Indexation et recherche d'images par arbres des coupes

Petra Bosilj

THESE / UNIVERSITE DE BRETAGNE LOIRE
sous le sceau de l'Université Bretagne Loire

pour obtenir le titre de
DOCTEUR DE L'UNIVERSITE DE BRETAGNE LOIRE
Mention : Informatique
Ecole doctorale SICMA

présentée par : Petra Bosilj

UMR 6074

Université Bretagne Loire

Institut de Recherche en Informatique et Systèmes

Aléatoires

Image indexing and retrieval using component trees

Thèse soutenue le 25 Janvier 2016 à Vannes
devant le jury composé de :

Jiří Matas
Professor, Czech Technical University in Prague // rapporteur

Philippe Salembier
Professor, Universitat Politècnica de Catalunya // rapporteur

Jean-Marc Ogier
Professeur, Université La Rochelle // examinateur

Michael H. F. Wilkinson
Senior Lecturer / Associate Professor, University of Groningen // examinateur

Erchan Aptoula
Associate Professor, Okan University // examinateur

Sébastien Lefèvre
Professeur, Université de Bretagne-Sud, IRISA // directeur de thèse

Ewa Kijak
Maître de Conférences, Université Rennes 1, IRISA // directeur de thèse

I would like to thank my advisors, Sébastien Lefèvre and Ewa Kijak, for all their guidance and advice.

A special thanks to the reviewers whose comments have helped improve the manuscript, and all the collaborators I have worked with during my Ph. D. studies.

I also want to express my gratitude to all my friends and especially my family, and all their support, understanding and encouragement that helped me keep my focus.

Contents

Chapter 1

Introduction

Contents

The field of image processing belongs to the discipline of signal processing dealing with processing of analog and digital signal, as well as storing, filtering and performing other operations on those signals. While image processing can be further divided into analog and digital image processing, the focus of this thesis are the applications belonging to the digital image processing field.

In digital image processing, the input signals are images represented as two-dimensional, discrete functions which can take on a finite range of values, representing the image intensity. The field of digital image processing refers to processing such signals with a digital computer [74]. While the related field of computer graphics is easy to distinguish from image processing, as it deals with the formation of images from object models as can be viewed as the "other side of the medal" to the image processing field, the boundary between image processing, image analysis and computer vision is somewhat less clear. A simple distinguishing criterion usually defines image processing operations to be those where both the input and output information are images. However, simple tasks such as computing the average intensity in an image would not be included in image processing under this definition [74]. The field of computer vision is usually considered to deal with more complex understanding tasks, with the goal of emulating human visions, learning and being able to

take actions based on input. As such, there also exists an overlap with artificial intelligence as well as machine learning, as the techniques from these fields are used to achieve image understanding.

While computer vision and image analysis tasks can be said to perform high-level processing on images where the goal is to "make sense" of the objects recognized in the image and perform cognitive functions associated with vision, we can define image processing as a field dealing with low-level and mid-level processing on images [74]. Here, the low-level processing involves primitive operations such as noise reducing preprocessing techniques, contrast enhancement and image sharpening. Mid-level processing then further processes the images, but typically outputs the attributes extracted from those images, such as a segmentation of the image into regions or objects, description of the objects as well as their classification. Image processing can thus be defined to encompass the processes whose inputs are images, and the outputs are either images or attributes extracted from the images (up to and including the recognition of individual objects) [74]. These methods work on images obtained by different acquisition techniques, such as X-ray imaging, satellite and radar imaging, as well as imaging in the visible and infrared bands, and includes processes such as image enhancement, image sharpening and restoration, image segmetation, representation and description as well as recognition and retrieval.

In the next section, we present different representations used in image processing, ranging from the simplest pixel-based representations to complex representations suited for various specific image processing applications. Different representations are used for different specific application domains within the field and depend both on the nature of the images being processed as well as the intended application, and we focus in this work on hierarchical image representations. The chosen application domain, image retrieval (and the related domain of classification) is presented in Sec. 1.2 which gives a short overview of the general image retrieval systems (a more detailed introduction to image retrival is given later). Finally, Sec. 1.3 summarizes the contributions presented in the rest of the thesis and gives the overview of the organization of the manuscript.

1.1 Image Representations

Many different image representations exist and, according to their properties, are suited for different application domains. Accuracy of the representation, redundancies present, the size of the representation and the number of elements, as well as the relations between the elements of the representation all have to be considered. For example, if the goal is to store a large collection of images, a representation using as little memory as possible while still allowing for perfect image reconstruction would be preferable. On the other hand, if the

goal is to manipulate with the represented image, the size of the representation is not as important as the direct access to image data allowing for easy modification of the image.

Here, we list several different families of image representations as well as their principal characteristics:

- *Pixel-based* representation of an image is the simplest to define, with elements in simple neighboring relations [175] and containing only direct, uninterpreted intensity (or color) information. In contrast with the simplicity of this representation is a large number of elements to be examined with no previous interpretation of associated local information [158].

- *Block-based* representations divide the image into the set of (rectangular) arrays of pixels. Different block-based representations have been developed for both binary images [118, 67, 147] and grayscale images [215, 48, 47].

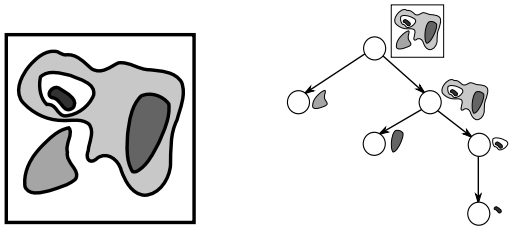
  The number of elements is slightly reduced compared to pixel-based representation, but the representation still does not include any interpretation of image data. Most common application include image compression [48, 215, 118], segmentation [147, 215], sliding window techniques [92] and efficient extraction of various features and attributes from the images [47, 147, 67].

- *Compressed domain* (or frequency domain) representations store the image as a set of coefficients in the transform domain. Different representations are based on Fourier transform [58, 200], wavelet theory [200, 103], Gabor wavelets [94], ridgelets [61], contourlets [60] etc.

  Some of typical uses of this representation include image compression [103, 200], denoising [104, 61], reconstruction [94] and texture analysis and segmentation for images [35]. While these representations reduce the size of the image, they are sensitive to translation, rotation and scaling of the image [119]. Additionally, in the frequency domain it is difficult to manipulate localized image content.

- *Region-based* representations differ from block-based in a way that regions are created by grouping similar and connected pixels, usually using a segmentation algorithm. The algorithm used typically produces an over-segmentation and the resulting regions are often called *superpixels* [1]. Information about the region adjacency is kept, usually in a region-adjacency graph (RAG) [156] or combinatorial maps [97].

  Different approaches to calculating the segmentation of the image into superpixels have been explored, e.g. normalized cuts [168], graph-based segmentation by Felzenszwalb and Huttenlocher [66] or different approaches to watershed segmentation [204,



(a) (b)

Figure 1.1: The tree in (b) is an example of a hierarchical representation of (a)

52]. A comparison between different approaches to superpixel calculation is presented in [1]. The theory of segmentation and their mathematical properties were also studied in-depth by [165, 155]. The number of regions is reduced compared to pixel-based representations, while the representation accuracy can be kept [158]. Still, a generic method for automatic segmentation of an image into semantic objects remains an open question and in order to detect semantic structures (e.g. objects) in the data, different unions of multiple regions have to be considered [202].

- *Hierarchical* representations propose most likely unions of regions (of a region-based representation) on *different scales* of the image, storing fine image details as well as coarse simplifications of images [202]. While they are built on (partial) segmentations, hierarchical representations hold more information than a simple collection of nested segmentations of an image. In addition to storing *horizontal* relations between regions (i.e. regions at the same level of detail), they also encode *vertical* relation between regions at different image scales which enable analysis of object details and provide the information on inclusion relations between the objects.

  The first applications were focused on image filtering and segmentation in the framework of Mathematical Morphology [88, 159, 158], and hierarchical representations are still used for this kind of applications [50, 178]. They bridge the gap between the classical filtering and segmentation techniques [161], enabling the construction of connected operators by simplifying different hierarchical representations. Various other applications have emerged since, such as object detection [202], video segmentation [159], image simplification [173, 120], feature extraction [136], image retrieval and classification [183, 182, 190, 9] and image registration [119]. An example of such an image decomposition is given in Fig. 1.1. More exhaustive list of applications is given later, according to hierarchy type (cf. Tab. 3.2).

Hierarchical representations of images are in the focus of this work. Ever since emerging, these representations have aimed to find better ways to capture the semantic information

about the image and propose complex regions corresponding to "meaningful" objects (components) of the image [88, 159]. For this reason, the term *component tree* was used to describe first proposed hierarchical representations [88]. Recently, many different such hierarchical representations have been developed; Trees of Shapes [120, 184, 73], Binary Partition Trees [158, 202] and trees based on them (e.g. BPT by Altitude Ordering [51], Hierarchies of Minimum Spanning Forests [50]), α-trees [173, 142] and constrained connectivity hierarchies, such as (ω)-trees [173, 135] being some of them.

1.2 Image Retrieval and Classification

The validation of the work presented herein is done in image retrieval and classification, akin application fields from computer vision and image processing. While the goal of image retrieval is to retrieve the database images describing the same object or scene as the query, in image classification the previously known images have already been grouped into classes based on a common object or a scene they are describing and the query image is assigned to the appropriate class. This is typically achieved by means of computing a description of the image, known then as a global image descriptor [140, 195, 45, 206], a numerical representation of the image which can then be used to get a measure of image similarity.

However, due to problems caused by occlusion, as well as objects in a scene belonging to different planes and thus behaving differently under various transformations (e.g. translation and rotation), descriptor schemes based on locally detected regions and features often tend to be more powerful [163]. The detection of distinctive, invariant and discriminative local features is used to provide a compact representation of the image by only focusing on the salient areas of the image. The development of affine invariant detectors was driven by their robustness against viewpoint change as one of the most common scene transformations between images. Popular detectors rely on different approaches to detect salient regions and points, operating in scale space or based on image gradient [99, 19, 115, 4, 14], relying on edges and boundaries [90, 198] or image and region contrast [108]. Detections returned by different detection approaches are often complementary and can be used in combination. Depending on the application and the type of content in the images under examination, predetermined parts of the image can also be selected as local features, covering the image in dense patches sometimes extracted from a regular rectangular grid covering the whole image (with or without overlap) [93, 189, 219, 146, 33].

After selecting or detecting features and keypoints locally, local description methods are applied to the detected salient points [99, 19, 189, 3] which are then aggregated and stored in an index. In small scale retrieval systems, as well as some application domains such as image matching or registration, all the local descriptors can be stored directly and searched

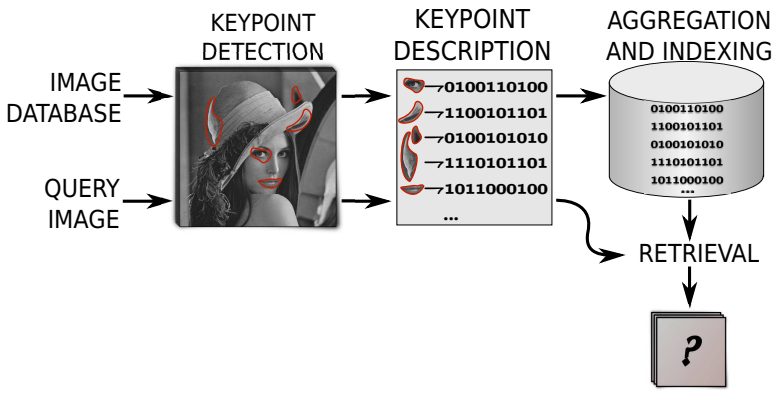


Figure 1.2: A representation of the main parts of the retrieval system based on a local approach. The steps of keypoint detection and description are performed for all the database images, after which the database descriptors are aggregated and stored in an index. For each new query image, the same steps of keypoint detection and description are first performed (the global query descriptor is also produced from the local ones if an aggregation step is included). Finally, index storing all the database descriptors is queried using the query descriptors to retrieve similar images from the database.

through using (approximate) nearest-neighbor based techniques [124, 96, 56, 167]. However, due to the curse of dimensionality [24, 27], especially when constructing a large scale image retrieval or object recognition system, the descriptors are often first aggregated before being stored in an index [170, 86, 16]. While some loss of information is always present, aggregating local descriptors using a combination of an aggregation and an indexing scheme produces a singular global descriptor for every image thus providing again a simple and efficient way to compare the similarity between two images. A simple depiction of such a scheme is shown in Fig. 1.2. Using different indexing schemes facilitates performing large scale database search thus making it possible to handle a very large collection of local descriptors, as well as balance the effects of uneven number of selected features coming from different scenes due to the difference in the type of content they represent.

1.3 Contributions and Content

The motivation to employ these hierarchical image representations for the classification and retrieval tasks comes from the specific semantic information captured by such hierarchies, which enables us to work directly on a reduced search space organizing the potentially salient image regions according to their complexity level as well as their inclusion relations to other regions. This claim is supported by the previous applications of such features to retrieval and classification, ranging from simpler approaches working with predefined classes

[183, 182] to more recent approaches where the trees and other morphological tools are used to perform large scale retrieval on either general databases, or specific databases comprising microscopic or satellite data [195, 190, 7, 136]. The saliency of the regions contained in the representations was previously demonstrated on the earliest of hierarchical representations, when an alternate detection algorithm for MSER regions [108] was presented using the Min and Max-tree hierarchies [136] as they contain all the potential MSER candidates as their building blocks. The claim of saliency and robustness of the regions represented by the trees is further supported by their wide use for segmentation and object detection [159, 202, 22, 112, 186]. In addition to working with salient regions directly, hierarchical representations were used to aggregate identifying information about the components they contain, thus producing distinctive descriptors. While previously only used as parts of global description schemes [195, 190] as (global) Pattern Spectra as well as to describe images at the pixel-resolution using Differential Attribute Profiles (DAP)[22, 55, 141] and more general and robust Differential Morphological Profiles (DMP) [21], these applications prove that discriminative information throughout different scales of the image can be sucessfully accumulated from examining the building blocks of such hierarchies.

Extending previous work exploiting hierarchical representations to construct different elements of the retrieval and classification systems, we present here several advances towards more versatile application of various component trees from mathematical morphology to these domains. The rest of the work presented herein is structured as follows:

- **Chapters 2 and 3** correspond to the morphological context of this thesis, offering an overview of different hierarchies present in the literature with the focus on explaining and comparing their structural characteristics as well as efficiency of computation. In addition to offering an exhaustive survey of the state-of-the-art hierarchies in a context different from other published works concerned with multiple such representations, we propose a classification of such representations into two superclasses, which follows the same argumentation in the hierarchical case as the seminal work of Serra [165] and Ronse [155] defining the relations between the frameworks of segmentations and partial segmentations of images. Included in the examination and formalization of the hierarchies from a general perspective, as well as each of the superclasses, is also a way of indexing i.e. assigning measures of scale or coarseness to the components in a hierarchy. The standard way of visualizing the indexed hierarchies using *dendrograms* [179] is extended to apply to both superclasses.

- **Chapters 4 and 5** deals with salient feature detection for image retrieval systems where local detection and description schemes are used. The presented work builds on the Maximally Stable Extremal Regions (MSER), a fast detector based on image intensity,

responding to blobs of high contrast and producing affine invariant, highly featured regions of arbitrary shapes [108]. Due to the hierarchical ordering of the extremal regions [63], all which are in turn contained in the Min and Max-tree hierarchies [159, 88], using the tree-based MSER algorithm [136] while replacing the hierarchy used corresponds to changing an ordering relation on the image pixels. Instead of detection regions based on strict intensity ordering, the detection algorithm can be applied to any component tree exhibiting invariant properties. We developed a new detector based on the Tree of Shapes [32, 34, 31], which we examine here together with α-tree and (ω)-tree based detectors and validate both in the standard matching framework by Mikolajczyk et al. [116]. As the Tree of Shapes detector exhibits the best performance, it is also evaluated in an image retrieval context using VLAD indexing [86].

- **Chapters 6 and 7** deal with image and feature description, and present and validate the usage of 2D Pattern Spectra [107, 195], with the focus on their calculation on regions of the image as local descriptors. Pattern spectra, the histogram-like structures originating in Mathematical Morphology, contain the information on the distribution of sizes and shapes of image components. As such, they are calculated on Min and Max-tree hierarchies, structures comprising all the components of the image, using a technique known as granulometry [37]. Their previous sucess in image retrieval applications [190] elicited the study into their behavior when applied to local patches as local descriptors. We examine the direct application of the standard calculation techniques to local patches [34], parameters to be used in initializing the descriptors [32] and finally achieving and validating the scale invariance properties of newly designed local versions of the Pattern Spectra [31]. The precursory experiments, examining the parameter choices, scale invariance and the stability under the change of scale settings, and performance dependance on the number of examples was done in a small retrieval setup [31]. In these experiments, the MSER regions [108] were used for computational efficiency, as they can also be computed using the same Min and Max-tree structures. Further validation was done in an image retrieval application targeting satellite imagery [33], where the descriptors were calculated on predetermined, densely selected local patches.

- **Chapter 8** is concerned with the indexing assigned to the hierarchies. While the construction algorithms always dictate (explicitely or implicitly) the coarseness measure, or level, to be assigned to each region present in the structure, this inherent measure does not, in the general case, accurately reflect the region complexity or level of region aggregation and can not be used directly to compare any two regions. However, knowing a coarseness measure for the objects of interest in the hierarchy prior to the main

image analysis or tree processing step could be used to rearrange the tree according to this more suitable metric. The proposed technique [30] can be interpreted as a tree filtering approach which means that the hierarchical relations between the remaining regions are preserved while removing the chosen regions simplifies the image and the representation. A hierarchy processed in this way (or the image reconstructed from it) can normally substitute a tree representation required by any application (including the retrieval techniques presented herein), providing the way to change the properties as well as limit the size of the search space used (comprising all the regions represented by the hierarchy).

- **Chapter 9** gives a final unification of the work presented herein. The performance and impact of all the presented techniques is summarized, offering an overview of open challenges, considered improvements and other potential application domains. The thesis concludes with offering the perspectives on the future research directions eminating from this work.

Chapter 2

Formalization of Component Trees

Contents

Partitions and partial partitions of the image domain can be viewed as the constituent parts of hierarchical representations. A unifying paper by Serra [165] presents the theory of connective segmentations, proving the equivalence of partitions and segmentations and presenting the lattices they make. This theory was further extended to allow handling of partial partitions and segmentations in the same framework by Ronse [155]. A choice of general lattice framework for studying the lattices made by (partial) partitions and segmentations and relations between them in both [165, 155] implies the theory can be adapted to different domains (e.g. images, speech, image sequences – videos). The papers [165, 155] also offer ways to combine different connections (and partitions) and to construct morphological operations on partitions, and further, constructing simple hierarchies by iterative application of connected (morphological) operators for producing nested segmentations is also considered.

In addition to the classic formalization of hierarchical representations through defining mandatory relations between the regions, a new approach to formalization through *stack of image region seeds* and *stackable hierarchies of regions* is presented (introduced in [29]) herein. This alternate definition corresponds more naturally to the way such hierarchies are con-

structed and treated, as it defines the representations through hierarchical inclusions of image details of increasing scale and coarseness. Additionally, the relations between parent – child nodes of such tree representations are explicated by the definition through the stackable hierarchies of regions. Following, two distinct superclasses of such representations are offered based on their similarities and differences (*inclusion* and *partitioning* trees, first briefly introduced in [30]). The distinction and relations between the two proposed superclasses is in accordance with the distinction between partitions and partial partitions presented by Ronse [155], on which the hierarchies are built on. For each of the presented superclasses, we identify the restrictions necessary to transform the general formalization of trees and their building elements (nodes) into the formalization for the specific category.

A consistent way of *indexing* (i.e. attributing scale parameters) the hierarchies is suggested for trees from either superclass. Properties introduced by indexing are examined, and a unified and formalized way to visualize the structure of such indexed hierarchies is presented based on *dendrograms* [179]. Since the ultrametric property present with (indexed) hierarchical clustering, as explained by Najman and Soille [135], dendrograms as well as ultrametric watersheds are proposed as convenient ways to represent such indexed hierarchies of partitions. We propose here an extension to the framework of representing indexed trees by dendrograms so that it allows indexing the inclusion trees in a way that holds meaningful information for the representation and while being similar to the classical way of indexing the partitioning trees. While theory allowing the representation of inclusion trees by dendrograms requires supplementing an inclusion tree with additional elements, the final representation for the inclusion trees, the *reduced dendrogram*, only depicts the original tree elements in a unambiguous way.

The framework chosen is purposefully simple and common in image processing: monochannel images represented by vertex-valued graphs, equipped with a standard 4-connectivity. In the context of [165, 155, 134], we could say we are working on the lattice of image regions. This choice was made to emphasize the focus on types of concrete hierarchies on images, their structure, properties and the way they incorporate information, independently and without the specific constraints of an application domain or liberties of a general framework. This way, we aim to provide a strong reference for understanding the basics of those hierarchies, which can then be apply and extended to more complex applications and needs.

The chapter begins by presenting the basic notions used throughout the thesis in Sec. 2.1. The component trees are formalized as stackable hierarchies of regions in Sec. 2.2, following which the categorization of the trees into superclasses based on their structure is offered in Sec. 2.3. Finally, Sec. 2.4 explains what indexing a hierarchy means, with special attention given to our proposed way of indexing inclusion trees, and how to display an indexed

hierarchy for each superclass.

2.1 Basic Notions

In this Section, the primary definitions from graph theory used throughout the thesis, such as graphs, trees and characteristics calculated on them, are revised and described.

Let I be a monochannel (e.g. grayscale) digital image which consists of a set of pixels, and $f : I \to \mathbb{N}_0$ a function that assigns to each pixel $p \in I$ its intensity value. The dual image, denoted by $-I$, is acquired by changing the intensity function: $f'(p) = lMax - f(p)$, where *lMax* is the maximum gray level allowed in the image (usually 255). In order to define the adjacency relations between the pixels of an image, we associate with the image an undirected graph $\mathcal{G} = (V, E)$.

The vertex set (set of nodes) V of the graph corresponds to the image pixels, and the edge set E consists of unordered pairs of vertices indicating the adjacency relations. Herein, we will be focusing on the most common, path-wise, connectedness for 2D images, such as 4- or 8-connectivity defined on the square image grid, or 6-connectivity on the hexagonal grid (cf. [175] for more details on connectivity). More complex connectivities exist [145, 144, 149], permitting, for example, handling of the objects made out of more than one connected component. However, the advanced hierarchies based on them [210, 149] are just briefly mentioned and not examined in detail. For a more theoretical analysis of connectivity and connections, the theory of *connective segmentation* [165] and its extension to partial connections [155] permit combining different connections.

If an edge between two pixels p and q exists it is denoted by $e_{p,q}$ or $e_{q,p}$ and the pixels p and q are said to be *adjacent* in $\mathcal{G}$. Sometimes, instead of working with pixel intensities, it is convenient to work with distances between adjacent pixels. In that case, we talk about an edge-weighted graph. The weight is assigned to every edge as the distance between the pixels connected by that edge:

$$F(e_{p,q}) = d(f(p), f(q)). \tag{2.1}$$

Most commonly, the intensity difference between the pixels is used as distance:

$$F(e_{p,q}) = |f(p) - f(q)|. \tag{2.2}$$

We distinguish the *image boundary pixels* as those pixels $p \in I$ that do not have the full set of neighbors, e.g. if 4-connectivity is used, boundary pixels are all the pixels with strictly less than 4 neighbors.

A *subgraph* of $\mathcal{G}$, denoted by $\mathcal{X} \subseteq \mathcal{G}$, is defined as $\mathcal{X} = (V_\mathcal{X}, E_\mathcal{X})$, where $V_\mathcal{X} \subseteq V$ and $E_\mathcal{X} \subseteq E$ such that $\forall e_{p,q} \in E_\mathcal{X} \implies p \in V_\mathcal{X}, q \in V_\mathcal{X}$. We say that a subgraph $\mathcal{X}$ is *spanning* for

the graph $\mathcal{G}$ if it covers all the vertices of the graph $\mathcal{G}$, i.e. $V_{\mathcal{X}} = V$ (but $E_{\mathcal{X}}$ can be different from E).

A *path* $\mathcal{P}$ in $\mathcal{X} = (V_{\mathcal{X}}, E_{\mathcal{X}}) \subseteq \mathcal{G}$ from p_1 to p_n is defined as $(p_1, \ldots, p_n)$ such that for all $1 \leq i < n$ the pixels p_i and p_{i+1} are adjacent in $\mathcal{X}$, that is $e_{p_i,p_{i+1}} \in E_{\mathcal{X}}$. A path from p_1 to p_n is a *cycle* if $p_1 = p_n$. For any two pixels p and q, we denote by $\mathrm{SP}(p, q)$ the set of all the possible paths in $\mathcal{X}$ (or in $\mathcal{G}$) between p and q. For any path $\mathcal{P} = (p_1, \ldots, p_n)$, the function $\mathrm{PD}(\cdot)$ calculates the *dynamics along the path*. The dynamics of the path is calculated as the sum of the edge weights along the path, e.g. if the intensity difference from Eq. (2.2) is used:

$$\mathrm{PD}(\mathcal{P}) = \sum_{i=2}^{n} F(e_{p_i,p_{i-1}}) = \sum_{i=2}^{n} |f(p_i) - f(p_{i-1})| \tag{2.3}$$

Two pixels p and q are connected in $\mathcal{X} = (V_{\mathcal{X}}, E_{\mathcal{X}}) \subseteq \mathcal{G}$ if and only if there is a path $\mathcal{P}$ in $\mathcal{X}$ from p to q or if $p = q$. A subgraph $\mathcal{X} \subseteq \mathcal{G}$ is said to be *connected* if all $p, q \in V_{\mathcal{X}}$ are connected in $\mathcal{X}$.

A *region* $\mathcal{R} = (V_{\mathcal{R}}, E_{\mathcal{R}})$ of I is defined as a closing $\varrho(\cdot)$ of a subgraph $\mathcal{X} = (V_{\mathcal{X}}, E_{\mathcal{X}})$:

$$\begin{aligned} \mathcal{R} = \varrho(\mathcal{X}) = (V_{\mathcal{R}}, E_{\mathcal{R}}) \\ \text{where } V_{\mathcal{R}} = V_{\mathcal{X}} \\ \text{and } E_{\mathcal{R}} = \{e_{p,q} \in E | p \in V_{\mathcal{X}}, q \in V_{\mathcal{X}}\}. \end{aligned} \tag{2.4}$$

A *connected region* or a *connected component* of the image I is a subgraph that is both connected and a region. Unless explicitly specified, all the subgraphs and regions in the remainder of the article will be connected.

