built on the same set of regions. By combining this distance with tools for dimensionality reduction and data visualization, we obtain a methodology to study and visualize the combinatorial space of possible hierarchies.

— **Chapter 5** proposes a general framework to learn, for given type of images and score to evaluate the quality of a segmentation, the hierarchy/combination of hierarchies and the cut level that are the better suited to the problem. It is illustrated on several segmentation problems.

— **Chapter 6** proposes to use the Gromov-Hausdorff distance between hierarchies built upon the same image to generate image descriptors that can be used for classification tasks. These descriptors are tested to classify images generated using dead-leaf process with different parameters, as well as texture image classes.

— **Chapter 7** presents a method that allows to take into account any prior spatial information for obtaining a hierarchical segmentation that emphasizes the contours or regions of

interest while preserving the important structures in the image. Several applications are presented that illustrate the versatility and efficiency of the method.

— **Chapter 8** draws some perspectives on how morphological hierarchical segmentations may complement Convolutional Neural Networks (CNN), a class of methods providing state-of-the-art results in many areas of computer vision.

The thesis concludes by a summary of the main contributions, and by offering some perspectives on future research directions.

Part I

Theoretical and methodological elements

Introduction

The object of this thesis is the introduction, study and use of several images structurings as series of nested partitions, i.e. hierarchical segmentations. These multi-scale representations are useful as the pertinent information about an image are often present at several different scales.

In the first part of the thesis, we show how these structurings can be built in a graph-based framework, by operating on graph representations of images. More specifically, the minimum spanning tree structure is introduced and intensively used throughout their construction: it is a tree (without cycles), containing all nodes, such that the sum of its edge weights is minimal. This structure is inherently linked with hierarchical clustering, and leads to hierarchies enjoying nice theoretical properties, as well as efficiency and practical convenience.

The main contributions of the methodological part of this thesis are the following:

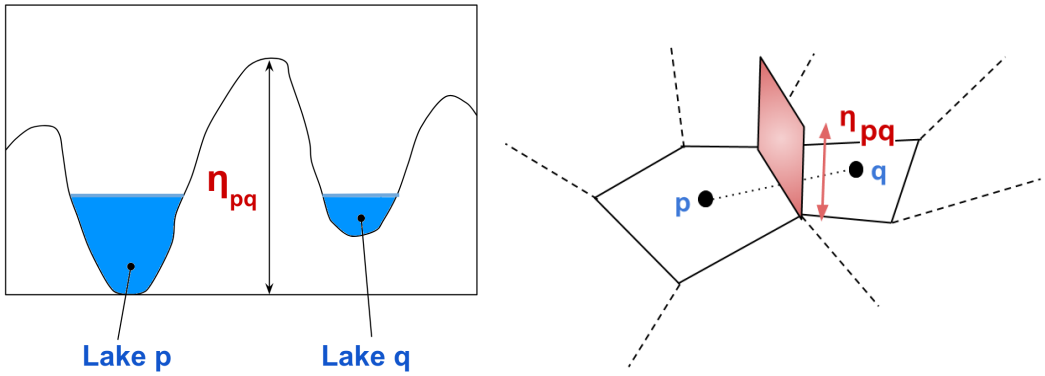
1. We present numerous morphological hierarchical segmentation methods in a graph-based framework.
2. We propose new comprehensive ways to combine them, by simple arithmetic or logical operations. In the particular case where we combine hierarchies built upon the same minimum spanning tree structure, such combinations are proved to be very fast to obtain in most cases.
3. We propose a framework to study the properties of the resulting high-dimensional space of possibilities, using the Gromov-Hausdorff distance between hierarchies, as long as visualization tools from the data analysis and visualization literature.
4. We propose and describe efficient implementations of these tools within the open-source library Smil.

Prologue

The purpose of this prologue is to motivate the presentation of theoretical tools regarding graphs that are going to be introduced in this part of the thesis. As was already stated, we often need to study the image at different levels, since pertinent information is present at several scales. A possible structure to capture such multi-scale information in an image is a series of nested partitions, or hierarchy. Different hierarchies can be constructed for the same image.

Furthermore, the hierarchies we introduce are for the most part morphological. In mathematical morphology, an image is often seen as a topographic relief. Flooding such a relief leads to watershed lines delineating regions of the image. This representation is called *critical lakes*: it is inherently linked with the notion of hierarchy, as to each type of flooding of the relief is associated a progressive fusion of regions, and therefore a hierarchy of them. When the image is represented as a graph, an equivalent artificial representation of this flooding process exists. The graph modeling the image is built on a fine partition of this image. Each node corresponds to a region of this fine partition, and adjacent regions are linked and weighted according to a

dissimilarity. One can then consider that, between any two adjacent regions, there is a wall with no thickness and a height equal to the dissimilarity between these regions. This “virtual” relief gives us a frame in which we develop several methods. The correspondence between those two modelizations is illustrated in figure 0.1.



(a) Topographic relief flooding (b) Graph modelization

Figure 0.1 – Correspondance between the topographic relief and the graph modelizations of an image flooding.

We have thus defined the frame in which this work takes place. The tools/representations we work with are defined for edge-valued graphs. This is why we begin by introducing in chapters 1 and 2 the theoretical notions on graphs and clustering/hierarchical clustering on them. We can then make use of these notions to present in chapter 3 several morphological hierarchical segmentations methods, as well as their efficient implementation. These methods follow a common mechanism. To a series of progressive floodings correspond watershed lines, which naturally produces a hierarchy. The hierarchies then differ depending on the flooding type. A specific type of flooding is the marker-based one, in which we choose some nodes to be the flooding sources. This way, we can specify which nodes we want to be separated in the output segmentation. The stochastic watershed model then constitutes a supplementary step, in which we compute for each contour statistical values corresponding to multiple flooding processes simulations with random markers. Finally, we extend the number of possible hierarchical structurings of the image in chapter 4, by showing how we can combine hierarchical segmentations. On this occasion, we introduce several tools to quantify and visualize the relative descriptive power of different hierarchical segmentation methods.

Chapter 1

Graph Theory

Résumé

Dans ce chapitre, nous introduisons les notations, définitions et propriétés sur les graphes et arbres, et notamment l'arbre de poids minimum qui va occuper un rôle important dans les méthodes introduites dans cette thèse. Nous faisons également un rappel sur la taxonomie des méthodes de classification des noeuds d'un graphe, ainsi qu'un état de l'art des méthodes non-hiérarchiques pour ce faire.

Abstract

In this chapter, we introduce notations, definitions and properties regarding graphs and trees, and especially about the minimum spanning tree that has an important role in the methods introduced throughout this thesis. We also do a reminder on the taxonomy of clusterings on graphs methods, and expose non-hierarchical techniques to do so.

1.1 Introduction

Graph theory appears in modern mathematics at the confluence of combinatorics and computer science. Fundamentally, a graph is a mathematical structure useful to model pairwise relations between objects. The seminal paper of this field is by Euler and dates back to 1736 [Euler (1736)]. It deals with the problem of the seven bridges of Könisgberg, illustrated in figure 1.1: two branches of the river are flowing around the Kneiphoff island in Könisgberg, and seven bridges are crossing these two branches, and the question is to determine whether a person can plan a walk such that he would cross each one of the bridges once and only once. Later on, ideas coming from both mathematics and chemistry got to fuse, contributing to the standard terminology of graph theory. Indeed, graph theory has been extensively used in chemistry for molecular modeling, for example for the study of isomers by Cayley [Cayley (1874)]. The term "graph" itself was introduced by Sylvester [Sylvester (1878)].

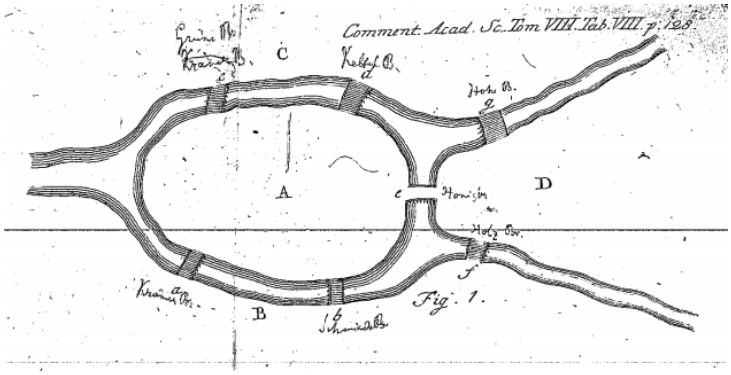


Figure 1.1 – Excerpt from Euler's paper about the seven bridges of Königsberg problem, which is the following. Two branches of the river are flowing around the Kneiphoff island in Königsberg, and seven bridges are crossing these two branches. The question is then to determine whether a person can plan a walk such that he would cross each one of the bridges once and only once.

Then, graph theory has mainly been developed as mathematical puzzles during the second half of the 19th century, such as the "Icosian Game" proposed by W.R. Hamilton, in which the goal is to find a Hamiltonian circuit, i.e. a loop along the edges of a icosahedron.

Progress in graph theory has also been driven by real-life applications regarding for example the study of networks and circuits. Historically, Kirchoff in 1847 [Kirchoff (1847)] was the first to model electrical network geometry in the form of a graph (known as electric scheme). Kirchoff rules then provide a way to determine the linearly independent currents and voltages equations. Furthermore, this graph representation enabled Kirchoff to compute, in the case of a network with a finite number of linear resistors, the currents in any of the resistors, when knowing the voltage in the network. Kirchoff's work is considered as the basis of the flow theory and related algebraic methods.

As a result of its successes, a growing number of mathematicians worked to improve the seminal works cited before, and links with other mathematical branches were made. Since 1930, thousands of papers have been published regarding theoretical considerations as well as a large spectrum of real life applications, with graphs representing data as various as electrical networks, phylogenetic information or social media relationships. Graph theory is now linked with algebra, topology, combinatorics, and provides state-of-the-art methods for problems in computer science, operations research, game theory, decision theory or machine learning. For example, algebraic graph theory aims at applying algebraic methods to graphs problems, and a sub-branch of this field resides in studying the spectrum of matrices representing graphs, which is known as spectral graph theory [Ng et al. (2002); Von Luxburg (2007)]. Other recent theoretical research directions consist in finding how to use successful learning-based approaches such as Convolutional Neural Networks (CNN) on graphs representations to be able to classify them [Niepert et al. (2016)], or in using graph structures to help learning algorithms to handle relational reasoning (i.e. reason the relations between entities) [Santoro et al. (2017)].

From an application point of view, image processing and more specifically image segmentation (a branch of image processing aiming at partitioning the image domain into a set of meaningful

regions according to some pre-specified criteria) methods using graphs have flourished. One may notice that two main tendencies emerge in these methods.

A Kirchoff-like approach consists in defining the notion of flow on the graph, and trying to maximize this flow with the minimum possible cuts on the graph. This approach, referred to as *Max-Flow Min-Cut*, has been widely used for image segmentation [Malik et al. (2001); Boykov et al. (2004); Felzenszwalb et al. (2004)].

An other approach is linked with the problem of the *Minimum Spanning Tree* whose history is described in [Nešetril et al. (2010)], and which has been initially addressed by Boruvka [Boruvka (1926)]. The initial issue was to find a way to minimize the total length of cables needed to electrify Romania. He did so by modeling this problem as a minimization one on graphs, where the goal is to find for each pair of nodes, the path with the lowest edges. The union of all these paths constitutes a minimum spanning tree (provided that edge valuations are all different), and cutting edges on this tree leads to clustering of the graph. This notion has then be used for graphs describing images, thus leading to new ways of obtaining images segmentations [Xu and Uberbacher (1997)] or filtering images [Stawiaski and Meyer (2009); Bao et al. (2014)]. This approach is somehow complementary to the previous one, as it attempts at finding the shortest path between any two nodes, instead of maximizing the flow. The work presented in this thesis falls within this category and makes a heavy use of the minimum spanning tree structure and its convenient properties.

Note that connexions have been made in the literature between these two techniques, showing that they solve similar minimization problems [Couprie et al. (2011)].

1.2 Notations and Definitions

This first sections introduces definitions regarding graphs and important related structures.

1.2.1 Definitions

Definition 1.1 (Graph)**.** A *graph* $\mathcal{G}$ is a pair (V,E), with V and E both finite sets. The elements $\mathsf{v} \in \mathsf{V}$ are called *vertices* or *nodes*, and the elements $\mathsf{e} \in \mathsf{E} \subset \{\{i,j\}, (i,j) \in \mathsf{V} \times \mathsf{V} := \mathsf{V}^2, i \neq j\}$ are called *edges*. An edge $\{i,j\}$ of E is denoted e_{ij} or (i,j).

Definition 1.2 (Adjacency)**.** We say that an edge e_{ij} connects i and j. In this case the vertices i and j are *neighbors*.
Similarly, two edges are *adjacent* if they have a node in common.

Definition 1.3 (Subgraph)**.** Given a graph $\mathcal{G} = (\mathsf{V}, \mathsf{E})$, the graph $\mathcal{G}' = (\mathsf{V}', \mathsf{E}')$ is a *subgraph* of $\mathcal{G}$ if and only if:

— $\mathsf{V}' \subset \mathsf{V}$.

— $\mathsf{E}' = \{\mathsf{e}_{pq} \in \mathsf{E} | (p,q) \in \mathsf{V}'^2\}$

Definition 1.4 (Partial graph)**.** Given a graph $\mathcal{G} = (\mathsf{V}, \mathsf{E})$, the graph $\mathcal{G}' = (\mathsf{V}', \mathsf{E}')$ is a *partial graph* of $\mathcal{G}$ if and only if $\mathsf{E}' \subset \mathsf{E}$ and $\mathsf{V}' = \{(p,q) \in \mathsf{V}'^2 | \mathsf{e}_{pq} \in \mathsf{V}'\}$.

Definition 1.5 (Edge-weighted graph)**.** An edge-weighted graph $\mathcal{G}$ is a triplet $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta)$, where (V, E) is a graph and η is a mapping of the set of edges E into an ordered set T (for example, $[0, N]$ for $N \in \mathbb{N}$, $\mathbb{R}^+$, $\mathbb{R}$, $\mathbb{Z}$ etc.). For each edge e_{ij} of $\mathcal{G}$ we write $\eta_{ij} = \eta(\mathsf{e}_{ij})$. η_{ij} is called the *weight* of the edge e_{ij}.

Definition 1.6 (Weight map)**.** The mapping η is called the weight map of $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta)$.

Definition 1.7 (Adjacency matrix)**.** An adjacency matrix is a square matrix $A_{\mathcal{G}}$ used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph. In the case of an undirected graph, it is a symmetric (0,1)-matrix with zeros on its diagonal.

$A_{\mathcal{G}} = (a_{ij})_{(i,j)\in\{1,\dots,N\}}$, with N the number of nodes in the graph $\mathcal{G}$, s.t.:

$$a_{ij} = \begin{cases} 1 & \text{if } i,j \text{ are adjacent in } \mathcal{G} \\ 0 & \text{otherwise} \end{cases}$$

.

Definition 1.8 (Node-weighted graph)**.** A node-weighted graph $\mathcal{G}$ is a triplet $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \nu)$, where (V, E) is a graph and ν is a mapping of the set of nodes V into an ordered set T. For each node v_i of $\mathcal{G}$ we write ν_i its weight.

Definition 1.9 (Node weight map)**.** The mapping η_{V} is called the node weight map of $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \nu)$.

Graphs can be directed or undirected. In the first case, edges e_{ij} and e_{ji} are considered to be distinct edges. In this context, edges are sometimes referred to as arcs or arrows. In the following, we will mainly focus on undirected graphs. A graphical representation of graphs is useful for explaining concepts or illustrating algorithms, as illustrated in figure 1.2.

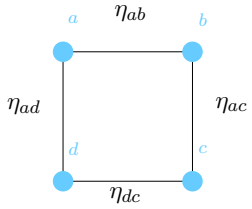


Figure 1.2 – Graphical representation of a graph. Nodes are represented by disks and edges by lines connecting nodes.

Definition 1.10 (Edge contraction)**.** The *edge contraction* of an edge $\mathsf{e}ij$ consists in replacing it with both its extremities (i, j) by a unique vertex which becomes an extremity for all edges adjacent to $\mathsf{e}ij$. It is illustrated in figure 1.3.

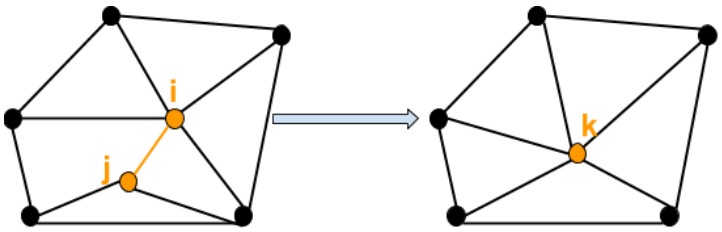


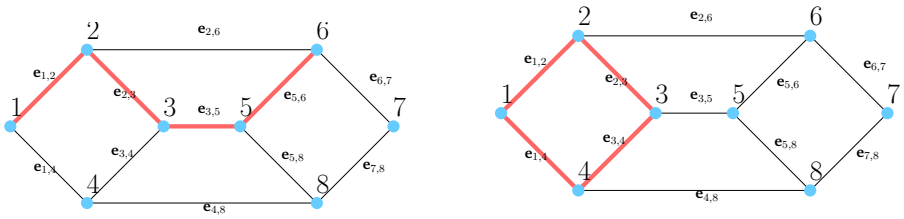
Figure 1.3 – Example of an edge contraction.

The notion of adjacency in a graph is useful to derive some geometrical properties of graphs. Let us imagine a graph representing a network of roads connecting cities. Neighborhood relations then model connections between cities. From them, it is possible to derive the notions of travels, walks and paths on a graph.

Definition 1.11 (Path)**.** A *path* π of the graph $\mathcal{G} = (\mathsf{V}, \mathsf{E})$ is a sequence of nodes and edges, interweaved in the following way: π starts with a vertex, say p, followed by an edge e_{ps}, incident to p, followed by the other endpoint s of e_{ps}, and so on. We write $\pi = (p, \mathsf{e}_{p,s}, s, \ldots, q)$ and call π a path from p to q. The set of all paths from p to q is denoted $\mathbf{\Pi}_{(p,q)}$. In the case where there two nodes are connected by at most one edge, there is no ambiguity and we can specify the nodes only, such that $\pi = (p, s, ..., q)$.

Definition 1.12 (Paths concatenation)**.** The *concatenation* of two paths π_{pq} and π_{qs} is the path formed by joining these two paths such that the extremity of π_{pq} is the origin of π_{qs}. We denote it $\pi_{pq} \triangleright \pi_{qs}$.

Definition 1.13 (Cycle)**.** A *cycle* (or circuit) is a path $\pi = (i, \mathsf{e}_{i,i+1}, ..., (k-1), \mathsf{e}_{k-1,k}, k)$ such that $i = k$.



(a) Path from node 1 to node 6. (b) Cycle example

Figure 1.4 – Examples of path and cycle.

The notions of path and cycle are illustrated in figure 1.4, and are close to the intuitive ideas they correspond to. In particular, the path is a fundamental notion that we will meet in all shortest paths problems.

Definition 1.14 (Cocycle)**.** Let us consider a graph $\mathcal{G} = (\mathsf{V}, \mathsf{E})$, and two separated sets of nodes A and B in V. The *cocycle* Cocycle(A, B) of both sets A and B is the set of edges with one extremity in A and the other in B:

$$\text{Cocycle}(A, B) = \{(p, q) \in \mathsf{E} | p \in A, q \in B\} \tag{1.1}$$

By extension, the cocycle Cocycle(A) of A is the set of edges with one extremity in A and the other in $A^{\complement}$.

1.2.2 Distances on graphs

In this thesis, we address the problem of graph partitioning in the context of image segmentation. Standard morphological approaches to clustering make a large use of various distances, the general method consisting in operating a SKIZ on a Voronoï tessellation defined for these different distances. In this process, germs are either regional minima or predetermined markers. We thus now introduce several distances on graphs. Their construction mechanism is similar:

1. We begin by defining the weight of a path π. We denote it $|\pi|$.
2. We then define the shortest path between any two nodes p and q as the path with minimal weight $\eta(p, q)$ between p and q (and we set it to ∞ is no path exists between these nodes). It indeed then verify the triangular inequality necessary to be a distance: if $\eta(p, s) = |\pi_{p,s}|$, $\eta(s, q) = |\pi_{s,q}|$ and $\eta(p, q) = |\pi_{p,q}|$ then $|\pi_{pq}| \leq |\pi_{ps} \triangleright \pi_{sq}|$, because the concatenation of $\pi_{p,s}$ and $\pi_{s,q}$ is one path among possible paths between the nodes p and q.

Definition 1.15 (Generalized Path Length, Shortest path distance, Lowest path distance)**.** The *n-generalized length* of a path π is defined by:

$$n \in \mathbb{N}^*, L_n(\pi) = \sqrt[n]{\sum_{\mathsf{e}_{i,j} \in \pi} \eta_{i,j}^n} \tag{1.2}$$

Moreover:

$$\begin{cases} \mathrm{L}_0(\pi) = \sum_{\mathsf{e}_{i,j} \in \pi} 1 \\ \mathrm{L}_1(\pi) = \sum_{\mathsf{e}_{i,j} \in \pi} |\eta_{i,j}| \\ \mathrm{L}_\infty(\pi) = \max_{\mathsf{e}_{i,j} \in \pi}(\eta_{i,j}) = \lim_{n \to +\infty} \left(\sqrt[n]{\sum_{\mathsf{e}_{i,j} \in \pi} |\eta_{i,j}|^n} \right) \end{cases} \tag{1.3}$$

The distance between two nodes then is the length of the minimal-length path between two nodes.

In this context, using the L_1-norm leads to the *shortest path distance*, and using the L_∞-norm leads to the *lowest path distance*. Both distances are illustrated by figure 1.5.

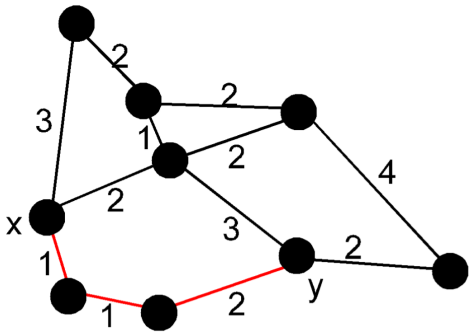


Figure 1.5 – (i) Shortest path distance: When using the L_1-norm, the length of a path is the sum of the weights of its edges, and the distance between two nodes is the minimal length of all paths: in this example it is equal to 4. (ii) Lowest path distance: When using the L_∞ norm, the length of a path is the highest weight of the edges along this path, and the distance between two nodes defined as before: here the associated distance is equal to 2. This disance corresponds to the ultrametric distance, so that $|\pi_{ps} \triangleright \pi_{sq}| = |\pi_{ps}| \vee |\pi_{sq}|$.

Definition 1.16 (Cheapest path)**.** When working with a node-weighted graph, one can define the length of a path as the sum of the nodes weights along this path:

$$L(\pi) = \sum_{\mathsf{v}_i \in \pi} \nu_i \tag{1.4}$$

Then the distance between two nodes corresponds to the length of the minimal-length path between them, and is referred to as the *cheapest path distance* between these two nodes. An example is provided in figure 1.6.

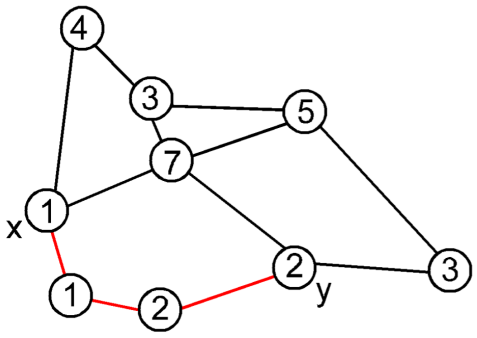


Figure 1.6 – On a node-weighted graph, the length of a path can be defined as the sum weights of the nodes along the path. The topographical distance then corresponds to the length of the minimal-length path among all paths. This distance is useful to interpret the classical watershed operator as a skeleton by zone of influence [Meyer (1994b)] for the topographic distance. In this example, the cheapest path between nodes x and y is equal to 6.

Definition 1.17 (Easiest path, Lexicographic distance)**.**

In an edge-weighted graph, we call toughness of a path π_{pq} between two nodes (p, q) the list of altitudes of the highest edges met along this path. It is obtained in the following way: (i) it is initiated with the weight of the highest edge when going from p to q, (ii) then we add the weight of the highest edge on the remaining path, (iii) and so on until arriving to q. Once the toughness has been defined, we can compare different paths using a lexicographic distance. Note that the toughness of a path depends on the departure and arrival nodes. An example of the toughness of a path is provided in figure 1.7. There are several possibilities to define a lexicographic distance [Meyer (2005)]:

— We can order the edges weights by decreasing order:

  1. by taking into account duplicate values when they exist.
  2. by taking into account each value only once.

— We can give an importance to the direction of path explorations: for a given path, we retain the highest weight, then the highest weight on the remaining path etc.

Note that the lexicographical distance is somehow finer than the lowest path distance. It conserves the inequalities between paths obtained with the shortest path distance, but if two paths are equivalent in view of the lowest path distance, they may differ for the lexicographical distance. In other words,the lexicographical distance is less short-sighted than the lowest path distance.

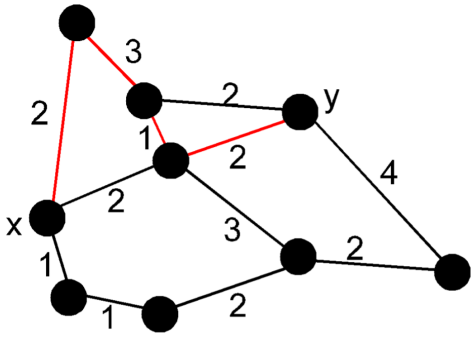


Figure 1.7 – On this example, the toughness of the path in red is [3,2] in boths way (node x to node y, or node y to node x).

Once we have defined these different distances, we may often want to compute the shortest path between nodes, using shortest paths algorithms. The classical shortest path algorithms, such as [Kruskal (1956); Prim (1957); Dijkstra (1959)], are defined for the L_1-norm, for which the weight of a path is its length. Gondran and Minoux have proposed an algebraic framework called path algebra, in which finding shortest paths amounts to solve a family of linear equations in this algebra [Gondran (1975)]. We will see in the next chapter how this indeed proves to be interesting in partitioning problems.

1.3 Trees and Forests

In this section, we expose definitions and notations regarding trees and forests.

1.3.1 Definitions

Definition 1.18 (Tree, forest)**.** A *tree* $\mathcal{T}$ is a connected graph without cycles. A graph $\mathcal{F}$ with no cycles is called a *forest*. Each tree is a connected subgraph of the forest.

Definition 1.19 (Spanning Tree)**.** A *spanning tree* $\mathcal{ST}$ of a graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$ such that is a tree which includes all of the vertices of $\mathcal{G}$. A graph may have several spanning trees.

Definition 1.20 (Spanning Forest)**.** A *spanning forest* $\mathcal{SF}$ of a graph $\mathcal{G}$ is a forest that spans all of the vertices of $\mathcal{G}$, meaning that each node of $\mathcal{G}$ is a vertex in $\mathcal{SF}$.

Theorem 1.21 (Spanning tree properties)**.** *Let* $\mathcal{T} = (\mathsf{V}, \mathsf{E})$ *be a graph with* n *vertices (*$|\mathsf{V}| = n$*). The following statements are equivalent:*

— $\mathcal{T}$ *is a spanning tree.*

— $\mathcal{T}$ *has* $(n-1)$ *edges and no cycles.*

— $\mathcal{T}$ *has* $(n-1)$ *edges and is connected.*

— $\mathcal{T}$ *contains a unique path between any pair of nodes.*

1.3.2 Connectivity

In order to address the problem of clustering on graphs, one must define the notions of connectivity and connected component on them. Indeed, an important aspect of clustering is to be able to connect nodes that share common properties, which is translated in these notions. In the following, we denote $\mathcal{S}$ a finite set.

Definition 1.22 (Equivalence relation)**.** A binary relation R on $\mathcal{S}$ is an *equivalence relation* if and only if it is reflexive, symmetric and transitive, i.e. $\forall(x, y, z) \in \mathcal{S}^3$:

— xRx (reflexivity);

— $xRy \Rightarrow yRx$ (symmetry);

— $((xRy) \wedge (yRz)) \Rightarrow xRy$ (transitivity).

Definition 1.23 (Equivalence class)**.** The *equivalence class* of an element $x \in \mathcal{S}$ is the set of all elements $y \in \mathcal{S}$ such that xRy.

Definition 1.24 (Path connectedness)**.** A path-connected space is a notion of connectedness requiring the structure of a path and the definition of a binary relation between neighbors nodes. Let us consider a given a symmetrical binary relation $\sim$ defined over couples of nodes on a metric set (S, d). We say that two nodes a and b are *path-connected*, and we note aRb, if there exists a path $(a, ..., b)$ connecting a to b such that between any two successive nodes x and y on this path, we have $x \sim y$. aRb is then an equivalence relation and the connected components the corresponding equivalence classes.

Definition 1.25 (Connected component)**.** A connected component $\mathcal{G}' = (\mathsf{V}', \mathsf{E}')$ of the graph $\mathcal{G} = (\mathsf{V}, \mathsf{E})$ is a subgraph of $\mathcal{G}$ such that there is a path connecting each pair of distinct nodes of $\mathcal{G}'$. A graph $\mathcal{G}$ is connected if $\mathcal{G}$ is itself a connected component.

Depending on the binary relation considered (cf. section 1.2), one obtains equivalence relations, and the associated partition may take several forms. For example, one may consider for a node-weighted graph, with each node v_i having a weight ν_i, the λ-flat condition as a binary relation:

$$\mathsf{v}_i R \mathsf{v}_j \Leftrightarrow |\nu_i - \nu_j| \leq \lambda \tag{1.5}$$

This leads to the partition of the λ-flat zones of the graph. When $\lambda = 0$, this leads to the flat zones of the graph. In the particular case where the graph represents an image, extracting λ-flat zones of the graph may be interesting as they correspond to regions with a weak gradient between one region and the next one.

1.3.3 Minimum Spanning Tree

The minimum spanning tree is a structure that we will often meet throughout this thesis, since it appears to be very useful for image segmentation. We define in this section the minimum

spanning problem and give some important properties of this structure. This structure is closely linked with the lowest path distance, as we shall see.

Definition 1.26 (Minimum Spanning Tree (MST))**.** Given an edge-weighted connected graph $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta)$, the minimum spanning tree is a spanning tree $\mathcal{MST} = (\mathsf{V}, \mathsf{E}_{MST})$ of $\mathcal{G}$ such that the sum of its edges is minimal.

$$\mathcal{MST}(\mathcal{G}) \coloneqq \underset{\mathcal{T} \in \boldsymbol{\mathcal{ST}}}{\arg\min} \left(\sum_{\mathsf{e}_{i,j} \in \mathsf{E}_{\mathcal{ST}}} \eta_{ij} \right) \tag{1.6}$$

Where $\boldsymbol{\mathcal{ST}}$ is the set of all spanning trees of $\mathcal{G}$.

In general, there is not a unique minimum spanning tree of a graph. In the particular case where all edges weights are different, the minimum spanning tree is unique. The number of minimum spanning trees of a given graph may thus be large. Note also that the minimum spanning tree property is invariant when applying a strictly increasing function to the weights of the graphs, because the ordering of edges is then preserved.

A slightly different problem is the one of the minimum spanning forest (MSF), which consists in finding an optimal forest with the constraint that each tree of the forest contains a root belonging to a collection of nodes called roots.

Definition 1.27 (Minimum Spanning Forest (MSF))**.** Let $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta)$ be a connected edge-weighted graph. A minimum spanning forest rooted on a set of k distinct nodes $\{t_1, ..., t_k\}$ is a spanning forest $\mathcal{MSF} = (\mathsf{V}, \mathsf{E}_{\mathcal{MSF}})$ of $\mathcal{G}$ such that each tree of $\mathcal{MSF}$ contains exactly one root t_i, and such that the sum of the edges weights of $\mathcal{MSF}$ is minimal.

$$\mathcal{MSF}(\mathcal{G}) \coloneqq \underset{\mathcal{F} \in \boldsymbol{\mathcal{SF}}_{t_1,...,t_k}}{\arg\min} \left(\sum_{\mathsf{e}_{i,j} \in \mathsf{E}_{\mathcal{SF}}} \eta_{i,j} \right) \tag{1.7}$$

Where $\boldsymbol{\mathcal{SF}}_{\{t_1,...,t_k\}}$ denotes the set of all spanning forests of $\mathcal{G}$ rooted in the set $\{t_1, ..., t_k\}$.

An example of use of this is the marker-based segmentation on a graph [Meyer (1994a)]. It corresponds to a $\mathcal{MSF}$ such that each tree is rooted in a marked node.

1.3.4 Characterizations of MST and MSF

In this section, we present useful characterizations of MST and MSF. These theorems are important as they highlight links between structures appearing in graph optimization problem. Indeed, MST and MSF present interesting properties regarding the paths and cuts induced by these structures. The problems linked with spanning trees are among the oldest ones in graph theory and have been studied intensively. We refer the reader to the textbook by Gondran and Minoux [Gondran et al. (1984)] for proofs and detailed explanations of the results hereby presented.

Theorem 1.28 (Path optimality [Hu (1961)]). *A spanning tree $\mathcal{MST} = (\mathsf{V}, \mathsf{E}_{\mathcal{MST}})$ of a graph $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta)$ is a minimum spanning tree if and only if it satisfies the following inequality:*

$$\forall \mathsf{e}_{k,l} \notin \mathsf{E}_{\mathcal{MST}}, \forall \mathsf{e}_{i,j} \in \pi, \eta_{k,l} \geq \eta_{i,j}, \tag{1.8}$$

where π is the unique path from node k to node l in the $\mathcal{MST}$.

Stated otherwise, if an edge e_{pq} does not belong to the $\mathcal{MST}$, then the weight η_{pq} is superior or equal to the weight of all edges belonging on the path connecting p and q in the $\mathcal{MST}$.

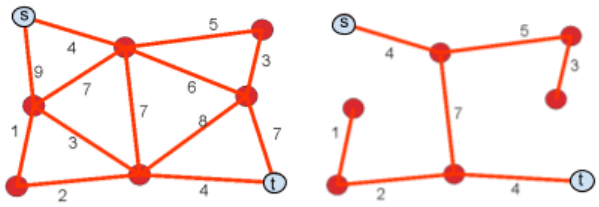


Figure 1.8 – Path optimality: all paths linking the nodes s and t in the graph have a highest edge with a valuation superior to 7, which is the valuation of the highest edge of the path linking these nodes in the MST.

Definition 1.29 (Graph cut). Since there is a unique path connecting two nodes in a spanning tree, deleting one of its edges creates two distinct connected components. If we denote $\mathsf{e}_{i,j}$ the edge that has been cut, the set of edges of $\mathcal{G}$ connecting these two components is a *graph cut* separating nodes i and j. We say that $C_{(i,j)}$ is a cut induced in the graph $\mathcal{G}$ by deleting the edge $\mathsf{e}_{i,j}$ in a spanning tree $\mathcal{ST}$ of $\mathcal{G}$.

Theorem 1.30 (Cut optimality [Gondran et al. (1984)]). *A spanning tree $\mathcal{MST} = (\mathsf{V}, \mathsf{E}_{\mathcal{MST}})$ of a graph $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta)$ is a minimum spanning tree if and only if it satisfies the following condition:*

$$\forall \mathsf{e}_{i,j} \in \mathsf{E}_{\mathcal{MST}}, \forall \mathsf{e}_{k,l} \in C_{(i,j)}, \eta_{i,j} \leq \eta_{k,l}, \tag{1.9}$$

where $C_{(i,j)}$ is the cut induced in the graph $\mathcal{G}$ by deleting the edge $\mathsf{e}_{i,j}$ in $\mathcal{MST}$.
Stated otherwise, if $\mathcal{T}_i$ and $\mathcal{T}_j$ are the subtrees obtained by cutting the edge e_{ij} of the $\mathcal{MST}$, then $\eta_{ij} \leq \eta_{pq}$ for any edge e_{pq} linking a node p of $\mathcal{T}_i$ with a node q of $\mathcal{T}_j$, i.e. belonging to the cocycle Cocycle$(\mathcal{T}_i, \mathcal{T}_j)$.

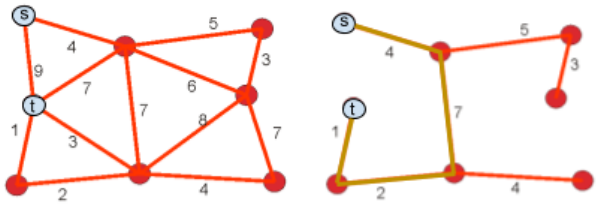


Figure 1.9 – Cut optimality: if we consider the edge e_{st} of the graph $\mathcal{G}$, which is not in the $\mathcal{MST}$ of the graph, we see that its valuation (9) is superior to all edges valuations on the path linking nodes s and t on the $\mathcal{MST}$ (in orange).

The path optimality property of a minimum spanning tree allows for the computation of a *minimax* path between two nodes of the graph, i.e. a path for which the maximum edge weight along it is minimal.

Theorem 1.31 (Minimax path [Gondran et al. (1984); Hu (1961)])**.** *A spanning tree $\mathcal{MST} = (V, E_{\mathcal{MST}})$ of a graph $\mathcal{G} = (V, E, \eta)$ is a minimum spanning tree if and only if for all distinct pairs of nodes (p, q) of $\mathcal{G}$, the unique path π^* between p and q in $\mathcal{MST}$ is a minimax path:*

$$\pi^* \in \arg\min_{\pi \in \mathbf{\Pi}_{(p,q)}} (L_\infty(\pi)) \tag{1.10}$$

where $\mathbf{\Pi}_{(p,q)}$ is the set of all paths from node p to node q in $\mathcal{G}$.

The minimax path corresponds to the path of the lowest distance (cf. definition 1.15). Indeed, the lowest distance between two nodes p and q of a graph is the maximal edge weight on the path linking p and q in a MST of the graph.

As a direct consequence of theorems 1.28 and 1.30, the minimum spanning tree of a graph only depends on the ordering of edges valuations by increasing values. In cases where there are duplicated edges valuations, there can exist several MST of the same graph. Figure 1.10 illustrates this possibility.

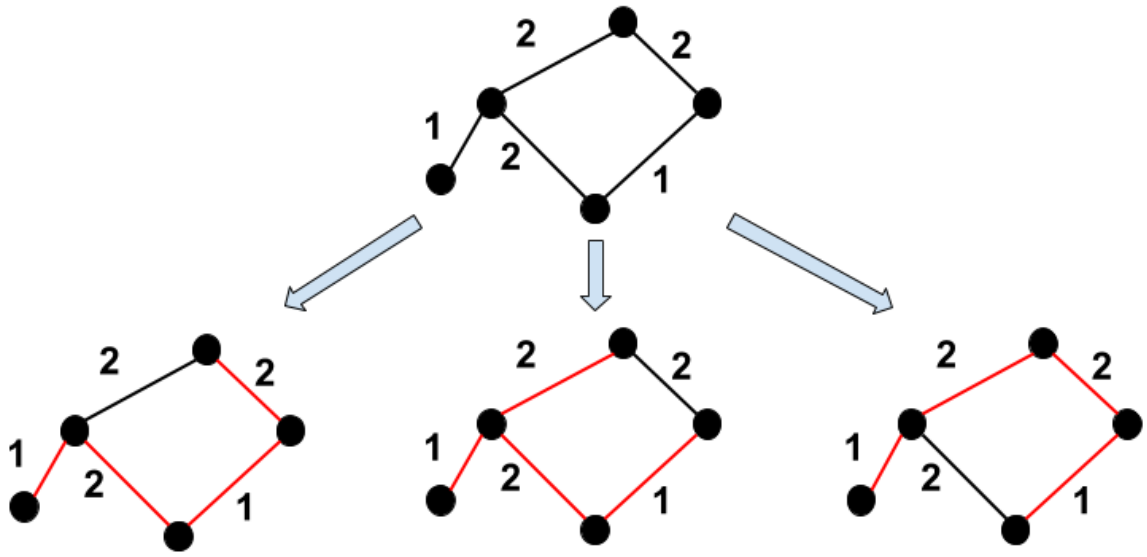


Figure 1.10 – When several equivalent choices are possible, there are several minimum spanning trees for the same graph. All these MST then have the same distribution of edges weights.

Remark 1.32 (Multiple minimum spanning trees case)**.** When having multiple minimum spanning trees of the same graph, there is a one-to-one correspondence between the edges numbers for each given edge valuation.

Stated otherwise, if we consider two MST $\mathcal{MST}_1$ and $\mathcal{MST}_2$ of the same graph $\mathcal{G}$, and if we order their edges by decreasing order of their weights, there is a one-to-one correspondence between edges of $\mathcal{MST}_1$ and edges of $\mathcal{MST}_2$, corresponding edges holding the same weights [Berge (1985)].

One may wonder whether the fact that the minimum spanning trees can be multiple may hamper robustness. In section 3.3.5, we will see that it is when doing image segmentation using them, and we will propose a way to handle such an instability by considering the union of all minimum spanning trees before pruning the result to get a single spanning tree.

1.3.5 Algorithms to compute MST

In this section we briefly list, explain and compare the main different algorithms that can be used to get a minimum spanning tree for a given connected edge-weighted graph.

The most popular algorithms for extracting MST from graphs are Boruvka's algorithm [Boruvka (1926)], Kruskal's algorithm [Kruskal (1956)] and Jarnik's/Prim's/Dijkstra's algorithm [Jarnik (1930); Prim (1957); Dijkstra (1959)].

Dijkstra/Jarnik/Prim (DJP)

The Jarnik's/Prim's/Dijkstra's algorithm has been developed by Jarnik in 1930 [Jarnik (1930)] and simultaneously rediscovered by Prim and Dijkstra in 1959 [Prim (1957); Dijkstra (1959)].

In the following we refer to it as to DJP algorithm.

It forms the MST by finding the lowest adjacent edge to the tree that has been formed by beginning from an arbitrary vertex and adding it to the tree, such that the edge to be added does not form a cycle. A simple way to check that no cycle is created is to make use of a label that is propagated from one node to the other.

Algorithm 1.1: DJP's algorithm to compute MST.

Data: A graph $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta)$;
Result: A $\mathcal{MST}$ of $\mathcal{G}$
Choose arbitrarily a start node s ;
Initially, $S = \{s\}$ and $\mathcal{MST} = \{\}$;
while $S \neq \mathsf{V}$ **do**
    Find an edge e such that:
        1. e starts in S and ends out of S
        2. e has the minimal weight of edges satisfying 1.
    Add e to $\mathcal{MST}$;
    Add the vertex at the end of e to S
end while

Kruskal

Kruskal's algorithm forms the MST by processing the edges in ascending order. To obtain a MST, each edge is added unless a cycle is created. An illustration of Kruskal's algorithm is provided in figure 1.11.

Algorithm 1.2: Kruskal's algorithm to compute MST.

Data: A graph $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta)$;
Result: A $\mathcal{MST}$ of $\mathcal{G}$
Initially, sort edges in ascending order of weight;
Set $\mathcal{MST} = \{\}$;
for *each edge* $\mathsf{e} \in \mathsf{E}$ **do**
    if $\tilde{\mathcal{G}} = (\mathsf{V}, \mathcal{MST} \cup \{\mathsf{e}\})$ *does not contain a cycle* **then**
        Add e to the $\mathcal{MST}$
    end if
end for

Note that a version of Kruskal exists that does not require edges to be ordered, following the process hereby:

— Add one edge at a time,

— If by adding an edge one create a cycle, withdraw the maximal weight edge in this cycle.

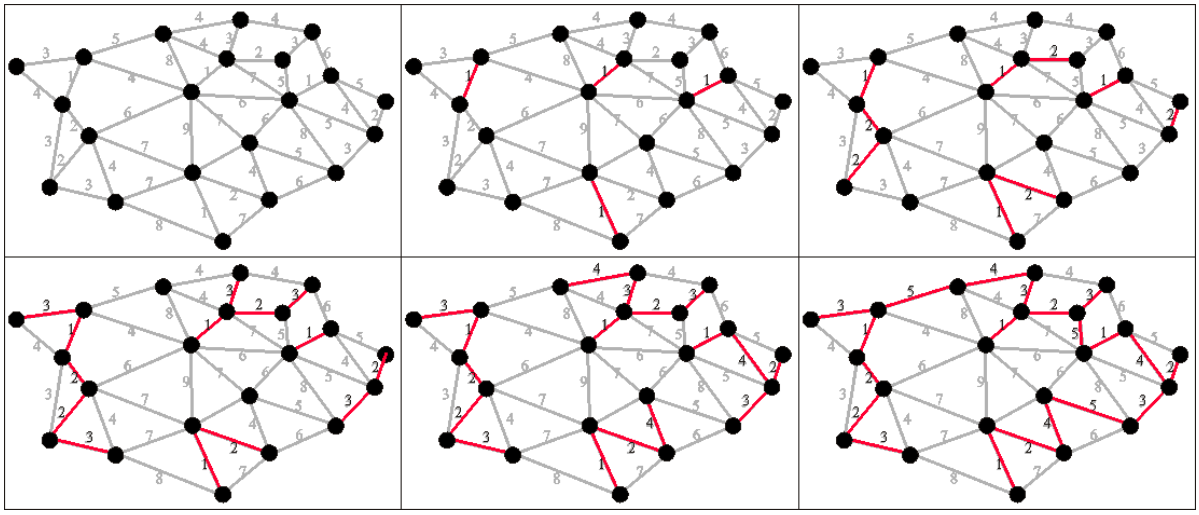


Figure 1.11 – Illustration of a minimum spanning tree computation using Kruskal's algorithm (Left to right, and top to bottom).

Boruvka

Boruvka's algorithm is similar to Kruskal's algorithm. It begins by making a set of trees that start out as single vertices. It then repeatedly iterates over these trees to find for each one the chepeast edge that has one node in the tree and the other not. Then it adds these edges to the $\mathcal{MST}$, and merge the trees they join.

The difference with Kruskal's algorithm is that Boruvka, at each iteration, finds all at once for all trees the cheapest edges that have endpoints in different trees, of processing the edges one by one following the increasing order of their weights.

Algorithm 1.3: Boruvka's algorithm to compute MST.

Data: A graph $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta)$;

Result: A $\mathcal{MST}$ of $\mathcal{G}$

1 Initialize all vertices as individual components (or sets) ;
2 Initialize $\mathcal{MST}$ as empty ;
3 **repeat**
4 Consider next component ;
5 Find the closest weight edge that connects this component to any other component ;
6 Add this closest edge to $\mathcal{MST}$ if not already added
7 **until** *There are more than one components*;

1.4 Clustering on Graphs

Clustering is the process of grouping together objects that are similar to form *clusters*. It can be seen as a particular case of classification in which clusters constitute the different classes of objects. It is widely used, with applications from biology and astronomy to economics or image analysis.

1.4.1 Definitions

We hereby present formal definitions relative to the problem of clustering.

Definition 1.33 (Partition)**.** A *partition* is a family of disjoint sets covering the whole space. Let $\mathcal{S}$ be a finite set and $\mathsf{P}(\mathcal{S})$ be the set of all subsets of $\mathcal{S}$. A partition P of $\mathcal{S}$ is a family (P_i) of elements of $\mathsf{P}(\mathcal{S})$ such that:

— (i) $\forall i \neq j, \mathsf{P}_i \cap \mathsf{P}_j = \emptyset$

— (ii) $\cup \mathsf{P}_j = \mathcal{S}$

Property 1.34 (Equivalence classes and partitions)**.** The set of all equivalence classes of an equivalence relation R on $\mathcal{S}$ is a partition of $\mathcal{S}$. Conversely, any partition can be identified as the equivalence classes of an equivalence relation.

Proof. Given an equivalence relation R on $\mathcal{S}$, the reflexivity of R ensures that every element x belongs to at least one equivalence class: its own. Transitivity ensures that two equivalence classes are either confused or disjoint, i.e. that two classes having a non-void intersection are identical.
Conversely, given a partition P of $\mathcal{S}$, we can associate an equivalence relation R to this partition that is defined by: $xRy \Leftrightarrow (\exists \mathsf{P}_i \in \mathsf{P} \text{ s.t. } (x, y) \in \mathsf{P}_i^2)$. Verifying that R is an equivalence relation is trivial. □

A taxonomy of clustering techniques

Following the taxonomy from [Jain et al. (1999)], we hereby introduce the different properties a clustering technique can verify:

1. It can be *exclusive* or *nonexclusive*.
   Exclusive clustering techniques yields clusters that are disjoint, while nonexclusive ones can lead to overlapping clusters. An exclusive clustering method generates a partition of the set of objects, and most of existing methods fall within this category.
2. It can be *intrinsic* or *extrinsic*.
   Intrinsic clustering is an unsupervised activity based only on the dissimilarities between the objects that we want to cluster. Extrinsic clustering, on the opposite, can rely on external information provided for example to specify which objects should be clustered together and which ones should not.
3. It can be *hierarchical* or *partitional*.
   Partitional clustering aims at generating a partition of the graph, either by relying on its internal structure in the case of intrinsic partitional clustering, or by introducing external information regarding the desired output in extrinsic partitional clustering.
   Hierarchical clustering algorithms aim at the construction of a sequence of nested partitions. More specifically, *hierarchical agglomerative clustering* (bottom-up approach) are built

starting from a set of objects whose blocks consist in single objects, and in which certain clusters are progressively merged together. Thus the result is an ascending chain of partitions (considering the inclusion as order relation), the first one being the partition of all separated objects and the last one the partition where all objects are merged. In *hierarchical divisive clustering* methods (top-down approach), it is the opposite, and successive partitions are built by subdividing blocks of the previous partition. Agglomerative clustering methods are generally considered to be more efficient since such bottom-up approaches bound the complexity to polynomial at worst, while divisive clustering methods suppose a combinatorial search of every possible division of each cluster, thus potentially leading to an exponential complexity. But as we may see in the following, these two approaches are equivalent when deriving hierarchical clustering from minimum spanning trees of the graphs.

In this thesis, we will mainly develop new clustering methods that are exclusive, intrinsic/extrinsic and hierarchical. We will delve into hierarchical clustering in chapter 2, and for now focus on partitional clustering in this section.

The aim of partitional clustering is to create a partition of the set of objects whose blocks are the clusters, such that objects in a cluster are more similar to each other than to objects belonging to different clusters, given a way to compare these clusters.

1.4.2 Unsupervised approaches

In this section we make a brief survey of partitional clustering, which can be seen as the unsupervised classification of patterns. As a result of its appeal and usefulness for a wide range of applications, especially regarding exploratory data analysis, this problem has been addressed in many contexts and disciplines, image segmentation being only one of them

An inherent problem accompanying partitioning algorithms is the choice of the desired numbers of clusters in the output. These methods usually produce clusters by optimizing a criterion function which can be defined either locally (on a subset of the features describing our data) or globally (defined over all features). Furthermore, since the combinatorial search of the best set of clusters among all possibilities is often computationally unfeasible, heuristics are used. Either the algorithm runs multiple times with different starting states and the best result is used, or an iterative optimization is operated, meaning a relocation schemes iteratively reassign points between the k clusters.

Probabilistic clustering

In a probabilistic approach, we can consider data to be a sample independently drawn from a mixture model of several probability distributions. The main assumption in such a framework is that data points are generated by randomly picking a model among the existing models and then drawing those points from the corresponding distributions. Thus, the overall likelihood of

the training data can be defined as its probability to be drawn from a given mixture model, and it can serve as an objective function that one wants to maximize.

This gives rise to the Expectation-Maximization (EM) method, described in details with references and examples in [Dempster et al. (1977); McLachlan et al. (2007)]. EM is a two-steps iterative optimization method: step (E) estimates for each data point its probability to be in each cluster, eventually leading to a (soft) reassignment of this point, step (M) then refines the approximation of current soft assignments to a mixture model. This process leads to a maximization of the likelihood, and finishes when it converges.

k-means methods

In the k-means algorithm, we represent each of the k clusters by the mean of its points, called *centroid*. The sum of discrepancies between a point and its centroid can then be expressed using appropriate distance function: when using the L_2-norm as a distance, the corresponding sum is the sum-of-squares error (SSE). This SSE corresponds to the total intra-clusters variance, and can be used as an objective function to minimize. Note that the SSE can be seen as the negative of the log-likelihood for normally distributed mixture model and thus can be derived from the probabilistic framework [Dempster et al. (1977)]. This is why Forgy's algorithm, one of the more popular versions of k-means algorithm, is very similar to EM algorithm and consists in two steps [Forgy (1965)]: (i) assigning each data point to its nearest centroid, (ii) recompute centroids of newly assembled groups.

In a graph, a dissimilarity is defined between neighboring nodes only, and thus defining the mean of a cluster is not obvious. This is why from a practical point-of-view, K-medoids methods are more often used when it comes to graph approaches.

k-medoids methods

In k-medoids methods, a cluster is represented by one of its points. A k-medoid is the most representative data point of a cluster of points, in the sense that it has the most central position in the cluster relative to all other members of the cluster. Such a representation presents two advantages. It is less sensitive to the presence of outliers since its location depends on the location of a predominant fraction of the points. It also presents no limitations regarding the attributes types that are considered. Once medoids are selected, clusters are defined as subsets of points close to respective medoids and thus the objective function is defined as the sum of distances between the points and their medoids. In a graph, several distances can be defined and a min-max approach can be used. The sum of distances is not always meaningful, but we can look fr the node such that its maximal distance to other nodes is minimal.

The more popular variant of k-medoids methods is called PAM ("Partition Around Medoids") [Kaufman et al. (2009)]. Just like in the two previously mentioned approaches, it can be decomposed in two phases. At first, medoids are selected and each data point is assigned to a cluster represented by the medoid of which it is the closest to: this is the *building phase*. Then

during the *swapping phase*, swapping data points and existing medoids is considered: medoids and nearby data points are iteratively swapped as long as these swaps do not diminish the centrality of the medoids. Note that the PAM algorithm is more robust than Forgy's variant of k-means as it minimizes the sum of the dissimilarities instead of the sum of the squared errors. More details on successive versions of k-medoids algorithms are provided in [Berkhin (2006)].

Spectral Graph Partitioning

Traditional clustering methods like k-means suppose that the points we want to regroup can be clustered using a spherical or elliptical metric. Hence they do not work when the clusters are non-convex. Spectral graph partitioning proposes a solution in such cases, as described in details in [Von Luxburg (2007)].

This method starts from a matrix of pairwise similarities of size $N \times N$, representing an edge-weighted graph with N nodes, with non-null values for pairs of connected nodes, and null values elsewhere. The idea is then to build similarity graphs that represent the local neighborhood relationships between points. It proceeds by finding the eigenvectors corresponding to the smallest eigenvalues of the Laplacian matrix associated with the matrix of pairwise similarities of the graph. Using a standard method like k-means, it then clusters the matrix of eigenvectors to yield to a clustering of the original data points.

Minimal Cuts Approaches

The minimum spanning tree structure is commonly used in graph-based algorithms for ease of use and speed. When working with the minimum spanning tree $\mathcal{MST}$ of a graph $\mathcal{G}$ (cf. definition 1.26), there are different ways to produce a group of clusters. When the desired number k of clusters is known, one can simply sort edges of the minimum spanning tree by decreasing order of their weights, and remove the edges with the first $k-1$ heaviest weights [Xu, Olman, et al. (2001); Asano et al. (1988)]. We refer to this method as to the *minimal cuts* method. It comes down to a minimization problem where the optimization criterion is the diameter of the remaining clusters, the diameter of a forest being defined as the valuation of its highest edge. Indeed, the clustering forest created by cutting the k highest edges in the MST is a forest with the lowest possible diameter among all possible clustering forests. Note that another minimal cuts approach exists that falls within the flow algorithms, and where the goal is to minimize the sum of valuations of edges belonging to cocycles between the source(s) and the sink(s) [Boykov et al. (2004)].

Such approaches are way faster than other methods proposed in the literature, such as normalized cuts [Shi et al. (2000); Wang, Jia, et al. (2008)], spectral graph partitioning [Hagen et al. (1992); Ng et al. (2002)] (even if parallelized versions recently appeared such as in [Naumov et al. (2016)]). Indeed, although these methods allow to get more balanced partitions by mainly minimizing a ratio between the length of cuts and the areas enclosed in them, they require more complex and costly tools such as linear algebra tools to compute eigenvalues and

vectors of large matrices. However, minimal cuts suffer from a shrinking problem: it often produces small cuts which are often not relevant.

Improvements have been proposed to the minimal cuts approach for situations when the desired number of clusters is not known in advance, with defined conditions regarding the edges that need to be cut. In [Zahn (1971)], Zahn defines *inconsistent* edges, basically depending on whether each edge connects two regions that are non-coherent between themselves but coherent in themselves. In [Felzenszwalb et al. (2004)], the authors define a predicate to measure the evidence between neighboring regions, using not only local information but also global information regarding the regions, such as cluster size and longest edge for each cluster.

In [Wang, Zhang, et al. (2014); Saglam et al. (2017)], the authors propose sequential clustering algorithms using the sequential edge list formed during Prim's algorithm to define inconsistent edges, thus creating a set of inconsistent edge queues. Then, depending on the desired number of clusters k, one of these queues is selected.

Watershed approaches

By definition, the watershed transform [Lantuéjoul et al. (1981)] is the SKIZ of regional minima and it can be defined for different distances, for example, and by order of precision, the ultrametric distance, the topographic distance or the lexicographic distance.

It has been applied with success to a number of partitioning problems, mainly in image partitioning tasks. In the classical representation of this technique, images are represented as topographic reliefs, which can be flooded. More precisely, we departure from a gradient of the image considered as a relief map, and minima of the image constitute the seeds of the regions that will be separated by watershed lines in the final segmentation. One can also impose markers to govern the process: instead of considering all minima, one can choose to consider only those of markers images as flooding sources.

We can interpret such an approach on a graph [Meyer (2014)]. Indeed, defining an equivalence relation on a graph, linked with a path-connectedness property, leads to obtaining influences zones skeletons, namely here partitions. Furthermore, there is then a natural connexion between watershed and minimum spanning forests algorithms, as we will see in this thesis.

1.4.3 Semi-supervised approaches

When trying to realize a clustering of a graph in homogeneous subgraphs, we often have prior information in the form of labels for given nodes of the graph called *markers* or *seeds*, so that we know in advance for example which are the nodes that we want to be separated in the output clustering. We are thus in the case of a semi-supervised clustering problem, which can also be formulated as a problem of clustering with constraints. We would like to make use of hints or advices in the form of constraints to guide the clustering process. This problem has been formalized notably in [Davidson et al. (2005); Basu et al. (2008)]. The authors remind that the most prevalent form of constraints are conjunctions of pair-wise constraints of the form must-link

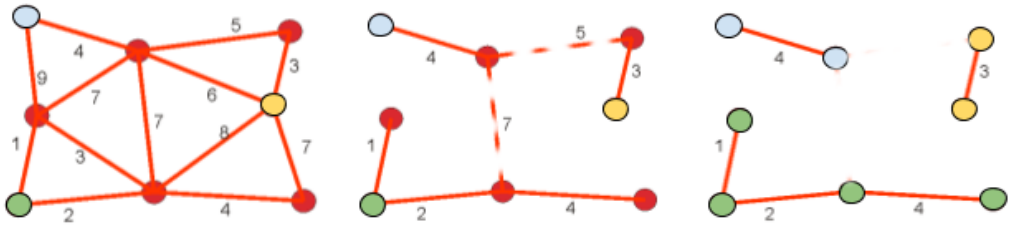


Figure 1.12 – **Left**: a graph with markers; **Middle**: we compute a MST of the graph, and cut the highest edges on the paths linking pairs of marked nodes; **Right**: final clustering.

(ML) and cannot-link (CL), which state that pairs of points should respectively be in the same or in different clusters. They also draw the two main paths for using such an information in clustering approaches: either one can use a standard dissimilarity and then try to satisfy all or as many constraints as possible, or one can use these constraints to learn a distance function that can then be used in a clustering algorithm. As we shall see throughout this thesis, these two paths can cross, as finding an organization of data points satisfying the constraints can help to define a distance function more suitable to the corresponding clustering problem.

Such an approach can also be interpreted as a Voronoi-like one: given markers, we would like to attribute to each node the label of its closest marker for a given distance (amongst the ones defined in section 1.2). This is why a standard morphological way to propose a solution to such a problem can easily be obtained and explained in a MST-based framework using the lowest path distance [Gomila (2001); Gomila and Meyer (2003)]. One can indeed select a node in each region or object of interest that will serve as a root in each wanted tree. We then construct a MSF in which each tree takes root in one of the selected nodes. The roots are also called *markers* and this process referred to as *marker-based segmentation* or *MSF segmentation*. The final result is obtained by suppressing, for each pair of markers, the highest edge on the unique path on the MST linking them. This is illustrated in figure 1.12.

In [Couprie et al. (2011)], the authors build on the works in [Sinop et al. (2007)] to provide proofs of connexions between the MSF algorithm and several other state-of-the-art approaches for semi-supervised graph partitioning. Indeed, they show that the watershed transform/markers-based segmentation, graph-cuts, random walker and shortest paths algorithms can all be seen as providing solutions to variations of a similar minimization problem. They also give insights regarding the performances of each approach. Graph cuts provide good results when the seeds are far from the boundary, but is slow and thus not adapted to heavy problem such as 3D images segmentation. Shortest paths is useful when the object to segment is well centered around foreground and background seeds. Random walkers is globally efficient and provides good results whether the markers are equidistant or strongly asymmetric. Minimum Spanning Forests algorithms are useful especially because of their robustness to markers not centered, since they

are not overdependent on the markers position. They also present the advantage of being very fast to obtain.

In the following chapter, we will show how we can easily extend the MSF framework to obtain hierarchies of segmentations, present some theoretical properties as long as a glimpse into the versatility of such an approach.

Chapter 2

Hierarchical Clustering on Graphs

Résumé

Ce chapitre s'intéresse spécifiquement à la classification hiérarchique. Nous commençons par introduire définitions, notations et propriétés concernant les hiérarchies, leur lien avec les distances ultramétriques, et leur représentation sous forme d'un dendrogramme. Dans le cas d'un graphe valué aux arêtes, nous voyons ensuite comment la structure de son arbre de poids minimum induit intrinsèquement une classification hiérarchique de ses noeuds. Enfin, nous montrons que l'espace de ces hiérarchiques admet une structure de treillis, avec l'existence d'une relation d'ordre, d'opérateurs permettant de combiner les hiérarchies et d'une distance permettant de géométriser leur espace.

Abstract

This chapter focuses specifically on hierarchical clustering. We begin by introducing definitions, notations and properties concerning hierarchies, their relation to ultrametric distances, and their representation as dendrograms. In the case of an edge-weighted graph, we then see how the structure of its minimum spanning tree inherently induces a hierarchical clustering of its nodes. Finally, we show that the space of these hierarchies admits a lattice structure, with the existence of an order relation, operators for combining hierarchies and a distance for assigning a geometric structure to the space of hierarchies.

2.1 Introduction

Throughout this thesis, we will present methods for segmenting images as a step for solving various tasks. Ideally, a segmentation of an image extracts all objects of interest for a later task, which oftens implies to understand the semantic meaning of the images. When there are only a few object types and the images to segment are obtained under controlled conditions, it is often possible to directly detect and segment them. But in most situations, we need to proceed by successive steps. Most of the methods we will present are based on a representation of the image

as a graph, in which each node corresponds to a region of the image, and edges connect adjacent regions of a partition. This partition is called fine partition, from which coarser partitions will be derived. Since there is a huge amount of information represented on such a connected graph, one can make progressively appear intermediate structures by successive cuts of the edges in a particular order. Such an approach is said to be divisive, or top-down. On the opposite, one can first extract homogeneous regions of the image according to local cues such as color, texture, movement, and then progressively merge these regions to get more semantically significant ones. This approach is agglomerative, or bottom-up.

In this chapter, we begin by introducing the mathematical structures and notations useful to describe such constructions, namely the notion of hierarchy. Hierarchies are indeed derived from classical structures for representing a taxonomy, that is to say a classification of objects of a set into ordered categories. For example, the Linnaean system classifies nature into a series of nested classes [Linné et al. (1758)]. Such structures are also encountered in image segmentation, as was stated above. We will place ourselves in the particular case where the dissimilarity is carried by the edges and expressed by weights.

Then, we introduce our model for representing the hierarchies, namely edge-weighted graphs, in which the nodes represent regions of a fine partition, edges link neighboring nodes and edge weights expressing the dissimilarity between the nodes. From this seminal representation are derived additional structures such as the minimum spanning tree and the dendrogram.

2.2 Definitions and Properties

2.2.1 Dendrogram

In many problems, it appears interesting to structure a set of objects into ordered categories: this is the object of a taxonomy. A well-known one is the Linnaean system [Linné et al. (1758)], which classifies the set of all parts of nature: a first decomposition is made in three classes (animal, vegetable, mineral), which can themselves be decomposed into several subclasses, and so on. The proper mathematical structure to represent such taxonomies/hierarchies is the dendrogram, of which we introduce the axiomatic in this section.

Definitions

Dendrograms are tree structures allowing to represent series of nested mergings of a family of objects, and which are therefore particularly adapted to the representation of hierarchical clusterings. We hereby present axiomatic definitions of dendrograms for a given order relation on the space, taken from [Benzécri et al. (1973)] and which entirely rely on the set intersection or union, and on the inclusion order relation between sets. Their construction is progressive: starting with this inclusion order relation, we then add axioms to define trees, hierarchies and stratified hierarchies.

Definition 2.1 (Structure associated with an order relation)**.** Let S be a set, $\mathcal{P}(S)$ be the set of all subsets of S, $\mathcal{X}$ be a subset of $\mathcal{P}(S)$ on which we consider an arbitrary order or preorder relation $\prec$ (for example the inclusion between sets $\subset$). The union of all sets belonging to $\mathcal{X}$ is called the *support* of $\mathcal{X}$. The subsets of $\mathcal{X}$ can be structured into:

— the summits: $\text{Sum}(\mathcal{X}) = \{A \in \mathcal{X} | \forall B \in \mathcal{X} : A \prec B \Rightarrow A = B\}$

— the leaves: $\text{Leav}(\mathcal{X}) = \{A \in \mathcal{X} | \forall B \in \mathcal{X} : B \prec A \Rightarrow A = B\}$

— the nodes: $\text{Nod}(\mathcal{X}) = \mathcal{X} \setminus \text{Leav}(\mathcal{X})$

— the predecessors, or ancestors: $\text{ancestor}(A) = \text{Pred}(A) = \{B \in \mathcal{X} | A \prec B\}$

— the immediate predecessor, or father: $\text{father}(A) = \text{ImPred}(A) = \{B \in \mathcal{X} | \{U | U \in \mathcal{X}, A \prec U \text{ and } U \prec B\} = \{(A, B)\}$

— the successors, or descendants: $\text{descendant}(A) = \text{Succ}(A) = \{B \in \mathcal{X} | B \prec A\}$

— the immediate successors, or sons: $\text{son}(A) = \text{ImSucc}(A) = \{B \in \mathcal{X} | \{U | U \in \mathcal{X}, B \prec U \text{ and } U \prec A\}\} = \{(A, B)\}$

— the cousins of A are the sons of an ancestor of A that are not themselves ancestors of A: $\text{cousin}(A) = \{B \in \mathcal{X} | B = \text{son}[\text{ancestor}(A)]; B \neq \text{ancestor}(A)\}$

We can then structure $\mathcal{X}$ using the tree structure of a dendrogram. Another possible name for dendrogram is “partial hierarchy”.

Definition 2.2 (Dendrogram)**.** $\mathcal{X}$ is a *dendrogram* if and only if the order relation induced by $\prec$ is a total order over the set $\text{Pred}(A)$ of the predecessors of A. Then the maximal element of this family is a summit: the unique summit containing A.

There exist several interesting characterizations of dendrograms.

Proposition 2.3 (Dendrograms characterizations)**.** The following properties are equivalent:

1. $\mathcal{X}$ is a dendrogram
2. $(U, V, A) \in \mathcal{X}^3 : A \subset U \text{ and } A \subset V \Rightarrow U \subset V \text{ or } V \subset U$
3. $(U, V) \in \mathcal{X}^2 : (U \not\subseteq V \text{ and } V \not\subseteq U) \Rightarrow U \cap V = \emptyset$

Representation of a dendrogram as a tree

When $\mathcal{X}$ is finite, $\mathcal{X}$ is a dendrogram if and only if any element $A \in \mathcal{X} - \text{Sum}(\mathcal{X})$ possesses a unique immediate predecessor.

We say that a dendrogram is *connected* if it possesses a unique summit.

A finite dendrogram is usually represented as a tree: each element $A \in \mathcal{X}$ is a node of the tree, and is linked by an edge with its unique immediate predecessor. An example of such a representation is provided in figure 2.1.

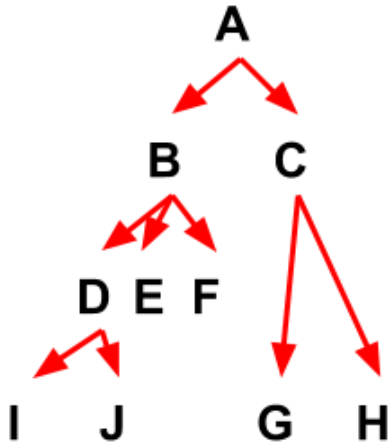


Figure 2.1 – A dendrogram.

Stratification indices

In a dendrogram, a node is included in all its predecessors, and this partial order relation governs the hierarchical structure of the tree. We can make this inclusion order more precise by the adjunction of a total order compatible with it.

Definition 2.4 (Stratified dendrogram)**.** A dendrogram $\mathcal{X}$ is *stratified* if there exists a *stratification index* $\mathbf{st} : \mathcal{X} \mapsto [O, L]$ which is strictly increasing with the inclusion order:

$$\forall (A, B) \in \mathcal{X}^2 : A \subset B \text{ and } B \neq A \Rightarrow \mathbf{st}(A) < \mathbf{st}(B).$$

Since E is finite, the number of distinct stratifications levels is finite, so:

$$\forall A \in \mathcal{X} : \mathbf{st}(A) < L \text{ and } \mathbf{st}(\emptyset) = L.$$

We can think of the stratification index as giving the level of the stratifications for which nodes progressively merge. It appears that many stratification indices are compatible with a given dendrogram, since it is the case as long as they properly reflect the order of fusions. Indeed, two stratification indices can be radically different but both be compatible with the partial order induced by the order relation expressed by the dendrogram (for example the inclusion order when working with sets).

2.2.2 Hierarchy

Definition 2.5 (Hierarchy)**.** Let S be a set and $\mathcal{P}(S)$ be the set of all subsets of S. A *hierarchy* on the set S is a collection of sets $\mathcal{H} \subseteq \mathcal{P}(S)$ that satisfies the following conditions:

1. the members of $\mathcal{H}$ are non empty
2. $S \in \mathcal{H}$

3. $\forall x \in S, \{x\} \in \mathcal{H}$
4. if $(H, H') \in \mathcal{H}^2$ and $H \cap H' \neq \emptyset$, then we have either $H \subseteq H'$ or $H' \subseteq H$

A hierarchy is a particular case of dendrogram. Indeed, $\mathcal{H}$ is a hierarchy if it is a dendrogram such that: $\cup \text{Leav}(\mathcal{H}) = S$.

Proposition 2.6. A dendrogram $\mathcal{X}$ is a hierarchy if and only if it verifies the *union axiom*:
(**Union axiom**) Any element A of $\mathcal{X}$ is the union of all other elements of $\mathcal{X}$ contained in A:

$$\forall A \in \mathcal{X} : \cup\{B \in \mathcal{X} | B \subset A; B \neq A\} = \{A, \emptyset\}$$

2.2.3 Ultrametrics

Dissimilarities are functions allowing us to evaluate the extent to which data objects are different.

Definition 2.7 (Dissimilarity)**.** A *dissimilarity* on a set S is a function $d : S^2 \to \mathbb{R}_{\geq 0}$ satisfying the following conditions:

1. $\forall x \in S, d(x, x) = 0$;
2. $\forall (x, y) \in S^2, d(x, y) = d(y, x)$

The pair (S, d) is a dissimilarity space.
Additional properties may be satisfied by dissimilarities. A non-exhaustive list is given here:

1. $\forall (x, y) \in S^2, d(x, y) = 0 \Rightarrow \forall z \in S, d(x, z) = d(y, z)$ (*evenness*)
2. $\forall (x, y) \in S^2, d(x, y) = 0 \Rightarrow x = y$ (*definiteness*)
3. $\forall (x, y, z) \in S^3, d(x, y) \leq d(x, z) + d(z, y)$ (*triangular inequality*)
4. $\forall (x, y, z) \in S^3, d(x, y) \leq max(d(x, z), d(z, y))$ (*ultrametric inequality*)
5. $\forall (x, y, u, v) \in S^4, d(x, y) + d(u, v) \leq max(d(x, u) + d(y, v), d(x, v) + d(y, u))$ (*Buneman's inequality*, or *four-point condition*)

The properties (3)(4)(5) can be seen as curvature conditions within the metric space, formulated in terms of properties of triangles

Definition 2.8 (Metric, tree metric, ultrametric)**.** A dissimilarity d on a set S is:

1. a *metric* if it satisties the definiteness property and the triangular inequality.
2. a *tree metric* if it satisties the definiteness property and the Buneman's inequality.
3. an *ultrametric* if it satisfies the definiteness property and the ultrametric inequality.

If d is a metric (resp. an ultrametric) on a set S, then (S, d) is a metric space (resp. an ultrametric space).

Theorem 2.9. *Every tree metric is a metric, and every ultrametric is a tree metric.*

Lemma 2.10 (Ultrametric characterization)**.** A function $d : S^2 \to \mathbb{R}_{\geq 0}$ is an ultrametric if and only if it has the following properties:

1. $\forall (x, y) \in S^2, d(x, y) = 0 \Leftrightarrow x = y$
2. $\forall (x, y) \in S^2, d(x, y) = d(y, x)$
3. $\forall (x, y, z) \in S^3, d(x, y) \leq max(d(x, z), d(y, z))$

Definition 2.11 (Open ball, closed ball, radius)**.** Let (S, d) be a metric space. The *closed ball* centered in $x \in S$ of radius r is the set:

$$B(x, r) = \{y \in S | d(x, y) \leq r\} \tag{2.1}$$

The *open ball* centered in $x \in S$ of *radius* r is the set:

$$C_d(x, r) = \{y \in S | d(x, y) < r\} \tag{2.2}$$

Definition 2.12 (Diameter)**.** Let (S, d) be a metric space. The *diameter* of a subset U of S is the number $diam_{S,d} = sup\{d(x, y) | (x, y) \in U^2\}$.

Theorem 2.13 (All triangles are isosceles in ultrametric spaces.)**.** *Let (S, d) be an ultrametric space. For every $(x, y, z) \in S^3$, two numbers $d(x, y)$,$d(x, z)$,$d(y, z)$ are equal and the third is not larger than the two other equal numbers.*
This means that in an ultrametric space, any triangle is isosceles and the side that is not equal to the two others cannot be longer than these.

Proof. Let us consider three distinct points p, q, r, and let us suppose that the largest edge of this triangle is e_{pq}: $d(p, r) \vee d(r, q) \leq d(p, q)$. Then according to the ultrametric inequality, $d(p, q) \leq d(p, r) \vee d(r, q)$. This shows that the two largest edges of the triangle have the same length. □

Lemma 2.14. Let $B(x, r)$ be a closed ball in an ultrametric space (S, d). If $z \in B(x, d)$, then $B(x, r) = B(z, r)$. Stated differently, in an ultrametric space, a closed ball has all its points as centers.