A *region boundary* of a region $\mathcal{R} = (V_{\mathcal{R}}, E_{\mathcal{R}})$ is defined as the set of edges $E_{\mathrm{bound}}(\mathcal{R})$ given by:

$$E_{\mathrm{bound}}(\mathcal{R}) = \{e_{p,q} \in E | p \in V_{\mathcal{R}}, q \notin V_{\mathcal{R}}\}. \tag{2.5}$$

The set of pixels of the inner boundary is then made out of all the end-points of the boundary edges that belong to the region $\mathcal{R}$:

$$V_{\mathrm{inbound}}(\mathcal{R}) = \{p \in V_R | e_{p,q} \in E_{\mathrm{bound}}(\mathcal{R})\}. \tag{2.6}$$

Similarly, the set of outer boundary pixels comprises all the end-points of the boundary edges not belonging to the region $\mathcal{R}$:

$$V_{\mathrm{outbound}}(\mathcal{R}) = \{p \in V | p \notin V_{\mathcal{R}}, e_{p,q} \in E_{\mathrm{bound}}(\mathcal{R})\}. \tag{2.7}$$

All the image pixels not belonging to a connected region $\mathcal{R}$ ($p \in V$ but $p \notin V_{\mathcal{R}}$) make a set of 0 or more connected regions of maximal size in the image domain I, $\overline{\mathcal{R}} = \{\overline{\mathcal{R}_1}, \ldots, \overline{\mathcal{R}_k}\}, k \geq 0$. If a $\overline{\mathcal{R}_i}$ does not contain any image boundary pixels, it is called a *hole*

of the region $\mathcal{R}$. The operation of *filling all the holes* of a connected region, $H(\cdot)$, adds all the pixels contained in all the holes of a region $\mathcal{R}$ to that region $\mathcal{R}$:

$$H(\mathcal{R}) = \mathcal{R} \bigcup (\bigcup_i \overline{\mathcal{R}_i}), i \geq 0 \quad \text{such that } \forall i, \overline{\mathcal{R}_i} \text{ is a hole in } \mathcal{R}. \tag{2.8}$$

A set of connected regions $\mathcal{R}_\mathcal{S} = \{\mathcal{R}_1, \ldots, \mathcal{R}_k\}, k \geq 1$ is said to *partition* the image domain if it covers the entire image domain, and the elements of the set are mutually disjoint, i.e. when it holds:

$$(\bigcup_{\mathcal{R}_i \in \mathcal{R}_\mathcal{S}} V_{\mathcal{R}_i}) = V = I \quad \text{and} \quad \forall \mathcal{R}_i, \mathcal{R}_j \in \mathcal{R}_\mathcal{S}, i \neq j, \quad V_{\mathcal{R}_i} \cap V_{\mathcal{R}_j} = \emptyset \tag{2.9}$$

A partition is usually determined by segmenting the intensity function of an image, $f(\cdot)$, using one of more than a thousand image segmentation algorithms proposed in literature [165].

Sometimes, a segmentation algorithm will also determine boundaries between the regions, or contain a residual. This means that the region set returned will not cover the entire image domain, and will not be a partition. These kind of algorithms can be handled in the framework of *partial partitions* introduced by Ronse [155]. Similarly to Eq.(2.9), the elements of the set $\mathcal{R}_{\mathcal{PS}} = \{\mathcal{R}_1, \ldots, \mathcal{R}_k\}, k \geq 0$ *partially partition* the image if they are disjoint, but it is not required that they cover the image domain. In fact, the set:

$$\mathbf{supp}(\mathcal{R}_{\mathcal{PS}}) = (\bigcup_{\mathcal{R}_i \in \mathcal{R}_\mathcal{S}} V_{\mathcal{R}_i}) \tag{2.10}$$

is called the *support* of the partial partition, and $\mathcal{R}_{\mathcal{PS}}$ partitions the image on its support $\mathbf{supp}(\mathcal{R}_{\mathcal{PS}})$.

Flat zones of the image are connected regions of the image I of maximal size comprised only of pixels at the same gray level [166, 160]. Flat zone $\mathcal{F}_k$ at a gray level k can be described as:

$$\mathcal{F}_k = \{p_1, \ldots, p_l\}, l \geq 1 \quad \text{such that } \forall p_i \in \mathcal{F}_k, f(p_i) = k \quad \text{and } \forall p_j \notin \mathcal{F}_k, \text{ if } e_{p_i,p_j} \in E \text{ then } f(p_j) \neq k. \tag{2.11}$$

We call a flat zone of the image a *local maximum* if the flat zone is surrounded only by pixels of strictly lower gray level. A flat zone $\mathcal{F}_k$ is a local maximum if the following holds:

$$\forall p \in \{p_j \in V | e_{p_i,p_j} \in E, p_i \in \mathcal{F}_k, p_j \notin \mathcal{F}_k\}, f(p) < k. \tag{2.12}$$

Local minima of the image are similarly defined as flat zones surrounded only by pixels of strictly higher gray level. The regions of local minima and local maxima in the image together make the *local extrema* of the image.

We can also define local minima and maxima in an edge-weighted graph. A set of edges $E_{\min}$ is a local edge minimum if a region defined as $\mathcal{R}_{\min} = (V_{min} = \{p | e_{p,q} \in E_{\min}\}, E_{\min})$ is a connected region of the image and:

$$\forall u \in E_{\min}, \quad F(u) = \text{const.} \tag{2.13}$$

and

$$\forall u \in E_{\min},$$

$$\forall v \in \{e_{p,q} | p \in V_{\min}, q \notin V_{\min}\}, \quad F(u) < F(v). \tag{2.14}$$

The *upper* and *lower level sets* of an image are sets of image pixels with gray level values higher or lower than a gray level k, where each level set can comprise several connected components:

$$\mathcal{L}^k = \{p \in I | f(p) \geq k\} \tag{2.15}$$

$$\mathcal{L}_k = \{p \in I | f(p) \leq k\} \tag{2.16}$$

The difference between undirected graphs and directed ones is that the edge set of a directed graph consists of *ordered* pairs of pixels. The edge $e_{p,q}$ in a directed graph is called an edge *from p to q*, and does not imply the existence of $e_{q,p}$. The in-degree of a vertex p in a directed graph is defined as $\text{ID}(p) = \text{card}(\{e_{p,q} \in E\})$.

A tree $\mathcal{T} = (M, P)$ can be defined as a directed graph such that the underlying undirected graph $\mathcal{T}' = (M, P')$ (where $e_{m,n} \in P$ implies $e_{m,n} = e_{n,m} \in P'$) is connected and acyclic (contains no paths that are cycles), and such that for every $n \in M$, $\text{ID}(n) \leq 1$. The definition of a path $\mathcal{P}$ in a tree $\mathcal{T}$ is identical to the definition of a path $\mathcal{P}$ in a (sub)graph. If there exists an edge $e_{m,n}$ in a tree, m is called *the parent of n* and *n a child of m*. The vertex m is called *an ancestor of n* if there exists a path $\mathcal{P}$ in $\mathcal{T}$ from m to n. The nodes that have no children are called *leaf* nodes. The only node in the tree with no parent (i.e. $\text{ID}(p) = 0$) is called the *root of the tree*. Let $C = \{n_1, \ldots, n_k\}$ be a set of nodes. C is a *cut of the tree* if every path $\mathcal{P}$ from the root to any leaf passes through exactly one node $n_j \in C$.

2.2 Component Trees as Stackable Hierarchies of Regions

After defining graphs associated to the image domain, subgraphs, image regions, and trees, we propose to formalize the hierarchical structure behind the concept of component tree as a *stackable hierarchy of regions* (SHoR).

Every such hierarchy is based on, or "seeded" in a *stack of image region seeds* $\mathcal{S}$. $\mathcal{S}$ is a finite sequence of (sub-)graphs defined on a graph $G = (V, E)$ corresponding to an image I with the following properties:

$$\begin{aligned} &\mathcal{S} = (\mathcal{X}_0 = (V_0, E_0), \ldots, \mathcal{X}_l = (V_l, E_l)), \\ &\text{such that } \forall i \geq 1, E_{i-1} \subseteq E_i, \\ &\qquad V_{i-1} \subseteq V_i, \\ &\qquad \text{and } V_l = V. \end{aligned} \tag{2.17}$$

Additionally, every subgraph $\mathcal{X}_i$ is such that it can be decomposed into one or more connected subgraphs of the image:

$$\forall i, \mathcal{X}_i = (V_{i,0} \cup \ldots \cup V_{i,k}, E_{i,0} \cup \ldots \cup E_{i,k}), k \geq 0, \tag{2.18}$$

where each connected subgraph $\mathcal{X}_{i,j} = (V_{i,j}, E_{i,j})$ defines a connected region $\mathcal{R}_{i,j} = \varrho(\mathcal{X}_{i,j}) = \varrho(V_{i,j}, E_{i,j})$.

A *stackable hierarchy of regions* (SHoR) $\mathcal{H}_\mathcal{S}$ is then constructed from the stack of image region seeds $\mathcal{S}$ by closing all the different connected subgraphs $\mathcal{X}_{i,j}$ appearing in S in Eq. (2.17) and (2.18). $\mathcal{H}_\mathcal{S}$ is the set of all the connected components $\mathcal{R}_{i,j} = \varrho(\mathcal{X}_{i,j})$.

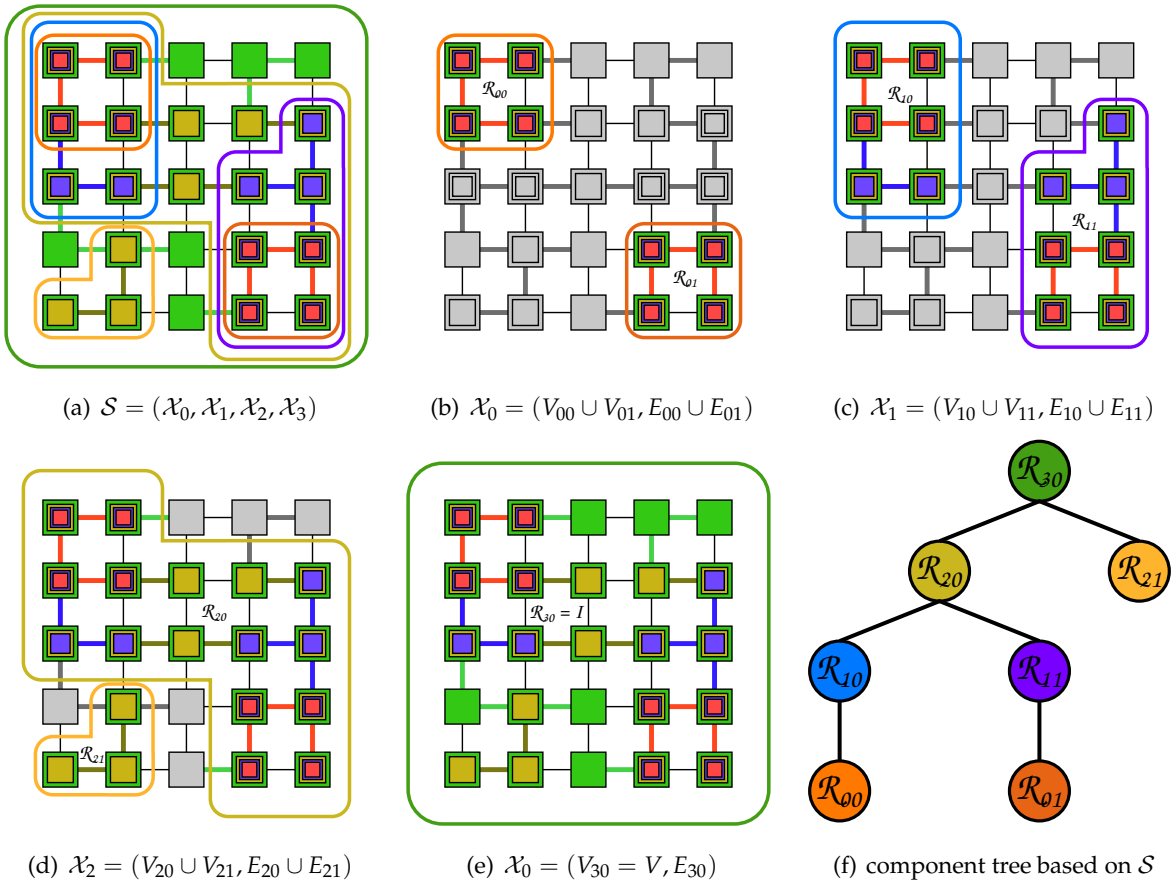
An equivalent definition of a stackable hierarchy $\mathcal{H}_\mathcal{S}$ of regions can also be written as:

i) $\mathcal{G} \in \mathcal{H}_\mathcal{S}$,

ii) for each two elements $\mathcal{R}_1, \mathcal{R}_2 \in \mathcal{H}_\mathcal{S}$ the following holds: $\mathcal{R}_1 \cap \mathcal{R}_2 \neq 0 \Rightarrow \mathcal{R}_1 \subseteq \mathcal{R}_2$ or $\mathcal{R}_2 \subseteq \mathcal{R}_1$.

An example of SHoR is shown in Fig. 2.1(a), with the subgraphs $\mathcal{X}_i$ of the corresponding stack of image region seeds displayed in Figs. 2.1(b) – 2.1(e).

The most straightforward way to represent inclusion relations between regions in such a hierarchy is by trees, where every node corresponds to a connected region of the image represented by the hierarchy. The regions in the leaf nodes correspond to small image details, coarse structures can be found in the nodes closer to the root, while the root of the tree corresponds to the whole image domain. Parental relations between nodes represent inclusion



(a) $\mathcal{S} = (\mathcal{X}_0, \mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3)$ (b) $\mathcal{X}_0 = (V_{00} \cup V_{01}, E_{00} \cup E_{01})$ (c) $\mathcal{X}_1 = (V_{10} \cup V_{11}, E_{10} \cup E_{11})$

(d) $\mathcal{X}_2 = (V_{20} \cup V_{21}, E_{20} \cup E_{21})$ (e) $\mathcal{X}_0 = (V_{30} = V, E_{30})$ (f) component tree based on $\mathcal{S}$

Figure 2.1: A full example of the SHoR and the stack of image region seeds is shown in subfigure (a). The color orange corresponds to nodes and edges constituting $\mathcal{X}_0$, purple for $\mathcal{X}_1$, yellow for $\mathcal{X}_2$ and green for $\mathcal{X}_3$. Subfigures (b) through (e) show the sub-graphs that are the building parts of $\mathcal{S}$. In the representation of every subgraph, $\mathcal{X}_i$, the grayed out nodes and edges, as well as black edges, are not part of the subgraph $\mathcal{X}_i$. Connected regions based on connected subgraphs of different $\mathcal{X}_i$ are encircled, and marked in the images. The stackable hierarchy of regions (SHoR) is finally equal to $\mathcal{H}_\mathcal{S} = \{\mathcal{R}_{00}, \mathcal{R}_{01}, \mathcal{R}_{10}, \mathcal{R}_{11}, \mathcal{R}_{20}, \mathcal{R}_{21}, \mathcal{R}_{30} = I\}$. The subfigure (f) displays the component tree corresponding to the SHoR in subfigure (a), where the colors used to enclose the connected regions $\mathcal{R}_{i,j}$ are utilized in the tree as the colors of the corresponding nodes.

relation between the regions, i.e. the set of pixels of the child region is a subset of the set of pixels of its parent (and all his ancestors). A simple example of such a tree, based on the SHoR from Fig. 2.1(a), is shown in Fig. 2.1(f).

To formalize the tree structure as a representation of the SHoR, some constraints are imposed on the general definition of the tree following the definition of $\mathcal{H}_\mathcal{S}$. In a tree $\mathcal{T} = (M, P)$ which corresponds to a SHoR of an image I (with corresponding $\mathcal{G} = (V, E)$), the

region represented by a node $n \in M$ is denoted by $\mathcal{R}(n) = (V(n), E(n))$. The root node of the tree $\mathcal{T}$, $r \in M$ such that $\mathrm{ID}(r) = 0$ corresponds to a region covering the whole image, i.e. $\mathcal{R}(r) = \mathcal{G} = (V, E)$. For any two nodes, n and m it is either true that their sets of pixels are disjoint, $V(n) \cap V(m) = \emptyset$, or one of the following holds: $V(m) \subseteq V(n)$ or $V(n) \subseteq V(m)$. If $V(m) \subseteq V(n)$, we say that n is an ancestor of m, i.e. n is one of the nodes on the path $\mathcal{P}$ from the root r to m. The relation between the region represented by a parent node and the regions represented by its children can be formalized as follows.

If m is a parent node in the tree and $n_1, \ldots, n_k$ are all the children of m, the following rules describe how to construct m from its children:

$$V(m) = V(n_1) \cup \ldots \cup V(n_k) \cup S(m), \tag{2.19}$$

where

$$\begin{aligned} S(m) = \{p_0, \ldots, p_l\}, l \geq 0 \\ \text{such that } \forall i \in \{0, \ldots, l\}, p_i \in I \\ \forall j \in \{1, \ldots, k\}, p_i \notin V(n_j). \end{aligned} \tag{2.20}$$

The edge set of the parent can be represented as:

$$E(m) = \{e_{p,q} \in E | p \in V(m), q \in V(m)\}, \tag{2.21}$$

and the pixel set $S(m)$ has to be such that the following holds:

$$\mathcal{R}(m) = (V(m), E(m)) \text{ is a connected region of } I. \tag{2.22}$$

The equation (2.19) dictates that a pixel set of the parent can be written as a union of the pixel sets of all its children, and optionally some additional pixels. The set of additional pixels $S(m)$ in Eq. (2.20) can be empty, allowing for the parent region to consist only of its (adjacent) children regions, but also allows us to construct a parent from non-adjacent regions by including new pixels using the set $S(m)$ so that the newly constructed region is still connected according to Eq. (2.22). The equation (2.21) only ensures that the newly constructed subgraph $\mathcal{R}(m)$ is indeed a region with the vertex set $V(m)$.

2.3 Categorization of Tree Representations into Superclasses

Equations (2.19)–(2.22) are general relations describing all types of trees. In this subsection, we present a categorization of all the tree classes into two superclasses and further constraints to the equations in order to specialize them for each of the superclasses. Such a categorization was already proposed in [30], and is explored here in more detail with added

illustrations and explanations. Based on the properties of the nodes and the nature of parent-child relations in the tree representations, we can distinguish between two different super-classes:

- **Inclusion trees**: the leaves contain only the finest image structures (typically, local extrema of pixel gray levels) and do not form a complete partition. Inner nodes are formed by region growing from the leaves until there is only one region (the root of the tree) covering the entire image domain.

- **Partitioning trees**: all the nodes of any cut of the tree form a full partition of the image. The initial partition contained in the leaves is a fine image segmentation. A parent node is an union of all its children with no additional pixels.

Such a categorization corresponds well to partitions and partial partitions: the stack of image region seeds $\mathcal{S}$ used in the construction of a partitioning tree will always comprise subgraphs that are partition of the image, while the set $\mathcal{S}$ of an inclusion tree will comprise partial partitions.

Inclusion trees. The leaf nodes of inclusion trees do not cover the whole image domain. Instead, they hold isolated points or small regions, typically local maxima or minima [88, 159] of the image, or both [120]. This way, the nodes in (and close to) the leaves correspond to bright or dark details of the image. As already mentioned, the stack of image region seeds $\mathcal{S}$ used in construction of an inclusion tree comprises partial partitions. The support of these partial partitions is nested, that is, the relations in Eq. (2.17) hold for any $\mathbf{supp}(\mathcal{X}_{i-1})$ and $\mathbf{supp}(\mathcal{X}_i)$ as well. Additionally, any cut of an inclusion tree is a partial partition as well.

New nodes are formed by a region growing process starting from the leaves, by adding one or more pixels (usually the whole image flat zones) to the regions in the leaf nodes. When the regions of two or more nodes merge in the course of this process, the newly constructed node becomes a parent of all the nodes representing the merged regions thus unifying several tree branches. This process continues until there is only one region covering the whole image domain, and the node representing this region becomes the root of the constructed hierarchical representation. In order to reflect the structure of inclusion trees, we add a further constraint in Eq. (2.19)–(2.22). The only modification is adding a strict inequality in Eq. (2.20), $l > 0$, to reflect that regions are only formed by adding new pixels to already existing regions (or a single region).

Simplifying the image represented by an inclusion tree includes cutting (removing) some branches from the leaves to the desired point (usually, up to a region satisfying a certain criterion). The areas of the removed regions are then assigned a gray level of the closest surviving ancestor node (i.e. the ancestor node with the greatest distance from the root) of

the regions that were cut off. This accomplishes removing small dark or bright structures in the image without changing the larger structures.

Partitioning trees. The principal difference of partitioning trees when compared to inclusion trees is that the leaves of the structure always form a (very fine) image partition. The same is true for any cut of such a tree [76] (as well as all the subgraphs $\mathcal{X}_l$ from the stack of image seeds $\mathcal{S}$ used in construction). The initial partition contained in the leaves can be the result of any segmentation algorithm, but among most common choices are the image pixels, flat-zones of the image [202, 173] or the result of watershed segmentation [100].

Regions of the inner nodes of the trees are formed by merging, as unions of the adjacent regions of other nodes, meaning that every new node has at least two child nodes. In contrast to the leaf nodes, a cut higher in the tree is a coarser segmentation of the image. To formalize this, a constraint $k > 1$ has to be added to Eq. (2.19) and $l = 0$ to Eq. (2.20) to reflect that no pixels are added that did not previously belong to a node.

Notions needed to define the iterative merging, which is at the core of the construction process of any partitioning tree were first introduced by Garrido et al. [72] (and only later put in the context of trees [158]):

1. *Region model* defines how simple regions and their unions are represented. It reflects the characteristics of the regions used in the construction process.

2. *Merging criterion* or *similarity* (or *dissimilarity*) *measure* describes the interest of possible merges. It is based on the region characteristics represented by the region model.

3. *Merging order* defines the rules used to merge the regions and which merge to perform next based on the merging criterion.

In order to simplify an image using a corresponding partitioning tree, a coarse enough cut is selected in the tree (the decision can be based e.g. on the number of desired elements of the partition or some more complex criteria). Each region represented by a node in the cut is then represented by a uniform gray level, which can be based on the region model, or take into account the average gray level of the region. This way, small variations in the gray levels of image pixels of can be removed from regions perceived as uniform. The difference between inclusion and partitioning trees is shown in Fig. 2.2.

2.4 Indexing the SHoR

While trees are sufficient to represent the inclusion relations between the regions in a SHoR, it is often desirable to assign an attribute to each node, corresponding to a measure of aggregation, called the *level* (of aggregation) of the node. Such an attribute λ is a non-negative

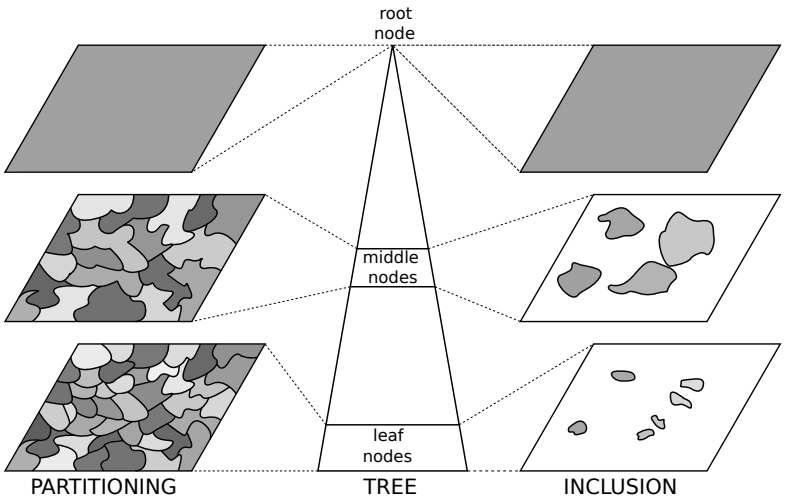
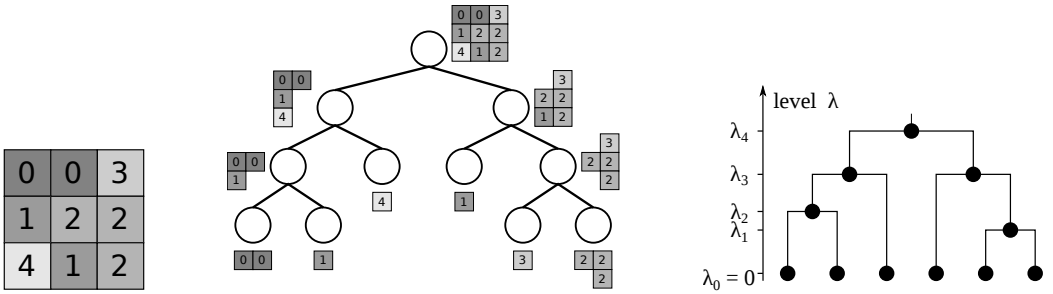


Figure 2.2: This image demonstrates the difference between the superclasses of partitioning and inclusion trees. Cuts of the partitioning tree near its bottom and the middle, as well as the root node are displayed on the left. A set of nodes from the inclusion tree close to the bottom and middle of the tree, and the root of the tree are displayed on the right.

function of the nodes. A tree with levels assigned in such a manner is then considered *indexed*. The attribute values λ are usually determined based on the definition or the construction algorithm for a specific tree. The rule for assigning the levels always reflects the fact that the coarseness of the nodes increases along each branch from the leaves towards the root and states that if m is an ancestor of n, then $\lambda(n) < \lambda(m)$. Hereafter, we explain the well-established representational framework for indexed partitioning trees [87, 83, 135]. We furthermore propose the way to represent the indexed inclusion trees in the same framework, whereas there is no current convention about the representational framework for inclusion trees.

An *ultrametric distance* is a constraint stronger than a *distance* on a set of elements, where the elements of the set obey an inequality stronger than the triangular inequality: the *ultrametric inequality*. An ultrametric inequality states that for any three elements of a set, $v_1, v_2, v_3 \in \Omega$, it is true that $d(v_1, v_2) \leq \max(d(v_1, v_3), d(v_2, v_3))$. If, while indexing a partitioning tree, we add an additional constraint $\lambda(n) = 0$ if and only if n is a leaf node, then there is a bijection between indexed partitioning trees and ultrametric distances [87, 83] defined on a same set. The definitions and construction algorithms of different types of partitioning trees always assign the same attribute value (usually $\lambda = 0$) to all the leaf nodes, so this additional constraint is in accordance with how the attribute value is naturally assigned for the partitioning trees. The levels of such an indexed partitioning tree induce an *ultrametric distance* on the nodes of the tree and the pixels of the image. To all the vertices of an image graph $\mathcal{G} = (V, E)$, with a corresponding SHoR and partitioning tree $\mathcal{T}$, we can assign the



(a) (b) (c)

Figure 2.3: Subfigure (b) shows a possible partitioning tree constructed for the image shown in subfigure (a). A dendrogram, corresponding to one possible indexing of the tree, is displayed in (c).

following ultrametric distance:

$$d(v_1, v_2) = \min\{\lambda(n) | n \in \mathcal{T}, v_1 \in V(n), v_2 \in V(n)\} \tag{2.23}$$

According to Eq. (2.23), a distance between any two image elements from I is given by the smallest level of a node n representing a region containing both image elements. Such indexed trees are conveniently represented in a form of a *dendrogram* [179], first introduced under the name *taxonomic tree* [172] for the purpose of hierarchical clustering. The height of each node in a dendrogram corresponds to the level assigned to that node (cf. Fig. 2.3). The reasoning behind using a separate representation for the structure of the hierarchy and to display the indexing imposed upon a hierarchy is that only the structure (in terms of inclusion relations) does not include all the information provided by the tree construction process. While comparing the tree structures would allow one to compare the composition of the image in terms of object and region inclusion, the indexing is usually needed when reconstructing the image and in other tasks where contrast between the regions (or other information used to construct the tree) is important.

In order to represent the inclusion trees in a similar manner, we propose extending them so their leaves partition the image. This corresponds to adding new child nodes to cover the regions previously added to the hierarchy through the non-empty sets $S(n)$ for every parent node n. Another issue is that the definitions and construction algorithms of inclusion trees often dictate assigning attribute values different from 0 and different from each other to the leaf nodes of the original tree (as will be detailed in Sec. 3). To resolve this in a uniform way, we will be adding new nodes as the children of all the original leaf nodes. As the extended tree is a partitioning tree, is it necessary to avoid having a node with a single child. For this reason, we will always be extending the tree with pairs of nodes.

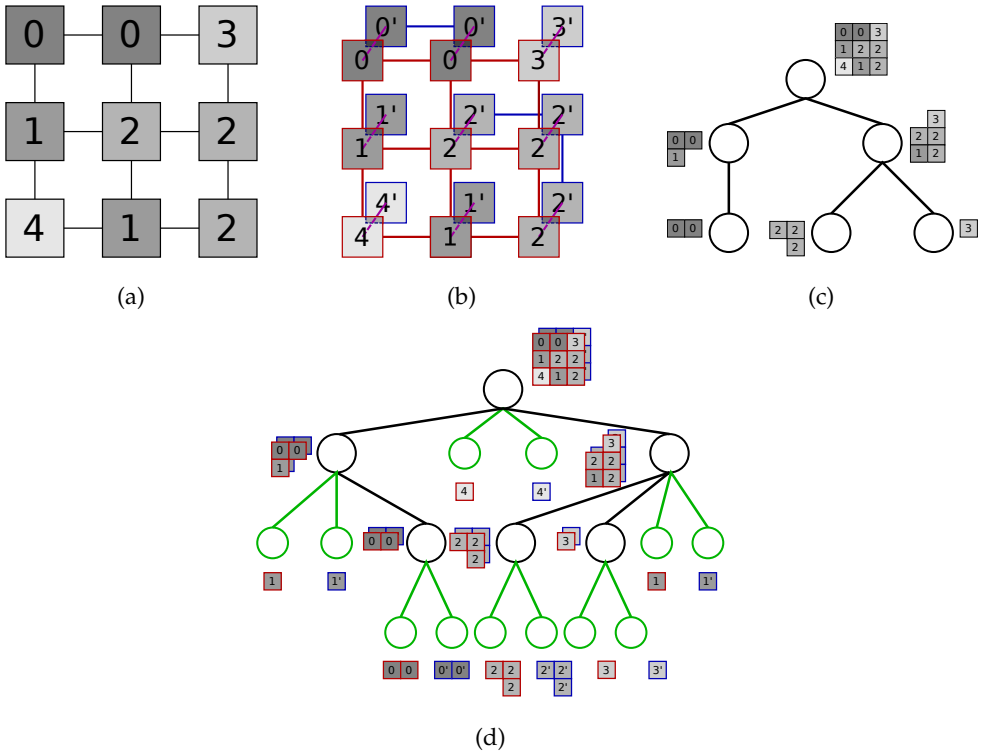


Figure 2.4: A possible inclusion tree for the image displayed in subfigure 2.4(a) is shown in subfigure 2.4(c). The extended image which includes the ghost regions is shown in subfigure 2.4(b), where the links between original image pixels are shown in red. The pixels of the ghost regions only have the connections to the ghost pixels belonging to the same ghost region, and are displayed in blue. Every ghost pixel is also connected to the corresponding pixel of the original image (purple links). The extended tree is shown in the subfigure 2.4(d) with the auxiliary nodes shown in green. The doubling of the represented regions for the inner nodes is due to the fact that they include both the original flat zones as well as their ghost region pairs.

All the nodes added by such an extension will be leaves of the extended tree, and will be considered *auxiliary nodes*. When extending the tree with the auxiliary nodes, we have to do it in a way that enables differentiating the auxiliary nodes from all the nodes present in the original tree. Henceforth, we explain our proposition for extending the tree in a way that can be indexed and represented by a (reduced) dendrogram.

First, we extend the original image with *ghost regions* corresponding to every flat zone (cf. the original image on Fig. 2.4(a) and extended image on Fig. 2.4(b)). For every flat zone $\mathcal{F}_k$ at every intensity level k in the image, a *ghost region* $\mathcal{F}'_k$ is considered to be connected to $\mathcal{F}_k$. A pair of auxiliary nodes holding the original flat zone region $\mathcal{F}_k$ and the corresponding ghost region $\mathcal{F}'_k$ are added to the tree. The parent of this new pair of auxiliary nodes is the

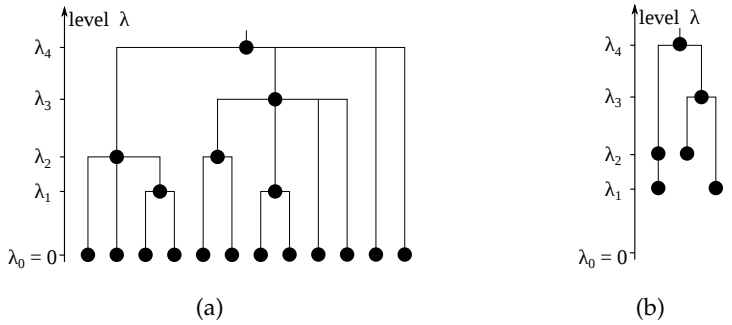


Figure 2.5: Subfigure (a) shows a dendrogram representation of a possible indexing of the extended tree displayed in Fig. 2.4(d). The reduced dendrogram corresponding to the same indexing is shown in subfigure (b). The only nodes displayed in the reduced dendrogram are the nodes of the original inclusion hierarchy, displayed in Fig. 2.4(c).

first node in the tree containing pixels of $\mathcal{F}_k$. In the extended tree, all the original nodes of the inclusion tree include all of their original flat zones, as well as all the corresponding ghost regions. The original tree, corresponding to the image in Fig. 2.4(a), is shown in Fig. 2.4(c). After extending the image with ghost regions, as in Fig. 2.4(b), the extended tree corresponding to this extended image is shown in Fig. 2.4(d). In the tree extended in this way, the leaf nodes of the original inclusion tree can be identified as the only nodes having only the (new) leaf nodes as children. Any other node n of the extended tree that has leaf nodes as children has had a non-empty set $S(n)$ in the original tree.

All the auxiliary nodes, that is the nodes representing the flat zones and their ghost regions, are assigned the level 0, $\lambda(\mathcal{F}_k) = \lambda(\mathcal{F}'_k) = 0, \forall \mathcal{F}_k, \mathcal{F}'_k$. In order to ensure $\lambda(n) > 0$ for all other nodes of the tree, we add a constant value to the attribute assigned to every node by the construction algorithm or based on the tree definition. For reasons of simplicity, in most examples, this constant will be equal to 1. With this kind of extension, the Eq. (2.23) holds for indexed extended inclusion hierarchies as well.

The inclusion tree extended in this manner can be directly represented by a dendrogram, and a dendrogram corresponding to the possible indexing of the extended tree of Fig. 2.4(d) is displayed in Fig. 2.5(a). However, since the auxiliary nodes are usually not in the focus of the representation, we propose to represent the inclusion trees by *reduced dendrograms*. All the auxiliary nodes have the attribute $\lambda(n) = 0$, and are the only nodes with this attribute value (a constant is added to attributes of all other nodes). Thus, we propose to simply omit them from the representation. In the resulting reduced dendrogram, the auxiliary nodes are still considered to be present, but are hidden and not displayed. An example of a reduced dendrogram indexing the tree in Fig. 2.4(c), and corresponding to the same possible indexing as the full dendrogram in Fig. 2.5(a), is shown in Fig. 2.5(b).

When indexing the inclusion tree, we encounter the same problem as Ronse when trying to represent an output of a segmentation algorithm producing a residual in the framework of connective segmentations and partitions. Directly adding the residual to the representation (or in our case, the nodes covering the regions of non-empty sets $S(n)$) to cover the whole image domain makes them indistinguishable from the original leaf nodes. However, while Ronse constricted the domain of the partition to the support [155], we instead chose to extend the domain. Constricting the domain in an (inclusion) hierarchy is difficult as the support changes through the hierarchy (cf. Subsec. 2.3, the supports are nested). By instead doubling the domain, we can extend an inclusion to a partitioning hierarchy (on an unchanging domain), and index it.

Chapter Summary

In this chapter, the basic notions used herein are first introduced. In addition to traditional formalization of hierarchical image representations, we offer a new formalization through stackable hierarchy of region (SHoR). Based on this formalization, we propose a classification of trees into two superclasses, namely inclusion and partitioning tree. Finally, indexing is introduced as a way to assign a level of aggregation to the elements of the hierarchy, imposing an ultrametric distance on the elements of the hierarchy. We extend the existing indexing principles as well as a representation and visualization framework using dendrograms used to handle partitioning trees, and apply them to inclusion hierarchies as well.

In the next chapter, we present different examples of both partitioning as well as inclusion hierarchies. An indexing method (i.e. a way to assign levels to the hierarchy) reflecting the construction process of each examined tree is proposed. Each inclusion tree is represented by its reduced dendrogram, and each partitioning tree by its dendrogram.

Chapter 3

Overview of Component Trees

Contents

This chapter presents the details and characteristics of a large number of hierarchical image representations, based on the comprehensive study by the author [29]. Their structure is presented withing the context of a taxonomy based on simplifications in the definition of the hierarchies applicable to a large number of tree representations. Indexing the hierarchies is done in an established framework based on dendrograms, presented and extended in Chap. 2 to enable indexing the full range of presented hierarchies. This comprehensive presentation of the trees and their characteristics was complemented by a summary of construction algorithms used to implement the hierarchies. The interest in such hierarchies is validated by the recent increase in processing techniques interacting with image regions or superpixels rather than individual image elements and requiring a representation extending through multiple scales, as well as a wide range of application domains attempting approaches based on trees specifically.

Table 3.1: Classification of the presented trees.

tree		
inclusion	Min and Max-tree (Sec. 3.1)	
	Tree of Shapes (Sec. 3.2)	Topological ToS (Subsec. 3.2.1)
partitioning	Binary Partition Tree (Sec. 3.3)	BPT by Altitude Ordering (Subsec. 3.3.1)
		Hierarchies of MSF (Subsec. 3.3.2)
	α-tree (Sec. 3.4)	
	(ω)-tree (Sec. 3.5)	

The characteristics of 5 distinct hierarchical representations, as well as 3 different special cases of those hierarchies are analyzed in detail and compared. In addition to explaining the structure of each hierarchy, they are mutually compared based on their characteristics of duality, ability to represent objects, completeness of the representation and complexity of construction. Additionally, the possibility of adapting the different representations using parametrization, where applicable, is also explored. The high-level, detailed study of the tree characteristics presented here offers a way to compare the presented representations independently of the intended application as well as it offers an extensive number of references concerning the recent advances as well as seminal historical work pertaining to the representations.

The characteristics the trees exhibit are examined and summarized for every introduced tree, as well as compared to characteristics of other presented trees. First, the trees from the inclusion tree superclass are listed, followed by partitioning trees. Following the definition of each tree, the most efficient algorithms suited for their implementation are discussed. The summary of different trees and their sub-types according to this classification is shown in Tab. 3.1. The chapter concludes by considering and comparing all the characteristics of all the trees considered in Subsec. 3.6

3.1 Min and Max-trees

Min-trees (and their dual structure, *Max-trees*) are from the superclass of inclusion trees. The concept and examples will first be given for the Min-tree structure, with the duality between trees explained at the end of the section.

The Min-tree is a structure aimed at representing the dark structures of the images. The

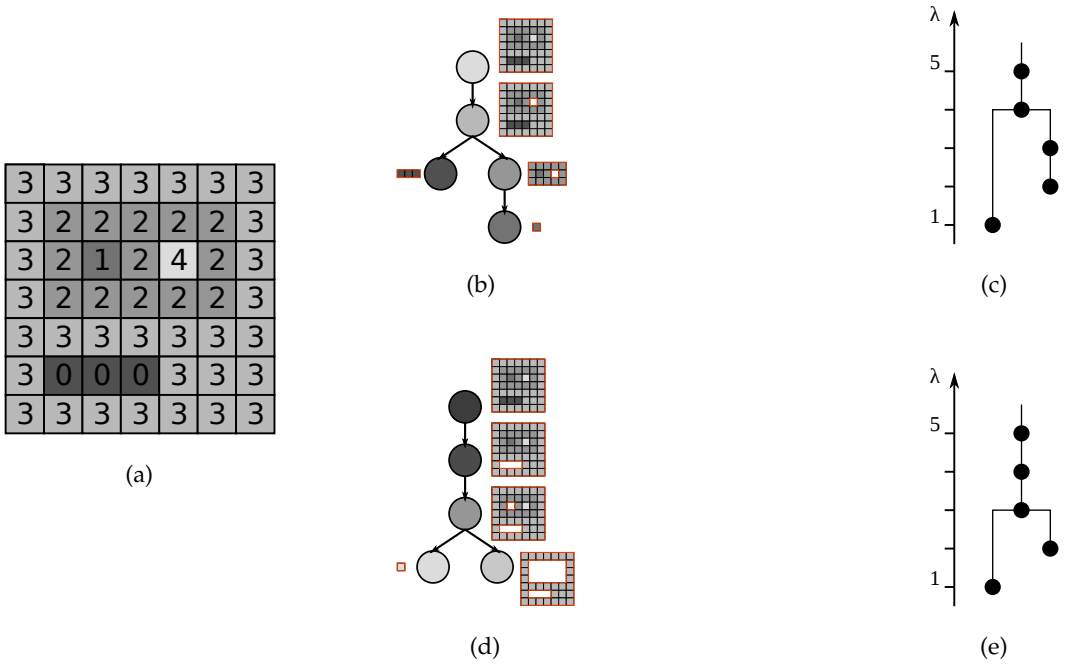


Figure 3.1: The original image is displayed in subfigure (a). Subfigure (b) shows the corresponding Min-tree, with the reduced dendrogram shown in (c). The Max-tree and its reduced dendrogram for the same image are shown in subfigures (d) and (e). Regions corresponding to the nodes in both subfigures (b) and (d) are shown next to the nodes (the white parts do not belong to the regions).

leaves of the image represent the regions corresponding to local minima in the image. All inner nodes are connected components of lower level sets of the image.

A connected component of the level set $\mathcal{L}_k$ will make a new node n_k with a region $\mathcal{R}(n_k)$. This node can either become:

- a *parent node* to all the previously constructed nodes at lower levels which are included in the region of the new node: $\mathcal{R}(n_{k'}) \subset \mathcal{R}(n_k), k' < k$,
- a *leaf node* if it does not include the regions of any previously constructed nodes.

Finally, the level set $\mathcal{L}_{lMax}$ at the highest gray level present in the image (usually $lMax = 255$) has only one connected component covering the whole image domain. This becomes the root of the tree, unifying all branches of the tree. An example of the Min-tree can be seen in Fig. 3.1(b).

This structure is not self dual: trying to construct a Min-tree of the dual image $-I$ will produce a different output since the local minima in the original image correspond to the local maxima of the dual. The dual structure of the Min-tree is the Max-tree: it can be seen as

either the Min-tree of the dual image, or the tree constructed in the same manner, but using the upper level sets of the image as components instead of the lower level sets.

Both Max- and Min-trees are complete representations of the image, since each allows full image reconstruction. The Min-tree is suitable for manipulating the dark image objects and Max-tree allows for easy manipulation of bright objects. Still, for dealing simultaneously with bright and dark objects, it is not satisfactory to keep both trees, as they are redundant. If we modify one of the trees, e.g. the Max-tree and the associated upper level sets, it is hard to ensure consistency with its dual tree [43].

We can assign levels to the nodes of the Min-tree corresponding to the lower level sets to which the regions they represent belong to. When extending the inclusion tree, a constant value is added to the attribute assigned to every node (cf. Sec. 2.4 for the detailed explanation). The level assigned to a node will then be calculated as the level of the corresponding lower level set increased by one. The lowest leaf nodes will thus be at level 1, representing the connected components of the set $\mathcal{L}_0$.

The construction process of the Min-tree already assigns levels to the nodes and produces an indexed tree. Even though the level of the nodes in the Min-tree does not directly reflect the coarseness of the region, it carries information. It corresponds to the gray level at which the node was first created and the intensity of the brightest pixel (the highest gray level) in the associated region. The example of a reduced dendrogram for the tree in the Fig. 3.1(b) is shown in Fig. 3.1(c). When indexing a Max-tree for an image I, the nodes are assigned the levels they would have if the tree was constructed as a Min-tree of an inverted image $-I$ (where the nodes correspond to the same regions of the image). An example of a Max-tree and its reduced dendrogram are shown in Figs. 3.1(d) and 3.1(e).

The first mention of this structure in literature was done by Salembier, Oliveras, and Garrido [159] with the name Max-tree and by Jones [88] with the name Component Tree. Upon closer inspection it can be seen that the information retained by each part of the two structures is equivalent, and the only difference is in the way the structure is stored [23]. The node n corresponding to a region $\mathcal{R}(n)$ in the component tree in [88] stores the whole set of pixels $V(n)$ of the region. The nodes in the Max-tree from [159] correspond to the same regions, but for parent nodes, only the pixels not belonging to any of the children nodes are stored which corresponds to the set $S(n)$ from Eq. (2.20). More complex hierarchies were developed based on the Min-tree and Max-tree, such as the Dual-Input Max-Tree for mask-based connectivity [145, 210] and hierarchies for hyperconnectivity [149]. However, since the advanced connectivities are out of scope of this work, these hierarchies are not discussed further herein.

Min and Max-tree construction. The complexity is expressed in relation to the number of image elements (pixels), N. Where relevant, quantization is also mentioned with q being

the number of bits used to code pixel values, and k the range of those values (i.e. $k = 2^q$).

A good recent comparison of Min-tree and Max-tree construction algorithms was offered by [40]. The algorithms were divided into three classes: the *immersion algorithms*, the *flooding algorithms* and the *merge-based algorithms*. The merge-based algorithms are mainly used for parallelism, and are not further examined here.

The first efficient algorithm for Max-tree computation was proposed by Salembier et al. [159] and belongs to the class of flooding algorithms. The Min-tree construction starts with a root pixel, at gray level *lMax*, from which the depth-first propagation through image elements is performed (using a hierarchical queue). For low quantization ($q \leq 16$ bit), both the original recursive version by Salembier et al. [159], and the non-recursive version by Nistér and Stewénius [136] achieve linear complexity $\mathrm{O}(kN) \sim \mathrm{O}(N)$, since k is small. However, for bigger q and k, the complexity does not reduce to linear but instead to quadratic, $\mathrm{O}(N^2)$, as most of the flooding algorithms need quantized data, and do not work for generic pixel values (e.g. *float* values). However, a recent flooding Max-tree algorithm by Wilkinson [210] replaces the hierarchical queue of [159] with a priority-queue, achieving a $\mathrm{O}(N \log N)$ complexity for any pixel type with a total order (including high dynamic range integers and float values).

The first step of immersion algorithms is *sorting all the image elements* according to the appropriate order (e.g. lowest-to-highest gray level for the Max-tree construction) and building N disjoint singleton sets, one for each image element. Those *sets are then merged to form a tree* in the second step, using Tarjan's union-find algorithm [185]. The complexity of the sorting step varies with the quantization: it can be bound to $\mathrm{O}(N + k)$ for small integers (typically $q \leq 12$ bit or $q \leq 16$ bit), but increases to $\mathrm{O}(N \log N)$ for generic data types. The time complexity of the union-find algorithm can be lowered by using techniques such as root path compression, union-by-rank and level compression (although it has to be balanced with memory usage). Time-wise, the most efficient union-find implementation has the quasi-linear complexity of $\mathrm{O}(N \times \alpha(N))$[1]. Both algorithms presented by Berger et al. [23] and Najman and Couprie [131] can be bound to a quasi-linear time complexity of $\mathrm{O}(N \times \alpha(N))$[1] for low quantized data, i.e. $q \leq 12$ bit. Since these algorithms are not dependent on a hierarchical queue, they can handle any quantization and any type of pixel value, with the worst-case complexity of $\mathrm{O}(N \log N)$ (same as in [210]).

[1] $\alpha(N)$ - very slow growing "diagonal inverse" of the Ackermann's function, $\alpha(10^{80}) \simeq 4$.

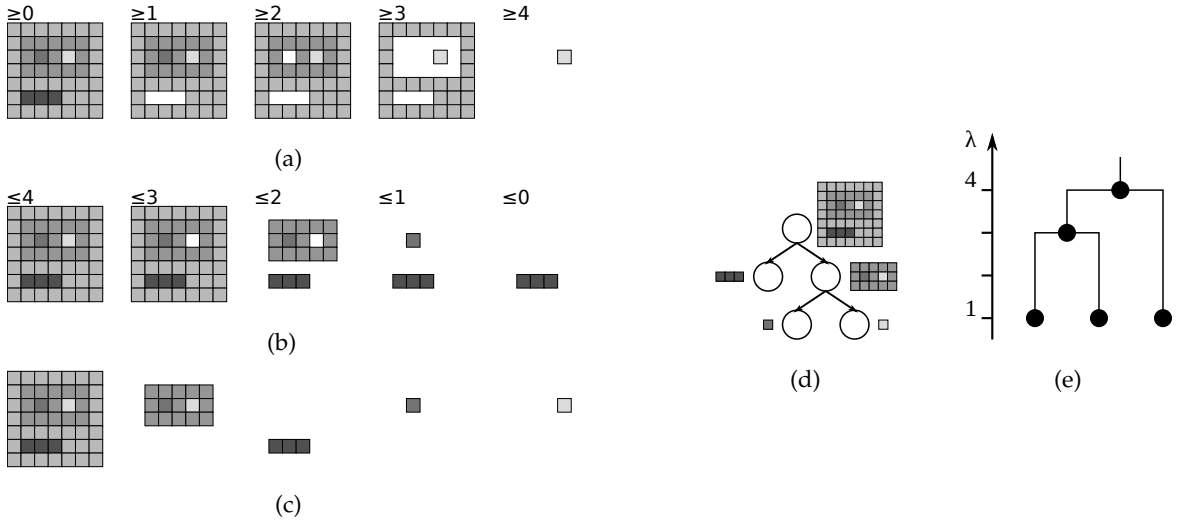


Figure 3.2: Subfigures (a) and (b) show upper and lower level sets of the image from Figure 3.1(a). All the different regions acquired by filling the holes of the level sets are displayed in subfigure (c). Finally, the Tree of Shapes of the image, composed of shapes displayed in (c) is displayed in subfigure (d). The corresponding dendrogram is shown in subfigure (e).

3.2 Tree of Shapes

Tree of Shapes, also belonging to the superclass of inclusion trees, is a structure aimed at representing both bright and dark structures of the image. It is a combination of Max and Min-trees [121], with even the first construction algorithm relying on calculating the Tree of Shapes from already constructed Max and Min-trees [120].

The leaves of this tree correspond to all local extrema of the image. Nodes of the tree correspond to *shapes* – connected regions acquired by applying the hole filling operation $H(\cdot)$ on all the connected components of all lower and upper level sets of the image. Shapes defined in this way have a property that they do not intersect, but instead either contain one another or are disjoint (cf. [17, 184] for the proof of this property). All the upper and lower level sets of the image in Fig. 3.1(a) are displayed in Figs. 3.2(a) and 3.2(b). After applying the operation $H(\cdot)$ on all the regions in Figs. 3.2(a) and 3.2(b), there are only 5 distinct shapes remaining, displayed in Fig. 3.2(c). Finally, the tree formed as a hierarchy of these shapes is displayed in Fig. 3.2(d).

A node n is defined to be the parent of node m if the shape $\mathcal{R}(n)$ is the smallest shape containing the shape $\mathcal{R}(m)$. Let $\mathcal{R}'$ be any shape obtained by filling the holes of a connected component of either upper or lower set of the image. The parent-child relation between nodes n and m means that $\mathcal{R}(m) \subset \mathcal{R}(n)$ and there is no shape $\mathcal{R}'$ such that $\mathcal{R}(m) \subset \mathcal{R}' \subset \mathcal{R}(n)$. The regions corresponding to local image extrema are the leaves of the tree because

there are no smaller shapes contained in the local extrema regions.

The Tree of Shapes is a self-dual structure: constructing the tree on the dual image $-I$ instead of the original image I would produce the same output. It is most easily seen for extrema: all local image minima become the local maxima, all the local maxima become the local minima and the regions included in the set of image extrema do not change. This further extends to the level sets of the image, where the upper level sets become the lower level sets in the dual image $-I$ and vice versa, while there is no change to the set of shapes of the image.

The structure is also a complete representation of an image, allowing for full reconstruction. But, unlike using both a Max-tree and a Min-tree to treat bright and dark image objects simultaneously, representing the image by the Tree of Shapes is non-redundant. The redundancy in combining the Max-tree and the Min-tree is eliminated by modifying the connected components of the level sets and using only shapes constructed by filling the holes of the connected components.