Proof. Let us suppose that y is an element of $B(x, r)$. Let us show that y is then also a center of this ball. If $z \in B(y, r)$: $d(x, z) \leq d(x, y) \vee d(y, z) \leq r$. Hence $z \in B(y, r)$, showing that $B(y, r) \subset B(x, r)$. Exchanging the roles of x and y shows that $B(x, r) = B(y, r)$. □

Lemma 2.15. Two closed balls $B(x, r)$ and $B(y, r)$ with the same radius are either disjoint or identical.

Proof. If $B(x, r)$ and $B(y, r)$ are not disjoint, then they contain at least one common point z. According to lemma 2.14, z is the centre of both balls $B(x, r)$ and $B(y, r)$, showing that they are identical. □

Lemma 2.16. In an ultrametric space, the radius of a ball is superior or equal to its diameter.

Proof. Let $B(x,r)$ be a ball of diameter $\lambda = diam_{B(x,r),d}$. By definition, λ is the maximal distance between two elements of the ball. Let us consider two nodes p and q such that $d(p,q) = \lambda$. Then for any x, $d(x,p) \leq r$, $d(x,q) \leq r$, and thus $\lambda = d(p,q) \leq d(p,x) \vee d(x,q) \leq r$. Hence $\lambda \leq r$. □

Open balls have exactly the same properties, and proofs for establishing them are similar.

2.2.4 Links between hierarchy, ultrametric distance, and dendrogram

We present now the link between an ultrametric defined on a finite set S and chains of equivalence relations on S (or chains of partitions on S).

Let us consider here a connected graph $\mathcal{G}$ with N nodes. Then the set containing all the nodes of the graph is a closed balls with a diameter being the weight of the highest edge in the graph. Let us write $r_{\mathcal{B}}$ (resp. $d_{\mathcal{B}}$) the radius (resp. diameter) of a ball $\mathcal{B}$.

For a ball $\mathcal{B}$, we define $\zeta_{\mathcal{B}}$ as being the infimum of weights of the edges belonging to the cocycle of $\mathcal{B}$: $\zeta_{\mathcal{B}} = \bigwedge\{\eta_{pq} | p \in \mathcal{B}, q \notin \mathcal{B}\}$.

We have $d_{\mathcal{B}} \leq r_{\mathcal{B}} < \zeta_{ball}$, and the smallest closed ball strictly containing the ball $\mathcal{B}(p, d_{\mathcal{B}})$ is the ball $\mathcal{B}(p, \zeta_{\mathcal{B}})$.

Let us then consider the sequence of closed balls centered at the node p:

$\mathcal{B}_0 = \{p\}$, with a radius equal to -∞

$\mathcal{B}_1 = \mathcal{B}(p, \zeta_{\mathcal{B}_0})$

...

$\mathcal{B}_k = \mathcal{B}(p, \zeta_{\mathcal{B}_{k-1}})$

...

$\mathcal{B}_N = \mathcal{B}(p, \zeta_{\mathcal{B}_\infty})$

These balls are strictly increasing, so that: $\forall (i,j) \in \{0, ...N\}^2, i < j, \mathcal{B}_i \subsetneqq \mathcal{B}_j$.

Theorem 2.17. *The closed balls of an ultrametric distance constitute a stratified dendrogram. If $\mathcal{B}$ is a closed ball, its stratification index is equal to its diameter.*

Proof. Let us consider a closed ball $\mathcal{B}(p,r)$. We want to show that Pred(A) is completely ordered for $\subset$. So let us consider two predecessors of A, a ball $B = \mathcal{B}(q, r_B)$ and a ball $C = \mathcal{B}(p, r_C)$. As the node p belongs to both balls B and C, it is also the center of these balls. Thus, B and C are two balls with the same center p, and we have: $\mathcal{B}(p, r_B \wedge r_C) \subset \mathcal{B}(p, r_B \vee r_C)$. □

The union of all closed balls of an ultrametric distance forms a hierarchy, that we refer to as the *ultrametric hierarchy*.

Theorem 2.18 (Ultrametric hierarchy)**.** *Let S be a finite set and let $d : S \times S \to \mathbb{R}_{\geq 0}$ be an ultrametric distance. The closed balls $\mathcal{B}(.,\zeta)$ constitute a hierarchy of the space S, since:*

— *these balls are either identical or disjoint,*

— *each point is the center of one of these balls, meaning that the whole domain covered.*

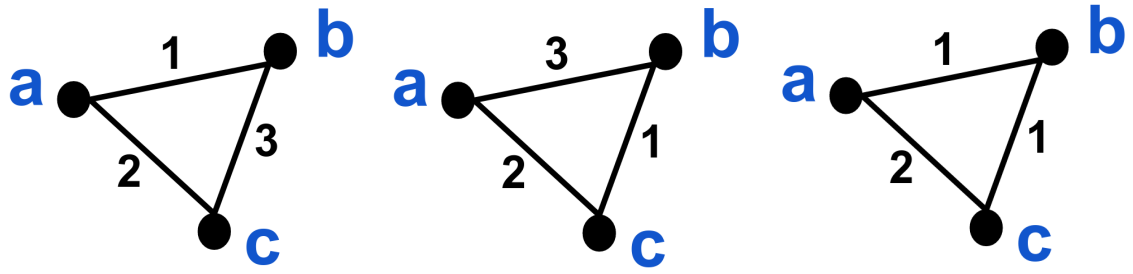
Inversely, let us consider a list of nested partitions $\{S = \mathsf{P}_0, \ldots, \mathsf{P}_n\}$*, such that for any* i, j*, such that* $j < i$*, each region of* P_i *is the union of regions of* P_j*. One can then define an ultrametric distance* d *between regions of* $S = \mathsf{P}_0$*:* $\forall (p,q) \in S$, $\begin{cases} d(p,q) = 0 \text{ if } p = q \\ d(p,q) = \bigwedge\{i | \exists \mathsf{R} \in \mathsf{P}_i, p \subset \mathsf{R}, q \subset \mathsf{R}\} \end{cases}$.

Corrolary 2.19. Stated differently, when operating hierarchical clustering, each ultrametric is associated to a unique stratified dendrogram, i.e. a sequence of nested partitions of the space represented by a dendrogram structure associated with a stratification index.

Theorem 2.20. *The supremum of two ultrametrics is an ultrametric.*

Proof. Let us consider two ultrametrics d_1 and d_2 defined on a finite set S, and let us consider the function $d_1 \vee d_2 : (x,y) \in S^2 \mapsto d_1(x,y) \vee d_2(x,y)$. To prove that $d_1 \vee d_2$ is an ultrametric, we must prove that it checks all properties of theorem 2.10. Properties 1 and 2 are trivial, so let us prove the property 3, namely the ultrametric inequality. Let us consider two points $(x,y,z) \in S^3$. Then:
$(d_1 \vee d_2)(x,y) \vee (d_1 \vee d_2)(y,z) = (d_1(x,y) \vee d_1(y,z)) \vee (d_2(x,y) \vee d_2(y,z)) > (d_1(x,z) \vee d_2(x,z)) = d_1 \vee d_2(x,z)$. Thus $d_1 \vee d_2$ is an ultrametric. □

Remark 2.21. The infimum of two ultrametrics is generally not an ultrametric, as illustrated on a simple example in figure 2.2. However, one may associate to this infimum the largest ultrametric distance below it, namely its subdominant ultrametric, as we shall see hereafter (cf definition 2.22).



(a) d_1 (b) d_2 (c) $d_1 \wedge d_2$

Figure 2.2 – Simple example illustrating that the infimum of two ultrametric is not necessarily an ultrametric, since it does not here respect the ultrametric inequality. Indeed: $2 = (d_1 \wedge d_2)(a,c) > (d_1 \wedge d_2)(a,b) \vee (d_1 \wedge d_2)(b,c) = 1$.

Definition 2.22 (Subdominant ultrametric)**.** Let S be a set, and let us denote by $\mathbf{D}_S$ the set of all possible dissimilarities on S, and by $\mathbf{U}_S$ the set of ultrametrics on S. Since each ultrametric is a dissimilarity, $\mathbf{U}_S$ is a subset of $\mathbf{D}_S$.

Let us then consider a dissimilarity $\mathbf{d}$ on S and let us denote $\mathbf{U}_\mathbf{d}$ the set of ultrametrics:

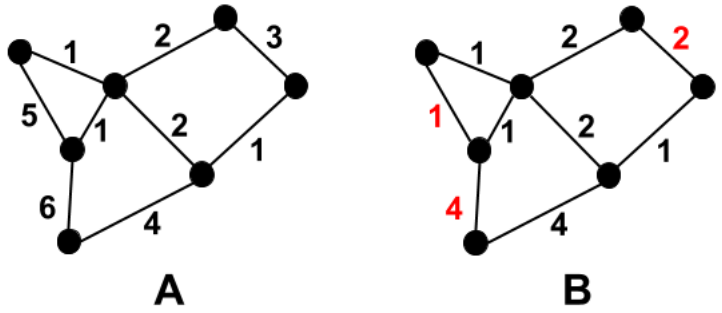
$$\mathbf{U}_\mathbf{d} = \{e \in \mathbf{U}_S | e \leq \mathbf{d}\}.$$

The set $\mathbf{U_d}$ has a largest element in the poset $(\mathbf{D}_S, \leq)$, which we call the *subdominant ultrametric* for the dissimilarity $\mathbf{d}$. We denote $\overline{\mathbf{d}}$ the subdominant ultrametric associated with a dissimilarity $\mathbf{d}$.

Proof. Let consider the zero dissimilarity $\mathbf{d}_0$ given by: $\forall (x, y) \in S^2, \mathbf{d}_0(x, y) = 0$. $\mathbf{d}_0$ is an ultrametric and $\mathbf{d}_0 \leq \mathbf{d}$, so $\mathbf{U_d}$ is a nonempty set.
Furthermore, the family $\mathbf{U_d}$ is closed by supremum according to theorem 2.20, hence it has a largest element. □

Remark 2.23 (Saliency)**.** In the image processing literature, it is often made reference to the notion of *saliency* [Najman et al. (1996); Arbelaez (2006)], to value the importance of a connexion between points of a studied space that has been hierarchically structured. Saliency then corresponds to the subdominant ultrametric defined, for a given dissimilarity, over any two pairs of points of this space.

Taking into account the saliency thus constructed instead of the initial metric can lead to a different understanding of the connexion between given nodes, as illustrated in figure 2.3.



A: graph with dissimilarity values. B: the same graph with the corresponding ultrametric values.

Figure 2.3 – Illustration of the impact of the choice to replace dissimilarity values (graph A) with saliency values (graph B). Note that an edge with a high dissimilarity value can have a low saliency value.

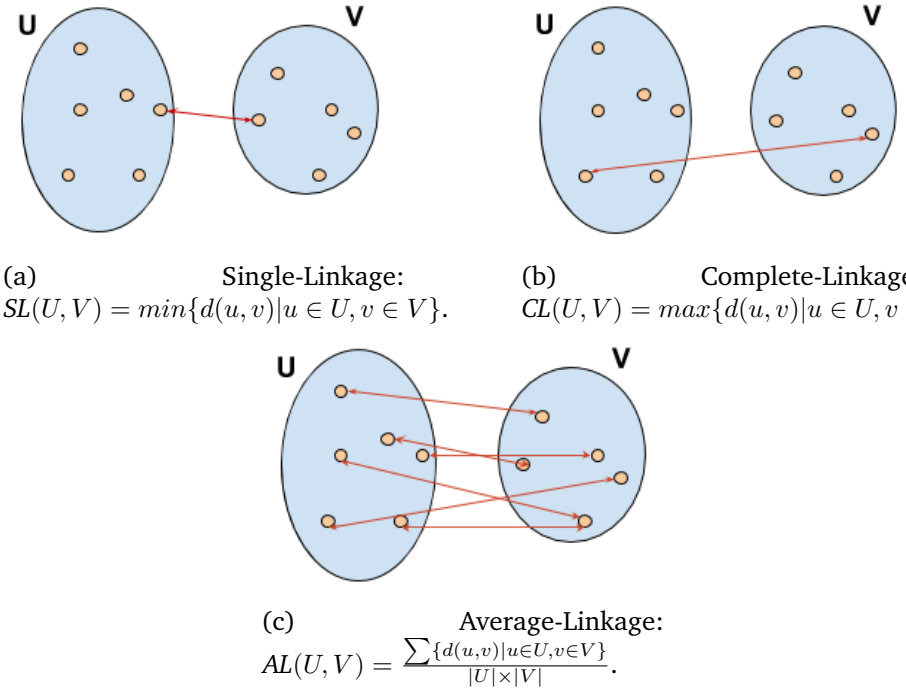
2.2.5 Choice of linkage in hierarchical clustering

Definitions

In an agglomerative hierarchical clustering approach, we seek to merge first the most similar clusters, which implies to define a dissimilarity between clusters. There are different ways of choosing such a dissimilarity between two clusters. Let (S, d) be a metric space of objects. Let us consider two clusters U and V at a given level of the hierarchy. The new dissimilarity between these two clusters can notably be defined using one of the following real-valued functions with two arguments defined on the set of subsets of S:

— $SL(U, V) = min\{d(u, v) | u \in U, v \in V\}$,

— $CL(U,V) = max\{d(u,v)|u \in U, v \in V\}$,
— $AL(U,V) = \frac{\sum\{d(u,v)|u \in U, v \in V\}}{|U| \times |V|}$,



(a) Single-Linkage: $SL(U,V) = min\{d(u,v)|u \in U, v \in V\}$.

(b) Complete-Linkage: $CL(U,V) = max\{d(u,v)|u \in U, v \in V\}$.

(c) Average-Linkage: $AL(U,V) = \frac{\sum\{d(u,v)|u \in U, v \in V\}}{|U| \times |V|}$.

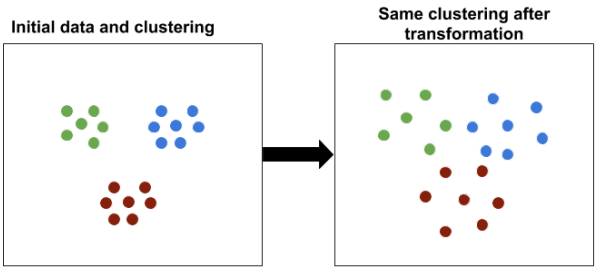
Figure 2.4 – Illustration of the different possible linkages for hierarchical clustering.

The names of the functions *SL*, *CL*, *AL* are acronyms for "single linkage", "complete linkage" and "average linkage" respectively. These are the most classical variants of the hierarchical agglomerative clustering we discussed. Complete-linkage and especially average-linkage are often preferred to single-linkage in many practical applications, as these two methods have the property to somehow favor the association of compact subsets of points. But as we will see hereafter, this can also be the case of single-linkage when provided with a more informative dissimilarity in input. Furthermore, single-linkage has interesting properties that are not shared by complete and average-linkage, and we expose them hereafter.

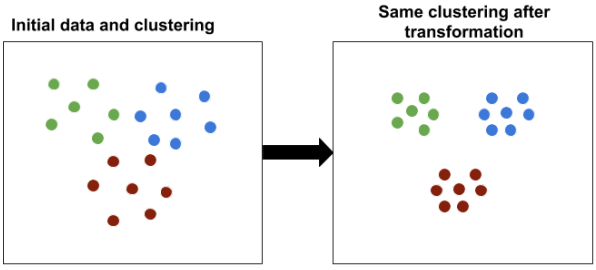
Kleinberg's impossibility theorem and discussions

In [Kleinberg (2003)], the author proposed that the ideal clustering function F^* should achieve three properties:

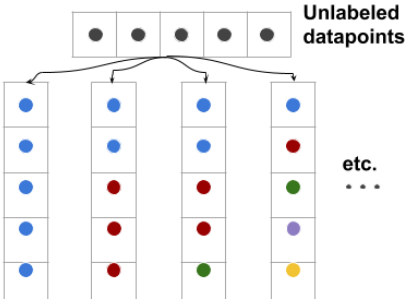
1. *Scale-invariance*: the method provides the same classification when all datapoints are scaled equally in all directions of the space. For any scalar α, F^* produces the same result when the distances d between all datapoints are multiplied, i.e. $F^*(d) = F^*(\alpha.d)$.



2. *Consistency*: if we stretch the data so that the distances between clusters increases and/or the distances within clusters decreases, then the clustering should not change. Let us consider two distances functions d and d'. If, for every pair (i, j) belonging to the same cluster, $d(i, j) \leq d'(i, j)$, and for every pair (k, l) belonging to different clusters, $d(k, l) \geq d'(k, l)$, then we should have $F^*(d) = F^*(d')$.



3. *Richness*: Suppose a dataset contains N points, but the distances between points is unknown. An ideal clustering function F^* should be flexible enough to produce all possible partitions/clusterings of the set it is applied to. It means that the range of this function is equal to all possible partitions of this set of length N.



Additionally, [Kleinberg (2003)] proves that no clustering method can satisfy at the same time these three desirable properties.

However, in [Zadeh et al. (2009)], the authors propose to relax the richness property to a K-richness property, arguing that in many cases, the (approximate) number K of desired clusters is known. They also introduce an order-consistency property.

4. *K-richness*: richness with known number K of desired clusters.

5. *Order-consistency*: for any two distances functions d and d', a number of clusters K, if the order of edges in d is the same as the order of edges in d', then $F(d, K) = F(d', K)$.

They show that single-linkage clustering is the only one to be consistent, K-rich, scale-invariant, and order-consistent at the same time.

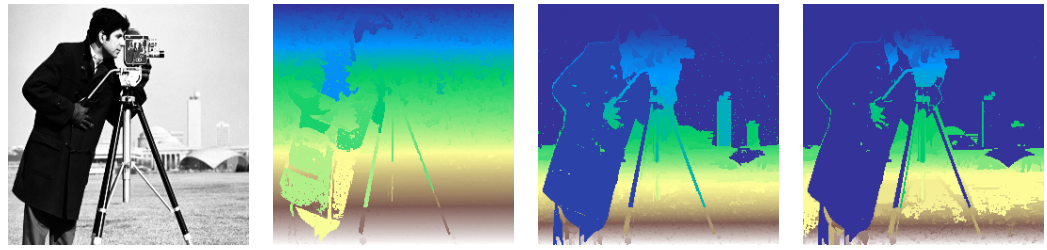
Stability

Furthermore, in [Carlsson et al. (2010)] the authors study hierarchical clustering schemes and compare their stability by quantifying the degree to which perturbations in the input metric space affect their results. To do that, they work with the Gromov-Hausdorff distance, defined over the set of hierarchies. We shall briefly introduce this distance in section 2.4.3. Such a distance indeed allows to compare the results of different hierarchical clustering methods in a precise manner. Their conclusion is that single-linkage enjoys the nice theoretical properties of being stable and continuous in the sense of the Gromov-Hausdorff distance, whereas average- or complete-linkage lack these properties. We can thus be sure that for any small pertubations of the set of objects in input, the outputs of single-linkage clustering for the original and the perturbated set of objects are at a small distance from one another. Thus, single-linkage hierarchical clustering appears to be more robust. This property does not hold true for average-linkage and complete-linkage linkage hierarchical clustering. Readers are invited to refer to [Carlsson et al. (2010)] for more details and proofs of the mentioned results.

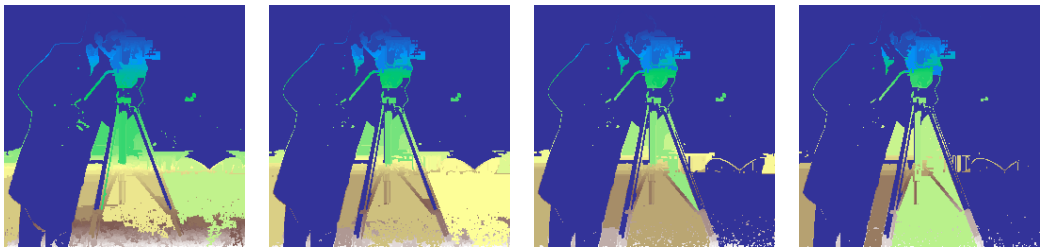
The problem of the chaining effect

Single-linkage hierarchical clustering is often perceived to produce clusters that are less coherent conceptually than other linkages, because it can suffer from the *chaining effect* which makes it more likely to produce clusterings which separate items should be together. Indeed, since the merging criterion is strictly local in the case of single-linkage, a chain of points can be extended for long distances without any regard for the overall shape of the emerging cluster. This way, two clusters globally different can be connected through a chain of two-by-two similar points: there may appear a long chain of points $(a, b, ..., z)$ such that every two consecutive elements are similar but that the distant elements no longer resemble each other at all. This chaining effect is illustrated in figure 2.5, where the graph we consider has been constructed upon an image, with each node corresponding to a pixel and adjacent nodes valuated by the gradient between them. In this example,we can see that when considering successive levels of the hierarchy, regions that are very different gradually merge.

But as it is noticed by the authors in [Carlsson et al. (2010)], this observation only accounts for the need to have a dissimilarity that takes into account some notion of density rather than simply a local geometric information alone. Indeed, as we shall see in the next chapter, we can construct a dissimilarity that takes into account what is happening directly around a node as well as over a larger integration domain. This way, we can multiply the points of view in a guided



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



(e) (f) (g) (h)

Figure 2.5 – Example illustrating the chaining effect on an image graph. Figures (b) to (h) show successive levels of the SL-hierarchy: regions that are very different but locally similar gradually merge.

way and avoid for this chaining effect to happen. The same kind of approach is for example adopted in [Soille (2011)].

Single linkage leads to a simple ultrametric computation

Another interesting aspect when using single-linkage clustering algorithm on a graph is that for a given dissimilarity defined over the graph, its subdominant ultrametric can easily be deduced. Indeed, the subdominant ultrametric value between two nodes can then be computed as the critical threshold value for which the two nodes are no longer in the same tree when cutting edges by decreasing valuations. This presents the benefit of being easy and extremely fast to compute.

For all those reasons, we chose to use single-linkage hierarchical clustering scheme. Unless otherwise specified, all hierarchical clusterings presented from now on will use this linkage.

2.3 Graph-based Hierarchical Clustering and Ultrametrics

2.3.1 Hierarchical Minimum Spanning Forests

As we saw in the previous section, the closed balls of an ultrametric form constitute a stratified dendrogram and thus a hierarchy, that we refer to as to the *ultrametric hierarchy*. There are many different types of ultrametrics. In this thesis, we will mainly work with ultrametrics created on undirected graphs.

Indeed, given such a graph with positive edge weights, an ultrametric distance between two nodes is given by the weight of the minimax path between these nodes, i.e. the path for which the maximum edge weight along it is minimal (also known as the lowest path distance between these nodes). This ultrametric is nothing but the subdominant ultrametric defined in definition 2.22 when using single-linkage hierarchical clustering (see section 2.2.5).

To easily compute this ultrametric as mentioned in section 2.2.5, we make use of minimum spanning trees which were introduced in section 1.3.3. As we saw in theorem 1.28, minimum spanning trees verify a property that allows a fast computation of this minimax path between any two nodes of a graph. This way, they easily lead to the definition of an ultrametric, and thus to an ultrametric hierarchy.

We can generate a MST of the graph using for example the Boruvka algorithm presented and illustrated in section 7. Once provided with it, we can derive an ultrametric from this MST: for any pair of nodes (p, q) of the graph, the ultrametric distance $\boldsymbol{\lambda}(p, q)$ is equal to the weight of the highest edge on the unique path linking them in the MST. Thus, for a hierarchy $(\mathcal{H}, \boldsymbol{\lambda})$ associated with a minimum spanning tree $\mathcal{MST}$, we have:

$$\forall (p, q), \boldsymbol{\lambda}(p, q) = \bigvee \{\eta_{st}, \mathsf{e}_{st} \in \pi_{pq} \subset \mathcal{MST}\} \tag{2.3}$$

The simplicity of the procedure for the obtaining of the MST and of the underlying ultrametric is illustrated in figure 2.6. An illustration of a hierarchy associated with a MST can also be found in figure 2.7.

Note that if we give new edges valuations to the MST, we can see it as any other spanning tree, and we thus can infer from it an ultrametric using equation (2.3). We will make a heavy use of this property in the following.

2.3.2 Equivalent global representations

The hierarchies we work with are ultrametric hierarchies obtained using single-linkage hierarchical clustering on an edge-weighted graph whose ultrametric distance is encoded in a minimum spanning tree. They can thus be represented in at least three different ways.

Representation as a minimum spanning forest

First, hierarchies can be represented as minimum spanning forests, as we can see it on figure 2.7.

Let us consider two nodes x and y of the graph $\mathcal{G}$. If $d(x, y) = \lambda$, there exists a minimax path linking x and y. If we cut all edges with a valuation superior to λ, there is no path anymore connecting them, and they are thus members of disjoint subgraphs. We obtain the same result by applying the same procedure to a minimum spanning tree of the graph: cutting all edges of the $\mathcal{MST}$ with a valuation superior to λ leads to a subset of trees $(\mathcal{T}_i)$. Each subgraph $\mathcal{G}i$ is spanned by a tree $\mathcal{T}_i$, so that both families $(\mathcal{G}_i)$ and $(\mathcal{T}_i)$ induce the same partition of the nodes, corresponding to the open ball of radius λ for the ultrametric distance $\boldsymbol{\lambda}$.

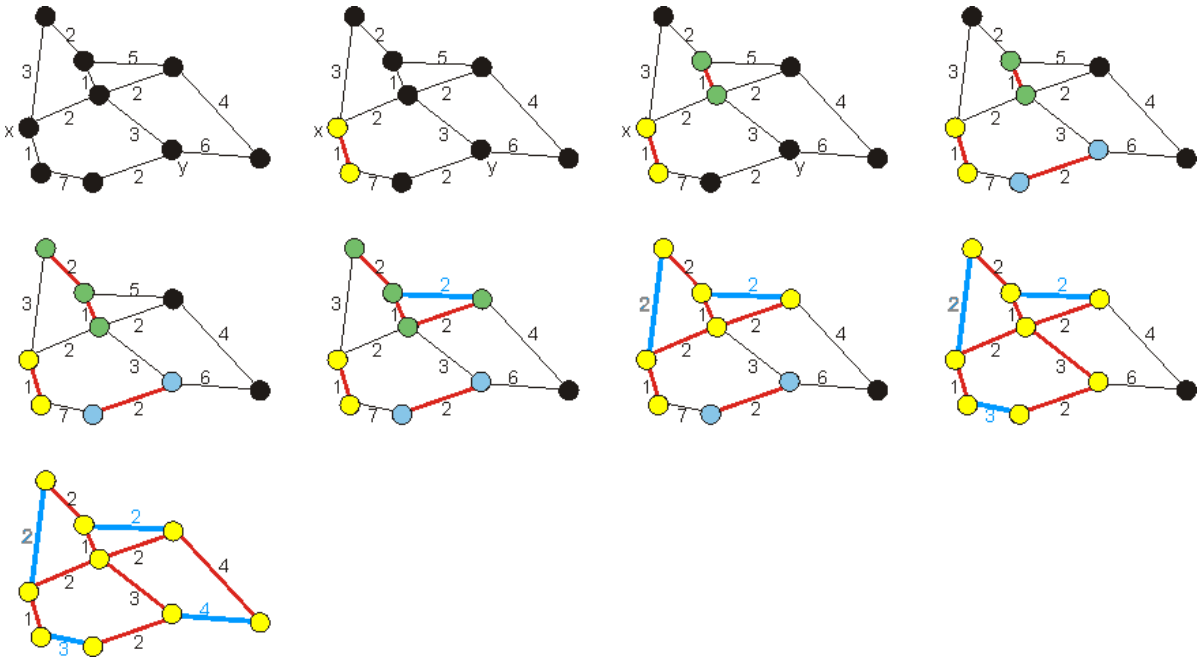


Figure 2.6 – Left to right, top to bottom. We make use of the Kruskal algorithm to generate a minimum spanning tree of the graph. Each time we add an edge (in red) with a weight equal to λ connecting two connected components of the graph, we add at the same time all the other edges (in blue) with a weight superior or equal to λ that connect these two components, and we give them the same weight λ. This way, the new edges weights are equal to the ultrametric distance between weights extremities. It is also the valuation of the highest edge on the unique path connecting the two extremities in the minimum spanning tree. It is sometimes referred to as the "saliency" (cf. definition 2.23).

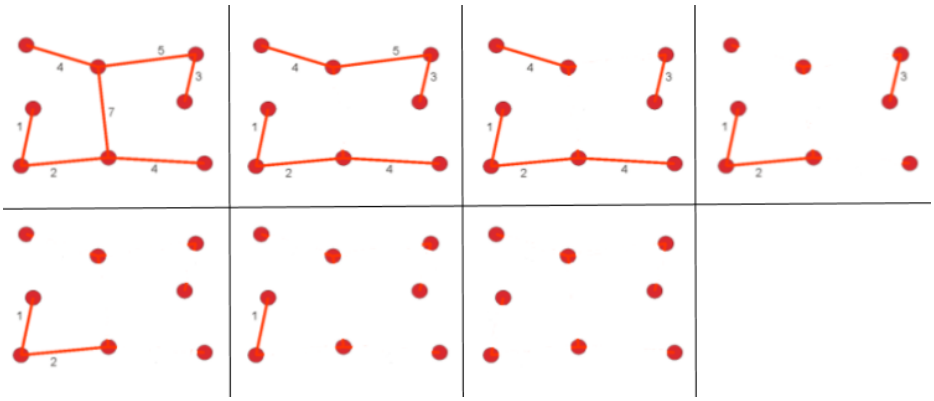


Figure 2.7 – Left to right, top to bottom. The successive cuts of the highest edges of the minimum spanning tree of the graph correspond to the successive levels of a hierarchical clustering.

Representation using the tree structure of a dendrogram

As explained in the previous sections, there is a one-to-one correspondance between a ultrametric hierarchy and a stratified dendrogram (cf corollary 2.19). Furthermore, a dendrogram structure may be represented as a tree. The nodes of this tree then represent the closed balls $\mathcal{B}(q, \lambda)$ associated with $\mathcal{G}$, as it was stated in theorem 2.17. The leaves of the tree are the balls with a radius equal to $-\infty$ (the individual nodes of $\mathcal{G}$). Each node is linked with its unique predecessor by an edge, except the largest ball containing all nodes, which constitutes the root of the tree. The stratification index correspond to the ultrametric distance.

An example of a dendrogram can be seen in figure 2.8. Note that even if it is the case in our illustration, dendrograms are not necessarily binary trees, i.e. a node can have more than two children in a dendrogram.

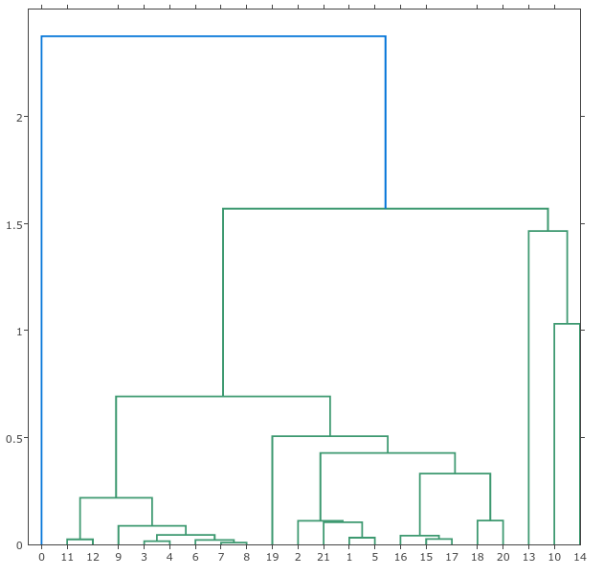


Figure 2.8 – Example of a dendrogram obtained on a set of points, generated using the data science python library plotly [Inc. (2015)].

However, even when dendrograms are not binary trees, we can represent them as binary trees. For example, if a dendrogram has a node having three children for a given stratification index value, we can consider it to be a succession of two binary relations by choosing arbitrarily their order. It will not matter when using dendrograms for our purpose, since the stratification index value is usually the only important information here, and it remains the same in both cases. But this reveals useful in terms of implementation, as we then have a structured and simple dendrogram model to work with. This correspondence is illustrated on an example in figure 2.9.

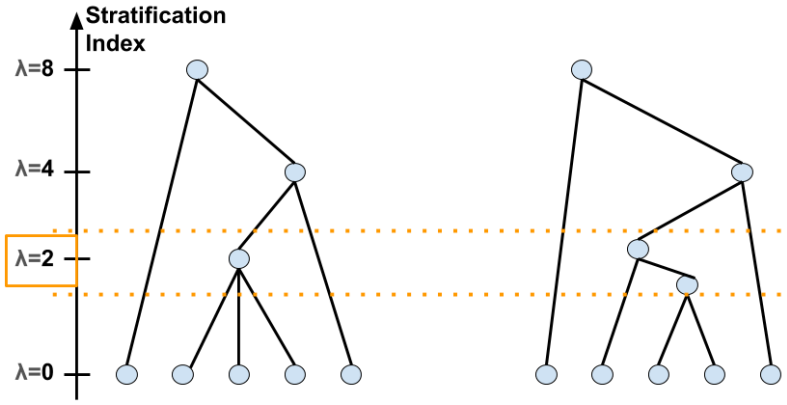


Figure 2.9 – Illustration of the correspondence between a non-binary dendrogram and a binary dendrogram. In this example, for a stratification index of 2, the non-binary kinship relation between nodes can be seen as a succession of binary ones, by choosing arbitrarily their orders but keeping the same stratification index for both of them.

In the specific case where we make use of a minimum spanning tree to induce a hierarchical clustering of the graph, we can visualize easily the correspondences between the minimum spanning tree and the associated dendrogram tree structure. These correspondences are presented in figure 2.10.

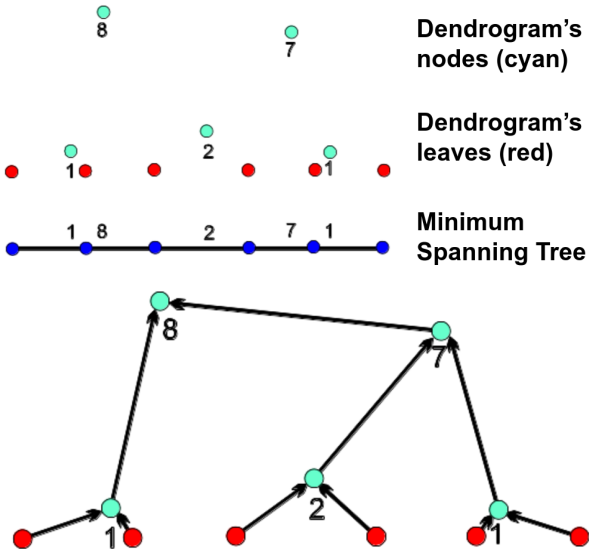


Figure 2.10 – Structure of a dendrogram corresponding to a MST-based hierarchical clustering. The MST nodes become the leaves of the dendrogram (in red), and the MST edges become the internal nodes of the dendrogram (in cyan).

The needed information for completely specifying a binary dendrogram is then limited. For implementing our algorithms we make use of the following structure:

1. Each node i (leave or internal node) has:
   — a father $Father(i)$,

— several forefathers $Forefather(i)$,

— one ancestor $Ancestor(i)$ the first of the lineage,

— one or several valuations linked to problem-related features.

2. Each internal node k represents an edge e_{ij} of the minimum spanning tree:
   — it has the valuation of this edge: $\mu_k = \omega_{ij}$
   — $neighbor(k) = i$
   — $neighbor(k) = j$
3. Each internal node k also has a son l and a daughter g:
   — $son(k) = l$
   — $daughter(k) = g$

An example of these is given in figure 2.11 for the case illustrated in figure 2.10.

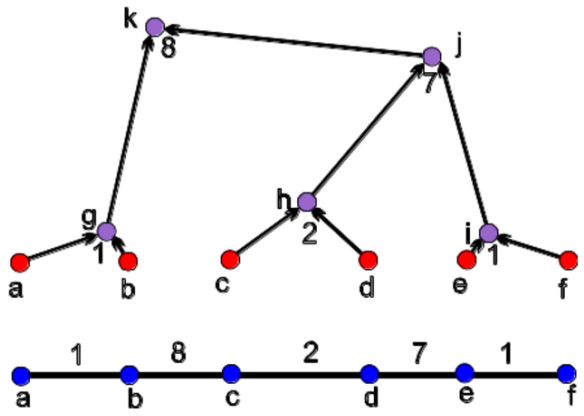


Figure 2.11 – Example of the internal structure of a dendrogram, for the node j:

1. The node j corresponds to the edge e_{de} of the minimum spanning tree:
   — it takes as valuation this edge valuation: $\mu_j = \omega_{de} = 7$,
   — $neighbor1(j) = d$,
   — $neighbor2(j) = e$.
2. The internal node j has a son h and a daughter i:
   — $son(j) = h$,
   — $daughter(j) = i$.

Remark 2.24 (Dendrogram and ultrametric)**.** In a stratified dendrogram, the ultrametric distance between two leaves p, q (that correspond to nodes of the graph) $\boldsymbol{\lambda}(p, q)$ is the level of stratification of the root of the smallest subdendrogram containing both p and q.

Representation as a matrix

Once a minimum spanning tree of the graph has been constructed, one can easily compute the minimax paths and thus compute the lowest path distance between any two adjacent nodes of the graph, which is nothing but the subdominant ultrametric between two nodes.

As we saw before, one can indeed easily compute this subdominant ultrametric starting from a minimum spanning tree of the graph. By cutting edges of the minimum spanning tree by decreasing values, we can identify the cut valuation for which two nodes of the graph become separated in the output clustering: this valuation correspond to the subdominant ultrametric distance between them (otherwise called saliency of this pair of nodes).

So one can obtain a matrix entirely representing the connexions between nodes and thus the hierarchical structure. Such a matrix M is square, symmetric, and of size $(n-1) \times (n-1)$, with n the number of nodes in the graph and thus $(n-1)$ the number of edges in the minimum spanning tree. If we denote by $\mathbf{U}$ the subdominant ultrametric easily computed between any pair of nodes, then $M(i,j) = \mathbf{U}(\mathsf{v}_i, \mathsf{v}_j)$ for $(i,j) \in \{1, ..., n-1\}^2$. An example is provided in figure 2.12, where we can see the correspondance between the dendrogram illustrated in figure 2.8 and the aforementioned matrix.

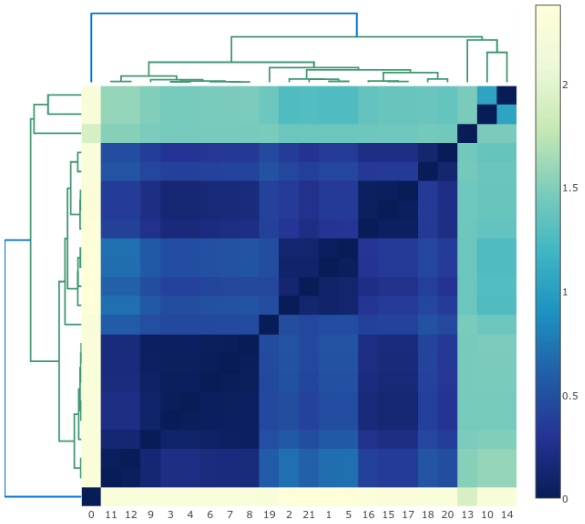


Figure 2.12 – A symmetric matrix containing values of the subdominant ultrametric for a given hierarchical clustering and the corresponding dendrogram.

The subdominant ultrametric distance can be derived from the adjacency matrix of the graph (see definition 1.7) in the $(\min, \max)$-algebra [Gondran (1975)].

Definition 2.25 (Matrix product in $(\min, \max)$-algebra)**.** Let A and B be two matrices. Their matrix product in the $(\min, \max)$-algebra $C = A \times B$ is defined by:

$$c_{ij} = \bigwedge_k a_{ik} \vee b_{kj} \tag{2.4}$$

If a_{ik} is the weight of the lowest path of length inferior to n between i and k, and b_{kj} is the weight lowest path of length inferior to n between k and j, then $a_{ik} \vee b_{kj}$ is the weight of the lowest path of length inferior to $2n$ between i and j. Thus $\bigwedge_k a_{ik} \vee b_{kj}$ corresponds to the weight of the lowest path of length inferior to $2n$ between i and j.

The matrix product applied to an adjacency matrix in this algebra allows for the computation of the shortest paths.

Property 2.26 (Shortest paths in the $(\min, \max)$-algebra)**.** Let A be the adjacency matrix of a graph $\mathcal{G}$. Then, for any i, j, a_{ij} represents the shortest path of length inferior or equal to 1 between i and j. More generally, $(a_{ij})^k$ represents the shortest path of length inferior or equal to k between i and j

Property 2.27 (Subdominant ultrametric and adjacency matrix in the $(\min, \max)$-algebra)**.** Let A be the adjacency matrix of a graph $\mathcal{G}$. Then:

$$\exists n, A^n \times A^n = A^n = A^* \tag{2.5}$$

A^* is then the adjacency matrix of the subdominant ultrametric distance associated with A.

The computation of the subdominant ultrametric on a graph, by successive multiplications of its adjacency matrix, is illustrated in figure 2.13.

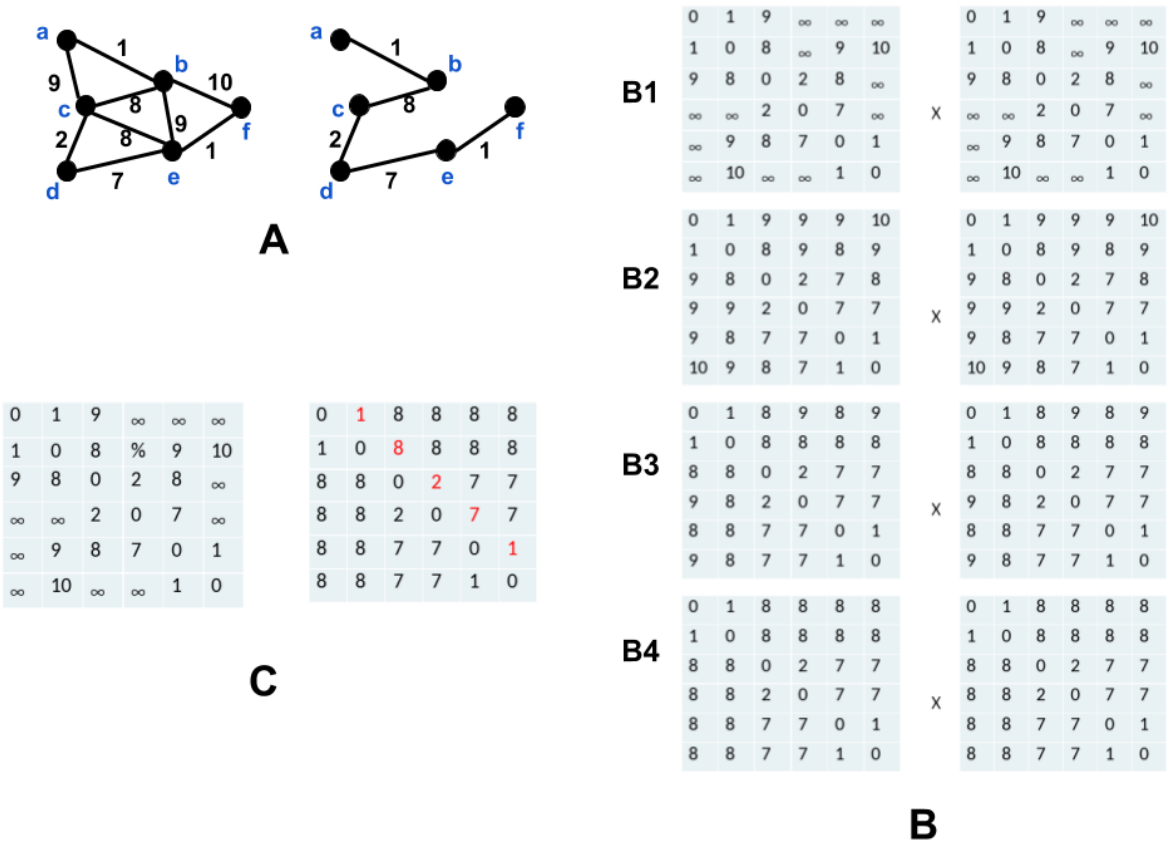


Figure 2.13 – **A**: an undirected positive-weighted graph and its associated minimum spanning tree. **B**: **B1** to **B4** represent the successive multiplications of the adjacency matrix by itself in the $(\min, \max)$-algebra until stability. **C**: the result of these operations, on the right, is the adjacency matrix of the subdominant ultrametrics. Note that the values that remain the same in the initial adjacency matrix and its subdominant ultrametric counterpart (in red) correspond to the edges of the minimum spanning tree.

2.3.3 Zahn's Clustering Algorithm

The usefulness of minimum spanning trees for hierarchical clustering was explored by C.T. Zahn [Zahn (1971)]. Indeed, we saw that suppressing one edge on a MST of the graph leads to its subdivision into two forests, each spanning a given portion of the metric space. So that by suppressing the more *inconsistent edges* first, one creates a hierarchical clustering in which similar objects remain longer in the same clusters. The notion of inconsistent edge is thus essential for Zahn's algorithm, and can be defined in multiple ways.

Algorithm 2.1: Zahn's clustering algorithm

Data: A complete graph $\mathcal{G} = (\mathsf{V}, \eta)$ defined over a metric space (S, d) such that $|\mathsf{V}| = |S|$, and with for $\forall (i,j) \in \{1,...,n\}^2, \eta_{i,j} = d(\mathsf{v}_i, \mathsf{v}_j)$;

Result: a clustering of the objects of S.

1 Construct a minimum spanning tree of $\mathcal{G}$;

2 identify inconsistent edges in the minimum spanning tree;

3 create a cluster hierarchy by successively removing inconsistent edges.

Usually, the strategy that consists in departing from the original minimum spanning tree and cutting edges by decreasing valuations suffers from the chaining effect described in section 2.2.5. So there is a need to quantify the consistency of the edges according to given criteria. In order to avoid suffering from the chaining effect and to highlight interesting clusters, one may want to propose hierarchical clustering methods in which the more inconsistent edges for these criteria have the higher values, so that cutting edges by decreasing values in the $\mathcal{MST}$ leads to a suitable hierarchy for these criteria. More specifically, the inconsistency of an edge may be determined or measured by taking into account a larger neighborhood of the considered edge. In the following of this thesis, we will propose methods to reevaluate the edges of the MST and thus generate a new hierarchy from a previous one.

Remark 2.28. Note that howsoever is defined the inconsistency, this algorithm is dependant on the minimum spanning tree, so that in cases where several minimum spanning trees exist, the result can be very sensitive to the choice of one or the other. Two solutions can be thought of to circumvent this issue for robustness purpose:

1. One can choose one MST based on supplementary criterias.
2. One can consider all possible MST during computations.

2.4 Lattice of Hierarchies

2.4.1 An order relation to form a complete lattice

Let us consider two utrametric hierarchies $\mathcal{H}_1$ and $\mathcal{H}_2$ and their associated ultrametrics $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$, defined over a set S. We can then define an order between two hierarchies by:

$$\mathcal{H}_1 < \mathcal{H}_2 \Leftrightarrow (\forall (\mathcal{T}_a, \mathcal{T}_b) \in \mathsf{P}(S)^2, \boldsymbol{\lambda}_1(\mathcal{T}_a, \mathcal{T}_b) > \boldsymbol{\lambda}_2(\mathcal{T}_a, \mathcal{T}_b)) \tag{2.6}$$

$\mathcal{H}_1 < \mathcal{H}_2$ can be read "$\mathcal{H}_1$ is finer than $\mathcal{H}_2$", and means that $\mathcal{H}_1$ has more regions than $\mathcal{H}_2$ at each level. Indeed, let us consider a node p and a radius ρ. Then $q \in \mathcal{B}_1(p, \rho)$ means that $\boldsymbol{\lambda}_1(p, q) \leq \rho$. Equation (2.6) implies that $\boldsymbol{\lambda}_2(p, q) \leq \rho$, which means that $q \in \mathcal{B}_2(p, \rho)$. Thus if we denote $\mathsf{P}_1(\rho)$ and $\mathsf{P}_2(\rho)$ the partitions associated with the stratification level ρ for $\mathcal{H}_1$ and $\mathcal{H}_2$, each region of $\mathsf{P}_1(\rho)$ is included in a region of $\mathsf{P}_2(\rho)$, i.e. $\mathsf{P}_1(\rho)$ is finer than $\mathsf{P}_2(\rho)$.

With such an order relation, the hierarchies of $\mathcal{P}(S)$ form a complete lattice. The maximum element of the hierarchy has a unique element S, and the smallest contains only singletons.

Different adjunctions relations are also definable on the lattice of hierarchies [Ronse (2010); Meyer (2013a)], which opens the way to possible definitions of basic morphological operators and thus to morphological filtering on hierarchies.

2.4.2 Combining hierarchies

It might be interesting to combine several hierarchies, for example to use information from several sources (in color or multi-spectral images for example). We can notably do so by taking the infimum of hierarchies, as we will show hereafter.

Definition 2.29 (Infimum of hierarchies)**.** The infimum of two hierarchies $(\mathcal{H}_1, \boldsymbol{\lambda}_1)$ and $(\mathcal{H}_2, \boldsymbol{\lambda}_2)$ is written $\mathcal{H}_1 \wedge \mathcal{H}_2$ (or $\mathrm{INF}(\mathcal{H}_1, \mathcal{H}_2)$) and is defined by its ultrametric $\boldsymbol{\lambda}$ being the supremum of the ultrametrics of both hierarchies $\boldsymbol{\lambda} = \boldsymbol{\lambda}_1 \vee \boldsymbol{\lambda}_2$ (cf theorem 2.20).

Definition 2.30 (Supremum of hierarchies)**.** The supremum of two hierarchies $(\mathcal{H}_1, \boldsymbol{\lambda}_1)$ and $(\mathcal{H}_2, \boldsymbol{\lambda}_2)$ is written $\mathcal{H}_1 \vee \mathcal{H}_2$ (or $\mathrm{SUP}(\mathcal{H}_1, \mathcal{H}_2)$), is the smallest hierarchy larger than $\mathcal{H}_1$ and $\mathcal{H}_2$. As the infimum of their ultrametrics is not an ultrametric, the largest ultrametric below this infimum is their subdominant ultrametric (cf remark 2.20), which we can choose to be the ultrametric associated with this supremum of hierarchies, i.e. $\boldsymbol{\lambda} = \overline{\boldsymbol{\lambda}_1 \wedge \boldsymbol{\lambda}_2} \neq \boldsymbol{\lambda}_1 \wedge \boldsymbol{\lambda}_2$.

Note that these definitions can easily be extended to a family of hierarchies with more than two elements.

2.4.3 Gromov-Hausdorff distance between hierarchies

Definition

A hierarchical clustering of a set of graphs nodes is completely defined by the ultrametric distance between these nodes. It constitutes a metric space defined on these nodes. Furthermore, for the same set of nodes, several hierarchies can be derived with different ultrametrics. One may then wonder if there are ways to quantify the differences or redundancies between hierarchical clusterings, through the definition of a distance between dendrograms.

In [Gromov et al. (1981)] (translated in English in [Gromov (2007)]), the authors propose a distance, called Gromov-Hausdorff (GH) distance, to measure how far two compact metric spaces are from being isometric. It gives a very useful and natural way to distinguish between metric spaces. By reducing this distance to the subclass of ultrametric spaces, we can in particular

quantify the relative contributions of different hierarchical clusterings. This distance, used intensively in several fields such as phylogenetics and data mining [Felsenstein (2014)], has also notably been used in image processing as a way to estimate the similarity between two points clouds [Mémoli (2004)].

We hereby present the general idea of the Gromov-Hausdorff distance, which is a classical distance between metric space. In our case, we will use it between ultrametric spaces, which are specific metric spaces (since they consist of metric space with a metric satisfying a stronger condition than the triangle inequality, per se the ultrametric inequality).

Let us consider two metric spaces (X_1, u_α) and (X_2, u_β). One supposes that we have defined two functions $f : X_1 \to X_2$ and $g : X_2 \to X_1$ that are maps from one space to the other. The GH-distance is expressed as:

$$\mathrm{d_{GH}}(X_1, X_2) := \frac{1}{2} \min_{f,g} \max(dis(f), dis(g), dis(f, g)) \tag{2.7}$$

With the distorsion $dis(f)$ and the joint distorsion $dis(f, g)$ defined as:

$$\begin{cases} dis(f) := \max_{(x,x')\in X_1^2} |u_\alpha(x, x') - u_\beta(f(x), f(x'))| \\ dis(f, g) := \max_{x\in X_1, x'\in X_2} |u_\alpha(x, g(x')) - u_\beta(x', f(x))| \end{cases} \tag{2.8}$$

Intuitively, it measures how close can we get to an isometric (distance-preserving) embedding between two metric spaces. As it is shown in [Gromov et al. (1981); Gromov (2007); Carlsson et al. (2010)], to determine (2.7) for two hierarchies defined over different sets, one must match data points before any distance computation, which is a computationally heavy operation that leads some authors to provide heuristics to approximate it in specific configurations [Mémoli (2004); Agarwal et al. (2015)].

Expression in a simpler case

However, there is a case in which the GH-distance computation is much simpler: the case where the two hierarchies have been computed upon the same set of points. Indeed, in such a case, the considered metric spaces differ only by their metrics and not by the space they cover, which means that the two distortions are symmetrical and equal to the joint-distortion as well. There is indeed no more need to map the points of the first on the second since these points are identical.
Thus, the GH distance (2.7) simply becomes:

$$\mathrm{d_{GH}}((X, u_\alpha), (X, u_\beta)) = \max_{x,x'\in X} |u_\alpha(x, x') - u_\beta(x, x')|. \tag{2.9}$$

An example of computation is provided in figure 2.14.

A natural idea that stems from this Gromov-Hausdorff distance is to see it as an extreme case (in the sense of an infinity-norm) of other possible distances.

We thus define two other distances between hierarchical spaces in this simple case:

$$\begin{cases} dis_1((X, u_\alpha), (X, u_\beta)) := \frac{1}{Card(X\times X)} \sum_{x,x'\in X} |u_\alpha(x, x') - u_\beta(x, x')| \\ dis_2((X, u_\alpha), (X, u_\beta)) := \frac{1}{Card(X\times X)} \sum_{x,x'\in X} |u_\alpha(x, x') - u_\beta(x, x')|^2 \end{cases} \tag{2.10}$$

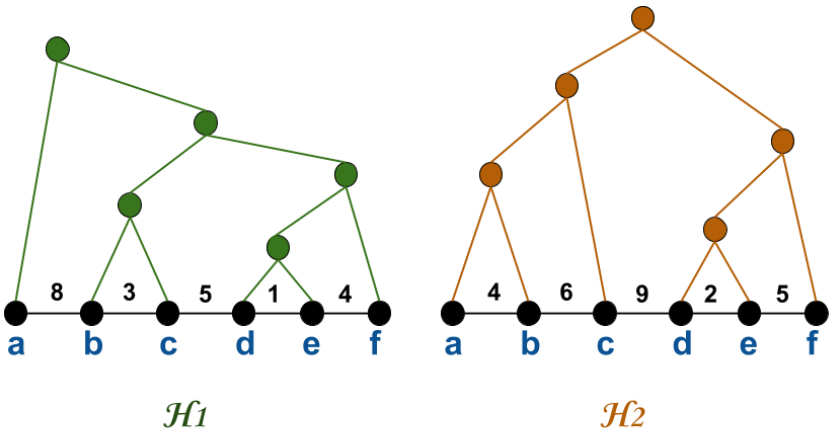


Figure 2.14 – Example of computation of the Gromov-Hausdorff distance between two hierarchies $\mathcal{H}_1$ and $\mathcal{H}_2$ generated upon the same set of points.

The Gromov-Hausdorff distance thus has a simple expression when it is computed for two hierarchies defined over the same set, i.e. corresponding to two dendrograms with the same leaves. Fortunately, this simple case will reveal itself to be interesting, as we shall see how to construct a multitude of different hierarchies based on the same departure set. We can thus, through the use of such a distance, quantify the relative contributions of each of these hierarchies.

We still have to note that this distance suffers from several limitations:

— It makes sense only if the compared ultrametrics are of the same scales and are commensurable.

— It uses the L_∞-norm and is thus very sensitive to outliers (to avoid this, one may use alternative distances defined in (2.10)).

— One can also imagine to compute such a distance only for a subset of all nodes that would have been selected otherwise.

2.4.4 The need for a normalization of ultrametric values

When wanting to compare or combine different hierarchical clusterings constructed upon the same set of objects, one can be faced with a disparity of scales in the ultrametric values associated with each one of them. Such a disparity is problematic for several reasons:

— It can give a false sense of predominancy of one hierarchy compared with another, when the ultrametric values of a hierarchy are far superior to the ones of another hierarchy.

— When values are not sufficiently well spread, and especially when they are all condensed in a small interval, it is hard to have a proper idea of what the hierarchical clustering method is actually doing.

— Distances between two hierarchies such as the Gromov-Hausdorff distance presented in the above section often make sense only if the compared ultrametric values are of the same scales and are commensurable.

To prevent such issues from happening, we propose a way to normalize these values according to the number of nodes present in each cluster compared with the total number of nodes.

Definition 2.31 (Ultrametric Normalization Relatively to the Number of Hierarchical Levels)**.** Let $(\mathcal{H}, \boldsymbol{\lambda})$ be an ultrametric hierarchy, with $\boldsymbol{\lambda} : \mathcal{X} \mapsto [O, L]$ which is strictly increasing with the inclusion order over $\mathcal{H}$. Let us denote $N = card(H)$, $(\mathcal{H}_0, ..., \mathcal{H}_N)$ the nested series of clusterings associated with the hierarchy, which are clusters with numbers of connected components $(n_0, ..., n_N)$, with $0 < n_0 \leq n_1 \leq ... \leq n_N = N$. Then we take as a *normalized ultrametric*:

$$\tilde{\boldsymbol{\lambda}} : \begin{cases} \mathcal{H} \to [0, 1] \\ \mathcal{H}_i \mapsto \frac{N - n_i}{N} \end{cases} \tag{2.11}$$

This corresponds to replacing the diameter of a cluster (i.e. its maximum weight edge in the cluster) with a value proportional to its number of leaves.

After normalization, for two hierarchies $\mathcal{H}_1$ and $\mathcal{H}_2$, the clusters with the same index have the same number of constitutive elements.

An other way of normalizing these values is to do it relatively to the maximum ultrametric value.

Definition 2.32 (Ultrametric Normalization Relatively to the Highest Level Valuation)**.** Let $(\mathcal{H}, \boldsymbol{\lambda})$ be an ultrametric hierarchy, with $\boldsymbol{\lambda} : \mathcal{X} \mapsto [O, L]$ which is strictly increasing with the inclusion order over $\mathcal{H}$. Let us denote $\boldsymbol{\lambda}_{max} = \max(\boldsymbol{\lambda}) = L$. Then we take as a *normalized ultrametric*:

$$\tilde{\boldsymbol{\lambda}} : \begin{cases} \mathcal{H} \to [0, 1] \\ \mathcal{H}_i \mapsto \frac{\boldsymbol{\lambda}_i}{\boldsymbol{\lambda}_{max}} \end{cases} \tag{2.12}$$

Chapter 3

Obtaining and Using Morphological Hierarchies in a Graph-Based Framework

Résumé

Dans ce chapitre, nous exposons des méthodes de segmentation hiérarchique d'images implémentées dans le cadre de la classification hiérarchique sur graphes valués aux arêtes introduite au chapitre précédent. Pour de tels graphes, chaque arête a un poids qui exprime une dissimilarité entre régions de l'image. Les hiérarchies introduites sont agglomératives et construites suivant un mode de construction de type single-linkage. Nous illustrons d'un point de vue qualitatif leurs effets et apportons quelques éclaircissements sur les choix que nous avons faits pour les mettre en œuvre.

Abstract

In this chapter, we expose various image hierarchical segmentations algorithms implemented within the graphs-based hierarchical clustering framework described in the previous chapter on edge-weighted graphs. For such graphs, each edge has a weight that expresses a dissimilarity between regions of the image.The hierarchies introduced are agglomerative and built according to a single-linkage type of construction. We illustrate from a qualitative point of view their effects and provide some insights regarding the choices we made to implement them.

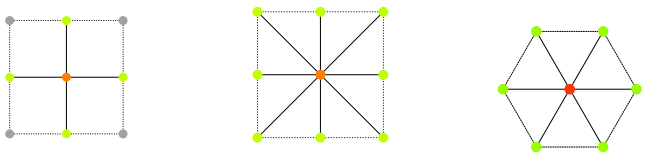
3.1 Graphs in Image Segmentation

In this section, we present the commonly encountered graphs in imaging applications. We consider 2D images whether univariate or multivariate. Note that a 2D color image can be seen as a 3D image with depth 3 (corresponding to the Red, Green, Blue channels for example). In

a first approach, one can use a *pixel adjacency graph*, which consists in a dense graph over all pixels of the image. Otherwise, an unsupervised low-level segmentation of the image (i.e. an over-segmentation of the image) can be used to build a *regions adjacency graph*.

3.1.1 Pixel Adjacency Graphs (PAG)

The easiest case is the one where we construct a dense graph over the image, in which each node corresponds to a pixel of the image and edges link adjacent pixels of the image. This graph is referred to as a "pixel adjacency graph". In order to define such a graph, we only need an image and an adjacency system. Different common adjacency systems are illustrated in figure 3.1.



(a) (b) (c)

Figure 3.1 – Adjacency systems and related pixel adjacency graphs. The figure illustrates some common adjacency systems used in 2D image processing. (a) 4-neighborhood adjacency system (noted V4) and its corresponding adjacency graph (b) 8-neighborhood adjacency system (noted V8) (c) 6-neighborhood adjacency system on a hexagonal grid.

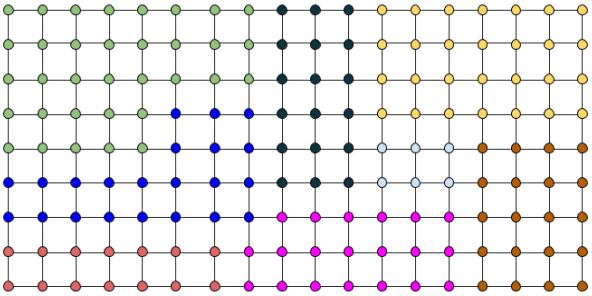
In our applications we will mainly consider the V6 adjacency. An underlying issue is the necessity to find a satisfying trade-off between the algorithmic complexity and the precision of our methods. Most graph algorithms have a complexity dependent on the number of edges in the graph. The adjacency system that we use thus has an important impact on the speed of the methods applied to the graph representing the image. One way to bypass this problem and obtain a higher computational efficiency is to consider a region adjacency graph instead of a pixel one. Like that, one reduces both the number of edges and the number of nodes of the graphs we work with. Furthermore, the fine partition it is built upon is a first partition of the image that does not suffer from the chaining effect that may affect single-linkage hierarchical clustering.

3.1.2 Region Adjacency Graphs (RAG)

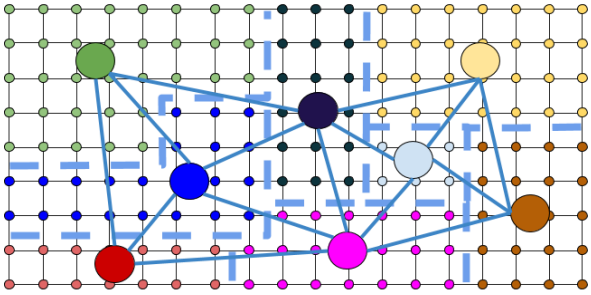
The main type of graphs that we consider in the present work is thus the "region adjacency graph". In this type of graph, the nodes represent regions of the image, and edges link neighboring regions. It is obtained by a low-level segmentation of the image, using unsupervised clustering methods such as a watershed transform [Beucher and Meyer (1992); Machairas et al. (2015)], λ-flat zones labeling [Crespo et al. (1997)] or k-means clustering [Achanta et al. (2012)]. We refer to this low-level segmentation as to a *fine segmentation* of the image. Since this fine segmentation constitutes the departure point for the following graph-based hierarchical segmentation method, its quality is mandatory: it must be sufficiently coarse to turn regions

into meaningful aggregations supports, while still capturing thin images structures. Ideally, all contours of interest of the original image should appear in the fine partition. By contours of interest, we mean contours of the objects or structures of interest in the image.

Once this fine segmentation is obtained, it is modeled by an edge-weighted graph, in which each node of the graph corresponds to one of its regions, and neighbor nodes are linked by an edge, weighted by a dissimilarity measure between its extremities. Since they are far less nodes in a region adjacency graph than in a pixel one, the computational time taken by the applied graph algorithms is reduced. A region adjacency graph is illustrated in figure 3.2.



(a) A fine partition of the image.



(b) Region adjacency graph of the fine partition.

Figure 3.2 – Region adjacency graph: this graph provides a compact representation of the image.

Another important question that arises is the choice of the dissimilarity to value edges between regions. Indeed, in order to work with regions adjacent graphs in segmentation workflows, we have to define a dissimilarity measure between adjacent tiles of the fine segmentation. These dissimilarity measures are usually based on local gradient information (or color or texture).

In the following sections we introduce different hierarchical clustering constructions and evaluate qualitatively the impact of these different possible choices on the output of our workflow.

3.2 Equivalent Global Representations of Hierarchical Segmentations

As we have seen in section 2.3.2, there are several different ways to represent and work with hierarchies obtained with single-linkage hierarchical clustering methods. We now present and discuss their specificities.

3.2.1 Dendrogram tree structure

First and foremost, single-linkage hierarchical clustering can be expressed in a minimum spanning forest framework as a series of nested forests with minimal possible total weight at each level. Such a hierarchical clustering can be obtained starting from a minimum spanning tree of the graph.

Let us consider a fine segmentation of the image into regions partitioning the domain $\mathcal{D}$ of the image $\mathbf{I}$, such that $\mathcal{D} = \{\mathsf{R}_i\}_i$. As previously stated, a hierarchical segmentation is then a family of partitions $(\mathsf{P}_0, \mathsf{P}_1, ..., \mathsf{P}_n)$, such that: (i) P_0 corresponds to the fine partition, (ii) P_n is the complete domain, (iii) regions from coarse levels are unions of regions from fine levels, i.e. for each i, $\mathsf{P}_i \subset \mathsf{P}_{i+1}$. Since to each level P_i of the hierarchy we can assign a real-valued index λ_i, we can represent it as a stratified dendrogram, i.e. a region tree where the height of each node is its index. Indeed, sweeping through all λ_i values can therefore be represented as a region tree, whose root is the region representing the whole image. The stratification index values are given by the contour strengths with which successive mergings are associated. An illustration of such a process in the context of hierarchical image segmentation is provided in figure 3.3. This representation of a hierarchical segmentation through the structure of a dendrogram, notably illustrated in section 2.3.2, allows for a fast obtention and handling of hierarchies using dynamic programming.

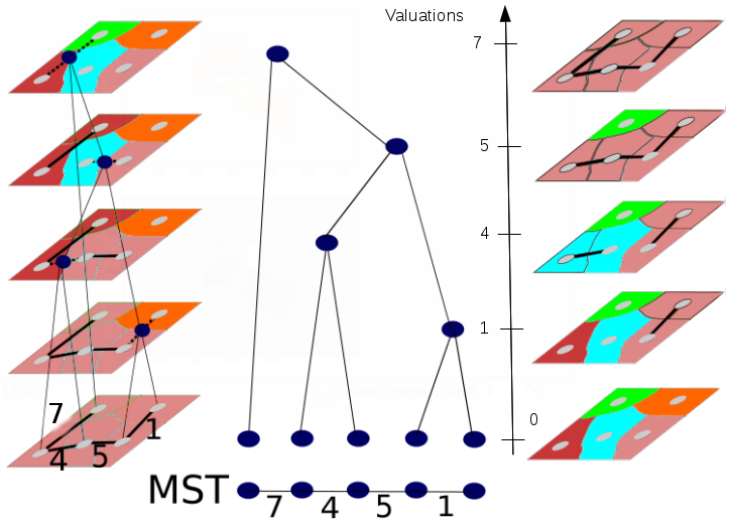


Figure 3.3 – Progressive fusion of regions as the cut level λ increases.

We can thus work with these dendrograms to compute dynamically new valuations for the MST edges that reflect certain properties of the image. A new hierarchical structure automatically stems from these new valuations once internal nodes of the MST have been reorganized accordingly. This is illustrated in figure 3.4.

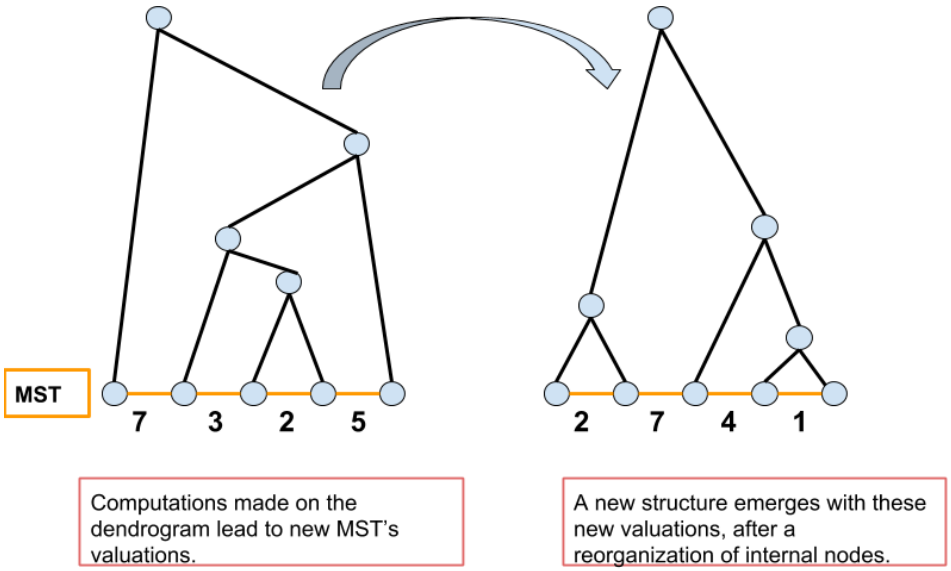


Figure 3.4 – Reorganization of internal nodes of the dendrogram once new valuations for the MST edges have been computed.

3.2.2 Ultrametric Contour Map (UCM), or Saliency Map

Furthermore, a hierarchical segmentation can also be represented as an ultrametric contour map ($\mathcal{UCM}$), which is an image obtained by weighting the boundary of each pair of adjacent regions in the hierarchy by the stratification index for which they are merged [Najman et al. (1996); Arbelaez (2006)]. The RAG regions belong to a connected fine partition of the image, that can possibly be represented by a label image. In this fine partition, regions are side by side and theirs frontiers are not represented. For visualization purposes, we can do an oversampling of scale 2 of the image in each dimension, thus creating room for contours. A pixel having two neighbors with different labels then belongs to a contour. Furthermore, we can attribute a gray-level value to each contour, proportional to the saliency between its two adjacent regions. This representation is useful for illustration purposes, as it provides a way to understand the relative strengths of contours in a hierarchy.

Note also that one of the main properties of a $\mathcal{UCM}$ is that by thresholding it and keeping only contours above a certain value, we obtain a closed boundary map, and thus a partition. Thus, thresholding the $\mathcal{UCM}$ for increasing λ_i leads to a merging-sequence partition: a step in this sequence corresponds to merging the set of regions sharing the boundary of strength exactly λ_i. For better visibility, we represent in this thesis the $\mathcal{UCM}$ with inverted contrast. An example of a $\mathcal{UCM}$ for the hierarchy directly derived from the initial dissimilarity (based on a local gradient information) is provided in figure 3.5.

As it can be seen in the example above (figure 3.5), it is clear that when departing from the hierarchy associated with the initial dissimilarity, the more salient contours are the ones of small contrasted regions. These are generally not the contours that we want to be highlighted by our hierarchical segmentation framework. It is linked with the fact that the dissimilarity of a contour

(a) Image (b) Trivial hierarchy $\mathcal{UCM}$

Figure 3.5 – Example of an Ultrametric Contours Map (UCM) with inverted contrast. It corresponds to the *trivial* hierarchy, i.e. the hierarchy associated with the initial dissimilarity.

then corresponds to the altitude of its saddle point, and thus to the lowest gradient value along this contour. And thus the longer the contour, the higher the chances to meet such a weak point.

This hierarchy can also suffer from the chaining effect we exposed in section 2.2.5, since the dissimilarity we use only takes into account a local geometric information such as the gradient. Hence the need to create better dissimilarities that build upon several scales and information sources.

It has to be kept in mind that the final goal of hierarchical segmentation is often to extract a segmentation of the image, and that the hierarchical segmentation constitutes a way to structure the data in order to make this extraction simpler. When looking at the first regions proposed in the hierarchy, they are often too rough. We must thus reorganize the partitions, in order for them to be closer to the expected partition. In the following section, we introduce different morphological hierarchical clusterings to do so.

3.3 Morphological Hierarchies on Graphs

Departing from a graph constructed over a fine partition of the image, one can imagine several ways to operate a hierarchical clustering of the nodes of the graph, and in particular to make use of the methods presented in chapter 2.

In this section, we illustrate schemes to obtain more pertinent dissimilarities in order to get new ultrametrics and thus different and complementary hierarchical clusterings starting from the same graph. These schemes are largely inspired by mathematical morphology approaches to obtain hierarchies of segmentations on images. There is especially a strong connection between single-linkage hierarchical clustering and hierarchies of flooded watersheds, that we describe hereafter.

3.3.1 Intrinsic connection between single-linkage hierarchical clustering and hierarchies of flooded watershed of a topographic relief

Most of the graph-based hierarchical segmentations techniques we introduce in this chapter take inspiration from classical morphological methods. In this section, we explain the intrinsic connection that exists between the single-linkage hierarchical clustering method that has been introduced in chapter 2, and hierarchies of flooded watersheds of a topographic relief.

Watershed and floodings: absorption of catchment basins during flooding

If g is a flooding of a reference function f, considered here as a topgraphic surface, we may imagine a progressive flooding of f producing g. During this flooding process, the catchment basins (CB) of f merge according to two mechanisms : a) the level of the water reaches a saddle point within a CB X but has a lower level in the neighboring catchment basin Y: X is absorbed by Y b) two lakes, previously disconnected merge, leading their corresponding CB to merge ; such mergings also occur at the localization of a saddle point. As a result, each CB of g is either identical with a CB of f or equal to the union of several CB of f.

As a result, the CBs of a flooded surface form a coarser partition than the CBs of the initial surface. This leads, for increasing floodings, to a hierarchy of CBs.

Hierarchy of the catchment basins

Let us indeed now consider a family F of increasing floodings $(g_i)_{i\leq N}$ of a reference function f, verifying $g_i \leq g_j$ for $i < j$, and $g_0 = f$. The set of catchment basins of g_i is called A_i ; $A = \bigcup A_i$ is the set of all catchment basins. Each A_i is a partition of E. The partitions are nested: each element a of A_i is the union of CB of A_{i-1}.

This corresponds to a hierarchy of segmentations associated with the watershed transform. To each particular flooding the watershed associates a partition. For a higher flooding, the partition is coarser, and is obtained by merging tiles of the finer partition. The successive partitions form a hierarchy.

We can now indicate useful families of floodings from which to derive hierarchies of segmentations. As a matter of fact, the quality of segmentation will depend to a great extent of the family of floodings on which it is build.