Indexing the Tree of Shapes relies on the notion of dynamics along a path, and the notion of *region dynamics*. Region dynamics $\mathrm{RD}(\cdot)$ of a region $\mathcal{R}(n)$ is chosen between all minimal dynamics along the paths between all the possible pixel pairs of that region, as the largest such path dynamic:

$$\mathrm{RD}(\mathcal{R}(n)) = \max\{\, \min\{\mathrm{PD}(\mathcal{P}) | \mathcal{P} \in \mathrm{SP}(p,q)\} \\ | \; p \in S(n), q \in \mathcal{R}(n)\}. \tag{3.1}$$

The level of the node n associated to the region $\mathcal{R}(n)$ is then equal to the region dynamics of the region increased by one (due to extending the inclusion tree explained in Sec. 2.4):

$$\lambda(n) = \mathrm{RD}(\mathcal{R}(n)) + 1. \tag{3.2}$$

The Tree of Shapes indexed in such a manner can then be represented by a reduced dendrogram, as shown in Fig. 3.2(e). To calculate the level of the node at the height 3 in the example of Fig. 3.2(e), we observe two different path dynamics, between the newly added node pixels and the pixels belonging to each of the children nodes. These region dynamics are 1 (from the leaf node at the gray level 1) and 2 (using the leaf node at gray level 4), and the higher one is used (increased by 1 according to Eq. (3.2) as the node level in the indexed tree.

This type of structure was independently presented by Monasse and Guichard [121] under the name *level line tree*, and by Song and Zhang [184] under the name *monotonic tree*. The structure presented by Monasse and Guichard in [121] and the proposed construction algorithm [120] use a combination of 4- and 8-connectivity when defining the shapes and their holes. The tree is defined through level sets and their boundaries, *level lines*. On the other

hand, Song and Zhang use 6-connectivity for the monotonic tree in both definition and algorithm [184, 182] and rely on inclusion of the *monotonic lines* to define a hierarchy. Outward-falling monotonic lines are defined for a grayscale image in [184] as boundaries of regions $E_{\text{bound}}(\mathcal{R})$ where all the pixels of the inner boundary assume higher values than any of the pixels of the outer boundary, i.e. $\forall p \in V_{\text{inbound}}(\mathcal{R}), \forall q \in V_{\text{outbound}}(\mathcal{R}), f(p) > f(q)$ (with a similar, reversed definition for outward-climbing monotonic lines). Upon closer inspection, the monotonic lines and the level lines of the image are equivalent, which is acknowledged in the following works by Song and Zhang, where they adopt the terminology of level lines and level sets [181]. The name "Tree of Shapes" is used in this manuscript, chosen because it prevails in recent literature [43, 73].

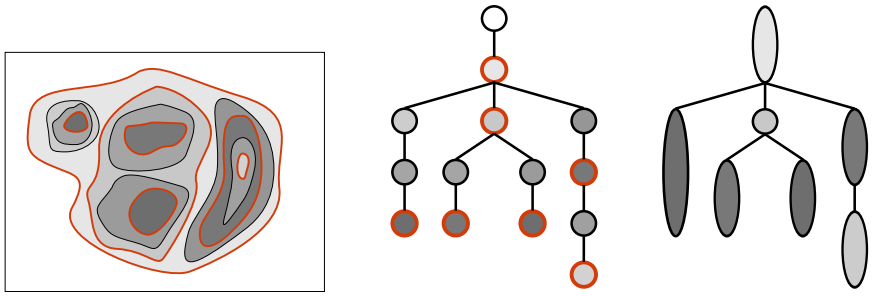
Tree of Shapes construction. Early approaches to ToS construction had drawbacks, such as ineffective extension to multidimensional (nD) images with more than 2 dimensions [120], or worst-case quadratic time complexity $\mathrm{O}(N^2)$ [43, 181]. Recently, a quasi-linear algorithm was proposed by Géraud et al. [73], which is easily applicable to nD images with low quantization (authors recommend $q \leq 12$ bit). The approach presented in [73] uses the immersion algorithms for Max-tree construction as a canvas, replacing only the sorting step.

In general, the sorting step in an immersion algorithm sorts the image elements so that image elements from the external shapes come before the elements from the internal shapes. For the ToS computation, this is achieved by representing the image as a set-valued map on a Kahlimsky grid [133]. This representation contains elements that materialize inter-pixel spaces and possesses some continuous properties with respect to both the domain and value space, which in turn enables the computation of the correct shape order for the Tree of Shapes. The implementation of the new sorting step depends on a hierarchical queue (with some modifications), so it can only handle quantized data. The time complexity of this new sorting step is $\mathrm{O}(kN)$, which can be considered linear for low quantized data. Since the immersion algorithm used for Max-tree construction, which is also used here, has the complexity of $\mathrm{O}(N \times \alpha(N))$, the whole algorithm can be considered quasi-linear with the complexity $\mathrm{O}(N \times \alpha(N) + kN) \sim \mathrm{O}(kN)$.

3.2.1 Topological Tree of Shapes

In [184], Song and Zhang propose a reduced structure based on the Tree of Shapes (under the name *topological* or *reduced monotonic tree* [184, 182] and *topological level line tree* [181]). This reduced tree aims to represent only the changes in the topology of the image.

A *monotonic sequence* is a maximal sequence of uniquely enclosing level lines of the same type (either outward-falling level lines produced as boundaries of lower level sets, or outward-climbing level lines resulting from upper level sets). In a tree representation where



(a) (b) (c)

Figure 3.3: The original image is displayed in subfigure (a). Subfigure (b) shows the corresponding Tree of Shapes. The Topological Tree of Shapes based on the same image (a) is shown in (c). The topological level lines in (a) and the nodes representing them in (b) are marked with a bold red outline.

the nodes represent the level lines, all the nodes belonging to a monotonic sequence except the first one are the only children of their parents. The last level line of such a sequence can then be called a *topological level line*, and all the nodes belonging to the monotonic sequence in the Tree of Shapes are reduced to only the node corresponding to the topological level line in the reduced hierarchy. The difference between the Tree of Shapes and the Topological Tree of Shapes is shown schematically in Fig. 3.3. Such nodes, corresponding to topological level lines, represent the areas in the image where the topology of the image changes [181]:

- a leaf node – contour creation/deletion (cf. Fig. 3.3(c), e.g. left branch), defines the "peak of a hill" or "bottom of a lake",
- a node with a child of the opposite type – contour creation/deletion (cf. Fig. 3.3(c), right branch), defines a "hole in a hill" or a "lump in a lake",
- a node with multiple children – contour merge/split (cf. Fig. 3.3(c), middle branch), defines a single mass splitting in two (e.g. a "hill with two peaks").

The Topological Tree of Shapes is a reduced hierarchy, and thus not a complete image representation. The self-dual nature of the Tree of Shapes is preserved when reducing it to the Topological Tree of Shapes. To index the Topological Tree of Shapes, a formula similar to Eq. (3.1) is used, except the dynamic along the path is replaced with *the number of topological changes along a path*. The number of topological changes along $\mathcal{P}$ is equal to the number of pixels along the path with intensities different from their successor pixel, i.e. $\text{card}\{x_j | f(x_j) \neq f(x_{j+1}), x_j \in \mathcal{P}, x_{j+1} \in \mathcal{P}\}$.

The *Topological Tree of Shapes* can be constructed by simple filtering from the ToS, and this process is linear in the number of nodes of the tree. As the ToS has at most N nodes, where N

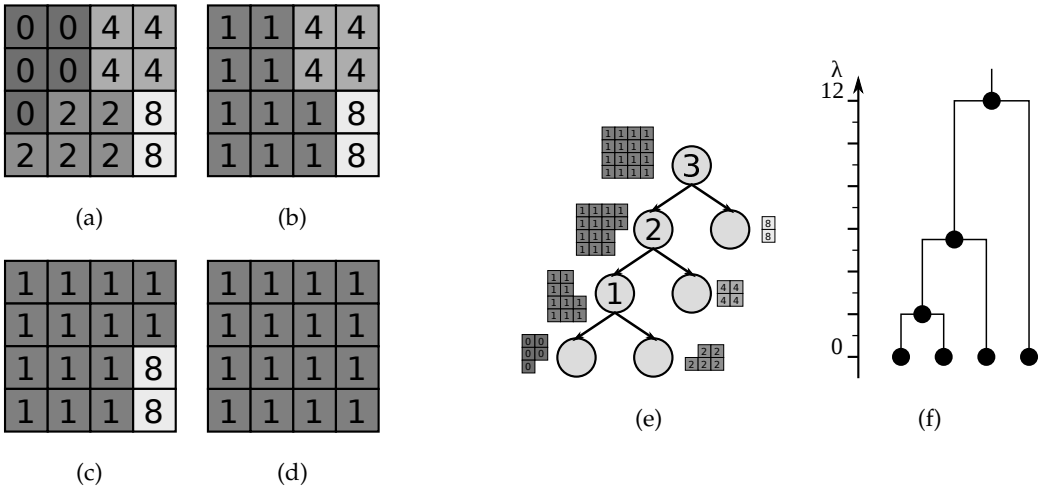


Figure 3.4: Creation of Binary Partition Tree. The original image is displayed in (a). The region model here is a constant gray level. When merging two regions, the model of the new region is the gray level of larger of the two children regions, or average gray level in case the merged regions have the same size. The merging criterion defines the dissimilarity between regions as the difference between their gray level models. Merging order dictates merging the pair of least dissimilar regions first. The initial partition comprises flat zones. Three merging steps are represented in (b), (c) and (d). The constructed BPT is displayed in (e) and the numbers in the nodes indicate the merging order. The dendrogram corresponding to the indexed tree is shown in (f).

is the number of image elements, the total construction complexity for the Topological Tree of Shapes is $\mathrm{O}(N + kN) \sim \mathrm{O}(kN)$.

3.3 Binary Partition Tree

Binary Partition Tree (BPT) belongs to the superclass of partitioning trees. Unlike the presented inclusion trees, this tree is not *extrema oriented* [158] and is thus suited for representing objects with low, high and intermediate gray levels.

This tree starts with each node being assigned a region from the *initial partition*, which can be as fine as having each pixel or flat zone as a region, or any more complex precomputed partition [72, 158]. When constructing a BPT, one needs to decide on the *region model* and *merging criterion*. The region model can be as simple as the average or median gray level of a region, and is typically assumed to be constant within a single region.

An example of a simple merging criterion (dissimilarity measure) is the absolute value of the difference between region models. While constructing merging criteria, both color

information (luminance as well as chrominance) and contour information can be taken into account. An important characteristic of the merging criterion is the dependence of the criterion on the region size – merging criteria that are size independent tend towards producing partitions with a small number of large regions and numerous extremely small regions [202]. A good discussion on various criteria and models is offered in [202].

The *merging order* used for all BPT states that the two regions most similar according to the merging criterion (i.e. the regions with the smallest dissimilarity) should be merged in the next step, with arbitrary choice of regions in case there is more than one pair of most similar regions. After constructing the initial partition and calculating the representations (region models) for all the regions, the dissimilarity measure is computed for each pair of neighboring regions. When two regions are merged, a new node for the region comprising all the pixels from both merged regions is constructed as a parent node to the merged regions. The similarity information is also updated before the next merging step. The merging sequence and the resulting tree for a BPT using a simple region model and criterion are shown in Fig. 3.4.

The length of the path from the tree root to the nodes does not reflect the complexity of the regions and the constructed tree is usually not well balanced (cf. tree examples in [158, 202]). Thus, indexing the BPT takes into account the values of the similarity measure in each merging step of the tree construction. The levels are assigned to nodes as follows:

- if a node m of $\mathcal{T}$ is a leaf node, then $\lambda(m) = 0$,
- if a node m is created as a union of regions corresponding to the nodes n_1 and n_2 (i.e. $\mathcal{R}(m) = \mathcal{R}(n_1) \cup \mathcal{R}(n_2)$), and the dissimilarity between the regions in the moment of merging was $D(\mathcal{R}(n_1), \mathcal{R}(n_2))$, the level of the new node m is calculated according to:

$$\lambda(m) = \max(\lambda(n_1), \lambda(n_2)) + D(\mathcal{R}(n_1), \mathcal{R}(n_2)). \tag{3.3}$$

A corresponding dendrogram for the tree in Fig. 3.4(e) is shown in Fig. 3.4(f).

As with all the partitioning trees, the most precise reconstruction of the original image is equal to the precision provided by the initial partition. Contrary to the inclusion trees, formed as a decomposition of the images (usually according to gray levels), the inner regions in the partitioning trees are produced through unifying the regions of the initial partition.

The self-duality of the constructed tree depends on the used region model and merging criterion. The (part of) merging criterion relying on contour information will not affect the self-duality of the tree in any way, while it is usually of interest to choose a region model and part of the merging criterion based on color homogeneity in such a way that they would produce the same results on a dual image (the merging criterion used in Fig. 3.4 is one such example).

The Binary Partition Tree was first introduced by Salembier and Garrido [158] where various image processing applications were considered as well as memory requirements needed to handle and store the structure. The BPT introduced in [158] is an extension of previous work by Garrido et al. [72], where a general merging algorithm is presented. While in [72], the authors note that the presented merging algorithm indeed produces partitions in hierarchical relations, keeping track of the merging steps performed in the algorithm is in fact what finally defines the BPT in [158]. The work was continued in [202], with the focus on object detection and a more suitable merging criterion. Binary Partition Tree and especially simplification methods were studied in [100]. Many trees based on Binary Partition Tree have been developed [132, 50, 82, 201], some of which are explained hereafter.

Binary Partition Tree construction. The algorithm for computing the BPT starts with the initial partition, and then iteratively updates the similarity information between the neighboring regions and merges the two most similar ones until only one region is left. The algorithm has been described in detail, elaborating the need for using a hierarchical queue for keeping the similarity information [72]. However, we only managed to find evidence in the literature of measured execution times [72, 202, 2], with one exception being the recently published paper [2] which offers the complexity analysis for a special case of 4-connectivity.

The iterative algorithm becomes the algorithm for hierarchical (agglomerative) clustering when no specific connectivity is imposed on the image elements [64]. Let us first analyze the simpler case, when the hierarchical queue is not used. The number of regions in the initial partition can be as high as the number of image elements N. As we have no connectivity specified, we have to calculate the similarity information for every pair of regions in the first step, with the complexity $\mathrm{O}(N^2)$. Next, we iteratively find the pair of most similar regions, remove the similarity information between the two selected regions and all of the other regions, merge those regions and re-calculate the similarity information between the new regions and all the others. Finding the smallest value in an unsorted list will have a complexity linear in the number of unsorted elements: $\mathrm{O}(N^2)$. Removing the old similarities and inserting the new ones has to pass all the regions once, bounding the complexity at $\mathrm{O}(N)$. Finally, the number of iterations is equal to the number of regions, which gives the final complexity of $\mathrm{O}(N^2 + N \times (N^2 + N)) \sim \mathrm{O}(N^3)$ when hierarchical queue is not used.

If the hierarchical queue is used for maintaining the similarity information, the algorithm requires an additional sorting step after calculating the initial similarities and before the first iteration. The complexity of this sorting step, if we have up to N regions and up to N^2 similarities, is bounded by $\mathrm{O}(N^2 \log(N^2)) \sim \mathrm{O}(N^2 \log N)$. However, determining the pair of most similar regions now becomes constant instead of quadratic, lowering the final complexity of the algorithm despite the fact that the cost of updating the information increases from $\mathrm{O}(N)$ to $\mathrm{O}(N \log N)$. Finally, the construction complexity of BPT in the case of general con-

nectedness relation can be bound by $O(N^2 + N^2 \log N + N \times (N + N \log N)) \sim O(N^2 \log N)$. It is important to note that in case of a specific connectivity, the speed of calculating the initial similarities, as well as updating the similarity information, might significantly decrease depending on the merging criterion used (the specific case analyzed in [2] approximates an upper bound of $O(\frac{2}{3}N^2) \sim O(N^2)$. However, the theoretical upper bound independent of the connectivity remains the same.

3.3.1 Binary Partition Tree by Altitude Ordering

Unlike other trees presented so far, weights are assigned to edges between pixels of the underlying image graph $\mathcal{G}$ when calculating the BPT by Altitude Ordering. Instead of using the weights assigned to image pixels, the initial partition, the region model and the similarity measure are based on the edge-weighted graph. A weight of the edge is calculated as the difference in intensities between the pixels (vertices) connected by the edge (cf. Eq. (2.2)).

In addition to assigning weights to the edges of $\mathcal{G}$, we also need an ordering relation $\prec$ on E which is a strict total ordering on E and also an *altitude ordering for* $F(\cdot)$. This means that, for any $u, v \in E$, if $u \prec v$ then $F(u) \leq F(v)$. The ordering relation $\prec$ is a parameter of the tree, and needs to be introduced since $F(\cdot)$ does not necessarily impose a strict total order on the edges. The weights assigned to two or more edges by $F(\cdot)$ can be equal, and an (arbitrary) ordering is needed to "break the ties" between equally-weighted edges. For any $k \in [1, \text{card}\,(E)]$, we denote by $u_k^{\prec}$ the k-th element of E with respect to $\prec$.

In order to fully define the BPT by Altitude Ordering, we need to define the initial partition, region model, and the merging criterion for the regions.

(1) The initial partition for BPT by Altitude Ordering is the partition to image pixels.

(2) Every region $\mathcal{R}$ can be represented by the region boundary edge set, $E_{\text{bound}}(\mathcal{R})$ (cf. Eq. (2.5)). Note that every edge $u \in E_{\text{bound}}(\mathcal{R})$ that is present in the region model of the region $\mathcal{R}$ will be present in the region model of exactly one more region $\mathcal{R}'$.

(3) The merging criterion ("dissimilarity measure") between two regions $\mathcal{R}$ and $\mathcal{R}'$ is the index k of the common edge $u_k^{\prec}$ belonging to the boundary of both regions, which is the smallest edge of E with respect to $\prec$:

$$D(\mathcal{R}, \mathcal{R}') = \arg\min_k \{u_k^{\prec} \mid u_k^{\prec} \in E_{\text{bound}}(\mathcal{R}), \; u_k^{\prec} \in E_{\text{bound}}(\mathcal{R}')\}. \tag{3.4}$$

This tree is always self dual, since the intensity differences between the neighboring pixels do not change when looking at the inverse image $-I$.

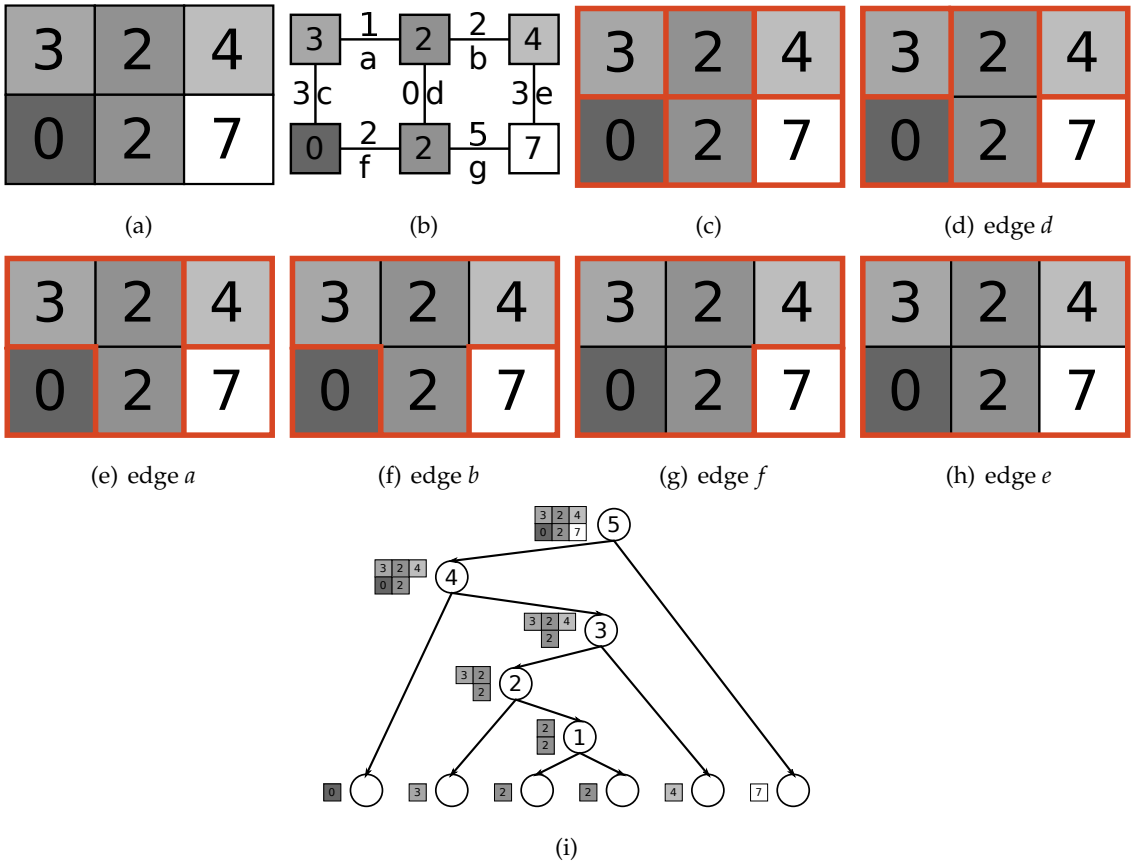


Figure 3.5: The original image is displayed in subfigure (a), with the corresponding edge-weighted graph on (b). The edges are ordered as $d \prec a \prec b \prec f \prec c \prec e \prec g$. The image partition after each merging of the regions is displayed in (c) – (h), where the edge used in the merging step is indicated under each partition. The BPT by Altitude Ordering is displayed in (i), and the numbers in the inner tree nodes enumerate the merging steps.

Such a hierarchy, with the construction based on Kruskal's Minimum Spanning Tree algorithm [91], was first explored by [123] where different segmentation methods based on the principal hierarchy were also proposed, as well as a possibility of different valuations of the graph edges. The concept was reintroduced to the community and formalized under the name of BPT by Altitude Ordering by Najman et al. [132], where different ways to post-process the tree were proposed. The relations of BPT by Altitude Ordering with other trees are explored in [51], where Cousty, Najman, and Perret prove that one can retrieve many hierarchies from this tree, such as Hierarchies of Minimum Spanning Forests (cf. next paragraph) and α-trees (cf. Subsec. 3.4). Figure 3.5 provides an example of BPT by Altitude Ordering.

The special case of BPT, the *Binary Partition Tree by Altitude Ordering*, can be computed

much faster. The construction of BPT by Altitude Ordering based on Kruskal's algorithm [91] is presented by Najman et al. [132]. The total complexity of construction depends on the speed of the sorting step, as the rest of the algorithm uses a disjoint-set data structure to keep track of the partially-constructed tree during the run of the algorithm, and runs with $O(N \times \alpha(N))$ time complexity when Tarjan's union-find is used [185]. Similarly to Min-tree construction, complexity reduction techniques such as *path compression* and *union by rank* are used to improve the performance of union-find algorithm. For images with low quantization, the sorting step is linear in the number of image elements so the total complexity is $O(N \times \alpha(N))$. For generic data types, the logarithmic complexity of the sorting step surpasses that of union-find, and the whole algorithm can be bound by $O(N \log N)$. The BPT by Altitude Ordering is a basis for construction of other hierarchies [51], and further post-processing algorithms are detailed in [132] for obtaining these other hierarchies from the BPT by Altitude Ordering.

3.3.2 Hierarchies of Minimum Spanning Forests

Similarly to BPT by Altitude Ordering, the calculation of Minimum Spanning Forests uses the definition of edge-weighted graphs and local edge minima (cf. Eq. (2.13)).

We say that a subgraph $\mathcal{X} = (V_\mathcal{X}, E_\mathcal{X})$ of $\mathcal{G}$ is *spanning* for $\mathcal{G}$ if $V_\mathcal{X} = V$. A *Spanning Tree* of a graph $\mathcal{G}$ is a connected subgraph that is spanning for $\mathcal{G}$ and has no cycles (i.e. is a tree). The weight of an edge-weighted subgraph $\mathcal{X}$ is equal to the sum of weights of all the edges of the subgraph: $F(\mathcal{X}) = \sum_{e \in E_\mathcal{X}} F(e)$. A *Minimum Spanning Tree* (MST) of $\mathcal{G}$ is a Spanning Tree $\mathcal{X} = (V_\mathcal{X}, E_\mathcal{X})$ whose weight $F(\mathcal{X})$ is less than or equal to the weight $F(\mathcal{Y})$ of any other Spanning Tree $(\mathcal{Y})$ of $\mathcal{G}$. More details about the classical graph theory problem of finding a MST for a given graph efficiently can be found in [91, 153, 75].

If we have two non-empty subgraphs of $\mathcal{G}$, $\mathcal{X} = (V_\mathcal{X}, E_\mathcal{X})$ and $\mathcal{Y} = (V_\mathcal{Y}, E_\mathcal{Y})$ (not necessarily spanning), we say that $\mathcal{Y}$ is *rooted* in $\mathcal{X}$ if $V_\mathcal{X} \subseteq V_\mathcal{Y}$ and if the vertex set of any connected component of $\mathcal{Y}$ contains the vertex set of exactly one connected component of $\mathcal{X}$ [52, 50]. $\mathcal{Y}$ is called a *Minimum Spanning Forest* (MSF) rooted in $\mathcal{X}$ if:

(1) $\mathcal{Y}$ is spanning for $\mathcal{G}$,

(2) $\mathcal{Y}$ is rooted in $\mathcal{X}$, and

(3) the weight of the graph $\mathcal{Y}$, $F(\mathcal{Y})$, is less than or equal to the weight of any other graph satisfying ((1)) and ((2)).

For a graph $\mathcal{G}$, any MSF rooted in a single vertex, i.e. in the graph $\mathcal{X} = (V_\mathcal{X} = \{v \in V\}, \emptyset)$, is a MST of that graph $\mathcal{G}$. By convention, we say that a MSF rooted in an empty graph $(\emptyset, \emptyset)$ is also a MST of $\mathcal{G}$ [51].

In [113], the equivalence between watersheds from markers and MSF is proven. This is further extended in [52], where the proof of equivalence of MSF and a possible definition of watersheds, called *watershed cuts*, is offered. For a MSF rooted in a graph $\mathcal{X}$, the graph $\mathcal{X}$ can be interpreted as a set of markers and used as a basis for the computation of watershed from markers. As subsets of minima of the original edge-weighted graph associated to the image constitute robust markers [13, 50, 51], they are also used as a basis for rooted MSF hierarchies. The set of all local minima edges of an edge-weighted graph is denoted by $M_{\mathcal{G}}$. An example image and the associated edge-weighted graph with the minima labeled are displayed in Fig. 3.6.