1. **Uniform flooding:** A flooding of a function f is uniform if the level of all lakes is at the same level λ. This is the simplest but also the least informative flooding, as the only criterion which is taken into consideration is the height of the pass-point separating two catchment basins. Nevertheless, it is this type of flooding which underlies the traditional method of morphological segmentation, based on the watershed. The watershed construction itself is based on uniform flooding. It is illustrated in figure 3.6, and constitutes a singular case of synchronous flooding, since the altitude of each lake is synchronized according to a global

parameter that is uniform through the whole image. For a uniform flooding, we obtain the *trivial hierarchy*.

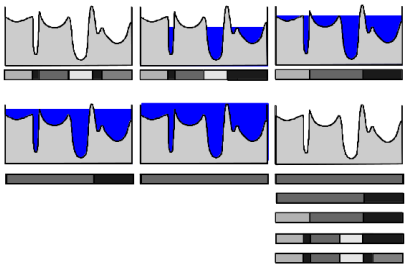


Figure 3.6 – The trivial hierarchy can be seen as a uniform flooding of the gradient image. The gradient image is flooded with the same flooding level. Its watershed produces a partition. For increasing flooding levels, the partitions get coarser. This hierarchy is often not very interesting.

2. **Size oriented flooding:** Controlling the flooding, notably by imposing non-uniform flooding schemes, can lead to different interesting hierarchies, as it is illustrated in figure 3.7.

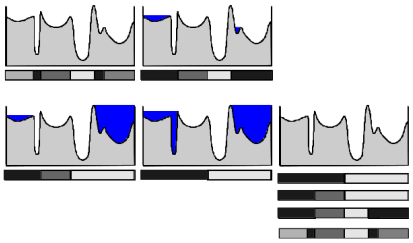


Figure 3.7 – We can imagine different criteria to control the image flooding. Each of them will enhance the saliency of different types of contours.

Size oriented flooding is produced by placing sources at each minimum and flooding the surface in such a way that all lakes share some common measure (height, volume or area of the surface). As the flooding proceeds, the level of some lakes cannot grow any further, as the level of the lowest pass point has been reached. Later, when the neighboring lake has reached the level of this same pass point, both lakes continue to grow together. In the figure 3.8, a flooding starts from all minima in such a way that all lakes always have a uniform depth. The resulting hierarchy is called dynamics in case of depth driven flooding and has first been introduced by M.Grimaud [Grimaud (1992)]. Deep catchment basins represent objects which are contrasted ; such objects will take long before being absorbed by a neighboring catchment basin. The most contrasted one will absorbe all others. This criterion obviously takes only the contrast of the objects into account and not their size. If we control the flooding by the area or the volume of the lakes [Vachier et al. (1995)], the size of the objects also is taken into consideration ; in multimedia applications, good results are often obtained by using as a parameter controlling the flooding the volume of the lakes, as if each source would pour water with a constant flow. As a summary, the

depth criterion ranks the region according to their contrast, the area according their size and the volume offers a nice balance between size and contrast.

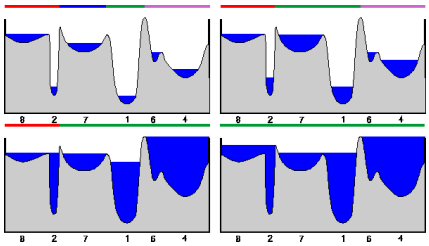


Figure 3.8 – Example of a height synchronous flooding. Four levels of flooding are illustrated ; each of them is topped by a figuration of the corresponding catchment basins.

Note that whenever the chosen type of flooding, the distance between two regions of the image can then be defined as the level of the smallest flooding for which these two regions belong to the same lake. This corresponds exactly to the ultrametric distance between elements of a set in a single-linkage hierarchical clustering scheme.

Interpreting the RAG as a topographic relief

The ultrametric hierarchies considered in this thesis have in common to rely all on a fine partition with a dissimilarity index between adjacent tiles.

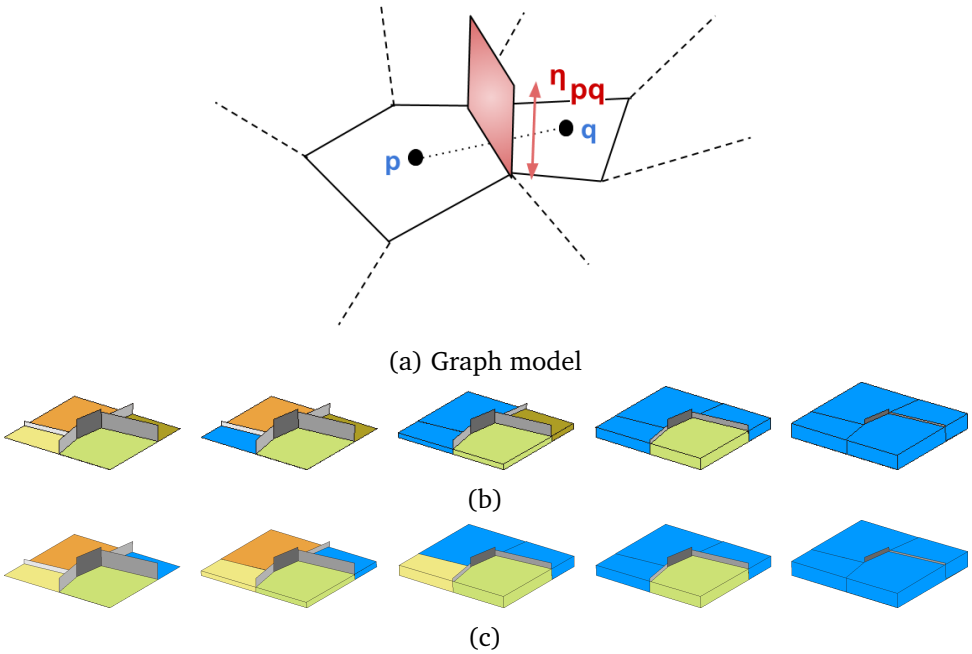
In order to be able to apply the same types of hierarchical clusterings based on flooding a topographic surface, we may associate a virtual topographic relief to the partition. One can separate the adjacent regions of the fine partition by a wall of 0 thickness and with a height equal to the dissimilarity between the tiles. It is illustrated in figure 3.9(a). This modelization presents the benefit of being applicable to any RAG defined over the a partition of the image, with adjacent regions connected by edges weighed according to any given dissimilarity. With this model in mind, the single-linkage hierarchical clustering scheme can be interpreted as a flooding of the graph, as shown in figure 3.9(b)(c). The flooding process then depends on the ultrametric distance between nodes of the graph.

As in the image case, one can choose a flooding mode that transcribes some desirable properties of the output clusterings.

Illustrations of those three types of hierarchical clusterings are provided for the same image in figure 3.10. The depth criterion ranks the regions according to their contrast, the area criterion according to their size, and the volume criterion offers a good trade-off between size and contrast. The volume-based hierarchy notably provides interesting results for multimedia applications.

3.3.2 Hierarchies of Flooded Watersheds in the MSF Framework

In section 3.3.1, we have introduced a modelization to translate the flooding process on partitions with a dissimilarity index between adjacent regions. The ultrametric distance between two nodes corresponds to the minimum flooding level for which these two nodes belong to the



(a) Graph model

(b)

(c)

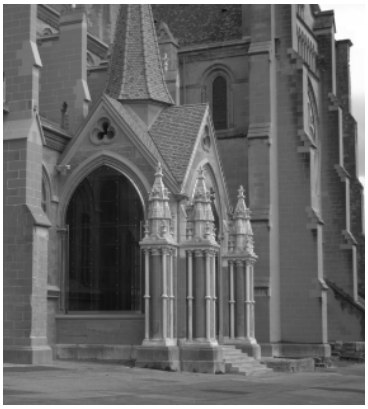
Figure 3.9 – (a) Graph model. Two nodes/catchment basins p and q are separated by a frontier of height η_{pq} modelized as an edge with similar weight between them. (b)(c) These figures illustrate how a progressive flooding for any given criteria ends up progressively merging the regions, thus creating a hierarchy.

same lake. This distance χ may be called flooding distance and is an ultrametric distance. The ultrametric inequality $\chi(p,q) \leq \chi(p,r) \vee \chi(r,q)$ has a simple interpretation. Indeed, $\chi(p,q)$ is the level of the lowest lake containing p and q, whereas $\chi(p,r) \vee \chi(r,q)$ is the level of the lowest lake containing all three nodes p, q and r.

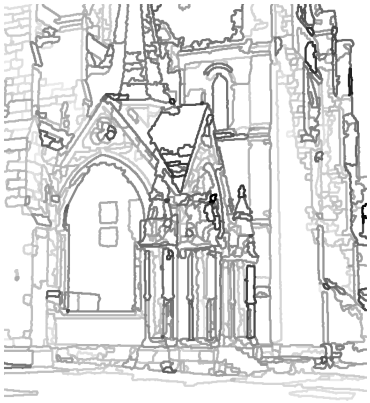
Then finding the distinct lakes on the graph for a given flooding level can be done by removing edges of valuations superior to a threshold λ on a minimum spanning tree of the graph. Indeed, such a tree encodes a hierarchical clustering, hence the valuation of each of its edges is equal to the ultrametric distance between its ends. When working with the initial dissimilarity defined over the graph, generally linked with the gradient between adjacent regions, one get a trivial hierarchy such as the one exposed before. Such a hierarchy only reflects a local contrast, so that the most salient regions in the image are the small contrasted ones.

Departing from this trivial hierarchy, one can get an interesting segmentation by marking some nodes as important and then computing a partition accordingly (markers-based segmentation), or by smartly editing the graph by finding the partition that minimizes an energetic function.

In a complementary approach, the quality of the obtained partitions also highly depends on the dissimilarity that we use, and thus changing this dissimilarity can lead to more suitable partitions. Instead of departing from a simple and rough dissimilarity such as contrast and then use a sophisticated technique to get a good partition out of it, one can also try to obtain a more informative dissimilarity adapted to the content of the image such that the simplest methods are sufficient to compute interesting partitions. This way, the aforementioned techniques lead to



(a) Image



(b) Trivial hierarchy

(c) Surface-based hierarchy

(d) Volume-based hierarchy

Figure 3.10 – Example of saliency images for the trivial, surface-based and volume-based hierarchies.

segmentations better suited for further exploitation. Such a remark can be linked with the usual critics that is formulated towards single-linkage hierarchical clustering regarding the chaining effect that it can suffer from. Indeed, one can avoid such a chaining effect by constructing dissimilarities that reflect more than only very local properties.

How can we construct these more pertinent and informative dissimilarities?

3.3.3 A Minimum Spanning Forest Associated To Markers

When having knowledge about the nodes we want to be separated in the output segmentation, one way to construct more pertinent and informative dissimilarities is to select a node in each region or object of interest that will serve as a root in each wanted tree. The roots are also called *markers* and this process referred to as *marker-based segmentation*. When working with the topographic relief flooding analogy, the marker-based segmentation corresponds to a flooding for which we have selected the markers as flooding sources. The catchment basins which do not contain any marker are then flooded starting from their neighbors.

Marker-based segmentation directly on the $\mathcal{MST}$ is possible [Meyer (1994a)]. Cutting all edges of the MST with a weight higher than λ creates a spanning forest. Among all forests with

the same number of trees, this forest is a minimum spanning forest MSF_λ, the sum of its weights being minimal. Two nodes p and q belonging to the same tree of the forest have a distance $\delta_{pq} < \lambda$: they belong to the same ball $\mathcal{B}(p, \lambda)$. Hence the trees of the forest and the balls $\mathcal{B}(p, \lambda)$ induce the same partition of the nodes. As shown above, this partition often does not well represent the salient features of an image.

More interesting partitions are obtained with the same number of trees, if we chose the roots of the trees. We select a subfamily (m_i) of nodes, (also called markers) within N and construct a minimum spanning forest where each tree is rooted in a marker. Each minimum spanning forest is obtained by cutting some edges of the MST. Consider two consecutive markers m_1 and m_2 on the MST, such that there exists no other marker along the path along the MST joining both markers. In order to get a forest, one has to cut an edge along this path ; in order to minimize the total weight of the edges, one cuts the highest edge. The same process applied to all pairs of edges produces the desired minimum spanning forest [Meyer (1994a)].

Consider two consecutive markers m_1 and m_2. Suppose that the highest edge e_{pq} on the path of the MST linking both markers has a weight λ. If p and m_1 (resp. q and m_2) are connected after cutting e_{pq}, then the altitude of the path linking p with $m1$ (resp. q with m_2) is lower then λ. This gives us a criterion for recognizing whether a given edge of the MST belongs or not to the MSF associated to a family of markers: the edge e_{pq} with weight λ does not belong to the MSF if and only if there exists 2 paths with an altitude lower than λ, one linking p with a marker and another linking q with another marker. Or equivalently, if the balls $\mathcal{B}(p, \lambda)$ and $\mathcal{B}(q, \lambda)$ contain each at least one marker. This criterion will be used all along of this section for deriving various features-driven hierarchies.

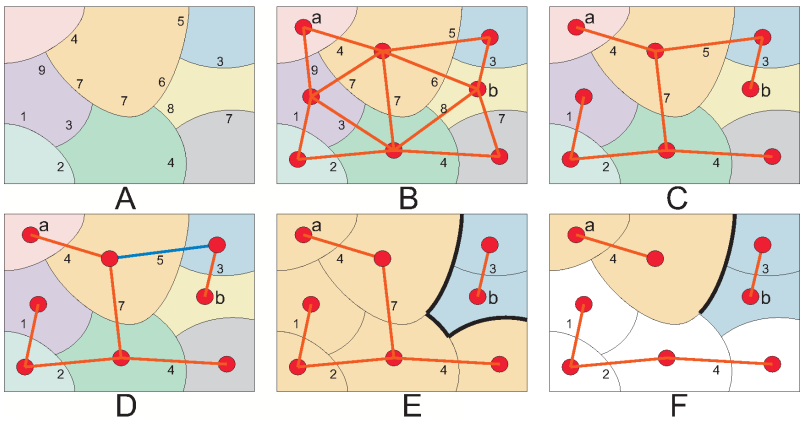


Figure 3.11 – A partition with dissimilarities in **A**, its region adjacency graph in **B**, the minimum spanning tree in **C**, the choice of two markers in **D**, the resulting segmentation in **E**, and two regions which should each contain a marker for generating the portion of contour (bold line) between them in the final segmentation.

Illustration and analysis: **A**: A partition in which each tile has a different color. A dissimilarity is defined between any couple of adjacent tiles.

B: The region adjacency graph is created: each node of the graph represents a region. An edge

links the nodes representing two adjacent regions. This edge is weighted by the dissimilarity between both regions.

C: A minimum spanning tree of the region adjacency graph

D: 2 nodes are chosen as markers: the nodes a and b. They are linked by a unique path on the minimum spanning tree. The highest edge along this path has a weight equal to 5. Cutting this edge cuts the minimum spanning tree in two subtrees $\mathcal{T}_a$ and $\mathcal{T}_b$, spanning each a region.

E: Both trees form a minimum spanning forest, where each tree is rooted in a marker. The regions spanned by these trees constitute the resulting segmentation associated to the markers a and b

F: If we cut all edges of the MST with a weight ≥ 5, we create two subtrees Θ_a and Θ_a, containing respectively the nodes a and the node b. These trees span two regions, one in orange color, the other in blue color. The boundary between these two regions will appear in a marker-based segmentation, if there is at least one marker in each of them.

Note that this process is robust to the choice of markers, since the selection of any node of a region as a root leads to a segmentation of this region.

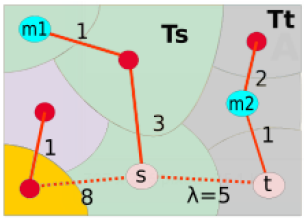


Figure 3.12 – A MST with markers. Considering two markers $m1$ and $m2$ falling into the regions spanned by the two subtrees $\mathcal{T}_s$ and $\mathcal{T}_t$ obtained when cutting an edge e_{st} on the MST, the highest edge on the path linking them is indeed e_{st}.

3.3.4 A Hierarchy Based On Prioritized Markers

The previous section has explained how to associate a partition to a family (m_i) of the nodes taken as markers. Let F be the minimum spanning forest associated to these markers. Suppose that we add a new marker n. A marker m_k of the family is a neighboring marker of n if there exists a path between n and m_k along the MST on which there is no other marker. Such a path belongs to the tree T_k rooted in m_k. The highest edge along this path has to be cut. Like that the tree T_k is cut in two parts. Hence, by adding new markers, one obtains finer partitions [Meyer (1994a)].

Consider now a family of markers ranked according to some priority (m_i). We want to construct a hierarchy associated to this family. The coarsest level of the hierarchy is the partition associated to the markers with the highest priority. Every time we add a marker, we obtain a finer partition, as a tile of the coarser partition is cut in several parts. Our goal is to define new weights θ_{pq} for the edges e_{pq} such that cutting all edges with a weight above k produces a minimum spanning forest associated to the k markers with the highest priorities.

The edge e_{pq} with weight $\eta_{pq} = \lambda$ does not belong to the MSF if the balls $\mathcal{B}(p, \lambda)$ and $\mathcal{B}(q, \lambda)$ contain each at least one marker. If there is no marker at all in one of the balls $\theta_{pq} = 0$. If μ_p and μ_q are the highest priorities of the markers present respectively in $\mathcal{B}(p, \lambda)$ and $\mathcal{B}(q, \lambda)$, then, by choosing all markers with a priority higher or equal than $\mu_p \wedge \mu_q$, there will be a marker in each of the balls. If we assign to the edge e_{pq} the weight $\mu_p \wedge \mu_q$, we obtain the desired result.

The algorithm visits all edges of the MST in the order of increasing weights. Repeat until all edges are processed:

Let e_{pq} the current edge to process with a weight λ.

If μ_p and μ_q are the highest priorities of the markers present respectively in $\mathcal{B}(p, \lambda)$ and $\mathcal{B}(q, \lambda)$, we assign to the edge e_{pq} the weight $\mu_p \wedge \mu_q$.

Illustration and analysis: In figure 3.13 a number of prioritized markers have been introduced ; they appear in the second image as disks whose brightness is proportional to the priority. The saliency of the hierarchy is indicated in the same image. The boundary between two regions has a shade of grey proportional to the hierarchy level for which it disappears. The last 4 images represent 4 partitions of the associated hierarchy with decreasing coarseness.

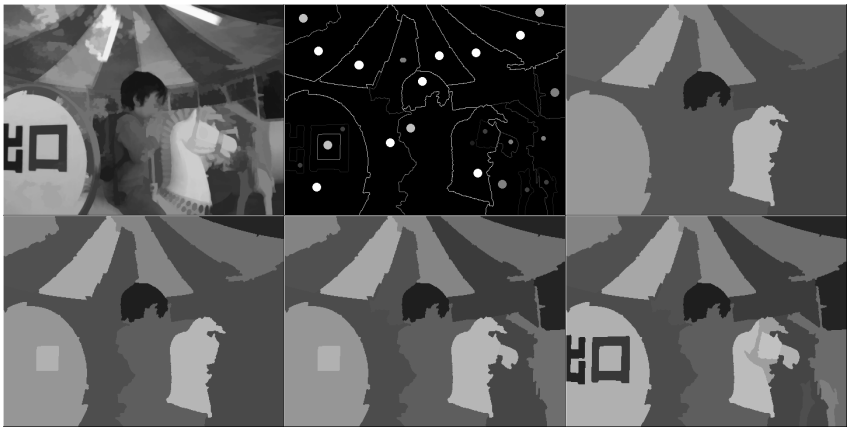


Figure 3.13 – A number of prioritized markers have been chosen: they appear as disks whose shade of grey is brighter for higher priorities. The associated hierarchy is illustrated through the saliency of the contours in the second image and through 4 levels of the associated hierarchy.

3.3.5 On the multiple-MST possibility

Marker-based segmentation may be done on the graph itself or on any of its minimum spanning trees. The result then depends on the particular choice of the MST.

Furthermore, when several MST of the graph exist, the arbitrary choice of one instead of an other can lead to different output clusterings, and some segmentations that do not seem natural can be obtained.

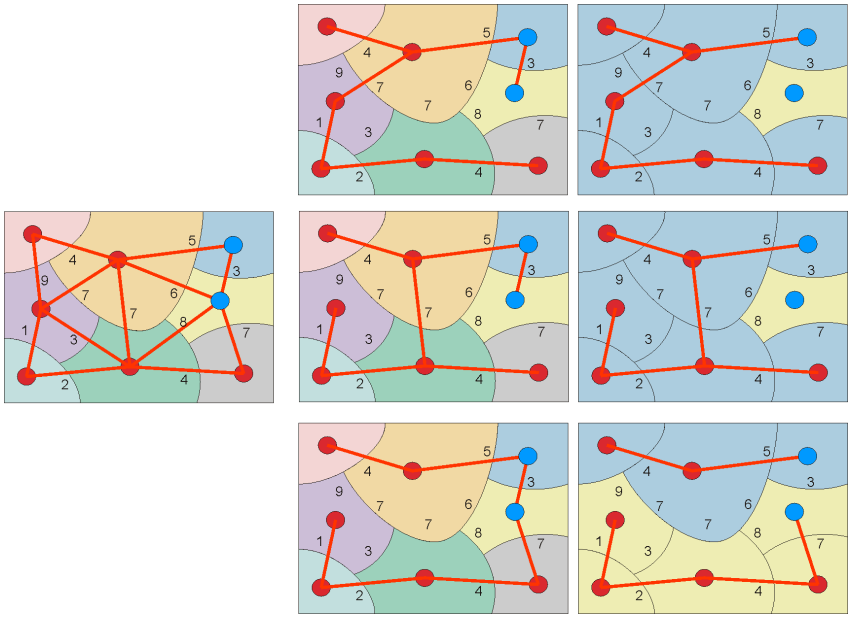
An example of how different equivalent MST can lead to different segmentations is provided in figure 1.10 of chapter 1. We illustrate this phenomenon for MST-based image segmentation in figure 3.14. Figure 3.14(a) shows how the ultrametric distance can be short-sighted, in the sense that the same ultrametric distance can correspond to different MST and thus different segmentations when doing marker-based segmentations. In figure 3.14(b), we compare the

different possible segmentations obtained with these different MST, with the one obtained using the classical hierarchical waiting queue watershed algorithm on the graph itself. It turns out that the result (which is the desired result) corresponds to the MST privileged when using the lexicographic distance between nodes to discriminate between the possible MST [Meyer (2005)].

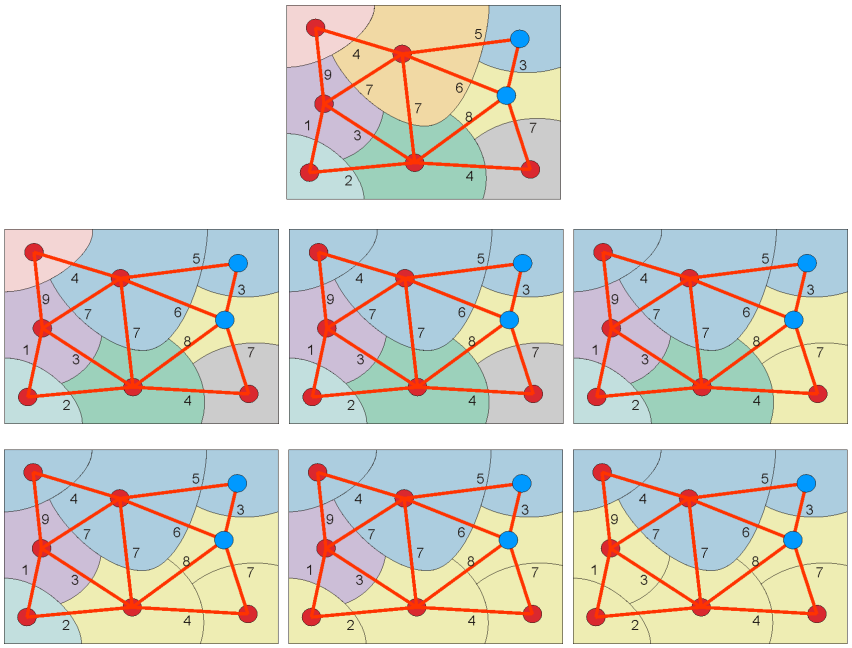
Thus two ways to handle the multiple-MST case can be thought of, as announced in section 2.3.3:

— Select one MST over the other possible ones by using additional criteria (as through the use of the lexicographical distance for example).

— Consider all MST at the same time during computations.



(a) Marker-based segmentation for the same 2 markers, but on different equivalent MST. Cutting the highest-weight edge between the two markers for the different MST does not lead each time to the same segmentation. Is it possible to know which is the best?



(b) We compare the result above with the segmentation obtained by using the zones of influence of the markers with a shortest ultrametric path algorithm scheduled by a hierarchical waiting queue. The final segmentation corresponds to the third minimum spanning tree in the above figure.

Figure 3.14 – Example of the lack of robustness of MST-based HC

3.3.6 Stochastic Watershed Hierarchies

Concept

Section 3.3.4 has shown how ranking the markers generates a hierarchy. We now replace deterministic markers by stochastic markers. Rather than using deterministic markers in the marker-based segmentation presented in section 3.3.3, one can indeed instead use random markers following a given distribution and thus generate random MSF. Then, one can assign to each edge of the MST the probability of appearance of the underlying contour. Depending on the distribution of markers used, the obtained segmentation can be very variable.

This idea finds its source in the stochastic watershed presented in [Angulo and Jeulin (2007); Angulo, Velasco-Forero, et al. (2009)]. If we see the image as a topographic relief, flooding this image leads to watershed lines, i.e. to a segmentation. By spreading random flooding sources multiple times and flooding the image accordingly, one can characterize each contour of the image by its frequency of appearance in the associated segmentations. This simulated version of the stochastic watershed is illustrated in figure 3.15. Large regions, separated by low contrast gradient from neighboring regions will be sampled more frequently than smaller regions and will be selected more often. On the other hand, strong contours will often be selected by the watershed construction, as there are many possible positions of markers which will select them. In [Bernander et al. (2013)], the authors introduce noise on the image and the use of a random grid to distribute markers to obtain more satisfying results with this technique. In [Franchi et al. (2015)], a re-sampling and bagging technique is introduced for the same purpose.

Evaluating the strength of the contours by simulation offers a great versatility : various laws for the implementation of point patterns, various shapes for the markers themselves may be used. The method suffers however from a serious handicap, if the contour strength is evaluated through simulations, as each of them requires the construction of a watershed segmentation. We show below, not only how simulations may be avoided, but also how to imagine scenarios which would be difficult or even impossible to simulate [Meyer and Stawiaski (2010)].

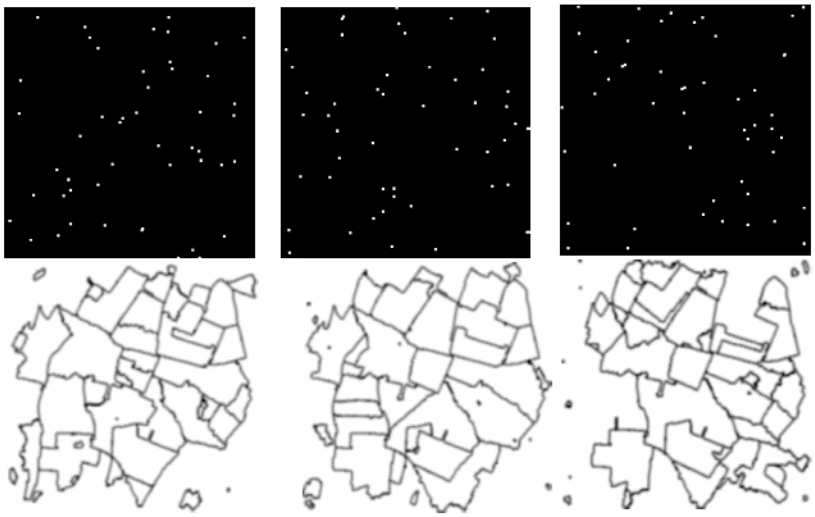
As we shall see hereafter, we can indeed obtain similar results with computations made directly on graphs, without the need for any simulation [Meyer and Stawiaski (2010)]. Furthermore, it also permits to construct stochastic watershed hierarchies with models which would have been otherwise difficult to simulate. Indeed, we can define many ways to generate markers, depending on the probability law used to implant them, but also on their sizes or shapes if they are sets. Each particular mechanism favors the emergence of a certain type of regions to the detriment of others. We take advantage of this versatility to compute new hierarchies that are more linked with the semantic information present in the image.

Principle of the Method

We imagine that we draw random germs on the domain where the image is defined and compute the probability of each piece of contour to appear in the associated segmentation. We

(a) Initial image

(b) Gradient image



(c) Watershed segmentations obtained by flooding the gradient image for 3 random markers distributions.

(d) Probability density function of the contours computed from 50 realisations of random markers.

Figure 3.15 – Illustration of the stochastic watershed by simulation.

have to assign to each edge e_{pq} of the MST with an initial weight η_{pq} a new weight θ_{pq} equal to the probability to appear as a contour. In a first stage we only consider points a markers. Later we will also considers arbitrary, stochastic or deterministic, sets as markers.

As shown above, the edge e_{pq} with weight $\eta_{pq} = \lambda$ does not belong to the MSF if the balls $\mathcal{B}(p, \lambda)$ and $\mathcal{B}(q, \lambda)$ contain each at least one marker. Thus the probability θ_{pq} is equal to the probability that there is at least one random marker in each of the balls $\mathcal{B}(p, \lambda)$ and $\mathcal{B}(q, \lambda)$.

For the sake of simplicity, we chose a Poisson distribution of germs over the domain. We fix the number of germs to be equal to ω ; the distribution is then uniform. Consider a set X of area A within a domain D of area S. The probability that there falls no one germ within the domain X is then $\left(1 - \frac{A}{S}\right)^{\omega}$. And the probability that there is at least one germ in X is then $1 - \left(1 - \frac{A}{S}\right)^{\omega}$.

Supposing that the Poisson distribution has an homogeneous density λ:

$$\Lambda(R) = \texttt{area}(R)\lambda, \tag{3.1}$$

Absorption of the smallest region

Area Oriented Absorption Consider the edge e_{pq} with weight $\eta_{pq} = \lambda$ and the balls $\mathcal{B}(p, \lambda)$ and $\mathcal{B}(q, \lambda)$. Let a_p and a_q be the areas of these balls. We place a deterministic marker in the region with the largest area and a random marker in the smallest. The probability θ_{pq} is then equal to the probability that there exists at least one random marker in the smallest region of area $a_p \wedge a_q$, i.e.:

$$\theta_{pq} = 1 - \left(1 - \frac{a_p \wedge a_q}{S}\right)^{\omega}. \tag{3.2}$$

This hierarchy is also referred to as area-based SWS hierarchy by extinction. It is the stochastic counterpart of the surface extinction hierarchy associated with flooding, which is deterministic.

"**Volume" Oriented Absorption** The previous criterion is based on the area of the balls $\mathcal{B}(p, \lambda)$ and $\mathcal{B}(q, \lambda)$. For high values of λ this area is likely to be larger than for small values. However, in order to reinforce the influence of the contrast, one may multiply the areas of the balls $\mathcal{B}(p, \lambda)$ and $\mathcal{B}(q, \lambda)$ by the value λ. This product λa_p may be considered as a kind of volume. Let $\lambda_{\max}$ be the highest weight of the edges of the MST. The probability that no marker falls within the volume λa_p within the total volume $\lambda_{mas} S$ is then:

$$\theta_{pq} = \left(1 - \lambda \frac{a_p \wedge a_q}{S * \lambda_{\max}}\right)^{\omega}. \tag{3.3}$$

Remark that whereas the absolute values of λ depend upon the global contrast of the image, the evaluation of the contour strength is nevertheless relatively robust against the change of contrast, as it is based on the ration $\lambda/\lambda_{\max}$.

This hierarchy is also referred to as volume-based SWS hierarchy by extinction.

Contrast Oriented Absorption Consider again the watershed segmentation. If the image on which the watershed is constructed is a gradient image, the significant features are the levels of the pass points between adjacent regions ; the level of the minima, often near to 0, has not much signification. In other situations one has to construct the watershed on images of a different type, for which the levels of the minima is significant. For instance, the micro-aneurisms in a retina appear as dark spots for which the level of the minima is significant. Another example is the segmentation of text on a document. In such situations, the noise often appear also as dark spots with less contrast. With the stochastic watershed less contrasted regions get absorbed by more contrasted regions. We measure the contrast of the ball $\mathcal{B}(p,\lambda)$ as the difference between λ and the deepest value ζ_p taken by the image in $\mathcal{B}(p,\lambda)$. We put a hard marker in the most contrasted region and compute the probability that there is a marker in the less contrasted region for ω markers uniformly distributed in the range $[0,\zeta_p]$, yielding:

$$\theta_{pq} = \left(1 - \frac{\lambda - \zeta_p \wedge \zeta_q}{\lambda_{\max}}\right)^{\omega}. \tag{3.4}$$

The Symmetrical Stochastic Watershed

The Area Based Stochastic Watershed We now consider the distributions of markers in both balls $\mathcal{B}(p,\lambda)$ and $\mathcal{B}(q,\lambda)$. In short we write $B_p = \mathcal{B}(p,\lambda)$ and $B_q = \mathcal{B}(q,\lambda)$. The weight θ_{pq} of the edge e_{pq} is then equal to the probability of the event: $E =$ {there is at least one marker in B_p} and {there is at least one marker in B_q} The opposite event is the union of two non exclusive events: $\mathrm{no}\, E =$ {there is no marker in B_p} or {there is no marker in B_q}.

Its probability is: $P(\mathrm{no}\, E)$ = P {there is no marker in B_p} + P {there is no marker in B_q} - P {there is no marker in $B_p \cup B_q$}. And:

$$\theta_{pq} = P(E) = 1 - \left(1 - \frac{a_p}{S}\right)^{\omega} - \left(1 - \frac{a_q}{S}\right)^{\omega} + \left(1 - \frac{a_p + a_q}{S}\right)^{\omega}. \tag{3.5}$$

An example of the surface-based SWS are given in figure 3.16.

Figure 3.16 – Example of the effect of a surface-based SWS. Note that the big objects are better highlighted in comparison with the trivial hierarchy (figure 3.5)

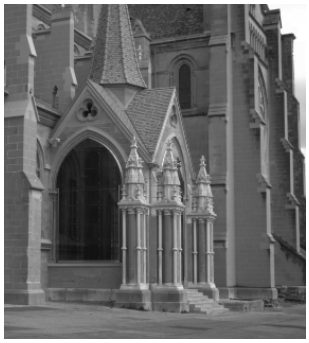
The Volume Based Stochastic Watershed For stressing more the strength of the gradient separating both regions $\mathcal{B}(p,\lambda)$ and $\mathcal{B}(q,\lambda)$, we replace the measures of the areas a_p and a_q by

the pseudo volumes λa_p and λa_q. The markers being distributed in a total volume $S \times \lambda_{\max}$. The probability that there exists at least one markers in both "volumes" is then:

$$\theta_{pq} = P(E) = 1 - \left(1 - \lambda \frac{a_p}{S \times \lambda_{\max}}\right)^{\omega} - \left(1 - \lambda \frac{a_q}{S \times \lambda_{\max}}\right)^{\omega} + \left(1 - \lambda \frac{a_p + a_q}{S \times \lambda_{\max}}\right)^{\omega} \tag{3.6}$$

An example of the volume-based SWS is given in figure 3.17.

Figure 3.17 – Example of the effect of the volume-based SWS.



(a) Image

(b) Trivial hierarchy.

(c) Surface-based SWS hierarchy.

(d) Volume-based SWS hierarchy.

Figure 3.18 – Comparison of surface-based and volume-based SWS hierarchies. In comparison with the surface-based SWS, the volume-based SWS favors contours that are not only connecting regions with important surfaces but also different from one another (with a high gradient between them). This is why the structures in the upper-right part of the image are more highlighted in (c) than in (d).

The Symmetrical Stochastic Watershed Within Transformed Domains

Until now we considered the domains $\mathcal{B}(p,\lambda)$ and $\mathcal{B}(q,\lambda)$ only through their area or the deepest value taken by the image within the balls. In order to now take into account also their shape, we apply an anti-extensive morphological operator ψ on the balls: $\psi(X) \subset X$. The area of $\psi(X)$ is thus smaller than the area of X. The most common operators are the erosion and the opening. This opens a large choice of possibilities : erosion or opening, type of structuring elements (often disks or segments in various directions), size of the structuring element, etc.

We define $\beta_p = \text{area}[\psi\mathcal{B}(p,\lambda)]$. The probability θ_{pq} to be assigned to the edge e_{pq} is then:

$$\theta_{pq} = 1 - \left(1 - \frac{\beta_p}{S}\right)^{\omega} - \left(1 - \frac{\beta_q}{S}\right)^{\omega} + \left(1 - \frac{\beta_p + \beta_q}{S}\right)^{\omega}. \tag{3.7}$$

It is noteworthy that this assignment of probabilities cannot be obtained by the simulation method, consisting in introducing real random germs in the image and constructing the watershed partition for each new simulation.

This hierarchy will respond to the presence of particular structures in the image such as elongated and thin structures or on the contrary to massive structures. It also permits to analyze the anisotropies of an image.

The Symmetrical Stochastic Watershed With Non Punctual Markers

The computation which follows corresponds to the experiment where one uses random markers, which are not reduced to points. We suppose that Z_x is a marker implanted at a random position x. For the sake of simplicity we suppose that Z is the same marker everywhere, and its implementation is random. It is possible to imagine and compute the probabilities using random markers (for instance disks with random radii, segments with random or regionalized length and orientation etc.). Recall that the structuring element Z_x hits a set X if its center x belongs to the dilation of X by $Z : x \in X \oplus Z$.

Taking the same notations as above : the edge e_{pq} will be cut for a random distribution of markers, if the 3 following events are verified:

— $A1 = \{\exists$ random marker Z hitting $B_p\} = \{\exists$ random point marker belonging to $B_p \oplus Z\}$

— $A2 = \{\exists$ random marker Z hitting $B_q\} = \{\exists$ random point marker belonging to $B_q \oplus Z\}$

— $A3 = \{\nexists$ random marker Z hitting B_p and $B_q\} = \{\nexists$ random point marker belonging to $(B_p \oplus Z) \cap (B_q \oplus Z)$.

The balls B_p and B_q before and after dilation by an horizontal segment, and the intersection $(B_p \oplus Z) \cap (B_q \oplus Z)$ of both dilated sets are illustrated in figure 3.19.

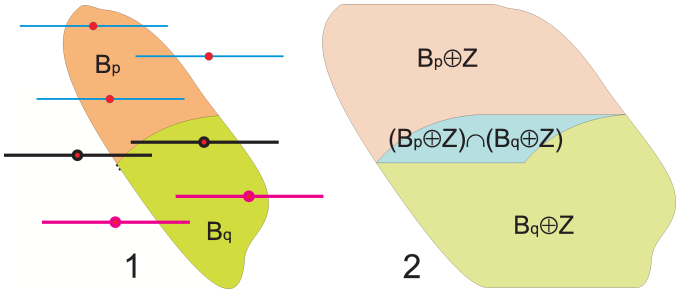


Figure 3.19 – 1: The two balls B_p and B_q and a number of structuring elements hitting the balls. The black ones hit both balls.
2: The dilated balls $B_q \oplus Z$ and $B_p \oplus Z$ and their intersection $(B_q \oplus Z) \cap (B_p \oplus Z)$ in cyan colour.

We have to compute $P(A1 \text{ and } A2 \text{ and } A3) = P(A1 \text{ and } A2 \mid A3) \times P(A3)$. If S_{pq} is the area of $(B_p \oplus Z) \cap (B_q \oplus Z)$, then $P(A3) = \left(1 - \frac{S_{pq}}{S}\right)^{\omega}$.

And $P(A1 \text{ and } A2 \mid A3) = 1 - P(\text{not } A1 \text{ or not } A2 \mid A3) = 1 - P(\text{not } A1 \mid A3) - P(\text{not } A2 \mid A3) + P(\text{not } A1 \text{ and not } A2 \mid A3)$.

The conditional probability $P(. \mid A3)$ means that all punctual germs have been distributed outside $(B_p \oplus Z) \cap (B_q \oplus Z)$, that is in an area $S - S_{pq}$. And the event $(\text{not } A1 \mid A3)$ means that there is no germ falling in $B_p \oplus Z$, knowing that there is also no germ falling in $(B_p \oplus Z) \cap (B_q \oplus Z)$, i.e. there is no germ falling in $(B_p \oplus Z) \diagup (B_q \oplus Z)$, domain with an area $S_{p \diagup q}$. Thus the probability is equal to $\left(1 - \frac{S_{p \diagup q}}{S - S_{pq}}\right)^{\omega}$. Exchanging the roles of p and q, we get $P(\text{not } A2 \mid A3) = \left(1 - \frac{S_{q \diagup p}}{S - S_{pq}}\right)^{\omega}$.

The event $\{\text{not } A1 \text{ and not } A2 \mid A3\}$ means that there is no punctual germ in $(B_p \oplus Z) \diagup (B_q \oplus Z)$ nor in $(B_q \oplus Z) \diagup (B_p \oplus Z)$. If $S_{p \triangle q}$ is the area of $(B_p \oplus Z) \diagup (B_q \oplus Z) \cup (B_q \oplus Z) \diagup (B_p \oplus Z)$, we obtain the probability $P(\text{not } A1 \text{ and not } A2 \mid A3) = \left(1 - \frac{S_{p \triangle q}}{S - S_{pq}}\right)^{\omega}$.

Putting everything together, we get the new weight:

$$\theta_{pq} = \left\{1 - \left(1 - \frac{S_{p \diagup q}}{S - S_{pq}}\right)^{\omega} - \left(1 - \frac{S_{q \diagup p}}{S - S_{pq}}\right)^{\omega} + \left(1 - \frac{S_{p \triangle q}}{S - S_{pq}}\right)^{\omega}\right\} \times \left(1 - \frac{S_{pq}}{S}\right)^{\omega}. \quad (3.8)$$

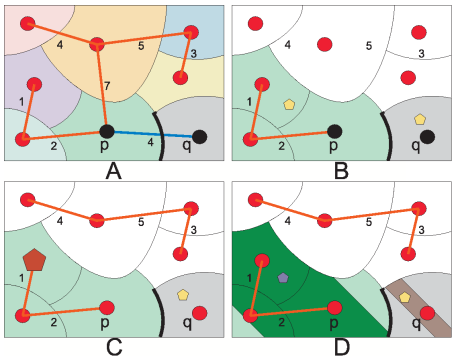
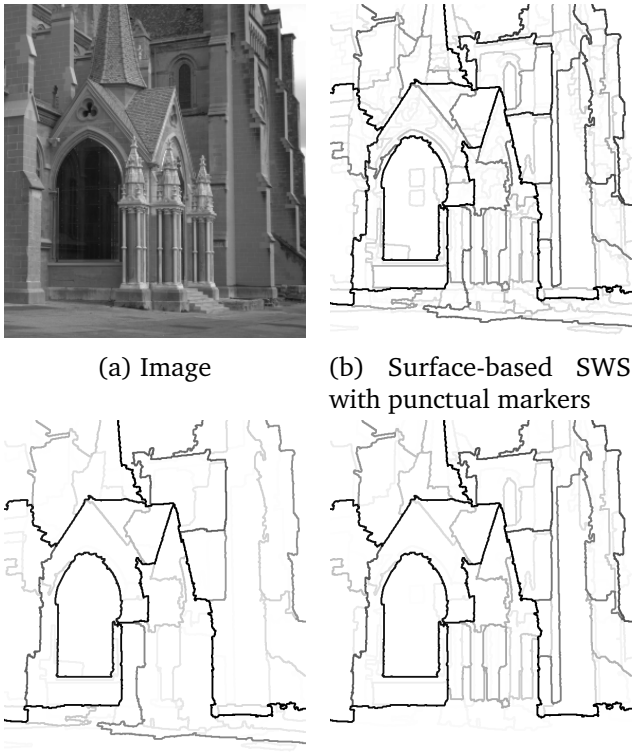


Figure 3.20 – A: Partition and its minimum spanning tree
B: Area stochastic watershed: a random marker in both colored regions B_p and B_q
C: Area oriented absorption stochastic watershed: a fixed marker in the largest region B_p and a random marker in the smaller region B_q
D: Area stochastic watershed with transformed domains: a random marker in the regions obtained by a linear opening of B_p and B_q.



(a) Image

(b) Surface-based SWS with punctual markers

(c) Surface-based SWS with horizontal markers of size 15

(d) Surface-based SWS with vertical markers of size 15

Figure 3.21 – Example of a surface-based SWS with return to the image during the construction process to test the shapes of emerging regions. (c): we can notice that when using horizontal markers, the thin vertical structures we can see in (b) are filtered out. (d): similarly, when using vertical markers, the thin horizontal structures are filtered out.

An example is provided in figure 3.21, where we compare on the same image the saliency maps of surface-based SWS hierarchies when drawing punctuals, horizontal and vertical markers. Similar results are presented for Poisson lines and Poisson flats markers in [Jeulin (2016)].

Computing new hierarchies thanks to the dendrogram structure

As we stated it in sections 2.3.2 and 3.2, we can fully represent a hierarchical clustering using a dendrogram structure. Such a structure, described in details in sections 2.2.1 and 2.3.2, is especially useful regarding implementation issues. This is why we took advantage of this representation to implement and work with hierarchies.

Algorithm 3.1: Area-based and volume-based stochastic watershed hierarchical clustering.

Data: An edge and node-weighted graph $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta, \eta_{\mathsf{V}})$ defined over a metric space (S, d) such that $|\mathsf{V}| = |S|$, and $\forall (i,j) \in \{1,...,n\}^2, \eta_{i,j} = d(\mathsf{v}_i, \mathsf{v}_j)$, and $\forall k \in \{1,...,n\}, \eta_k^{\mathsf{V}} = \mathtt{A}(\mathsf{R}_k)$ the area of the region that the node represents;
A number N of markers;

Result: A stochastic watershed hierarchy of the input graph in the form of new edges weights $\tilde{\eta}$.

1 Compute a minimum spanning tree $\mathcal{MST} = (\mathsf{V}, \mathsf{E}_{MST}, \eta)$ of $\mathcal{G}$;

2 Get the dendrogram structure associated with the initial weights ;

3 Compute the areas of the regions spanned by each internal node of the dendrogram in a bottom-up fashion, thus getting a new area for each region ;

4 Sort edges of the $\mathcal{MST}$ by increasing weights and visit all edges in this order ;

5 **while** *all edges have not been processed* **do**

6 Given the current edge e_{st} with a weight λ compute:

— the balls $B_s(s, \lambda)$ and $B_t(t, \lambda)$

— the diameter $diam_{B_s}$ (resp. $diam_{B_t}$) as the highest weight taken by the function $\tilde{\eta}$ in B_s (resp. B_t).

We write $\mathtt{A}_s$ (resp. $\mathtt{A}_t$) the area of the region spanned by all nodes below the node s (resp. the node t) in the dendrogram.

Then for the area-based SWS:

$\tilde{\eta}_{st} = 1 - (1 - \frac{\mathtt{A}_s}{S})^N - (1 - \frac{\mathtt{A}_t}{S})^N + (1 - \frac{\mathtt{A}_s + \mathtt{A}_t}{S})^N$.

And for the volume-based SWS:

$\tilde{\eta}_{st} = 1 - (1 - \frac{\eta_{st}\mathtt{A}_s}{V})^N - (1 - \frac{\eta_{st}\mathtt{A}_t}{V})^N + (1 - \frac{\eta_{st}(\mathtt{A}_s + \mathtt{A}_t)}{V})^N$.

7 **end while**

The computation in a bottom-up fashion of the areas of the regions spanned by the internal nodes of a dendrogram is given in figure 3.22.

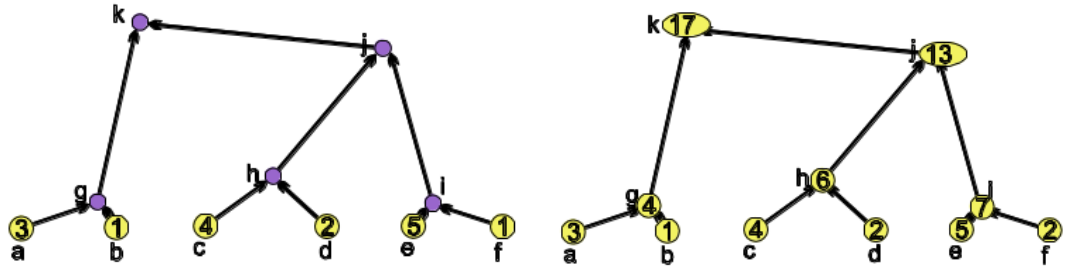


Figure 3.22 – Computation of the surfaces of the regions spanned by the internal nodes of dendrogram in a bottom-up approach. It requires one pass through the dendrogram.

The obtaining of the hierarchy for prioritized markers is also illustrated in figure 3.23.

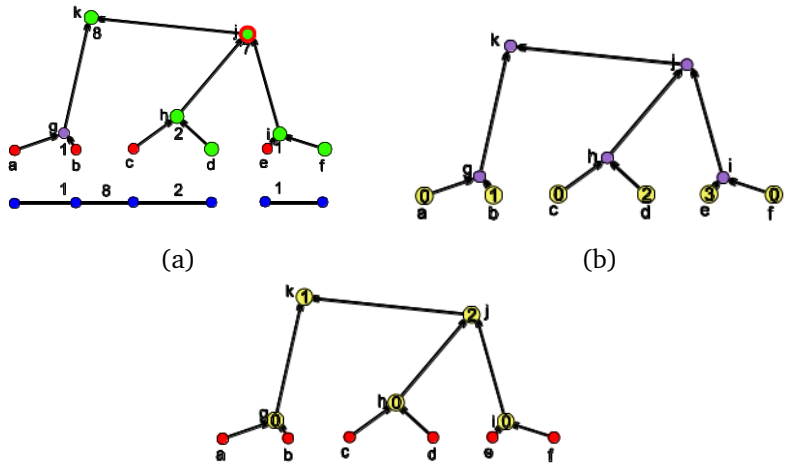


Figure 3.23 – Indexed markers-based hierarchical clustering using dendrograms. (a) To each leave of the dendrogram we attribute an index corresponding to the degree of priority of the marker; (b) To each node we attribute the maximum index of its left and right sons ; (c) Then we attribute to each node the minimum index of its sons indexes. The clustering of index λ is then obtained by merging all regions separated by an edge with an index inferior or equal to λ.

Influence of the number of markers

One can wonder what is the impact of the numbers of markers that we spread. We illustrate here the $\mathcal{UCM}$ for different numbers of markers.

As was expected, the higher the number of spread markers, the higher the probability for each contour to appear when doing marker-based segmentation using them. However, the order of disappearance of the contours in the hierarchy remains the same. Since the saliencies values do not seem very important by themselves, we privilege a number of drawings that allow us to obtain $\mathcal{UCM}$ that can be well interpreted by simple observation, that is to say a number of markers that allows for a good dispersion of saliency values. In practice, we chose to draw 80 markers, for images with a resolution of around 1000×1000 pixels and approximately 1000 regions.

In a complementary way, we can also represent the hierarchy by an alternative saliency map that we call “Persistence Contour Map” (PCM) in which each contour takes as value the level of stratification for which it disappears in the hierarchy, independently from its saliency value. We can see it as a kind of normalized $\mathcal{UCM}$ representing an ultrametric normalized relatively to the number of regions (see definition 2.31).

3.3.7 Energetic Approach: Binary-Scale Climbing Hierarchy

The Binary-Scale Climbing Hierarchy (BSC hierarchy)

In the literature, many simplification or segmentation methods consist in the optimization of an energy functional. Such an approach can also be applied to a hierarchical tree in order

to select its meaningful lines. This has been investigated notably in [Cao et al. (2005); Pardo (2002)].

Such approaches can be analyzed within the scale-space theory, which constitutes a fully coherent theory of multi-scale low-level image processing. It is inherently linked with hierarchical image segmentation, as it is based upon the same fundamental principal: the *causality principle*, which formalizes the common-sensical idea that details can only disappear by zooming out. The idea behind the use of energetic approaches for segmentation is to view a segmentation as a more or less simple modelization of the image. Following the Occam's Razor principle, the idea is then, between two models that equally fit the data, to favor the simpler. Its dual formulation states that between two equally complex models, the closer to the data of the two should be preferred. This can be translated into an optimization problem and thus into the minimization of its associated Lagrangian. Under large assumptions, this can be reduced to the optimization of an energetic expression of the form $D + \lambda C$, with D a "goodness-of-fit" term and C a "regularization" term. λ controls the precision of the approximation and thus can be thought of as a scale parameter.

A classical energetic functional that we can use is the Mumford-Shah functional, which is as stated above composed of two main terms: an image-fit term and a regularization term.

1. If f is the image and a partition segmenting the image, a piecewise constant model is adjusted to the image: each tile R_i of the partition π gets a grey-tone value equal to the mean value μ_i of f within this tile. The quality of the fit is then measured by the sum of variances of the image in all tiles of the partition:

$$D(\pi) = \sum_{R_i \in \pi} \sum_{k \in R_i} (f_k - \mu_{R_i})^2 \tag{3.9}$$

2. The complexity of the partition is measured by the length of the contours $C(\pi) = \partial_\pi$.

The problem is then to minimize the sum of variances under the constraint that the total length of the contours is below some limit :

$$\text{Find } \pi^* = \arg\min_{\pi} \Big(\sum_{R_i \in \pi} \sum_{k \in R_i} (f_k - \mu_{R_i})^2 + \lambda * \partial_\pi \Big) \tag{3.10}$$

A popular method to operate Mumford-Shah functional optimization is the seminal work of [Mumford et al. (1989)]. Curve evolution methods are usually used to solve such a problem, as they are effective and present theoretical foundations. However, they are computationally expensive, which makes them hard to use in an automated segmentation framework for example.

Instead of trying to optimize such a functional for all possible segmentations, some authors propose to minimize a similar functional subordinated to a given input hierarchical segmentation. This way, they find an optimal hierarchical segmentation in the sense of energy minimization, as in [Guigues et al. (2006); Kiran et al. (2014b); Xu, Géraud, et al. (2016)]. We take inspiration in the work of [Guigues et al. (2006)] to obtain, for each input hierarchical segmentation, a hierarchy that has been regularized as a trade-off between boundary length and image adherence.

In this work, the authors minimize the MS energy within a restricted number of solutions to create a hierarchy transcribing the successive values of the parameter λ for which the regions fuse, which makes the problem solvable by dynamic programming. We adapt this approach by performing this computation subordinated to the regions proposed by an input hierarchy. In this regard, our proposal can be linked with the work in [Xu, Géraud, et al. (2016)], in which the authors have a similar approach but use the tree of shapes instead of a hierarchy of segmentations such as ours.

In our case, the dynamic programming consists in sweeping accross the hierarchy, from fine to coarse and consider each tile. A region of the hierarchy is then retained for the $\lambda - optimal$ partition if the variance of this region is higher than the sum of the energies of its children. To each λ corresponds a partition and for increasing values of λ, the partitions get coarser. This means that one gets a new hierarchy with this process. Like that, one produces a new hierarchy whose stratification levels are equal to the scale parameter λ. This hierarchy is produced by non-horizontal cuts accross the initial hierarchy. It may then be used as any other hierarchy.

Description of the model

In our case we consider the same "energy" $D(\pi) + \lambda C(\pi)$ depending upon a scale parameter λ but we do consider a more restricted family of regions which will be merged or not. If e_{pq} is an edge linking the nodes p and q with a weight ρ ; we consider the balls $B_p = Ball(p; \rho)$ and $B_q = Ball(q; \rho)$. The length of their common boundary is l_{pq}. $D(B_p)$ and $D(B_q)$ are the variances of the image within the balls B_p and B_q : We will merge both balls if the variance of the union of both balls is lower than $D(B_p) + D(B_q) + \lambda l_{pq}$.

Efficient Computation

The advantage of this formulation is that it will be possible to construct in one pass through the edges of the minimum spanning tree the associated hierarchy, by assigning new weights to the MST generating a new ultrametric distance. In the more complex case of the Mumford Shah model, one has to construct a partition for each value of λ.

Algorithm to construct the hierarchy The algorithm visits all edges of the MST in the order of increasing weights. Repeat until all edges are processed:

— e_{pq} the current edge to process with a weight ρ

— we assign to each edge the weight λ^* for which it is equivalent to merge or to keep both balls separated

— if the balls merge : $Energy(B_p \cup B_q) = D(B_p \cup B_q)$; if the balls remain separated : $Energy(B_p \cup B_q) = D(B_p) + D(B_q) + \lambda l_{pq}$

— For a higher value $\mu > \lambda^*$; we have $D(B_p) + D(B_q) + \mu l_{pq} > D(B_p) + D(B_q) + \lambda^* l_{pq}$, showing that both balls should be merged. For a lower value, they should keep separated.

— Let us compute this value λ^* which constitutes the point where merging or keeping them separated are equivalent choices: $\lambda^* = \frac{D(B_p \cup B_q) - D(B_p) - D(B_q)}{2l_{pq}}$

Algorithmic trick By keeping in memory for each leaf of the dendrogram (i.e. each node of the finest partition) the size of its contours with each of its neighbors, its moments of orders 1,2 ($\sum_{k \in R_i} f_k$, $\sum_{k \in R_i} f_k^2$), and the parenthood relationships between nodes, one can fastly compute the fitting-term and the contour term of each level of a hierarchy of partitions represented as a dendrogram.

Proof.

$$D(\pi) = \sum_{R_i \in \pi} \sum_{k \in R_i} (f_k - \mu_{R_i})^2 \tag{3.11}$$

$$= \sum_{R_i \in \pi} \sum_{k \in R_i} (f_k - \frac{m_1^i}{m_0^i})^2 \tag{3.12}$$

$$= \sum_{R_i \in \pi} [\sum_{k \in R_i} f_k^2 - 2\frac{m_1^i}{m_0^i} \sum_{k \in R_i} f_k^2 + \frac{{m_1^i}^2}{{m_0^i}^2} \sum_{k \in R_i} 1] \tag{3.13}$$

$$= \sum_{R_i \in \pi} (m_2^i - \frac{{m_1^i}^2}{m_0^i}) \tag{3.14}$$

So let us say we want to have the new fitting-term $D(R_p \cup R_q)$ providing the moments of the subregions R_p and R_q :

$$D(R_p \cup R_q) = \sum_{k \in R_p \cup R_q} (f_k - \mu)^2 \tag{3.15}$$

$$= \sum_{k \in R_p \cup R_q} [(m_2^p + m_2^q) - \frac{(m_1^p + m_1^q)^2}{m_0^p + m_0^q}] \tag{3.16}$$

□

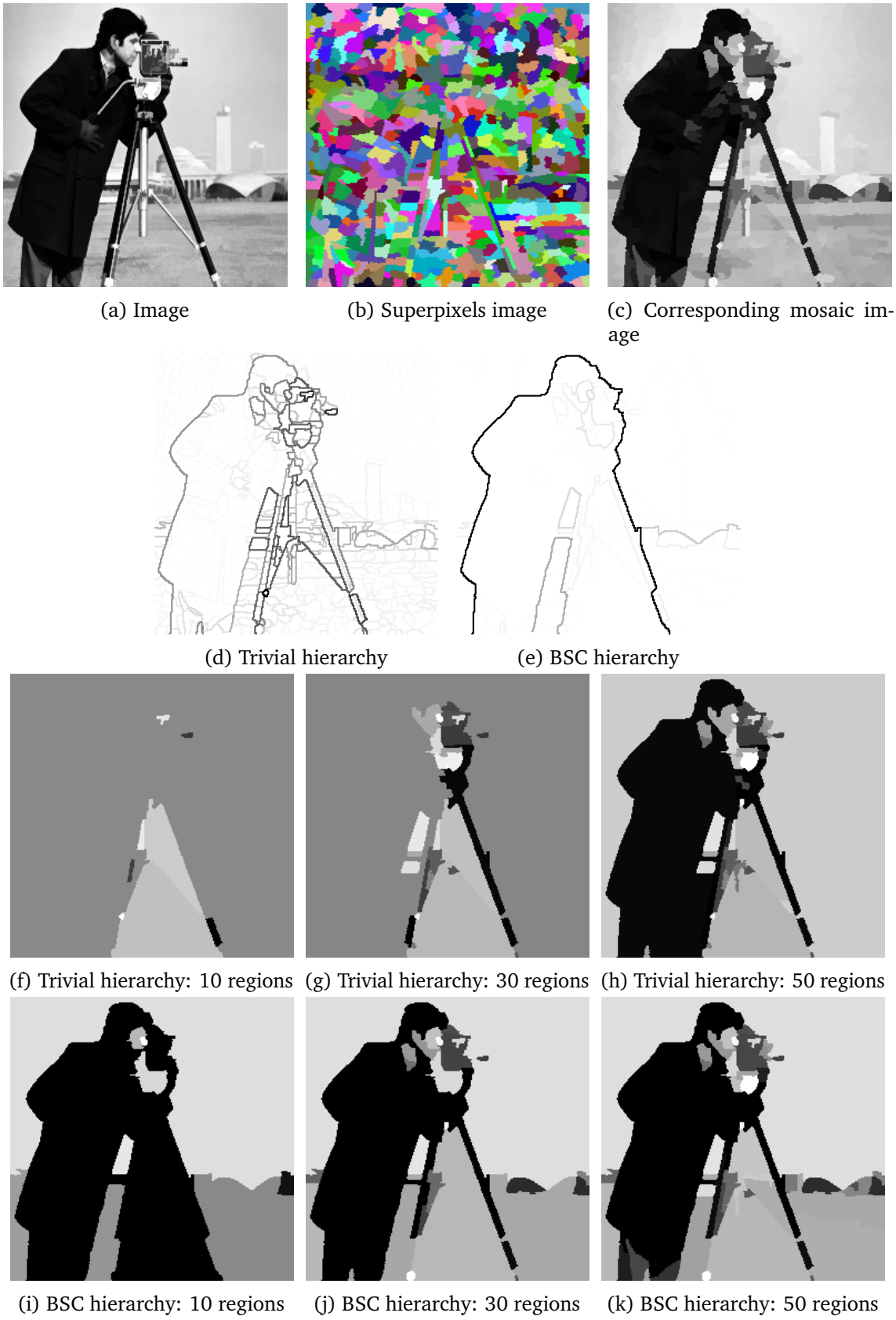
We will also present in section 3.6.3 how to compute quickly the contours length of each hierarchical level using the dendrogram structure and dictionaries.

Qualitative Analysis

This new hierarchy is interesting because it provides us with a way to balance the output of a hierarchy with respect to a trade-off between image adherence and simplicity. Just as in the expression of the Mumford-Shah energy, the parameter λ can be seen as playing the role of a "potentiometer". For each λ (corresponding to a cut level of the new hierarchy), we have the optimal segmentation with respect to the input hierarchy regions proposals and to the energetic expression to optimize.

We present some visualizations of the effect of this hierarchy in figures 3.24 and 3.25. As expected, the regions proposed by this hierarchy tend to capture large homogeneous regions first, as it favors the simpler model that can explain (i.e. correspond as much as possible to) the

departure image. Hence, applying this hierarchy can be seen as a useful first step for low-level hierarchical segmentation, in order to avoid missing any important large region.



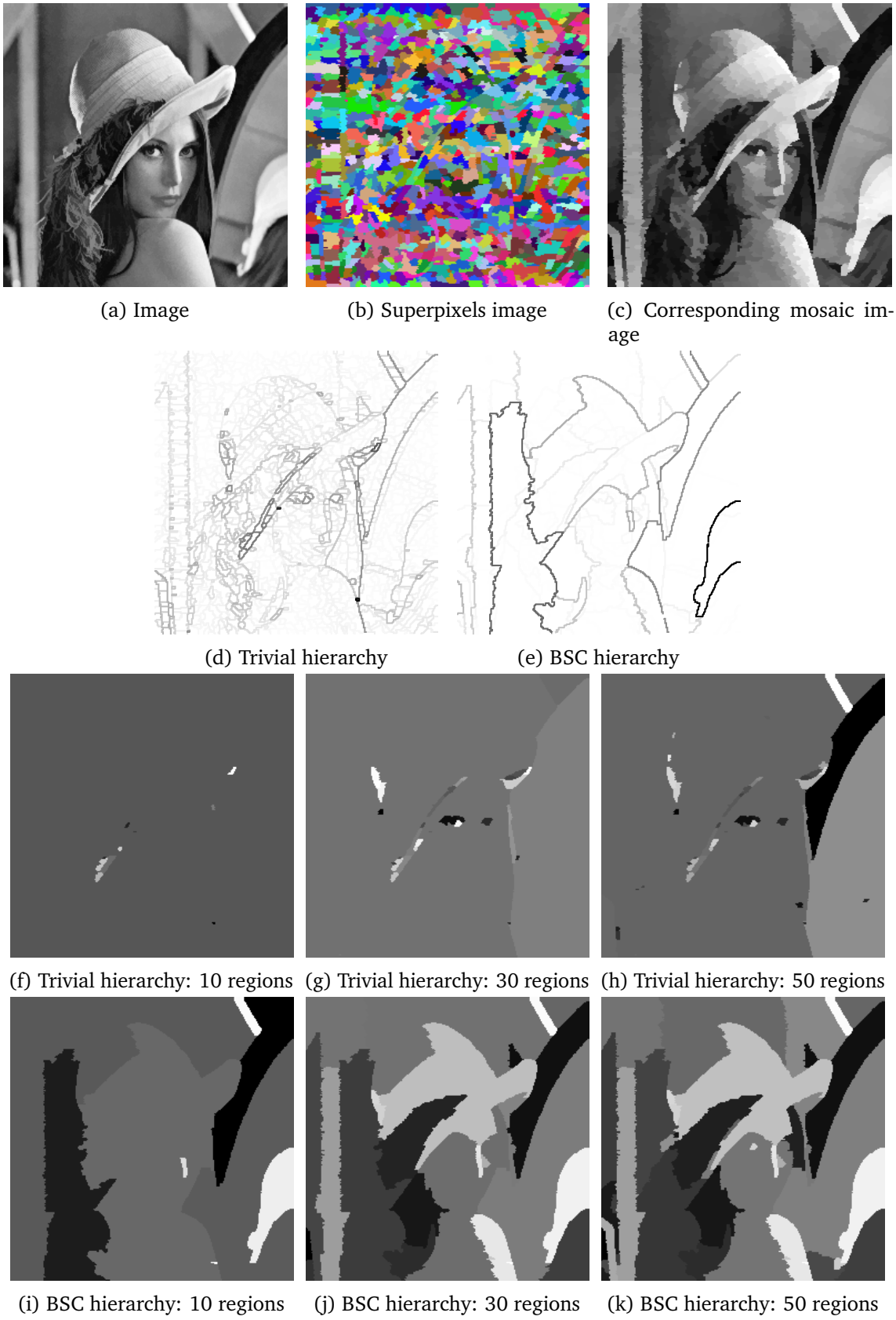
(a) Image (b) Superpixels image (c) Corresponding mosaic image

(d) Trivial hierarchy (e) BSC hierarchy

(f) Trivial hierarchy: 10 regions (g) Trivial hierarchy: 30 regions (h) Trivial hierarchy: 50 regions

(i) BSC hierarchy: 10 regions (j) BSC hierarchy: 30 regions (k) BSC hierarchy: 50 regions

Figure 3.24 – Illustration of the effect of the BSC hierarchy. It allows to balance the output of any hierarchy (here the trivial hierarchy) to obtain a hierarchy presenting a trade-off between adherence to the image and model simplicity.



(a) Image (b) Superpixels image (c) Corresponding mosaic image

(d) Trivial hierarchy (e) BSC hierarchy

(f) Trivial hierarchy: 10 regions (g) Trivial hierarchy: 30 regions (h) Trivial hierarchy: 50 regions

(i) BSC hierarchy: 10 regions (j) BSC hierarchy: 30 regions (k) BSC hierarchy: 50 regions

Figure 3.25 – Illustration of the effect of the BSC hierarchy. It allows to balance the output of any hierarchy (here the trivial hierarchy) to obtain a hierarchy presenting a trade-off between adherence to the image and model simplicity.

3.3.8 Other classical morphological hierarchies

Waterfall hierarchy

This hierarchy has first been described in the context of a topographic surface flooding [Beucher (1990)].

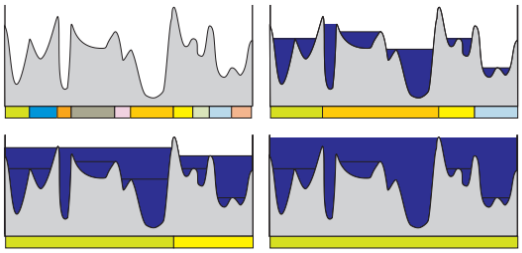


Figure 3.26 – Four levels of the waterfall hierarchy. Each topographic surface has been flooded up to the lowest pass point of the catchment basins associated to the preceding surface.

In a graph-based approach, let us suppose that the nodes correspond to the catchment basins a topographic surface. We can then build a minimum spanning forest in which each tree takes root in one of these minima. All of the edges internal to those trees take as value 1 (the first level of the waterfall). The edges of each of these trees are then contracted until reduced to a node. We then have a new graph on which we can proceed as previously, and edges internal to the trees take the value 2. We keep going until all edges have been valuated and contracted.

To go further, it is possible to produce this waterfall hierarchy in one pass through the edges of the MST [Meyer (2015)]. We assign to each edge e_{st} of the MST, with an initial weight η_{st}, a new weight $\tilde{\eta}_{st}$ expressing its level in the waterfall hierarchy.

Figure 3.27 presents the waterfall hierarchy: the initial image, the waterfall saliency of the contours, followed by 4 levels of the waterfall hierarchy.

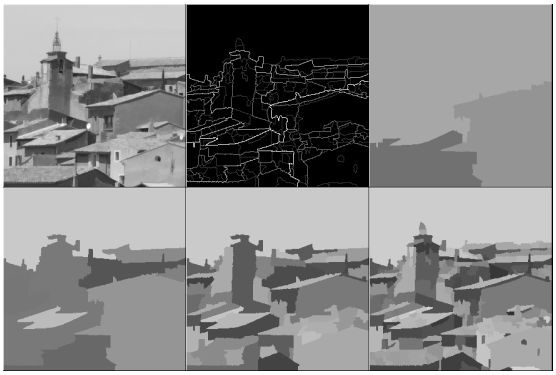


Figure 3.27 – The waterfall hierarchy. The saliency of the contours followed by 4 levels of the waterfall hierarchy.

Algorithm 3.2: Waterfall Hierarchical Clustering

Data: A graph $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta)$ defined over a metric space (S, d) such that $|\mathsf{V}| = |S|$, and with for $\forall (i,j) \in \{1, ..., n\}^2, \eta_{i,j} = d(\mathsf{v}_i, \mathsf{v}_j)$;

Result: a waterfall hierarchical clustering of the input graph in the form of new edges weights $\tilde{\eta}$.

1 Compute a minimum spanning tree $\mathcal{MST} = (\mathsf{V}, \mathsf{E}_{MST}, \eta)$ of $\mathcal{G}$;

2 Sort edges of the $\mathcal{MST}$ by increasing weights and visit all edges in this order ;

3 **while** *all edges have not been processed* **do**

4 Given the current edge e_{st} with a weight λ compute:

— the balls $B_s(s, \lambda)$ and $B_t(t, \lambda)$

— the diameter $diam_{B_s}$ (resp. $diam_{B_t}$) as the highest weight taken by the function $\tilde{\eta}$ in B_s (resp. B_t).

Then: $\tilde{\eta}_{st} = 1 + min(diam_{B_s}, diam_{B_t})$.

5 **end while**

Hierarchy of Levelings

An interesting family of hierarchies are the hierarchies of levelings, which rely on the levelings operators that we reintroduce hereafter [Meyer (1998); Matheron (1997); Serra et al. (n.d.)]. This type of hierarchies is not obtained using a graph-based framework but directly through successive appliances of levelings operators.

Quasi-flat zones An image is a grid of pixels. Segmenting means regrouping similar pixels in the same class. Hence the simplest segmentation technique consists in regrouping all pixels having the same grey tone or the same color. Such a group of pixels with the same color is called flat zone. The relation "to have the same color" is an equivalence relation ; its equivalence classes constitute the simplest segmentation technique.

Definition 3.1 (Flat zones)**.** Two points x, y belong to the same *flat-zone* of a function f if and only if there exists a $n-$tuple of points $(p_1, p_2, \ldots, p_n)$ such that $p_1 = x$, $p_n = y$ and for all i, (p_i,p_{i+1}) are neighbours and have the same value through f: $f(p_i) = f(p_{i+1})$.

Definition 3.2 (Quasi-flat zones (or lambda-flat zones))**.** Two points x, y belong to the same *quasi-flat-zone* of a function f if and only if there exists a $n-$tuple of points $(p_1, p_2, \ldots, p_n)$ such that $p_1 = x$, $p_n = y$ and for all i, (p_i,p_{i+1}) are neighbours and have their value through f verify a symmetrical relation.
Examples of symmetrical relations:

— $|f(p_i) - f(p_{i+1})| < \lambda$, leading to slope levelings for $\lambda > 0$.

— $f(p_{i+1}) \geq (\gamma \circ f)(p_i)$ and $f(p_i) \geq (\gamma \circ f)(p_{i+1})$.

— $f(p_{i+1}) \leq (\phi \circ f)(p_i)$ and $f(p_i) \leq (\phi \circ f)(p_{i+1})$.

— or any logical combination of the preceding ones.

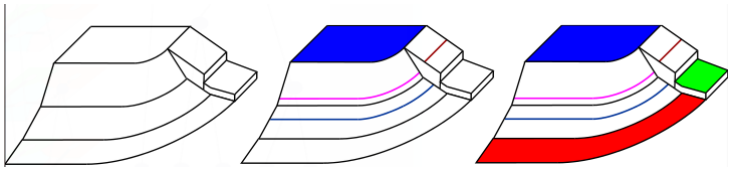


Figure 3.28 – Flat and λ-flat zones. The levelings depend upon the definition of the flat zones. For the same topographic surface A, two types of flat zones are produced: (i) In B strict flat zones : the blue area + level lines are detected, (ii) In C three lambda flat zones (slope ≤ 1) are detected.

From the preservation of contours to the levelings A good filter Φ should transform an image f into an image g with less details and simpler to segment. However, the contours of any segmentation produced on g should exactly match the contours of the same objects as seen in f. In other words, there should be no displacement of the contours when one goes from f to g. Crossing a contour between two pixels p and q in the simplified image g means finding a large transition between two neighboring values g_p and g_q. A down transition between the neighboring pixels p and q will be written $g_p \succ g_q$. It may be any relation defined on the neighborhoods of g_p and g_q verifying $g_p \succ g_q \Rightarrow g_p > g_q$. In the present paper we will focus on the relation $g_p \succ g_q \Leftrightarrow g_p > g_q + \lambda$. In practice, many more types may be defined. To any definition of an up and down transition, we easily associate quasi flat zones. We first define the negation of the up relation : $Not\,[g_p \succ g_q] \Leftrightarrow g_p \preccurlyeq g_q$. The relation $\left| \begin{array}{l} g_p \preccurlyeq g_q \\ g_q \preccurlyeq g_p \end{array} \right| \Leftrightarrow g_p \approx g_q$ is symmetrical and pixels verifying $g_p \approx g_q$ belong to the same *quasi-flat-zone*, hence it permits to construct quasi-flat zones as explained in the definition above. In the case where $g_p \succ g_q$ means $g_p > g_q + \lambda$, $g_p \approx g_q$ will mean $|g_p - g_q| \leq \lambda$.

Suppose now that $g_p \succ g_q$ As we require that no contour be displaced when going from f to g, a similar contour (by similar we mean that to an up transition should correspond an up transition) should exist between the pixels p and q for the image f. This basic requirement is at the heart of the definition of levelings:

Definition 3.3 (Leveling)**.** A function g is a *leveling* of a function f if and only if: for any couple of neighboring pixels (p, q): $g_p > g_q + \lambda \Rightarrow f_p \geq g_p$ and $g_q \geq f_q$. When $\lambda = 0$, we talk of *flat leveling*, and when $\lambda > 0$ of *lambda-leveling*.

The definition clearly shows that to any transition $g_p > g_q + \lambda$ corresponds an even bigger transition, since the interval $[g_q, g_p]$ is included in the interval $[f_q, f_p]$. The definition is illustrated for $\lambda = 0$ in the figure 3.29, on two couples of pixels for which $f_s = g_s < g_t < f_t$ and $f_q < g_q < g_p = f_p$.

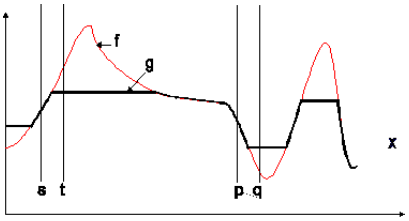


Figure 3.29 – Illustration of the definitions of levelings: $g_p > g_q \Rightarrow f_p \geq g_p$ and $g_q \geq f_q$

Definition 3.4 (Rho-leveling)**.** A rho-leveling is a generalization of the classical leveling. A function g is a rho-leveling of a function f if and only if:

$$f \vee \delta\gamma g \leq g \leq f \wedge \epsilon\varphi g \tag{3.17}$$

Where ϵ is the adjunct erosion of δ, γ and φ are the associated openings and closings. The rho-leveling introduces a viscosity that allows to avoid the reconstruction of small details through narrow paths.

Levelings are connected operators The property of levelings which permits to derive hierarchies from them, is the fact that the levelings merge or extend quasi-flat zones.

A function g is a leveling of a function f if and only if for any couple of neighboring pixels (p, q): $g_p > g_q + \lambda \Rightarrow f_q \leq g_q$ and $g_p \leq f_p$

It follows $g_p > g_q + \lambda \Rightarrow f_q + \lambda \leq g_q + \lambda < g_p \leq f_p \Rightarrow f_q + \lambda < f_p$.

The contraposition of this implications states: $f_q + \lambda \geq f_p \Rightarrow g_p \leq g_q + \lambda$

Inverting the roles of p and q : $f_p + \lambda \geq f_q \Rightarrow g_q \leq g_p + \lambda$

Regrouping both expressions : $f_q + \lambda \geq f_p \geq f_q - \lambda \Rightarrow g_q + \lambda \geq g_p \geq g_q - \lambda$, which shows that if p and q belong to the same quasi-flat zone for the function f, expressed by the fact that $f_q + \lambda \geq f_p \geq f_q - \lambda$, then p and q also belong to a quasi-flat zone of g. In other words, the leveling extends the quasi flat zones of f. This extension can only occur from merging pre-existing flat zones of f, showing that the partition of the quasi-flat zones of f is finer that the partition of the quasi-flat zones of g.

Property 3.5. "To be a leveling" is a preorder relation. In particular if g is a leveling of f and h is a leveling of g, then h is a leveling of f.

A sequence of levelings creates a hierarchy of quasi-flat zones

Property 3.6. The levelings extend the flat-zones, the lambda-levelings extend the lambda-flat zones. This is illustrated in figure 3.30.

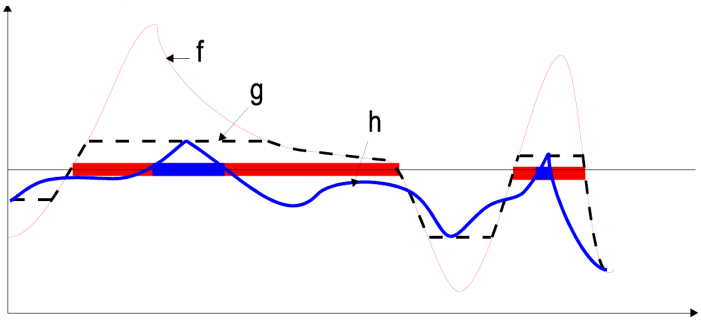


Figure 3.30 – The levelings extend the flat-zones, the lambda-levelings extend the lambda-flat zones.

We have seen that "to be a leveling" is a preorder relation. In particular if g is a leveling of f and h is a leveling of g, then h is a leveling of f. This property permits to derive from an initial function f_0 a sequence of increasing levelings $(f_0, f_1, ...f_k)$ where each function f_i is a leveling of all functions f_j for $j < i$. As stated above the partition of quasi-flat zones of the functions f_i is coarser then the partition of quasi-flat zones of f_j for $j <$, the coarser partition resulting from the merging of regions of the finer partition. We have thus established that the quasi-flat zones of a sequence of levelings forms a hierarchy.

A large set of possible hierarchies exist can be conceived in this regard, as we can choose:

— The hierarchy building type: linear or as a cascade (see figure 3.31).

— The type of levelings considered: flat, lambda, rho, lambda-rho.

— The type of markers consider: alternate sequential filters (ASF), gaussian filters etc.

— The structuring element form (hexagonal and thus isotropic, horizontal, vertical) and size.

In particular, we can note the choice of markers images can be directly tailored to adapt to the content and the task at hand.

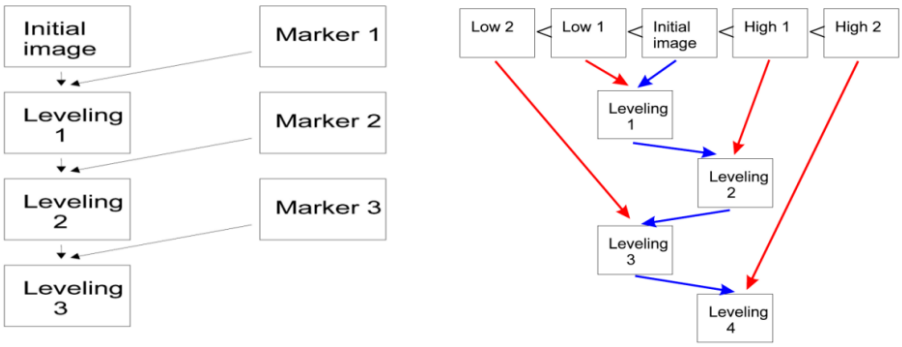


Figure 3.31

On the contrary to all hierarchies presented otherwise in this section, hierarchies of levelings have not been obtained using a graph-based framework, although they could have been [Meyer and Lerallut (2007); Meyer (2013c); Meyer (2013b)]. Furthermore, one can easily obtain a

saliency matrix and a saliency map fully describing such a hierarchy for a given image. We are thus able to compare the effect of this type of hierarchies with other ones obtained in a different fashion as we have unified representations of them. In particular, having saliency matrices (corresponding to ultrametric values) allows for a computation of distances between hierarchical clusterings such as the Gromov-Hausdorff distance.

As we have seen, the designing of levelings hierarchies presents several degrees of freedom. They consist in a progressive enlargement of flat zones of the images. By choosing the type of images markers or the shape of the structuring elements we can favor certain directions. One can also choose the type of leveling used to allow a less restrictive slope (λ-leveling) or an expansion through fine structures of the image (ρ-leveling). This is illustrated in figure 3.32.

In a way, the mode of operation of these hierarchies is the counterpart of the watershed hierarchies ones, in which we introduce prior constraints on the regions to highlight. Instead, they operate in a more direct way, by working directly on the image and benefiting from the structural property stating that a leveling can only enlarge flat zones (cf. property 3.6). Furthermore, whereas watershed hierarchies are based upon image gradients and naturally highlight contrasted objects, levelings hierarchies are based upon razings and floodings and thus highlight peaks and valleys of the image, that stand out from its background. There thus seems to be an intrinsic complementarity between those two approaches.

3.4 Comparison with State-of-the-art

In this section, we present the performances of the classical surface-based and volume-based SWS hierarchies in comparison with other state-of-the-art hierarchical segmentations algorithms over the BSD500 dataset [Arbelaez et al. (2011)]. The hierarchical segmentations algorithms are evaluated for segmentation tasks upon a variety of different images by comparison with the segmentations provided by human users considered to be the ground-truths.

3.4.1 Scores

Probabilistic Rand Index (PRI)

The *Rand Index* has initially been introduced for clusterings evaluation. It operates by comparing pairs of points of the compared clusters. The Rand Index between a cluster S and a ground-truth G is the sum of the number of pixels pairs that have the same labels in S and G, and of those which have different labels in the two segmentations, divided by the total number of pixels pairs. The *Probabilistic Rand Index* (PRI) is a variant introduced for the case when multiple ground truthes are available. If we consider a set of ground-truthes $(G_k)_k$, the PRI is given by:

$$PRI(S, G_k) = \frac{1}{T} \sum_{i<j} [c_{ij} p_{ij} + (1 - c_{ij})(1 - p_{ij})], \tag{3.18}$$

(a) Image. (b) Fine partition. (c) Associated mosaic image

(d) Flat leveling - 10 regions (e) Flat leveling - 30 regions (f) Flat leveling - 50 regions (g) Flat leveling - Saliency map

(h) λ-leveling - 10 regions (i) λ-leveling - 30 regions (j) λ-leveling - 50 regions (k) λ-leveling - Saliency map

(l) $\lambda - \rho$ leveling - 10 regions (m) $\lambda - \rho$ leveling - 30 regions (n) $\lambda - \rho$ leveling - 50 regions (o) $\lambda - \rho$ leveling - Saliency map

Figure 3.32 – Saliency maps and 10,30,50 regions images for different levelings hierarchies upon the same image and fine partition. These hierarchies have been built using the cascade building type with eroded and dilated images as markers.

Where c_{ij} is the event "the pixels i and j" have the same labels and p_{ij} its probability. T is the total number of pixels.

Variation of Information(VI)

The *Variation of Information* (VI) is also a metrics that has been introduced to compare clusterings. It measures the distance between two segmentations relatively to their average conditional entropies. It is given by:

$$VI(S, S') = H(S) + H(S') - 2I(S, S'), \tag{3.19}$$

where H and I respectively represent the entropies and mutual information between two data clusterings S and S'. In our case, the clusterings are the test and ground-truth segmentations.

Optimal Dataset Scale (ODS), Optimal Image Scale (OIS)

Let us consider a $\texttt{score}(S, GT)$ to attest of the quality of a segmentation S given a ground truth GT. `score` can correspond to the PRI or the VI presented above.

A hierarchical segmentation method applied on an image provides a hierarchy $(\mathcal{H}, \boldsymbol{\lambda})$, where the successive ultrametric levels $(\lambda_1, \ldots, \lambda_N)$ correspond to a series of nested segmentations $(S_1, \ldots, S_N)$. To properly evaluate it, one has to compute the $\texttt{score}(S_i, GT)$ for any level $\boldsymbol{\lambda}_i$ of the hierarchy of each image.

One can then either retain these best level λ^* on the overall dataset, and the corresponding score is the *Optimal Dataset Scale* (ODS), or retain the best level λ_i for each image and average the best individual scores for all images, which correspond to the *Optimal Image Scale*(OIS). By definition, the ODS is inferior or equal to the OIS.

3.4.2 Results

In table 3.1 are presented the performances of the surface-based and volume-based SWS hierarchies. We notice that their results are in the same range than state-of-the-art algorithms. As expected, the volume-based SWS hierarchy is more adapted than the surface-based SWS to segment multimedia images.

Table 3.1 – Segmentation results on BSD500 test set, with a comparison to the state-of-the-art competitors

	PRI(↑)		VI(↓)	
	ODS	OIS	ODS	OIS
SWS hierarchies				
`Volume-based SWS`	0.81	0.84	1.84	1.59
`Surface-based SWS`	0.75	0.79	2.12	1.87
α-tree	0.78	0.82	—	—
Tree based shape space [Xu, Carlinet, et al. (2016)]	0.80	0.83	—	—
Ncuts [Salembier and Garrido (2000)]	0.78	0.80	2.23	1.89
Felz-Hutt [Felzenszwalb et al. (2004)]	0.80	0.82	2.21	1.87
Mean Shift [Comaniciu et al. (2002)]	0.79	0.81	1.85	1.64
UCM [Arbelaez et al. (2011)]	0.83	0.86	1.69	1.48
ISCRA [Ren et al. (2013)]	0.82	0.85	1.60	1.42
PFE+mPb [Yu et al. (2015)]	0.84	0.86	1.61	1.43
PFE+MCG [Yu et al. (2015)]	0.84	0.87	1.56	1.36

Further evaluation within a framework better-suited to hierarchical segmentation could be conducted [Perret et al. (2018)].

3.5 Practical Considerations Regarding the Choices Made

As a reminder, here are the main steps to get a hierarchy of segmentations in the graphs-based hierarchical segmentation framework we use:

1. get a fine partition $\mathcal{FS}(\mathbf{I})$ out of the image $\mathbf{I}$
2. get a Region Adjacency Graph (RAG) $\mathcal{G}$ out of $\mathcal{FS}(\mathbf{I})$, considering a given dissimilarity between regions of the fine partition
3. compute the Minimum Spanning Tree $\mathcal{MST}(\mathcal{G})$ from $\mathcal{G}$
4. find the inconsistent edges on the $\mathcal{MST}(\mathcal{G})$ according to some criteria and cut them, resulting in a partition of the image. Cutting edges in a MST produces forests. The more edges are cuts, the smallest the trees in the resulting forest. To a series of cuts corresponds a decreasing sequence of forests, and at each step, the trees of one forest underly a partition. Thus we obtain a series of nested partitions, i.e. a hierarchy. In particular, a simple way of proceeding to get such a hierarchy is to cut edges by decreasing valuations. When cutting edges by decreasing order of consistency, one get a hierarchy of segmentations.

Different choices regarding the construction of the graph, namely concerning the construction of the fine segmentation and the choice of the dissimilarity, have an important influence on the output of this workflow. We hereby describe with more details the choices of methods and implementations we made for each of these steps.

3.5.1 Obtaining a satisfying fine partition

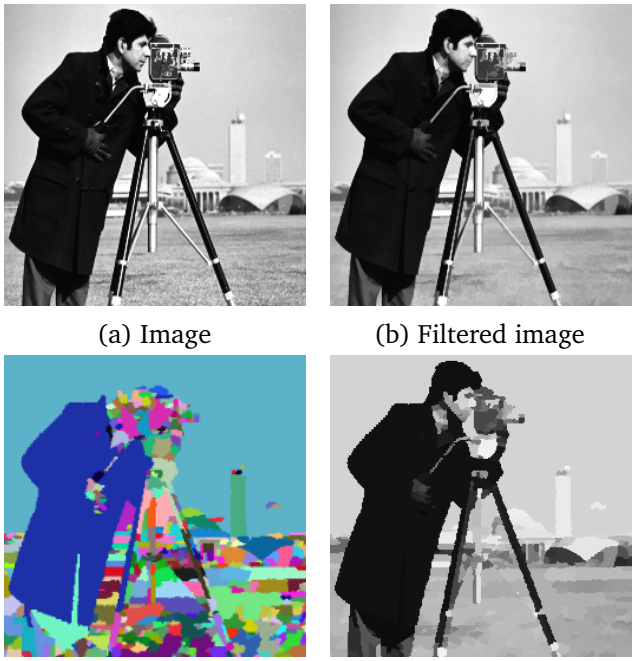
For the sake of computational efficiency, we chose to work with region adjacency graphs instead of pixel ones. This way, there are far less nodes in the graph than there are pixels in the image. This is why producing a fine segmentation of the image constitutes the first part of our framework. We want this fine segmentation to delineate most of the relevant contours based on perceptual cues. There are several adequate methods to obtain fine segmentations. The initial image can also be filtered out for furthers treatments to be more simple.

First approach

To do so, a first possibility consists in operating a watershed transformation on a gradient image of the image with its minima as flooding sources. In order to get a useful departure oversegmentation, one must first suppress the meaningless minima. We do so by first filtering the image, using an alternate sequential leveling of small size. Then we operate a h-reconstruction of the gradient image. The h-reconstruction of a function f is the highest flooding of f that is under the function $f + h$. Its effect is that some minima are filled, and the corresponding lakes are absorbed by their neighbors. Once the gradient image has been processed, we can compute a watershed transform on it. Finally, we aggregate the regions that are still too small at the end of the entire process. An example of the type of results obtained with such a process is provided in figure 3.33. Note that there exist some alternative to the h-reconstruction. For example, one can filter the image according to the regions sizes, by computing the highest flooding of the gradient image under the dilated gradient image.

Second approach

An other method, illustrated in figure 3.34, consists in making a heavier use of the concept of lambda-flat zones, as long as of levelings, introduced in section 5. Just as previously, we begin with a filtering of the image using an alternate sequential filter of small size (for example 2). Then we operate a lambda-rho leveling which presents the double interest of extending the flat zones, thus creating larger regions, while still being able to capture fine and elongate forms thanks to the introduction of a viscosity parameter. We usually choose the lambda and rho parameters to be small (typically 2 and 1 respectively). Labelling the lambda-flat zones (for the same lambda) of the result gives us a first labels image, with some regions generally being too small. So we suppress these undesired regions by setting them to zero and then operate a watershed transform on the gradient image using this labels image as a marker image. Just as before, we label the lambda-flat zones of the output, this time set to zero the zones that are narrower than a parameter (typically 2), and do a new watershed transform with this new image as marker.



(a) Image (b) Filtered image

(c) Fine partition obtained (d) Corresponding mosaic image

Figure 3.33 – Example of a fine partition obtained using : (i) Filtering of the image using alternate sequential levelings (ii) h-reconstruction of the watershed (iii) Removal of the too small regions

Third approach: superpixels

Both these methods, while they can provide suitable fine partitions, rely on a lot of parameters tuning to work properly. For example, in the first approach, depending on the initial image (its size, texture) the initial filtering or the choice of the height in the h-reconstruction can dramatically modify the output and possibly results in missing some potentially interesting contours. Yet at this stage, our goal is precisely to extract all of these potentially interesting contours, in order to structure them in following process. On one hand, this possibility of fine-tuning reveals the versatility and adaptibility of these approaches, since they can be adapted to very specific problems in order to get the best possible results on very specific tasks. But on the other hand, this makes them less pertinent when placed at the beginning of an automated segmentation process that we would like to be as adaptative as possible.

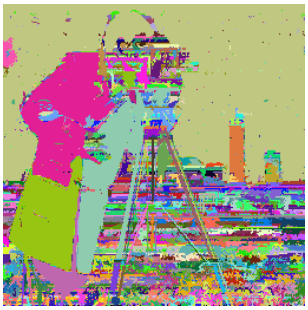
This is why we explored an other popular approach that consists in using superpixels methods to produce our initial fine partitions. Superpixels are regions resulting from a low-level segmentation in order to be used as primitives for further analysis such as detection, segmentation, and classification of objects. Superpixels aim to have the following properties : (i) *homogeneity*, i.e. pixels belonging to a given superpixel present similar colors or gray levels (ii) the superpixels constitute a partition of the image (iii) *adherence* to object boundaries: object boundaries should be included in superpixels object boundaries (iv) *regularity*: superpixels should form a regular pattern on the image.

Many approaches exist to compute superpixels, most of them based on graphs [Felzenszwalb

(a) Image.

(b) Filtered image using an ASF leveling of size 1, and a lambda-rho leveling with lambda=rho=1.

(c) Associated gradient image that we use for watershed transform.



(d) Lambda-labelling of the output of the first watershed transform.

(e) Removal of the regions with an area inferior to 15.

(f) Lambda-labelling of the output of the second watershed transform, using the labels image with small areas removed as marker image.

(g) Removal of the regions narrower than 2.

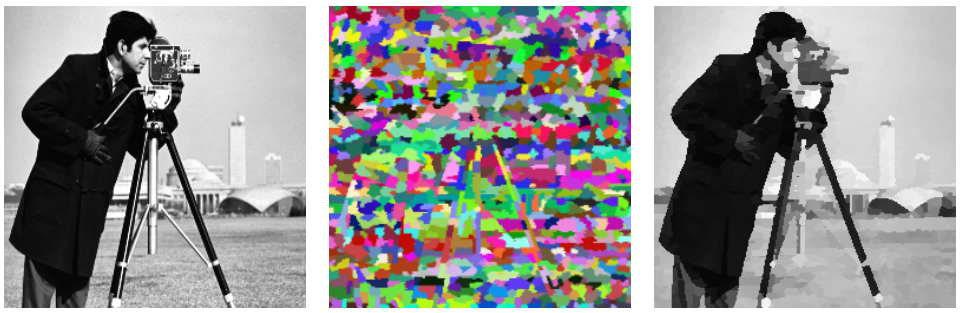
(h) Final labels image output by a watershed transform using previous labels image as marker.

(i) Corresponding mosaic image.

Figure 3.34 – Example of a fine partition obtained using : (i) Filtering of the image using ASF and lambda-rho leveling (ii) Removal of small and narrow regions to create markers images for watershed transform.

et al. (2004)], geometrical flows [Levinshtein et al. (2009)] or k-means [Achanta et al. (2012)]. Since these superpixels will be used in an already long framework, we focus on linear complexity methods such as those presented in [Achanta et al. (2012); Machairas et al. (2015)].

We privilege the waterpixels approach [Machairas et al. (2015)], which main idea is to operate a markers-based watershed transform on the image by choosing seeds on a regular grid of this image. While giving slightly better results than the SLIC superpixels, the waterpixels also present the advantage of being connected by definition, on the contrary to SLIC superpixels which require some ad-hoc postprocessing step to obtain this property. It is indeed an advantage since our framework departures from a connected fine partition of the image, which is indeed the basis for the construction of a regions adjacency graph. It also let us choose an approximate number of regions wanted per image. Experiments show that 1500 regions in the fine segmentation allow to capture most of the significant contours in the image. An example of the output given by such an approach can be found in figure 3.35.



(a) Image (b) Superpixels (c) Corresponding mosaic image

Figure 3.35 – Example of a fine partition obtained using waterpixels.

This way, we obtain over-segmentations of our images that do not miss (or only in rare cases) any potentially interesting contour, given that the gradient image we operate on present sufficiently accurate contours strength. The drawback is that some of these contours are not at all pertinent. For example, the sky in figure 3.35, which is a zone with an almost null gradient, is nonetheless divided by the superpixel method.

This approach, such as the preceding ones, is gradient-based. Standard methods can be used to compute this gradient, or a specific gradient computation method can be designed for a given application. In the following, we expose the different gradients we considered.

3.5.2 Choice of the gradient

The regions adjacency graphs that we get out an image is dependent of the gradient we consider at two levels:

— Whether it is in a watershed or superpixels approach, the fine segmentation will be of better quality if all potentially important contours are accentuated in the gradient.

— The dissimilarities between regions of the graph are often based on local gradient information between those regions, for example the means of the gradient along the borders separating adjacent regions.

This is why it is of primary importance to have a gradient that highlights all potentially important contours. The morphological gradient of an image is defined as:

$$\mathcal{G}(\mathbf{I}) = \delta(\mathbf{I}) - \varepsilon(\mathbf{I}) \tag{3.20}$$

It corresponds to the gradient magnitude of the topographical surface associated with the processed image and provides a mean to highlight the high frequencies of the image. However, using this gradient can lead to contours that are not sufficiently well accentuated, and this for several reasons.

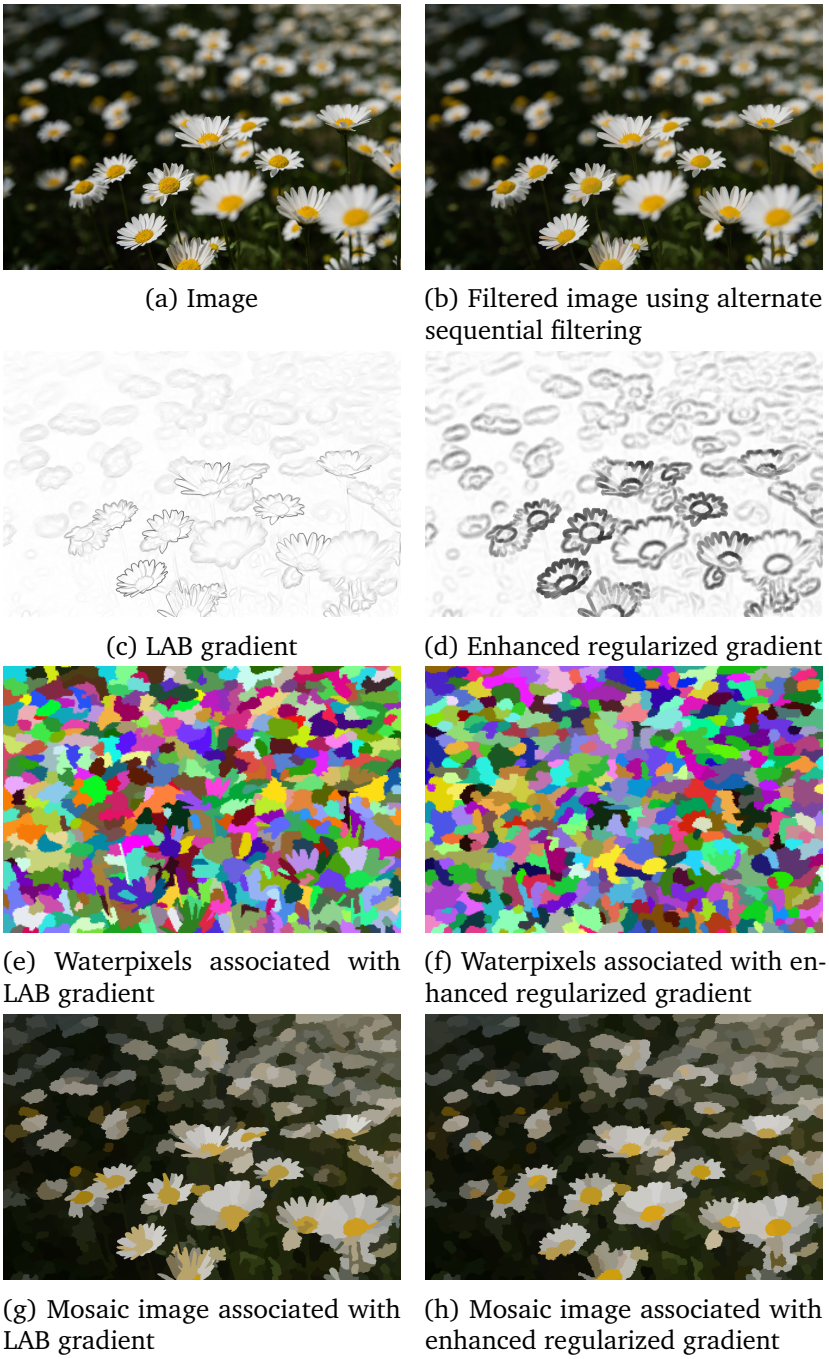
The first is that noise is systematically amplified in the morphological gradient. So it should be either filtered out before computation or be modified to take this into account. For example, in [Lerallut (2006)] the author defines the gradient at a given image point as the distance between the two most significant modes of the brightness distribution covering the neighborhood of the pixel under consideration. Similarly, in [Arbelaez et al. (2011)] the authors build a gradient by comparing brightness distributions from either sides of a circular patch centered at the pixel of interest and for different orientations.

Secondly, one must choose the size of the considered patches in such approaches. Indeed, a sufficiently large patch size can widen the scope of analysis, thus enabling to be attentive to smoothly evolving brightness transitions, which is particularly useful when dealing with out-of-focus images. But if the patches are too large, thin homogeneous regions can be filled with high gradient values, which is not desirable for the detection of contours. In [Bricola et al. (2015)], the authors derive from the regularized gradient [Rivest et al. (1992)] a multi-scale morphological gradient that addresses both those issues. The main idea of the regularized gradient [Rivest et al. (1992)] is to compute a morphological gradient at different scales and retains the highest gradient values across all scales, and then to thin the result using an erosion. In [Bricola et al. (2015)], the authors propose enhancements to this technique: they avoid high gradient values in high frequencies images regions by filtering the input image beforehand, and replace the erosion step by a brightness transitions detection step. An example illustrating its effectiveness is provided in figure 3.36.

For the above explained reasons, we choose to use this gradient in the waterpixels approach as we found it to provide guarantees for not missing important contours while still fast enough to compute to keep the entire chain not prohibitively long for the user.

3.5.3 Exploitation of colour information

There are many different ways to compute color gradients, and the adaptation of morphological operators to colour images remain an active research topic. The reason behind this is that there is no obvious way to order colours and to make sense of dilations or erosions on colours



(a) Image

(b) Filtered image using alternate sequential filtering

(c) LAB gradient

(d) Enhanced regularized gradient

(e) Waterpixels associated with LAB gradient

(f) Waterpixels associated with enhanced regularized gradient

(g) Mosaic image associated with LAB gradient

(h) Mosaic image associated with enhanced regularized gradient

Figure 3.36 – Illustration of the usefulness of the enhanced regularized gradient.

without a specific application domain. However, when we compute gradients, we are interested in the distance between any two colours. So we first have to represent our images in a color space, whether it is the classical (R,G,B) space or a perceptually-based one such as the CIE-LAB space, in which perceptually different pixels are far from one another. Then we have to define a distance between pixels. It can be intrinsic to the color space (for example a euclidean distance between pixels of the color space) or marginal, i.e. defined on each component. We choose to use the CIE-LAB and marginal distances.

Then we can replace the gradient between two pixels by a given combination of the colour distances between them. In our case, we prefer to retain the maximum rather than the mean of all channels gradients in order to avoid missing any potentially important contours. Indeed, for building a satisfying oversegmentation of the image, the recall of perceptually meaningful contours retrieving is more important than its precision.

3.5.4 Choosing the dissimilarity

In our framework, we construct a $\mathcal{RAG}$ based upon a fine partition produced by an initial segmentation (tipycally a set of superpixels) and containing all contours making sense in the image. We must define a dissimilarity measure between adjacent tiles of this fine partition that correspond to the edges valuations of the $\mathcal{RAG}$.

We choose to construct dissimilarities based on local gradient information, as this information naturally corresponds to discrepancies between regions, and thus is connected to the notion of dissimilarity.

Since the frontier between two tiles in a fine partition is not limited to one pixel, we need to choose a way to integrate the gradient information along the frontier. We tested several possibilities to compute these valuations:

— Taking the maximum of the gradient values along the frontier, which would be the ideal choice if the image is not noisy, as it would ensure us that we do not miss any interesting contour.

— Taking the mean of the gradient values along the frontier, which is the usual choice but appears to be potentially misleading, as it sensitive to extreme values.

— Taking the min of the gradient values along the frontier, which is the usual choice in mathematical morphology, because floodings are made along the saddle points of the departure relief, which are the minimums.

— Taking the median of the gradient values along the frontier: we chose this possibility, as it provides a good estimate of the average significance of the gradient on this frontier, while still being more robust to noise and outlier values.

3.6 Presentation of the Smil Library and the Implemented Module

3.6.1 The Smil Library

A large majority of the algorithms used to generate and use the morphological hierarchies exposed in this thesis have been implemented within Smil, created at the Center of Mathematical Morphology [Smil. (2015)]. Smil is a 2D/3D image library written in C++, which aims to be lightweight, fast, easy to use and to extend. It is developped using templates, which means that it can be used with any standard types, and it uses swig to be wrapped in several languages, such as Python, Java, Ruby or Octave.

3.6.2 The implemented module

During this thesis, we have thus developed a large module within Smil to compute, modify and visualize different kind of hierarchies. This represented an important part of this work. These hierarchies are implemented mainly using dynamic programming on graphs, which makes the related algorithms very efficient. It indeed allows to compute new characteristics for each region of the hierarchy in a bottom-up fashion and to reorganize the internal structure of the hierarchy at the same time. In a few lines of codes one can thus get from an image to the computation and visualization of a large number of morphological hierarchical segmentations of this image.

We make full use of the modular environment of Smil by working with C++ templates, thus making possible to easily use the implemented methods with different images types (UINT8, UINT16 or UINT32 for example). We think of and implement the hierarchies as dendrograms, i.e. binary trees representing the progressive merging of fusions for different levels of the hierarchies. Such structures are compatible with dynamic programming using pointers in C+ +, and thus lead to efficient algorithms. We rely on the object-oriented aspect of C++ by creating two classes: the class DendroNode, which correponds to a given node in a dendrogram representing a hierarchy, and the class Dendrogram, which essentially consists in a vector of DendroNode all connected between them through pointers. A comprehensive view of the structure of this code is provided in figure 3.37.

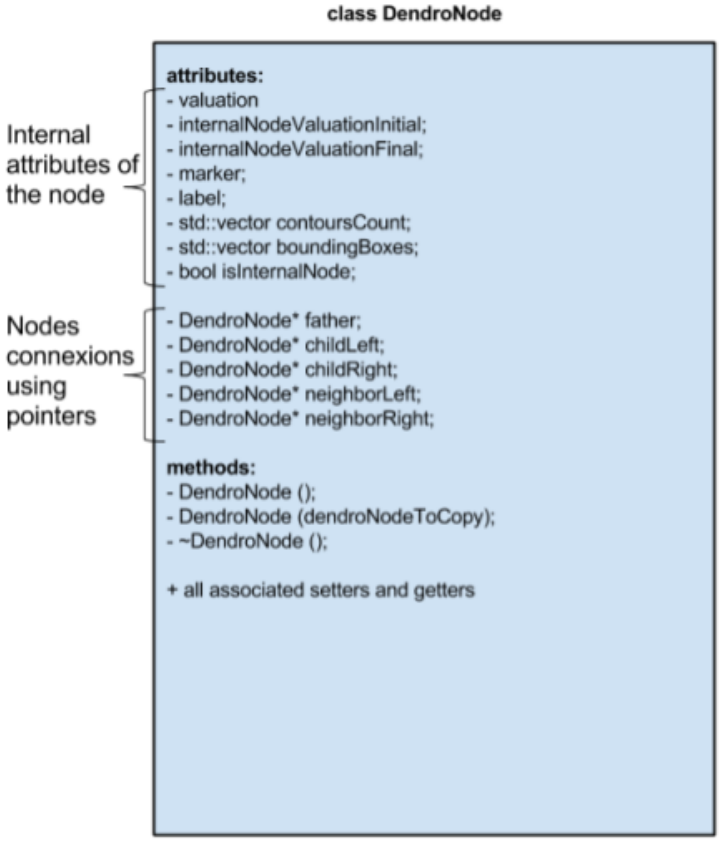
Once a hierarchy has been created, one can get a segmentation out of it using different possibilities:

— simply thresholding the highest saliency values,

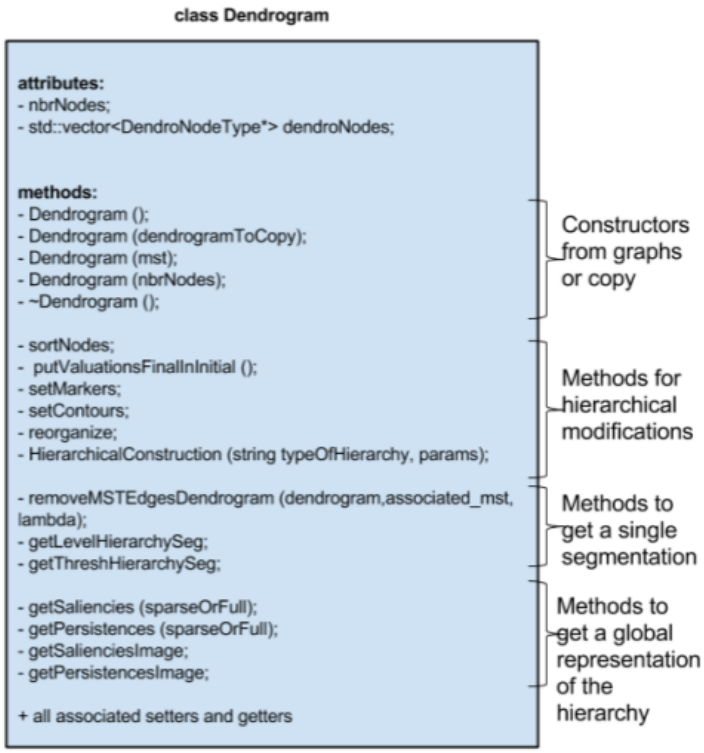
— marking some nodes as important ones and then computing a partition accordingly, which is known as marker- based segmentation,

— smartly editing the graph by finding the partition that minimizes an energetic function (such as for example the Mumford-Shah functional).

All these possibilities have been implemented and tested. More specifically, the first one allows us to very simply get the different levels produced by a hierarchical segmentation, whether it is



(a) Class DendroNode.



(b) Class Dendrogram.

Figure 3.37 – Comprehensive view of the internal structure of the main classes of the implemented hierarchical segmentation module in Smil.

by specifying a threshold saliency value or more simply by choosing the number of regions we want to have in the output segmentation.

One can also obtain several equivalent global representations of such a hierarchical segmentation to work with.

— An adjacency sparse matrix in which each non-zero value corresponds to the saliency attributed to the edge linking two adjacent regions in the hierarchy.

— A full matrix in which all couples of nodes have a valuation corresponding to the subdominant ultrametric between them, that is to say the highest edge on the path linking them for this ultrametric.

— A $\mathcal{UCM}$ (Ultrametric Contour Map), i.e. an image in which each contour takes as value its saliency in the hierarchy. Representing a hierarchy by its $\mathcal{UCM}$ is an easy way to get an idea of its effect because thresholding an $\mathcal{UCM}$ always provides a set of closed curves, that is to say a partition. In the thesis, for better visibility, we represent $\mathcal{UCM}$ with inverted contrast.

— A dendrogram.

3.6.3 Examples of implementations

The implementation of the module hereby presented has been made in order to be fast and robust. While giving all its technical details may be too heavy and useless, we illustrate in this section the kind of procedures that were set up to solve efficiently technical issues.

These procedures make a heavy use of the dendrogram structure. Let us consider as an example the problem of efficiently computing the areas of all regions of a hierarchical segmentation. Intuitively, working only on the sequence of nested partitions to compute them would imply a scan of all pixels in all partitions. Working on the dendrogram structure allows us to scan the image only once, in order to compute the areas of the leaves, and then propagate the to all parents as the sum of the areas of children. Of course, the drawback is that algorithms then become intricate in terms of coding and necessary data structures. For example, computing neighbors of a certain region, which is a trivial task when scanning the successive partitions of a hierarchical segmentation, requires tailored data structures and algorithms in a dendrogram. Formally, let us consider an image with p pixels, simplified to an image with s superpixels such that m mergings are operated in the hierarchical segmentation process. Then computing areas on all regions of the successive partitions of the hierarchical segmentation has a cost equal to the one of scanning all pixels in these partitions, i.e. $p \times (m + 1)$. Using the dendrogram structure, we only need to scan the image once and then we can propagate the area values, so that the computational complexity is $p + m$, which is a major improvement. This is why we made use of tailored algorithms and data structures to compute the bounding boxes, perimeters, neighbors and other interesting cues concerning regions using the dendrogram representation.

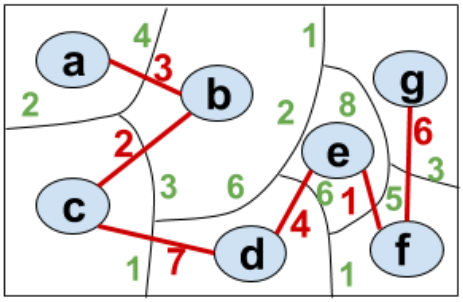
Computation of the $\mathcal{UCM}$ using the dendrogram structure

In order to compute the $\mathcal{UCM}$ associated with a hierarchy, one must retain for each pair of adjacent nodes the subdominant ultrametric value of the edge linking them for the consider hierarchy. A naive approach to do so is to work with the segmentations themselves and stack the contours of the successive segmentations. Instead of that computationally expensive method, we once again make use of the dendrogram structure describing the hierarchy.

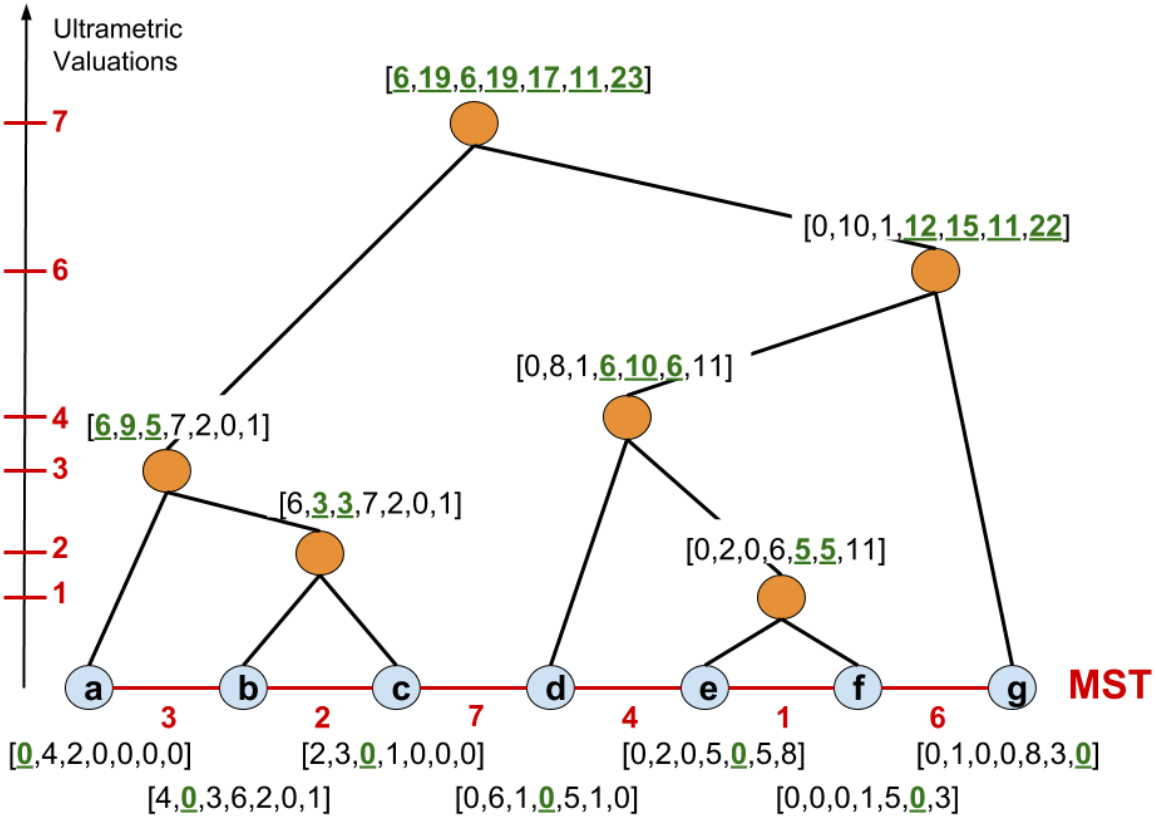
To do so, we have to keep in memory the adjacent regions as we are computing the subdominant ultrametric incrementally. We thus assign to each node of the dendrogram a dictionary $\overline{\sim}_{regions}$ of size the number of nodes, initiliazed for each leaf at 1 for the leaf index (corresponding to the label number of the underlying region of the fine partition) and 0 elsewhere. Then for internal nodes of the dendrogram, the dictionary is equal to the supremum of the dictionaries of the left and right children, and provides the information of the underlying regions for any sub-dendrogram.

We also retain the information regarding the adjacent nodes and the length of the contours between them. To keep this information, we explore once the labels image of the fine partition and assign to each leave of the dendrogram a second dictionary $\overline{\sim}_{contours}$ of size the number of nodes: for each leave i, $\overline{\sim}^{i}_{contours}(j) = l_{ij}$ the length of the frontier between the regions corresponding to the nodes i and j of the graph. In the same way as with $\overline{\sim}_{regions}$, we can easily propagate this information in a bottom-up fashion in the dendrogram, and thus have access to the information regarding which nodes are connected to one another.

The obtention of both dictionaries are illustrated on an example in figure 3.38.



(a) Region Adjacency Graph.



(b) Corresponding Dendrogram.

Figure 3.38 – (a) Region Adjacency Graph: in red are represented the $\mathcal{MST}$ and the associated edges valuations, in green the lengths of the contours between adjacent regions. (b) Corresponding dendrogram with the dictionaries initialized for the leaves and computed in a bottom-up fashion for the internal nodes. For the sake of clarity, we represent for each node both dictionaries $\eqsim_{contours}$ and $\eqsim_{regions}$ with a unified representation:
- The valuations correspond to $\eqsim_{contours}$, i.e. to the aggregated sum of contours values over all children of the sub-dendrogram associated with the considered internal node.
- The underlined bold numbers correspond to $\eqsim_{regions}$, i.e. to the regions that have already fuse for the sub-dendrogram associated with the considered internal node.

Now that we are provided with these dictionaries, computing the $\mathcal{UCM}$ can be made fast. The idea is to obtain a sparse adjacency matrix with non-null subdominant ultrametric values for regions that are connected in the image. To do so we operate as described in algorithm 3.3.

First, a traversal of the dendrogram leaves allows to determine which labels are adjacent in the image. Secondly, a traversal of the dendrogram internal nodes leads to the computation of the sudbominant ultrametric values for these pairs of labels, as illustrated in figure 3.39. Note that this algorithm is similar to the process described in section 7 of [Cousty et al. (2017)], although it is obtained with structures allowing us to get rid of the use of the initial graph and to work only with the dendrogram. However, its time complexity is in $O(|\mathsf{V}|^2)$, as it does not operate on an union-find basis that would imply a joint construction of the $\mathcal{MST}$ and the hierarchy. In our framework, we indeed modify the values of the $\mathcal{MST}$ edges, and then induce a new ultrametric/saliency map out of it.

Algorithm 3.3: $\mathcal{UCM}$ computation.

Data: A dendrogram for which the dictionaries $\eqsim_{contours}$ and $\eqsim_{regions}$ have been computed, the associated labels image.

Result: The $\mathcal{UCM}$ corresponding to this hierarchy.

Initialize a list of adjacent nodes $\mathrm{L_{Adj}}$.
while *all leaves have not been traversed* **do**
  For a node i:
  if $\eqsim^i_{contours}(j)$ **then**
    $\mathrm{L_{Adj}} += (i, j, 0)$
  end if
end while
Sort internal nodes of the dendrogram by decreasing valuations.
while *all internal nodes have not been traversed* **do**
  For a node i of valuation $\boldsymbol{\lambda}$, and with children the nodes k and l:
  if $(\exists j \text{ s.t. } \eqsim^k_{regions}(j) \wedge \eqsim^l_{regions}(j))$ **then**
    if $\exists \mathrm{L_{Adj}}(k, l, 0)$ **then**
      $\mathrm{L_{Adj}}(k, l, 0) \leftarrow \mathrm{L_{Adj}}(k, l, \boldsymbol{\lambda})$
    end if
  end if
end while
Generate the $\mathcal{UCM}$ from $\mathrm{L_{Adj}}$ and the labels images.

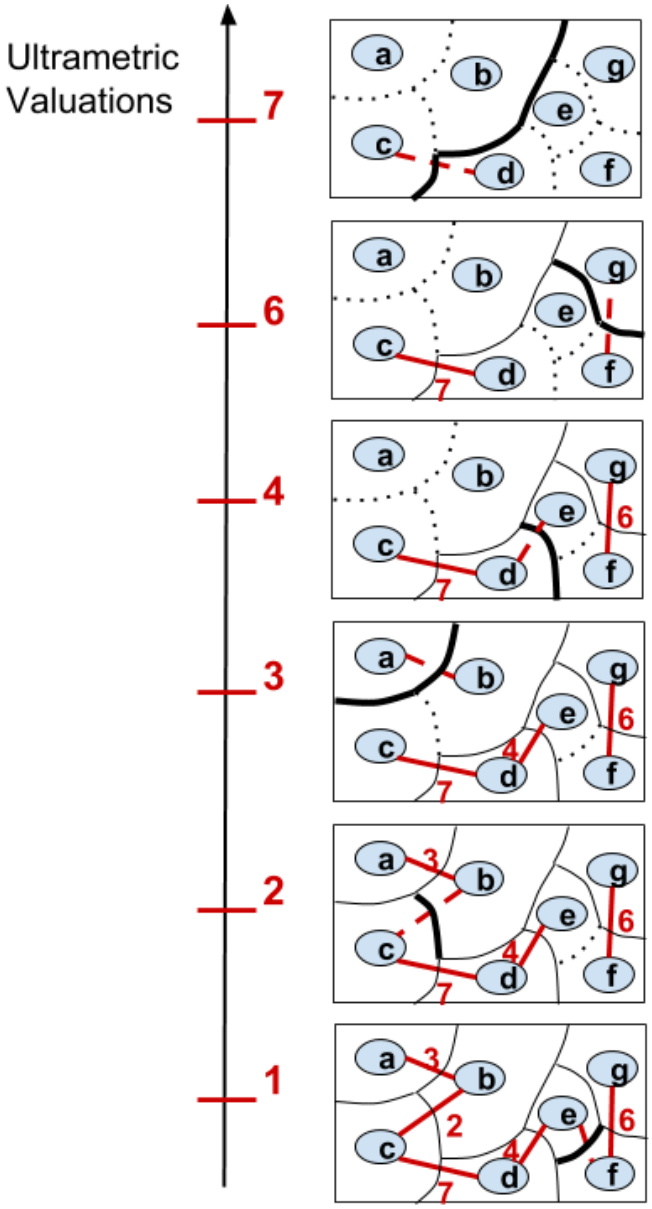


Figure 3.39 – Illustration of the computation of the ultrametric values for each pair of adjacent nodes of the image. The example and structures are the same as in figure 3.38. As the ultrametric values rise, the regions progressively fuse: on the figure, the bold contours admit as saliency the ultrametric value for which they disappear.

Fast computation of the contours length for each level of the hierarchy

Furthermore, as we have seen in section 3.3.7, some types of hierarchies need the information of the contours length between regions in the hierarchy in their construction. To obtain this information at each level of the dendrogram/hierarchy, we also make use of the dictionaries $\bar{\sim}_{contours}$ and $\bar{\sim}_{neighbors}$, as described in the algorithm 3.4. An example is also provided in figure

3.40 for a specific level of the hierarchy presented in figures 3.38.

Algorithm 3.4: Fast contours length computation for all levels of the hierarchy.

Data: A dendrogram for which the dictionaries $\approx_{contours}$ and $\approx_{regions}$ have been computed.

Result: The length of the contours for each level of the associated hierarchy.

1 Initialize a vector V of size $|\mathsf{E}|$ to contain the contours length for each level of the hierarchy.

while *all internal nodes have not been traversed* **do**

2 For the level l:
$V(l) = \sum_k \approx^l_{contours}(k) \times (1 - \approx^l_{regions}(k))$

3 **end while**

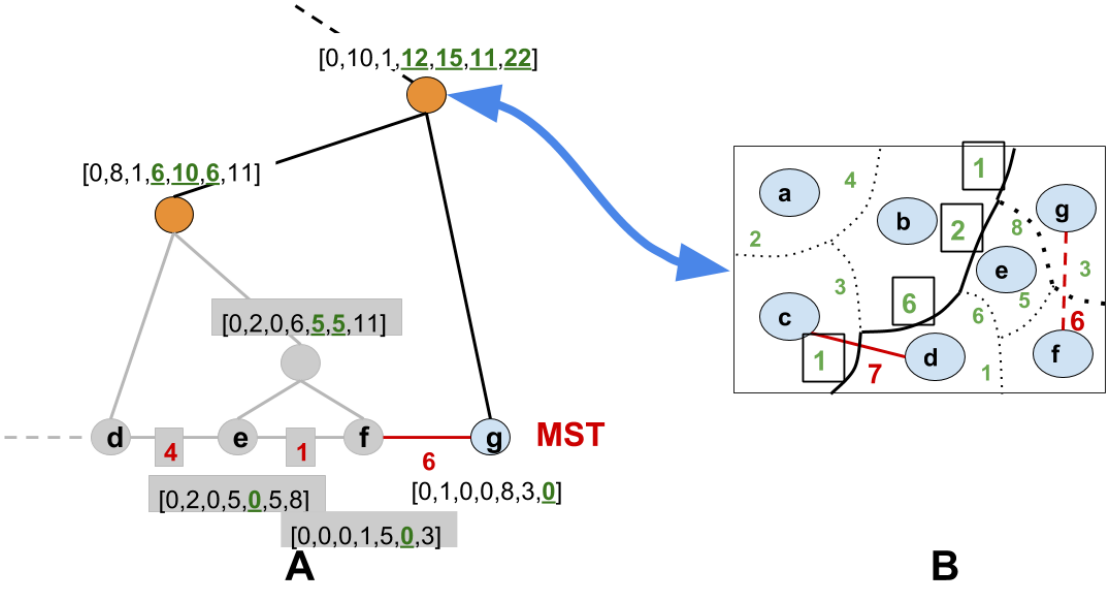


Figure 3.40 – Computation of the contours length for the ultrametric value of 6 in the hierarchy introduced in figure 3.38. **A**: sub-dendrogram ; **B**: its associated segmentation. We can see that the length of the only remaining contour in the segmentation is indeed given by the formula of the algorithm 3.4, as $1 + 6 + 2 + 1 = 11 = (0 + 10 + 1) \times 1 + (12 + 15 + 11 + 22) \times 0$.

Computation time

In table 3.2 are presented the computation times of the different steps of the hierarchical segmentation framework that we introduce in this chapter. The tests have been conducted on a laptop equipped with an Intel i5-4210M CPU (2.60GHz) and 16 GO of RAM. Note that the graph creation has not been optimized in the context of this work, and that it is part of a Smil version to be published.

Table 3.2 – Table presenting the computation time of each step of the hierarchical segmentation framework presented in this thesis for a 600×600 color image, and a fine segmentation containing approximately 1000 regions.

Algorithm Step	**Time (in s)**
Fine partition (waterpixels)	0.863
Graph creation	1.360
MST from graph	5.379×10^{-3}
Dendrogram from MST	0.711
Hierarchy computation from original dendrogram	4.044×10^{-2}
UCM computation from dendrogram and labels image	0.805

3.6.4 A User-Friendly Environment

The Smil library, although it is written in C++ for the sake of efficiency, can be wrapped into Python for the sake of simplicity of use. Indeed, Python is a high-level language, which makes it easier to work with. Furthermore, it allows the Smil library to be used within an extremely rich environment of open-sources libraries focused on data analysis and machine learning (e.g. scikit-learn), deep learning (e.g. keras), image processing (e.g. numpy) or computer vision (e.g. OpenCV), thus facilitating the test of additional ideas and algorithms. Finally, it enables us to provide a user-friendly experience through the creation of IPython notebooks [Pérez et al. (2007)] documenting the method in form of an executable algorithm which may be run on any image and which presents the results of the processing.

Combining interactive programming with markdown documentation and portability, IPython notebooks are indeed a powerful scientific-computing tool. Working within a notebook gives one the ability to experiment as in a terminal-based interpreter and to save the work and results when over.

By working with these tools, the user simply has to provide new images or markers needed by the segmentation algorithms. He can then easily visualize the outputs of different hierarchical segmentation methods, through their $\mathcal{UCM}$ or by visualizing different levels of those hierarchies. Users can also overlay the segmentation over the image or draw the underlying graphs. Combinations of all these visualization methods allow for an easy and fast interpretation of the segmentations results. Thus, complex morphological processes can be explored easily and interactively.

To conclude, these methods have been made available to the Smil users community. They have constituted a great programming effort to lead to a powerful, easy to use and versatile tool. An example of the environment in a simple case is presented in appendix 3.

Chapter 4

Combinations of Hierarchies

Résumé

Nous avons introduit dans le chapitre précédent des méthodes pour obtenir de façon contrôlée des segmentations hiérarchiques obéissant chacune à un critère particulier. Toutefois, dans de nombreuses applications de traitement d'image, les images que l'on cherche à segmenter ou caractériser se distinguent par des traits discriminants différents. C'est pourquoi, dans ce chapitre, nous proposons des méthodes pour obtenir des hiérarchies dérivées qui capturent des caractéristiques complexes de l'image.

Nous distinguons deux modes principaux pour enrichir les hiérarchies. L'enrichissement séquentiel associe à une hiérarchie de départ une hiérarchie dérivée en remplaçant les valuations initiales de l'arbre de poids minimum par de nouvelles valuations calculées dans l'image.

La combinaison parallèle se fait simplement en définissant une nouvelle dissimilarité sur les arêtes du graphe, comme fonction des distances ultramétriques des hiérarchies à combiner. Cette nouvelle dissimilarité n'est en général pas une ultramétrique et il faut en dériver l'ultramétrique sous dominante associée pour obtenir la hiérarchie finale, ce qui a un certain coût. Ce coût de calcul est fortement réduit dans le cas où les hiérarchies à combiner ont en commun le même arbre de poids minimum. La hiérarchie résultant de la combinaison garde alors le même arbre de poids minimum, à partir duquel on peut calculer aisément les poids de toutes les autres arêtes.

Cependant, le nombre des hiérarchies explose avec toutes ces possibilités de les chaîner ou de les composer. Or chaque hiérarchie munie de sa distance ultramétrique devient un espace métrique et on peut géométriser l'espace des hiérarchies grâce à la distance de Gromov-Hausdorff qui définit une distance entre espaces métriques. Cette distance est difficile à calculer dans le cas général mais est très aisée à calculer entre deux hiérarchies qui ont en commun la même partition fine. En combinant cette distance avec des outils de réduction de dimensionnalité et de visualisation des données, nous obtenons ainsi une méthodologie pour étudier l'espace combinatoire des hiérarchies.

Abstract

The previous chapter has introduced various methods to construct hierarchies, each of them stressing a particular feature of the image. However, in many cases, the images to segment or analyze are characterized not by one but by the combination of several features. For this reason, we propose in this chapter methods for obtaining derived hierarchies, triggered by more complex features of the image to analyze.

We distinguish two modes for enriching hierarchies. The sequential mode associates to a starting hierarchy a derived hierarchy by replacing the weights on the MST by new weights, evaluated by taking into account new image features. The parallel combination of hierarchies consists in defining new dissimilarity weights on the edges of the graph, as a function of the ultrametric distances of the hierarchies to combine. This new dissimilarity is in general not an ultrametric and one has to derive the associated subdominant ultrametric, which is a relatively costly operation. This cost is markedly reduced in the case where the hierarchies to be combined share a common minimum spanning tree. The derived hierarchy then keeps this minimum spanning tree. The new weights have then simply to be computed on the edges of the MST and then spread out to all other edges of the graph.

However with these methods for chaining or composing hierarchies their number literally explodes. As any hierarchy provided with its ultrametric distance becomes a metric space, it is possible to study the distance between them using the Gromov Hausdorff distance. This distance, difficult to compute in the general case, is easily computed in the case where the hierarchies have in common the same finest partition. Applying dimensionality reduction techniques with data visualization methods to the space of hierarchies provides us with a powerful tool for exploring and analyzing this space.

4.1 Introduction

In the previous chapter, we introduced several ways to generate hierarchies, and now have at our disposal a versatile framework to characterize images through different hierarchies. We can choose the type of hierarchy (trivial, surface-based or volume-based SWS, waterfall etc.), decide whether we wish to return to the image to measure additional properties during construction, set a number of markers in certain cases. Thus, the number of hierarchies we can generate is substantial. The question of how to use them for particular images characterizations now arises. Indeed, in a lot of real-life applications, the images we deal with are structured and share similar characteristics. We can then use these hierarchical segmentations to characterize them in a controlled way, whether it is for example for segmentation purposes (by finding the hierarchy that facilitates it), image classification or anomaly detection.

When dealing with images, they are often hard to analyze, filter or segment, and their content can only be characterized by the conjunction of multiple criteria. This is however a major difficulty in many segmentation methods, as taking advantage of several distinct characteristics

which are not necessarily homogeneous is not straighforward. Several approaches exist to do so:

— It is rare that a simple thresholding on the output of a watershed transform applied to a gradient image leads to a proper extraction of the objects of interest. This is why it is often necessary to inject prior knowledge in the form of markers to direct the segmentation process.

— One can also decompose images as hierarchies of segmentations, such as the min-tree or max-tree [Salembier, Oliveras, et al. (1998)], the tree of shapes [Monasse et al. (2000); Ballester et al. (2003)] or MSF hierarchies [Meyer (1994a)], and prune the tree structures of the hierarchies. These hierarchies are attribute-based, and such attributes, while leading to interesting results, are not necessarily increasing: this makes them difficult to handle and may lead to hierarchies lacking stability properties related to morphological filtering, although some solutions have been proposed to force their increasingness.

— Energetic approaches consist in the designing of functionals that are weighted sums of heterogeneous terms that counterbalanced each others. A good example is the Mumford-Shah functional [Mumford et al. (1989)], which integrates measures of photometric and boundary lengths. The goal is then to optimize such functionals to find the best trade-off between their different terms. These methods however necessitate a lot of parameters-tuning.

— One can also search in a discrete or continuous space the graph-cut optimizing the flow [Stawiaski (2011)].

— Combine criteria to build tailored energetic hierarchies [Kiran et al. (2014a)].

In this chapter, we investigate how to build a multi-parameter approach of images by combining several hierarchies. One can see each hierarchy as a filter of *pure* color. To obtain the best contrast between objects of an image, one has to choose the best color filter, so that the segmentation problem is reduced to a thresholding problem. For example, in an ophthalmics image of the back of the eye, the best way to separate the retina from the red vessels is to apply a green filter: this isolates vessels and makes their subtraction from the image easier. In a similar way, morphological hierarchies can be seen as *pure* geometrical filters that can be combined to obtain *derived* hierarchies that can discriminate complex structures. We will expose two ways to do so: one *sequential*, corresponding to chainings of hierarchies, and one *parallel*, corresponding to functional combinations of hierarchies.

Since the number of possible hierarchies is even bigger once these combinations have been introduced, we expose tools to structure and study the hierarchical space. We take advantage of the fact that several hierarchies constructed upon the same set of regions of a fine partition can each be seen as distinct metric spaces. The Gromov-Hausdorff distance then allows to measure the distance between these metric spaces. For a given set of hierarchies, we can thus generate a distance map between them. This opens new ways to suppress redundancies and create a restrained descriptive family of hierarchies. It also allows to study the properties of hierarchical combinations or visualize the effects of several different hierarchies.

We begin by presenting how we can structure the hierarchical space, and introduce a new methodology to analyze and visualize the relative descriptive power of different hierarchies, that leverages dimensionality reduction algorithms as well as the Gromov-Hausdorff distance. Then we present different ways to combine hierarchies and apply this methodology to study their properties.

4.2 Structuring the Hierarchical Space

Several levels of image representation, illustrated in figure 4.1, have until now been introduced in the context of this work:

1. The finer level is the level of the image itself.
2. Computing a fine partition of the image already constitutes a rougher model of it, that we can see as a region-based representation.
3. A structuring of the regions of the fine partition as a hierarchical segmentation. Each hierarchy is fully characterized by the set of points that it regroups as long as by the ultrametric it defines between these points.

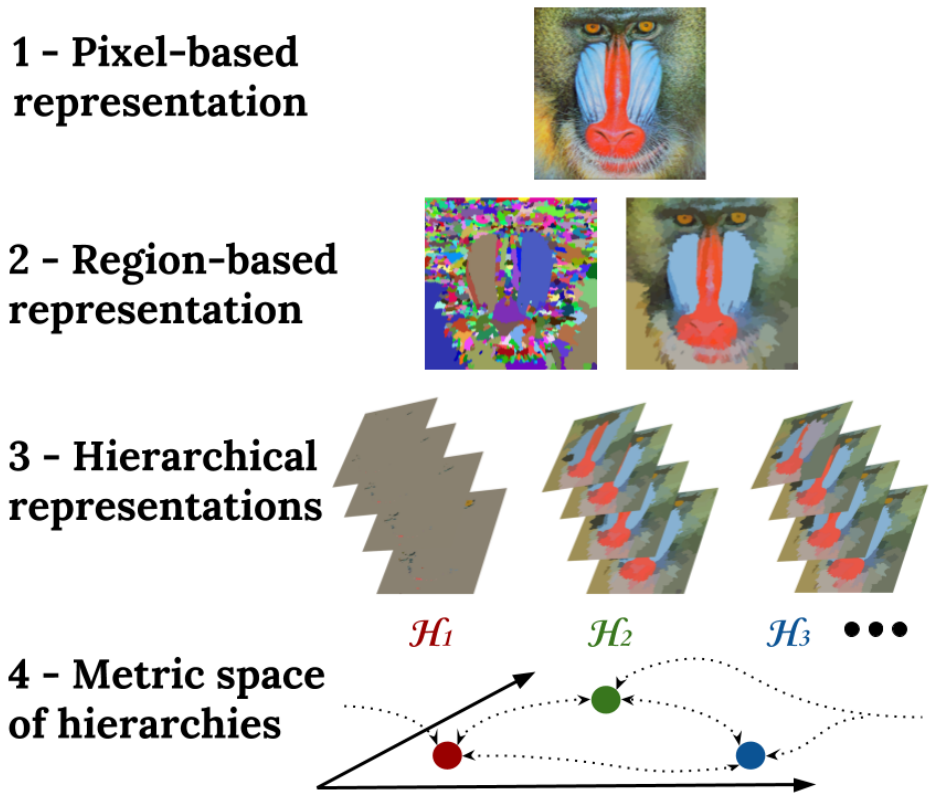


Figure 4.1 – The different image representations considered in this work.

Since we can generate for the same image several different hierarchies, a new level of representation naturally emerges:

4. One can indeed wonder whether we can structure the space of hierarchies itself into a metric space. Doing so would open a path to study the properties of different hierarchical clustering methods, to evaluate to which extent they differ and are complementary.

In order to operate this structuring, one need to define a distance between these ultrametric spaces. The Gromov-Hausdorff distance [Gromov et al. (1981); Gromov (2007)] defined between metric spaces is a possible distance. It has been introduced in section 2.4.3. However, it is generally difficult to compute between two metric spaces defined over different sets, as one must then match data points before any distance computation. This computationally heavy operation leads some authors to provide heuristics to approximate the Gromov-Hausdorff in specific configurations [Mémoli (2004); Agarwal et al. (2015)].

In the context of this thesis, instead of attempting to compute distances between hierarchies from different images, we generate several hierarchies over the same image and compute GH-distances between them. Indeed, in this case, these ultrametric spaces are defined over the same set of points and the GH-distance computation is much easier, as seen in section 2.4.3. More formally, given an image $\mathbf{I}$, one can compute several hierarchies of this image $((\mathcal{H}_1, \boldsymbol{\lambda}_1), \ldots, (\mathcal{H}_N, \boldsymbol{\lambda}_N))$, and also compute for any $(i, j) \in \{1, \ldots, N\}^2$ the distance $\mathrm{d_{GH}}((\mathcal{H}_i, \boldsymbol{\lambda}_i), (\mathcal{H}_j, \boldsymbol{\lambda}_j))$. One thus define a metric space over hierarchies, that constitutes a way to define new discriminative image representations, as we shall see in chapter 6.

It also allows for a quantitative study of the different hierarchies using a variety of tools. Notably, once we have defined a metric space on the space hierarchies through the computation of inter-hierarchies distances, the interpretation of these distances is not straightforward. This is the reason why, in the next section, we introduce dimensionality reduction algorithms for a better visualization and understanding of hierarchical spaces. Thus we are able to see what hierarchies are similar, which ones are complementary, and so on. Provided with these tools, we then introduce different ways to combine hierarchies and illustrate how these analysis tools can help us understand their effects and properties.

To sum up, starting from a set of unordered regions, we obtain a set of structured and rich representations of this set in a low-dimensional space. In the intermediary step of this process, each hierarchy has been built by introducing a lot of prior information on the type of regions that one wants to pop up in it.

4.3 Dimensionality Reduction Algorithms for Hierarchical Space Visualization

When provided with data embedded into a high-dimensional space, one may want to be able to project them in a space of lower dimension. This is the purpose of dimensionality reduction techniques, which aim at preserving as much as possible the structure of the high-dimensional data in a low-dimensional map. We use such techniques to visualize the relative effects of hierarchical segmentations methods.

Many approaches exist to do dimensionality reduction. Traditional ones include Principal Component Analysis (PCA) [Hotelling (1933)] or classical Multidimensional Scaling (MDS) [Torgerson (1952)]. These are linear techniques which mainly aim at keeping the low-dimensional representations of dissimilar datapoints far from one another.

However, when dealing with high-dimensional data, it is usually more important to preserve the low-dimensional representations of very similar data points close together. Indeed, the curse of dimensionality [Bellman (1957)] implies, among others, that data points in high-dimensional spaces are usually far from one another. Non-linear techniques thus put emphasis on keeping the similar data points close to one another in the low-dimensional space. A review of most of them can be found in [Lee et al. (2007)]. In this section, we restrain ourselves to the use of multidimensional scaling [Torgerson (1952); Cox et al. (2008)] and t-distributed stochastic neighbor embedding (t-SNE) [Maaten et al. (2008)].

In the context of this work, we can generate for each image a set of hierarchies, that can be seen as a feature representation space of it. We are then interested in a dimensionality reduction of this space to have a better understanding of these features.

We begin by presenting a short description of these algorithms. Then, we explain how these techniques allows the visualization of relative contributions of hierarchies generated upon the same set of points.

4.3.1 Multidimensional Scaling (MDS)

Multidimensional Scaling (MDS) is a low-dimension visualization technique of the level of similarity of individual cases in a dataset. More precisely, it is a form of dimensionality reduction that takes as input a distance matrix representing distances between objects in a space of dimension M, potentially big, and places each object in a space of inferior dimension $m << M$(typically 2 or 3) while trying to preserve distances between objects. Choosing $m = 2$ allows for a visualization as a scatterplot of the distances between objects.

Classical MDS [Torgerson (1952)]

It is also known as *Principal Coordinate Analysis (PCoA)*. It takes as input a matrix of dissimilarities between pairs of items, and outputs a coordinate matrix whose configuration minimizes a loss function called *strain*.

Let us write $X = (x_1, ..., x_N)$ the vector of points in the high-dimensional space, and $D = [d_{ij}]_{(i,j)\in\{1,...,N\}^2} = [d(x_i, x_j)]_{(i,j)\in\{1,...,N\}^2}$ the dissimilarity matrix associated with the Euclidean distance $d : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}^+$ defined between them.

In the classical MDS, the *strain* loss function is given by:

$$Strain_D(x_1, ..., x_N) = \left(\frac{\sum_{i,j}(b_{i,j} - \langle x_i, x_j \rangle)^2}{\sum_{i,j} b_{ij}^2} \right)^{\frac{1}{2}}, \tag{4.1}$$

with $b_{i,j}$ are the terms of the matrix B defined hereafter. The classical MDS algorithm uses the fact that the coordinate matrix can be derived by eigenvalue decomposition from $B = XX^T$.

Accordingly, the matrix B can be computed from the proximity matrix D by using double centering transformation, as follows:

1. Set up the square proximity matrix $D = [d_{ij}^2]$.
2. Double centering: $B = -\frac{1}{2}JDJ$, with J the centering matrix $J = I - \frac{1}{N}11^t$.
3. Determine the m largest eigenvalues $\lambda_1, ..., \lambda_m$ and corresponding eigenvectors $e_1, ..., e_m$ of B (where m is the number of dimensions desired for the output).
4. Projection as $X = E_m \Lambda_m^{\frac{1}{2}}$, where E_m is the matrix of m eigenvectors and Λ_m is the diagonal matrix of the m largest eigenvalues of B.

Classical MDS supposes Euclidean distances, and needs to be generalized to use other types of distances.

Metric MDS [Cox et al. (2008)]

The metric MDS generalizes the classical MDS application to a variety of loss functions and input matrices of known distances.

For a (low) dimension m and a monotone function f, the metric MDS seeks to find an optimal configuration $\tilde{X} \subset \mathbb{R}^m$ such that $f(d_{ij}) \approx \hat{d}_{ij} = ||x_i - x_j||_2$ is as close as possible. This function f can be taken to be a parametric monotonic function (i.e. $f(d_{ij}) = \alpha + \beta d_{ij}$).

The useful loss function to optimize is then called *stress*, and the optimization process referred to as *stress majorization*. This stress function is:

$$Stress_D(x_1, ..., x_N) = \left(\frac{\sum_{i \neq j = 1,...,N} (f(d_{ij}) - ||x_i - x_j||)^2}{\sum_{i,j} d_{ij}^2} \right)^{\frac{1}{2}} \tag{4.2}$$

The *usual* metric MDS corresponds to the special case where $f(d_{ij}) = d_{ij}$, so that the stress function to optimize becomes:

$$Stress_D(x_1, ..., x_N) = \left(\frac{\sum_{i \neq j = 1,...,N} (d_{ij} - ||x_i - x_j||)^2}{\sum_{i,j} d_{ij}^2} \right)^{\frac{1}{2}} \tag{4.3}$$

Note that the usual metric MDS optimization solution in equation (4.3) differs from the classical MDS optimization solution in equation (4.1).

4.3.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

The t-Distributed Stochastic Neighbor Embedding (t-SNE) is another non-linear dimensionality reduction technique introduced in [Maaten et al. (2008)], and is a variation of the Stochastic Neighbor Embedding (SNE) [Hinton et al. (2003)].

The t-SNE algorithm can be decomposed in two main steps. Given a set of N high-dimensional objects, it begins by building a probability distribution between any two high-dimensional objects, so that similar objects have a high probability of being picked. For two objects x_i and x_j, it defines the following probabilities using a Gaussian density form:

$$\begin{cases} p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)} \\ p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \end{cases} \tag{4.4}$$

Additionally, the parameters σ_i are set using a bisection method so that the perplexities of all the conditional distributions are equal, the perplexity of a probability distribution being a measurement of how well it predicts the sample.

In the same way, the t-SNE defines a similar probability distribution over any two points (y_i, y_j) in the low-dimensional map, this time using a Student-t distribution:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||y_i - y_k||^2)^{-1}} \tag{4.5}$$

The goal is then to minimize, using gradient descent, the (non-symmetric) Kullback-Leibler divergence between the two distributions with respect to the locations of the points in the low-dimensional map $(y_i)_{i \in \{1,\dots,N\}}$.

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{4.6}$$

It ends up projecting high-dimensional data in a low-dimensional space while retaining both its local and global structure.

In the next section, we see how to use these two methods to visualize and compare the relative descriptive powers of hierarchies.

4.3.3 Using dimensionality reductions techniques to visualize the relative descriptive power of each hierarchy

We would like to visualize the relative contributions of different hierarchical clustering methods constructed upon the same points, i.e. the same regions of a fine partition. We have seen in section 2.4.3 that in such a case, computing Gromov-Hausdorff distances between hierarchies is straightforward. This appears interesting as computing the distance between two hierarchies can give us an idea of their relative contributions, provided that we normalize their values first, as described in section 2.4.4.