The definition of rooted MSF hierarchies was first introduced by Cousty and Najman [50]. The definition of the MSF hierarchy is parametrized by an ordered sequence $\mathbf{M} = (M_1, \ldots, M_l)$ of pairwise-distinct edge-minima of the graph. The subgraphs $\mathcal{X}_i$ of the stack of image region seeds $\mathcal{S} = (\mathcal{X}_0, \ldots, \mathcal{X}_l)$ (cf. Eq. (2.17)) are then computed such that:

$$\begin{gathered} \mathcal{X}_i \text{ is a MSF rooted in the graph } \mathcal{X}_i^{\text{root}} : \\ \mathcal{X}_i^{\text{root}} = \Big(V_{\mathcal{X}_i^{\text{root}}} = \{p | e_{p,q} \in M_{\mathcal{G}} \setminus (\bigcup_{j \in [1,i]} M_j)\}, \\ E_{\mathcal{X}_i^{\text{root}}} = M_{\mathcal{G}} \setminus (\bigcup_{j \in [1,i]} M_j) \Big) \end{gathered} \tag{3.5}$$

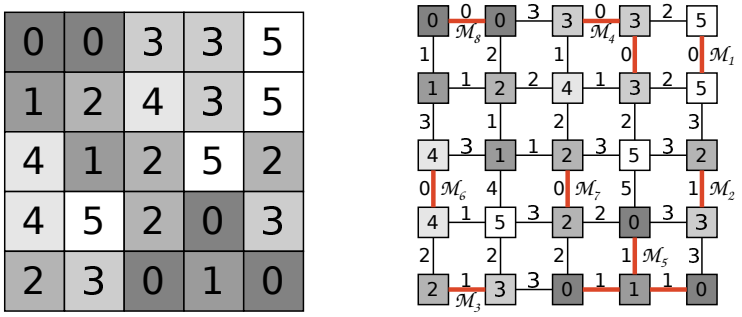
The SHoR is then constructed as described in Sec. 2.2. It is important to note that in order for MSF hierarchy to be a proper partitioning tree (i.e. $\varrho(\mathcal{X}_l) = \mathcal{G}$), the size of the sequence of minima $\mathbf{M}$ has to be $l = (\text{card}\,(M_{\mathcal{G}}) - 1)$. If $l < (\text{card}\,(M_{\mathcal{G}}) - 1)$, $\mathcal{X}_l$ will have multiple connected components instead of a single one covering the whole image domain. If $l = \text{card}\,(M_{\mathcal{G}})$, then we have $\varrho(\mathcal{X}_{l-1}) = \varrho(\mathcal{X}_l) = \mathcal{G}$ and the constructed hierarchy is the same as for $l = (\text{card}\,(M_{\mathcal{G}}) - 1)$.

To put the MSF hierarchy in the context of BPT, the initial partition, the region model and merging criterion for the regions have to be defined.

(1) The initial partition is comprised of connected components of $\mathcal{X}_0$, a MSF rooted in $M_{\mathcal{G}}$ where every connected component contains exactly one edge-minimum.

(2) At the iteration i, when determining $\mathcal{X}_i$, a region $\mathcal{R}$ is represented by two distinct components. The first component is the same as the region model for the BPT by Altitude Ordering, the set of region boundary edges $E_{\text{bound}}(\mathcal{R})$. The second component of the region model is the (only) minimum $M \in \{M_{\mathcal{G}} \setminus (\cup_{j \in [1,i]} M_j)\}$ contained in $\mathcal{R}$.

(3) The merging criterion used for calculating the dissimilarity between neighboring regions $\mathcal{R}_i$ and $\mathcal{R}_j$, containing M_i and M_j respectively, is a pair of values. The dissimilarity depends on the less significant of the two minima and on the distance between



(a) (b)

Figure 3.6: Subfigure (b) is the edge-weighted graph corresponding to the image in (a). The edge minima on (b) are emphasized by bold, red edges, and ordered by increasing importance. The indexed minima M_i form an ordered sequence $\mathbf{M} = (M_1, \ldots, M_l)$, where a higher index i of M_i indicates a minimum of higher importance. The ordered sequence of minima from (b) is used to construct the MSF hierarchy in Fig. 3.7.

the two regions:

$$D(\mathcal{R}_i, \mathcal{R}_j) = (k, \text{dist}(\mathcal{R}_i, \mathcal{R}_j)) \tag{3.6}$$

where

$$k = \min\{i, j\}$$

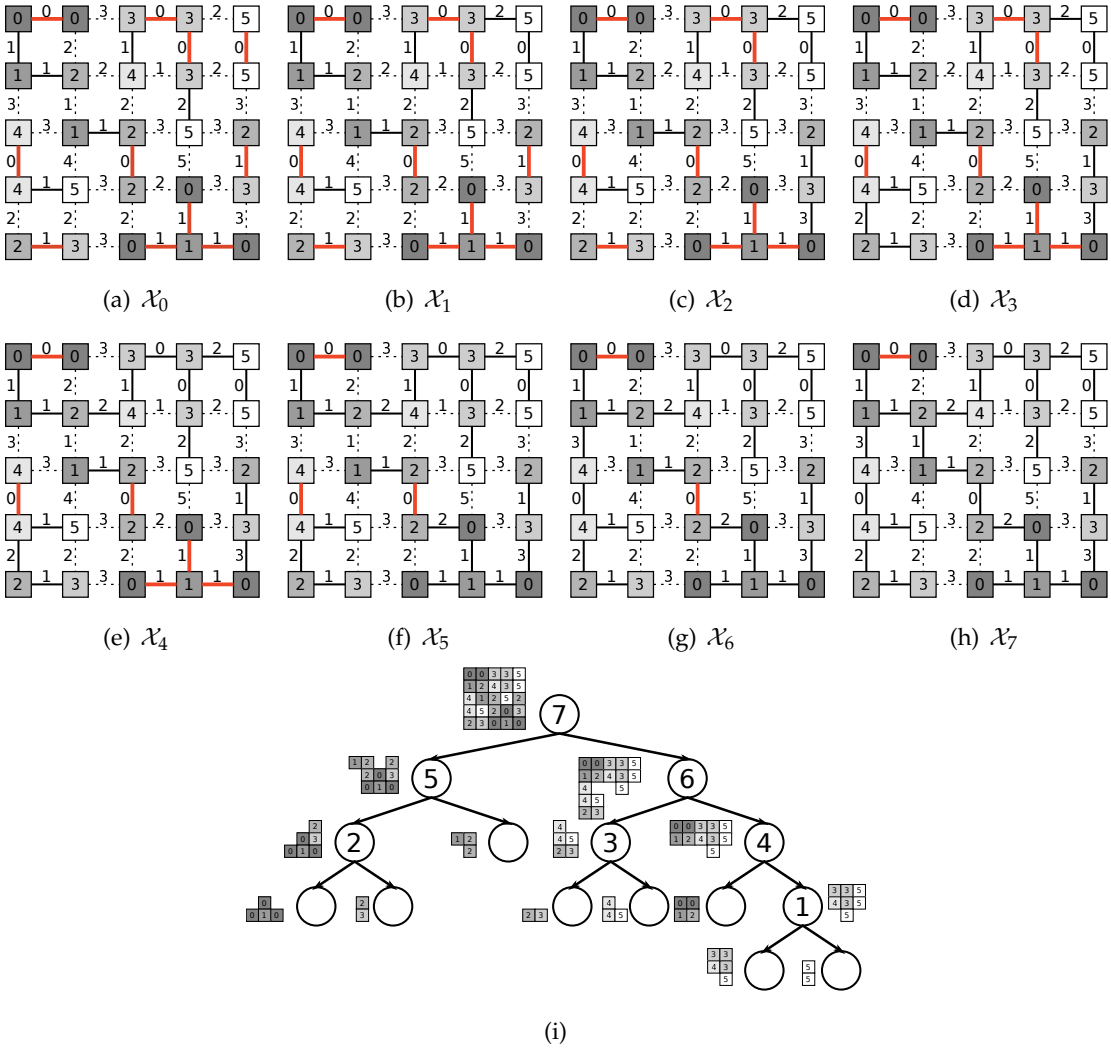
and

$$\text{dist}\,(\mathcal{R}_i, \mathcal{R}_j) = \min\{F_{(e_{p,q})} | p \in \mathcal{R}_i, q \in \mathcal{R}_j\}.$$

The smallest dissimilarity between two regions is the one containing the least significant minima (i.e. the smallest k). In case of multiple such dissimilarities with the least significant minima, the decision is based on the smallest distance.

Similarly to the BPT by Altitude Ordering, the MSF hierarchy is a self-dual structure since inverting the image does not change the edge-weighted graph associated to the image. They are also optimal in the context of preserving the minimum spanning tree of the underlying image graph [134]. An example of the MSF hierarchy for the image in Fig. 3.6 is displayed in Fig. 3.7.

The *Hierarchies of Minimum Spanning Forests* can be calculated based on BPT by Altitude Ordering for the same image. A transformation, linear in the number of image elements, is proposed in [132] to transform the BPT by Altitude Ordering into a Hierarchy of Minimum Spanning Forests, given that the order of the minima is provided. Thus, the total construction complexity for this hierarchy is the same as for the BPT by Altitude Ordering, $\mathrm{O}(N \times \alpha(N))$ for low quantized data.

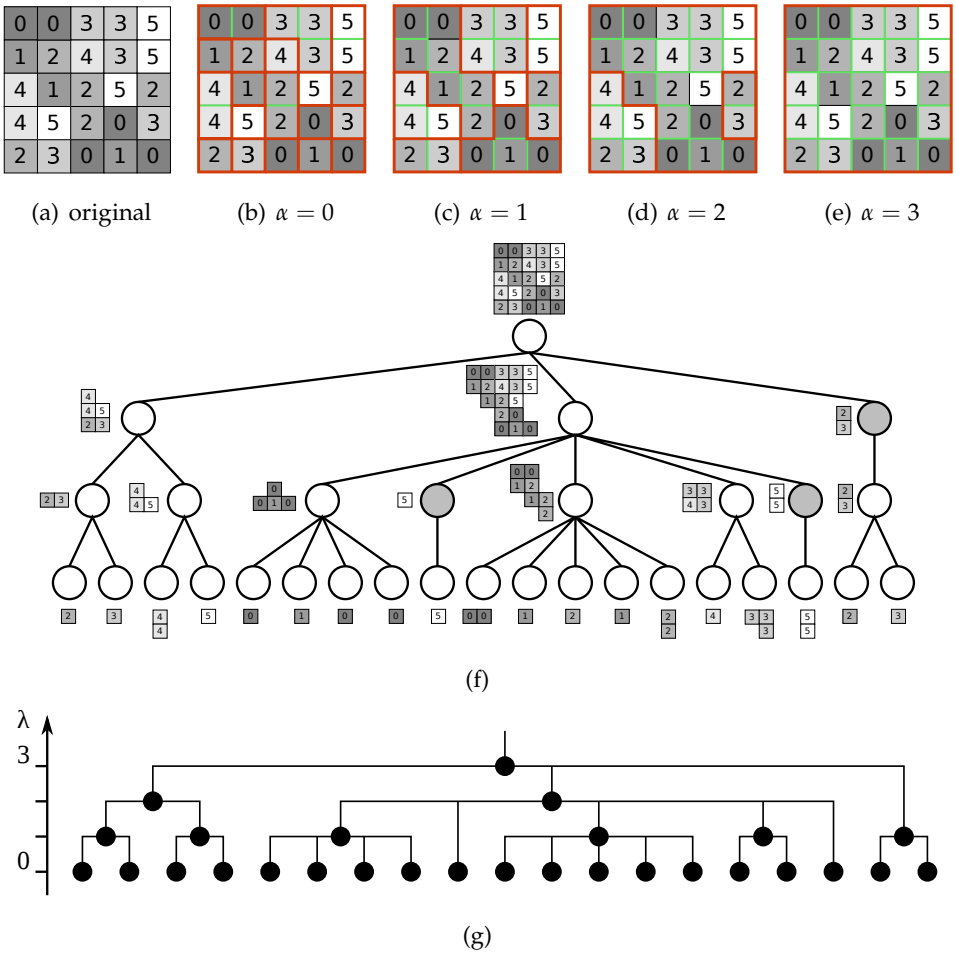


(a) $\mathcal{X}_0$ (b) $\mathcal{X}_1$ (c) $\mathcal{X}_2$ (d) $\mathcal{X}_3$

(e) $\mathcal{X}_4$ (f) $\mathcal{X}_5$ (g) $\mathcal{X}_6$ (h) $\mathcal{X}_7$

(i)

Figure 3.7: An example of MSF hierarchy based on the image displayed in Fig. 3.6(a) and 3.6(b). The stacks of image region seeds $\mathcal{X}_i$ for all the levels 0 through 7 of the MSF hierarchy are displayed in (a) – (h). All the image elements connected through a path of full edges belong to the same connected component for $\mathcal{X}_i$, while the bold red edges represent the minima present in step i (exactly one such minimum per connected component). The dashed edges are between the image elements not considered connected in $\mathcal{X}_i$. The final tree is displayed in (i) with the numbers in the nodes indicating the merging order.



(a) original (b) $\alpha = 0$ (c) $\alpha = 1$ (d) $\alpha = 2$ (e) $\alpha = 3$

(f)

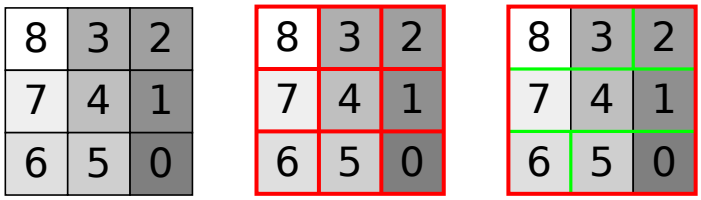
(g)

Figure 3.8: The original image is displayed in subfigure (a). Subfigures (b) through (e) show partitions of the original image for $\alpha = 0$ through 3, with edges between connected pixels shown in thin green lines and region borders in thick red. The α-connected components are shown in (f) (there is no merging in the gray nodes and they are not represented in the final tree). The dendrogram for this tree is shown in (g).

3.4 α-tree

The second tree from the class of partitioning trees is the α-tree. Unlike for the BPT, not only the merging order is strictly defined for the α-tree, but also the initial partition, region model and merging criterion. We examine here the α-tree for gray level images in detail, while the proposed adaptations for multichannel images will be discussed briefly in Subsec. 9.3, along with other open challenges.

The initial partition in the α-tree is always the partition to image flat zones. Using the notions introduced in Sec. 2.3 to describe the α-tree, the region model, merging criterion and



(a) original (b) $\alpha = 0$ (c) $\alpha = 1$

Figure 3.9: An example for the unwanted chaining effect. The original image is displayed in subfigure (a). The hierarchical decomposition of this image into α-connected components has only two different levels, displayed in subfigures (b) and (c). Although all the pixels are at different gray levels, the decomposition has no intermediate steps: all the pixels are separate components for $\alpha = 0$ and there is only one component for $\alpha = 1$.

merging order are defined as follows:

(1) The region model for each region $\mathcal{R}$ is the boundary of that region, $E_{\text{bound}}(\mathcal{R})$.

(2) The merging criterion defines the similarity between two neighboring regions as the lowest-valued edge common to models of both regions: $D(\mathcal{R}, \mathcal{R}') = \min\{F(u) | u \in E_{\text{bound}}(\mathcal{R}), u \in E_{\text{bound}}(\mathcal{R}')\}$, where edges are valued by gray level difference between neighboring pixels.

(3) The merging order dictates that in the i-th step, all the regions with the similarity equal to i should be merged (the initial partition is considered step 0).

The explanation with the region model, merging criterion and merging order is not the most common for α-trees, with most published works using the more natural definitions through α-connected components. All the leaves in the tree are considered to be at level 0 and represent regions for $\alpha = 0$, i.e. 0-connected regions, with the levels (and α) increasing towards the root of the tree. When defining α-connected regions for some α, all the neighboring pixels with gray level difference less than or equal to α become connected. Two pixels then belong to the same α-connected component if there is a path between them passing only through connected pixels. For increasing values of α, these regions form a SHoR, which can be represented by a tree. If we denote an α-connected component to which a pixel p belongs to by α-CC(p), the hierarchical relation between the regions can be expressed as:

$$\alpha\text{-CC}(p) \subseteq \alpha'\text{-CC}(p) \quad \forall\, \alpha \leq \alpha' \tag{3.7}$$

The α-connected components for an image in Fig. 3.8(a) are shown in Figs. 3.8(b)-3.8(e), with the α-tree shown in Fig. 3.8(f). When indexing an α-tree, the level of a node is sim-

ply the α value of the α-connected component represented by that node. The indexed tree corresponding to Fig. 3.8(f) is depicted in Fig. 3.8(g).

In this tree, the α corresponding to each level is indicative of the coarseness of the regions on that level. However, since the similarity measure used here is very local (it considers only neighboring pixels), the gray level variations within a single region can be higher than expected and constructed regions can be more complex than expected. An example of very different coarseness of regions on the same level can be seen on Fig. 3.8(f), where the level for $\alpha = 2$ contains both a region with 18 and 2 pixels. In the region of size 18, there is both a pixel with gray level value 0 and 5 even though the local range constraint α equals 2. This behavior is referred to as the *chaining effect* and the extreme case is shown in Fig. 3.9. Different approaches have been proposed as a solution to this problem [173, 178, 174, 177, 135]. One of the first surveys dealing with more than one approach to hierarchical image partitioning was done by Soille [173] and proposes a solution to the problem with the α-hierarchy related to the chaining effect, based on the constrained connectivity paradigm. All the proposed methods provide means to generate nested partitions in a unique manner while using a limited number of input parameters. The proposed methods include limiting the global range of the component, introducing a strong connectivity constraint, relying on a connectivity index of a component and the combination of those. Although the paper lists many potential modifications to α-connectivity, presently only the global range constraints are widely used, forming a hierarchy known as (ω)-tree (presented in Sec. 3.5).

This tree is self-dual, as the gray level differences between neighboring pixels will not change in the dual image $-I$. As the initial partition is always the flat zones of the image, the complete image is always contained in the representation.

The idea of simplifying the images by assigning a constant gray level to each pixel of a fine partition, where the regions in a partition are formed by respecting a local range parameter between pixels, was first introduced in [128]. The term α-connected component was first used much later, by Soille in [176, 173] where the hierarchical properties of such regions were also asserted.

α-tree construction. It was established by Najman and Soille [135] and Najman [130] that the α-tree is equivalent to a Min-tree defined on the edges (valued as intensity differences between the elements they connect), and can be calculated by any Max-/Min-tree construction algorithm (e.g. [131].) The papers [135, 130], however, did not explicate an algorithm for α-tree computation.

An indirect construction approach is not necessary, as proven by the work of Havel et al. [77], where an efficient implementation of the α-tree construction algorithm well suited for multithreaded construction was put forward. The implementation from [77] is still based on the Min-tree construction idea, but is realized so that the α-tree is directly built using a

modification of Tarjan's union-find [185].

Another algorithm, inspired by Kruskal's Minimum Spanning Tree algorithm [91] and using Tarjan's union-find [185], was proposed by Najman, Cousty, and Perret [132]. The proposed algorithm constructs a BPT by Altitude Ordering, from which the α-tree can be obtained with a linear post-processing step [132, 51]. To conclude, the complexity of current α-tree construction algorithms is the same as for Max-trees and Min-trees, and is quasi-linear in the number of image pixels for low quantization, $\mathrm{O}(N \times \alpha(N))$.

3.5 (ω)-tree

The (ω)-tree is another kind of partitioning tree, inspired by the need to solve the problems with chaining effect present with α-trees. If the parameter α is viewed as a parameter restricting the maximal range between locally connected pixels, the parameter ω restricts the maximal global gray level range inside a connected component.

The global gray level range of a component is denoted by $\mathrm{GR}(\cdot)$ and is defined as the difference between the value of the pixels with the highest and lowest gray level value belonging to the component. The notion of (α,ω)-connected components was first introduced together with the notion of α-connected components by Soille [176]. The (α,ω)-connected component of a pixel p is denoted by (α,ω)-$\mathrm{CC}(p)$ and is defined as the α'-$\mathrm{CC}(p)$ with the maximal possible $\alpha' \leq \alpha$ such that the global range is still lower or equal to ω:

$$\begin{aligned}(\alpha,\omega)\text{-CC}(p) &= \alpha'\text{-CC}(p), \\ \text{where } \alpha' &= \max\{\alpha'' \mid \\ &\qquad \alpha'' \leq \alpha \text{ and } \mathrm{GR}(\alpha''\text{-CC}(p)) \leq \omega\}.\end{aligned} \tag{3.8}$$

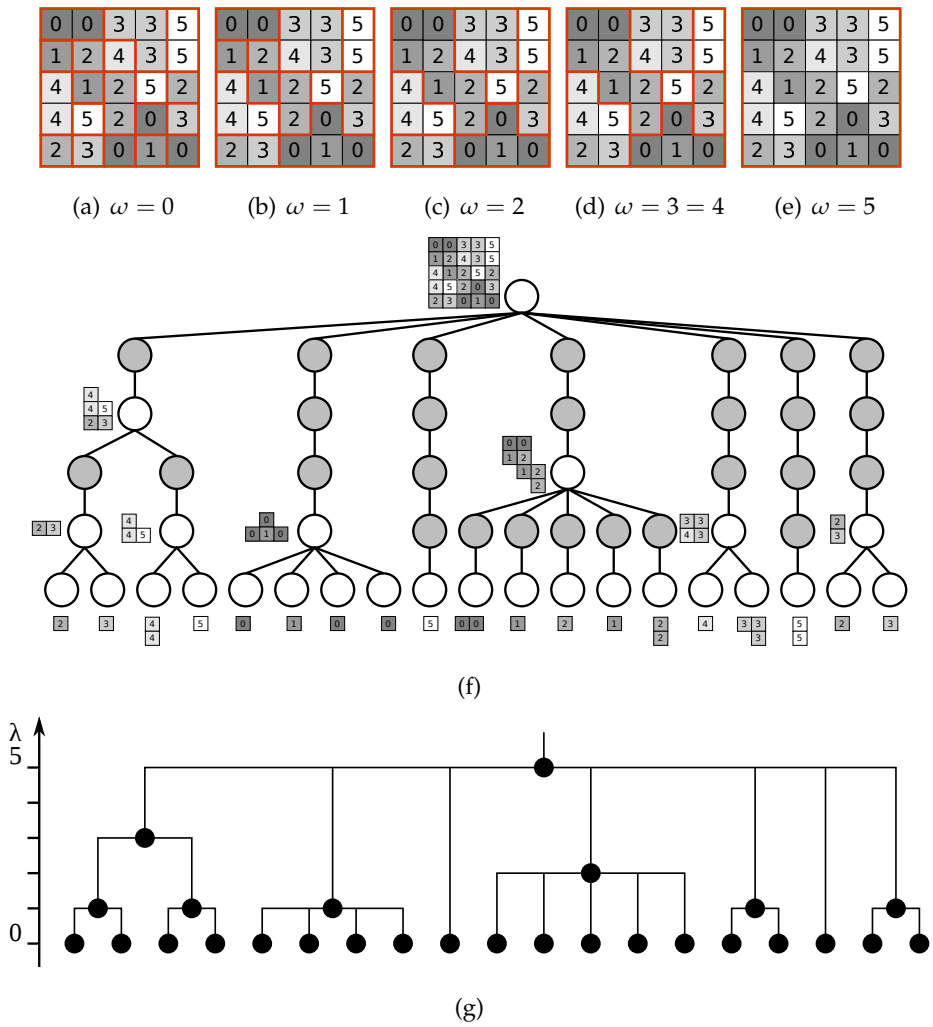
Even though the following relation holds for every pixel p:

$$(\alpha,\omega)\text{-CC}(p) \subseteq (\alpha',\omega')\text{-CC}(p) \quad \forall \alpha \leq \alpha' \text{ and } \omega \leq \omega', \tag{3.9}$$

the set of all (α,ω)-connected components can not form a SHoR since the order between (α,ω)-$\mathrm{CC}(p)$ and (α',ω')-$\mathrm{CC}(p)$ can not be determined for $\alpha \geq \alpha'$ and $\omega \leq \omega'$.

In [173] it was assessed that the (α,ω)-connected components for $\alpha \geq \omega$ are equivalent to those obtained for $\alpha = \omega$, i.e. the local range parameter α does not play a role for $\alpha \geq \omega$. Thus, we can define the (ω)-connected component of a pixel p as the largest α-$\mathrm{CC}(p)$ with the global range still less than or equal to ω:

$$\begin{aligned}(\omega)\text{-CC}(p) =&(\alpha \geq \omega,\omega)\text{-CC}(p) = \\ &\max\{\alpha'\text{-CC}(p) \mid \mathrm{GR}(\alpha'\text{-CC}(p)) \leq \omega\}.\end{aligned} \tag{3.10}$$



(a) $\omega = 0$ (b) $\omega = 1$ (c) $\omega = 2$ (d) $\omega = 3 = 4$ (e) $\omega = 5$

(f)

(g)

Figure 3.10: The partitions by levels of the (ω)-tree, for the original image from Fig. 3.8(a). Partitions for $\omega = 0$ through 3 are shown in subfigures (a) through (d). The partition for level $\omega = 4$ is equal to the partition for $\omega = 3$ and it is not separately displayed. The final partition for $\omega = 5$ encloses the whole image and is shown in (e). The tree is displayed in (f), with the duplicate nodes displayed in gray. The indexed tree (omitting the duplicate nodes) is displayed in (g).

Unlike the components defined by Eq. (3.8), the (ω)-CC define a SHoR, called the (ω)-tree. An example for such a hierarchy for the original image displayed in Fig. 3.8(a) is depicted in Fig. 3.10. As the (ω)-tree is a partitioning tree, it could be defined through the region model, merging criterion and merging order, but that would lead to increasingly complex definitions. Instead, only the usual definitions through the α-connected components of maximal size are provided.

Indexing the (ω)-tree is similar to indexing the α-tree: a node is assigned a level

corresponding to the ω value of the (ω)-connected component represented by the node (cf. Fig. 3.10(g)). Levels assigned to the indexed (ω)-tree are a better indication of region complexity than the levels of the α-tree. The problems with the chaining effect are not completely solved however, since the only two partitions present in the (ω)-decomposition of the image from Fig. 3.9 are the same as for the α-tree, except that the only region shown in Fig. 3.9(c) would belong to the level 8 of the (ω)-tree, while levels 0 through 7 would contain the same regions and represent the partition shown in Fig. 3.9(b).