Let us consider a set of indexed hierarchies $\boldsymbol{\mathcal{H}} = \{(\mathcal{H}_1, \boldsymbol{\lambda}_1), (\mathcal{H}_2, \boldsymbol{\lambda}_2), \dots, (\mathcal{H}_{|\boldsymbol{\mathcal{H}}|}, \boldsymbol{\lambda}_{|\boldsymbol{\mathcal{H}}|})\}$. Whether it is to apply MDS or t-SNE, we start from a symmetrical matrix D of size $|\boldsymbol{\mathcal{H}}| \times |\boldsymbol{\mathcal{H}}|$ filled with Gromov-Hausdorff distances between pairs of hierarchies:

$$\mathrm{D} = (\mathrm{d}_{ij})_{(i,j) \in \{1,\dots,|\boldsymbol{\mathcal{H}}|\}^2}, \text{ s.t. } \forall (i,j) \in \{1,\dots,|\boldsymbol{\mathcal{H}}|\}^2, \mathrm{d}_{ij} = \mathrm{d}_{\mathrm{GH}}(\mathcal{H}_i, \mathcal{H}_j). \tag{4.7}$$

MDS and t-SNE can then be used for different purposes. If we have computed the inter-hierarchies matrices for a given image, using MDS allows us to visualize the distances between hierarchies for this image. This way, we can study their interrelations for this specific image, determine their complementarity or redundancy, and more generally define patterns between

them. Since we are not using a Euclidean distance but a Gromov-Hausdorff one, we make use of the usual metric MDS and optimize the stress function given in equation 4.3 to project distances between hierarchies from a space of dimension $|\boldsymbol{\mathcal{H}}| \times |\boldsymbol{\mathcal{H}}|$ into a space of dimension 2 or 3. Given a reference hierarchy (for example the trivial hierarchy), this gives us an idea of how the hierarchies distance themselves from one another by looking at how they place themselves relatively to the reference hierarchy on the resulting figure. This way, we can estimate the respective contributions of each hierarchical construction scheme.

In a complementary fashion, one may compute for several images the same hierarchies and corresponding distances matrices, and characterize images this way, which will be the topic of chapter 6. We then make use of t-SNE to project such images representations from a space of dimension $|\boldsymbol{\mathcal{H}}| \times |\boldsymbol{\mathcal{H}}|$ to a space of dimension 2 or 3. This way, we can estimate whether the departure hierarchies set was discriminative enough to separate different types of images.

In the next section, we present different ways to combine hierarchies and make use of the techniques we introduced to study some of their properties.

4.4 Sequential combinations of hierarchies through chaining

4.4.1 Definition

In chapter 3, we have introduced many morphological hierarchical segmentation techniques that can be obtained within a graph-based framework. We briefly remind the reader of the process we follow to generate such hierarchies:

1. Get a fine partition $\mathcal{FS}(\mathbf{I})$ of the image $\mathbf{I}$.
2. Construct a region adjacency graph $\mathcal{G}$, with the regions of $\mathcal{FS}(\mathbf{I})$ being the nodes of the graph, and adjacent regions being connected and valued according to an initial dissimilarity (for example the median value of the gradient along the contour).
3. Compute a minimum spanning tree $\mathcal{MST}$ of $\mathcal{G}$, associated with a dendrogram structure representing the associated single-linkage hierarchical clustering.
4. Using this dendrogram structure, we efficiently compute new valuations for the MST's edges, which leads to a new associated dendrogram and thus to a new hierarchy.

We now draw the reader's attention to the fact that departing from a tree which valuations are those of the initial graph, one obtains a tree with identical structure but different edge valuations. To go further, this new tree can then be used as a departure point for a similar construction but based upon different criteria and parameters. More specifically, at the end of step 3, we have obtained a MST with valuations η_0. After step 4, we obtain a new MST with the same structure but different valuations η_1. This process can be iterated as much as desired, so that it takes in entry a MST with valuations η_k and outputs a MST with the same structure and with valuations η_{k+1}. We call this process *chaining* or *composition* of hierarchies, and denote $\mathcal{H}_2 \circ \mathcal{H}_1$ a chaining of a hierarchy $\mathcal{H}_1$ followed by a hierarchy $\mathcal{H}_2$, and $\mathcal{H}^{(n)} = \underbrace{\mathcal{H} \circ \ldots \circ \mathcal{H}}_{n \text{ times}}$. It can

be seen as a form of *sequential* combination, as in such a process, each hierarchy is built starting from the preceding one. Note that one can iterate the same hierarchy several times, but also chain different hierarchies. This hierarchical chaining process is depicted in figure 4.2.

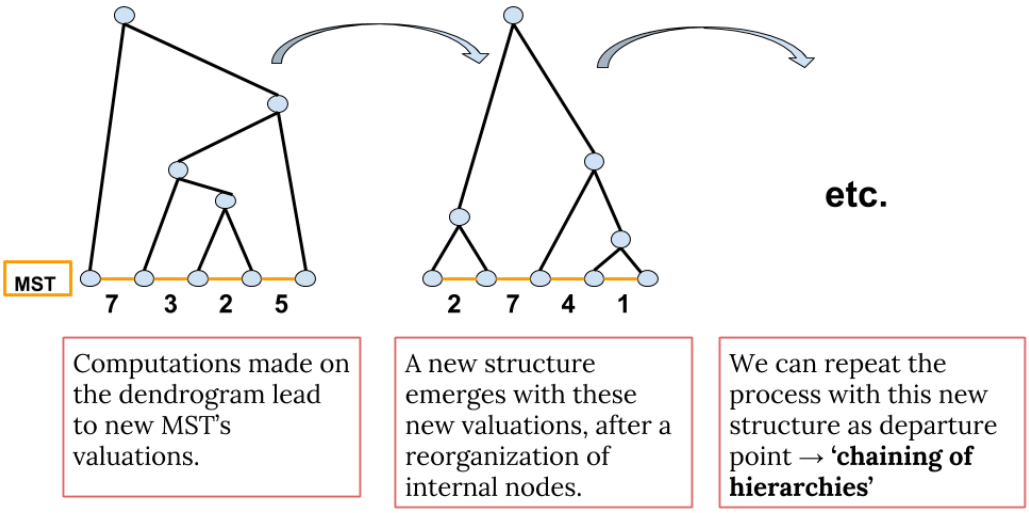


Figure 4.2 – Process of hierarchical chaining.

We would like to study the properties of such chainings, taking as hierarchies the morphological hierarchies we use. Several questions arise.

— Do hierarchies commute when we chain them, i.e. does the order of chaining matters? More formally, for two hierarchies $((\mathcal{H}_1, \boldsymbol{\lambda}_1), (\mathcal{H}_2, \boldsymbol{\lambda}_2))$, do we have $\mathcal{H}_1 \circ \mathcal{H}_2 = \mathcal{H}_2 \circ \mathcal{H}_1$?

— Does a series of hierarchical chainings possibly converge? More formally, for a given hierarchy $\mathcal{H}$ and a distance d between hierarchies, do we have $\lim_{n \to +\infty} d(\mathcal{H}^{(n)}, \mathcal{H}^{(n+1)}) = 0$?

4.4.2 The need for a normalization

When chaining hierarchies, two reasons account for the need to normalize the ultrametric values at each step in the way that was described in section 2.4.4, i.e. to normalize them either with respect to the maximal ultrametric value, or with respect to the number of regions that are in the resulting segmentation for each level of the hierarchy.

1. The first reason is actually valid for all kind of hierarchical combinations: hierarchical clusterings may be of very different natures. Indeed, their ultrametric values can represent different things, as for example the surface-based (resp. volume-based) extinction hierarchy ultrametric values are surfaces (resp. volumes), and the waterfall hierarchy ultrametric values represent stacking orders. Furthermore, hierarchies can have different scales, as typically SWS hierarchies have their probabilistic values in the range $[0, 1]$, whereas other types of hierarchies do not necessarily have ultrametric values in a fixed range. This is why,

in order to properly combine hierarchies, we normalize their ultrametric values. This has the effect of aligning them.

2. Furthermore, in the specific case of the chaining of hierarchies, another reason accounts for this normalization need. In the specific case of SWS hierarchies, the distribution of ultrametric values in the $[0, 1]$ range depends on number of spread markers: its choice does not impact the merging orders, but may lead to ultrametric values being poorly spread along the $[0, 1]$ range. When chaining hierarchies taking into account the ultrametric values of the previous hierarchies such as the volume-based SWS hierarchy, this effect can be amplified and lead to ultrametric values being progressively crushed towards extreme values (0 and 1), until machine precision is no more sufficient to separate them. In such cases, it becomes very difficult to compare hierarchies or measure distances between them. Carefully select a number of spread markers could allow to avoid this phenomenon, but this choice is not straightforward. A simpler solution is then to normalize ultrametric values at each step. The effect of this solution on the histogram of ultrametric values is illustrated in figure 4.3.

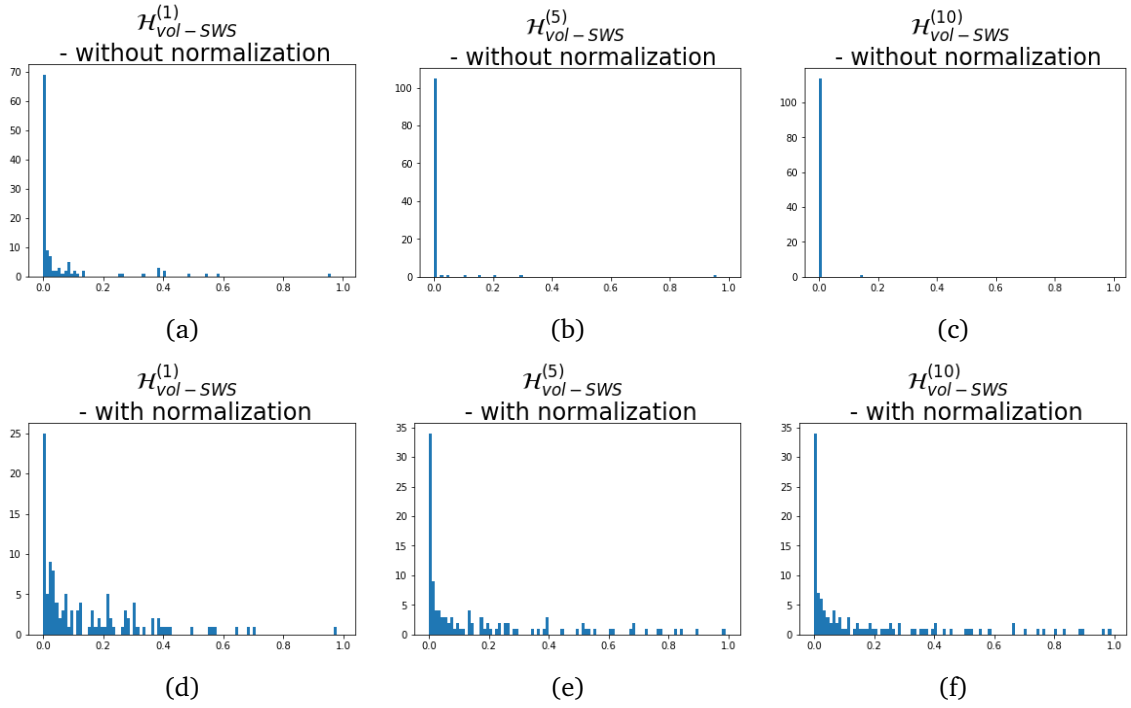
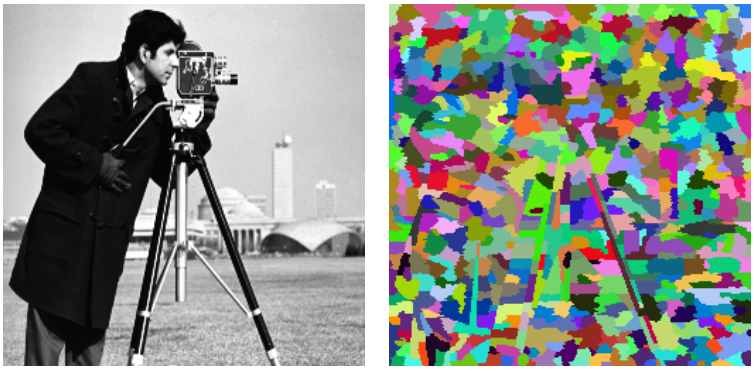


Figure 4.3 – Illustration of the effect of a normalization at each step of the hierarchical chaining process. For a volume-based SWS hierarchy, the computation at a step $t + 1$ depends on the ultrametric values at the step t. (a)(b)(c): Thus, when chaining such hierarchies, ultrametric values tend to be progressively crushed towards extreme values, as one can observe it on the values histograms of chainings of hierarchies $\mathcal{H}^{(1)}_{vol-SWS}$, $\mathcal{H}^{(5)}_{vol-SWS}$ and $\mathcal{H}^{(10)}_{vol-SWS}$. (d)(e)(f): However, when normalizing ultrametric values at each step with respect to the number of levels in the hierarchies (see section 2.4.4), we avoid this undesirable phenomenon.

4.5 Hierarchical chaining analysis

In this section, we want to study the properties of such chainings, and notably determine whether they: (i) commute, i.e. do we have $\mathcal{H}_1 \circ \mathcal{H}_2 = \mathcal{H}_2 \circ \mathcal{H}_1$?, (ii) converge, i.e. are there hierarchies $\mathcal{H}$, for which $\lim_{n\to+\infty} d(\mathcal{H}^{(n)}, \mathcal{H}^{(n+1)}) = 0$?. Since we have now at our disposal a distance between hierarchies, we can address these questions.

For better visualization, we make use of multidimensional scaling (MDS) to project the metric space of hierarchies equipped with the Gromov-Hausdorff distance into a subspace of dimension 2, as explained in section 4.3.3. We can then visualize the effect of the chaining of several type of hierarchies on the same plot, in order to have an idea of the effect of each one. For illustration purposes, we rely on the example image in figure 4.4, but results presented were consistent on many examples.



(a) Image (b) Fine partition using water-pixels

Figure 4.4 – The image and fine partition we build hierarchies upon as example.

4.5.1 Commutation

The first question that arises is to know whether some hierarchical clusterings commute. In order to evaluate such a property, we generate the associated hierarchies for a given image, compute the distances between them and use MDS to visualize the output. It appears that none of the hierarchical schemes that we have implemented commute. We illustrate this on several examples, that can be found in figure 4.5.

On figure 4.5, we can notice how the trivial hierarchy is distant from all others. It may be due to the fact that it is the only hierarchy which ultrametric values are based only on local cues, whereas others make use of a more regional information in their computation. Another interesting aspect is the impact of the waterfall hierarchy in chainings: each chaining containing a waterfall hierarchy step is very distant from others that do not. It may be due to the fact the waterfall hierarchy inherently presents a nested structure with very few levels. So that once the number of significant levels has been drastically decreased in an initial waterfall hierarchy, the subsequent hierarchy cannot retrieve the fineness of structure which has been lost. On the

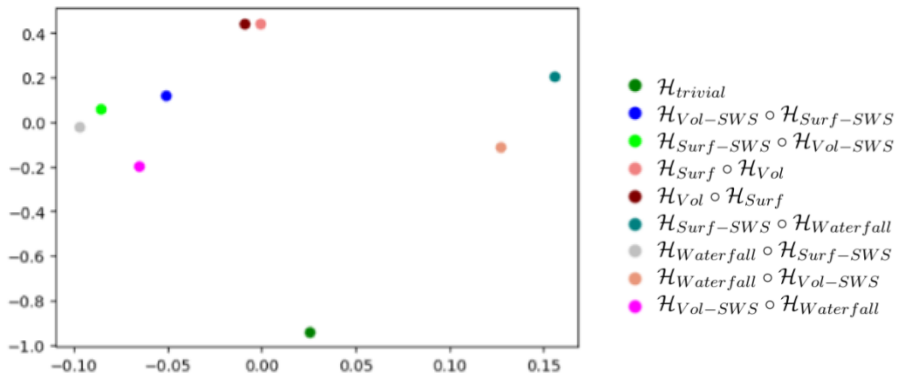


Figure 4.5 – Visualization using MDS of the distances between different combinations of hierarchies. We note that these hierarchies do not commute.

contrary, surface-based and volume-based hierarchies are close, probably due to the fact that they are based on similar increasing attributes.

4.5.2 Convergence

We also want to know whether convergence properties are observable when chaining hierarchical clusterings. Do some of them converge? If yes, do they all do? Does this convergence follow a different trajectory depending on the type of hierarchy considered? Are the different fixed points the same or not?

For example, starting from the image and fine segmentation presented in figure 4.4, we compute the GH-distance between successive chainings of surface-based SWS hierarchies: we observe that this distance is decreasing until reaching zero, as illustrated in figure 4.6. This property has been observed for most hierarchies when they are being chained.

For better visualization, we can also generate a MDS visualization by projecting the Gromov-Hausdorff distances between pairs of hierarchies in a space of dimension 2, as can be seen in figure 4.8. Looking at such results confirm that almost all hierarchical chainings seem to converge toward fixed points in the hierarchical space, i.e. to given hierarchical organizations of all the regions of the fine partition. Furthermore, these fixed points differ depending on the type of hierarchy that is being chained. Examples of visualizations of this phenomenon are provided in figures 4.9 and 4.10. We also illustrate on an example the type of changes in the regions that are being highlighted until convergence of the surface-based SWS hierarchy in figure 4.7 on the same image.

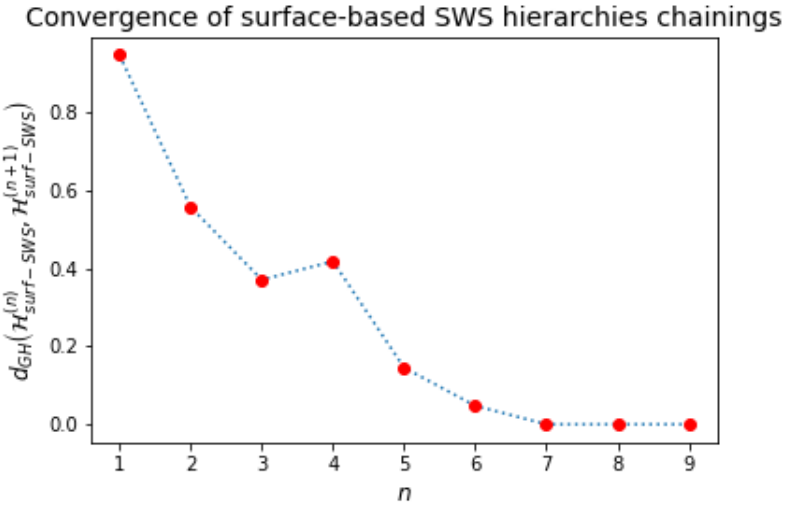
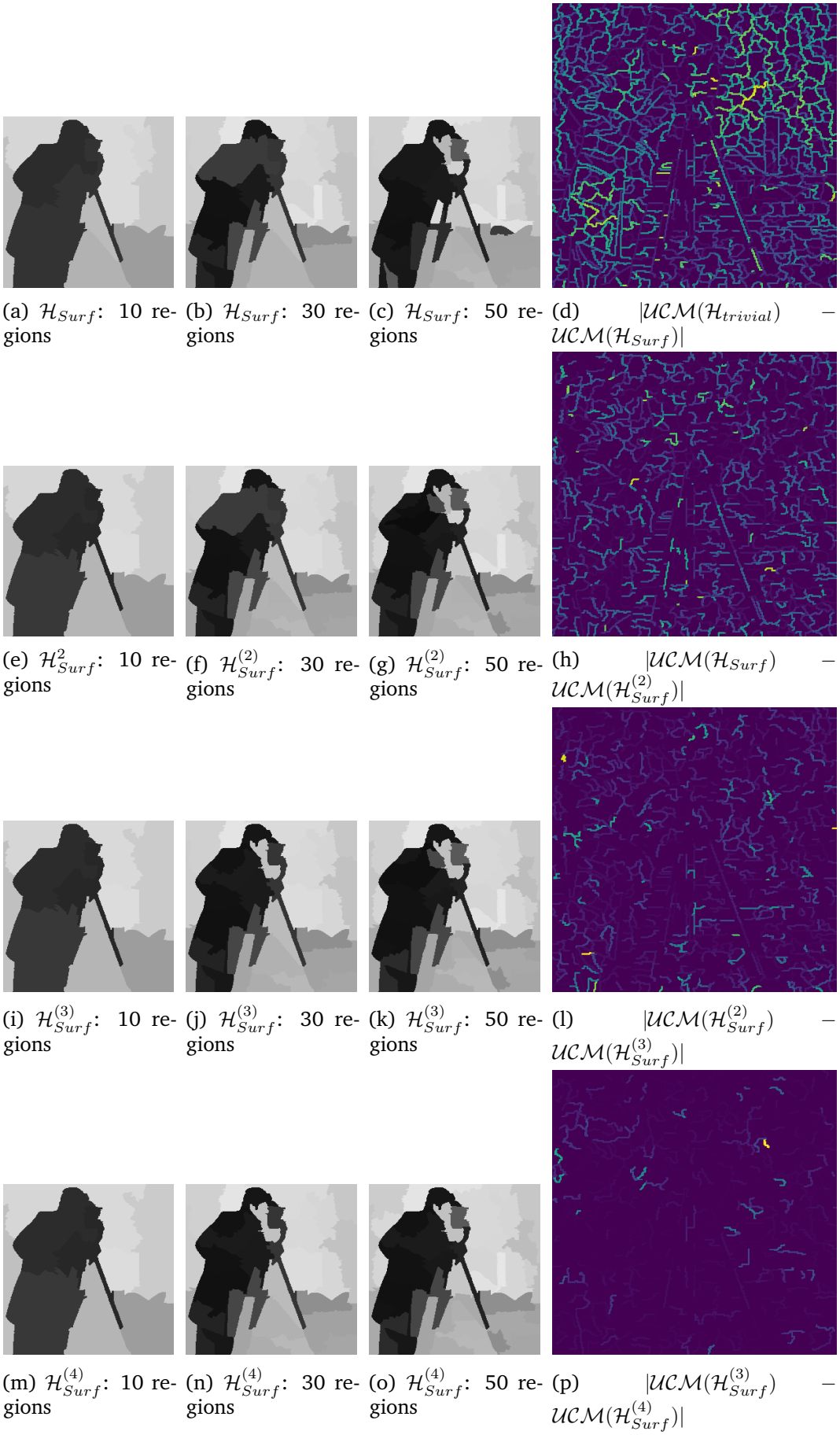


Figure 4.6 – Convergence of chainings of surface-based SWS hierarchies.

A notable exception to this empirical verification of the convergence of hierarchies concerns the chaining of binary-scale climbing hierarchies (see section 3.3.7), which erratic behavior is illustrated in figure 4.11. An intuition to understand its behavior might be that this hierarchy is built depending on a non-homogeneous sum of two terms: one that is increasing with the number of regions (the “regularization” term, sum of contours lengths), and the other that is decreasing with it (the “goodness-of-fit” term, sum of squared differences between the image and the segmentation). On the contrary to others hierarchies considered, modifying the regions proposals has in this case opposite effects on each term, which constitutes a nonrobustness and instability factor.



(a) $\mathcal{H}_{Surf}$: 10 regions (b) $\mathcal{H}_{Surf}$: 30 regions (c) $\mathcal{H}_{Surf}$: 50 regions (d) $|\mathcal{UCM}(\mathcal{H}_{trivial}) - \mathcal{UCM}(\mathcal{H}_{Surf})|$

(e) $\mathcal{H}^2_{Surf}$: 10 regions (f) $\mathcal{H}^{(2)}_{Surf}$: 30 regions (g) $\mathcal{H}^{(2)}_{Surf}$: 50 regions (h) $|\mathcal{UCM}(\mathcal{H}_{Surf}) - \mathcal{UCM}(\mathcal{H}^{(2)}_{Surf})|$

(i) $\mathcal{H}^{(3)}_{Surf}$: 10 regions (j) $\mathcal{H}^{(3)}_{Surf}$: 30 regions (k) $\mathcal{H}^{(3)}_{Surf}$: 50 regions (l) $|\mathcal{UCM}(\mathcal{H}^{(2)}_{Surf}) - \mathcal{UCM}(\mathcal{H}^{(3)}_{Surf})|$

(m) $\mathcal{H}^{(4)}_{Surf}$: 10 regions (n) $\mathcal{H}^{(4)}_{Surf}$: 30 regions (o) $\mathcal{H}^{(4)}_{Surf}$: 50 regions (p) $|\mathcal{UCM}(\mathcal{H}^{(3)}_{Surf}) - \mathcal{UCM}(\mathcal{H}^{(4)}_{Surf})|$

Figure 4.7 – Visualization of the successive highlighted regions when chaining surface-based SWS hierarchies. We also plot the absolute difference between successive saliency maps: they reveal a progressive decreasing of the number of contours that are changing, as well as of the magnitude of this change, from one hierarchy to the next one.

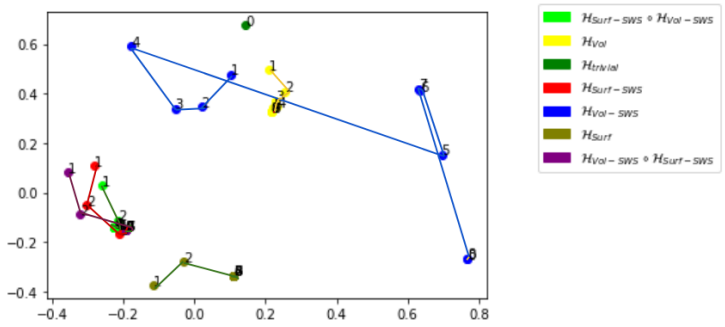
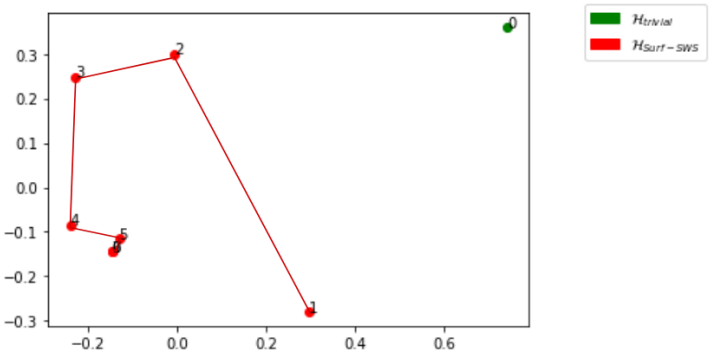
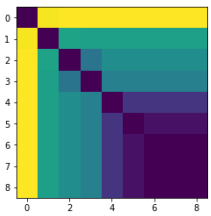


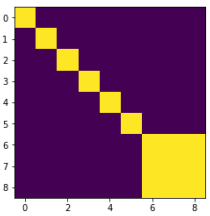
Figure 4.8 – Visualization using MDS of the distances between the different chainings of hierarchical schemes. The reference hierarchy (number 0, in green) used for this MDS visualization is the trivial one. Each dot corresponds to a hierarchy, its color indicate the type of hierarchy that is chained and the associated number the number of iterations of this chaining operation.



(a) MDS for chaining of surface-based SWS.

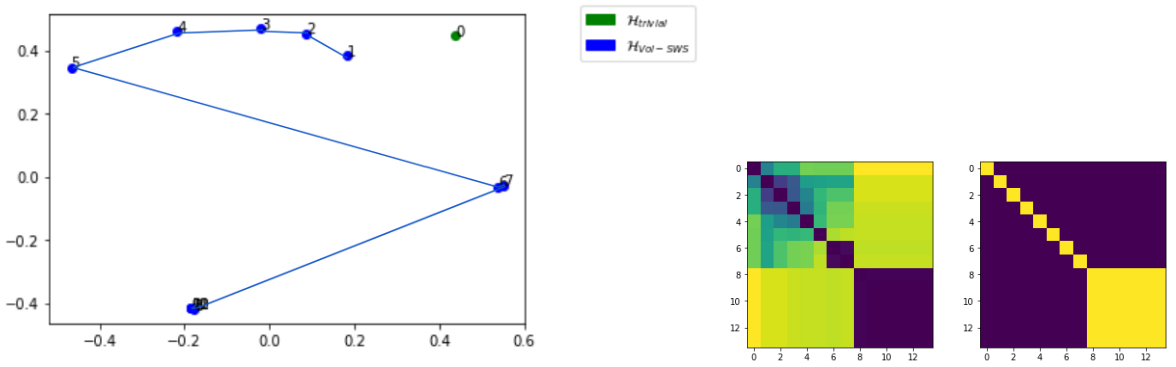


(b) Inter-hierarchies similarities matrix



(c) Null elements of this matrix in yellow

Figure 4.9 – Convergence in the sense of the Gromov-Hausdorff distance when chaining surface-based SWS hierarchies. There is no difference between the cases with or without normalization at each step, since the surface-based SWS does not take into account this information in its computation. The similarities matrix D is presented with null values in yellow. The matrix value in (i, j) corresponds to $\mathrm{d}_{ij} = \mathrm{d}_{\mathrm{GH}}(\mathcal{H}^{(i)}_{Surf-SWS}, \mathcal{H}^{(j)}_{Surf-SWS})$. We can see that at some point the chaining do not modify the output and thus that there is a convergence when operating such a chaining.

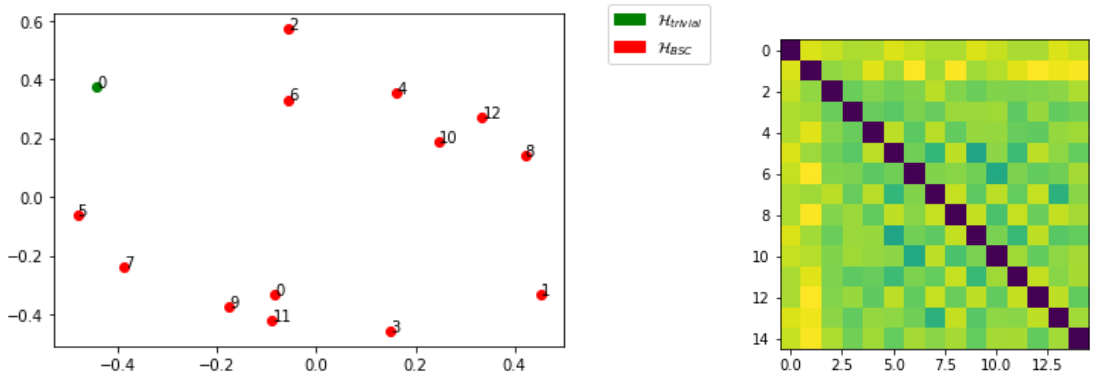


(a) MDS for chaining of volume-based SWS with normalization of saliences at each step.

(b) Inter-hierarchies similarities matrix

(c) Null elements of this matrix in yellow

Figure 4.10 – Convergence in the sense of the Gromov-Hausdorff distance when chaining volume-based SWS hierarchies. On the contrary to the surface-based SWS chaining, here the results differ whether we renormalize at each step the result or not. This stems from the fact that the volume is a composite measure equal to the area multiplied by the contrast, and is thus sensitive to the normalization of values. The similarities matrix D is presented with null values in yellow. The matrix value in (i, j) corresponds to $\mathrm{d}_{ij} = \mathrm{d}_{\mathrm{GH}}(\mathcal{H}^{(i)}_{Vol-SWS}, \mathcal{H}^{(j)}_{Vol-SWS})$. We can see that at some point the chaining do not modify the output and thus that there is a convergence when operating such a chaining.



(a) (b)

Figure 4.11 – (a) Visualization using MDS of the GH distances between chainings of BSC hierarchies. This is the only type of hierarchy (for those implemented), for which no convergence is observed. (b) Corresponding distances matrix D, with lower values in blue. The matrix value in (i, j) corresponds to $\mathrm{d}_{ij} = \mathrm{d}_{\mathrm{GH}}(\mathcal{H}^{(i)}_{BSC}, \mathcal{H}^{(j)}_{BSC})$.

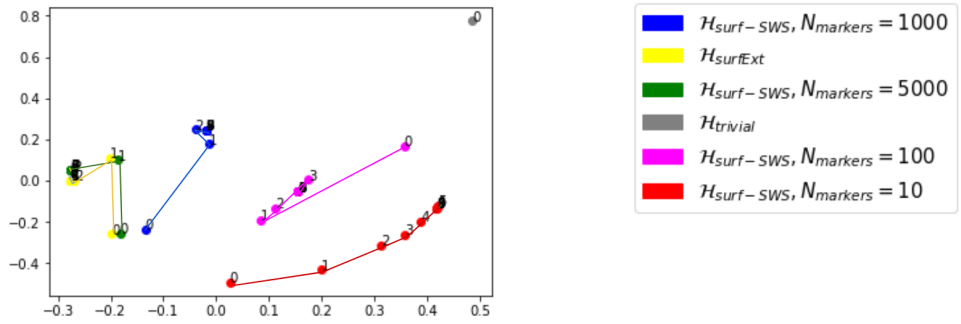
Joint convergence

An interesting empirical result concerns the comparison of the SWS hierarchies and the extinction hierarchies. When the number of drawn markers in the SWS model tends towards infinity, we observe that:

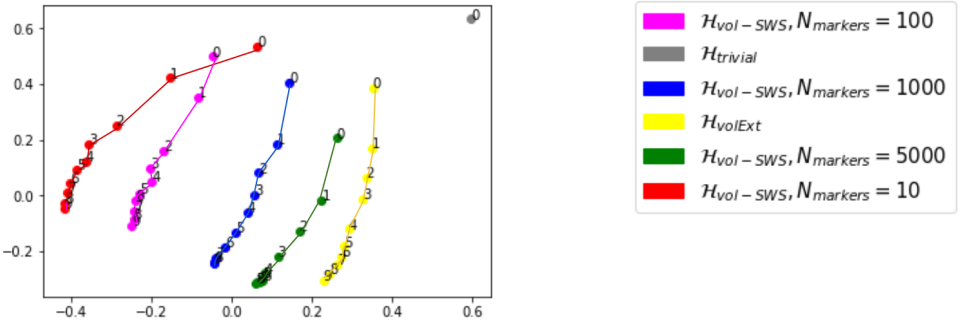
— The surface-based SWS hierarchy and its iterations tend toward the surface-based extinction hierarchy and its iterations.

— The volume-based SWS hierarchy and its iterations tend toward the surface-based extinction hierarchy and its iterations.

These results are illustrated in figures 4.12 and 4.13.



(a)



(b)

Figure 4.12 – (a) When the number of markers increases, the surface-based SWS hierarchy and its chainings tend towards the surface-based extinction hierarchy and its chainings. (b) A similar effect is observable between the volume-based SWS hierarchy and its chainings, and the volume-based extinction and chainings.

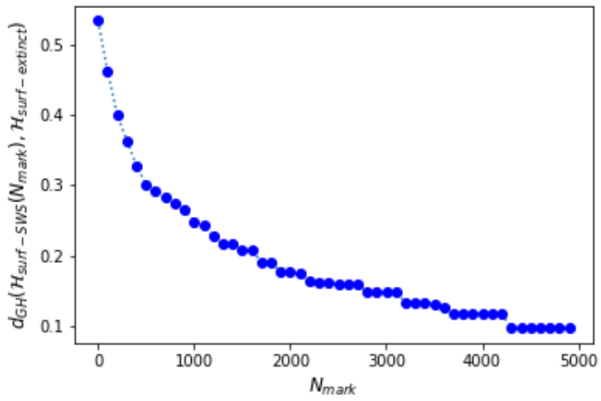


Figure 4.13 – Plot of the evolution of the GH-distance between chainings of the surface-based SWS hierarchy and chainings of the surface-based extinction hierarchy, depending on the number of markers drawn in the SWS model.

Evolution of regions

To better grasp the reasons for this convergence phenomenon, we investigate the precise evolution of regions when doing such chainings. In figure 4.14 we consider, for consecutive hierarchical chainings of surface-based SWS hierarchies, the contour for which the saliency changes the most, and highlight the adjacent regions (in rose and blue) on the corresponding segmentations. A consistent result that we obtain is that the more advanced the chaining, the higher the saliency of the contour for which the saliency change is the bigger. This seems to suggest that the convergence of hierarchical chainings proceeds in a bottom-up way: the modifications in the hierarchical organization of images regions from one step to the next one concern higher and higher levels of the hierarchy, until a stable organization has been reached.

In our specific case, a convergence in the Gromov-Hausdorff sense means that the maximum of the difference between two consecutive chainings saliency maps tends to 0. In order to better understand what it implies, we represent in figure 4.14:

— On the left, the the image representing the difference of saliency maps between successive chainings. The lighter the contour, the more important the difference. We notice that the more chainings are made, the less contours are different.

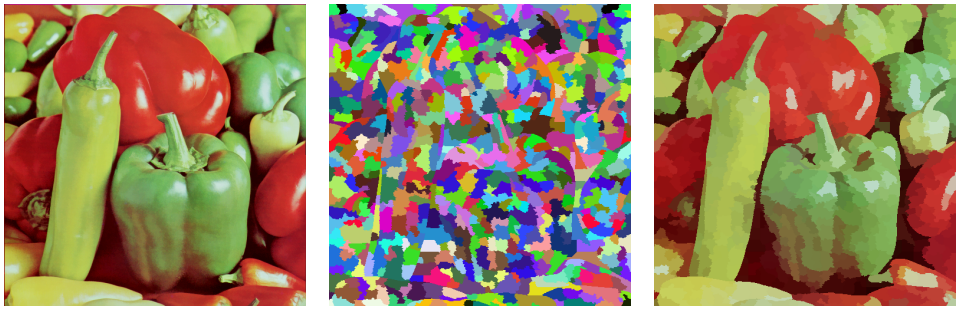
— Furthermore, we can observe each time the precise contour for which this difference is maximal (and which thus defines the GH-distance). We then can represent the hierarchical levels corresponding to the appearance of this contour. We notice that these contours concern low levels of the hierarchy in the beginning, and higher ones when the convergence is almost reached. This results suggests that the progressive convergence of hierarchical chainings operates in a bottom-up fashion: the progressive stabilization of the hierarchical structure begins within its low levels, before extending to higher ones.

Evolution of the information support

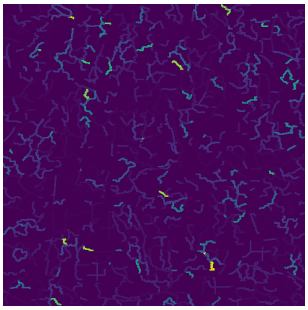
The observation made in the previous section can certainly be linked with a study of the evolution of the information support when chaining hierarchies. We call *information support* (IS) of an image contour the part of the image that has been explored to value this contour. More formally, let us consider a contour corresponding to an e_{pq} of the RAG, with weight λ in a hierarchy $\mathcal{H}$: this contour has two adjacent regions that are ultrametric balls $B(p,\lambda)$ and $B(q,\lambda)$ (these adjacent regions do not correspond to the adjacent regions in the fine mosaic). Thus its information support is equal to:

$$IS(\mathsf{e}_{pq},\mathcal{H}) = B(p,\lambda) \cup B(q,\lambda) \tag{4.8}$$

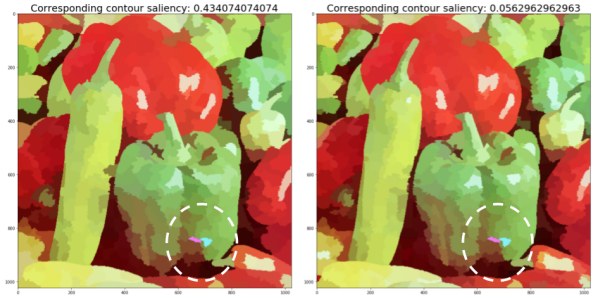
For the first hierarchy, the information support of each contour is limited to the adjacent regions of this contour. But when we chain hierarchies, it progressively gets larger, since at each step



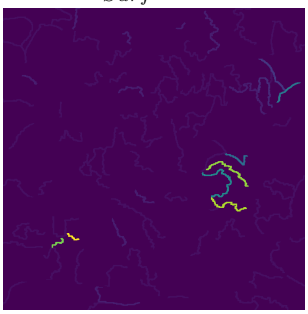
(a) Image (b) Waterpixels image (c) Mosaic image



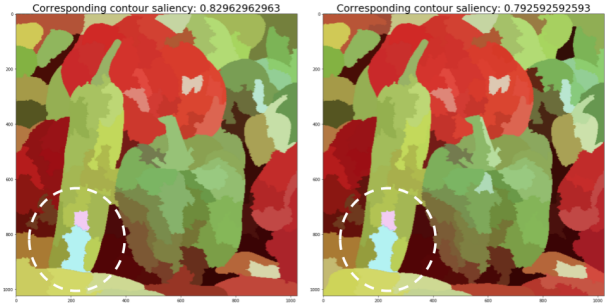
(d) $|\mathcal{UCM}(\mathcal{H}^{(2)}_{Surf}) - \mathcal{UCM}(\mathcal{H}^{(3)}_{Surf})|$



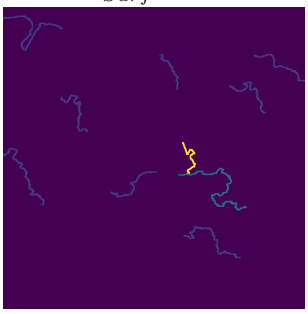
(e)



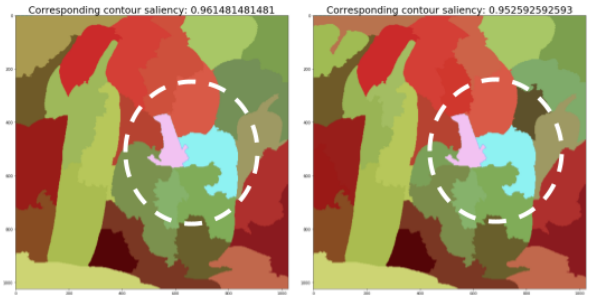
(f) $|\mathcal{UCM}(\mathcal{H}^{(5)}_{Surf}) - \mathcal{UCM}(\mathcal{H}^{(6)}_{Surf})|$



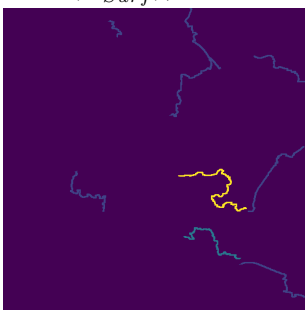
(g)



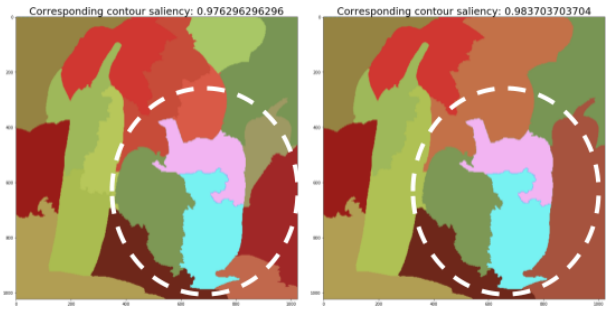
(h) $|\mathcal{UCM}(\mathcal{H}^{(7)}_{Surf}) - \mathcal{UCM}(\mathcal{H}^{(8)}_{Surf})|$



(i)



(j) $|\mathcal{UCM}(\mathcal{H}^{(8)}_{Surf}) - \mathcal{UCM}(\mathcal{H}^{(9)}_{Surf})|$



(k)

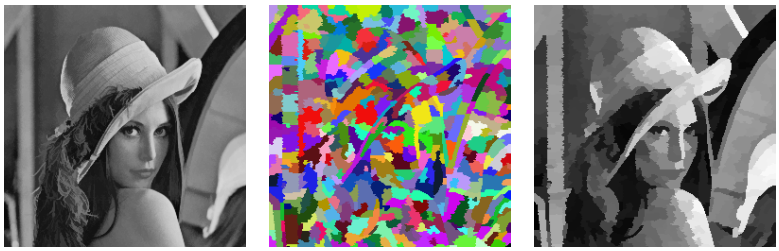
Figure 4.14 – Hierarchical chaining of surface-based SWS hierarchies.

there is a reorganization of the hierarchy, and that by definition the IS is increasing:

$$IS(\mathsf{e}_{pq}, (\mathcal{H}, \ldots, \mathcal{H}^{(n)})) = \bigcup_{k\in\{1,\ldots,n\}} IS(\mathsf{e}_{pq}, \mathcal{H}^{(k)}) \tag{4.9}$$

$$= \bigcup_{k\in\{1,\ldots,n\}} (B_k(p, \lambda_k) \cup B_k(q, \lambda_k)) \tag{4.10}$$

Indeed, at any point a hierarchy is the result of previous chainings, and studying the evolution of this information support for a contour is a way to measure how much information of the image has been explored to value a contour. In figure 4.15, we illustrate it on an example for a given contour. The fact that the mean of the information support over all contours is large also accounts for the robustness of the hierarchies to the chaining effect described in section 2.2.5.

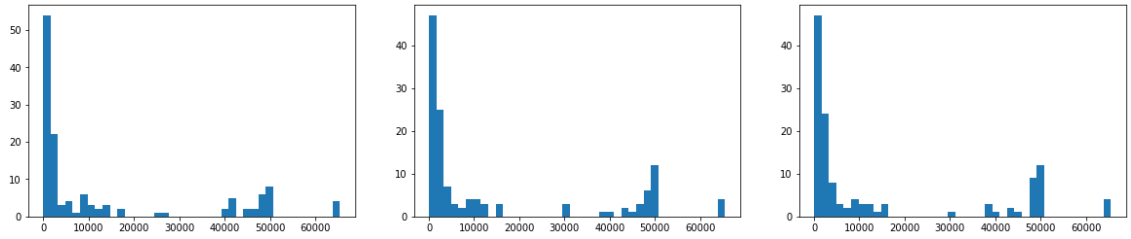


(a) Image (b) Superpixels (c) Mosaic image

(d) $\mathcal{H}_{Surf} \rightarrow \mathcal{H}^{(2)}_{Surf}$ (e) $\mathcal{H}^{(2)}_{Surf} \rightarrow \mathcal{H}^{(3)}_{Surf}$ (f) $\mathcal{H}^{(3)}_{Surf} \rightarrow \mathcal{H}^{(4)}_{Surf}$

Figure 4.15 – Hierarchical chaining of surface-based SWS hierarchies. We represent the evolution of the information support for a given contour. It is increasing until the position of this contour no longer changes in the hierarchy.

However, one must note that the information support is low for most contours when using superpixels. This is due to the fact that a lot of superpixels delineate contours within homogeneous region. As a consequence, such contours disappear quickly in any hierarchy, as they do not have any justification in the image but are solely due to the superpixels construction process. This is why their information support is consistently low when chaining hierarchies. On the other hand, contours that present an interest regarding the attribute highlighted by the hierarchy see their information support grow. This is illustrated in figure 4.16. Experimentally, a stabilization of the IS of a given contour is usually obtained quite fast when chaining hierarchies, in less than 3 chainings.



(a) $IS(\mathcal{H}_{Surf-SWS})$ (b) $IS(\mathcal{H}^{(2)}_{Surf-SWS})$ (c) $IS(\mathcal{H}^{(3)}_{Surf-SWS})$

Figure 4.16 – Evolution of the information support (IS) for the different contours of the image when chaining surface-based SWS hierarchies. While most of them have a very low IS, contours that are specifically highlighted by the hierarchy see their IS grow.

A local optimum of a cost-function defined over the whole hierarchy?

Finally, an interesting property to highlight concerns a link between morphological hierarchical chainings and a cost-function defined over the entire hierarchy.

The Dasgputa cost function has been introduced in ([Dasgupta (2016)]). The author notices that most of the hierarchical clusterings techniques are defined procedurally, which makes the study of their effect complex, since the objective function they optimize is hard to figure out. However, being provided with precise objective functions to optimize might help the development of hierarchical clustering techniques, as it has helped other kinds of data analysis methods such as the classification ones.

Thus, the author defines a cost-function that, given pairwise similarities between data points, assigns a score to any possible hierarchy/dendrogram on those points. For a connected graph $\mathcal{G}$, any hierarchical clustering represented as a dendrogram Δ is valued by:

$$cost_{\mathcal{G}}(\Delta) = \sum_{\{i,j\}\in \mathsf{E}} \eta_{ij} |\text{leaves}(\Delta[i \vee j])| \tag{4.11}$$

With, for any edge $\{i, j\} \in \mathsf{E}$:

— η_{ij} its weight, reflecting locality (with $\eta_{ij} = 0$ if nodes i and j are not neighbors in the graph),

— $|\text{leaves}(\Delta[i \vee j])|$ the number of leaves of the minimal subdendrogram of Δ containing both nodes i and j.

The idea is to find a hierarchical clustering in which similar edges connecting very similar nodes are cut as far down the dendrogram as possible. It is illustrated in figure 4.17. Note that it can easily be obtained with the data structures introduced in section 3.6.3, since finding for two leaves the smallest subdendrogram containing both using them is easy.

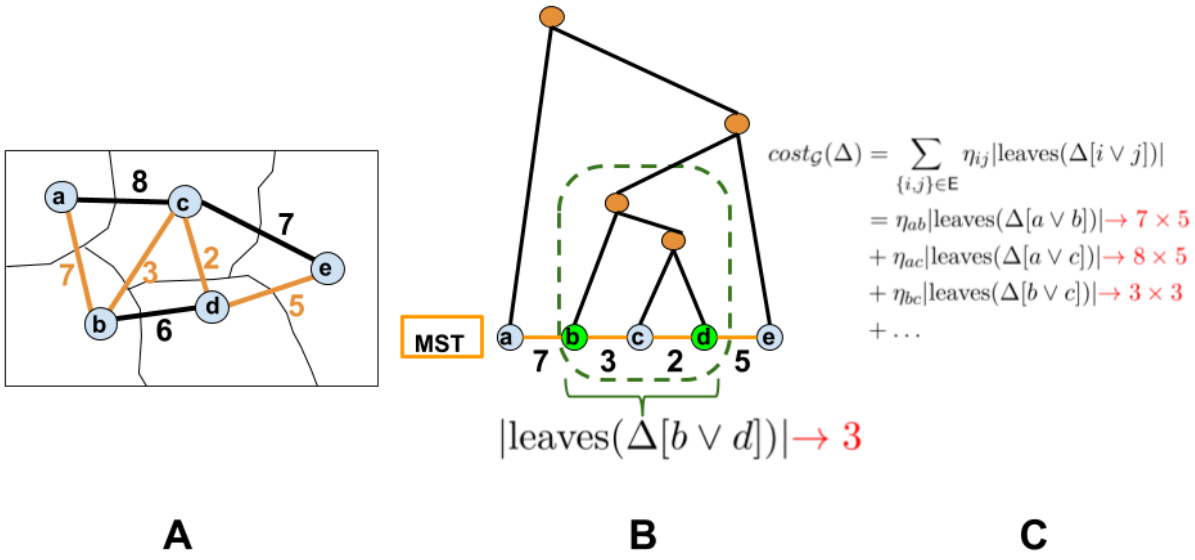


Figure 4.17 – Computation of the Dasgupta cost on an example. **A**: a RAG constructed for a fine partition, and its MST in orange. **B**: for the associated dendrogram, the smaller subdendrogram containing both nodes b and d has 3 leaves, and $\eta_{bd} = 6$. **C**: we can thus compute compute the Dasgupta cost.

We observe that this cost function is systematically decreasing when chaining morphological hierarchies we defined, once again to the notable exception of the BSCH hierarchy. This is illustrated in figure 4.18. This opens a research path to study with more details this link between such a global cost function and hierarchical chaining of incrementally defined hierarchies.

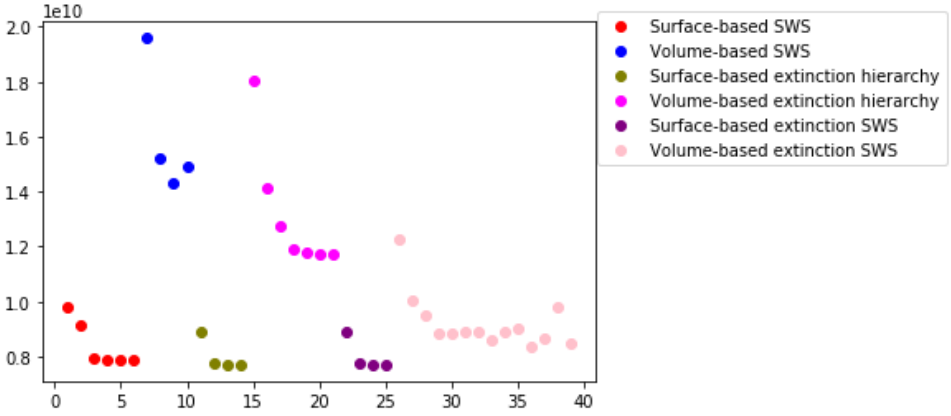


Figure 4.18 – Evolution of the Dasgupta score (see equation (4.11)) when chaining different types of morphological hierarchies until convergence. It seems as if the chainings converge towards local minima of this objective function. This plot was obtained for the cameraman image, but we obtained similar results with many examples.

4.6 Parallel combinations of hierarchies

4.6.1 Introduction

By construction, each hierarchy fuses adjacent regions to make emerge specific structures and characteristics, such as regions with larger areas, contrast, close to some geometrical shapes, etc. We call these hierarchies *pure hierarchies*, by analogy with the pure colors. Yet, most of the images we study admit several of these characteristics as descriptive traits. This is why we would like to describe such mixtures images by combining pure hierarchies.

In sections 4.4 and 4.5, we have presented sequential combinations of hierarchies, corresponding to successive hierarchical constructions, for which each hierarchy was built starting from the preceding one. In this section, we study *parallel* combinations of hierarchies that are obtained by functional combinations of the ultrametrics describing different hierarchies.

4.6.2 General case

When considering two hierarchies $(\mathcal{H}_1, \boldsymbol{\lambda}_1)$ and $(\mathcal{H}_2, \boldsymbol{\lambda}_2)$, their ultrametrics induce a distance between any two points p, q of the graph, respectively $\boldsymbol{\lambda}_1(p,q)$ and $\boldsymbol{\lambda}_2(p,q)$. We can then choose any function $\oplus : \mathbb{R}^2 \to \mathbb{R}$ to obtain a new dissimilarity $\oplus(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)$. We alternatively will also write $\boldsymbol{\lambda}_1 \oplus \boldsymbol{\lambda}_2$ in the following. However, in the general case, this new dissimilarity will no longer be an ultrametric. To obtain an ultrametric from $\oplus(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)$, one can thus compute the subdominant ultrametric $\overline{\oplus(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)}$ associated with this dissimilarity. The exploration of the hierarchical space with such processes has notably been studied in [Santana Maia et al. (2017)].

For example, we can in particular consider the supremum or infimum of hierarchies. We remind the reader that an order relation over the set of hierarchies has been introduced in section 2.4.1: $\mathcal{H}_1 < \mathcal{H}_2$ can be read "$\mathcal{H}_1$ is finer than $\mathcal{H}_2$", and means that $\mathcal{H}_1$ has more regions than $\mathcal{H}_2$ at each level. We have also introduced in section 2.4.2 the supremum and infimum of two hierarchies.

The infimum of two hierarchies $\mathcal{H}_1$ and $\mathcal{H}_2$ is written $\mathcal{H}_1 \wedge \mathcal{H}_2$ or $\mathrm{INF}(\mathcal{H}_1, \mathcal{H}_2)$ and is defined by its ultrametric being the supremum of the ultrametrics of both hierarchies $\boldsymbol{\lambda} = \boldsymbol{\lambda}_1 \vee \boldsymbol{\lambda}_2$ (cf theorem 2.20). Indeed, if $\oplus = \vee$, we have:

$$\forall (p,q,r), \begin{cases} \boldsymbol{\lambda}_1(p,q) \leq \boldsymbol{\lambda}_1(p,r) \vee \boldsymbol{\lambda}_1(r,q) \\ \boldsymbol{\lambda}_2(p,q) \leq \boldsymbol{\lambda}_2(p,r) \vee \boldsymbol{\lambda}_2(r,q) \end{cases} \tag{4.12}$$

Thus:

$$\begin{aligned} \boldsymbol{\lambda}_1 \vee \boldsymbol{\lambda}_2(p,q) &= \boldsymbol{\lambda}_1(p,q) \vee \boldsymbol{\lambda}_2(p,q) && (4.13) \\ &\leq [\boldsymbol{\lambda}_1(p,r) \vee \boldsymbol{\lambda}_1(r,q)] \vee [\boldsymbol{\lambda}_2(p,r) \vee \boldsymbol{\lambda}_2(r,q)] && (4.14) \\ &\leq [\boldsymbol{\lambda}_1(p,r) \vee \boldsymbol{\lambda}_2(p,r)] \vee [\boldsymbol{\lambda}_1(r,q) \vee \boldsymbol{\lambda}_2(r,q)] && (4.15) \end{aligned}$$

Thus the commutativity and associativity of the $\vee$ operator make the computation of the associated ultrametric easy: we just have to assign to each edge the valuation $\boldsymbol{\lambda}_1 \vee \boldsymbol{\lambda}_2$.

However, in most cases for a function $\oplus : \begin{cases} \mathbb{R}^2 \to \mathbb{R} \\ (\lambda_1, \lambda_2) \mapsto \oplus(\lambda_1, \lambda_2) \end{cases}$, we have:

$$\boldsymbol{\lambda}_1 \oplus \boldsymbol{\lambda}_2(p,q) = \boldsymbol{\lambda}_1(p,q) \oplus \boldsymbol{\lambda}_2(p,q) \tag{4.16}$$

$$\leq [\boldsymbol{\lambda}_1(p,r) \vee \boldsymbol{\lambda}_1(r,q)] \oplus [\boldsymbol{\lambda}_2(p,r) \vee \boldsymbol{\lambda}_2(r,q)], \tag{4.17}$$

and as the function $\oplus$ is not necessarily distributive with respect to the function $\vee$, we cannot obtain an ultrametric by simply computing $\boldsymbol{\lambda}_1 \oplus \boldsymbol{\lambda}_2$, and must instead compute the subdominant ultrametric $\overline{\boldsymbol{\lambda}_1 \oplus \boldsymbol{\lambda}_2}$. In particular and for example, the supremum of two hierarchies $\mathcal{H}_1$ and $\mathcal{H}_2$ is written $\mathcal{H}_1 \vee \mathcal{H}_2$ or $\mathrm{SUP}(\mathcal{H}_1, \mathcal{H}_2)$, and is the smallest hierarchy larger than $\mathcal{H}_1$ and $\mathcal{H}_2$. Its ultrametric is $\boldsymbol{\lambda} = \overline{\boldsymbol{\lambda}_1 \wedge \boldsymbol{\lambda}_2} \neq \boldsymbol{\lambda}_1 \wedge \boldsymbol{\lambda}_2$. Such a process has notably been described in [Santana Maia et al. (2017); Cousty et al. (2017)] to compute infimum, supremum and linear combinations of hierarchies.

4.6.3 Simpler parallel combinations between hierarchies built upon the same MST

In our work, we are in a particular situation, as we often build different hierarchies upon the same initial tree. Indeed, starting from a $\mathcal{MST}$ of the RAG $\mathcal{G}$ associated with a fine partition of the image, we generate a new set of valuations on this tree. The resulting ultrametric is the one induced by this MST, and for any pair p, q of nodes of the graphs, its value is equal to the maximal weight of edges on the unique path linking p to q in the MST. In such circumstances, one can easily combine two hierarchies $(\mathcal{H}_1, \boldsymbol{\lambda}_1), (\mathcal{H}_2, \boldsymbol{\lambda}_2)$ for a combination function $\oplus$ verifying a given property, as we shall see hereafter.

Theorem 4.1. *Let us consider two edge-weighted graphs $\mathcal{G}_1 = (\mathsf{V}, \mathsf{E}, \eta_1)$ and $\mathcal{G}_2 = (\mathsf{V}, \mathsf{E}, \eta_2)$ defined over the same set of nodes. Let us also suppose that their respective MST $\mathcal{MST}_1 = (\mathsf{V}, \mathsf{E}_{\mathcal{MST}}, \eta_1)$ and $\mathcal{MST}_2 = (\mathsf{V}, \mathsf{E}_{\mathcal{MST}}, \eta_2)$ have the same structure, i.e. the same nodes and edges but different edge weights.*
Let $\oplus$ be a function such that:

$$\forall (x_1, x_2, y_1, y_2) \in \mathbb{R}_+^4, (x_1 \leq x_2) \text{ and } (y_1 \leq y_2) \Rightarrow \oplus(x_1, y_1) \leq \oplus(x_2, y_2) \tag{4.18}$$

Then the tree $\mathcal{T}_{1,2} = (\mathsf{V}, \mathsf{E}_{\mathcal{MST}}, \oplus(\eta_1, \eta_2))$ is a MST of the graph $\mathcal{G}_{1,2} = (\mathsf{V}, \mathsf{E}, \oplus(\eta_1, \eta_2))$.

Proof. Let e_{pq} be any edge of $\mathcal{G}_1/\mathcal{G}_2$ (they share the same set of edges, but not the same edges weights). By path optimality (cf. theorem 1.28):

$$\forall \mathsf{e}_{pq} \in \pi_{\mathcal{MST}}(p,q) : \eta^1_{st} \leq \eta^1_{pq} \text{ and } \eta^2_{st} \leq \eta^2_{pq}.$$

Thus, if $\oplus$ verifies equation 4.18, we have:

$$\forall \mathsf{e}_{pq} \in \pi_{\mathcal{MST}}(p,q) : \oplus(\eta^1_{st}, \eta^2_{st}) \leq \oplus(\eta^1_{pq}, \eta^2_{pq})$$

Thus by path optimality, the tree $\mathcal{T}_{1,2} = (\mathsf{V}, \mathsf{E}_{\mathcal{MST}}, \oplus(\eta_1, \eta_2))$ is a MST of the graph $\mathcal{G}_{1,2} = (\mathsf{V}, \mathsf{E}, \oplus(\eta_1, \eta_2))$. □

Table 4.1 – Supremum, infimum and mean of ultrametrics.

Type of combination	**Associated ultrametric**
INF $((\mathcal{H}_1, \boldsymbol{\lambda}_1), (\mathcal{H}_2, \boldsymbol{\lambda}_2))$	SUP$(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)$
SUP $((\mathcal{H}_1, \boldsymbol{\lambda}_1), (\mathcal{H}_2, \boldsymbol{\lambda}_2))$	INF$(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)$
MEAN $((\mathcal{H}_1, \boldsymbol{\lambda}_1), (\mathcal{H}_2, \boldsymbol{\lambda}_2))$	$\frac{1}{2}(\boldsymbol{\lambda}_1 + \boldsymbol{\lambda}_2)$

Corrolary 4.2. Let us consider two hierarchies $(\mathcal{H}_1, \boldsymbol{\lambda}_1)$, $(\mathcal{H}_2, \boldsymbol{\lambda}_2)$ defined over the same graph $\mathcal{G} = (\mathsf{V}, \mathsf{E}, \eta)$, and constructed upon two MST sharing the same structure $\mathcal{MST}_1 = (\mathsf{V}, \mathsf{E}_{\mathcal{MST}}, \eta_1)$ and $\mathcal{MST}_2 = (\mathsf{V}, \mathsf{E}_{\mathcal{MST}}, \eta_2)$. Then theorem 4.1 ensures us that for any function $\oplus$ verifying equation 4.18, the MST of $\mathcal{G}_{1,2} = (\mathsf{V}, \mathsf{E}, \overline{\oplus(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)})$ is $\mathcal{T}_{1,2} = (\mathsf{V}, \mathsf{E}_{\mathcal{MST}}, \oplus(\eta_1, \eta_2))$.

Thus, when combining hierarchies with a function $\oplus$ verifying equation 4.18, one can simply apply this function to edge weights of both MST and directly infer the subdominant ultrametric associated with this combination as in equation (2.3):

$$\begin{cases} \forall \mathsf{e}_{pq} \in \mathsf{E}_{\mathcal{MST}}, \overline{\oplus(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)}(p,q) = \oplus(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)(p,q) \\ \forall \mathsf{e}_{pq} \notin \mathsf{E}_{\mathcal{MST}}, \overline{\oplus(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)}(p,q) = \bigvee\{\eta_{st}, \mathsf{e}_{st} \in \pi_{pq} \subset \mathcal{T}_{1,2} = (\mathsf{V}, \mathsf{E}_{\mathcal{MST}}, \oplus(\eta_1, \eta_2))\} \end{cases} \tag{4.19}$$

This procedure is less computationally costly than the one consisting in computing the function $\oplus$ over all edges of the two complete graphs and extracting the subdominant ultrametric consequently, as in [Santana Maia et al. (2017)].

Note that the condition given by equation 4.18 is verified by many two-variables functions that will show interesting: the supremum, infimum, any linear combination with positive coefficients, as well as the logical operators AND and OR between probabilistic variables.

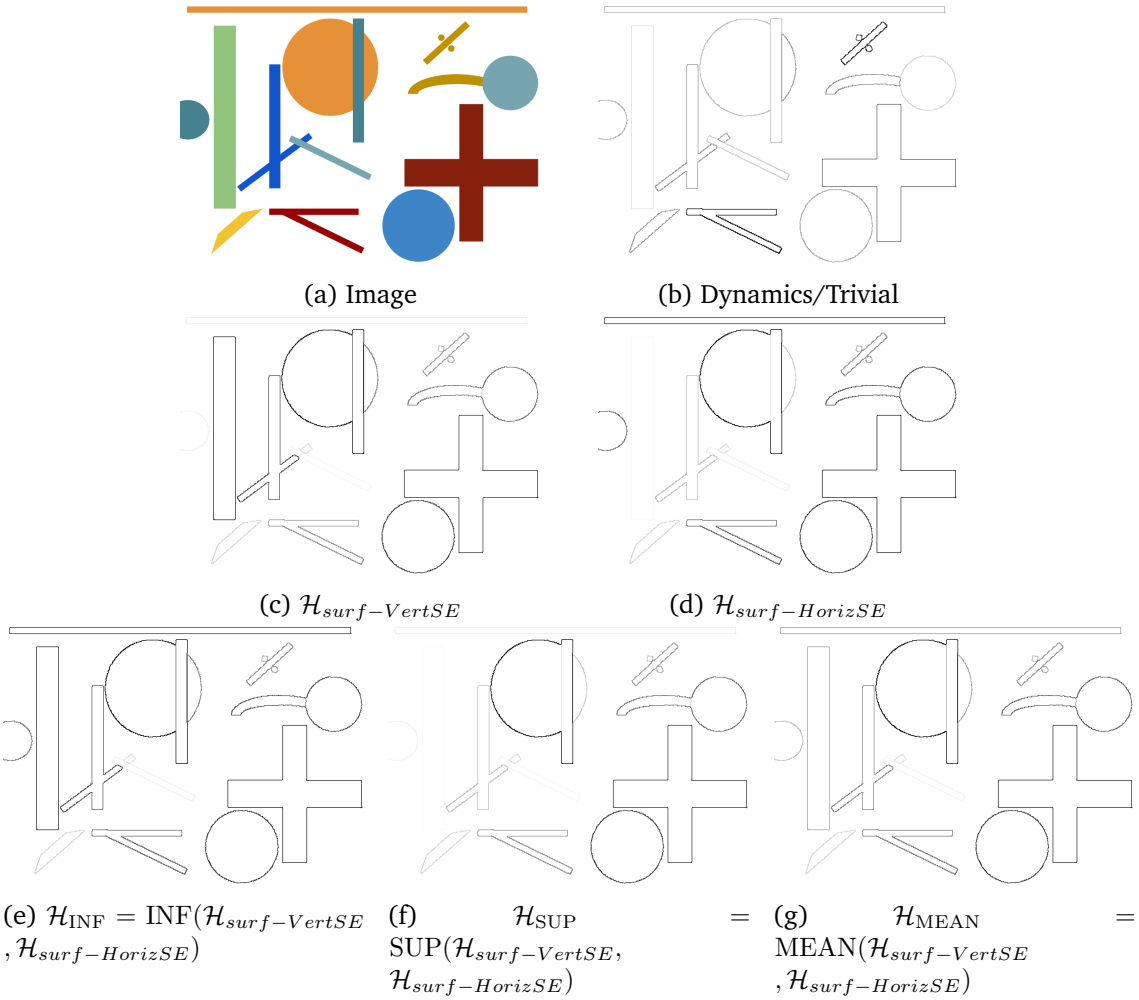
To sum up, our approach consists in choosing a MST from the initial graph and then work with it structure to generate new hierarchies. Its potential weakness resides in the choice of a unique MST to work with for all next steps. In the case where several MST exist, we can affine the choice (lexicographical distance) or work with the union of MST (cf. section 3.3.5). But in exchange, combining hierarchies is often straightforward, and we can easily obtain structurings of the image translating complex yet understandable properties of it. We now present some of these possible combinations.

4.6.4 Supremum, infimum and mean of two hierarchies

Computation

Since we are generating different hierarchies starting from the same MST, and since the SUP, INF and MEAN functions verify equation (4.18) of theorem 4.1, we can simply compute these functions on the MST and infer the subdominant ultrametrics from them. Note that the same property applies to any linear combination (with positive coefficients) of hierarchies.

Interpretations



(a) Image (b) Dynamics/Trivial

(c) $\mathcal{H}_{surf-VertSE}$ (d) $\mathcal{H}_{surf-HorizSE}$

(e) $\mathcal{H}_{\text{INF}} = \text{INF}(\mathcal{H}_{surf-VertSE}, \mathcal{H}_{surf-HorizSE})$ (f) $\mathcal{H}_{\text{SUP}} = \text{SUP}(\mathcal{H}_{surf-VertSE}, \mathcal{H}_{surf-HorizSE})$ (g) $\mathcal{H}_{\text{MEAN}} = \text{MEAN}(\mathcal{H}_{surf-VertSE}, \mathcal{H}_{surf-HorizSE})$

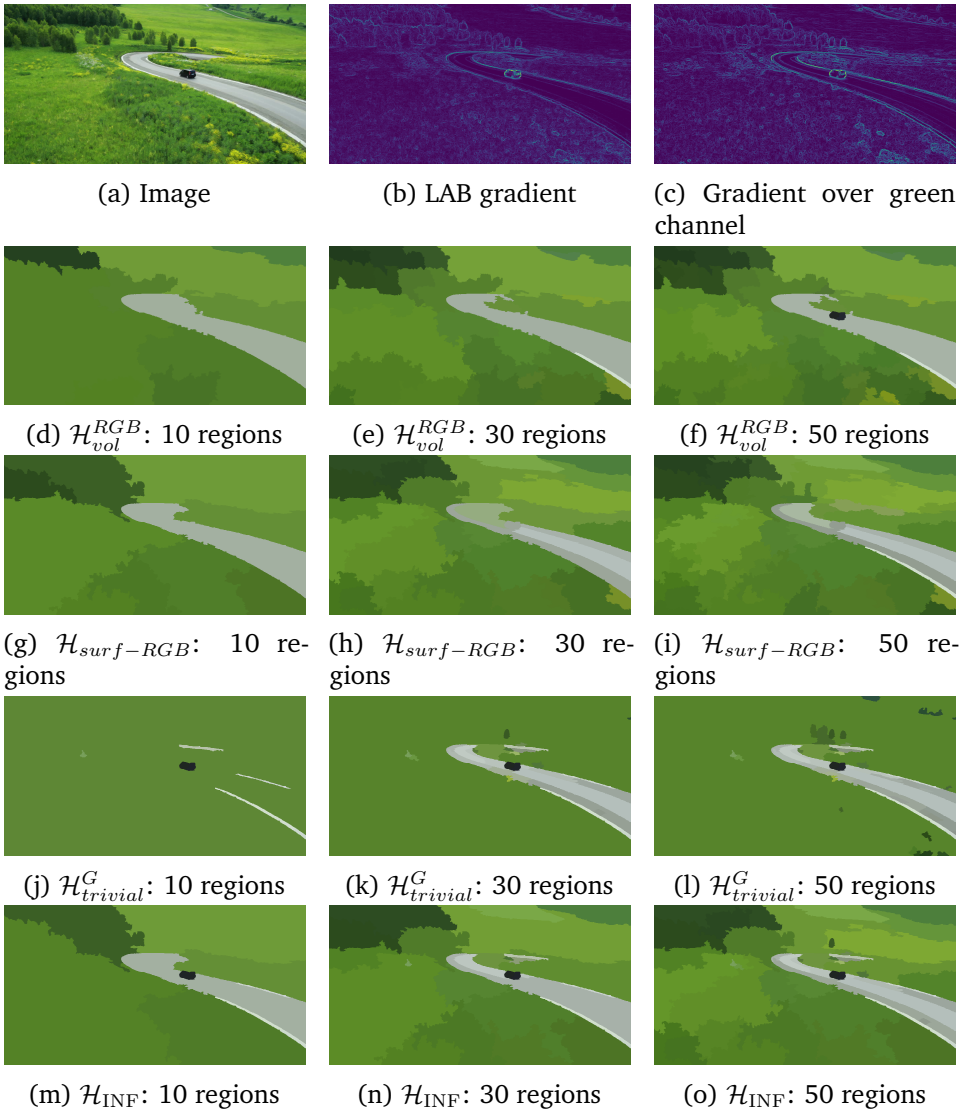
Figure 4.19 – Saliency maps for different hierarchies. One can notice how vertical (resp. horizontal) objects are highlighted in figure (c) (resp. (d)), and that $\mathcal{H}_{\text{AND}}$ ends up highlighting objects both horizontal and vertical.

In practice, computing the supremum of two hierarchies does not lead to interesting results. Indeed, the supremum of two partitions is a partition with contours that are shared by both departure partitions: this is rarely the case, or for very specific frontiers, so it ends up featuring rough frontiers. Taking their infimum or mean can on the other hand help to obtain a good balance between both hierarchies. We illustrate the effects of these operators on a toy example in figure 4.19. We generate, for an image containing different geometric shapes, two surface-based SWS hierarchies with horizontal and vertical markers. These two hierarchies highlight either horizontal or vertical shapes in the image. Whereas the supremum of these hierarchies ends up featuring the drawbacks of both of them, taking their mean and especially their infimum ends up highlighting objects both horizontal and vertical.

We also illustrate the effects of these combinations on a real-life example in figure 4.20. Ideally, on this example, we would like to highlight both the car and the road structure to remain

until the last levels of the output hierarchy. The car is here a small contrasted object, whereas the road is a bigger one with a large surface.

On one hand, one could of course think of using a volume-based hierarchy to obtain a tradeoff between these two properties, which is a classical way of doing it. On the other hand, we test the combination by infimum of: (i) a surface-based hierarchy built on the image using the gradient over the green channel as initial dissimilarity, (ii) a trivial hierarchy built using a LAB gradient as initial dissimilarity. We notice that the combination by infimum is useful in this case as it allows for the car structure, the road and the grass to exist until late stages in the hierarchy whereas the other hierarchies do not manage to do so.



(a) Image (b) LAB gradient (c) Gradient over green channel

(d) $\mathcal{H}_{vol}^{RGB}$: 10 regions (e) $\mathcal{H}_{vol}^{RGB}$: 30 regions (f) $\mathcal{H}_{vol}^{RGB}$: 50 regions

(g) $\mathcal{H}_{surf-RGB}$: 10 regions (h) $\mathcal{H}_{surf-RGB}$: 30 regions (i) $\mathcal{H}_{surf-RGB}$: 50 regions

(j) $\mathcal{H}_{trivial}^{G}$: 10 regions (k) $\mathcal{H}_{trivial}^{G}$: 30 regions (l) $\mathcal{H}_{trivial}^{G}$: 50 regions

(m) $\mathcal{H}_{\text{INF}}$: 10 regions (n) $\mathcal{H}_{\text{INF}}$: 30 regions (o) $\mathcal{H}_{\text{INF}}$: 50 regions

Figure 4.20 – Comparison of 10, 30, 50 first regions for the following hierarchies: (i) $\mathcal{H}_{vol}^{RGB}$: volume-based hierarchy obtained using LAB gradient as initial dissimilarity, (ii) $\mathcal{H}_{surf}^{RGB}$: surface-based hierarchy obtained using LAB gradient as initial dissimilarity, (iii) $\mathcal{H}_{trivial}^{G}$: trivial hierarchy obtained using morphological gradient for the green channel as initial dissimilarity, (iv) $\mathcal{H}_{\text{INF}} = \text{INF}(\mathcal{H}_{surf}^{RGB}, \mathcal{H}_{trivial}^{G})$: infimum of (ii) and (iii).

Table 4.2 – Probabilistic combinations of ultrametrics.

Type of combination	**Associated ultrametric**
$\mathrm{AND}\,((\mathcal{H}_1, \boldsymbol{\lambda}_1), (\mathcal{H}_2, \boldsymbol{\lambda}_2))$	$\boldsymbol{\lambda}_1 \times \boldsymbol{\lambda}_2$
$\mathrm{OR}\,((\mathcal{H}_1, \boldsymbol{\lambda}_1), (\mathcal{H}_2, \boldsymbol{\lambda}_2))$	$\boldsymbol{\lambda}_1 + \boldsymbol{\lambda}_2 - (\boldsymbol{\lambda}_1 \times \boldsymbol{\lambda}_2)$
$\mathrm{NOT}\,((\mathcal{H}, \boldsymbol{\lambda}))$	$1 - \boldsymbol{\lambda}$

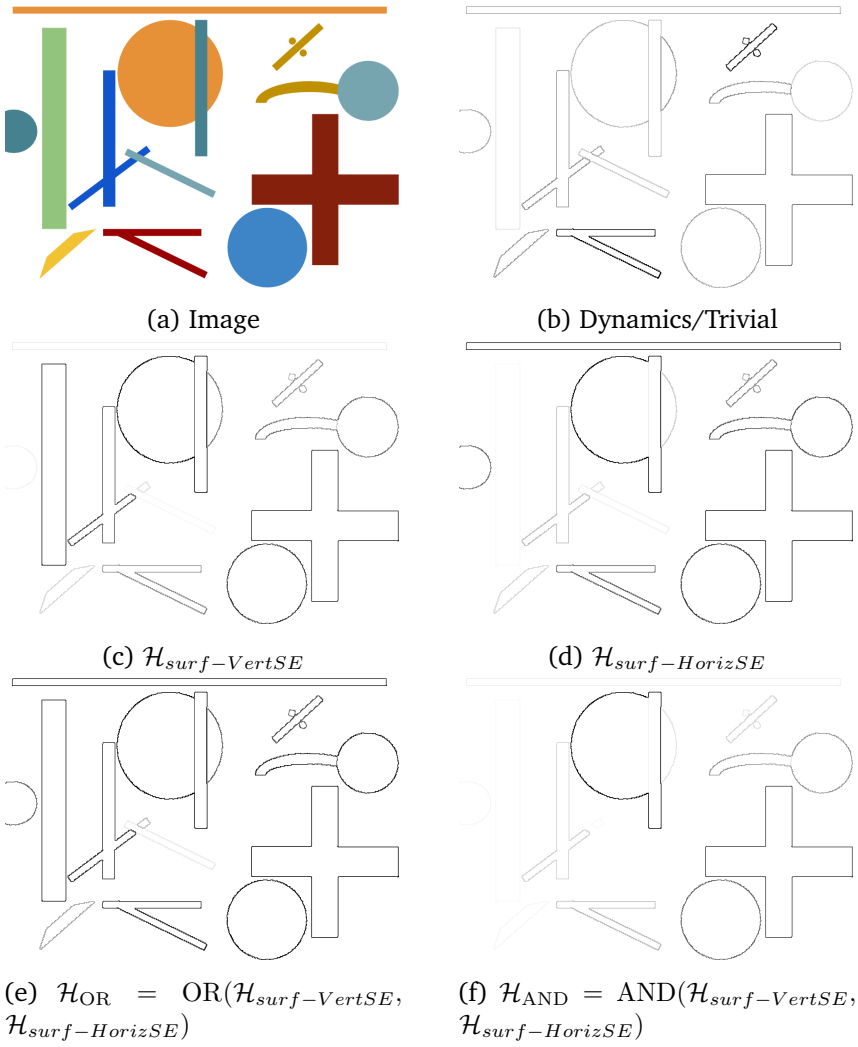
4.6.5 Logical operators of probabilistic ultrametrics

As we have noted it in previous sections, hierarchies have a discriminative power that allows us to discriminate contours in a controlled way. Among the possible hierarchies, the stochastic watershed model presents a versatility that makes it extremely interesting for the characterization of scenes or images: different construction types are possible, supplementary measures can be made on the image to account for the existence of certain forms, the number, size and form of the markers can be controlled, and the choice of the initial dissimilarity (for example depending on the gradient of a given channel) conditions the output hierarchy. In addition, it provides to each contour of the fine partition a probability value. This facilitates their combinations as well as their interpretation.

In this specific case when ultrametric values correspond to probabilities, new possible combinations can be considered through the effect of the boolean operators AND and OR between two ultrametrics associated with two probabilistic events. The AND, OR of two ultrametrics do not directly constitute an ultrametric. Their expressions for two input ultrametrics are given in table 4.2, supposing that the two events are independent. It appears that the two-variable functions that allow for their computation verify the equation (4.18) of theorem 4.1, and we are thus provided with an efficient way to compute them when combining two hierarchies that share the same MST. Note that the NOT operator does not verify this property, and we are thus replaced in the general case of section 4.6.2 to compute the subdominant ultrametric for it.

Using and combining probabilistic hierarchies is interesting for several reasons. First, it is indeed a way to regroup different types of experiments in an homogeneous set of representations and homogeneous measures. Furthermore, starting from a large and homogeneous set of SWS hierarchies, one can combine them in an understandable manner. For example, the AND and OR of two hierarchies have straightforward effects. This is illustrated on a toy example in figure 4.21.

Furthermore, this opens an exploration path for potentially complex combinations using binary logical expressions. We can indeed use all boolean operators and this way build hierarchies combining diverse characteristics, in a way that is more understandable and refined than with the $\mathrm{SUP/INF}$ combinations, and more interpretable than linear combinations such as the mean. We illustrate it in figure 4.22, where we use logical operators between three hierarchies to highlight in a controlled manner a desired type of contours. Let us consider the *Irises* painting by Van Gogh, and let us suppose we want to highlight contours between red and non-red regions, and belittle transitions between blue/non-blue regions. We consider, for the same image,



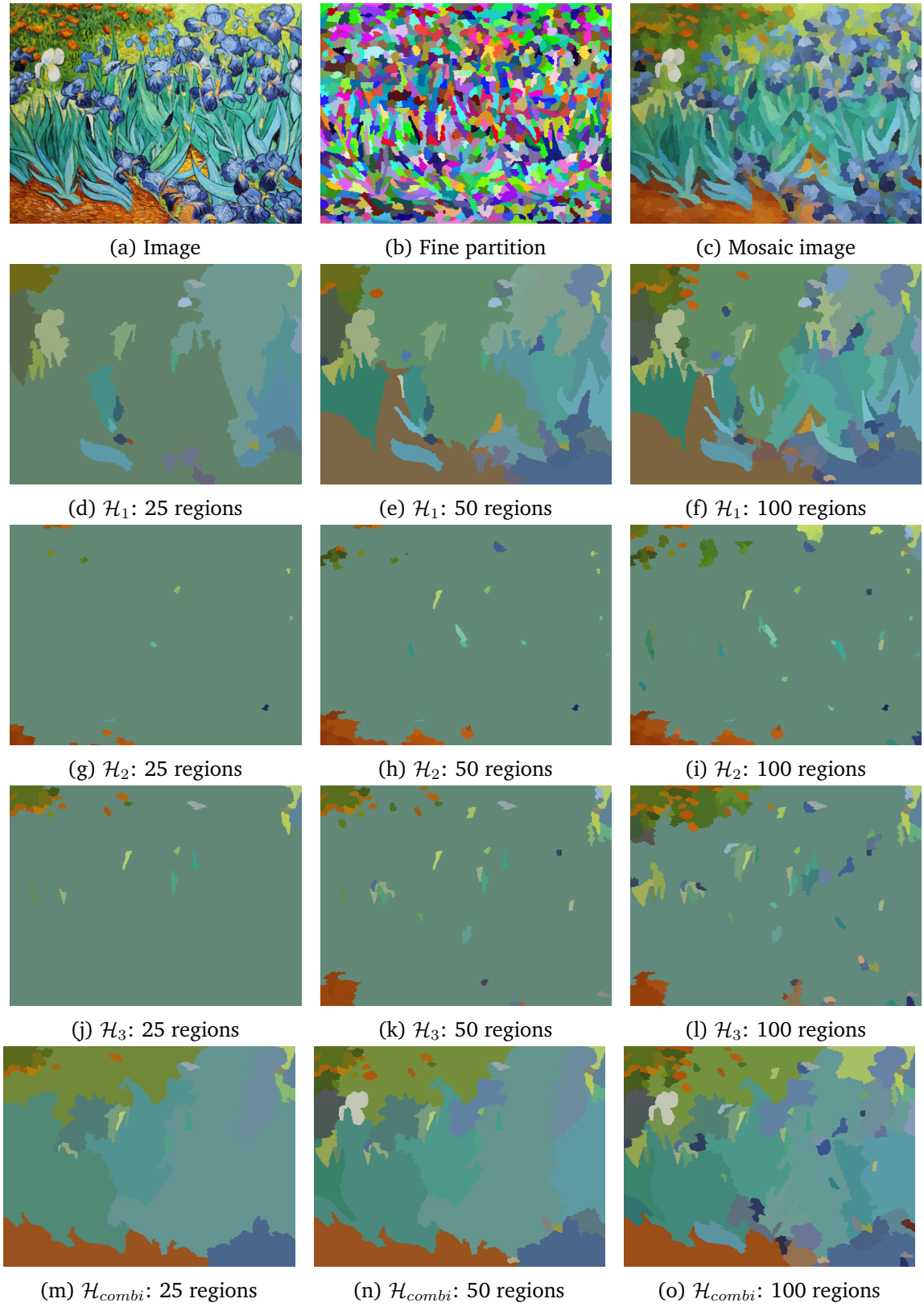
(a) Image (b) Dynamics/Trivial

(c) $\mathcal{H}_{surf-VertSE}$ (d) $\mathcal{H}_{surf-HorizSE}$

(e) $\mathcal{H}_{\text{OR}} = \text{OR}(\mathcal{H}_{surf-VertSE}, \mathcal{H}_{surf-HorizSE})$ (f) $\mathcal{H}_{\text{AND}} = \text{AND}(\mathcal{H}_{surf-VertSE}, \mathcal{H}_{surf-HorizSE})$

Figure 4.21 – Saliency maps for different hierarchies. One can notice how vertical (resp. horizontal) objects are highlighted in figure (c) (resp. (d)), and that $\mathcal{H}_{\text{AND}}$ ends up highlighting objects both horizontal and vertical.

three hierarchies $\mathcal{H}^{RGB}_{Vol-SWS}$, $\mathcal{H}^{R}_{Vol-SWS}$ and $\mathcal{H}^{B}_{Vol-SWS}$ generated using respectively the LAB gradient and the gradients over the red and blue channels. Since these three hierarchies are expressed in terms of probabilities, we can combine them using the logical operators mentioned above. $\mathcal{H}_1 = \text{AND}(\mathcal{H}^{RGB}_{Vol-SWS}, \mathcal{H}^{R}_{Vol-SWS})$ selects contours strong both in the red channel and the (R, G, B) color space. $\mathcal{H}_2 = \text{NOT}(\mathcal{H}^{B}_{Vol-SWS})$ belittle transitions between blue/non-blue regions. Computing the hierarchy $\mathcal{H}_3 = \text{AND}(\mathcal{H}_1, \mathcal{H}_2)$ highlights contours between red/non-red zones and not blue/non-blue zones, but loses the global structures in the images. Finally, the hierarchy $\mathcal{H}_{combi} = \text{OR}(\mathcal{H}_3, \mathcal{H}^{RGB}_{Vol-SWS})$ retains the global structures of the image, with a particular emphasis on red flowers, while for example $\mathcal{H}^{R}_{Vol-SWS}$ or $\mathcal{H}^{RGB}_{Vol-SWS}$ do not manage to do it.



(a) Image (b) Fine partition (c) Mosaic image

(d) $\mathcal{H}_1$: 25 regions (e) $\mathcal{H}_1$: 50 regions (f) $\mathcal{H}_1$: 100 regions

(g) $\mathcal{H}_2$: 25 regions (h) $\mathcal{H}_2$: 50 regions (i) $\mathcal{H}_2$: 100 regions

(j) $\mathcal{H}_3$: 25 regions (k) $\mathcal{H}_3$: 50 regions (l) $\mathcal{H}_3$: 100 regions

(m) $\mathcal{H}_{combi}$: 25 regions (n) $\mathcal{H}_{combi}$: 50 regions (o) $\mathcal{H}_{combi}$: 100 regions

Figure 4.22 – Illustration of a complex combination of probabalistic hierarchies using logical operators. (a) Image. (b) Fine partition. (c) Associated mosaic image. The remaining images represent levels of different hierarchies for 25,50 and 100 regions. (d)(e)(f) $\mathcal{H}_1 = \text{AND}(\mathcal{H}^{RGB}_{Vol-SWS}, \mathcal{H}^{R}_{Vol-SWS})$. (g)(h)(i): $\mathcal{H}_2 = \text{NOT}(\mathcal{H}^{B}_{Vol-SWS})$. (j)(k)(l) $\mathcal{H}_3 = \text{AND}(\mathcal{H}_1, \mathcal{H}_2)$. (m)(n)(o) $\mathcal{H}_{combi} = \text{OR}(\mathcal{H}_3, \mathcal{H}^{RGB}_{Vol-SWS})$.

4.7 Conclusion

In the previous chapters, we have notably presented certain procedures to build hierarchical segmentations of images, the main one being the SWS model, which is very versatile and configurable:

— Markers can be chosen to be punctual or not. In the second case, the choice of the markers can allow the emergence of given shapes in the hierarchical segmentation.

— Different hierarchical schemes can be thought of, each one of them highlighting different kind of contours.

— The distribution of markers in itself can be uniform or non-uniform.

Furthermore, we saw in this chapter that we can also combine all these hierarchical methods in different ways. We have indeed presented several ways to combine hierarchies as long as a methodology to study the properties of these combinations. Note that although these methods have been illustrated on watershed hierarchies, similar approaches can be thought of for any type of hierarchical clustering. The result of it is an extremely large number of possibilities to interrogate the content of images. Several questions then arise.

First, one can wonder how to quantify the extent to which two hierarchical methods give similar results? A first answer proposal has been brought previously, by using Gromov-Hausdorff distances between hierarchies built on the same initial points. A directly related issue is the one of selecting the more pertinent and informative hierarchical clustering methods to characterize images for a given problem. One possibility to do so is, as seen previously, to compute the distances between hierarchical clusterings using a Gromov-Hausdorff distance, and possibly to visualize using multidimensional scaling their relative contributions.

Secondly, and more importantely for practical matters, one may wonder what is a proper way to use these tools wisely. Of course, their use will depend on the class of considered problems. However, a general idea is to use as much prior information we have on the task at hand as possible in the process of building hierarchical segmentations. Our tools allow us to do so in an informative, comprehensive and controlled fashion, as we have seen on several examples.

We note that there are thus two interesting directions in this exploration of possible hierarchies. One can use different construction types for the same input information to characterize it in different ways: for example, a surface-based SWS hierarchy and a volume-based SWS hierarchy do not highlight the same types of contours. At the same time, using the same construction types with complementary information, such as the blue and the red channel of a color image as in the example previously mentioned, also allows to shed a different light on the image. The combinations of these different possibilities, especially using the logical operators AND, OR and NOT for probabilistic hierarchies, then provides us with the possibility to highlight interesting objects with interpretable and coherent tools.

In the next chapters, we propose to make use of this versatility and profusion of images characterizations to better understand the content of images, whether it is for segmentation or classification tasks.

Part II

Applications

Introduction

In part I, we have introduced a variety of hierarchical segmentations methods, showed how we can combine them, and studied their properties and relative contributions. The information retrieved using them can be used in several ways:

1. First and foremost, the computation of a hierarchical segmentation is usually an intermediary step in a segmentation framework, where the goal is generally to derive a unique segmentation of a scene. Hierarchies are then useful to structure the contours in the image and help us, if not to extract directly the exact regions of interest, at least to get a segmentation of the image that is closed to the desired output. This has been studied in many works, such as for example [Gueguen et al. (2013)] in which the authors study how to produce robust segmentations of high-dimensional data, or in [Kiran et al. (2014b)] to find optimal partitions from a given functional.

   Furthermore, obtaining a pertinent segmentation is often difficult for a human operator. It is even harder to do so by automatic means. The automation of a segmentation process for a given task is thus an interesting related issue.

2. Secondly, the contours saliencies we can compute for each image and each hierarchy constitute interesting descriptors to characterize images, be it at a local level or at the image level. This has been studied for example in [Ouzounis et al. (2012)], in which morphological hierarchical segmentations have been studied as a robust way to extract important features of the image, especially for remote sensing applications.

3. Finally, hierarchies are useful to get simplified yet informative representations of images, whether it is for compression or artistic purposes, and allow us to keep only the necessary level of details for a given image and task.

In this part of the thesis, we make use of these versatile tools for several applications, that fall within the categories listed above:

— In chapter 5, we propose an automatic segmentation framework that makes use of SWS hierarchies and their combinations. Our goal is to find the hierarchy and cut level providing the best segmentation result according to a score to evaluate the quality of a segmentation.

— In chapter 6, we show how one can make use of a variety of comprehensive hierarchies as features to characterize images. We prove their discriminative power for the image classification task on several examples.

— In chapter 7, we refine the hierarchical segmentation construction method introduced in part I. The hierarchical segmentation techniques described so far treat the image in an homogeneous manner all over the domain. We show how a slight modification in the SWS model leads to a hierarchical segmentation method that can take into account spatial prior information of all kinds during its construction. We illustrate the goodness of this approach on several examples.

Chapter 5

Learning the Best Combination of Hierarchies for a Domain-specific Segmentation Task

Associated publication:

— [Fehri et al. (2016)] A. Fehri et al. [2016]. « Automatic Selection of Stochastic Watershed Hierarchies ». In: *24th European Signal Processing Conference*. IEEE, pp. 1877–1881

Résumé

Dans ce chapitre, nous proposons une méthode pour apprendre, pour un type d'images données et un score pour évaluer la qualité d'une segmentation, la hiérarchie/combinaison de hiérarchies et le niveau de coupe de cette hiérarchies les plus adaptés audit problème. Pour ce faire, on profite de l'efficience de nos méthodes pour explorer de façon gloutonne l'espace des possibles hiérarchies. On se restreint ici à des combinaisons de type chaînage de longueur deux au plus, mais la méthode est généralisable à tous types de combinaisons, d'images et de scores.

Abstract

In this chapter, we propose a framework to learn, for given type of images and score to evaluate the quality of a segmentation, the hierarchy/combination of hierarchies and the cut level that are the better suited to the problem. To do this, we take advantage of the efficiency of our methods to explore in a greedy way the space of possible hierarchies. We restrict ourselves here to chainings of hierarchies of length at most two, but the method is generalizable to all types of combinations, images and scores.

5.1 Introduction

In chapters 3 and 4, we have presented many different morphological hierarchical clustering methods, as long as ways to combine them. In order to simplify our work and avoid redundancies, we have also studied properties of the space they constitute. These methods can be seen as various ways to structure and interrogate in a controlled way the content of images. The information we retrieve doing so can be used in several ways.

In this section, we propose a workflow to use it for segmentation tasks. The computation of a hierarchical segmentation is usually an intermediary step in a segmentation framework, and the goal is generally to get to a unique segmentation of a scene. Hierarchies are then useful to structure the contours in the image and help us, if not to extract directly the exact regions of interest, at least to get a segmentation of the image that is closed to the desired output. This has been studied in many works, such as for example [Gueguen et al. (2013)] in which the authors study how to produce robust segmentations of high-dimensional data, or in [Kiran et al. (2014b)] to find optimal partitions from a given functional.

An interesting related issue, that we address hereby, is the one of designing a workflow that does this automatically for a given task. Formally, image segmentation is the transformation often described as the partitioning of the image domain into a set of meaningful regions according to some pre-specified criteria. As was briefly mentioned above, it is generally difficult to directly find pertinent contours of an image, and is thus useful to follow a two-steps strategy: first, we produce a hierarchy, and then extract the meaningful contours out of it. In this regard, the hierarchical segmentations techniques presented in chapters 3 and 4 are especially useful.

There are many ways to go from a hierarchical segmentation to a segmentation. Generally, methods in the literature start from a trivial hierarchy and explore it to extract a suited segmentation. We hereby list some of these approaches:

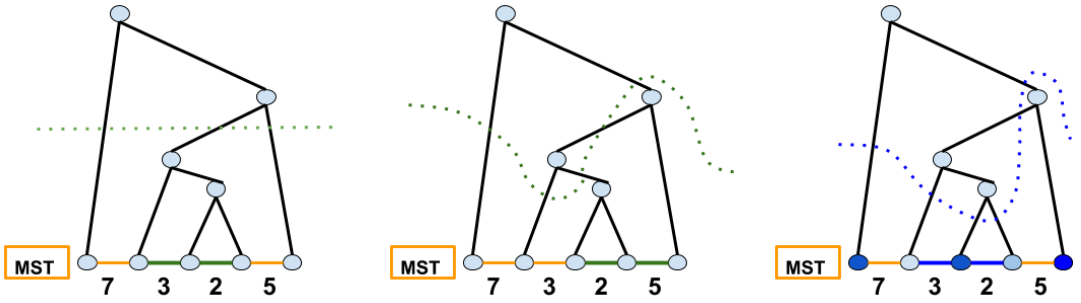
— *Horizontal cut* by thresholding (see figure 5.1(a)), which can be interpreted as a minimization of the diameter of partitions (the diameter of a partition being equal to the maximum diameter of the ultrametric balls that compose it) constrained by the number of regions. This is a classical approach in hierarchical clustering.

— Markers can also be imposed (see figure 5.1(c)), for example to perform interactive segmentation [Meyer and Beucher (1990); Zanoguera et al. (1999)]. Such a *markers-based cut* can be seen as the solution to a problem of minimization of the diameter of the partition constrained by the fact that each region has a marker in it.

— More generally, one can obtain a segmentation from a hierarchy by extracting tiles from different levels of this hierarchy. We call this process to this process an *oblique cut* of the hierarchy.

  For example, in [Drouyer et al. (2017)], the authors propose a new way to select regions of a hierarchical segmentation. To address the problem of stereovision, they make use of a hierarchical segmentation to propose regions to complete and enhance the disparity

map output by any stereovision method. The approach is to go through the hierarchy in a top-down way: based on robust regression models computed for each internal node of the dendrogram associated with the hierarchy, the algorithm decides whether to pursue exploration of the subdendrogram or simply retain this node and stop exploration.
Another similar approach consists in computing an energetic function for each node of the hierarchy and decide based on its value to cut or not the associated edge, thus resulting in a segmentation optimizing an energetic functional. This type of oblique cut has notably been studied for the Mumford-Shah functional, as in [Guigues et al. (2006)]. In [Kiran et al. (2014b)], the authors provide properties for energetic functionals to respect in order to generate hierarchies in linear time.



(a) Horizontal cut (b) Oblique cut (c) Markers-based cut

Figure 5.1 – Different types of cuts to go from a hierarchy of segmentations to a segmentation. The remaining MST edges are highlighted in bold for each example.

However, one may wonder whether the trivial hierarchy is always the best departure point for these techniques aiming at extracting a partition. Amongst those multiple ways to go from a hierarchy to a segmentation, we focus in this section on the simplest one, which consists in doing horizontal cuts of the hierarchy, i.e. in chosing a particular partition of the hierarchy. This can work only if the output hierarchy allows it, i.e. if it takes into account the characteristics of the image. It is a first rough way to explore the potential of each hierarchy, and to measure how favorable a hierarchy is for a given task, as some hierarchies lead us closer to the goal segmentation than others. Of course, to obtain better results, one could use any of the other techniques mentioned above.

It is indeed important to note that there is no such thing as a universal segmentation framework. However, one can learn, for a series of homogeneous images, a suitable segmentation framework. This is what we propose to do using the hierarchical segmentations techniques presented in the previous chapters. The question then is to know if there is an adapted hierarchy/combination of hierarchies for a given task.

We saw in chapter 4 how composing hierarchies in various ways can lead to better or at least complementary results in the sense of the extraction of the more significant part of the images. We now make use of this property and present a workflow to automatically and simultaneously

select the best hierarchy of segmentations and cut-level from a given training set for a given task. This way we have, for a given segmentation task and homogeneous images set, a procedure enabling us to obtain automatically a pertinent hierarchy and cut level in order to get an adequate segmentation.

5.2 Finding a Well-suited Hierarchy and Cut Level from a Training Set

Using the tools and methods presented in chapters 3 and 4, we have now many ways to interrogate our images using different hierarchies or combinations of hierarchies. A segmentation of the image is given by choosing a level of a hierarchy applied to this image. It is hard to know, for a given set of images, which hierarchy and which level of this hierarchy would give good results regarding the segmentation task. However, it is common that images to segment share similar properties, due to their nature or to the tools allowing us to visualize them, as for example cells images in microscopy, or bones and tissues images in radiography. To make easier the obtainment of a satisfying segmentation, it is in our interest to find a hierarchy that takes into account these shared properties amongst the images belonging to the same collection. In a tailor approach, we thus propose a methodology to automatically select a pertinent hierarchy and a good cut level of it for a given set of homogeneous images, so that a suitable segmentation can be obtained for a new image of the same kind without effort.

Let us suppose we have at our disposal a $\texttt{score}(\mathbf{I}, (\mathcal{H}, \lambda))$ to judge the quality of a segmentation $(\mathcal{H}, \lambda)$ obtained for an image $\mathbf{I}$. Note that $(\mathcal{H}, \lambda)$ is the partition obtained after setting the value of the indexed hierarchy $(\mathcal{H}, \boldsymbol{\lambda})$ to λ (corresponding to a horizontal cut of the hierarchy). Thus, we would like to find the best hierarchy and the best cut level λ according to the score evaluated on a training set of images. We call $\texttt{score}(\mathbf{I}, (\mathcal{H}, \boldsymbol{\lambda}))$ the value of the score for an image $\mathbf{I}$, a hierarchy $\mathcal{H}$ and a level λ of the hierarchy. We will expose in the next sections some examples of possible scores to consider.

Formally, let us consider a training set $T = \{\mathbf{I}_1, \ldots, \mathbf{I}_{|T|}\}$ and a set of indexed hierarchies $\boldsymbol{\mathcal{H}} = \{(\mathcal{H}_1, \boldsymbol{\lambda}_1), (\mathcal{H}_2, \boldsymbol{\lambda}_2), \ldots, (\mathcal{H}_{|\boldsymbol{\mathcal{H}}|}, \boldsymbol{\lambda}_{|\boldsymbol{\mathcal{H}}|})\}$.

For any image, there is a best hierarchy and cut level that minimizes the score on this image, that we call *oracle*:

$$(\mathcal{H}^{\texttt{oracle}}, \lambda^{\texttt{oracle}}) := \underset{(\mathcal{H}, \lambda \in \boldsymbol{\lambda}) \in \boldsymbol{\mathcal{H}}}{\arg\min}\ \texttt{score}(\mathbf{I}, (\mathcal{H}, \lambda)). \tag{5.1}$$

Let us consider a set of homegeneous images, that we subdivide into training and testing subsets, and a set of indexed hierarchies $\boldsymbol{\mathcal{H}}$ (possibly composition of hierarchies as in chapter 4). We take advantage of the low computational cost of our approach (only involving updates in the MST) to find the optimal hierarchy in (5.2) by an exhaustive search on the training subset. During the training phase, we are interested in finding the hierarchy $\mathcal{H}$ and cut level λ that

minimize the score on average over the whole set, i.e.,

$$(\mathcal{H}^*, \lambda^*) := \underset{(\mathcal{H}, \lambda \in \boldsymbol{\lambda}) \in \boldsymbol{\mathcal{H}}}{\arg\min} \sum_{i=1}^{|T|} \texttt{score}(\mathbf{I}_i, (\mathcal{H}, \lambda)). \tag{5.2}$$

We call this learned hierarchy the *model* hierarchy.

To sum up, we follow a two-steps procedure, given a segmentation score and a set of homogeneous images :

1. For each image, we extract a wide variety of structured contours information using morphological hierarchies.
2. We select the best hierarchical segmentation and cut level among all possible ones using a greedy feedforward search.

To test the pertinence of this learned model, we compare its result, on each image of the test set, with the oracle model computed for this image.

One can say we have effectively found a good model hierarchy for the set of images if the difference between the scores obtained for the model (equation 5.2) and the oracle (equation 5.1) is on average low on the test subset.

5.3 Experimental Results

In a first approach we consider here, as illustrated in figure 5.2, all chainings of hierarchies (introduced in section 4.4) up to depth two of the following SWS hierarchies: watershed hierarchy (or gradient based), surface-based, volume-based, surface-based after erosion and volume-based after erosion. This corresponds to the work published in [Fehri et al. (2016)]. However, note that this that this approach is generic and can be adapted to any type of hierarchy. Notably, we could test similar approaches with logical combinations of probabilistic hierarchies.

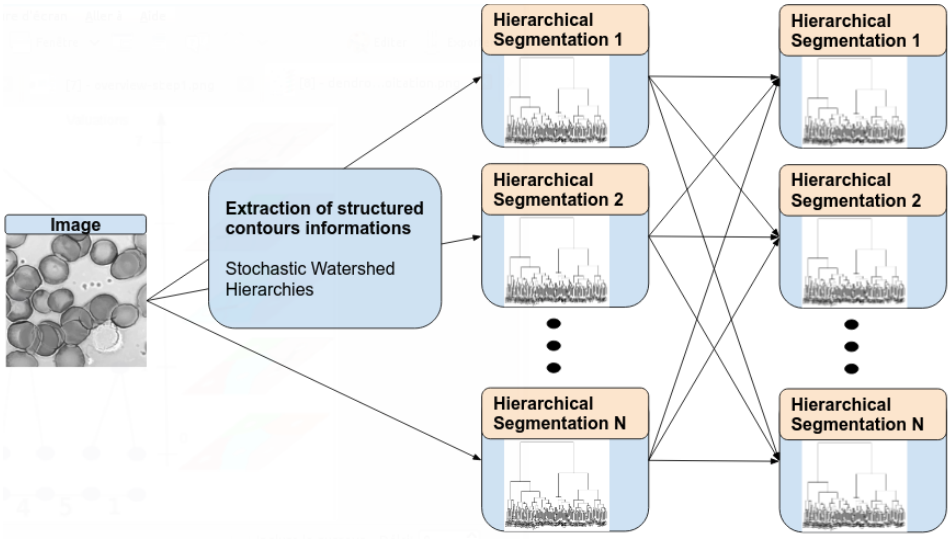


Figure 5.2 – Exhaustive search illustration. At each step, we keep in memory the hierarchy and cut level that gives the best result. We stop the search if the score stops to get better.

5.3.1 Type of Scores

The methodology proposed in section 5.2 is furthermore suitable for any score that we want to minimize (or maximize) in order to get a good segmentation. To test this model, we use two different scores.

Mumford-Shah

The first score used is a Mumford-Shah score [Mumford et al. (1989); Pock et al. (2009)], so that the problem we want to solve is an energy minimization problem. This score contains two terms, a data fidelity term and a regularization term: by climbing in the hierarchy towards coarser levels, the value of the first term increases and the value of the second one decreases. Both terms are linked by a scale parameter. It has this form:

$$\mathtt{MS}(\pi = (\mathbf{I}, \mathcal{H}, \lambda)) = \sum_{\mathsf{R}_i \in \pi} \mathtt{var}(\mathsf{R}_i) + sC(\pi), \tag{5.3}$$

where $\mathtt{var}(\mathsf{R}_i)$ represents the total variance of the image in the region R_i of the partition $\pi = (\mathbf{I}, \mathcal{H}, \lambda)$, C_π represents the length of the contours present in the partition π, and s is a scale parameter that allows to have a trade-off between data fidelity and a simplification of the image.

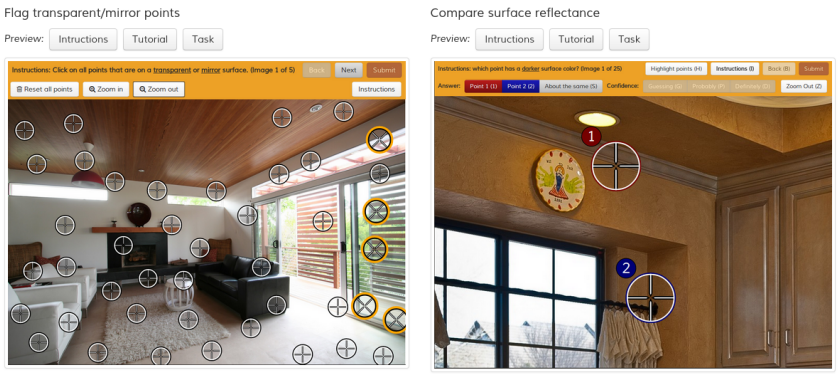
Weighted Human Disagreement Rate (WHDR)

The second score that we used is a metric called "weighted human disagreement rate"(WHDR), introduced in [Bell et al. (2014)] to evaluate intrinsic image decomposition results, aiming at separating images into reflectance and shading layers. This metric is associated with the large-scale public database, Intrinsic Images in the Wild (IIW), built in [Bell et al. (2014)], and composed of 5230 manually annotated images of complex real indoor scenes.

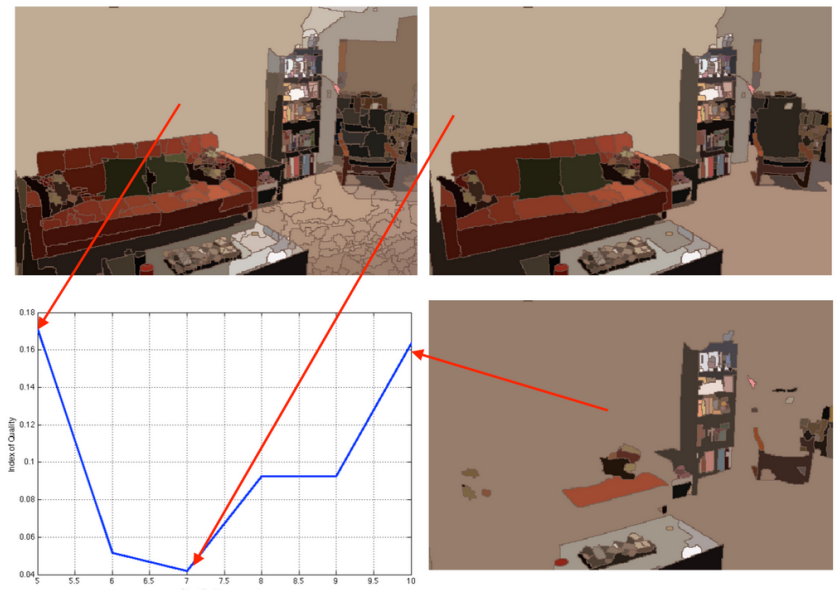
A set of Poisson-disks-sampled points are chosen in each image and these points are connected by edges. Between every pair of connected points, users were invited to evaluate which point has a darker surface color or if they have the same level of brightness. Then the WHDR measures the level of agreement between the judgements made by algorithms being evaluated and those of humans. The cut of the hierarchy allows us to obtain a kind of reflectance image for each test image, by giving to each region of the subsequent partition its mean value in the original image. We can thus use this score to evaluate it. The WHDR varies between 0 and 1, being close to 0 when the reflectance image is consistent with human judgment, and close to 1 otherwise. The goal is then to minimize it. Illustrations regarding WHDR can be found in figure 5.3.

5.3.2 Results

In a first approach, we tested our strategy with a set of cells images for the Mumford-Shah score, and with homogeneous subsets of images from the IIW database, of bedrooms, bathrooms and people. Some visual results can be found in figure.5.4 for the Mumford-Shah score, and in figures 5.5, 5.6 and 5.7 for the WHDR score.



(a) Ground truth creation.



(b) Visualization of WHDR score evolution with image simplification.

Figure 5.3 – Weighted human disagreement rate [Bell et al. (2014)]. **(a)** A ground truth is created using online crowdsourcing tools, translating a human estimation of the continuity of color perception. Users are asked, for random pairs of nearby points: do these points have the same color? Transparrent and mirror surfaces are put aside, as they have particular reflectance properties. **(b)** The WHDR score varies between 0 and 1, being close to 0 when the reflectance image is consistent with human judgment, and close to 1 otherwise.

For each set of images, we train our system on a subset to learn the best hierarchy among any hierarchy or combination of two hierarchies presented before, which provides us with a model hierarchy. Then, for each image of the test subset, we compute the optimal hierarchy for this precise image, that is the oracle hierarchy, the score attached to it, as long as the score given by the model hierarchy on this test image. A summary of the results for the WHDR score is given in Table 5.1.

Database	μ(**WHDR**$^{\texttt{oracle}}$)	μ(**WHDR**$^{\texttt{model}}$)	μ(`error`)	σ(`error`)
Bathrooms	0.154	0.178	0.024	0.025
People	0.133	0.282	0.148	0.093
Bedrooms	0.125	0.237	0.112	0.107

Table 5.1 – Mean and standard deviation of the error between `oracle` and `model`, i.e. the difference between WHDR$^{\texttt{model}}$ and WHDR$^{\texttt{oracle}}$, and averages of the scores for the oracle and model for the different test databases and the WHDR measure.

Furthermore, we can have insights about why a hierarchy has been chosen for a given set of images, and some qualitative remarks can be done as the hierarchies used have understandable effects. For the cells images, the model hierarchy found by the algorithm is a chaining of a volume-based SWS followed by a surface-based SWS. We can interpret it as a first step eliminating non-pertinent objects with a trade-off between area and gradient, and a second pass emphasizing the cells based on their surface, since it is often of the same order. For the IIW images, the hierarchy selected is a composition of volume-based SWS, with different sizes and orientations for structuring elements depending of the dataset. One can interpret that the type of objects usually present in the image differs regarding the scenes, and thus the adaptive hierarchies depend on the shapes found in the images. surface-based SWS are not very pertinent here since there is a wide variety of pertinent objects of different sizes in the images.

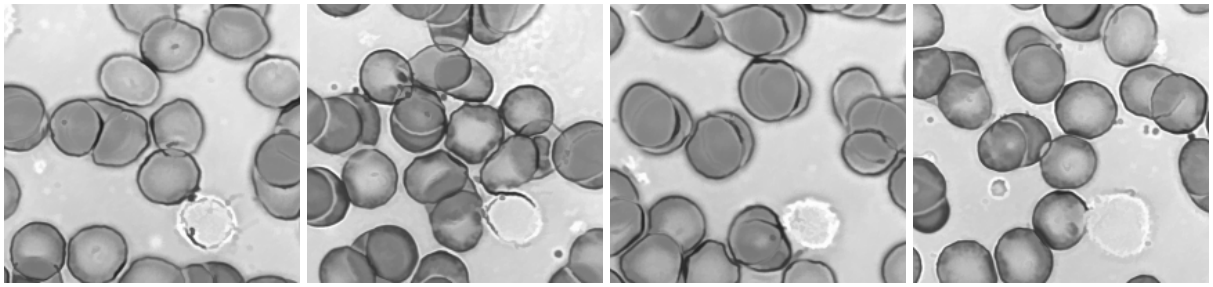
5.4 Conclusions

In this chapter we have presented a novel approach to use the composition of hierarchies of segmentations in a segmentation workflow. The workflow has been evaluated for a difficult task: the obtention of the best hierarchy and cut to perform image simplification given an evaluation score. To go further, several enhancements of the system are conceivable.

First, one could use the analysis tools introduced in chapter 4 to select, for a specific type of images, a set of complementary hierarchies, i.e. hierarchies that properly reflect the image characteristics with being redundant.

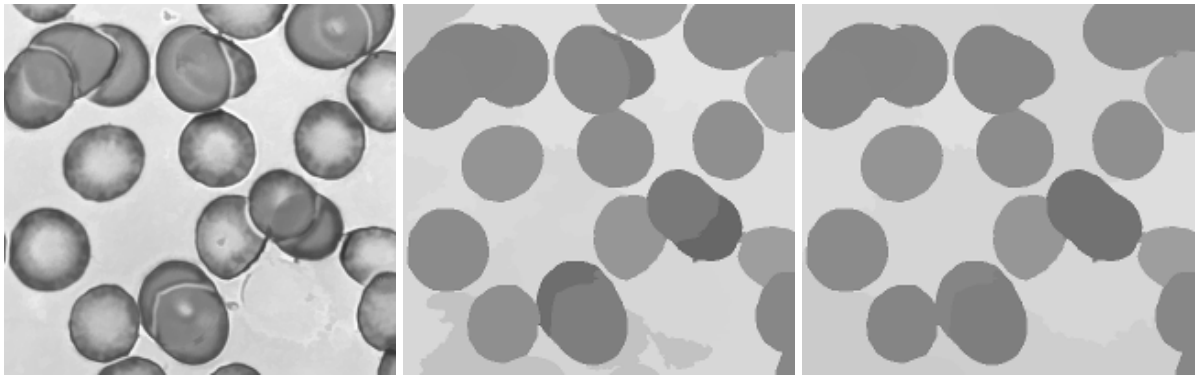
Furthermore, one could use chainings of hierarchies longer than two, or conceive a similar approach with other types of combinations, notably logical combinations of probabilistic hierarchies introduced in section 4.6.5. The first used scores here may also be replaced by other ones, more adapted for a given task, for example a score to use for interactive segmentation in which the score would express the difficulty for the user to get the desired result from the obtained segmentation.

Finally, one could use a more sophisticated algorithm to learn the best hierarchy and cut level for a given type of images. For example, a reinforcement learning approach [Mnih et al. (2015)] could be used for the system to explore more efficiently the space of all possible hierarchies combinations given a score to optimize on a training set.

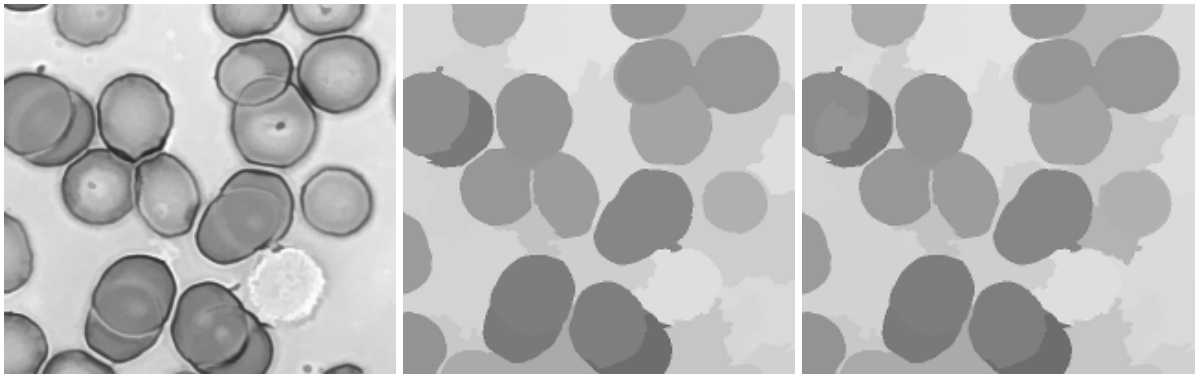


(a) Some examples in the training set

(b) $\mathbf{I}$ (c) `model` (d) `oracle`

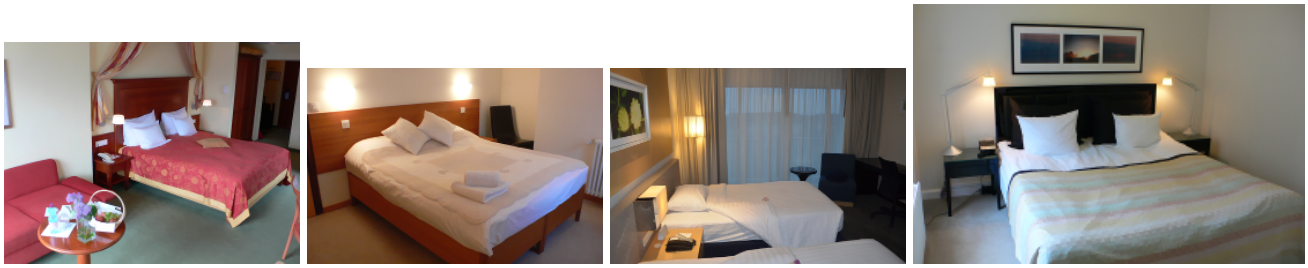


(e) $\mathbf{I}$ (f) `model` (g) `oracle`



(h) $\mathbf{I}$ (i) `model` (j) `oracle`

Figure 5.4 – Results on some examples of cells, for a Mumford-Shah score with a scale parameter $s = 1.168$. (b),(e),(h) are images from the testing set, (c),(f),(i) the model segmentations and (d),(g),(j) the oracle segmentations.



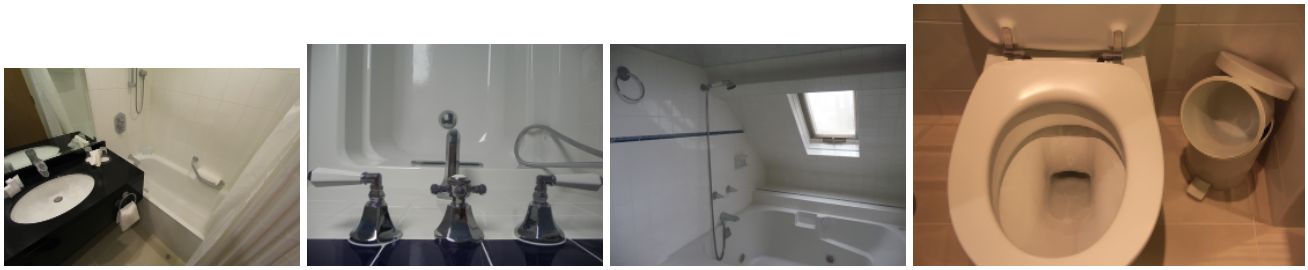
(a) Some examples in the training set

(b) **I** (c) `model` (d) `oracle`

(e) **I** (f) `model` (g) `oracle`

(h) **I** (i) `model` (j) `oracle`

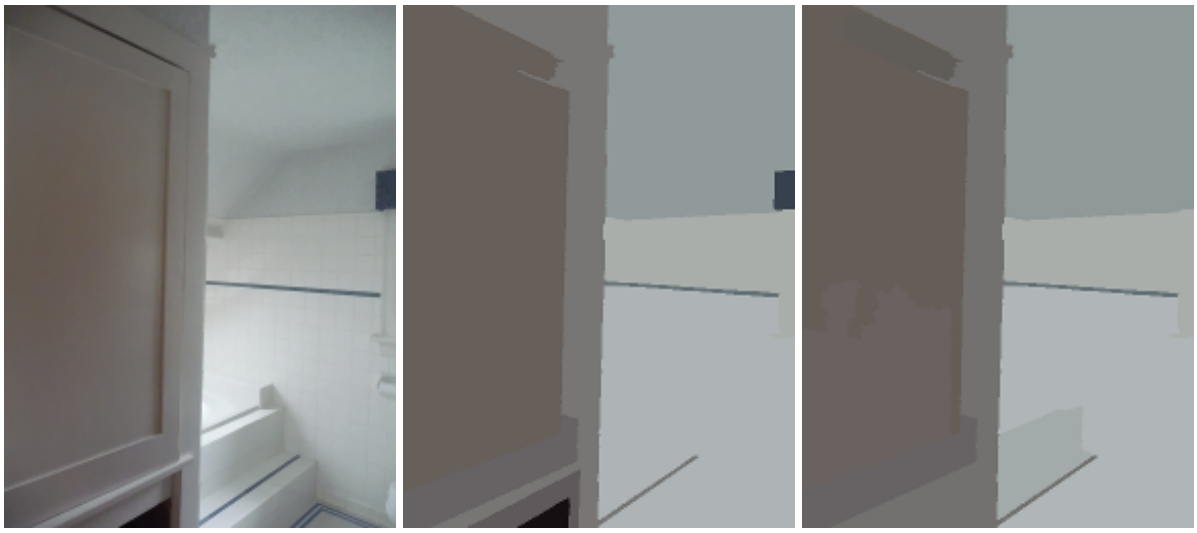
Figure 5.5 – Results on some examples of Intrinsic Images in the Wild, for a WHDR score.(b),(e),(h) are images from the testing set, (c),(f),(i) the model segmentations and (d),(g),(j) the oracle segmentations.



(a) Some examples in the training set

(b) **I** (c) `model` (d) `oracle`

(e) **I** (f) `model` (g) `oracle`



(h) **I** (i) `model` (j) `oracle`

Figure 5.6 – Results on some examples of Intrinsic Images in the Wild, for a WHDR score.(b),(e),(h) are images from the testing set, (c),(f),(i) the model segmentations and (d),(g),(j) the oracle segmentations.

(a) Some examples in the training set

(b) **I** (c) `model` (d) `oracle`

(e) **I** (f) `model` (g) `oracle`

(h) **I** (i) `model` (j) `oracle`

Figure 5.7 – Results on some examples of Intrinsic Images in the Wild, for a WHDR score.(b),(e),(h) are images from the testing set, (c),(f),(i) the model segmentations and (d),(g),(j) the oracle segmentations.

Chapter 6

Using the Distances Between Hierarchies As Features Characterizing Images

Associated publication:

— [Fehri et al. (2018)] A. Fehri et al. [2018]. « Characterizing Images by the Gromov-Hausdorff Distances Between Derived Hierarchies ». In: *2018 IEEE International Conference on Image Processing (ICIP)*

Résumé

Dans ce chapitre, nous proposons d'utiliser les matrices des distances inter-hiérarchies comme descripteurs d'images pour des tâches de classification. Nous testons ces descripteurs pour classifier des classes d'images générées suivant un modèle de feuilles mortes, ainsi que des classes d'images de textures. Les résultats obtenus montrent la pertinence et l'efficacité de ces descripteurs pour caractériser les propriétés géométriques des images.

Abstract

In this chapter, we propose to use the inter-hierarchies distances matrices as image descriptors for classification tasks. We test these descriptors to classify images generated using dead-leaf process with different parameters, as well as texture image classes. The results obtained show the relevance and effectiveness of these descriptors for characterizing the geometric properties of the images.

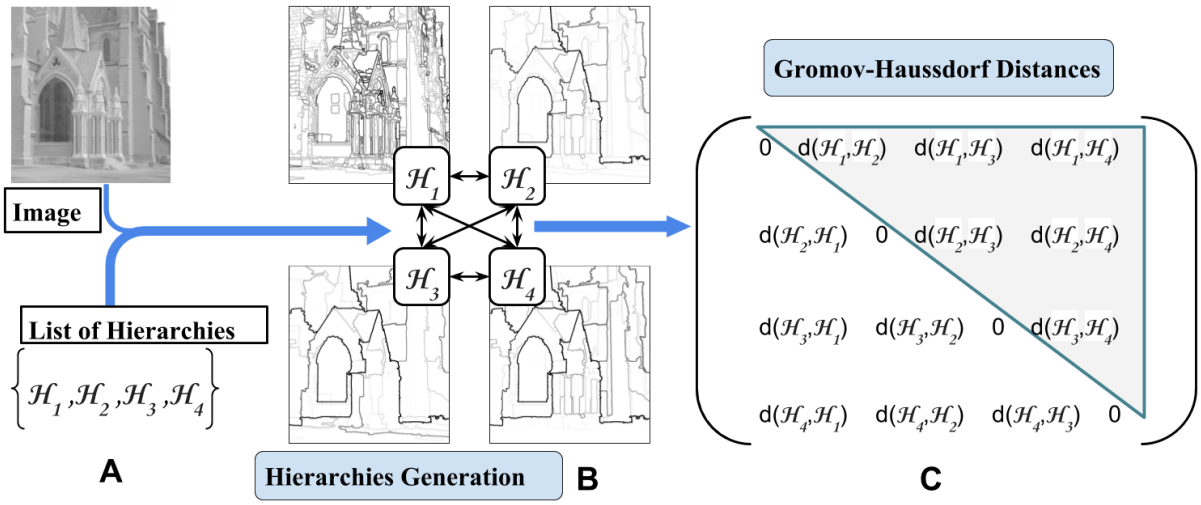


Figure 6.1 – Overview of our approach **A**: We start from an image and a set of morphological hierarchical segmentations techniques. **B**: Each of these techniques highlight different contours of the image, and we can quantify their relative contribution using the Gromov-Hausdorff distance. **C**: We propose to use the inter-hierarchies Gromov-Hausdorff distances matrix as features leveraging the discriminative power of all these hierarchies.

6.1 Introduction

In the previous chapter, we sought to identify the hierarchy best able to bring us closer to a good segmentation. We looked for these hierarchies among a collection chosen a priori. However, the resulting hierarchical segmentation approach is application-dependent, and its efficiency depends on the types of structures present in the images. In this chapter, we propose a new approach to take advantage of such multi-scale analysis. The main idea is to quantify the distance between different hierarchical representations to characterize images, and use them as features to characterize the images. In the examples presented, we only use selected morphological hierarchical segmentation methods such as SWS hierarchies. However, any hierarchical method can fall within its scope.

We begin by showing how, starting from a set of hierarchies and a given image, the computation of inter-hierarchies distances provides us with a rich analytical representation (i.e. feature) of the image that can be used to characterize it. We then illustrate the discriminative and explanatory power of such representations on generated shapes images. An overview of the process can be found in figure 6.1.

6.2 Features on Hierarchies using the Gromov-Hausdorff Distance

6.2.1 A structured richness of representations

Hierarchical segmentation is a type of low-level image analysis often used to make easier the obtention of a suitable segmentation. In this work, we limit ourselves to morphological hierarchies that can efficiently be obtained within the graph-based framework described in

chapter 3. In particular, the model of the stochastic watershed (SWS) hierarchies (section 3.3.6) is able to adapt to the specific objects properties to enhance (e.g. elongation, surfaces equilibrium, contrast). By only choosing high-levels parameters such as the density governing the distribution of markers or their shape and size, we can thus generate at will various multi-levels representations of the images highlighting various types of regions. Thanks to this versatility, specific hierarchies can be built: for example, we can favor certain shapes in the images by considering a surface-based SWS hierarchy in which the measured areas are the ones of regions eroded with anisotropic structuring elements.

An additional layer of supplementary complexity can be added via the possibility to combine hierarchies to obtain new ones, as we have seen in chapter 4, whether this process is linear (hierarchical chaining) or parallel (functional combination of ultrametrics). Each hierarchy then expresses certain particular images characteristics. To use the analogy of colors, each hierarchy is a black and white image resulting from the passage of a color image through a colored filter. Multiplying these filters makes it possible to obtain many black and white images that characterize the distribution of colors in the image.

Finally, each hierarchy provided with its ultrametric is a metric space. The Gromov-Hausdorff distance measures the distance between such metric space. This is why, as we have seen in section 4.2, the space of hierarchies provided with the Gromov-Hausdorff distance can itself be structured as a metric space.

We argue that the wealth of controlled understandable options to generate hierarchies, as long as the possibility to measure their relative specific descriptive power, can lead to powerful image features.

6.2.2 A condensed and descriptive image feature

Indeed, once provided with a family of hierarchies, one may wonder if there are ways to use the different information they provide to characterize images. The usual approach to do so it to extract information at various levels of the hierarchies, which often requires a hard parameter-tuning. Furthermore, it obliterates the interesting property that hierarchical segmentations are more informative than flat segmentations as they capture simultaneously cluster structure at all levels of granularity.

Instead, we propose to make use of the Gromov-Hausdorff distance between hierarchies to build a features space that capture the their relative specific descriptive power of several hierarchies applied on the same image. This distance is a distance between metric spaces that has been introduced in section 2.4.3. Intuitively, it measures how close can we get to an isometric (distance-preserving) embedding between two metric spaces. By reducing this distance to the subclass of ultrametric spaces, we can in particular quantify the relative contributions of different hierarchical clusterings. Furthermore, it is hard to compute in the general case, but can easily be computed for two ultrametric spaces (i.e. hierarchies) built upon the same set of points (in our case, starting from the same fine partition of the image).

Provided with such a distance, we can quantify the relative contributions of different hierarchies built upon the same image. This provides us with a condensed representation leveraging the information provided by all the different levels of these different hierarchies. To do so, let us consider an image $\mathbf{I}$ and a set of complementary hierarchies $((\mathcal{H}_1, \boldsymbol{\lambda}_1), \ldots, (\mathcal{H}_N, \boldsymbol{\lambda}_N))$ built upon this image. As we have seen in section 2.4.3, it is then straightforward to compute the GH-distance between these hierarchies, as they constitute ultrametric spaces upon the same set. We take advantage of it by building the following symmetrical distances matrix:

$$M(\mathbf{I}, (\mathcal{H}_1, \ldots, \mathcal{H}_N)) = \left[\mathrm{d_{GH}}(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j)\right]_{(i,j)\in\{1,\ldots,N\}^2} \tag{6.1}$$

Since this matrix is symmetrical, we retain for each image its upper triangular part only. This constitutes a descriptor of the image for which we only had to specify the high-levels parameters governing the hierarchies generation. Summarizing, an overview of the proposed process to extract features from a family of hierarchies can be found in figure 6.1.

6.3 Experimental Results

In this section, we present some experimental results highlighting the properties of the descriptors we have proposed in the previous section.

6.3.1 Dead leaves process classification

In a first experiment, we highlight the discriminative power of the unsupervised hierarchical features we introduced, as long as their understandability. In the spirit of [Yan et al. (2017)], we want to test if these features capture pertinent information leading to a quicker understanding of the images. To do so, we consider a classification problem on a set of simulated images from different dead leaves process [Jeulin (1997); Matheron (1975)].

In a dead leaves model, two dimensional textured surfaces (which are called “leaves” or “primary grains”) are sampled from a shape and size distribution and then placed on the image plane at random positions, occluding one another to produce an image. It is well-known that such a model creates images which share many properties with natural images such as scale invariance and other statistical properties [Zoran et al. (2012); Pitkow (2010)].

In our experiment, we have simulated five classes with 100 images each, by using dead leaves model with different primary grains: circles, crosses, flowers, horizontal and vertical lines. Examples of simulated images can be found in figure 6.2. Note that we have included different sizes and orientations tweaks to increase the difficulty of the identification. For each of these images, we compute the following hierarchies: trivial, surface-based SWS hierarchies with structuring elements of various sizes and forms (cross, circle, diagonals, horizontal and vertical lines), as long as logical combinations AND and OR of these SWS hierarchies. Then we generate for each of these images the inter-hierarchical distances matrices of equation 6.1.

We can then use these matrices as features in a classical classification pipeline using a linear support vector machines (SVM) to classify images of each class. During the training phase, we

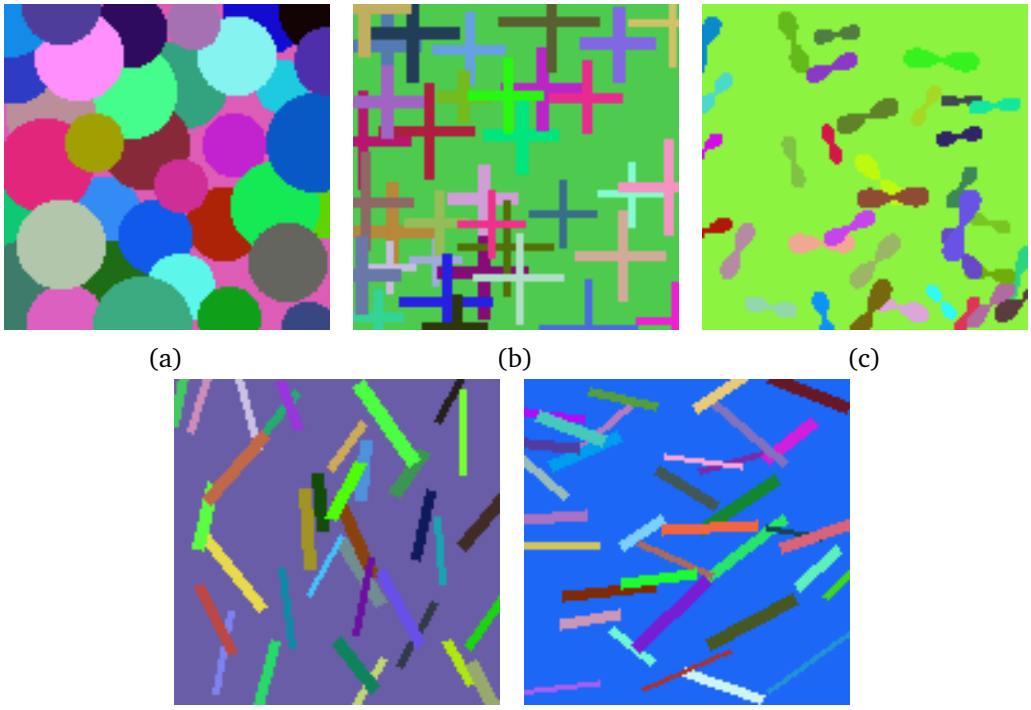


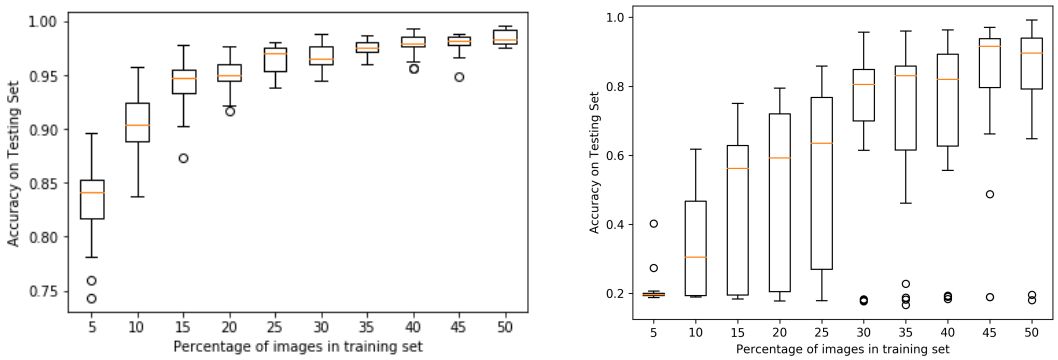
Figure 6.2 – False-color representation of simulated images by dead leaves model with different primary grains.

provide the SVM with inter-hierarchical distances matrices as images features, and ground truth label for each training image. We thus train the SVM to discriminate between the diffferent classes by maximizing the margins on the training set. We then test the efficiency of this trained classifier on the testing set. We notice that the system can learn with very few examples how to discriminate properly these five classes. In order to have a comparison point, we conduct the same experiment using a Convolutional Neural Network (CNN) with a two-layers architecture [1] without image augmentation to make a fair comparison.

In figure 6.3 are represented for both experiments the evolutions of the average F-score with respect to the percentage of images used in the training set. In the first experiment (using the distances matrices as features), we notice that using only 5% of them (so 25 images out of 500) already leads to a 85% F-score over the remaining images, and that this figure quickly goes up. In the CNN experiment, the number of required training images to get to the same results is significantly larger.

It is thus as if, on the contrary to CNN that have a black-box behavior, our approach shows what is often referred to as an “aha moment”, i.e. a moment of sudden realization and comprehension [Yan et al. (2017)]. This translates a form of understanding of the content of the image, which is corroborated by the study of the importance of which specific inter-hierarchies

1. (12 Conv + 12 Conv + MaxPolling(3×3) + Dropout(0.3)) + (24 filters + 24 filters + MaxPolling(3×3)+ Dropout(.5)) + FullyCon64 + Dropout (.5) + SoftMax. Categorical cross-entropy as loss function and adaptive gradient (Adagrad) as optimizer.



(a) (b)

Figure 6.3 – Classification error vs the number of images in the training set (25 repetitions) : (a) Linear SVM on proposed features, (b) CNN.

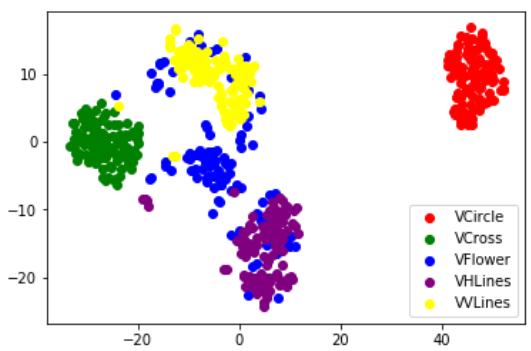
distances were the more useful to discriminate between two types of classes.

For example, discriminating between horizontal and vertical lines will mainly be due to $\mathrm{d_{GH}}(\mathcal{H}_{surf-VertSE}, \mathcal{H}_{surf-HorizSE})$, while discriminating between crosses and circles will mainly be due to $\mathrm{d_{GH}}(\mathcal{H}_{surf-CrossSE}, \mathcal{H}_{surf-HexSE})$. A visualization of the quality of the features space thus generated can be found in figure 6.4(a), where we project the features in a space of two dimensions using the t-SNE algorithm introduced in section 4.3.2 [Maaten et al. (2008)]. Furthermore, using the variable selection method L_1-SVM [Zhu et al. (2004)], we can isolate the more discriminative distances for two specific classes to separate. For example, the t-SNE visualization in figure 6.4(a) shows us that discriminating between the classes "Flowers" and "Horizontal Lines" is not straightforward. The more discriminative variable between these two classes is the distance between $\mathcal{H}_{surf-VertSE}$ and $\mathcal{H}_{\mathrm{AND}(surf-VertSE,surf-HexSE)}$: this is a geometrical interpretation of the image content, as they respectively capture straight lines and lines with a protuberance (i.e. flowers). Projecting the distances features onto the subspace of the two more discriminative variables properly separates these two classes, as can be seen in figure 6.4(b).

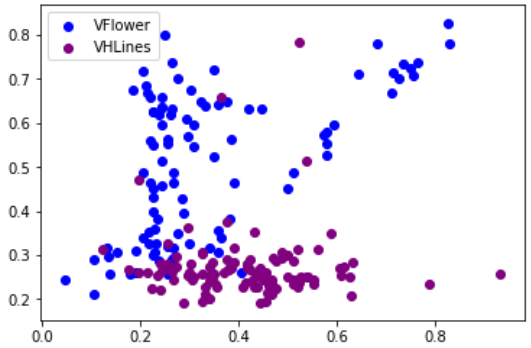
We are thus provided with a good way to control and understand features generation to be further used in vision-based system, especially when we have prior information about the type of objects we are looking for in images, and/or when we have very few training examples and want to maximize their usefulness. Furthermore, this could be used in a method to obtain a small descriptive collection of non-redundant hierarchies, to be further used in an automatic segmentation workflow as described in chapter 5.

6.3.2 Mixture models

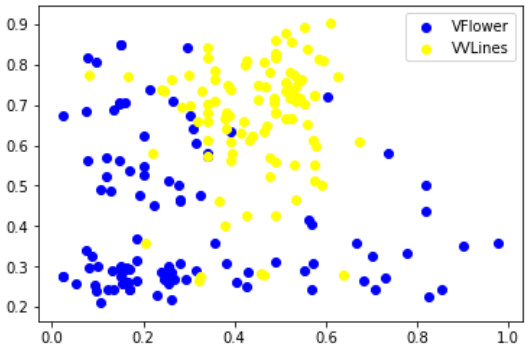
Furthermore, one may want to use such features to estimate the composition of mixtures models. In a first approach, we generate different mixtures of horizontal and vertical lines and want to evaluate how our approach can estimate their relative proportions. In figure 6.5 is



(a) 2D scatterplot by t-SNE



(b) 2D scatterplot by L_1-SVM



(c) 2D scatterplot by L_1-SVM

Figure 6.4 – (a) We notice that the classes “Flowers” and “Horizontal Lines” are not well separated (b)These two distances between hierarchies provide a geometrical understanding of the images content.Projecting along these features does indeed separate these classes efficiently. (c) The same can be done for example for the classes “Flowers” and “Vertical Lines”

represented the different classification scores between the 50/50 class and several other mixtures class. This plot confirms that these features capture internal hierarchical structures and can be used to operate mixture classification.

6.3.3 Textures classification

In order to have an idea of the benefits and limits of the introduced features, we use them for a classification problem of textures images. We select images from five classes of textures within the *Describable Textures Dataset* (DTD), a collection of textural images in the wild, annotated with a series of human-centric attributes, inspired by the perceptual properties of textures [Cimpoi et al. (2014)]. The five classes are: *banded*, *chequered*, *dotted*, *fibrous* and *interlaced*. Examples of images of these classes can be found in figure 6.6.

As in the previous experiment of section 6.3.1, we compute inter-hierarchies distances matrices on each image, and then use them as features in a classical classification pipeline using a linear support vector machines (SVM) to classify images of each class. The results obtained on the test set are provided in figure 6.7 in the form of a confusion matrix. One can notice that classes expressing strong geometrical features such as *banded*, *chequered*, and *dotted*

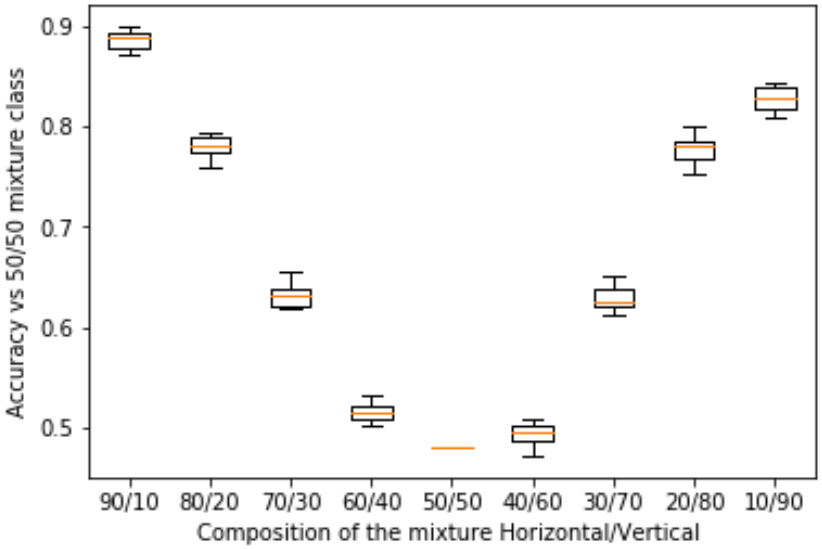
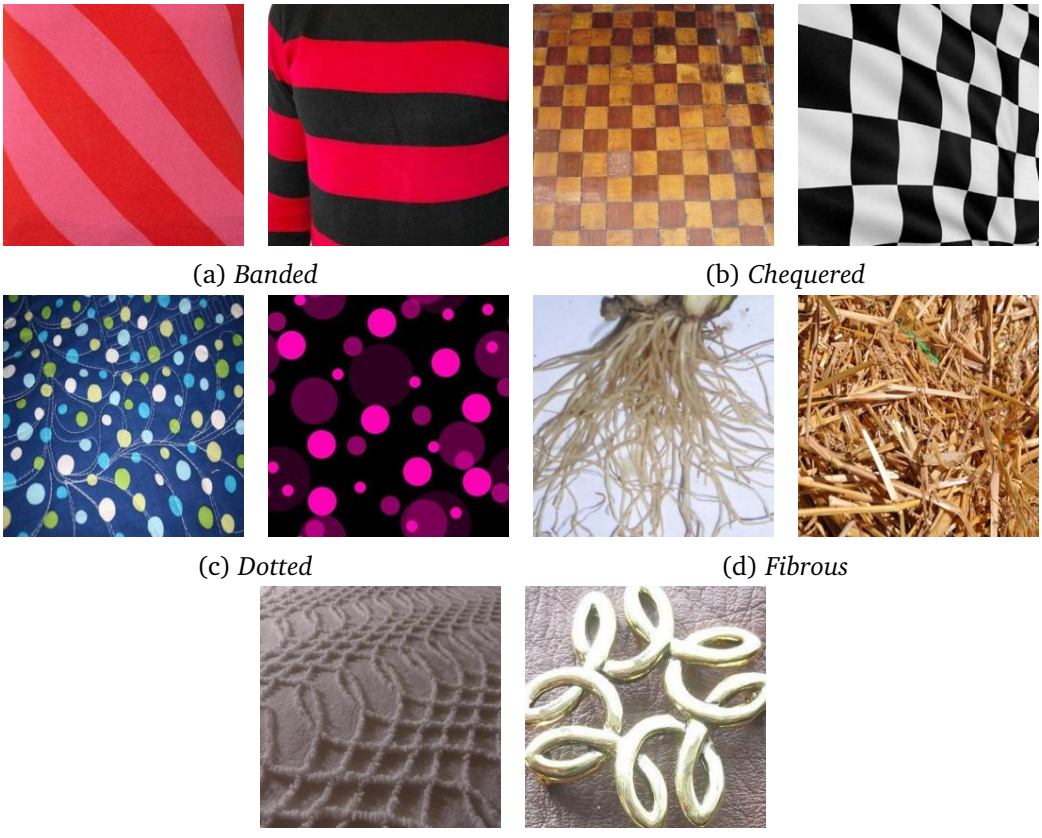


Figure 6.5 – Classification between different mixtures and the 50/50 mixture class.



(a) *Banded* (b) *Chequered*

(c) *Dotted* (d) *Fibrous*

(e) *Interlaced*

Figure 6.6 – Examples images of textures classification problem.

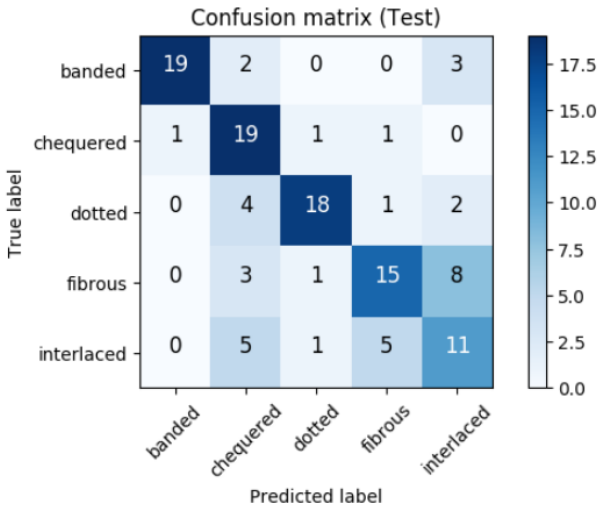


Figure 6.7 – Confusion matrix representing results obtained for the textures classification problem when using inter-hierarchies distances matrices as features with a linear SVM on the testing set.

are usually well-classified, whereas classes that can only be understood based on periodicity properties as *fibrous* or *interlaced* are poorly classified. This is consistent with the fact that the hierarchies we compute are based solely on geometrical properties and do not aim at capturing frequency properties for example. In this regard, they appear to be potentially complementary with other multi-scales features computed in the frequency domain, that are better suited to capture periodicities in images [Sifre et al. (2013)].

6.4 Conclusions and Perspectives

In this chapter, we have proven the efficiency of features created by estimating the differences between different hierarchical segmentations of the same images. The proof-of-concept experiments we conducted show that these features can provide a geometrical interpretation of the image content, and help to analyze images classes with very few examples. Furthermore, this approach can be extended to any type of hierarchies to capture various types of information. We see three ways to use such information.

First, if we know well the type of images we want to discriminate, we can generate at will the hierarchies that will be the more discriminative. Secondly, if we build a sufficiently wide range of complementary hierarchies for an unknown class of images, we can characterize these images by looking at the way they react to these operators through the study of inter-hierarchies distances. In line with classical morphological approaches, we can thus gain knowledge about images by studying the way they react to given operators. Finally, many methods exist to extract a segmentation out of a hierarchical segmentation, and our approach provides a way to identify the hierarchies that will lead to the best results in such a process. This is then an unsupervised technique that helps us to analyze the images content.