Since the (ω)-tree is based on the α-tree, the characteristics of the two trees of the same image are related. When compared to the α-tree, the (ω)-tree of the same image always has more levels, and the number of regions decreases in general more slowly from the leaf levels towards the root (compare the dendrograms on Figs. 3.8(g) and 3.10(g)). However, the number of *different* regions represented by the nodes of the (ω)-tree is always lower or equal to the number of different regions represented in the α-tree (compare the number of white nodes on Figs. 3.8(f) and 3.10(f) or the number of nodes in the dendrograms on Figs. 3.8(g) and 3.10(g)), since all the (ω)-connected components are chosen from the already constructed α-connected components. Since it is based on the self-dual α-tree, the (ω)-tree is also a self-dual structure.

(ω)-tree construction. The (ω)-tree, and other constrained connectivity hierarchies, are based on the α-tree. Thus, the first step in calculating the (ω)-tree is the α-tree calculation, with $\mathrm{O}(N \times \alpha(N))$ time complexity (for $q \leq 12$ bit). One approach to transform the α-tree into an (ω)-tree, based on ultrametric watersheds, is presented by Najman [130]. Ultrametric watersheds are a framework suitable for visualizing the partitioning hierarchies, since they can be interpreted as images. The transformation is based on Lowest Common Ancestor (LCA) calculation [20] for every pair of neighboring elements in the ultrametric watershed representation of the (ω)-tree. The LCA calculation [20] is done in constant time for every neighboring pair, making the whole transformation linear.

The authors also propose an alternate approach to (ω)-tree construction [30] which will be discussed in detail in Chap. 8. In the context of the proposed method, the (ω)-tree can be interpreted as the α-tree where the global component range is used as a complexity measure for the regions of the tree. It performes a filtering of the α-tree, realized as a bottom-up traversal of the tree. Some nodes are removed and a new level (of aggregation) is assigned to the remaining nodes by the transformation, producing an indexed (ω)-tree hierarchy. This transformation is linear in the number of nodes of the tree, and, as shown in [30], also linear in the number of image elements. Finally, using any of the proposed approaches, the (ω)-tree can be constructed in the same complexity as the α-tree, $\mathrm{O}(N \times \alpha(N))$ for low quantized image data.

3.6 Comparative summary

For each tree, the characteristics such as duality, completeness and types of regions held in the representation were determined. While the complete representations should be better suited for searching various regions provided by the representation and as a basis for filtering, the representations which can not be used to reconstruct the original image might be better suited for examining the properties of the input image. Additional parameters needed to fully define the representations and the duality of the representations both contribute to defining the type of regions held by different trees, which is either stated explicitly or implicitly in the tree definition. The region types of different trees were closely examined as they have a direct influence on the type of objects and scenes each representation is well suited for. A list of various application domains where different hierarchies were sucessfully used is given in Tab. 3.2.

In addition to the characteristics of the constructed tree, and its suitability according to its content, another important factor when choosing a representation is the construction complexity. The most efficient construction algorithms are also presented for each tree, giving an insight into the possible implementation and additionally exposing implicit links between many of the hierarchies.

All these characteristics as well as the construction complexity are summarized for inclusion trees in Tab. 3.3 and partitioning trees in Tabs. 3.4 and 3.5. All the construction complexities presented in the Tabs. 3.3–3.5 are valid for images with low quantization (pixel values encoded as 12 bit integer values), in relation to the number of image pixels N and the number of possible pixel values k (construction complexity for different quantization and data types was examined more closely as the construction algorithms were presented for each tree type). Additionally, to provide a practical example of using these trees in image processing, the results of a simple filtering using all the main presented tree types is shown in Fig. 3.11.

Chapter Summary

This chapter presented and examined different inclusion and partitioning hierarchies: Max and Min-trees (cf. Sec. 3.1), Trees of Shapes (cf. Sec. 3.2), Binary Partition Trees (cf. Sec. 3.3), α-trees (cf. Sec. 3.4) and (ω)-trees (cf. Sec. 3.5), as well as selected special cases of those hierarchies. For each tree, their properties and type of regions represented by them was examined as well, and an indexing approach was proposed for each tree according to the tree construction algorithm. The chapter concludes with a comparative summary of the presented trees, offering additionally examples of different application domains as well as

Table 3.2: Some applications of the presented trees from the literature.

tree	applications
	inclusion
Min and Max-tree	• filtering [88, 131, 209, 161] • image [88] and video [159] segmentation • image compression [191], object detection [194, 186] • astronomical imaging [23, 186, 148] • feature extraction [136] and description [32] • image retrieval [136, 190, 32] and classification [195]
Tree of Shapes	• filtering [210, 120, 217, 42] and simplification [42] • segmentation [182, 39, 79, 216, 42] • image comparison [120] and registration [119] • image compression [180, 79] • edge detection [59] • feature extraction [43, 218, 28]
	partitioning
Binary Partition Tree	• filtering [161], segmentation [113, 158, 197, 2] • object detection [202, 22] and recognition [22] • hyperspectral imaging [197, 22]
α-tree	• simplification [173, 178] and filtering [178] • image segmentation [173, 135, 112] • video segmentation [112] • image classification [95] • hyperspectral imaging [95, 111]
(ω)-tree	• simplification [173, 10], segmentation [173, 135, 10] • image retrieval, remote sensing [7] • classification [8], hyperspectral imaging [10, 8]

results of image filtering using each presented hierarchy.

Hereafter, we present different techniques using the presented component trees and apply them to image retrieval problems. We chose to focus on the Min and Max-trees and the ToS from the domain of inclusion trees, and the α-tree and the (ω)-tree for the partitioning trees. The reasons for this selection are two-fold: first, the flexibility of the BPT comes at

Table 3.3: Summary of characteristics for inclusion trees.

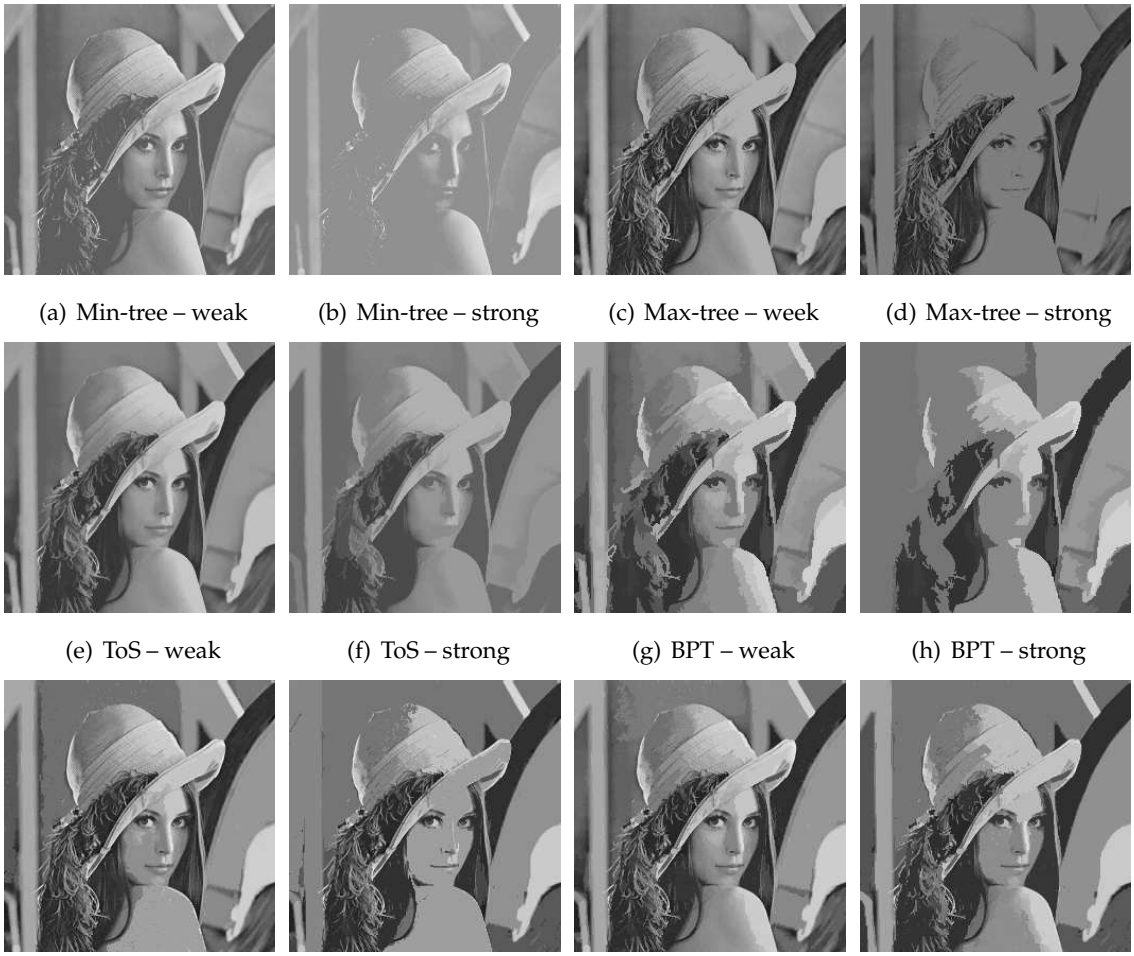
Tree	**Max tree**	**Min tree**	**Tree of shapes**	**Topological tree of shapes**
Dual tree	Min tree	Max tree	*self-dual*	*self-dual*
Type of objects	dark objects	bright objects	shapes	rising and falling slopes
Complete representation?	Yes	Yes	Yes	No
Construction complexity	$\mathrm{O}(N \times \alpha(N))$	$\mathrm{O}(N \times \alpha(N))$	$\mathrm{O}(kN)$	$\mathrm{O}(kN)$
Additional parameters	No	No	No	No

an expence of increased construction complexity and less efficient calculation. Second, all the selected trees can use the Max-tree construction algorithm implementation (Min-tree on the inverted image, ToS on the reordered pixels and α-tree on edges with a filtering step to produce the (ω)-tree), unlike the BPT which would require a separate implementation. The following two chapters will apply the component trees to the problem of feature detection, with Chap. 4 presenting the theoretical description of the proposed feature detector and Chap. 5 examining the performance of the detector in the context of image matching and retrieval.

Table 3.4: Summary of characteristics for BPT and its special cases.

Tree	**Binary partition tree**	**Hierarchy of MSF**	**BPT by altitude ordering**
Dual tree	*self-dual*[a]	*self-dual*	*self-dual*
Type of objects	unions of initial partition	watershed cuts	regions sequence reflecting the strict total ordering on the edges
Complete representation?	Yes[b]	No	Yes
Construction complexity	$O(N^2 \log N)$	$O(N \times \alpha(N))$	$O(N \times \alpha(N))$
Additional parameters	Initial partition, region model, similarity measure	Sequence of minima	Strict total ordering of the edges

[a] depends on region model and similarity measure, but usually desirable
[b] depends on the initial partition



(a) Min-tree – weak (b) Min-tree – strong (c) Max-tree – week (d) Max-tree – strong

(e) ToS – weak (f) ToS – strong (g) BPT – weak (h) BPT – strong

(i) α-tree – weak (j) α-tree – strong (k) (ω)-tree – weak (l) (ω)-tree – strong

Figure 3.11: Filtering the standard 256×256 grayscale *Lena* image using different trees. For all the trees except the BPT, the filtering was done by choosing a threshold level and keeping only the nodes above that level. The images for the BPT were generated by the tool presented in [2], which uses a different indexing method and filtering strategy. For every tree, a weaker and stronger filtering was performed. Weak and strong filtering of the Min-tree is shown in (a) and (b) respectively, and for the Max-tree in (c) and (d). Filtering using the ToS correspond to (e) (weak) and (f) (strong). The images resulting from BPT are shown in (g) (weak filtering) and (h) (strong filtering). The α-tree weak filtering results are displayed in (i), and strong in (j). Finally, filtering using the (ω)-tree is depicted in (k) and (l) for weak and strong filtering.

Table 3.5: Summary of characteristics for the α-tree and the (ω)-tree.

Tree	α**-tree**	(ω)**-tree**
Dual tree	*self-dual*	*self-dual*
Type of objects	α-CC (quasi flat zones)	(ω)-CC
Complete representation?	Yes	Yes
Construction complexity	$O(N \times \alpha(N))$	$O(N \times \alpha(N))$
Additional parameters	No	No

Chapter 4

Component Tree based Maximally Stable Regions

Contents

Detection of local features is the base step in many computer vision applications, providing a compact representation of the image by only considering the selected salient points. A good feature detector will provide features which are distinctive, invariant and discriminative.

Based on the hierarchical ordering of the MSER detections shown in [63], an algorithm using the Min and Max-tree hierarchies [159, 88] to determine MSER regions was introduced by Nistér and Stewénius [136]. Extending the idea put forward in [43], the algorithm [136] can be applied to any component tree exhibiting invariant properties. In this chapter, we study the detectors based on three different hierarchies, using the Tree of Shapes from the class of partitioning trees as well as α-tree and (ω)-tree from the class of inclusion trees.

The performance of the proposed detectors is evaluated using the image matching framework of Mikolajczyk et al. [116]. The show that the proposed Tree of Shapes based Maximally Stable Regions (ToS-MSR) detector achieves a small but consistent increase in the number of responses, thus mitigating the drawback of the MSER detector due to a sometimes small

number of features while still remaining suited for image retrieval applications. As such, the ToS-MSR detector is additionally evaluated in a retrieval setup using VLAD [86] indexing achieving an improvement over using the original MSER features.

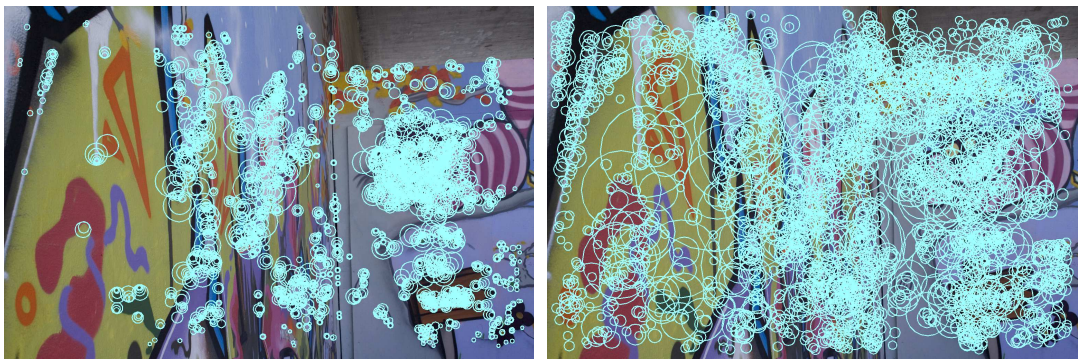
An introduction to region detection is given in the next section. Following, in Sec. 4.2 we recall the MSER regions, and specifically their detection on a Min and a Max-Tree. Finally, the Section 4.3 motivates and explains substituting the hierarchy used in the tree-based MSER detection algorithm and the three constructed detectors.

4.1 Salient Regions Detection

The development of affine invariant detectors was driven by their robustness against viewpoint change as one of the most common scene transformations between images. Many different detectors were developed; detectors such as DoG (introduced for the SIFT descriptors) [99], SURF [19], Hessian and Harris-Affine [115] as well as KAZE [4] and AKAZE [5] operate in scale space to achieve multiscale image processing. A recent MFD detector [14] is also based on image gradient, but without explicit scale space construction. Others, like BPLR [90], FOCI [222] and WαSH [198] rely on edges and boundaries, while MSER [108] detects features on multiple scales based on image contrast and region intensity. These detectors are often complementary (and can be used in combination), providing features responding to corners, ridges or blobs (contrasted regions).

Due to the large variety of the type and structure of the detections returned by different feature detectors, which can be either points or regions of pre-defined or arbitrary shapes, the *detected regions* are often replaced by *measurement regions* [108]. The *detected region (DR)* refers to the set of pixels that have effectively contributed to the affine detector response. Given a detected affine-covariant region or its scaled version, a *measurement region (MR)* of arbitrary size may be associated with each DR if the construction is affine-covariant. The construction of measurement regions can then for example rely on scaling, taking the convex hull or fitting an ellipse over the region based on its second order moments [116]. These measurement regions are then used for the remainder of the task being performed instead of the detected regions, usually for selecting the parts of the image for computing the invariants. Smaller measurement regions are more likely to satisfy the planarity condition and to not cross a discontinuity in depth or orientation. Despite such small regions being less likely to be unique and thus being less discriminative, the increase in the region size is limited because larger regions will likely include parts of the background which should not be considered. An analysis of the effect of the region size on detector performance in matching experiments was further analyzed in [116]. Output examples for several detection methods in terms of measurement regions, calculated by approximating the region with an ellipse with same

(a) DoG (b) Hessian Affine



(c) AKAZE (d) FOCI

(e) WαSH (f) MSER

Figure 4.1: Example of measurement regions for different detectors. The measurement regions are fitted based on second order region moments. The input image is taken from the dataset used in [117][1]. The author would like to especially thank Dmytro Mishkin for providing the output of multiple detection methods.

shape moments up to the second moments where applicable, are shown in Fig. 4.1.

4.2 Maximally Stable Extremal Regions

We focus here on the Maximally Stable Extremal Regions (MSER), a fast detector based on image intensity, responding to blobs of high contrast and producing affine invariant, highly featured regions of arbitrary shapes. Performance benchmarking done both by Mikolajczyk et al. [116] as well as Fraundorfer and Bischof [70] has identified the MSER detector as one of the best local region detectors due to its robustness against viewpoint, rotation, scale and lighting changes. As such, it has been used in applications ranging from object recognition [139], image retrieval [137], recognition and matching [68], tracking [63], to recent use in text detection [46, 80]. Extensions for color [68] and for better robustness against blur [69] were also proposed in the literature. Due to excellent performance and prevalent use of the MSER detector, as well as the fact that it can be constructed using the Min-tree and Max-tree hierarchies [136] prompted the examination of MSER-like detectors.

The MSER detector was first introduced by Matas et al. [108], returning regions only based on their intensity. Informally, the output of the MSER detector corresponds to connected regions which are present and stable over multiple consecutive thresholdings of a gray level image I. They are established to have the following desirable properties:

- *Invariance* to affine transformation of image intensities.
- *Covariance to continuous transformations* on the image domain.
- *Stability*, since only the regions with a stable support are selected.
- *Multi-scale* detection, because the detection process does not require smoothing. This allows for the detection of both fine and large structures.

The salient MSER regions are selected amongs the *extremal regions* of an image I, defined by the extremal property of the image intensity function f on the region outer boundary. For a *minimal extremal region* $\mathcal{R}$, the intensity $f(p)$ of any region pixel p is smaller than that of any pixel q belonging to the outer region boundary $V_{\text{outbound}}(\mathcal{R})$:

$$f(p) < f(q) \mid \forall p \in \mathcal{R}, q \in V_{\text{outbound}}(\mathcal{R}) \tag{4.1}$$

We will denote such a region according to the maximal intensity level k of all the elements of the region (such pixels will always lie among the pixels belonging to the inner region boundary, $p \in V_{\text{inbound}}(\mathcal{R})$). $\mathcal{R}_k$ is a minimal extremal region with the maximal intensity level k. Similarly, $\mathcal{R}^k$ denotes a maximal extremal region of minimal intensity level k among the region elements. Minimal extremal regions are nested for increasing k, i.e. $\mathcal{R}_k \subseteq \mathcal{R}_l$ for any $k < l$, and similar relation holds true for maximal extremal regions [63].

[1]The dataset from which the image is taken is publicly available at http://cmp.felk.cvut.cz/wbs/index.html.

The seminal algorithm proposed by Matas et al. [108] relies on a union-find implementation [185, 164] to keep track of the nested region sequences of connected components. It tracks each sequence of connected components (i.e. the development of a single connected component over a series of thresholds), where a merge of two components at a certain gray level is viewed as a termination of the existence of the smaller component sequence. For all such sequences, the stability function $q(\cdot)$ is calculated for regions at each different gray level k, and the regions corresponding to the local minima of this function are selected by the detector. This function measures the rate of growth of a region with the change of intensity, and is originally defined as:

$$q(\mathcal{R}_k) = \frac{|\mathcal{R}_{k+\Delta} \setminus \mathcal{R}_{k-\Delta}|}{|\mathcal{R}_k|}. \tag{4.2}$$

where $|\cdot|$ denotes cardinality. The parameter Δ is the parameter of the method. A larger Δ parameter requires the region to be stable through a greater range of gray levels. The region $\mathcal{R}_{k+\Delta}$ is determined from the sequence of nested regions to be the largest region such that $\mathcal{R} \subset \mathcal{R}_{k+\Delta}$ and:

$$d(\mathcal{R}_k, \mathcal{R}_{k+\Delta}) \leq \Delta. \tag{4.3}$$

The distance between any two regions $\mathcal{R}_k$ and $\mathcal{R}_l$ of a nested sequence is here simply defined as a difference between the region gray levels, $d(\mathcal{R}_k, \mathcal{R}_l) = |l - k|$. The fact that the algorithm works only with single-thread sequences of regions and does not keep track of the merge events (i.e. one of the region sequences is terminated in the case of a region merge) allows it to similarly calculate the region $\mathcal{R}_{k-\Delta} \subset \mathcal{R}$ as the smallest region in a sequence satisfying:

$$d(\mathcal{R}_{k-\Delta}, \mathcal{R}_k) \leq \Delta. \tag{4.4}$$

In order to resolve the ambiguity causing the function $q(\cdot)$ to be undefined close to the merge points between the regions, and to speed up the computation of the MSER regions, implementations in popular computer vision libraries (e.g. VLFeat [199], OpenCV [36]), as well as our implementation, use a simplified version of the stability function:

$$q'(\mathcal{R}_k) = \frac{|\mathcal{R}_{k+\Delta} \setminus \mathcal{R}_k|}{|\mathcal{R}_k|}. \tag{4.5}$$

Output very close to the original one can be achieved by simply using twice as big Δ values when using Eq. (4.5) instead of Eq. (4.2). An example illustrating the difference in algorithm output depending on the Δ parameter is shown in Fig. 4.2.

Additional parameters are also included to better control the region selection process. The simplest selection criterion is the size of the regions: all the regions with the size below *minSize* or above *maxSize* will be rejected. Further, the parameter *minDiversity* exists to prune regions that are too similar (e.g. differ only in a few pixels). The diversity for a maximally

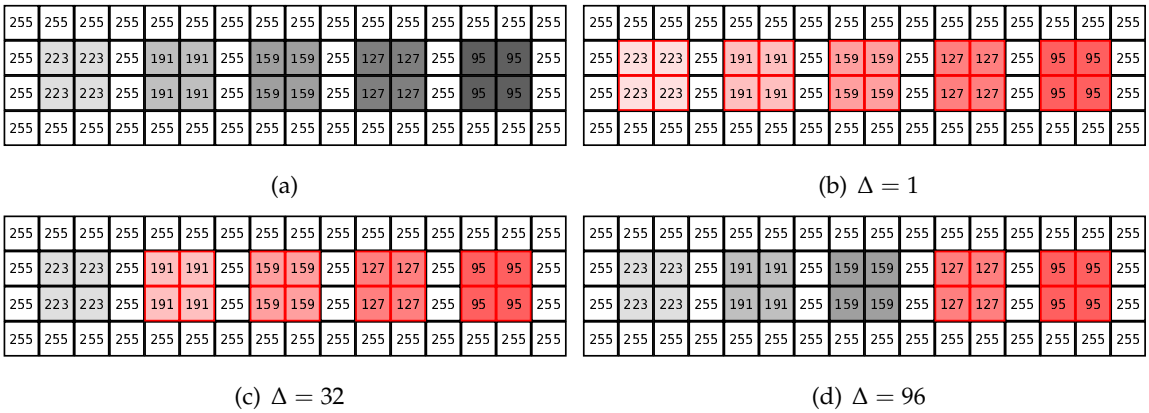


Figure 4.2: Effect of the Δ parameter in the MSER detector. The example image used for detection is shown in (a), with the output of the MSER detection for different values of Δ displayed in (b)–(d). Fewer and fewer regions get detected with the increase in Δ parameter.

stable regions $\mathcal{R}_{k_1}$ is calculated by:

$$diversity(\mathcal{R}_{k_1}) = \frac{|\mathcal{R}_{k_2} \backslash \mathcal{R}_{k_1}|}{|\mathcal{R}_{k_1}|}, \tag{4.6}$$

where $\mathcal{R}_{k_2}$ is a parent MSR region of $\mathcal{R}_{k_1}$ such that that $k_2 > k_1$, $\mathcal{R}_{k_2}$ is maximally stable and there is no $k_2 > k_x > k_2$ such that $\mathcal{R}_{k_x}$ is a maximally stable region. This effectively calculates the size difference between an MSR region and the first bigger MSR region detected from the same nested sequence. If this $diversity(\cdot)$ is smaller than *minDiversity*, the region $\mathcal{R}_{k_1}$ is removed from the list of resulting regions, effectively restricting the detections to only sufficiently different regions of a sequence. The final selection is done based on examining the actual values of the used stability function of the region for the selected Δ rather than the the local minima of the function along the branches, and the region is rejected if $q(\cdot)$ from Eq. (4.5) is larger than *maxVariation*. The list of these parameters as well as their effect on the number of detections can be found in Table 4.1.

The traditional way of calculating the MSER, based on union-find, can be viewed as the same flooding simulation used for computing watershed segmentation [204]. In immersion analogies, the gray-level profile of the image is treated as a landscape height-map. The union-find based immersion analogy supposes that the landscape is porous, or that holes have been pierced in all the local minima allowing the flooding water to reach the same level ("height") everywhere in the image at any moment during the flooding. All the components, corresponding to different basins of water (catchment basins) in the flooding analogy, are discovered and treated at the same time. The merge between two components corresponds to the water level rising sufficiently for the two disconnected basins to merge into a single flooded basin.

Table 4.1: MSER parameters and their effect on the number of detections.

Parameter	Effect of increase on the number of detections
Δ	decrease
minSize	decrease
maxSize	increase
maxVariation	increase
minDiversity	decrease

According to the definition of the inner nodes of the Min and Max-tree (cf. Sec. 3.1), their nodes correspond to extremal regions, which are candidate regions for MSER. In fact, the regions of the Min-tree at the gray level k correspond to minimal extremal regions $\mathcal{R}_k$, and similarly for the Max-tree. For this reason, a new algorithm for MSER detection relying on Min and Max-tree was proposed by Nistér and Stewénius [136]. This algorithm adapts the bottom-up approach, relying on a hierarchical queue. Using an immersion analogy, the flooding originates from a single arbitrary point, at which the water is being poured on an opaque landscape. The flooding procedure first fills up the basin at which the water is being poured on and then proceeds to spill into neighboring basins. Basins correspond to the connected components in the tree, and their creation and merging corresponds to creation of nodes and merging of different branches in the hierarchy.

More generally, any Min and Max-tree construction algorithm computes the extremal regions as well as the stability function according to Eq. (4.5), typically simultaneously with the tree construction. Selecting the regions, as well as enforcing other restrictions from Table 4.1 is done by filtering, and the resulting selected regions retain the hierarchical structure in the filtered tree. Selecting regions from the Min-tree results in detecting the minimal MSERs, while the Max-tree has to be used for maximal MSER detection. The distance between two nodes in an ancestral relation, $\mathcal{R}_k$ and $\mathcal{R}_l$, with $k < l$, is again calculated as their gray level difference $d(\mathcal{R}_k, \mathcal{R}_l) = |l - k|$. This distance is then used to determine the corresponding region $\mathcal{R}_{k+\Delta}$ from Eq. (4.5), when the Min and Max-trees are used for MSER detection.

Despite not being commonly integrated and exploited in the state-of-the-art retrieval schemes, additional information the MSER detector provides could prove an added advantage of this detector in image retrieval. Firstly, the arbitrary shape of these regions allows constructing feature descriptors including shape information [32, 69] (as opposed to using only the prevalent SIFT [99] descriptor). Secondly, the MSER detector organizes its responses into (two) nested hierarchies [63]. This allows for the possibility of constructing an indexing

scheme utilizing the provided spatial relations between the salient regions of the image in addition to region descriptors, unlike the state-of-the-art approximate search schemes [170, 86, 96] which include no spatial information.

4.3 Maximally Stable Regions from Component Trees

While the MSER regions are based on strict intensity ordering of the pixels, Maximally Stable Regions could be detected in a similar manner using a different ordering and thus producing features with different stability properties [108]. A feature detector constructed by replacing the Max and Min-trees with a different component tree in the MSER construction algorithm effectively replaces the ordering of the pixel intensity before region detection.

Furthermore, the Min and Max-tree only model the dark and bright image structures respectively, while most other component trees are constructed to be self-dual (cf. Tabs. 3.3, 3.4 and 3.5). Due to this property, the bright and dark structures in the image are treated equivalently with the aim of better modeling non-homogeneous objects as well as certain textures. Additionally, if the tree is self-dual, only one tree is used to detect all the interest regions for an image, and all detected regions belong to a single hierarchy. The output of the detector is consequently also hierarchically organized (in one or more non-overlapping trees) and thus provides hierarchical spatial relations between all the regions, which could be exploited in following image processing steps.

Two general conditions must be met in order to make replacing the tree used for stable region detection viable. First, the construction complexity of such a tree must be low enough to ensure acceptable detector speed. While a previous attempt has been done to replace the Min and Max-trees with the Tree of Shapes [43], it was never deeply explored as there were no efficient state-of-the-art construction algorithms. However, as recently linear or near-linear construction algorithms were proposed for a large number of component trees (cf. Tabs. 3.3, 3.4 and 3.5), exchanging the tree used for region detection became possible.

Second, a distance to be used for Eq. (4.3) needs to be defined for a new tree to be used. It is possible to directly use the levels of the nodes (i.e. the ultrametric distance) assigned by the indexing proposed in Chap. 3, in which case the same criterion used in tree construction is used for determining the stability of a region. A new distance function can also be defined between the nodes of the tree, based on some other property used in tree construction or another attribute defined on the regions of the tree. The first attempt in substituting the tree with the Tree of Shapes [43] avoided this issue by always calculating the stability using the first parent region in the tree. An alternative approach was also proposed by Xu et al. [218], circumventing the use of the stability function and instead relying only on the tree topol-

ogy to make a choice of regions (i.e. only the nodes located at the point where two or more branches merge are considered). The method proposed by Xu et al. also achieves competitive repeatability scores in the Mikolajczyk et al. [116] matching framework, however the number of responses is greatly increased (up to a 6-fold increase compared to MSER). These properties make it well suited for the target applications of 3D reconstruction and image registration, however make it unfeasible to use in image retrieval application.

4.3.1 Maximally Stable Regions on Tree of Shapes

The Tree of Shapes encodes image composition in a similar way to the Min and Max-trees, representing it as a hierarchy of objects and shapes based on their contrast with their background. Due to its self-dual property, it handles both dark and bright objects simultaneously working both with local image minima and maxima. The *Tree of Shapes based Maximally Stable Regions (ToS-MSR)* detector was proposed [28], giving a more featured response than the original MSER detector, while still responding to similar types of regions.

In order to construct this MSER-like detector, we define a new distance between the regions of the ToS based on the pair-wise difference between neighboring node levels. During the ToS construction, a new node n will be composed of the pixels originating from its child nodes, as well as some extra pixels $S(n)$ (cf. Eq. 2.19). All the additional pixels $S(n)$ of such a node n will belong to the same gray level. Then, assuming $f(p) = k, \ \forall p \in S(n)$, we denote by $\mathcal{R}_k(n)$, or simply $\mathcal{R}_k$ a region introducing the pixels at gray level k to the connected component. The distance between any two regions $\mathcal{R}_k \subseteq \mathcal{R}_l$ amounts to the sum of consecutive distances of all the nested regions on a path $\mathcal{R}_k \subseteq \mathcal{R}_{k_0} \subseteq \cdots \subseteq \mathcal{R}_{k_x} \subseteq \mathcal{R}_l$ and is equal to:

$$d(\mathcal{R}_k, \mathcal{R}_l) = |k - k_0| + |k_0 - k_1| + \cdots + |k_x - l|. \tag{4.7}$$

We chose not to use the given ultrametric distance assigned with the indexing of the ToS, as the distance in Eq. (4.7) better reflects the region contrast with the background. In the tree shown in Fig. 4.3 (i.e. the tree previously displayed in Fig. 3.2), both of the single-pixel leaves at gray levels 1 and 4 would have the same distance and thus the same stability in relation to their parent node. However, as they have a different contrast in relation to the background represented by the parent node, we can benefit from using a distance like the one proposed in Eq. (4.7) which can make a distinction between these two situations. This is due to the fact that Eq. (4.7) in fact corresponds to examining the path dynamics along only one branch stemming from the ancestral node (i.e. the node higher up the hierarchy between $\mathcal{R}_k$ and $\mathcal{R}_l$). On the other hand, the ultrametic distance for the ToS in Eq. 3.2 calculated the region dyamic, which corresponds to examining the path dynamics between a node and all of its different child nodes.

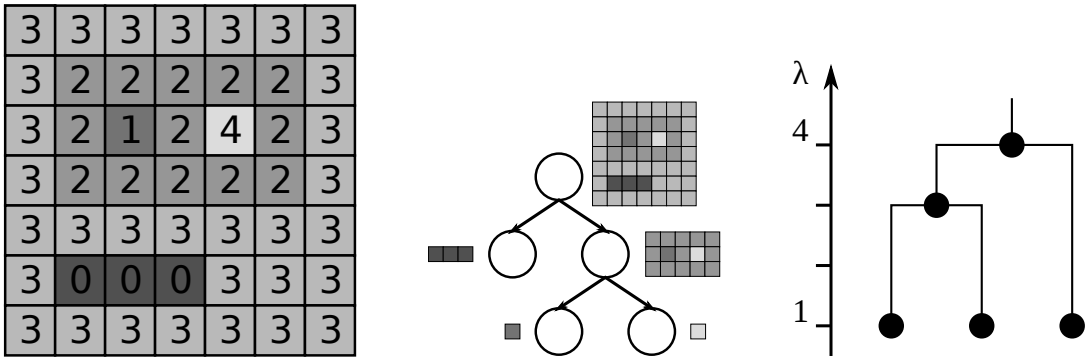
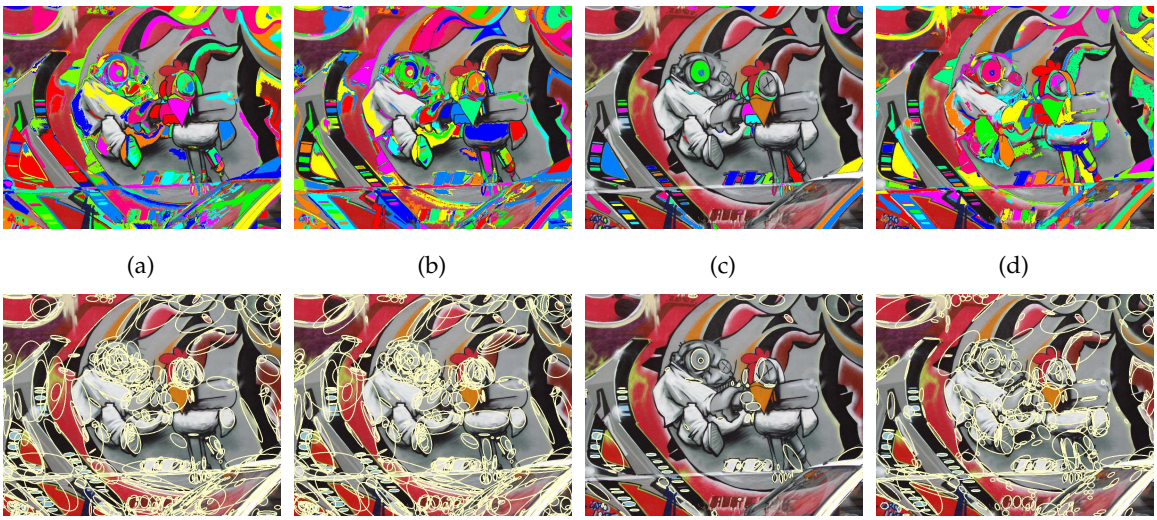


Figure 4.3: The Tree of Shapes of the original image displayed in 4.3(a) is shown in 4.3(b), with the dendrogram in 4.3(c). This figure is a summary of previously shown Figs. 3.1(a) and 3.2. It can be seen that, using the assigned ultrametric distance depicted by the dendrogram in 4.3(c), both of the leaf nodes, corresponding to pixels at gray levels 1 and 4 respectively, have an equal distance from their parent. This is not a desirable distance function to use in the MSR detector construction as the proposed distance is the same despite the components having different contrast from their background at gray level 2.

Using the state-of-the-art construction algorithm [73], we construct a feature detector running in near-linear complexity in the number of image pixels. It only uses the one, self-dual, tree to determine the salient regions and thus also provides spatial relations between all the regions as a single hierarchy. The regions detected by the tree-based MSER implementation and the ToS-MSR detector are displayed in Figs. 4.4(a),4.4(b), 4.4(e) and 4.4(f).

The regions are still of arbitrary shape, but no longer have holes as the shapes in the Tree of Shapes can handle objects containing both dark and bright parts. Better shape information could be exploited in region description, but it also benefits the results of applying an affine construction method to calculate measurement regions from the detected distinguished regions. As an example, fitting an ellipse based on up to second shape moments of the region will result in a better centralized ellipse region and cover a more discriminative patch in the picture as the input to description methods.

While a small number of responses has limited the use of MSER in applications requiring a higher number of matches (e.g. mosaicking, 3D modeling, registration), it is important to limit the number of responses for applications such as image retrieval, where the vocabulary size and consequently indexing and search speed will depend on the number of descriptors provided. While a low number of MSER responses can still be a drawback in retrieval applications, the proposed ToS-MSR detector produces 20%–40% more responses resulting in improved retrieval performance over MSER (the experimental results will be presented in



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

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

Figure 4.4: Detections for the four different tree-based detectors. In the top row, the exact detected regions are shown while in the bottom row the regions are approximated by ellipses. The output of tree-based MSER is shown in (a) and (e), the ToS-MSR in (b) and (f), the α-MSR in (c) and (g). Finally, the detections of the (ω)-MSR detector are shown in (d) and (h). All the outputs are calculated for the first image of the *'graffiti'* dataset from the Mikolajczyk et al. [116] framework.

the next chapter). This small but consistent increase in the number of detections can be seen as an advantage, as it does not have a severe effect on the retrieval speed, unlike using the more heavily featured detectors (such as e.g. Hessian-Affine [115]).

4.3.2 Maximally Stable Regions on α-tree and (ω)-tree

In addition to using the Tree of Shapes, which is an inclusion tree like the Min and Max-trees used in the original MSER, we also attempt to construct a detector using partitioning trees. The α and (ω)-trees are also both self-dual, producing responses organized in only one hierarchy. Moreover, the (ω)-tree is a filtered version of the α-tree, so using it for detection purposes can be seen as changing the stability function used with the α-tree. Both trees can also be efficiently constructed using algorithms of quasi-linear complexity [77, 132].

For both of these detectors, the ultrametric distance defined by the tree indexing is used in the calculation of the stability function according to Eq. (4.5) (cf. Sec. 3.4 for the α-tree and Sec. 3.5 for the (ω)-tree). This corresponds to looking for regions stable according to their local gray level difference, or range (between neighboring pixels) on the α-tree. The detector using the (ω)-tree responds to regions stable according to the global range of region pixel

values, and could also be implemented directly on an α-tree using the global region range difference as the region distance.

Unlike with the Tree of Shapes, the regions contained in the α and (ω)-trees can contain holes, but are still of arbitrary shape. The type of regions that these detectors respond to differs however from the regions returned by both the MSER detector and the ToS-MSR detector. While it is no longer important that the region be either bright or dark in respect to its background (possibly containing sub-regions of the opposite contrast in the case of ToS), they require a strong gradient along all the borders of the region. As such, they turn out to be much more selective (cf. Fig. 4.4 for comparison of all four tree-based detectors) than the considered inclusion tree-based detectors, thus responding to even less regions than MSER. Additionally, they are also very sensitive to scene type and benefit only from the scenes featuring strong edges and contrasted region borders (e.g. the *'graffiti'* dataset from the Mikolajczyk et al. [116] framework used for examples in Fig. 4.4).

Despite the (ω)-tree construction attribute being stricter (i.e. global range instead of local), it allows for a better fine tuning during the selection process of the detector. This is explained by the criterion also being more descriptive of the regions characteristics and better reflecting the complexity of the composite regions. Thus the global range is more pertinent to the stability of the region, allowing the stability function to return a wider range of values such that a small change in parameter values (and especially, the Δ parameter) will only cause a small change in the number of features in the detector output. This in turn allows for a finer tuning of the parameters as they are more discriminative regarding region quality, instead of discarding a large number of both good and bad regions with a small change in parameter values like in the case of the α-MSR detector. While the detector is presented and evaluated in the next section in the general matching framework [116], it would be interesting to apply the detectors to specific types of imagery containing strong edges, e.g. drawn images, caricatures or cartoons.

Chapter Summary

In this chapter, we extend the MSER [108] detection algorithm defined on Min and Max-trees [136] to the self-dual inclusion tree, the Tree of Shapes, as well as to two partitioning trees: the α-tree and the (ω)-tree. After explaining the MSER detection, we define the conditions needed to substitute the hierarchy used in the algorithm with a different component tree.

While the general difference between inclusion-tree and partitioning-tree based detectors remains a topic for future exploration, we offer an analysis based on the visual comparison between the regions detected using the detectors based on different component tree super-

classes. The following chapter will further examine those difference by evaluating the performance of the proposed detectors in image matching problems. Finally, the performance of the ToS-based detector, found to be the highest performing proposed detector in image matching, is evaluated in an image retrieval framework as well.

Chapter 5

Validation of Tree-MSR

Contents

In this chapter, we evaluate the different component tree based MSR feature detectors proposed in previous chapter, namely the ToS-MSR, α-MSR and (ω)-MSR detectors. We evaluate the detectors for two different applications that benefit from region detectors with a moderate number of responses, as well as compare the performance of the partitioning-tree based detectors with the proposed ToS-MSR and tree-MSER detector (which is also an inclusion-tree based detector).

The first experimental setup (also used for parameter tuning) evaluates the potential and actual performance of the detector when used for region matching. We use the benchmark framework proposed by Mikolajczyk et al. [116] which expresses the performance in terms of *repeatability* and *matching score*. The dataset provided with the framework is divided into 8 categories of images sequences. Each image sequence is used to test the robustness of the detector against different types of image and scene transformations by comparing the results of performed matching with the projections obtained using the homographies provided as the ground truth. We compare our detectors to the performance of the MSER detector provided with the framework [116] as well as our own tree-based MSER implementation (using SIFT descriptors to obtain the matching scores). While the minimal and maximal region size are chosen to be the same for all the detectors (except the (ω)-tree detector which detects

small regions of very bad quality), the Δ parameter differs between the two MSER implementations due to using either Eq. (4.2) or Eq. (4.5) to calculate the stability function. Thus, this framework is also used to tune the Δ parameter for all the tree-based MSR implementations, as well as the tree-based MSER which is later used to obtain a performance baseline. The final value of all the parameters used for all the tree-based MSR detectors is shown in Tab 5.1.

The second setup evaluates the performance of the detectors as a part of a large-scale image retrieval system. Only the best performing component-tree based MSR detector based on the ToS is included in the retrieval experiments, and it is again compared to the MSER detector (both the implementation provided for the Mikolajczyk et al. [116] framework as well as the tuned tree-based implementation). The performance is measured in terms of *mean Average Precision (mAP)*, using three different publicly available datasets [152, 151, 84], and we show that we achieve a stable improvement over the MSER detectors. The VLAD indexing method [86] is used to aggregate the SIFT [99] descriptors used on the detected patches.

The following Section will focus on region matching, explaining the problem as well as the details of the evaluation framework by Mikolajczyk et al. [116] and the particularities of the used dataset. Section 5.1 ends with presenting the evaluation results of the proposed detectors. The application of image retrieval is addressed in Sec. 5.2. The experimental setup and the datasets are presented, followed by the presentation of the results.

5.1 Region Matching

The first experimental setup used to evaluate the proposed MSR detectors evaluates the performance of the detectors in the context of image matching. Covariant feature detection (cf. Sec. 4.1) is performed on an image. Following, a vector pattern corresponding to a region descriptor (cf. Sec. 6.1) is associated to every detected region or the associated measurement region. The detector-descriptor combination aims to ensure the invariance of the produced descriptors to a wide variety of changes (i.e. viewpoint, illumination, scale, and affine changes), as well as produce descriptors that are distinguished.

Finally, the correspondences are established with descriptors calculated for another image of the same scene. Estimating the geometric transformation from the correspondences between this view pair of the same scene (that are separated by a wide baseline) describes the wide baseline stereo problem [154, 18, 192, 108]. Dependent on the scene type, correct correspondences are successfully established between images with viewing angle differences up to 60° for planar objects [116, 114] using simple two-view matching approaches, up to

an increased range of 80° using more complex approaches such as ASIFT [122] and recent MODS [117].

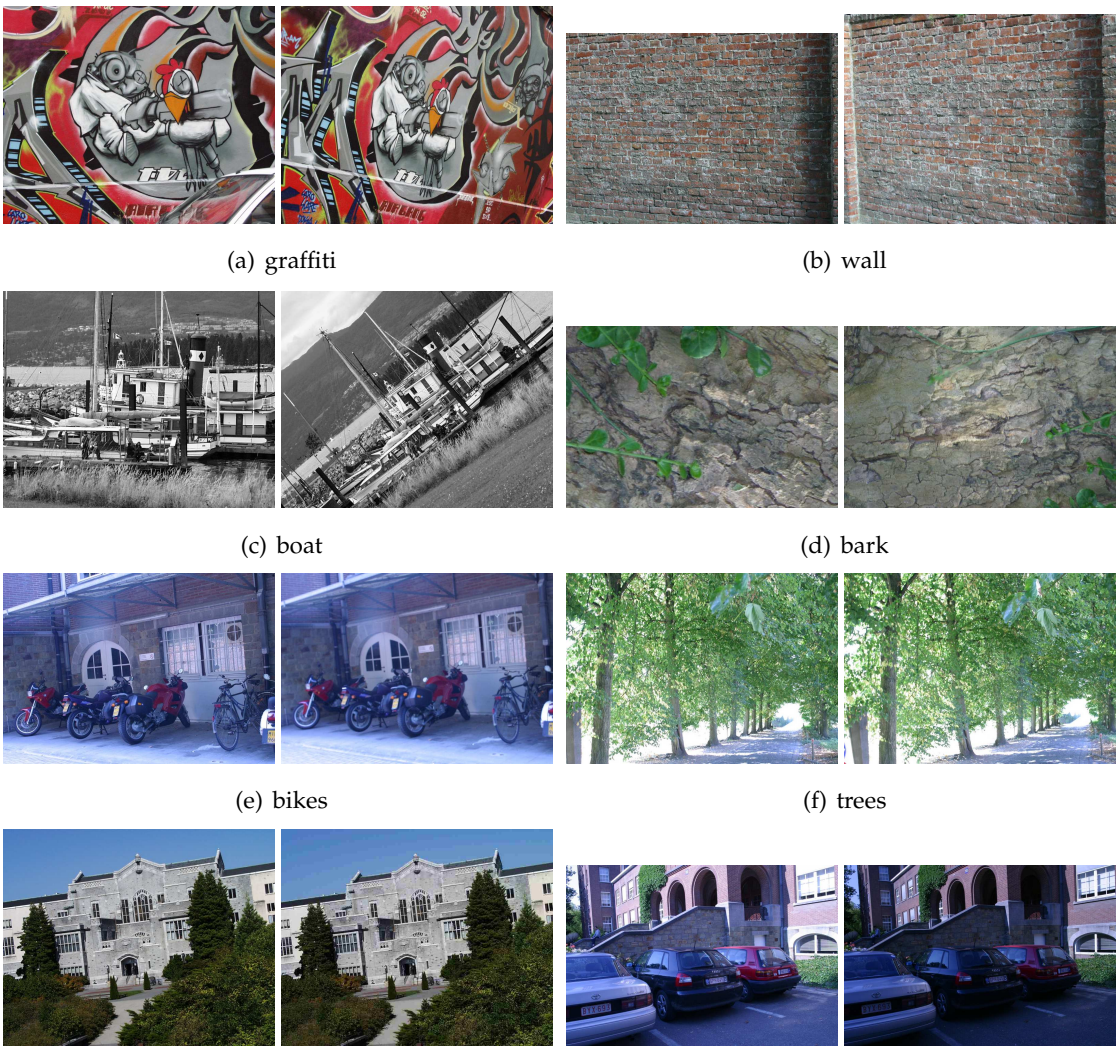
5.1.1 Evaluation Framework

The evaluation of the detector performance for matching is carried out in the framework proposed by Mikolajczyk et al. [116]. While the *repeatability* measure provides a theoretical upper limit of the performance regardless of the descriptor, the *matching scores* are obtained using the 128-dimensional SIFT [99] descriptor (implementation provided with the datasets). These measures as well as the absolute *number of matched correspondences* are presented here in comparison to the original MSER implementation as well as the tree-based implementation of the MSER detector.

The framework investigates the impact of using different detectors on the performance of matching application when 5 different types of changes in imaging conditions are introduced: viewpoint changes, scale changes, image blur, JPEG compression and illumination changes. Additionally, scenes are divided into *textured* and *structured*, depending on the scene type. The structured scenes contain predominantly homogenous edges with distinctive boundaries, while the textured scenes contain one or more highly textured areas. Using these 8 distinct image sequences enables measuring the effect of each imaging condition separately, as well as determine the suitability of the detector for the two different scene types. A known homography is provided between the first and every other image in the sequence as the ground truth. Examples of images from the different datasets are shown in Fig. 5.1.

The *repeatability* measures the ability of the detector to determine corresponding region patches, without any use of region descriptors. This is done by measuring the overlap between the ground truth and the detected regions. More precisely, the measurement region is first determined for the detected regions by estimating ellipses with the same first and second order moments as the detected regions. Following, the overlap is measured between the measurement regions of a reference image for each sequence, and the measurement regions of the other images projected back to the original image (using the ground truth homography). As larger regions naturally have a larger chance of overlap, the framework normalizes all the regions to a common size before checking for overlap. Finally, the repeatability score between a pair of images is computed as the ratio between the number of corresponding regions and the smaller of the number of regions in the common part of the pair of images.

On the other hand, *the matching score* aims to measure the distinctiveness of the detected patches as the more practical compliment to the fairly theoretical *repeatability* measure. To determine how distinguishable the detected regions are, the framework first describes the chosen measurement regions using the 128-dimensional SIFT [99] descriptors after mapping



(a) graffiti (b) wall

(c) boat (d) bark

(e) bikes (f) trees

(g) UBC (h) Leuven

Figure 5.1: Examples of images used in Mikolajczyk et al. [116] dataset. The scene types and changes in imaging conditions are, in order: (a) – structured and (b) – textured viewpoint change, (c) – structured and (d) – textured zoom and rotation, (e) – structured and (f) – textured image blur, (g) JPEG compression and (h) light change. Both scenes used in (g) and (h) contain both structured and textured scene elements.

all the elliptical regions onto circular patches of the same size and determining the dominant gradient. The matching score is again calculated between the reference image for each image sequence and all other images from that sequence. A ground truth for accepted matches is calculated based on the provided homography, and only one best match is accepted per each measurement region from the reference image (a threshold is also used for minimal acceptable overlap error). The matching score is then computed as the number of correct

Table 5.1: MSER parameter values for all detectors.

Parameter	Max-/Min-tree	ToS	α-tree	(ω)-tree
Δ	7	5	8	85
minSize	30	30	30	70
maxSize	1%	1%	1%	1%
maxVariation	0.45	0.4	0.5	0.3
minDiversity	0.25	0.25	0.2	0.4

matches as compared to the total number of detected regions, where a match is determined as the nearest neighbor in descriptor space. Additionally, as the applicability of a detector for a particular domain also depends on the number of correct matches, the absolute number of correctly matched correspondences is also observed.

The matching framework was used as a tuning framework for all the tree-based MSR detectors, including the tree-based MSER implementation. The parameters (listed in Tab. 5.1) were chosen so their repeatability and matching score would follow that of the original MSER implementation on viewpoint datasets of the framework (cf. Figs. 5.1(a) and 5.1(b)). As the partitioning tree based MSR detectors (α-MSR and (ω)-MSR) exhibit different behavior and respond to different types of regions than the inclusion tree based detectors (ToS-MSR and MSER), the tuning was done using only the structured *'graffiti'* dataset for the partitioning tree MSR, while both viewpoint datasets were used for the ToS-MSR and MSER detectors. Despite the possibility of further improving the achieved repeatability and matching scores on the viewpoint datasets by further adjusting the parameters, we remain by our choice to use the viewpoint dataset performance as target performance while tuning for several reasons. First, the repeatability and matching scores of the MSER detector are already considered good, and further parameter tuning would most likely result in reducing the number of detections returned by the other MSR detectors, which is a drawback (especially if we consider image retrieval as a target application rather than image matching). Furthermore, the viewpoint changes are considered one of the most prominent and common transformations between images of the same scene. Finally, we aim to study the detector behavior on all the proposed scene and transformation types, but due to the different nature of the detector their behavior does not exactly follow the trends exhibited by the MSER detector across different datasets. As we do not perform an extensive parameter tuning across all the available scene transformations, we want to avoid overfitting any of our detectors to a singluar image transformation as it is likely to negatively influence the performance of the detector on other datasets.