Chapter 7

Prior-Based Hierarchical Segmentation

Associated publications:

— [Fehri et al. (2017a)] A. Fehri et al. [2017a]. « Prior-based Hierarchical Segmentation Highlighting Structures of Interest ». In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, pp. 146–158

— [Fehri et al. (2017b)] A. Fehri et al. [2017b]. « Segmentation hiérarchique faiblement supervisée ». In: *Actes du 26e Colloque GRETSI, Juan-Les-Pins, France*.

Résumé

Dans les chapitres précédents, nous avons vu comment utiliser des hiérarchies morphologiques complémentaires pour extraire des régions pertinentes dans des images pour des tâches données. L'idée était de structurer de façon contrôlée l'information présente dans l'image de diverses manières, cependant cette analyse d'image était faite de manière spatialement homogène. D'un autre côté, de nombreuses méthodes nous permettent d'avoir une information préalable sur la position des objets d'intérêt dans les images. Dans ce chapitre, nous montrons comment une légère modification du modèle des hiérarchies de Ligne de Partage des Eaux (LPE) stochastique permet de prendre en compte toute information spatiale préalable pour obtenir une segmentation hiérarchique qui privilégie les contours ou régions d'intérêt tout en préservant les structures importantes de l'image. Plusieurs applications sont présentées qui illustrent la polyvalence et l'efficacité de la méthode.

Abstract

In the previous chapters, we saw how to use complementary morphological hierarchies to extract pertinent regions in images for given tasks. The idea was to structure the information present in the image in various controlled ways, however this image analysis was done in a spatially homogeneous way. On the other hand, many methods allow us to have prior information on the position of structures of interest in the images. In this chapter, we show how a slight modification

in the SWS hierarchies model allows to take into account any prior spatial information for obtaining a hierarchical segmentation that emphasizes the contours or regions of interest while preserving the important structures in the image. Several applications are presented that illustrate the method versatility and efficiency.

7.1 Introduction

In the previous chapters, we saw how to use complementary morphological hierarchies to extract pertinent regions in images for given tasks. The idea was to structure the information present in the image in various controlled ways and use the resulting characterization for segmentation or classification tasks. The information could be complex and was analyzed with discriminative hierarchies regarding the image local content, but they were however spatially homogeneous.

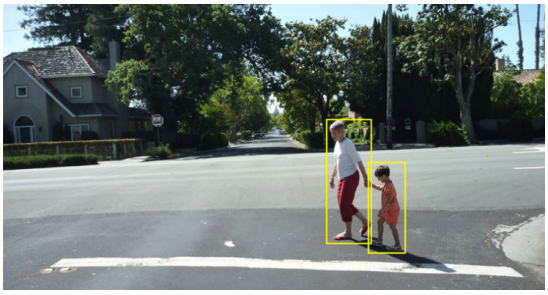
To go further, in this chapter we show how to treat images which content is not homogeneously distributed in the spatial domain, and how to take advantage of any prior information over the regions of interests in the images. This way, we can make use of various exogenous spatial information sources to direct the hierarchical segmentation construction. Such information can take numerous forms, as illustrated in figure 7.1. Indeed, significant developments have been made over the last decades in learning-based recognition methods for various tasks [Oquab et al. (2015); Long et al. (2015); Redmon et al. (2016)]. A variety of sources and modalities have emerged as well, with for example depth sensors [Fankhauser et al. (2015)] or multispectral and hyperspectral cameras [Chang (2003)], and provide additional information that can be useful for segmentation. Having a versatile hierarchical segmentation method that can take into account such information during its construction process thus appears to be very interesting.

Building upon the stochastic watershed model introduced in section 3.3.6 and used in previous chapters, we thus propose a method to take advantage of any prior spatial information previously obtained on an image to get a hierarchical segmentation of this image that emphasizes its regions of interest. This allows us to get more details in the designated regions of interest of an image while still preserving its strong structural information.

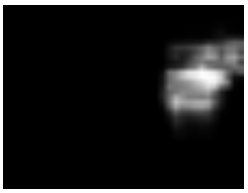
Indeed, we note that such hierarchical clusterings are useful to understand scenes as they enlarge the information support in a controlled way, and hence bring out significant salient features. We already have seen that the SWS model is versatile as multiple construction types can be thought of, the markers we use can be punctual or non-punctual and the number of markers we draw can be changed. In this chapter, we go a step further by slightly modifying the SWS model: using a probability density function governing the markers distribution that depends on a spatial exogenous information is a simple yet powerful way to have a very task-ajdustable hierarchical segmentation method.

Potential applications are numerous. When having a limited storage capacity (for very large images for example), this would allow us to keep details in the regions of interest as a priority. Similarly, in situations of transmission with limited bandwidth, one could first transmit the

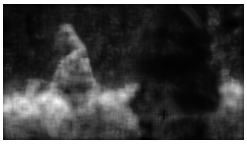
(a) RGBD image. The depth information can help to direct the segmentation process.



(b) Many methods provide rough localizations of classes of interests, for example here the class "People".



(c) Image and *heatmap* of the main class in the image



(d) Image and output of a non-blur zones detector

Figure 7.1 – A growing number of spatial exogenous information sources:
- localization methods adapted to each problem
- different channels
How can we use such exogenous information to pilot the hierarchical segmentation process?

important information of the image: the details of the face for a video-call, the pitch and the players for a soccer game and so on. One could also use such a tool as a preprocessing one, for example to focus on an individual from one camera view to the next one in video surveillance tasks. Finally, from an artistic point of view, the result is interesting and similar to a combination of focus and cartoon effects.

As we propose a hierarchical segmentation algorithm that focuses on certain predetermined zones of the image, the hierarchical aspect also allows us, for tasks previously described, to very simply tune the level of details wanted depending on the application.

Furthermore, our algorithm is very versatile, as the spatial prior information that it uses can be obtained for example by any of the numerous learning-based approaches proposed over the last decades to roughly localize objects [Oquab et al. (2015); Lampert et al. (2008); Sermanet et al. (2013)]. In this regard, our work joins an important research point that consists in designing approaches to incorporate prior knowledge in the segmentation, as shape prior on level sets [Chan et al. (2005)], star-shape prior by graph-cut [Veksler (2008)], use of a shape prior hierarchical characterization obtained with deep learning [Chen, Yu, et al. (2013)], or related work making use of stochastic watershed to perform targeted image segmentation [Malmberg et al. (2017)].

7.2 Hierarchy with Regionalized Fineness (HRF)

The SWS model was introduced in details in section 3.3.6, and in this section we reuse the notations previously introduced. The output of the SWS algorithm depends on the departure $\mathcal{MST}$ (structure and edges valuations) and of the probabilistic law governing the markers distribution. Furthermore, SWS hierarchies can be chained or combined, leading to a wide exploratory space that can be used in a segmentation workflow (cf. chapter 5), or to design discriminative image features for classification (cf. chapter 6).

Because of its versatility and good performance, SWS represents a good model into which prior information may be injected. Indeed, when having a prior information about the image, is it possible to use it in order to shed more light in some regions of the image than in others?

In the original SWS, a uniform distribution of markers is used (whatever size or form they may have). In order to have stronger contours in a specific region of the image, we adapt the model so that more markers are spread in this region.

Let E be an object or class of interest, for example $E =$ "face of a person", and $\mathbf{I}$ be the studied image. We denote by θ_E the probability density function (PDF) associated with E obtained separately, and defined on the domain D of $\mathbf{I}$, and by $\mathrm{PM}(\mathbf{I}, \theta_E)$ the probabilistic map associated, in which each pixel $p(x, y)$ of $\mathbf{I}$ takes as value $\theta_E(x, y)$ its probability to be part of E. Given such an information on the position of an object of interest in an image, we obtain a hierarchical segmentation focused on this region by modulating the distribution of markers. We refer to this hierarchy as to a *Hierarchy with Regionalized Fineness* (HRF).

If λ is a density defined on D to distribute markers (uniform or not), we set $\theta_E\lambda$ as a new density, thus favoring the emergence of contours within the regions of interest.

Considering a region R of the image, the mean number of markers falling within R is then:

$$\Lambda_E(\mathsf{R}) = \int_{(x,y)\in\mathsf{R}} \theta_E(x, y)\lambda(x, y)\,\mathrm{d}x\mathrm{d}y \tag{7.1}$$

Note that if we want N markers to fall in average within the domain D, we work with a slightly modified density:

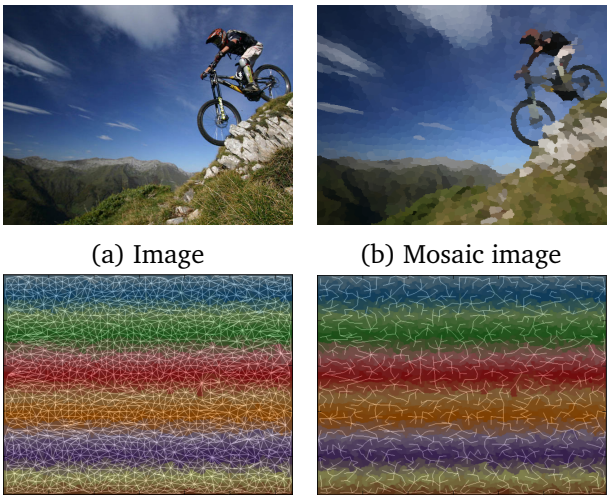
$$\hat{\lambda} = \frac{N}{\mu(D)}\lambda \tag{7.2}$$

Furthermore, this approach can be easily extended to the case where we want to take advantage of information from multiple sources. Indeed, if θ_{E_1} and θ_{E_2} are the PDF associated with two events E_1 and E_2, we can combine those two sources by using as a new density for example $\mathrm{AND}(\theta_{E_1}, \theta_{E_2})\lambda = (\theta_{E_1} \times \theta_{E_2})\lambda$, $\mathrm{OR}(\theta_{E_1}, \theta_{E_2})\lambda = (\theta_{E_1} + \theta_{E_2} - \theta_{E_1} \times \theta_{E_2})\lambda$, $\mathrm{SUP}(\theta_{E_1}, \theta_{E_2})\lambda$ or $\mathrm{INF}(\theta_{E_1}, \theta_{E_2})\lambda$.

7.3 Methodology

We present here the steps to compute a HRF for an event E given a probabilistic map $\mathrm{PM}(\mathbf{I}, \theta_E)$ providing spatial prior information on an image $\mathbf{I}$:

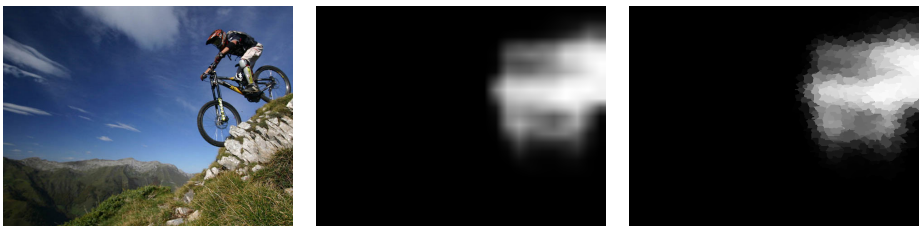
1. compute a fine partition π_0 of the image, define a dissimilarity measure between adjacent regions and compute the RAG $\mathcal{G}$, and then the $\mathcal{MST}(\mathcal{G})$ to easily work with graphs,



(a) Image (b) Mosaic image

(c) RAG (d) MST

2. compute a probabilistic map $\pi_\mu = \pi_\mu(\pi_0, \mathrm{PM}(\mathbf{I}, \theta_E))$ with each region of the fine partition π_0 taking as a new value the mean value of $\mathrm{PM}(\mathbf{I}, \theta_E))$ in this region,



(a) Image (b) Probabilistic Map $\mathrm{PM}(\mathbf{I}, \theta_E))$ associated with “Bike" class (c) π_μ

3. compute new values of edges by a bottom-up approach as described in section 7.2, where for each region R_i of π_0, $\Lambda(\mathrm{R}_i)$ corresponds to the mean value taken by pixels of the region R_i in π_μ. Note that this approach allows a highly efficient implementation using dynamic programming on graphs. The resulting hierarchy retains more details in the region of interest, while still capturing the important structures of the image.

(a) 10 regions (b) 100 regions (c) 200 regions

Making use of multiple sources is also very simple and consists in fusing the probability maps in a controlled way, depending on the objective. It is illustrated in figure 7.5.

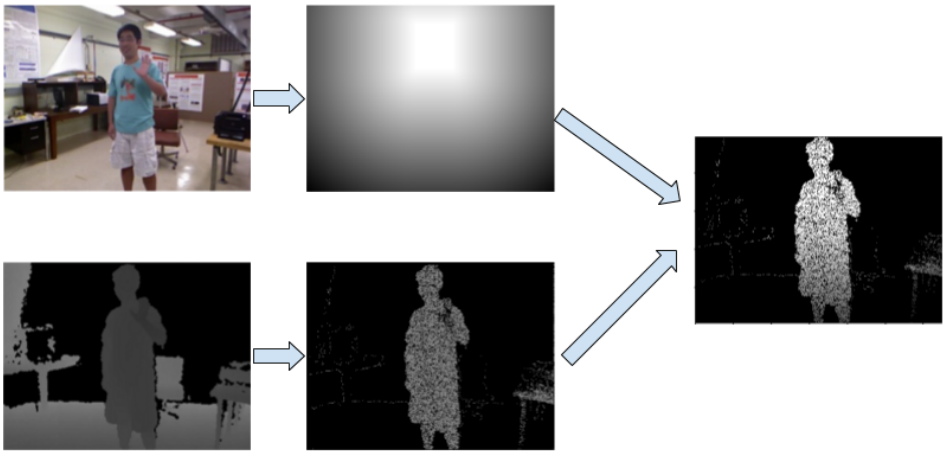


Figure 7.5 – Dummy example of the combination of two probability maps to get a new one that be put as input of the HRF. One probability map highlights the face of the person in the image, while the other corresponds to the specific depth at which the character is, extracted from a depth image. Starting from θ_{E_1} and θ_{E_2}, the new probability density function we use is $(\theta_{E_1} \times \theta_{E_2})\lambda$, corresponding to the probability density function of the event "both E_1 and E_2 are present".

7.4 Modulating the HRF depending on the couple of regions considered

If we want to favor certain contours to the detriment of others, we can modulate the density of markers in each region by taking into account the strength of the contour separating them but also the relative position of both regions.

We use the same example and notations as in section 7.2, and thus want to modulate the distribution of markers relatively to R_s, R_t and their frontier. For example, to stress the strength of the gradient separating both regions we can locally spread markers following the distribution $\chi(\mathsf{R}_s, \mathsf{R}_t)\lambda$, with $\chi(\mathsf{R}_s, \mathsf{R}_t) = \omega_{st}$. This corresponds to the classical volume-based SWS, which allows to obtain a hierarchy that takes into account both surfaces of regions and contrast between them.

To go further, one can use any prior information in a similar way. Indeed, while using prior information to influence the output of the segmentation workflow, one might also want to choose whether the relevant information to emphasize in resulting segmentations is the foreground, the background or the transitions between them.

For example, having more details in the transition regions between background and foreground allows us to have more precision where the limit between foreground and background is

actually unclear. As a matter of fact, the prior information often only provides rough positions of the foreground object with blurry contours, and such a process would allow to get precise contours of this object from the image.

Let us consider this case and define for each couple of regions $(\mathsf{R}_s, \mathsf{R}_t)$ a suitable $\chi(\mathsf{R}_s, \mathsf{R}_t)$. We then want $\chi(\mathsf{R}_s, \mathsf{R}_t)$ to be low if R_s and R_t are both in the background or both in the foreground, and high if R_s is the background and R_t in the foreground (or the opposite). We use:

$$\begin{cases} \tilde{\lambda} = \chi\lambda \\ \chi(\mathsf{R}_s, \mathsf{R}_t) = \frac{\max(m(\mathsf{R}_s), m(\mathsf{R}_t))(1-\min(m(\mathsf{R}_s), m(\mathsf{R}_t)))}{0.01+\sigma(\mathsf{R}_s)\sigma(\mathsf{R}_t)}, \end{cases} \tag{7.3}$$

$m(\mathsf{R})$ (resp. $\sigma(\mathsf{R})$) being the normalized mean (resp. normalized standard deviation) of pixels values in the region R of $\mathrm{PM}(\mathbf{I})$. Thus the number of markers spread will be higher when the contrast between adjacent regions is high (numerator term) and when these regions are coherent (denominator term).

Then for each edge, its new probability to be cut is :

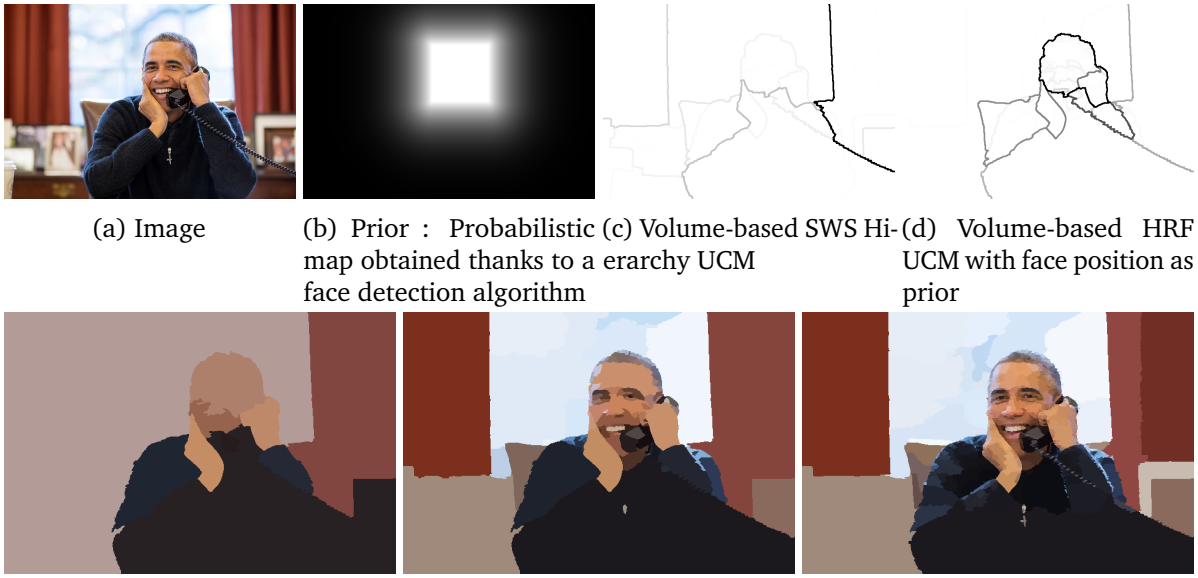
$$\begin{aligned} \mathbb{P}[(\mu(\mathsf{R}_p) \geq 1) \wedge (\mu(\mathsf{R}_q) \geq 1)] = 1 - \exp^{-\chi(\mathsf{R}_s,\mathsf{R}_t)\Lambda(\mathsf{R}_p)} - \exp^{-\chi(\mathsf{R}_s,\mathsf{R}_t)\Lambda(\mathsf{R}_q)} \\ + \exp^{-\chi(\mathsf{R}_s,\mathsf{R}_t)\Lambda(\mathsf{R}_p \cup \mathsf{R}_q)} \end{aligned} \tag{7.4}$$

In the spirit of the work presented in [Chen, Dai, et al. (2016)], this mechanism provides us with a way to "realign" the hierarchy with respect to the relevant prior information to get more details where the information is blurry. Similar adaptations can be thought of to emphasize details of background or foreground regions.

7.5 Applications

7.5.1 Scalable transmission favoring regions of interest

Let us consider a situation where one emitter wants to transmit an image through a channel with a limited bandwidth, e.g. for a videoconference call. In such a case, the more important informations to transmit are details on the face of the person on the image. Besides, we nowadays have highly efficient face detectors, using for example Haar-wavelets as features in a learning-based vision approach [Viola et al. (2001)]. Considering that for an image in entry, the face can be easily detected, we can use this information to produce a hierarchical segmentation of the image that accentuates the details around the face while giving a good sketch of the image elsewhere. Depending on the bandwidth available, we can then choose the level of the hierarchy to select and obtain the associated partition to transmit, ensuring us to convey the face with as much details as possible. Some results are presented in Figure 7.6, with notably a comparison between a classical volume-based SWS UCM and a volume-based HRF UCM.



(a) Image

(b) Prior : Probabilistic map obtained thanks to a face detection algorithm

(c) Volume-based SWS Hierarchy UCM

(d) Volume-based HRF UCM with face position as prior

(e) Segmentation obtained with HRF - 10 regions

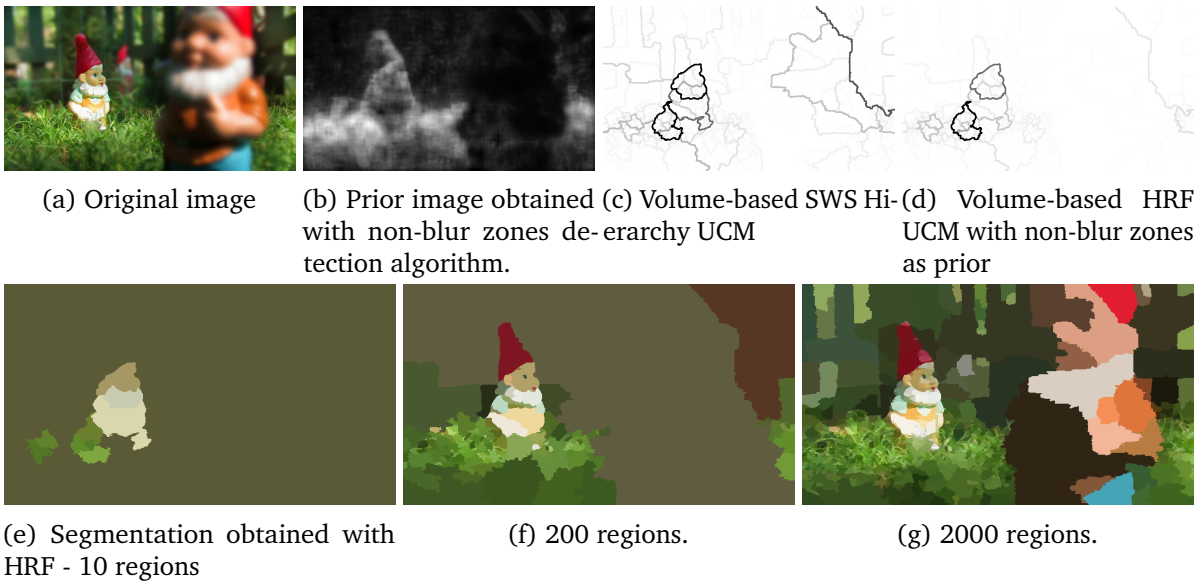
(f) Segmentation obtained with HRF - 100 regions

(g) Segmentation obtained with HRF - 1000 regions

Figure 7.6 – Hierarchical segmentation of faces.

7.5.2 Artistic aspect: focus and cartoon effect

The same method can also be used for artistic purposes. For example, when taking as prior the result of a blur detector [Su et al. (2011)], we can accentuate the focus effect wanted by the photograph and turn it into a cartoon effect as well - see Figure 7.7 for an illustration of the results.



(a) Original image

(b) Prior image obtained with non-blur zones detection algorithm.

(c) Volume-based SWS Hierarchy UCM

(d) Volume-based HRF UCM with non-blur zones as prior

(e) Segmentation obtained with HRF - 10 regions

(f) 200 regions.

(g) 2000 regions.

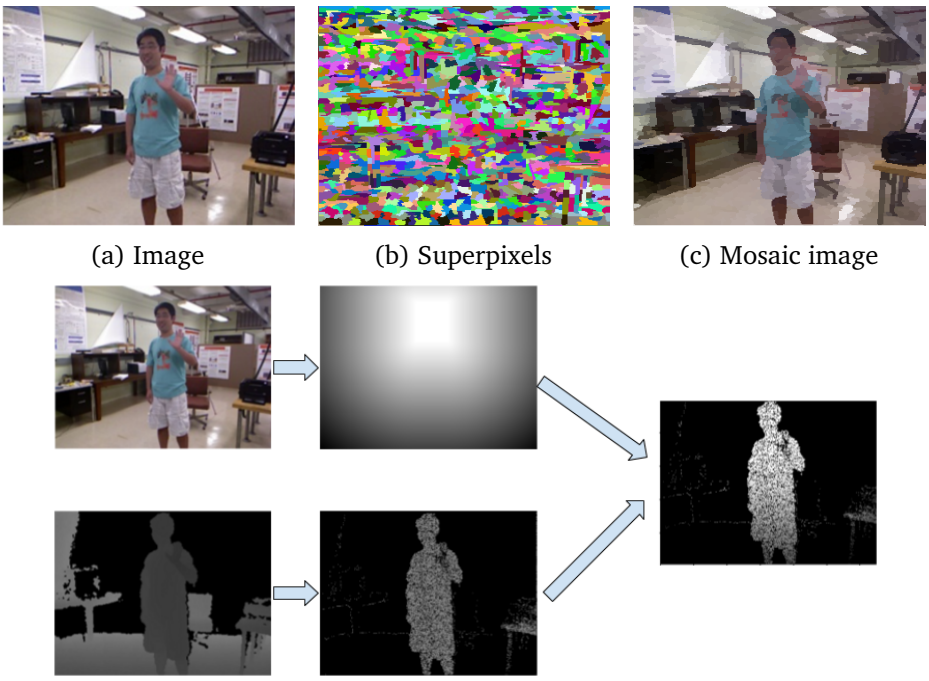
Figure 7.7 – Hierarchical segmentation of non-blur objects.

7.5.3 Combining hierarchies using different sources

We have exposed in chapter 4 various ways to combine hierarchies. One can as well combine hierarchies obtained with prior information coming from multiple sources. Let us consider for example a RGB-D image in which a person is present. A face detection algorithm can be used to localize the face of the person in the image. Using this information, one can then extract the particular depth at which the face of the person (and thus the person in most cases) is present in the depth image.

One thus obtain two probability maps: one associated with the face of the person, that we denote PM_F, the other translating the depth at which the person is, that we denote PM_D. Combining these two probability maps by multiplying them (which corresponds to the AND operator) then leads to a new probability map $\mathrm{PM}_{\mathrm{AND}(F,D)}$ that highlights the whole body of the person with a particular emphasis on his face, as one can see in figure 7.8.

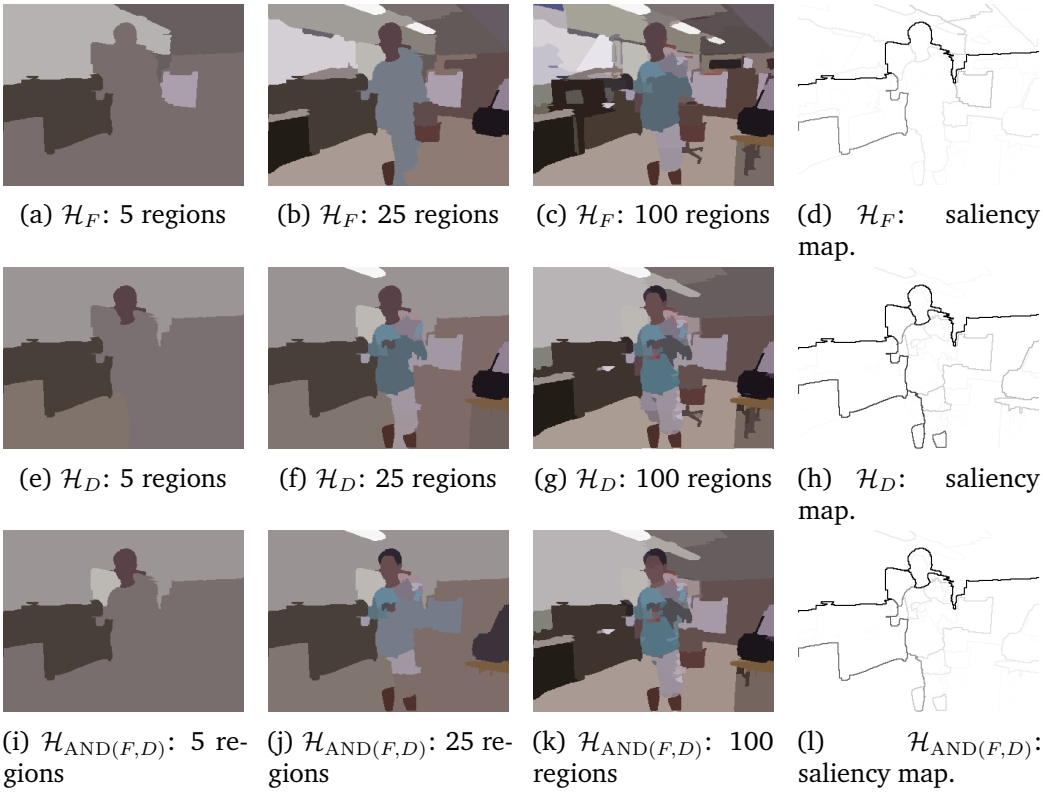
The HRF $\mathcal{H}_{\mathrm{AND}(F,D)}$ built using $\mathrm{PM}_{\mathrm{AND}(F,D)}$ better captures the pertinent information than the two HRF $\mathcal{H}_F$ and $\mathcal{H}_D$ built using the two probability maps separately, as illustrated in figure 7.9.



(a) Image (b) Superpixels (c) Mosaic image

(d) Combination of probability maps

Figure 7.8 – Combining two probability maps associated with two information sources.



(a) $\mathcal{H}_F$: 5 regions (b) $\mathcal{H}_F$: 25 regions (c) $\mathcal{H}_F$: 100 regions (d) $\mathcal{H}_F$: saliency map.

(e) $\mathcal{H}_D$: 5 regions (f) $\mathcal{H}_D$: 25 regions (g) $\mathcal{H}_D$: 100 regions (h) $\mathcal{H}_D$: saliency map.

(i) $\mathcal{H}_{\text{AND}(F,D)}$: 5 regions (j) $\mathcal{H}_{\text{AND}(F,D)}$: 25 regions (k) $\mathcal{H}_{\text{AND}(F,D)}$: 100 regions (l) $\mathcal{H}_{\text{AND}(F,D)}$: saliency map.

Figure 7.9 – Example of the effect of the combination of probability maps for HRF.

7.5.4 Weakly-supervised HRF

In the same spirit, various methods now exist to automatically roughly localize the principal object in an image. We inspire ourselves from [Oquab et al. (2015)] to do so. Using the state-of-the-art convolution neural network (CNN) classifier VGG19 [Simonyan et al. (2014)] trained on the 1000 classes ImageNet database [Deng et al. (2009)], we first determine what is the main class in the image. Note that this CNN takes as input only images of size 224×224 pixels. Once it is known, we can then, by rescaling the image by a factor $s \in \{0.5, 0.7, 1.0, 1.4, 2.0, 2.8\}$, compute for sub-windows of size 224×224 of the image the probability of appearance of the main class. By simply superimposing the results for all sub-windows, we thus obtain a probabilistic map of the main class for each rescaling factor. By max-pooling, we keep in memory the result of the scale for which the probability is the highest. The *heatmap* thus generated takes as value in each pixel the probability that this pixel belongs to the class of interest. This probability map can then be used to feed our algorithm. This way, we have at our disposal an automatized way to focus on the principal class in the scene with the desired level of detail.

All in all, the combination of this approach and the HRF is very interesting as it constitutes an all-in-one method to classify the image (i.e. to find the main class in the image), to localize the objects of this main class in it, and to get from it a hierarchy of segmentation in which those elements are highlighted. An overview of this method can be found in figure 7.10 and some results are presented in figure 7.11 and 7.12.

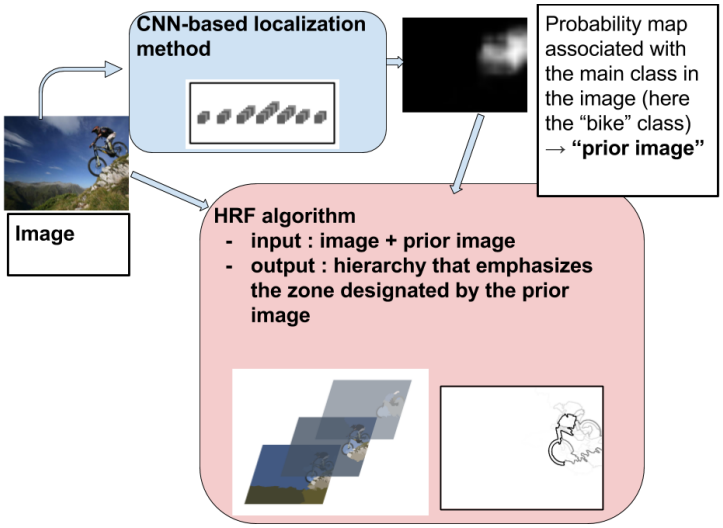
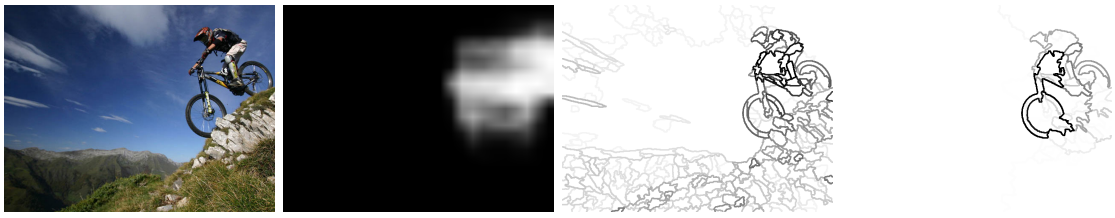


Figure 7.10 – Weakly-supervised HRF algorithm.



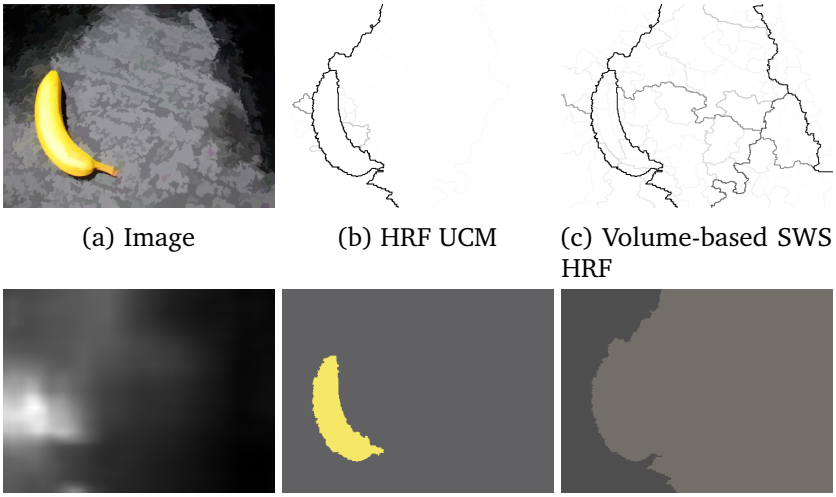
(a) Original image (b) Heatmap issued by the CNN-based localization method (c) Volume-based SWS hierarchy UCM (d) HRF UCM.

(e) Segmentation obtained with HRF - 10 regions. (f) 100 regions. (g) 200 regions.

Figure 7.11 – Hierarchical segmentation of the main class (class “bike”) in an image with heatmap issued of a CNN-based method as input.

7.5.5 Hierarchical co-segmentation

Another potential application is to co-segment with the same fineness level an object appearing in several different images. For example, when given a list of images of the same object taken from different perspectives/for different conditions, we can follow the state-of-the-art matching procedure [Lowe (2004)]:



(a) Image (b) HRF UCM (c) Volume-based SWS HRF

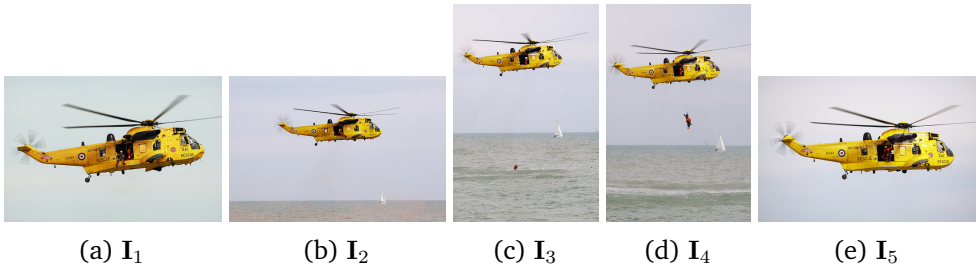
(d) Heatmap of the main class (e) 2 first regions of the HRF. (f) 2 first regions of the volume-based SWS hierarchy.

Figure 7.12 – Hierarchical segmentation of the main class (class “banana”) in an image with heatmap issued of a CNN-based method as input.

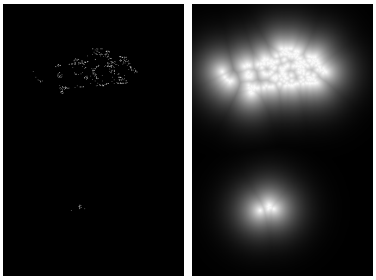
1. compute all key-points in all images, using for example the FAST algorithm [Rosten et al. (2006)]
2. compute local descriptors at these key-points, such as SIFT [Lowe (2004)]
3. match those key-points using a spatial coherency algorithm as RANSAC [Fischler et al. (1987)]

Once it is done we retain these matched key-points between all images, and generate probability maps of the appearance of the matched objects using a morphological distance function to the matched key-points.

These probability maps can then feed our algorithm, resulting in a hierarchical co-segmentation that emphasizes the matched zones of the image. Some results are presented in figure 7.13. This example speaks for the interest of such an approach. While the contours in the helicopter are not necessarily the more salient ones in each image taken separately, the matching of keypoints throughout images results in a prior image emphasizing regions inside it. The comparison of the volume-based SWS hierarchy and HRF saliency maps confirms that the details in the helicopter are much more efficiently highlighted in the latter. The process thus helps us obtaining representations of images emphasizing shared objects between them, and with the desired level of detail. Note that such a method could highly be of benefit for approaches making use of hierarchical segmentation for co-saliency detection such as [Liu et al. (2014)], by producing hierarchical segmentations that already retain contours of shared objects between images, in a robust manner (since the keypoints and descriptors used are robust to image transformation, occlusions etc.).

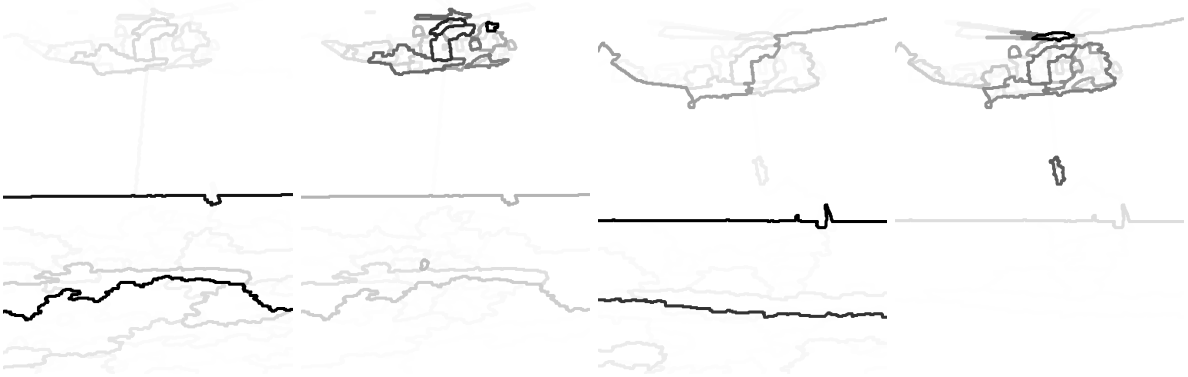


(a) $\mathbf{I}_1$ (b) $\mathbf{I}_2$ (c) $\mathbf{I}_3$ (d) $\mathbf{I}_4$ (e) $\mathbf{I}_5$



(f) All matched key-points of $\mathbf{I}_3$ with other images key-points

(g) Prior for $\mathbf{I}_3$: probability map generated thanks to morphological distance function



(h) Volume-based SWS Hierarchy UCM for I_3

(i) Volume-based HRF UCM for I_3 with matched key-points as prior.

(j) Volume-based SWS Hierarchy UCM for I_4

(k) Volume-based HRF UCM for I_4 with matched key-points as prior.

Figure 7.13 – Hierarchical co-segmentation of matched objects.

7.5.6 Effect of the HRF highlighting transitions between foreground and background

We illustrate in this section the HRF highlighting transitions between foreground and background presented in section 7.4. We present its effect in the face detection example in figure 7.14, and its effect in the weakly-supervised HRF example in figure 7.15. One can notice that insisting on the transition between background and foregroud helps to better delineate the contours corresponding to these transitions in both examples. In figure 7.14(k), we thus observe how the hands and face of the character emerge first. Similarly, in figure 7.15(e)(f), we see that

(a) Original image

(b) Case 1: UCM of HRF depending on the couples of regions (section 7.4).

(c) Case 2: UCM of HRF depending on the regions only (section 7.2).

(d) Case 3: Both combined.

(e) Case 1: 4 regions.

(f) Case 1: 10 regions.

(g) Case 1: 25 regions.

(h) Case 2: 4 regions.

(i) Case 2: 10 regions.

(j) Case 2: 25 regions.

(k) Case 3: 4 regions.

(l) Case 3: 10 regions.

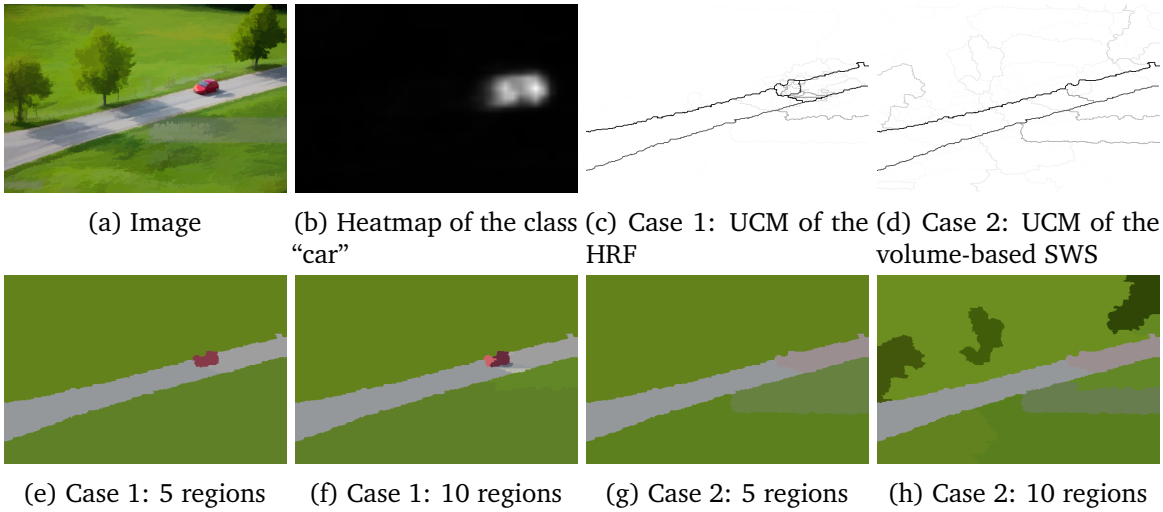
(m) Case 3: 25 regions.

Figure 7.14 – Hierarchical segmentation of faces highlighting transitions between background and foreground.

the car structure emerges in early levels of the hierarchy, whereas we do not obtain such a result otherwise.

7.6 Conclusion and Perspectives

We have proposed a novel and efficient hierarchical segmentation algorithm that emphasizes the regions of interest in the image by using spatial exogenous information on it. The wide variety of sources for this exogenous information makes our method extremely versatile and its potential applications numerous, as shown by the examples developed in the last section.



(a) Image (b) Heatmap of the class "car" (c) Case 1: UCM of the HRF (d) Case 2: UCM of the volume-based SWS

(e) Case 1: 5 regions (f) Case 1: 10 regions (g) Case 2: 5 regions (h) Case 2: 10 regions

Figure 7.15 – Hierarchical segmentation of the main class in the image (class "car") highlighting transitions between background and foreground.

To go further, we could find a way to efficiently extend this work to videos. One could also imagine a semantic segmentation method that would go back and forth between localization algorithm and HRF to progressively refine the contours of the main objects in the image.

Chapter 8

Combining Convolutional Neural Networks With Morphological Hierarchical Segmentations

Résumé

Au cours des dernières décennies, l'analyse et le traitement d'images ont été fortement impactées par l'émergence de méthodes fondées sur l'apprentissage. En particulier, les techniques utilisant les réseaux de neurones convolutionnels (en anglais *Convolutional Neural Networks*, ou CNN) ont resurgi [Krizhevsky et al. (2012)], du fait de l'explosion du nombre de données annotées disponibles et de la puissance de calcul nécessaire pour les traiter. Les CNN fournissent l'état de l'art dans beaucoup de domaines de la vision par ordinateur. Dans ce chapitre, nous ouvrons quelques perspectives sur la manière dont les filtrages ou segmentations morphologiques pourraient se combiner avec les CNN pour en améliorer les performances.

Abstract

Over the past decades, image analysis and image processing have been strongly impacted by the emergence of learning-based methods. In particular, convolutional neural networks (CNN) techniques have resurfaced [Krizhevsky et al. (2012)] because of the explosion of the number of annotated data available and of the computing power necessary to handle them. CNNs nowadays provide the state of the art in many areas of computer vision. In this chapter, we open some perspectives on possible combinations of CNN with morphological filtering or segmentation in order to improve their performances.

8.1 Introduction

In the last decades, image processing and image analysis have been highly impacted by the emergence of learning-based methods for a large number of tasks. Historically, several machine learning models, like SVM [Boser et al. (1992); Cortes et al. (1995)] or random forests [Breiman (2001)], have been used in combination with handmade features [Mikolajczyk et al. (2005)] and applied with success to a variety of problems, for example face detection [Viola et al. (2001)], human detection [Dalal et al. (2005)], image classification [Csurka et al. (2004)] or object tracking [Zhou et al. (2009)].

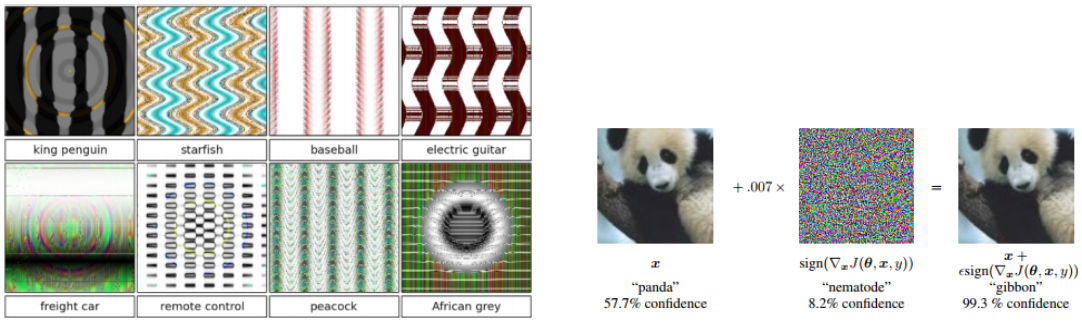
Recently, a paradigm shift has been operated, arguing that task-specific features should be learned for better efficiency [Mairal et al. (2012)]. In this regard, the neural networks model has arisen again, fueled by the explosion of the amount of annotated data available [Deng et al. (2009); Lin et al. (2014)], the increase of computing power (namely GPUs) to handle them, as well as theoretical advances that resolved some of their inefficiencies. When applied to images, it makes use of convolutions to aggregate local information, and shares parameters, and we thus refer to *Convolutional Neural Networks* (CNN). The high number of parameters to adjust in such models also led to coin the expression *Deep Learning* to designate them. In CNNs, instead of using handmade features to discriminate between classes of elements in in a high-dimensional space, the model attempts to directly infer from the data the features that simplify this discrimination for a specific given problem by progressively adjusting its parameters using back-propagation to optimize a cost-function on the training set.

The intuition behind CNNs can be traced back to visual perception mechanism of animals [Hubel et al. (1968)] and McCulloch-Pitts model [McCulloch et al. (1943)], proposed as *neocognition* by Fukushima in 1980 [Fukushima (1980)]. It is the first computational model using local connectivity between neurons in a hierarchical structure on images. Later, LeCun *et al.* [LeCun, Denker, et al. (1990)] obtained state-of-the-art performances on handwritten character recognition by means of a multilayer CNNs. Recently, researchers have developed more refined methods to overcome main difficulties in training complex CNN architectures. Especially, AlexNet [Krizhevsky et al. (2012)] is considered as a milestone in the design and training of CNNs due to its excellent performance in large-scale image classification in the well-known ImageNet Challenge [Deng et al. (2009)]. It includes two novels components, *ReLUs* (Rectified Linear Units) instead of $\tanh$ units, which makes the training processes several times faster; and *Dropout* of coefficients, which is proved to be very effective in diminishing overfitting. In the last five years, several variants including VGGNet [Simonyan et al. (2015)], GoogLeNet [Szegedy et al. (2015)] and ResNet[He, Zhang, et al. (2016)], have been designed to further improve the performance of CNNs on different visual recognition tasks.

Despite these impressive results, CNNs present certain limitations [Marcus (2018)]. We expose some of these limitations and draw some perspectives on how morphological hierarchical segmentations may complement CNNs.

8.2 Making the CNN more robust to noise using levelings

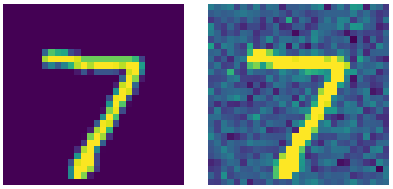
Many state-of-the-art CNN architecture for image classification [Krizhevsky et al. (2012); Simonyan et al. (2014); Szegedy et al. (2015)] have been trained and tested on noise-free images such as the ones from the ImageNet database [Deng et al. (2009)]. However, it turns out that such models are non-robust when noise is added to the input image. Indeed, it has been shown that CNN can be fooled by carefully designed images that are unrecognizable for the human eye, but falsely recognized despite a high confidence by the CNN [Nguyen et al. (2015)], as illustrated in figure 8.1(a). Conversely, it has also been shown that adding a specific noise to an image using an adversarial network, with as a goal to mislead the classifier, can lead to classification results that are grossly incorrect for images that are easily recognized by the human eye [Goodfellow et al. (2014)], as one can see in figure 8.1(b).



(a) (b)

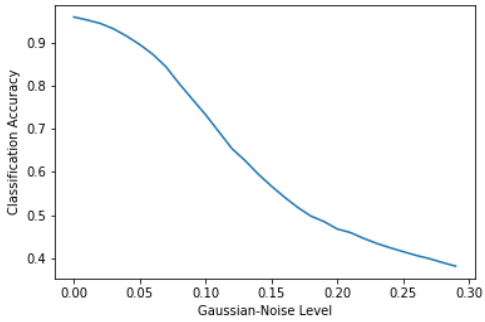
Figure 8.1 – Illustration on adversarial attacks that fool CNN. (a) Carefully designed unrecognizable images can be believed to be a familiar object by a CNN with a certainty superior to 99% (excerpt from [Nguyen et al. (2015)]). (b) Conversely, adding noise designed by an adversarial network can lead a CNN to be completely wrong with high confidence for an image that is easily recognizable by the human eye (excerpt from [Goodfellow et al. (2014)]).

However, even simple noise such as a Gaussian one, can lead to results being very wrong. In figure 8.3, we thus represent as an example the percentage of good classification of the MNIST dataset [LeCun, Bottou, et al. (1998)] images by a CNN, with respect to the amount of Gaussian noise added on the image to classify. One can see that the addition of noise can severely mislead the CNN. One potential solution to this problem would be to add, during the CNN training, images presenting such noise. This is the type of approach privileged by adversarial networks, in which it is proposed to train the classifier to be robust to noisy images, while an adversarial network attempts at designing a type of noise that can mislead the classifier [Goodfellow et al. (2014)]. However, this appears to be a never-ending process, as we cannot know with certainty that the classifier will be able to cope with a given type of noise. This is why we propose an other research path to make the CNN more robust to adversarial attacks.



(a) Image example

(b) Image with added Gaussian noise

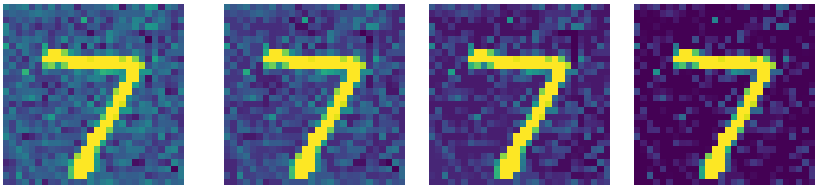


(c) Evolution of the classification accuracy with the amount of added noise

Figure 8.2 – The more Gaussian noise is added to the test image to classify, the less accurate the classifier that has been trained on the MNIST database.

The leveling is a classical mathematical morphology filtering technique, that has previously been introduced in section 5. Its main interest resides in the fact that it does not create any contour in the image while it simplifies and denoises it. On the contrary, levelings extend flat or lambda-flat zones, and thus applying them, associated to a series of rougher and rougher markers, has the effect of progressively simplifying the image while keeping its main contours intact for the most time, as was exposed in section 5.

We hereby apply alternate sequential filtering (ASF) using levelings to noisy images before classifying them with a CNN trained to classify images of the MNIST database. More precisely, we generate a hierarchy of levelings $\mathcal{H}_{ASF-lev}$ with the alternate sequential model using as markers erosions and dilations of increasing sizes (see section 5 of chapter 3 for more details). We then compare the classification results over the whole MNIST database with and without this filtering, for different levels of the hierarchy. The results are very encouraging for certain levels of the hierarchy (linked with the width of the characters to recognize), as we can see a major enhancement of the classification score when the noise is important. Levelings and hierarchies of levelings, thus appear to be interesting tools to use to obtain more robust images to feed the CNN architectures with.

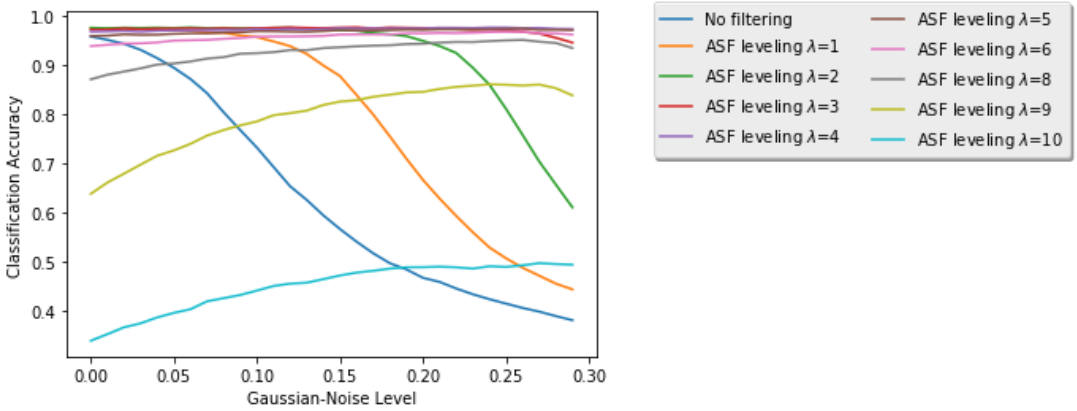


(a) Image with added Gaussian noise

(b) $\mathcal{H}_{ASF-lev}$: $\lambda = 1$

(c) $\mathcal{H}_{ASF-lev}$: $\lambda = 3$

(d) $\mathcal{H}_{ASF-lev}$: $\lambda = 5$



(e) Evolution of the classification accuracy on MNIST dataset for images with added Gaussian noises, with respect to the amount of added noise, and for different levels of a hierarchy of flat levelings obtained following an alternate sequential construction.

Figure 8.3 – The filtering of the noisy image using levelings increases classification accuracy. However, for a level of ASF levelings hierarchies that simplifies the image too much, the classification accuracy begins to decrease.

Several new research directions can be thought of based on this experiment:

— One can try to learn in a greedy feed-forward way, for a given CNN, the appropriate hierarchy of levelings, and within this hierarchy the leveling that makes the CNN the more robust to adversarial attacks, in a similar way that what has been proposed in chapter 5.

— To go further, one may want to incorporate leveling filters in the CNN, in order to be able to directly infer from given data the best possible leveling to apply to make the system more robust to adversarial attacks. Unfortunately, backpropagation requires differentiable operators. This is not the case of morphological operators in general, and levelings in particular. An interesting research direction thus consists in finding differentiable approximate functions having similar effects than levelings, in order for them to be a direct component of the CNN training process.

8.3 Enriching the Feature Space: Application to Video Object Segmentation

8.3.1 Concept

In this section, we explore the possibility to enrich the input features spaces on which the objects of interest representation is learned using a CNN: instead of working on the gray-scale or (R, G, B) feature space as input space, one can indeed incorporate more layers to make this space richer.

In particular, we want to study the impact of enriching the CNN input features space with given levels of morphological hierarchical segmentations. Instead of taking as input 3 layers (R, G, B), we propose to study the effect of enriching the CNN input by stacking upon (R, G, B) several color images coming from different hierarchies and levels of these hierarchies. These additional images are of the form $(R_{(\mathcal{H}_k,\lambda_l)}, G_{(\mathcal{H}_k,\lambda_l)}, B_{(\mathcal{H}_k,\lambda_l)})$, with $k \in \{1, \dots, n\}$ and $l \in \{1, \dots, m\}$, given n normalized hierarchies $\{(\mathcal{H}_1, \boldsymbol{\lambda}_1), \dots, (\mathcal{H}_n, \boldsymbol{\lambda}_n)\}$ and m levels of their ultrametrics $\{\lambda_1, \dots, \lambda_m\}$. We call this process *input features space augmentation*.

To properly test this possibility, we operate on the semi-supervised video object segmentation problem, for which we can work with the *Densely Annotated Video Segmentation* (DAVIS) database and its given evaluation metrics [Perazzi, Pont-Tuset, et al. (2016)].

In the next section, we begin by properly introducing the video object segmentation problem and its difficulties. Then we detail the method we propose to address these difficulties. Finally, we present qualitative and quantitative results, and evaluate the impact of the feature space enrichment on obtained results.

8.3.2 Video Object Segmentation

Video object segmentation is a two classes labeling problem aiming at separating foreground objects from the background region of a video. This bipartition of the video is a spatio-temporal two-classes segmentation which is fundamental to several applications including action recognition, object tracking, video summarization, and cooperative multi-camera video processing.

Depending on the considered scene and the goal of the user, this "foreground" object may vary and multiple valid solutions are conceivable. Indeed, the choice of the object to follow across the video is highly dependent on the application. This is why we hereby present a semi-supervised approach that uses sparse user-given cues to produce the video object segmentation. These cues can be masks specified on one or a few key frames, in which case the problem is a *mask propagation* one. More simply, they can consist in a sparse set of labels given by the user. In our case, we align with evaluation criteria from the DAVIS database [Perazzi, Pont-Tuset, et al. (2016)], and thus only use the mask of the object of interest given for the first frame of the video. The difficulties associated with this problem are multiple, and are listed in figure 8.4, excerpt from [Perazzi, Pont-Tuset, et al. (2016)].

ID	Description
BC	*Background Clutter*. The back- and foreground regions around the object boundaries have similar colors (χ^2 over histograms).
DEF	*Deformation*. Object undergoes complex, non-rigid deformations.
MB	*Motion Blur*. Object has fuzzy boundaries due to fast motion.
FM	*Fast-Motion*. The average, per-frame object motion, computed as centroids Euclidean distance, is larger than $\tau_{fm} = 20$ pixels.
LR	*Low Resolution*. The ratio between the average object bounding-box area and the image area is smaller than $t_{lr} = 0.1$.
OCC	*Occlusion*. Object becomes partially or fully occluded.
OV	*Out-of-view*. Object is partially clipped by the image boundaries.
SV	*Scale-Variation*. The area ratio among any pair of bounding-boxes enclosing the target object is smaller than $\tau_{sv} = 0.5$.
AC	*Appearance Change*. Noticeable appearance variation, due to illumination changes and relative camera-object rotation.
EA	*Edge Ambiguity*. Unreliable edge detection. The average ground-truth edge probability (using [11]) is smaller than $\tau_e = 0.5$.
CS	*Camera-Shake*. Footage displays non-negligible vibrations.
HO	*Heterogeneus Object*. Object regions have distinct colors.
IO	*Interacting Objects*. The target object is an ensemble of multiple, spatially-connected objects (*e.g.* mother with stroller).
DB	*Dynamic Background*. Background regions move or deform.
SC	*Shape Complexity*. The object has complex boundaries such as thin parts and holes.

Figure 8.4 – Challenges encountered when doing video object segmentation (excerpt from [Perazzi, Pont-Tuset, et al. (2016)]).

The method we propose takes advantage of the highly efficient and accurate representations of objects obtainable thanks to CNNs which are currently the state-of-the-art image classifiers, as long as a unified approach to learn a sequence of morphological filters from representations produced by CNNs denoted by CNN+MM. Our main contributions are:

— A video object segmenter that includes both a texture detector from a CNN and shape attributes from MM filters (section 8.3.5).

— An efficient morphological filters selection, which in practice complements and improves the performance of CNN in (sect. 8.3.5).

We have furthermore conducted a comprehensive empirical validation on the DAVIS video dataset (section 8.3.7) and report competitive results with state-of-the-art.

8.3.3 Related Work

Classical approaches

Graph-cuts Video Segmentation Images and videos can be represented by a graph structure in which edges connect neighboring objects, be it pixels for images or voxels for video sequences. The video segmentation problem can then be seen as an optimization one, aiming to assign

coherent labels to nodes while complying with an object model or user constraints. Graph-cuts techniques have been proposed to solve this problem for video segmentation.

Using this framework, one can lower the computation cost of graph-cuts techniques by operating regionwise instead of pixelwise, thus reducing the number of nodes in the graph. This can be done using clustering techniques such as spatio-temporal superpixels (supervoxels) [Xu and Corso (2012)] [Chang et al. (2013)] that oversegment the video into spacetime homogeneous and perceptually distinct regions, or per-frame watershed algorithm as in [Li et al. (2005)] and [Price et al. (2009)].

Interactive video segmentation based on hierarchies of segmentations In a similar fashion, some works have been conducted that make use of hierarchies of segmentations for video object segmentation [Marcotegui et al. (1999); Zanoguera (2001)]. The specifics of these methods are:

1. Computation of a hierarchy of partitions on the first frame and interactive segmentation with markers specified by the user.
2. Projection of a partition from one frame to the next one, based on change detection followed by motion analysis: (i) camera motion estimation and compensation, (ii) scene-cut detection: if a scene-cut is detected, all parameters of the segmentation algorithm are reset to their initial values and the user is asked to provide markers for a new segmentation (cf. step 1), (iii) estimation of change-detection mask, (iv) uncovered background elimination, (v) contour adaptation to luminance edges.

Silhouettes/masks propagation Besides graph-based methods, several other solutions have been proposed to the video object segmentation problem. For example, optical flow and nearest neighbor fields can be used to propagate silhouettes or masks over multiple frames [Agarwala et al. (2004)],[Bai et al. (2009)],[Chuang et al. (2002)],[Fan et al. (2015)], [Lang et al. (2012)].

Probability foreground methods Video SnapCut [Bai et al. (2009)] uses local classifiers that predict the foreground probability, which are propagated and refined over time. A robust version has been introduced in [Price et al. (2009)]. Final binary segmentation is inferred using a conditional random field formulation.

Bilateral Space Bilateral filtering is a powerful tool used for edge-adhering image processing operations [Tomasi et al. (1998)]. In [Marki et al. (2016)], the authors propose an efficient graph-cut approach taking advantage of it. To do so, they use the *bilateral grid* introduced in [Chen, Paris, et al. (2007)], a structure aiming at speeding the bilateral filtering by projecting pixels into a higher-dimensional space based on position and color, but with a significantly lower resolution, with improved results in terms of speed and performances.

CNN-based segmentation methods

In a way, the efficiency of the use of bilateral space in [Marki et al. (2016)] is due to the fact that it constitutes a good representation space. Indeed, once the object pixels have been projected into this space, they are easier to recognize in the subsequent frames. In the same spirit, we propose to use CNNs to learn for each object that we want to follow in the video its representation. This approach has been inspired by the numerous segmentation methods that emerged in the last years. Other works have a similar approach.

One-Shot Video Object Segmentation [Caelles et al. (2017)] The specifics of this method are:

1. Departure from a CNN trained on ImageNet, called *base network*.
2. The use of transfer learning using images+label from DAVIS leads to the *parent network*.
3. Use of a Fast Bilateral solver to snap the background prediction to the image edges.
4. Learn a CNN to detect contours, then superpixels, compute Ultrametric Contours Map and eliminate low values.
5. Majority votes in superpixels from parent network.

Learning Video Object Segmentation from Static Images [Perazzi, Khoreva, et al. (2017)] The specifics of this method are:

1. Learning operates from $(R, G, B, MASK)$ to $MASK$.
2. Image augmentation on $(R, G, B, MASK)$ (which is classical) and on $MASK$ only by elastic deformations, to simulate the movement of the object from one frame to the other.
3. Makes use of Optical Flow and box annotation.
4. Then to test the network for a new frame: given the mask estimate at time $t-1$, a dilation is applied and the resulting rough mask is used as input for object segmentation at time t.

Video Propagation Networks [Jampani et al. (2017)] The specifics of this method are:

— A bilateral filtering in temporal space is learned.

— A prediction model (of the regression type) is then used from previous mask and images to predict the next mask.

8.3.4 Method overview

In this section, we present our method to do semi-supervised video object segmentation. Given the segmentation ground truth on the first frame, we first learn an object representation of the object using a CNN as well as an appropriate morphological filters workflow to get a proper segmentation as an output of the CNN, as will be described in details in section 8.3.5. We can then apply this learned model to the object to track along the successive frames. Furthermore,

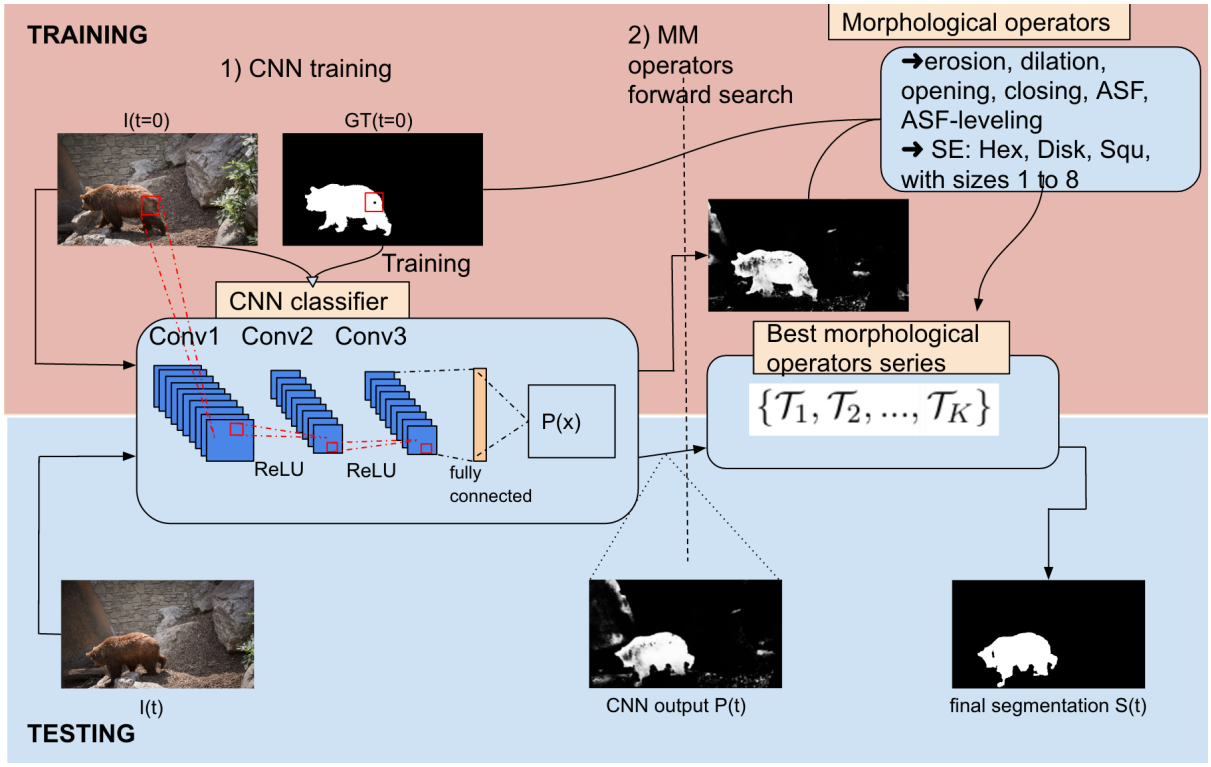


Figure 8.5 – An overview of our workflow. *Top*: A CNN is trained on the first frame. A forward selection of of the best sequence of morphological transformations. *Bottom*: For a new frame CNN+MM predicts the binarization

we also take advantage of the information redundancy between frames to ensure temporal coherency through them and to simplify the segmentation task, using for this levelings and in particular reconstruction openings. This will be detailed in section 8.3.6.

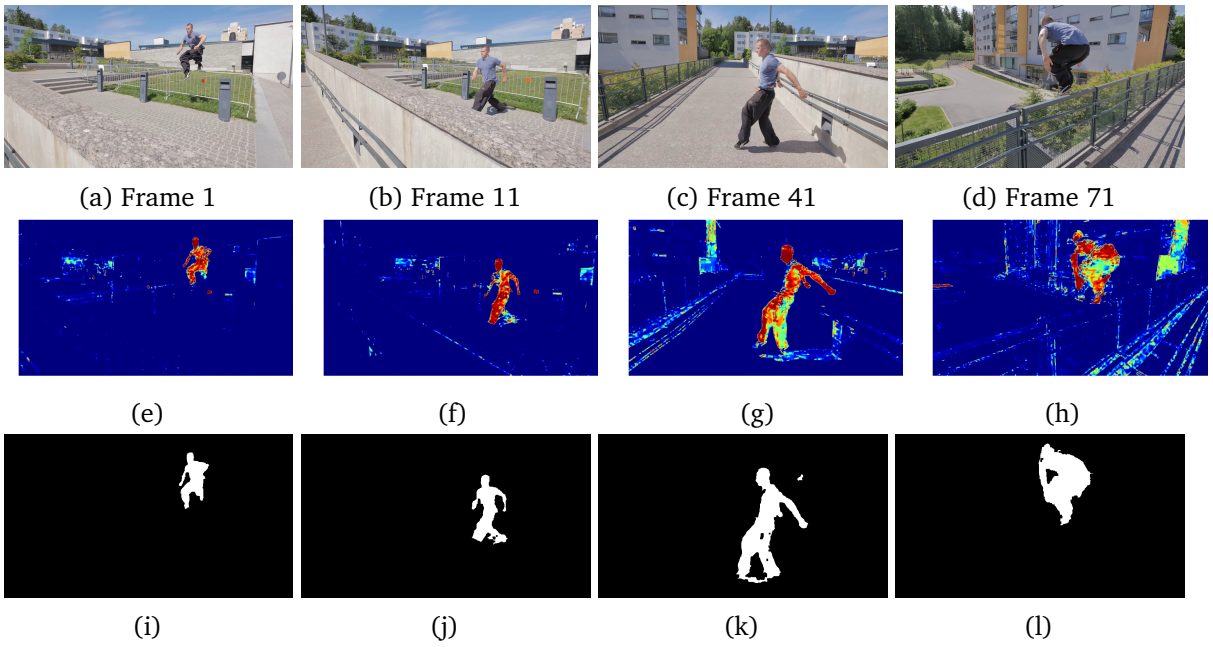
A sketch giving an overview of the global method can be found in figure 8.5, and an example of the obtained results is provided in figure 8.6. The two next sections expose the details of the method.

8.3.5 Learning phase

Representation Learning using Convolutional Neural Networks (CNN)

Given the first frame and the associated ground truth segmentation, we wish to train a classifier which, for each pixel of the image, outputs its probability to be in the object. To do that, we use a CNN.

In our case, the procedure is the following. For each pixel with a given ground truth value (e.g. 1 if in the object, -1 if not), a convolution window is extracted and weights of the links between CNN nodes are progressively refined using backpropagation. We have thus learned on the first frame a representation of the object. At test time, when fed with a new video frame, the CNN ouputs an image in which each pixel takes as value its probability to be in this object. This is illustrated in figure 8.7. The next step is devoted to producing a binary segmentation out of this gray-scale image.



(a) Frame 1 (b) Frame 11 (c) Frame 41 (d) Frame 71

(e) (f) (g) (h)

(i) (j) (k) (l)

Figure 8.6 – Qualitative video segmentation results on a sequence of DAVIS [Perazzi, Pont-Tuset, et al. (2016)].

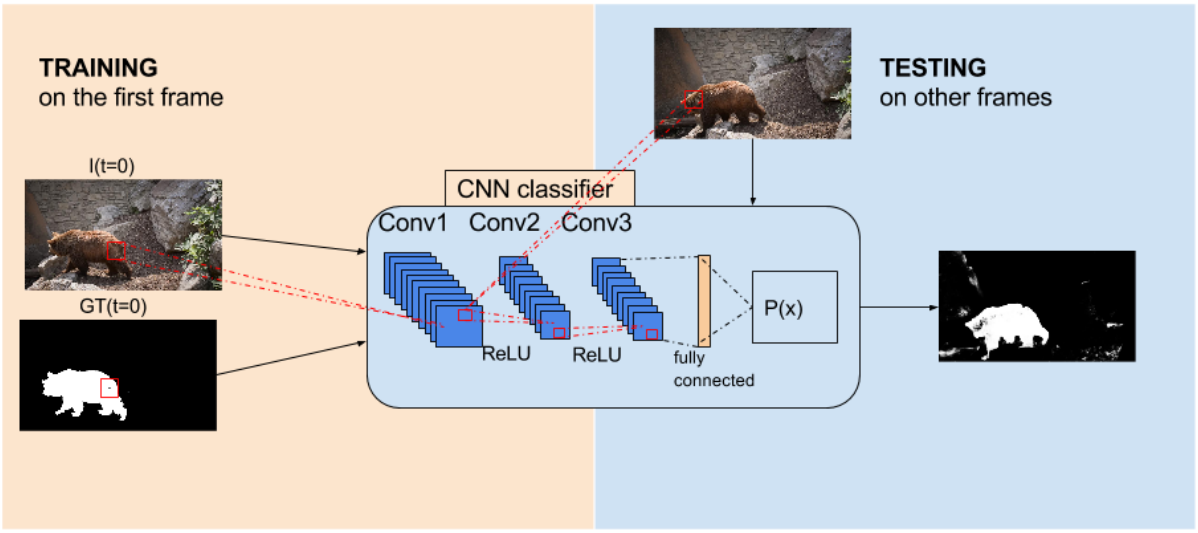


Figure 8.7 – A CNN is trained on the first frame. At test time, for a new frame, it outputs for each pixel its probability to be in the object.

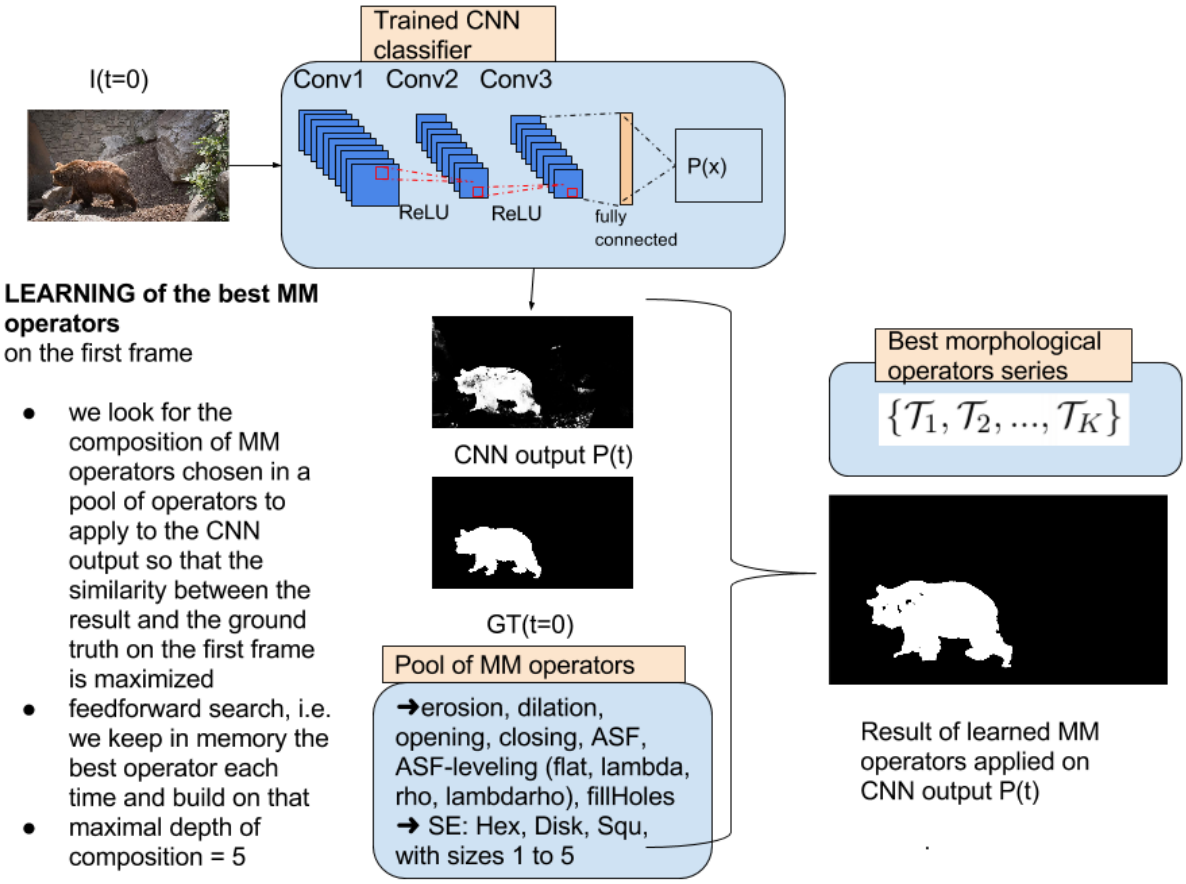


Figure 8.8

Automatic selection of morphological filters

To this effect, we learn on the first frame of the video the best composition of morphological filters to extract from the output of the CNN the best possible segmentation according to some error measure. In the experimental section, we fix the maximal composition depth of morphological filters to 5 and for the error measure we choose the Jaccard index J between segmentation result and ground truth. An overview of this process is depicted in figure 8.8.

Morphological filters are added to the detection workflow in a *forward stepwise* fashion, i.e. at each step we consider the best filter Φ of the image $\mathbf{I}$ to be included if and only if there is a considerable improvement according to the error measure J, given the ground truth segmentation $\mathbf{Y}$. If there is not a filter that reduces the error, we terminate the process.

$$\max_{\lambda} \mathrm{J}(\mathbf{Y}, \Phi(\mathbf{I}) \geq \lambda) \tag{8.1}$$

Notice that since the morphological operators we use operate on gray-scale images, at each step we select the best filter Φ and threshold λ according to the error measure J, but only keep in memory the best filter and let the threshold change as the composition of filters gets deeper.

The considered morphological filters are the following, with structuring elements of sizes 1 to 5:

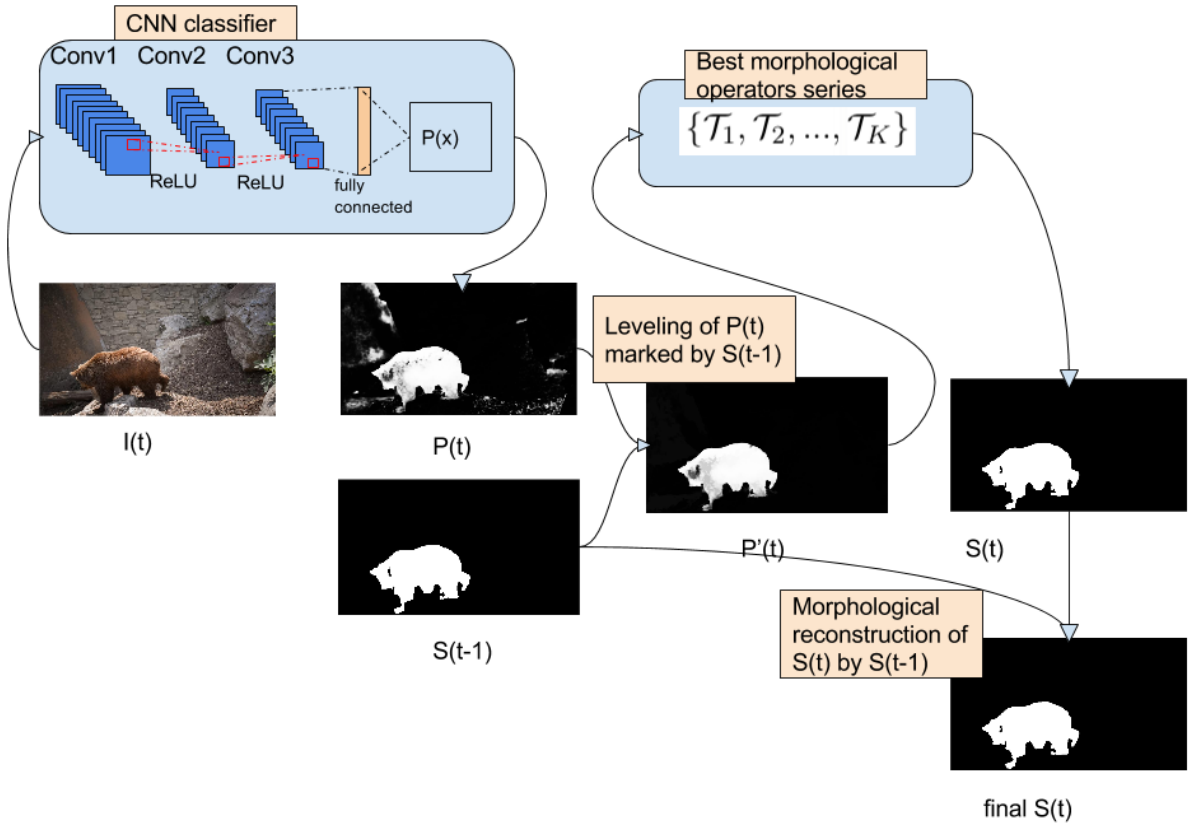


Figure 8.9 – Workflow of this temporal coherence enforcement.

— Erosion and dilation

— Opening and closing

— Alternate Sequential Levelings

— Fill holes functions

We are thus provided with an adaptive way to filter CNN output images and get binary segmentations out of them. But each time the CNN is tested on a new video frame, false positives can appear due to the appearance of a new object or a change in the background, as well as false negatives due to object deformations or occlusions. To handle these issues, we need to take advantage of the information redundancy between frames.

8.3.6 Temporal Coherence via Leveling and Morphological Reconstruction

This is why we extend the single frame method to video volumes by adding an additional step of morphological filtering. The workflow of this temporal coherence enforcement can be found in figure 8.9.

First we use a leveling to filter the results of the CNN output in the temporal dimension. Levelings simplify image in suppressing details without displacing or blurring the contours of the remaining structures. Spatio-temporal filtering should follow the motion of points between frames.

Let us thus consider two consecutive frames of the image $I(t)$ and $I(t+1)$, and assume we have a segmentation $S(t)$ of $I(t)$ and wish to use this information to get a segmentation $S(t+1)$ of $I(t+1)$. Then we simply begin by filtering the CNN output $P(t+1) = CNN(I(t+1))$ by applying a leveling to it, using $S(t)$ as marker function, and obtain $\hat{P}(t+1)$. We then apply the learned MM filters on $\hat{P}(t+1)$ as described in section 8.3.5, giving us $S(t+1)$. This prevents incorrectly averaging information across object boundaries and eliminate false detection on single frames.

Furthermore, to tackle the false negatives due to object deformations, we use a morphological reconstruction ρ to reconstruct the object in $S(t+1)$ using $S(t)$ as a marker, that we denote $\rho_{S(t+1)}(S(t))$ [Vincent (1993)]. Since it can lead to a disappearance of the object in case of object occlusion, we however introduce a safeguard condition at this step:

— $\hat{S}(t+1) = \rho_{S(t+1)}(S(t))$

— $s = \frac{area(\hat{S}(t+1) \cap S(t))}{area(S(t))+1}$, with $area$ the function returning the number on non-zero pixels in an image

— Then:

$$\begin{cases} S(t+1) \leftarrow \hat{S}(t+1) \text{ if } s > 0.5 \\ S(t+1) \text{ remains unchanged otherwise} \end{cases}$$

8.3.7 Results and conclusions

In order to evaluate the method, we conducted extensive experiments on the DAVIS dataset. The scores that are evaluated are the ones from [Perazzi, Pont-Tuset, et al. (2016)], namely the Jaccard index J, the F-score F and a temporal stability score T.

In the table above, the methods that we implemented, called OUR_1 and OUR_2, both followed the process described in sections 8.3.5 and 8.3.6, only differing by the input feature space:

— OUR_1 only uses the (R, G, B) channels at training and test time.

— OUR_2 uses as input features of the CNN, both a training and test time, a stacking of (R, G, B) and $(R_{(\mathcal{H}_{ASF-lev}, \lambda_l)}, G_{(\mathcal{H}_{ASF-lev}, \lambda_l)}, B_{(\mathcal{H}_{ASF-lev}, \lambda_l)})$ for $l \in \{1, \ldots, m\}$ levels of a hierarchy of alternate sequential flat levelings $\mathcal{H}_{ASF-lev}$.

Several remarks can be made about these results. Tests of input features space augmentation using watershed hierarchies did not lead to an improvement of results, and this is why they are not presented in the table. This is probably due to the fact that such hierarchies are computed separately on each frame of the video, which leads to results potentially highly varying from one frame to the next one. In order to apply this approach with these hierarchies, one should first ensure a better coherence between the hierarchies of segmentation produced in each frame. However, even then, the watershed transform can present some natural instabilities, as slight modifications of the catchment basins can lead to changes in the regions order of fusions.

Features augmentation using ASF levelings, on the other hand, led to slight improvements of the results, as one can see in table 8.1 and provided competitive results with the state-of-the-art of late 2016.

Method	J Mean ↑	F Mean ↑	T ↓
BVS [Marki et al. (2016)]	0.665	0.656	0.316
FCP[Perazzi, Wang, et al. (2015)]	0.631	0.546	0.285
JMP[Fan et al. (2015)]	0.607	0.586	0.131
HVS[Grundmann et al. (2010)]	0.596	0.576	0.296
SEA [Ramakanth et al. (2014)]	0.556	0.533	0.137
TSP [Chang et al. (2013)]	0.358	0.346	0.329
OUR_1	0.711	0.735	0.566
OUR_2	0.717	0.736	0.561
OSV [Caelles et al. (2017)]	0.798	0.806	0.376
OSV -BS[Caelles et al. (2017)]	0.774	0.781	0.335
OSV -PN-BS[Caelles et al. (2017)]	0.646	0.667	0.609
OSV -OS-BS[Caelles et al. (2017)]	0.525	0.477	0.538
OSV -PN-OS-BS[Caelles et al. (2017)]	0.176	0.203	0.460
MaskTrack [Perazzi, Khoreva, et al. (2017)]	0.748	XXX	XXX
MaskTrack +Flow [Perazzi, Khoreva, et al. (2017)]	0.784	XXX	XXX
MaskTrack +Flow+CRF [Perazzi, Khoreva, et al. (2017)]	0.803	XXX	XXX
VPN-DeepLab [Jampani et al. (2017)]	0.750	0.724	0.295
VPN-Stage2 [Jampani et al. (2017)]	0.713	0.689	0.302

Table 8.1 – Quantitative comparison against state-of-the-art method

However, doing such a features augmentation has the effect of slowing down the training and testing phases. It is thus probably not useful when transfer learning can be operated [Yosinski et al. (2014)], which consists in reusing CNN weights learned on a big dataset for a specific classification problem, and finetuning them for a new problem. Indeed, usually CNNs that have been trained on million of examples take (R, G, B) images as input, and modifying the input data shape is not straightforward. But using levels of morphological hierarchies could be useful to do data augmentation, to artificially increase the size of a dataset for more efficient learning.

More generally, our approach to do video segmentation is highly perfectible, and could take inspiration in methods presented in section 8.3.3, notably by doing transfer learning, by operating elastic deformations on the mask image to simulate objects movements or by doing data augmentation with specific changes of the background to make the CNN more robust to the appearance of new objects. However, the exploration of these research paths, while interesting, is beyond the scope of this thesis subject, and could by themselves be the subject of an important research work.

8.4 Image characterization by global and local hierarchical features

8.4.1 Global image features: distances between hierarchies

Throughout this thesis, we have explored several ways to compute and combine morphological hierarchies. In particular, we have seen that we may generate multiple hierarchies upon the

same fine partition of an image. In chapter 6, we have seen how using distances between such hierarchies as images features led to interesting and promising results on image classification problems, both in terms of learning efficiency and interpretability of the results.

We have exposed some examples for supervised image classification, but we would like to put the emphasis here on the fact that such an approach would be interesting to analyze an image in an unsupervised way. For a pool of *pure* hierarchies that have understandable effects and characterize the images in a controlled way, the analysis of the interhierarchies distances can provide insights about unknown types of images, and possibly help us discriminate between classes of images in an unsupervised way. Furthermore, it could be used for example for anomaly detection, as outliers in a dataset would react differently that other images to the application of the same hierarchies. In this regard, this approach is in line with scale-space and granulometrics analysis, since objects are characterized by various hierarchies that reflect specific properties, similar to series of sieves filtering differently the contents of a mixture.

8.4.2 Local images features: contours signatures

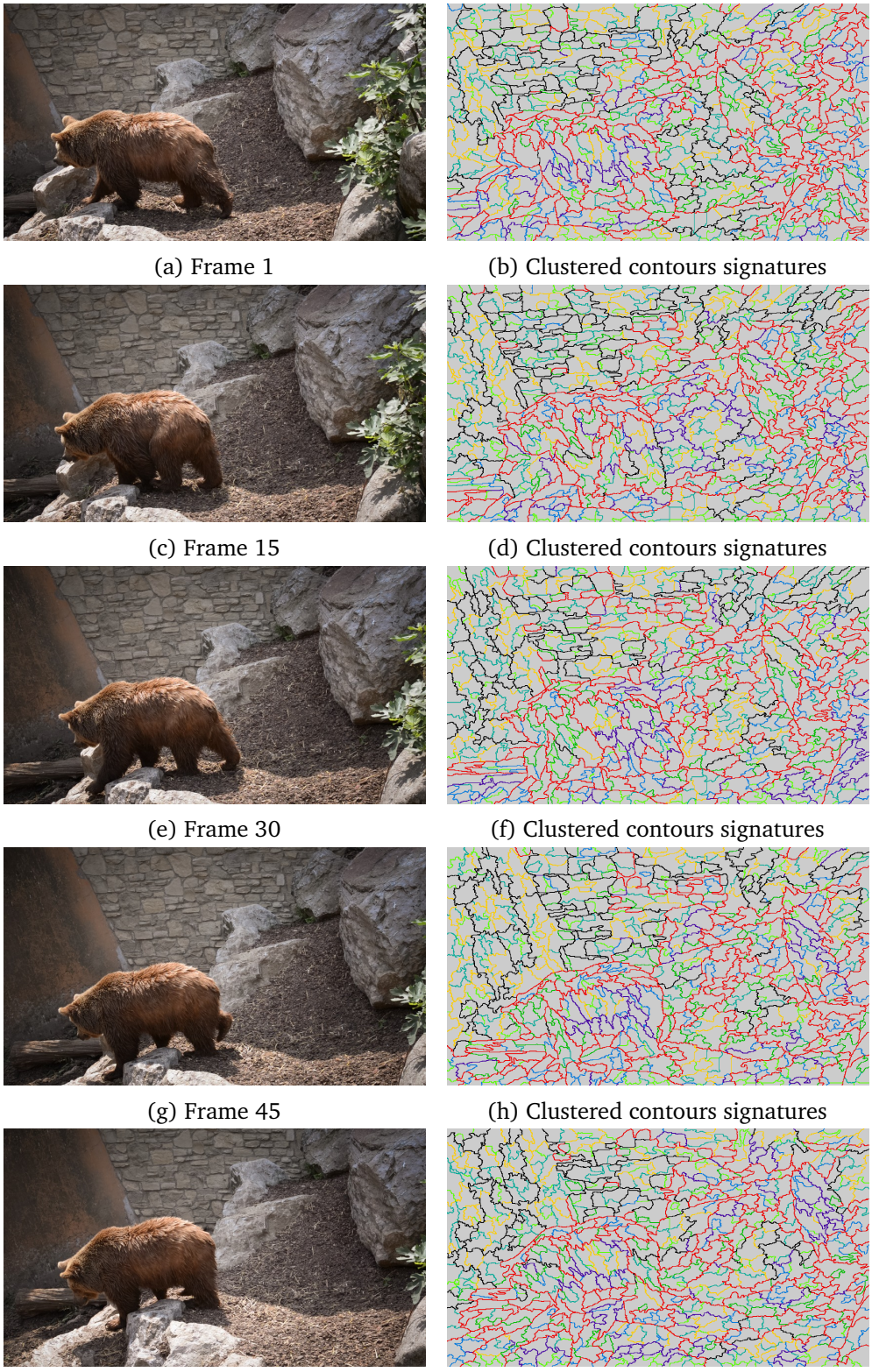
In this section, we expose how one could, using the same principles, obtain a local image feature for each contour of an image. When computing several hierarchies for the same fine partition of an image, for each contour, each hierarchy provide us with several saliency values (that we can normalize in order for them to be comparable). For each contour, we can this way obtain a local descriptor in the form of a vector of saliency values, that we call *signature* of the contour.

To illustrate the discriminative power of such contours descriptors, we conduct the following experiment. For several images of the same object in different frames of a video, we generate contours signatures for all contours of the computed fine partitions of those images, with considered hierarchies being $(\mathcal{H}_{trivial}, \mathcal{H}_{surf-SWS}, \mathcal{H}_{vol-SWS}, \mathcal{H}_{waterfall}, \mathcal{H}_{BSC}, \mathcal{H}_{surf-SWS} \circ \mathcal{H}_{vol-SWS}, \mathcal{H}_{vol-SWS} \circ \mathcal{H}_{surf-SWS})$. On the first image, we train a K-means to learn in an unsupervised way a clustering of the contours based on these signatures representations, with $K = 8$. We then apply this learned model to the next images, and display for all images the contours associated with each class with different colors. A visual inquiry of the results reveals a strong correspondence between the type of retrieved contours for each class, which likely confirms the potential usefulness of contours signatures to characterize contours. This is illustrated in figure 8.10.

We do the same experiment with stereo images excerpt from the Middlebury dataset [Scharstein et al. (2014)]. The results are very promising and suggest a possible use of these contours signatures to find proper contours to be a basis for the completion of a disparity map, as illustrated in figure 8.11.

A second attempted experiment consisted in learning the contours of interest on an image, by integrating the contours ground-truth over each contour. This way, for each contour of the ground-truth, we have a value between 0 and 1 which reflects its pertinence. We can then

attempt to learn the type of contours signatures that best capture such contours of interest in subsequent images. Obtained results are mixed, as it turns out this problem is a very unbalanced machine learning problem, with much more negative than positive examples.



(a) Frame 1 (b) Clustered contours signatures

(c) Frame 15 (d) Clustered contours signatures

(e) Frame 30 (f) Clustered contours signatures

(g) Frame 45 (h) Clustered contours signatures

(i) Frame 65 (j) Clustered contours signatures

Figure 8.10 – Qualitative visualization of the pertinence of images signatures for several frames of a video sequence. We train a K-means with $K = 8$ on the contours of the first frame and apply it to the subsequent ones. The contours colors represent the class they have been attributed by the K-means.

(a) *Teddy-1* (b) Clustered contours signatures

(c) *Teddy-2* (d) Clustered contours signatures

(e) *Cones-1* (f) Clustered contours signatures

(g) *Cones-2* (h) Clustered contours signatures

Figure 8.11 – Qualitative visualization of the pertinence of images signatures for stereo images. The two examples *Teddy* and *Cones* are excerpt from the Middlebury dataset [Scharstein et al. (2014)]. We train a K-means on the contours of the first image with $K = 8$ and apply it to the next one. The contours colors represent the class they have been attributed by the K-means.

8.5 Exploration of Hierarchies for Finer Contours Retrievals

An intrinsic complementarity seems to exist between mathematical morphology, which studies shapes in images, and CNN-based methods which seem mostly learn textures and local patterns. In particular, numerous CNN-based methods provide state-of-the-art for semantic segmentation, object detection, etc. The results are very good to retrieve the proper pixel class for most classes, but these methods generally have difficulties to obtain precise contours of the objects. Several methods have been proposed to complement them and solve this problem, such as Conditional Random Fields [Zheng et al. (2015)] or Fast Bilateral Solver [Barron et al. (2016)].

We briefly explored an other possibility that would require further inquiry. The main idea is to build a hierarchy that takes into account a prior spatial information coming from a localization/semantic segmentation method. In the spirit of [Drouyer et al. (2017)], one can then explore this hierarchy in a top-down fashion, with a given stop condition to assess that the proper level of fineness has been attained. Notably, one can build a hierarchy that put the emphasis on the transitions between the foreground (the object of interest) and the background, which precisely correspond to the contours who are uncertain and that we would like to precisely retrieve. Then, we can explore this hierarchy in a bottom-up fashion, with stopping conditions, to retrieve new zones of interest that are more precise. This way, we have an exploration space of contours that is adapted to the areas of uncertainty. Preliminary experiments have been conducted a first result is presented in figure 8.12.

(a) Image

(b) Rough probability map associated with the class *face*

(c) Refined probability map with contours extracted from a HRF

Figure 8.12 – Example of the interest of traversing a HRF taking into account a prior spatial information to refine its contours.

Conclusion

Summary of our main contributions

A brief overview of our contributions is provided below.

— We introduce several ways to combine hierarchies, in a sequential way (by hierarchical chainings) or parallel by functional combinations of ultrametrics. In particular, combinations of SWS hierarchies are interesting as they operate in a homogeneous framework, the strength of the contours being expressed by probabilities, and can provide understandable and coherent contours valuations. Furthermore, we have shown that hierarchies sharing the same MST may be combined by restricting the computation of the new ultrametric distance to this MST. The family of operators applicable with this method is extremely large.

— We introduce a methodology to study the space of hierarchies. It is based on the use of the Gromov-Hausdorff distance between hierarchies, that we show to be extremely simple to compute for hierarchies generated upon the same fine partition. This way, we can quantify the distances between hierarchies, and estimate the degree to which they differ. Using this distance in conjunction with dimensionality reduction techniques provides a comprehensive framework to study the relations between different hierarchies built on the same image.

— We propose a segmentation framework to automatically learn the best hierarchy and cut-level, given a set of images and a score to quantify the quality of segmentations.

— We show that inter-hierarchies distances matrices can constitute powerful geometric features of images. Using them in a classical classification pipeline leads to promising results, as the results exhibit a "aha-moment" during the learning phase that is more linked with the human way of learning as compared with Convolutional Neural Networks, which require a lot of examples to reach convergence.

— Using regionalized random germs for constructing SWS permits to incorporate any spatial prior information during the hierarchical segmentation process. The versatility and usefulness of such a method is illustrated on several examples.

— One can also incorporate in their construction information about shapes, preferential directions, or sizes distribution.

— The thesis introduces the Binary-Scale Climbing (BSC) hierarchy. It is created by minimizing a Mumford-Shah functional subordinated to a given input hierarchical segmentation. The resulting hierarchy has been regularized as a trade-off between boundary length and image adherence. It permits to explore the evolution of the partitions derived from a given hierarchy if one displaces the cursor between fidelity to the data and length of the contours. Applied to distinct hierarchies obtained for the same image, it acts as another kind of normalization in order to compare them.

— In order to implement all these methods, a module for hierarchical segmentation has been implemented within the open-source library Smil. This implementation has been coded using dynamic programming and graphs in C++, and can be called in Python in order to be used with the numerous libraries and frameworks (such as numpy, OpenCV, pandas, scikit-learn, etc.) that exist in this language to work with images and data. It has been made public and can now be used, modified and enhanced by the research and industrial communities. This part of the work is hidden behind the results outlined in this thesis but has represented a significant programming effort.

Perspectives

Several promising perspectives can be thought of, both in terms of methodology and applications.

— We remind the reader that the chapter 8 has presented several perspectives on how morphological hierarchical segmentations may be combined with CNN-based methods.

— Most of the hierarchical segmentation methods introduced in this thesis make a heavy use of the MST structure. We have seen that the MST is generally not unique (sections 1.3.3 and 3.3.5) and have outlined possible solutions in section 3.3.5 to avoid an arbitrary choice of one of them, namely to consider all possible MST at once before pruning the output, or to select one MST based on the lexicographic distance rather than ultrametric distance. Tests should be conducted to validate and complete these proposals.

— In chapter 4, we propose a methodology using Gromov-Hausdorff distance between hierarchies as long as dimensionality reduction techniques to structure and visualize the combinatorial space of hierarchies. More specifically, we make use of *metric* multidimensional scaling (MDS) to project the space of inter-hierarchies distances into a space of dimension 2 and visualize hierarchies relatively to one another. Another variant of MDS, called *non-metric* MDS, is useful in cases where dissimilarities are known only by their rank order, and where the spacing between successively ranked dissimilarities is of no interest. The use of non-metric MDS could be tested, with hope to achieve a better robustness.

— In chapter 4, sequential combinations (or chainings) of hierarchies are introduced. We show that when iterated, such chainings converge, and a link is made between this evolution and the local minima of the Dasgupta cost-function (see section 4.5.2). An interesting perspective consists in drawing more bridges between the different hierarchies introduced in this work and possible costs functions they maybe optimize. For now, the hierarchies we have introduced are defined procedurally, which makes the study of their effect complex, since the objective function they optimize is hard to figure out. Having such cost-functions to optimize would open the way to the application of more direct machine learning methods to learn the best hierarchy for given type of images and cost-function. However, this would require a fine-tuning of parameters, and would not necessarily present understandable effects. In a way, our approach is more controlled, as we can build combinations of hierarchies that capture distinctive features of images.

— In the designing of a learning-based workflow to do segmentation using hierarchies (as in chapter 5), the choice of the initial pool of hierarchies is adamant. Furthermore, whereas we only used hierarchical chainings in chapter 5, one could also use functional combinations of hierarchies, especially using the AND, OR and NOT operators between SWS hierarchies, which would provide more interpretable results. Instead of doing this choice by-hand, one could study the complementarity of different hierarchies for the considered set of images by using Gromov-Hausdorff distance and data analysis tools introduced in chapter 4. Finally, the exploration of the hierarchical space could be done in a cleverer way than with a greedy approach, which would allow to go deeper. For example, a reinforcement learning algorithm could be applied to select the combination of hierarchies that optimize a segmentation score.

— In chapter 6, distances between hierarchies were used to characterize images. We saw that the hierarchies that have been tested essentially conveyed geometrical information about images, and were thus not adapted to the study of images possessing periodicities properties. One could conceive to design hierarchies that capture such information to enrich the pool of hierarchies. For example, one could think of replacing a random marker by a couples of random markers and repeat the construction processes with various spacings between the markers.

— In chapter 6, distances between hierarchies were used as features for supervised classification tasks. One could also think of using them in an unsupervised fashion to do anomaly detection, which aims at identifying items, events or observations which do not conform to an expected pattern in a dataset. Given a set of images, for example images of items in an assembly line, one could compute these features for each image, and compare them using classical anomaly detection algorithms such as one-class SVM [Manevitz et al. (2001); Heller et al. (2003)] to detect outliers that present defaults.

— In chapter 6, we saw how to extract images features from a set of hierarchies. One could also use hierarchies to extract local features. For example, given a set of hierarchies applied

to the same image, one could constitute, for each contour, a vector of the different saliences of this contour in all hierarchies. Similar characterization vectors could be thought of for triple points of the image, or regions. This approach is similar to the classical SIFT approach, in the sense that such local descriptors are constructed by multiplying the points of view within a local domain around each contour/region/triple point in a controlled way. This interesting perspective has been explored in a an early phase of our work to follow the contours of an object of interest in a video. However, a major difficulty to face is the underrepresentation of the object of interest contours, compared with other contours: indeed, there are usually much more contours within a fine partition of an image than there are contours of interest within this fine partition. This makes this problem a very unbalanced one, a situation for which solutions exist and that would require further inquiry [He and Garcia (2009); He and Ma (2013)]. However, in a more controlled environment, and for very similar images as it is the case in stereovision, the matching of contours using these descriptors should be much more simple and provide interesting results. This has been presented in section 8.4.2 of chapter 8.

— In this thesis, we have worked with images modeled as graphs. This high-level of abstraction has proved useful to generate new algorithms and give us ideas about potential applications. However, it could potentially be applied to any type of data on which a neighborhood relation has been defined, as well as valuations between neighbors, generating a weighted graph. Methods introduced throughout the thesis could then be applied to others domains than images. A promising application would be the use of interhierarchies Gromov-Hausdorff distances applied on the same set of points to characterize it. This could be used as a non-supervised way to characterize and discriminate between different graphs.

— In chapter 7, we have seen how to introduce prior information during the construction of a hierarchical segmentation, which has proved to be useful for several applications. Furthermore, we observe the emergence of powerful semantic segmentations algorithms based on CNN in the literature [Long et al. (2015)], that provide impressive results for localization but can still be improved when it comes to precisely extract the contours of objects. Such prior-based hierarchies that would take as input the outcome of semantic segmentations algorithms, combined with a smart exploration of the hierarchy as in [Drouyer et al. (2017)], could help to refine their contours as other existing methods [Zheng et al. (2015)].

— We have presented the hierarchies of quasi-flat zones associated to levelings. Rather than choosing once for all a method to create a fine partition from which a hierarchy will be constructed, one may start with the image, construct a first hierarchy of quasi-flat zones and build upon it a second coarser hierarchy, for instance a SWS one. To go further, we could explore the complementarity of approaches that can be built on the same fine partition, encoded as a mosaic image. Levelings study the distribution and entanglement of peaks and valleys in the image, while watershed hierarchies provide a hierarchies of

contours in relation with the characteristics of their neighborhood.

— For analyzing stereo images, one may construct joint hierarchies based on the pair of images. One could construct SWS hierarchies where the random markers would be used in pairs, one for each image of the stereo pair.

In this thesis, we have proposed constructive and analytical approaches of a geometrical nature to study and characterize images. We believe that these approaches can complement other types of image analysis methods, particularly those based on convolutional neural networks. Their obtaining is made efficient and fast by dynamic programming on graphs and the important use of the minimum spanning tree structure.

I would be happy if this work aroused an interest in the community to resume and amplify it. I am thus available to exchange and work with those who will want to tackle it and go further in similar research directions.

Interface Example

Interface_example

February 27, 2018

This document presents an example of the framework implemented and used during this thesis to work with hierarchical segmentations.

As a reminder, here are the main steps to get a hierarchy of segmentations in the graphs-based hierarchical segmentation framework we use:

1. get a fine partition $FP(I)$ out of the image I
2. get a Region Adjacency Graph (RAG) $\mathcal{G}$ out of $FP(I)$, considering a given dissimilarity between regions of the fine partition
3. compute the Minimum Spanning Tree $\mathcal{MST}(\mathcal{G})$ from $\mathcal{G}$
4. find the inconsistent edges on the $\mathcal{MST}(\mathcal{G})$ according to some criteria and cut them, resulting in a partition of the image. Cutting edges in a MST produces forests. The more edges are cuts, the smallest the resulting forests. To a series of cuts corresponds a decreasing sequence of forests, and at each step, the trees of one forest underly a partition. Thus we obtain a series of nested partitions, i.e. a hierarchy. In particular, a simple way of proceeding to get such a hierarchy is to cut edges by decreasing valuations. When cutting edges by decreasing order of consistency, one get a hierarchy of segmentations.

In [1]:
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
#  Morphological hierarchies using dendrograms - Hands on
#
#  Amin Fehri - amin.fehri@mines-paristech.fr
import sys
sys.path.append('/home/fehri/Dev/Smil/LibDendro/')

import smilPython as sm
from MorphOperators.levelings import *
from Visualization.funcViz import *
from Segmentations.finePartition import *
from Segmentations.Waterpixels import *
from Dendrograms.graphFunct import *
from Dendrograms.dendroDist import *
from Dendrograms.salience import *

import cv2
from matplotlib import pyplot as plt
import matplotlib.cm as cm
from IPython.display import Image as ImagePy
import scipy
from scipy import misc

%matplotlib inline
```

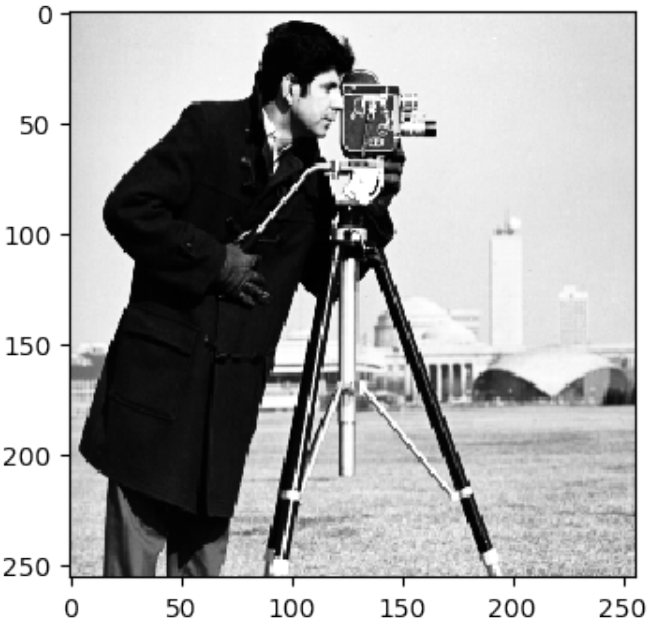
1 Getting a fine partition (labels image) of the image using waterpixels

We chose to use the waterpixels approach to compute the fine partition, as we found it to provide guarantees for not missing important contours while still being fast enough to keep the entire chain not prohibitively long for the user.

1.1 Open and print image

We open and print the classical cameraman image.

```
In [2]: imIn = sm.Image("cameraman.png")
        smilToNumpyPlot(imIn)
```



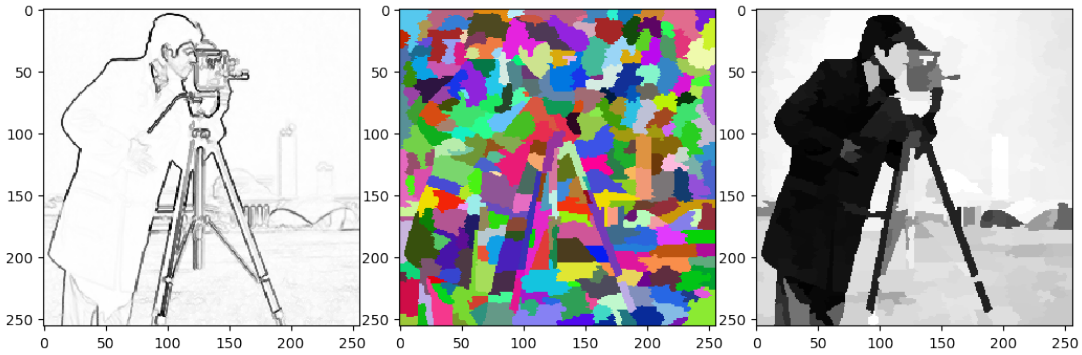
1.2 Generate a fine segmentation

We can now compute the fine segmentation using waterpixels.

```
In [3]: gradientLambs=[]
        paramsWP = [15,0.1,True,gradientLambs]
        imLabelsWP,  imGrad,  imMinimaVal = m_waterpixels(imIn,
                                                          step=paramsWP[0],
                                                          d_weight=paramsWP[1],
                                                          filter_ori=paramsWP[2],
                                                          lambs = paramsWP[3])
        imFP = sm.Image(imIn)
        segToMosaic(imIn,imLabelsWP,imFP)

        imGradInv = sm.Image(imGrad)
        sm.inv(imGrad,imGradInv)
```

```
        print "From left to right : gradient image, waterpixels, and corresponding mosaic image."
        smilToNumpyPlot3(imGradInv,imLabelsWP,imFP,[False,True,False])
        print "Nb of regions = " + str(sm.maxVal(imLabelsWP))
```

```
From left to right : gradient image, waterpixels, and corresponding mosaic image.
```




```
Nb of regions = 289
```

2 Get a Region Adjacency Graph (RAG) from the fine partition

```
In [4]: imLabels = sm.Image(imIn,"UINT32")
        imNodeValues = sm.Image(imIn,"UINT32")
        imEdgeValues = sm.Image(imIn,"UINT32")

        sm.copy(imLabelsWP,imLabels)
        sm.sub(imLabels,1,imLabels) # For labels to begin at 0
        sm.labelWithArea(imLabels,imNodeValues)

        imGrad = sm.Image(imIn,"UINT8")
        if imIn.getTypeAsString() == "RGB":
            sm.gradient_LAB(imIn,imGrad)
        else:
            sm.gradient(imIn,imGrad)
        sm.copy(imGrad,imEdgeValues)

        mosaicToGraphMedian = medianMosaicToGraphFunct()
        g=mosaicToGraphMedian(imLabels,imEdgeValues,imNodeValues)
```

3 Get a Minimum Spanning Tree (MST) from the RAG

```
In [5]: mst = g.computeMST()
```

4 Get an dendrogram structure from this MST

```
In [6]: imIn32 = sm.Image(imIn,"UINT32")
        sm.copy(imIn,imIn32)
        dendroFromG = sm.DendrogramEnergy_UINT32_UINT32_UINT32(mst,imLabels,imIn32)
```

5 A simple way to obtain and compare different hierarchical clusterings

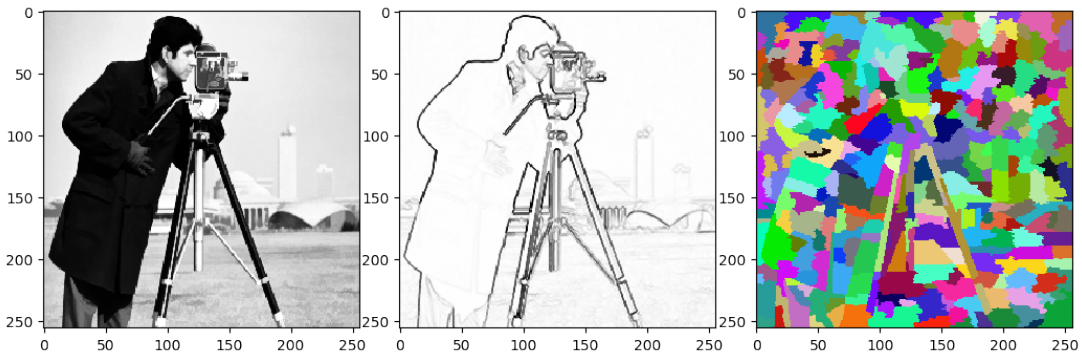
5.1 Compute hierarchical clusterings

In [7]:
```
nMarkers = 50

dendroTmp1 = sm.DendrogramEnergy_UINT32_UINT32_UINT32(dendroFromG)
dendroTmp2 = sm.DendrogramEnergy_UINT32_UINT32_UINT32(dendroFromG)

dendroTmp1.HierarchicalConstruction("stochasticSurfacic",nMarkers)
dendroTmp2.HierarchicalConstruction("stochasticVolumic",nMarkers)
```

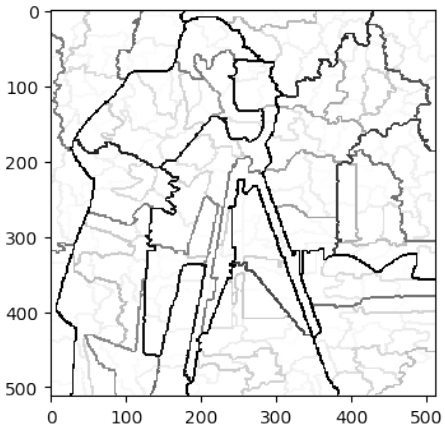
5.2 Print saliency images for these different hierarchies

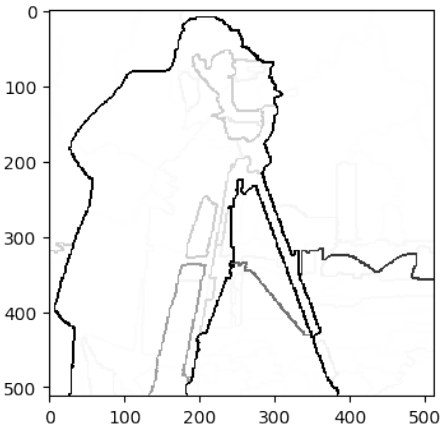
In [8]:
```
imS,imMask = getSalienceImage(imLabels,dendroTmp1,g)
imVisu = (imS*imMask).T
scipy.misc.imsave("imS1.png",imVisu)
imSaliences1 = sm.Image("imS1.png")
sm.inv(imSaliences1,imSaliences1)
sm.erode(imSaliences1,imSaliences1,sm.HexSE())

imS,imMask = getSalienceImage(imLabels,dendroTmp2,g)
imVisu = (imS*imMask).T
scipy.misc.imsave("imS2.png",imVisu)
imSaliences2 = sm.Image("imS2.png")
sm.inv(imSaliences2,imSaliences2)
sm.erode(imSaliences2,imSaliences2,sm.HexSE())

print "From left to right : image, gradient image, fine partition of the image."
smilToNumpyPlot3(imIn,imGradInv,imLabels,[False,False,True])

print "Saliency images of hierarchical clusterings : surface-based and" + \
        " volume-based stochastic watersheds."
smilToNumpyPlot2(imSaliences1,imSaliences2)
```

```
From left to right : image, gradient image, fine partition of the image.
```




```
Saliency images of hierarchical clusterings : surface-based and volume-based stochastic watersheds.
```


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500

References

Achanta, R. et al. (2012). « SLIC super pixels compared to state-of-the-art super pixel methods ». In:

Agarwal, P. K. et al. (2015). « Computing the Gromov-Hausdorff distance for metric trees ». In: *International Symposium on Algorithms and Computation*. Springer, pp. 529–540.

Agarwala, A. et al. (2004). « Keyframe-based tracking for rotoscoping and animation ». In: *SIGGRAPH*.

Angulo, J. and D. Jeulin (2007). « Stochastic watershed segmentation ». In: *ISMM* 1, pp. 265–276.

Angulo, J., S. Velasco-Forero, and J. Chanussot (2009). « Multiscale stochastic watershed for unsupervised hyperspectral image segmentation ». In: *Geoscience and Remote Sensing Symposium, 2009 IEEE International, IGARSS 2009*. Vol. 3. IEEE, pp. III–93.

Arbelaez, P. et al. (2011). « Contour detection and hierarchical image segmentation ». In: 33.5, pp. 898–916.

Arbelaez, P. (2006). « Boundary extraction in natural images using ultrametric contour maps ». In: *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*. IEEE, pp. 182–182.

Asano, T. et al. (1988). « Clustering algorithms based on minimum and maximum spanning trees ». In: *Proceedings of the fourth annual symposium on Computational geometry*. ACM, pp. 252–257.

Bai, X. et al. (2009). « Video SnapCut: robust video object cutout using localized classifiers ». In: *ACM Transactions on Graphics* 28.1.

Ballester, C., V. Caselles, and P. Monasse (2003). « The tree of shapes of an image ». In: *ESAIM: Control, Optimisation and Calculus of Variations* 9, pp. 1–18.

Bao, L. et al. (2014). « Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree ». In: *IEEE Transactions on Image Processing* 23.2, pp. 555–569.

Barron, J. T. and B. Poole (2016). « The fast bilateral solver ». In: *European Conference on Computer Vision*. Springer, pp. 617–632.

Basu, S., I. Davidson, and K. Wagstaff (2008). *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press.

Bell, S., K. Bala, and N. Snavely (2014). « Intrinsic Images in the Wild ». In: *ACM Trans. on Graphics (SIGGRAPH)* 33.4.

Bellman, R. (1957). *Dynamic Programming*. 1st ed. Princeton, NJ, USA: Princeton University Press.

Benzécri, J.-P. et al. (1973). *L'analyse des données*. Vol. 2. Dunod Paris.

Berge, C. (1985). *Graphs*. Vol. 6. North-Holland.

Berkhin, P. (2006). « A survey of clustering data mining techniques ». In: *Grouping multidimensional data*. Springer, pp. 25–71.

Bernander, K. B. et al. (2013). « Improving the stochastic watershed ». In: *Pattern Recognition Letters* 34.9, pp. 993–1000.

Beucher, S. (1990). « Segmentation d'images et morphologie mathématique ». PhD thesis. Ecole Nationale Supérieure des Mines de Paris.

Beucher, S. and F. Meyer (1992). « The morphological approach to segmentation: the watershed transformation ». In: *Optical Engineering-New York-Marcel Dekker Incorporated-* 34, pp. 433–433.

Boruvka, O. (1926). « O jistém problému minimálním [About a certain minimal problem] ». In: *Práce mor. přírodověd. spol. v Brně III (in Czech and German)* 3, pp. 37–58.

Boser, B. E., I. M. Guyon, and V. N. Vapnik (1992). « A training algorithm for optimal margin classifiers ». In: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, pp. 144–152.

Boykov, Y. and V. Kolmogorov (2004). « An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision ». In: *IEEE transactions on pattern analysis and machine intelligence* 26.9, pp. 1124–1137.

Breiman, L. (2001). « Random forests ». In: *Machine learning* 45.1, pp. 5–32.

Bricola, J.-C., M. Bilodeau, and S. Beucher (July 2015). « A multi-scale and morphological gradient preserving contrast ». In: *14th International Congress for Stereology and Image Analysis*. Liège, Belgium.

Caelles, S. et al. (2017). « One-Shot Video Object Segmentation ». In: *Computer Vision and Pattern Recognition (CVPR)*.

Cao, F., P. Musé, and F. Sur (2005). « Extracting meaningful curves from images ». In: *Journal of Mathematical Imaging and Vision* 22.2, pp. 159–181.

Carlsson, G. and F. Mémoli (2010). « Characterization, stability and convergence of hierarchical clustering methods ». In: *Journal of machine learning research* 11.Apr, pp. 1425–1470.

Cayley, F. (1874). « LVII. On the mathematical theory of isomers ». In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 47.314, pp. 444–447.

Chan, T. and W. Zhu (2005). « Level set based shape prior segmentation ». In: vol. 2, pp. 1164–1170.

Chang, C.-I. (2003). *Hyperspectral imaging: techniques for spectral detection and classification*. Vol. 1. Springer Science & Business Media.

Chang, J., D. Wei, and J. W. F. III (2013). « A Video Representation Using Temporal Superpixels ». In: *CVPR*.

Chen, F., H. Yu, et al. (2013). « Deep learning shape priors for object segmentation ». In: pp. 1870–1877.

Chen, J., S. Paris, and F. Durand (Aug. 24, 2007). « Real-time edge-aware image processing with the bilateral grid. » In: *ACM Trans. Graph.* 26.3, p. 103.

Chen, Y., D. Dai, et al. (2016). « Scale-Aware Alignment of Hierarchical Image Segmentation ». In: pp. 0–0.

Chuang, Y. et al. (2002). « Video matting of complex scenes ». In: *SIGGRAPH*.

Cimpoi, M. et al. (2014). « Describing textures in the wild ». In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, pp. 3606–3613.

Comaniciu, D. and P. Meer (2002). « Mean shift: A robust approach toward feature space analysis ». In: 24.5, pp. 603–619.

Cortes, C. and V. Vapnik (1995). « Support-vector networks ». In: *Machine learning* 20.3, pp. 273–297.

Couprie, C. et al. (2011). « Power watershed: A unifying graph-based optimization framework ». In: 33.7, pp. 1384–1399.

Cousty, J. et al. (Oct. 2017). « Hierarchical segmentations with graphs: quasi-flat zones, minimum spanning trees, and saliency maps ». In: *Journal of Mathematical Imaging and Vision*.

Cox, M. A. and T. F. Cox (2008). « Multidimensional scaling ». In: *Handbook of data visualization*. Springer, pp. 315–347.

Crespo, J. et al. (1997). « The flat zone approach: a general low-level region merging segmentation method ». In: *Signal Processing* 62.1, pp. 37–60.

Csurka, G. et al. (2004). « Visual categorization with bags of keypoints ». In: *Workshop on Statistical Learning in Computer Vision, ECCV*, pp. 1–22.

Dalal, N. and B. Triggs (2005). « Histograms of oriented gradients for human detection ». In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, pp. 886–893.

Dasgupta, S. (2016). « A cost function for similarity-based hierarchical clustering ». In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. ACM, pp. 118–127.

Davidson, I. and S. Ravi (2005). « Agglomerative hierarchical clustering with constraints: Theoretical and empirical results ». In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, pp. 59–70.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). « Maximum likelihood from incomplete data via the EM algorithm ». In: *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38.

Deng, J. et al. (2009). « ImageNet: A Large-Scale Hierarchical Image Database ». In:

Dijkstra, E. W. (1959). « A note on two problems in connexion with graphs ». In: *Numerische mathematik* 1.1, pp. 269–271.

Drouyer, S. et al. (2017). « Sparse stereo disparity map densification using hierarchical image segmentation ». In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, pp. 172–184.

Euler, L. (1736). « Solutio problematis ad geometriam situs pertinentis ». In: *Commentarii Academiae Scientiarum Imperialis Petropolitanae* 8, pp. 128–140.

Fan, Q. et al. (2015). « JumpCut: Non-Successive Mask Transfer and Interpolation for Video Cutout ». In: *SIGGRAPH*.

Fankhauser, P. et al. (2015). « Kinect v2 for mobile robot navigation: Evaluation and modeling ». In: *Advanced Robotics (ICAR), 2015 International Conference on*. IEEE, pp. 388–394.

Felsenstein, J. (2014). *Inferring phylogenies*. Vol. 2. Sinauer associates Sunderland, MA.

Felzenszwalb, P. F. and D. P. Huttenlocher (2004). « Efficient graph-based image segmentation ». In: 59.2, pp. 167–181.

Fischler, M. A. and R. C. Bolles (1987). « Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography ». In: *Readings in computer vision*. Elsevier, pp. 726–740.

Forgy, E. W. (1965). « Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications ». In: *biometrics* 21, pp. 768–769.

Franchi, G. and J. Angulo (2015). « Bagging stochastic watershed on natural color image segmentation ». In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, pp. 422–433.

Fukushima, K. (1980). « Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position ». In: *Biological cybernetics*.

Gomila, C. (2001). « Mise en correspondance de partitions en vue du suivi d'objets ». PhD thesis. École Nationale Supérieure des Mines de Paris.

Gomila, C. and F. Meyer (2003). « Graph-based object tracking ». In: *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*. Vol. 2. IEEE, pp. II–41.

Gondran, M. (1975). « Path algebra and algorithms ». In: *Combinatorial programming: methods and applications*. Springer, pp. 137–148.

Gondran, M., M. Minoux, and S. Vajda (1984). *Graphs and Algorithms*. New York, NY, USA: John Wiley & Sons, Inc.

Goodfellow, I. J., J. Shlens, and C. Szegedy (2014). « Explaining and harnessing adversarial examples ». In: *arXiv preprint arXiv:1412.6572*.

Grimaud, M. (1992). « New measure of contrast: the dynamics ». In: *San Diego'92*. International Society for Optics and Photonics, pp. 292–305.

Gromov, M., J. Lafontaine, and P. Pansu (1981). *Structures métriques pour les variétés riemanniennes*. Cedic.

Gromov, M. (2007). *Metric structures for Riemannian and non-Riemannian spaces*. Springer Science & Business Media.

Grundmann, M. et al. (2010). « Efficient hierarchical graph-based video segmentation ». In: *CVPR*.

Gueguen, L., S. Velasco-Forero, and P. Soille (2013). « Local Mutual Information for Dissimilarity-Based Image Segmentation ». In: pp. 1–20.

Guigues, L., J. P. Cocquerez, and H. Le Men (2006). « Scale-sets image analysis ». In: 68.3, pp. 289–317.

Hagen, L. and A. B. Kahng (1992). « New spectral methods for ratio cut partitioning and clustering ». In: *IEEE transactions on computer-aided design of integrated circuits and systems* 11.9, pp. 1074–1085.

He, H. and E. A. Garcia (2009). « Learning from imbalanced data ». In: *IEEE Transactions on knowledge and data engineering* 21.9, pp. 1263–1284.

He, H. and Y. Ma (2013). *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons.

He, K., X. Zhang, et al. (2016). « Deep residual learning for image recognition ». In: *CVPR*.

Heller, K. A. et al. (2003). « One class support vector machines for detecting anomalous windows registry accesses ». In: *Proceedings of the workshop on Data Mining for Computer Security*. Vol. 9.

Hinton, G. E. and S. T. Roweis (2003). « Stochastic neighbor embedding ». In: *Advances in neural information processing systems*, pp. 857–864.

Hotelling, H. (1933). « Analysis of a complex of statistical variables into principal components. » In: *Journal of educational psychology* 24.6, p. 417.

Hu, T. (1961). « The Maximum Capacity Route Problem ». In: *Operations Research*, pp. 898–900.

Hubel, D. and T. Wiesel (1968). « Receptive fields and functional architecture of monkey striate cortex ». In: *The Journal of physiology*.

Inc., P. T. (2015). *Collaborative data science*. URL: https://plot.ly.

Jain, A. K., M. N. Murty, and P. J. Flynn (1999). « Data clustering: a review ». In: *ACM computing surveys (CSUR)* 31.3, pp. 264–323.

Jampani, V., R. Gadde, and P. V. Gehler (2017). « Video Propagation Networks ». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 451–461.

Jarnik, V. (1930). « O jistém problému minimálním.(Z dopisu panu O. Boruvkovi) ». In:

Jeulin, D. (1997). « Dead leaves models: From space tessellation to random functions ». In: *Advances in Theory and Applications of Random Sets*. World Scientific Publishing, pp. 137–156.

Jeulin, D. (2016). « Morphological probabilistic hierarchies for texture segmentation ». In: *Mathematical Morphology-Theory and Applications* 1.1.

Kaufman, L. and P. J. Rousseeuw (2009). *Finding groups in data: an introduction to cluster analysis*. Vol. 344. John Wiley & Sons.

Kiran, B. R. and J. Serra (2014a). « Global–local optimizations by hierarchical cuts and climbing energies ». In: *Pattern Recognition* 47.1, pp. 12–24.

Kiran, B. R. and J. Serra (2014b). « Global-local optimizations by hierarchical cuts and climbing energies ». In: *Pattern Recognition* 47.1, pp. 12–24.

Kirchoff, G. (1847). « Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung det linearen Verteilung galvanischer Ströme gefurht wird ». In: *Annalen der Physik u. Chemie (Poggendorff's Annalen)* 72, pp. 497–508.

Kleinberg, J. M. (2003). « An impossibility theorem for clustering ». In: *Advances in neural information processing systems*, pp. 463–470.

Krizhevsky, A., I. Sutskever, and G. Hinton (2012). « Imagenet classification with deep convolutional neural networks ». In: *NIPS*.

Kruskal, J. B. (1956). « On the shortest spanning subtree of a graph and the traveling salesman problem ». In: *Proceedings of the American Mathematical society* 7.1, pp. 48–50.

Lampert, C., M. Blaschko, and T. Hofmann (2008). « Beyond Sliding Windows: Object Localization by Efficient Subwindow Search ». In: pp. 1–8.

Lang, M. et al. (2012). « Practical temporal consistency for image-based graphics applications ». In: *ACM Transactions on Graphics* 31.4.

Lantuéjoul, C. and S. Beucher (1981). « On the use of the geodesic metric in image analysis ». In: *Journal of Microscopy* 121.1, pp. 39–49.

LeCun, B., B. Denker, et al. (1990). « Handwritten digit recognition with a backpropagation network ». In: *NIPS*.

LeCun, Y., L. Bottou, et al. (1998). « Gradient-based learning applied to document recognition ». In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Lee, J. A. and M. Verleysen (2007). *Nonlinear dimensionality reduction*. Springer Science & Business Media.

Lerallut, R. (2006). « Modelization and Interpretation of Images using Graphs ». PhD thesis. École Nationale Supérieure des Mines de Paris.

Levinshtein, A. et al. (2009). « Turbopixels: Fast superpixels using geometric flows ». In: *IEEE transactions on pattern analysis and machine intelligence* 31.12, pp. 2290–2297.

Li, Y. et al. (2005). « Video object cut and paste ». In: *ACM Transactions on Graphics* 24.3, pp. 595–600.

Lin, T.-Y. et al. (2014). « Microsoft coco: Common objects in context ». In: *European conference on computer vision*. Springer, pp. 740–755.

Linné, C. v. and L. Salvius (1758). *Systema naturae per regna tria naturae :secundum classes, ordines, genera, species, cum characteribus, differentiis, synonymis, locis.* Holmiae :Impensis Direct. Laurentii Salvii., p. 881.

Liu, Z. et al. (2014). « Co-saliency detection based on hierarchical segmentation ». In: *IEEE Signal Processing Letters* 21.1, pp. 88–92.

Long, J., E. Shelhamer, and T. Darrell (2015). « Fully convolutional networks for semantic segmentation ». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440.

Lowe, D. G. (2004). « Distinctive Image Features from Scale-Invariant Keypoints ». In: *IJCV* 60.2, pp. 91–110.

Maaten, L. v. d. and G. Hinton (2008). « Visualizing data using t-SNE ». In: *Journal of machine learning research* 9.Nov, pp. 2579–2605.

Machairas, V. et al. (2015). « Waterpixels ». In: 24.11, pp. 3707–3716.

Mairal, J., F. Bach, and J. Ponce (2012). « Task-driven dictionary learning ». In: *IEEE transactions on pattern analysis and machine intelligence* 34.4, pp. 791–804.

Malik, J. et al. (2001). « Contour and texture analysis for image segmentation ». In: *International journal of computer vision* 43.1, pp. 7–27.

Malmberg, F., C. L. L. Hendriks, and R. Strand (2017). « Exact evaluation of targeted stochastic watershed cuts ». In: *Discrete Applied Mathematics* 216, pp. 449–460.

Manevitz, L. M. and M. Yousef (2001). « One-class SVMs for document classification ». In: *Journal of machine Learning research* 2.Dec, pp. 139–154.

Marcotegui, B. et al. (1999). « A video object generation tool allowing friendly user interaction ». In: *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*. Vol. 2. IEEE, pp. 391–395.

Marcus, G. (2018). « Deep Learning: A Critical Appraisal ». In: *arXiv preprint arXiv:1801.00631*.

Marki, N. et al. (2016). « Bilateral Space Video Segmentation ». In: *CVPR*.

Matheron, G. (1997). « Les nivellements ». In: *Tech. Rep. Ecole des Mines de Paris*. Vol. No N-07/97/MM.

Matheron, G. (1975). *Random sets and integral geometry*. Wiley New York.

McCulloch, W. and W. Pitts (1943). « A logical calculus of the ideas immanent in nervous activity ». In: *Bulletin of Mathematical Biophysics*.

McLachlan, G. and T. Krishnan (2007). *The EM algorithm and extensions*. Vol. 382. John Wiley & Sons.

Mémoli F.and Sapiro, G. (2004). « Comparing point clouds ». In: *ACM SIGGRAPH symposium on Geometry processing*. ACM, pp. 32–40.

Meyer, F. and S. Beucher (1990). « Morphological segmentation ». In: *Journal of visual communication and image representation* 1.1, pp. 21–46.

Meyer, F. (1994a). « Minimum spanning forests for morphological segmentation ». In: *Mathematical morphology and its applications to image processing*. Springer, pp. 77–84.

— (1994b). « Topographic distance and watershed lines ». In: *Signal processing* 38.1, pp. 113–125.

— (1998). « The levelings ». In: pp. 199–206.

— (2005). « Grey-weighted, ultrametric and lexicographic distances ». In: *Mathematical Morphology: 40 Years On*. Springer, pp. 289–298.

— (2013a). « Adjunctions on the lattice of dendrograms ». In: *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer, pp. 91–100.

Meyer, F. (2013b). « Flooding edge or node weighted graphs ». In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, pp. 341–352.

— (2013c). « Flooding Edge Weighted Graphs ». In: *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer, pp. 31–40.

— (2014). « Watersheds on weighted graphs ». In: *Pattern Recognition Letters* 47, pp. 72–79.

— (2015). « The waterfall hierarchy on weighted graphs ». In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, pp. 325–336.

Meyer, F. and R. Lerallut (2007). « Morphological operators for flooding, leveling and filtering images using graphs ». In: *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer, pp. 158–167.

Meyer, F. and J. Stawiaski (2010). « A stochastic evaluation of the contour strength ». In: *Joint Pattern Recognition Symposium*. Springer, pp. 513–522.

Mikolajczyk, K. and C. Schmid (2005). « A performance evaluation of local descriptors ». In: *IEEE transactions on pattern analysis and machine intelligence* 27.10, pp. 1615–1630.

Mnih, V. et al. (2015). « Human-level control through deep reinforcement learning ». In: *Nature* 518.7540, p. 529.

Monasse, P. and F. Guichard (2000). « Fast computation of a contrast-invariant image representation ». In: *IEEE Transactions on Image Processing* 9.5, pp. 860–872.

Mumford, D. and J. Shah (1989). « Optimal approximations by piecewise smooth functions and associated variational problems ». In: *Communications on pure and applied mathematics* 42.5, pp. 577–685.

Najman, L. and M. Schmitt (1996). « Geodesic saliency of watershed contours and hierarchical segmentation ». In: *IEEE Transactions on pattern analysis and machine intelligence* 18.12, pp. 1163–1173.

Naumov, M. and T. Moon (2016). *Parallel Spectral Graph Partitioning*. Tech. rep. NVIDIA Technical Report, NVR-2016-001.

Nešetril, J. and H. Nešetrilová (2010). « The Origins of Minimal Spanning Tree Algorithms–Boruvka and Jarnık ». In:

Ng, A. Y., M. I. Jordan, and Y. Weiss (2002). « On spectral clustering: Analysis and an algorithm ». In: *Advances in neural information processing systems*, pp. 849–856.

Nguyen, A., J. Yosinski, and J. Clune (2015). « Deep neural networks are easily fooled: High confidence predictions for unrecognizable images ». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436.

Niepert, M., M. Ahmed, and K. Kutzkov (2016). « Learning convolutional neural networks for graphs ». In: *International conference on machine learning*, pp. 2014–2023.

Oquab, M. et al. (2015). « Is object localization for free? Weakly-supervised learning with convolutional neural networks ». In:

Ouzounis, G. K., M. Pesaresi, and P. Soille (2012). « Differential area profiles: decomposition properties and efficient computation ». In: 34.8, pp. 1533–1548.

Pardo, A. (2002). « Semantic image segmentation using morphological tools ». In: *Image Processing. 2002. Proceedings. 2002 International Conference on*. Vol. 2. IEEE, pp. II–II.

Perazzi, F., J. Pont-Tuset, et al. (2016). « A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation ». In: *Computer Vision and Pattern Recognition*.

Perazzi, F., O. Wang, et al. (2015). « Fully Connected Object Proposals for Video Segmentation ». In: *ICCV*.

Perazzi, F., A. Khoreva, et al. (2017). « Learning Video Object Segmentation From Static Images ». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2663–2672.

Pérez, F. and B. E. Granger (May 2007). « IPython: a System for Interactive Scientific Computing ». In: *Computing in Science and Engineering* 9.3, pp. 21–29. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.53. URL: http://ipython.org.

Perret, B. et al. (2018). « Evaluation of hierarchical watersheds ». In: *IEEE Transactions on Image Processing* 27.4, pp. 1676–1688.

Pitkow, X. (2010). « Exact feature probabilities in images with occlusion ». In: *Journal of vision* 10.14, pp. 42–42.

Pock, T. et al. (2009). « An algorithm for minimizing the Mumford-Shah functional ». In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, pp. 1133–1140.

Price, B. L., B. S. Morse, and S. Cohen (2009). « LIVEcut: Learning based interactive video segmentation by evaluation of multiple propagated cues ». In: *ICCV*.

Prim, R. C. (1957). « Shortest connection networks and some generalizations ». In: *Bell Labs Technical Journal* 36.6, pp. 1389–1401.

Ramakanth, S. A. and R. V. Babu (2014). « SeamSeg: Video Object Segmentation Using Patch Seams ». In: *CVPR*.

Redmon, J. et al. (2016). « You only look once: Unified, real-time object detection ». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Ren, Z. and G. Shakhnarovich (2013). « Image segmentation by cascaded region agglomeration ». In: pp. 2011–2018.

Rivest, J.-F., P. Soille, and S. Beucher (1992). « Morphological gradients ». In: *SPIE/IS&T 1992 Symposium on Electronic Imaging: Science and Technology*. International Society for Optics and Photonics, pp. 139–150.

Ronse, C. (2010). « Adjunctions on the lattices of partitions and of partial partitions ». In: *Applicable Algebra in Engineering, Communication and Computing* 21.5, pp. 343–396.

Rosten, E. and T. Drummond (2006). « Machine learning for high-speed corner detection ». In: *European conference on computer vision*. Springer, pp. 430–443.

Saglam, A. and N. A. Baykan (2017). « Sequential image segmentation based on minimum spanning tree representation ». In: *Pattern Recognition Letters* 87, pp. 155–162.

Salembier, P. and L. Garrido (2000). « Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval ». In: 9.4, pp. 561–576.

Salembier, P., A. Oliveras, and L. Garrido (1998). « Antiextensive connected operators for image and sequence processing ». In: *IEEE Transactions on Image Processing* 7.4, pp. 555–570.

Santana Maia, D. et al. (2017). « Evaluation of Combinations of Watershed Hierarchies ». In: *Mathematical Morphology and Its Applications to Signal and Image Processing: 13th International Symposium, ISMM 2017, Fontainebleau, France, May 15–17, 2017, Proceedings*. Springer International Publishing, pp. 133–145.

Santoro, A. et al. (2017). « A simple neural network module for relational reasoning ». In: *Advances in neural information processing systems*, pp. 4974–4983.

Scharstein, D. et al. (2014). « High-resolution stereo datasets with subpixel-accurate ground truth ». In: *German Conference on Pattern Recognition*. Springer, pp. 31–42.

Sermanet, P. et al. (2013). « OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks ». In: *ICLR*.

Serra, J., C. Vachier, and F. Meyer (n.d.). « Levelings ». In: *Mathematical Morphology: From Theory to Applications* (), pp. 199–228.

Shi, J. and J. Malik (2000). « Normalized cuts and image segmentation ». In: *IEEE Transactions on pattern analysis and machine intelligence* 22.8, pp. 888–905.

Sifre, L. and S. Mallat (2013). « Rotation, scaling and deformation invariant scattering for texture discrimination ». In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, pp. 1233–1240.

Simonyan, K. and A. Zisserman (2014). « Very Deep Convolutional Networks for Large-Scale Image Recognition ». In: *arXiv* 1409.1556.

Simonyan, K. and K. Zisserman (2015). « Very deep convolutional networks for large-scale image recognition ». In: *ICLR*.

Sinop, A. K. and L. Grady (2007). « A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm ». In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, pp. 1–8.

Smil. (2015). *Simple Morphological Image Library*. URL: http://smil.cmm.mines-paristech.fr.

Soille, P. (2011). « Preventing chaining through transitions while favouring it within homogeneous regions ». In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, pp. 96–107.

Stawiaski, J. (2011). « Optimal path: Theory and models for vessel segmentation ». In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, pp. 417–428.

Stawiaski, J. and F. Meyer (2009). « Minimum spanning tree adaptive image filtering ». In: *Image Processing (ICIP), 2009 16th IEEE International Conference on*. IEEE, pp. 2245–2248.

Su, B., S. Lu, and C. L. Tan (2011). « Blurred Image Region Detection and Classification ». In: *ACM International Conference on Multimedia*. MM '11, pp. 1397–1400.

Sylvester, J. (1878). « Chemistry and Algebra ». In: *Nature* 17, p. 284.

Szegedy, C. et al. (2015). « Going deeper with convolutions ». In: *CVPR*.

Tomasi, C. and R. Manduchi (1998). « Bilateral filtering for gray and color images ». In: *Computer Vision, 1998. Sixth International Conference on*. IEEE, pp. 839–846.

Torgerson, W. S. (1952). « Multidimensional scaling: I. Theory and method ». In: *Psychometrika* 17.4, pp. 401–419.

Vachier, C. and F. Meyer (1995). « Extinction value: a new measurement of persistence ». In: *Proceedings of the IEEE Workshop on Non Linear Signal/Image Processing*, pp. 254–257.

Veksler, O. (2008). « Star shape prior for graph-cut image segmentation ». In: *ECCV*. Springer, pp. 454–467.

Vincent, L. (1993). « Morphological grayscale reconstruction in image analysis: applications and efficient algorithms ». In: *IEEE transactions on image processing* 2.2, pp. 176–201.

Viola, P. and M. Jones (2001). « Rapid object detection using a boosted cascade of simple features ». In: pp. 511–518.

Von Luxburg, U. (2007). « A tutorial on spectral clustering ». In: *Statistics and computing* 17.4, pp. 395–416.

Wang, G.-W., C.-X. Zhang, and J. Zhuang (2014). « Clustering with Prim's sequential representation of minimum spanning tree ». In: *Applied Mathematics and Computation* 247, pp. 521–534.

Wang, J., Y. Jia, et al. (2008). « Normalized tree partitioning for image segmentation ». In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, pp. 1–8.

Xu, C. and J. Corso (2012). « Evaluation of Super-Voxel Methods for Early Video Processing ». In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

Xu, Y., E. Carlinet, et al. (2016). « Hierarchical Segmentation Using Tree-Based Shape Space ». In: PP.99, pp. 1–1. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2554550.

Xu, Y., V. Olman, and D. Xu (2001). « Minimum spanning trees for gene expression data clustering ». In: *Genome Informatics* 12, pp. 24–33.

Xu, Y. and E. C. Uberbacher (1997). « 2D image segmentation using minimum spanning trees ». In: *Image and Vision Computing* 15.1, pp. 47–57.

Xu, Y., T. Géraud, and L. Najman (2016). « Hierarchical image simplification and segmentation based on Mumford-Shah-salient level line selection ». In: *Pattern Recognition Letters*.

Yan, Z. and X. S. Zhou (2017). « How intelligent are convolutional neural networks? » In: *arXiv preprint arXiv:1709.06126*.

Yosinski, J. et al. (2014). « How transferable are features in deep neural networks? » In: *Advances in neural information processing systems*, pp. 3320–3328.

Yu, Y., C. Fang, and Z. Liao (2015). « Piecewise Flat Embedding for Image Segmentation ». In: pp. 1368–1376.

Zadeh, R. B. and S. Ben-David (2009). « A uniqueness theorem for clustering ». In: *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, pp. 639–646.

Zahn, C. T. (1971). « Graph-theoretical methods for detecting and describing gestalt clusters ». In: *IEEE Transactions on computers* 100.1, pp. 68–86.

Zanoguera, F., B. Marcotegui, and F. Meyer (1999). « A toolbox for interactive segmentation based on nested partitions ». In: *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*. Vol. 1. IEEE, pp. 21–25.

Zanoguera, M. F. (2001). « Segmentation interactive d'images fixes et de séquences vidéo basée sur des hiérarchies de partitions ». PhD thesis. École Nationale Supérieure des Mines de Paris.

Zheng, S. et al. (2015). « Conditional random fields as recurrent neural networks ». In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1529–1537.

Zhou, H., Y. Yuan, and C. Shi (2009). « Object tracking using SIFT features and mean shift ». In: *Computer vision and image understanding* 113.3, pp. 345–352.

Zhu, J. et al. (2004). « 1-norm support vector machines ». In: *Advances in NIPS*, pp. 49–56.

Zoran, D. and Y. Weiss (2012). « Natural images, Gaussian mixtures and dead leaves ». In: *Advances in NIPS*, pp. 1736–1744.

Personal publications

Fehri, A., S. Velasco-Forero, and F. Meyer (2016). « Automatic Selection of Stochastic Watershed Hierarchies ». In: *24th European Signal Processing Conference*. IEEE, pp. 1877–1881.

— (2017a). « Prior-based Hierarchical Segmentation Highlighting Structures of Interest ». In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, pp. 146–158.

— (2017b). « Segmentation hiérarchique faiblement supervisée ». In: *Actes du 26e Colloque GRETSI, Juan-Les-Pins, France*.

— (2018). « Characterizing Images by the Gromov-Hausdorff Distances Between Derived Hierarchies ». In: *2018 IEEE International Conference on Image Processing (ICIP)*.

Résumé

Cette thèse porte sur l'extraction de descripteurs hiérarchiques et multi-échelles d'images, en vue de leur interprétation, caractérisation et segmentation. Elle se décompose en deux parties.

La première partie expose des éléments théoriques et méthodologiques sur l'obtention de classifications hiérarchiques des nœuds d'un graphe valué aux arêtes. Ces méthodes sont ensuite appliquées à des graphes représentant des images pour obtenir différentes méthodes de segmentation hiérarchique d'images. De plus, nous introduisons différentes façons de combiner des segmentations hiérarchiques. Nous proposons enfin une méthodologie pour structurer et étudier l'espace des hiérarchies que nous avons construites en utilisant la distance de Gromov-Hausdorff entre elles.

La seconde partie explore plusieurs applications de ces descriptions hiérarchiques d'images. Nous exposons une méthode pour apprendre à extraire de ces hiérarchies une bonne segmentation de façon automatique, étant donnés un type d'images et un score de bonne segmentation. Nous proposons également des descripteurs d'images obtenus par mesure des distances inter-hiérarchies, et exposons leur efficacité sur des données réelles et simulées. Enfin, nous étendons les potentielles applications de ces hiérarchies en introduisant une technique permettant de prendre en compte toute information spatiale a priori durant leur construction.

Mots Clés

Traitement d'images, Morphologie mathématique, Segmentation, Segmentation hiérarchique, Classification hiérarchique, Apprentissage statistique, Théorie des graphes.

Abstract

This thesis deals with the extraction of hierarchical and multiscale descriptors on images, in order to interpret, characterize and segment them. It breaks down into two parts.

The first part outlines a theoretical and methodological approach for obtaining hierarchical clusterings of the nodes of an edge-weighted graph. These methods are then applied to graphs representing images and derive different hierarchical segmentation techniques. In addition, we introduce different approaches to combine hierarchical segmentations. Finally, we propose a methodology for structuring and studying the space of hierarchies by using the Gromov-Hausdorff distance as a metric.

The second part explores several applications of these hierarchical descriptions for images. We expose a method to learn how to automatically extract a segmentation of an image, given a type of images and a score of evaluation for a segmentation. We also propose image descriptors obtained by measuring inter-hierarchical distances, and expose their efficiency on real and simulated data. Finally, we extend the potential applications of these hierarchies by introducing a technique to take into account any spatial prior information during their construction.

Keywords

Image Processing, Mathematical Morphology, Segmentation, Hierarchical Segmentation, Hierarchical Clustering, Machine Learning, Graph Theory.