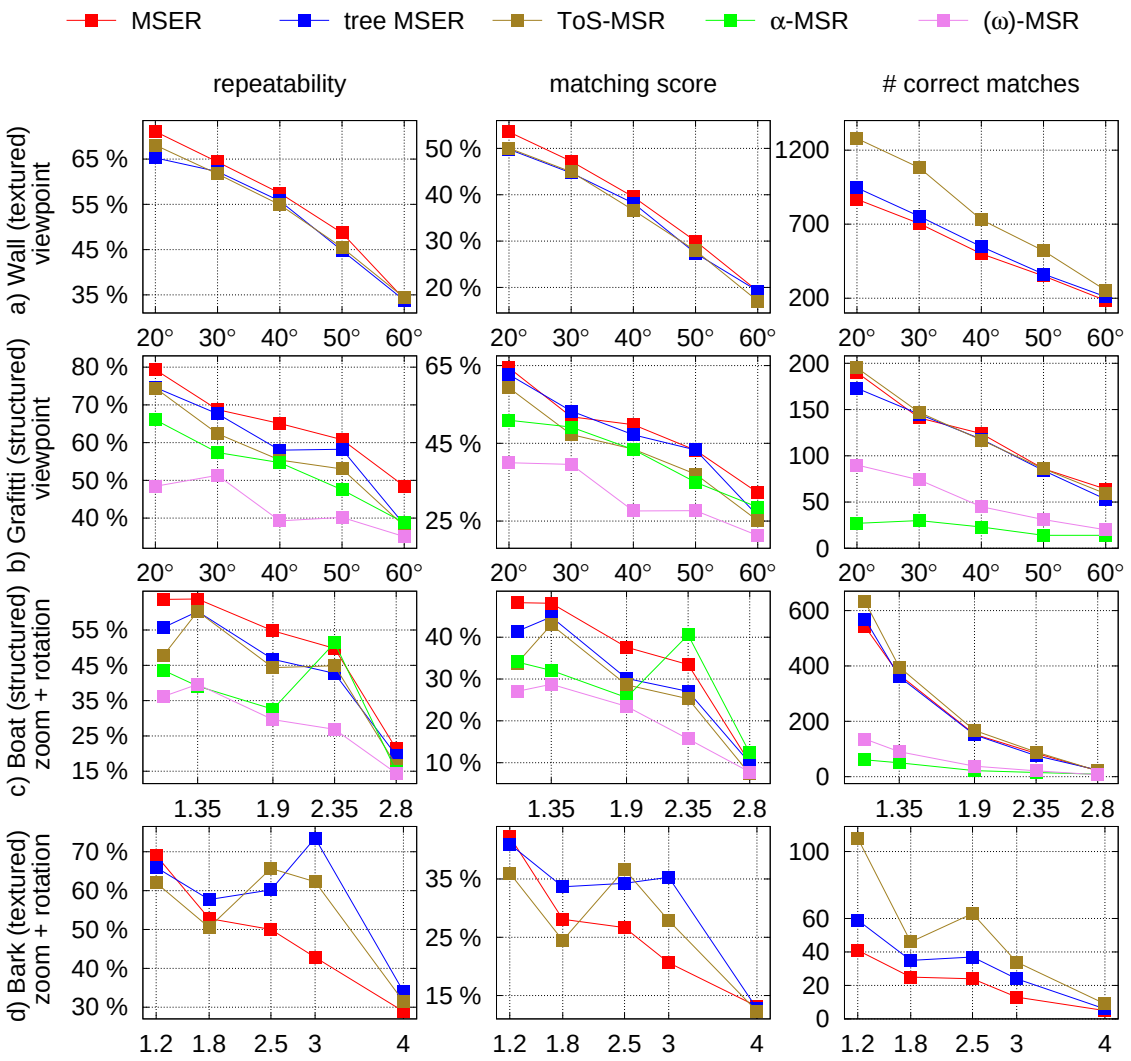


Figure 5.2: Repeatability, matching score and number of correct matches for both structured and textured viewpoint change and zoom and rotation change datasets of the Mikolajczyk et al. [116] framework (the labels on the x-axis correspond to severity of the transformation present in the particular dataset).

5.1.2 Matching Results

The results for all the datasets of the framework are shown in Figs. 5.2 and 5.3. The viewpoint datasets were used to determine the parameters of the detectors, where the goal was to achieve similar repeatability and matching scores to the original baseline MSER (cf. Fig. 5.2(b) for the *'graffiti'* dataset used for all detectors, and Fig. 5.3(a) for the *'wall'* dataset only applicable to the inclusion tree MSR detectors).

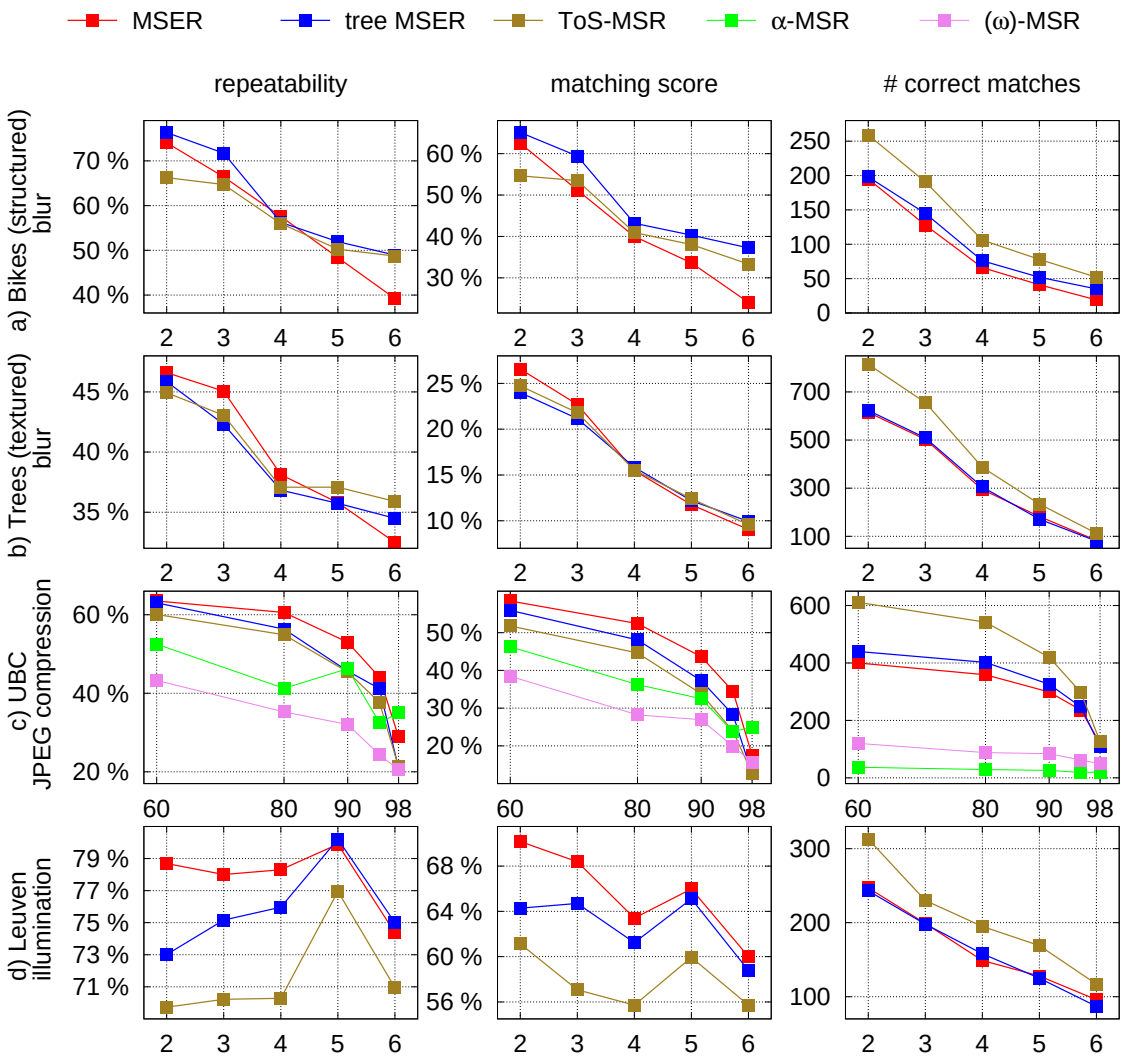


Figure 5.3: Repeatability, matching score and number of correct matches for select representative datasets of the Mikolajczyk et al. [116] framework (the labels on the x-axis correspond to severity of the transformation present in the particular dataset). Datasets shown in (b)–(d) are the ones with the lowest number of MSER detections in the framework.

The ToS-MSR detector shows comparable performance with MSER in terms of repeatability and matching scores on all the datasets (the difference is within 5% on 7 out of 8). The textured dataset focusing on scale and rotation changes (cf. Fig. 5.3(d), *'bark'*) shows the ToS-MSR outperforming the original MSER implementation. The illumination changes dataset (*'Leuven'*), shown in Fig. 5.3(d), is the dataset with the poorest performance of the ToS-MSR. However, this difference is still less than 10% in repeatability and matching scores when compared to the MSER implementations, while the absolute number of correct matches is

increased, and the ToS-MSR detector still outperforms other detectors compared in [116] for this dataset. While the detectors maintain the similar repeatability and matching scores, the number of correctly matched features is consistently higher for the ToS-MSER detector (cf. Fig. 5.2(a) and (d) and Fig. 5.3). This is particularly important for difficult image transformations where an extremely low number of MSER correspondences becomes a limiting factor.

The evaluation results for the α-MSR and (ω)-MSR detectors are only displayed for the datasets where the difference with the inclusion tree based MSR detectors was less than 20% in repeatability and matching score (Figs. 5.2(b) and 5.2(c) as well as Fig. 5.3(c)). This clearly confirms the preference of these detectors for structured scenes. As the *'graffiti'* dataset is close to a cartoon drawing with clear edges, the partitioning tree detectors come close to the performance of the inclusion tree based detectors. However, the better performing α-MSR detector has a very limited response (less than 50 matches per image pair), the increased number of responses from the (ω)-MSR detector comes at the cost of repeatability and matching score performance.

It is interesting to note the performance on images degraded by lossy image compression (shown in Fig. 5.3(c)), where even though neither of the detectors show leading performance on the dataset, they exhibit more stability (i.e. the inclination of the repeatability and matching score curves is closer to horizontal) than the inclusion tree based detectors. A probable explanation for this lies in the fact that compression exaggerates the sharp edges so the regions contained in the α-tree and (ω)-tree are less affected. It would be worth examining the performance of this detector under same kind of compression transformation, but with the images more alike that of the *'graffiti'* dataset (cf. Fig. 5.1(a)). While the α-MSR achieve better performance than the (ω)-MSR, the global range used as the distance in the (ω)-MSR allows for a better control over the accepted regions thus significantly improving on the exceedingly low number of responses returned by the α-MSR. On the other hand, the α-tree contains more candidate regions and allows for a greater choice of regions for the detector. While directly applying the global range as the distance between the nodes for the α-tree used in Eq. (4.3) would result in an identical response to the (ω)-MSR, it would be interesting to study the combination of these two parameters as the region distance function. Due to their lower performance and especially the low number of responses, the two partitioning tree based detectors are not considered for the image retrieval experiments hereafter.

5.2 Image Retrieval

Image retrieval, or more specifically content-based image retrieval (CBIR) comprises all the techniques and approaches that help organize digital image archives by their visual content [57]. While complex image annotation and indexing schemes naturally belong to the field

of image retrieval, in its broadest meaning this also includes concepts as simple as image similarity functions. In short, the aim is to retrieve the database images describing the same object or scene as the query image presented to the system.

In the early years of image retrieval, overcoming the *sensory gap* and the *semantic gap* were posed as the two main challenges pertaining to the field [171]. The sensory gap is then defined as the disparity between the object ground truth in the world and the information stored in an image representation derived from a recording of that scene. This lack of information in the digital representation can arise from a difference in viewpoint and illumination, as well as occlusion and clutter. On the contrary, the semantic gap stands for the lack of correspondence and agreement between the information extracted (automatically) from the visual data and the interpretation of that same data given by a user in a certain context. A user interpretation and examination of images will typically be in relation to a specific object, situation or message, where the user then looks for images containing those objects or conveying that message. Image descriptors used for retrieval tasks rely however on data-driven features, causing disparity between the user interpretation and the information retained by the system.

One more categorization of the image retrieval tasks was offered by Smeulders et al. [171] according to the range and diversity of the image collection under examination:

- *Narrow domain* images have a limited and predictable variability to all the aspects of the object or scene appearance. This mainly applies to the content of the images, but may also extend to the imaging conditions under which the images were taken.

- *Broad domain* problems deal with collections of images with unlimited and unpredictable variability in the appearance even when representing the same semantic meaning. They may also deal with images of unknown object classes or with multiple scene interpretations.

The prior knowledge of the domain can be helpful in selecting features and designing the system. Many problems of practical interest have the image domain falling in between these two categories. However, generic problems of public interest will typically be closer to the broad domain, while specialized and professional applications often have characteristics close to that of a narrow domain. This means that the gap between the features and their semantic interpretation is usually smaller in narrow-domain problems, and domain-specific models are often key to achieving high performance.

In general, when designing an image retrieval system, several steps have to be performed to achieve final content description [171, 57]. The first step is the image processing (or, preprocessing) step where the purpose is to enhance aspects of the image data relevant to the

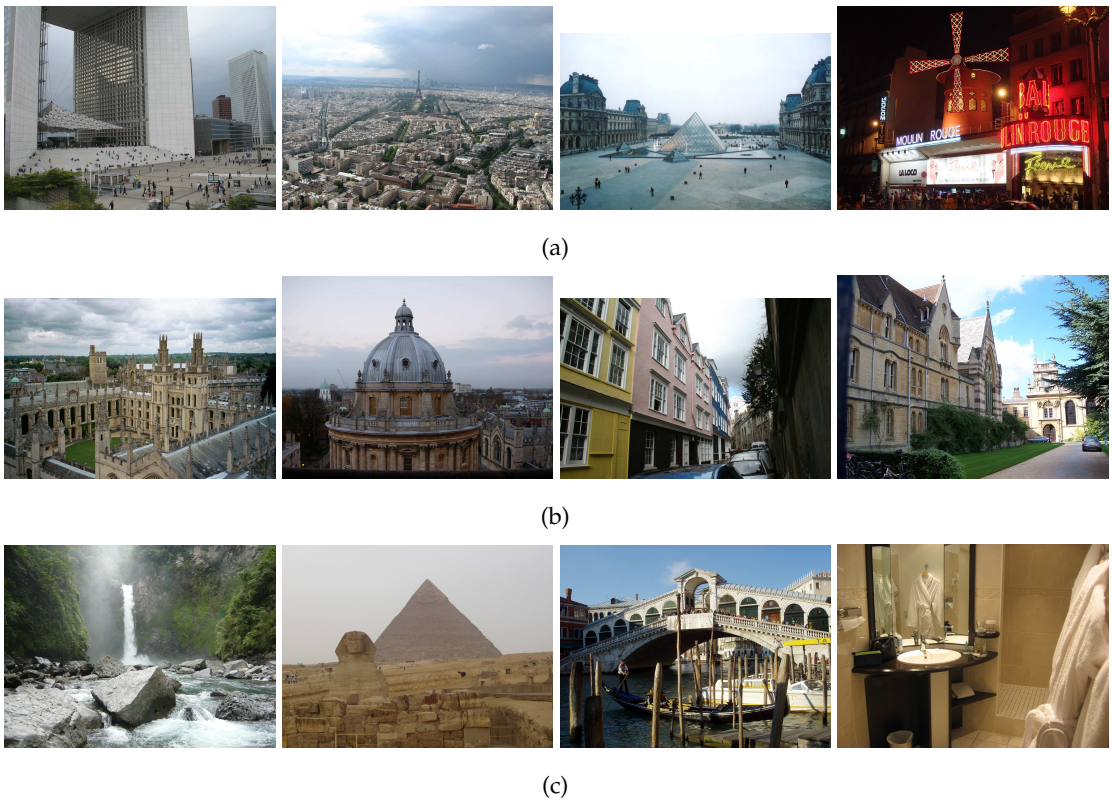
query, and reduce the remaining aspects. This can be done using prior color, shape and texture information about the domain [171].

The goal of the second step is to mathematically describe the image and assess the similarity between images based on these abstract descriptors [57]. This is done primarily by extracting features from the image, also based on color, texture, shape or saliency. The straightforward approach is to work with global features calculated from the entire image. The main advantage of a single-vector image representation (i.e. the representation with global descriptors) is that algebraic and geometric operations can be performed effectively and directly [57]. In this case, the knowledge of the domain can be expressed by formulating a similarity measure between the vectorial image representations [171]. A large number of similarities and measures well suited for retrieval problems is outlined in [171, 57]. However, these representations often lack the detail to represent complex image semantics [57].

On the other hand, locally detected salient regions and features often tend to be more powerful and capable of dealing with problems such as occlusion and non-planar scenes [163]. Especially for broad domain retrieval problems, there is a clear trend moving from global image descriptors towards local approaches [57]. This results in a now-typical retrieval pipeline, consisting of feature detection (cf. Sec. 4.1), feature description (which will be the topic of the next Chapter), and finally aggregation and indexing of the features. Such a system, as well as a representation thereof, were described previously in the introduction to image retrieval in Sec. 1.2 and Fig. 1.2. Aggregation schemes, often strongly related to quantization, are again used to obtain a global representation of the image content. The metric between those global representations is then aimed at measuring the presence of the same set of feature points in two images [171]. Many successful global descriptors have been designed using aggregation techniques such as Bag of Visual Words (BoVW) [170, 137], Vectors of Locally Aggregated Descriptors (VLAD) [86], Fisher kernels [81, 150] and Efficient Match Kernels (EMK) [26]. Finally, an indexing method can be used to perform approximate search in large collections of global descriptors (either obtained directly or by aggregation), either based on structures similar to the KD-tree, like FLANN [124, 125], the NV-tree approach [96] or the Set Compression Tree [11] or relying on compact codes for the speed up [208, 62, 85, 15].

5.2.1 Evaluation Framework

Datasets. For the second experiment, we compare the performance of the ToS-MSR detector in a large-scale image retrieval setup to that of the original and tree-based MSER detector. In order to get reliable retrieval results which we can compare to the state-of-the-art, three different public and widely used datasets were used in this setup (example images shown



(a)

(b)

(c)

Figure 5.4: Example images from the different datasets used for the retrieval experiments. Images from the *'paris6k'* dataset are shown in (a), images from the *'oxford5k'* are displayed in (b), while (c) shows example images from the *'INRIA holidays'* dataset.

in Fig. 5.4):

- *'paris6k'* dataset was used as a training set, to create the visual vocabulary. The dataset contains 6392 images of Paris landmarks [151].
- *'oxford5k'* dataset is first of the datasets used to measure the performance of the detector. The dataset comprises 5062 images of Oxford landmarks as well as distractor images [152]. 55 of the landmark images serve as query images.
- *'INRIA holidays'* database contains a total of 1491 images divided into 500 categories, with 500 images designated to be query images. It includes a large variety of scene types [84].

Image Description and Indexing. As a small preprocessing step, all the images were resized to a maximum of 786432 pixels while keeping the original height to width ration, and a slight intensity normalization was performed (similarly to [84]). The local image features (all three kinds used in the experiments, i.e. ToS-MSR, MSER and tree-MSER) are then

extracted for all the images in the database. We use the SIFT descriptors [99] and their extension to RootSIFT [12] to produce local descriptors corresponding to every feature. The vocabulary is created using a random subset of descriptors belonging to the images of the *'paris6k'* database, using the VLAD aggregation scheme which was ran with 8 cluster centers [86] (the vocabulary creation step is done anew in every repetition of the experiment using 100 times more descriptors than the number of cluster centers used). The VLAD aggregation scheme starts similarly to the BoVW schemes, learning a codebook of k visual words with k-means clustering [101] and assigns each local descriptor to its nearest visual word. Unlike the BoVW approaches, it then accumulates the differences between the cluster center and each assigned local descriptor, for each of the k cluster centers. Assuming the local descriptors are d-dimensional, the total length of the aggregated global descriptors will be $D = k \times d$. The obtained global vectors are also normalized using L_2 norm. Finally, the testing database (i.e. *'oxford5k'* or *'INRIA holidays'*) is described using the provided vocabulary, producing global descriptors for all the images in the database.

Evaluation Metrics. The evaluation measure used is the *mean Average Precision (mAP)*. Unlike the precision, recall and the F-mesure which are set-based measures computed when the output is an unordered set of documents, the mAP is used to evaluate the ranked retrieval results [106]. This measure provides a single measure of quality across recall levels of a system for a set of multiple queries, which are assumed to be diverse enough to be representative of the system effectiveness [106]. For a single query image, if a retrieval system returns K results (i.e. images) we can calculate *precision* and *recall* considering only the first m returned images in an unordered fashion. *Precision at m* is calculated as the ratio between the number of correct (relevant) images in the set of results and the total number of images retrieved at that point, m:

$$\text{precision}(m) = \frac{\text{relevantRetrieved}(m)}{m}, \tag{5.1}$$

while the *recall at m* is defined as the ratio between the number of relevant images in the set of results and the total number of relevant images for that query:

$$\text{recall}(m) = \frac{\text{relevantRetrieved}(m)}{\text{relevantTotal}}. \tag{5.2}$$

In a ranked retrieval context, the precision and recall values for a single query can be computed at each rank and further plotted to produce a *precision-recall* curve. The area under the precision-recall curve over all K results corresponds to the Average Precision AP of a query:

$$\text{AP} = \sum_{m=1}^{K} \text{precision}(m) \times \Delta\,\text{recall}(m). \tag{5.3}$$

This is equivalent to averaging the precision values obtained for the set of top K retrieval results, after retrieving each new relevant result:

$$\mathrm{AP} = \frac{\sum_{m=1}^{K} \mathrm{precision}(m) \times \mathrm{relevant}(m)}{\mathrm{relevantTotal}}, \tag{5.4}$$

where $\mathrm{relevant}(m)$ is an indicator variable with the value 1 if the i-th retrieved image is relevant. Finally, the mAP is calculated as the mean value of the Average Precision for all the queries. The possible values of mAP are between 0.0 and 1.0, indicating the worst and best performance respectively. Summarizing, the main properties of mAP are:

- it does not penalize incorrect predictions, so a large number of retrieved results can be considered, however
- the order of predictions is important and ranking incorrect predictions before the relevant ones in the retrieval results is penalized.
- The performance for each query is weighted equally in the final reported value, even when there exists a variation in the number of relevant documents between the queries.

Aditionally, the mAP is a property characterizing the performance of a retrieval system (on a particular dataset), and the values of mAP can widely vary for different queries. When comparing different retrieval approaches, the value of mAP should be compared on the whole dataset, while the values for the individual queries are indicative of the difficulty of said query.

5.2.2 Image Retrieval Results

The evaluation experiments were repeated 8 times for each detector. Each time, the vocabulary was reinitialized and the k-means clustering for the VLAD aggregation scheme was ran anew. Typical settings for the VLAD aggregation scheme were selected, using $k = 8$ cluster centers for the k-means, which in combination with using 128-dimensional rootSIFT descriptors limits the produced global image descriptors to the length of $8 \times 128 = 1024$ for each database and query image. A random selection of $100 \times k$ descriptors from *'paris6k'* database is chosen every time for determining the cluster centers.

The performance of the ToS-MSR detector compared to the performance of both original and tree-based MSER implementation is shown in Tab. 5.2, with the best and average mAP shown over the 8 experiment runs. The ToS-MSER detector outperforms both MSER implementations on both *'INRIA holidays'* and *'oxford5k'* datasets. The average improvement in terms of mAP when compared to the best performing MSER implementation is 1.7% on the *'INRIA holidays'* and 1.2% on *'oxford5k'*.

detector	*'holidays'*			*'oxford5k'*		
	avg # features/image	MAP		avg # features/image	MAP	
		mean	high		mean	high
MSER	914.78	0.434	0.451	874.02	0.227	**0.252**
tree MSER	1000.57	0.419	0.431	931.08	0.222	0.232
ToS-MSR	1295.85	**0.451**	**0.462**	1160.98	**0.239**	0.250

Table 5.2: Results of the image retrieval experiments, using *'paris6k'* for vocabulary training for the VLAD indexing, and *'holidays'* and *'oxford5k'* for validation. Mean and best MAP values are obtained over 8 experiments with randomly reinitialized vocabulary.

As mentioned in Subsec. 5.2.1, the mAP measure of performance of a single detector across different datasets can often vary more than the performance of different detectors on a single dataset. All the detectors have around 20% lower mAP on the *'oxford5k'* dataset as compared to the mAP achieved when evaluating on *'INRIA holidays'*. This is most likely due to the increased dataset size as well as the presence of the distractor images.

One of the factors contributing to the increased performance is also the increased number of high quality features returned by the detector (already observed in the matching experiments in Sec. 5.1). The increase is present in all the three datasets used in the experiment, ranging from 20–30% compared to tree-based MSER implementation and 30–40% when compared to the provided implementation. This small but consistent increase in the number of features does not cause a noticable decrease in the speed of the retrieval system (when compared to detectors with many responses, e.g. Hessian-Affine which return up to 4 times more detections), and is still comeasurable with MSER response size. Better results could be achieved by further augmenting the number of features returned as their quality is also a factor (e.g. there is around 10% increase in the number of features from the original MSER implementation and tree-based one, but no improvement in retrieval performance). However, this would require the use of a more sophisticated method of tuning the detector parameter as well as a more complex reference setup than the matching framework by Mikolajczyk et al. [116] to compare the performance using different parameters. Analyzing general tree characteristics (i.e. number of nodes, distribution of their sizes, distribution of nodes through tree levels) as part of the future work could also prove useful in determining the optimal parameter choices allowing for good-quality regions without overly restricting the number of responses. Further improvements could be made in the different parts of the retrieval pipeline, such as using different measurement regions, or adding a shape component to the descriptors used to take advantage of the arbitrary shape of the returned regions

as well as the lack of any holes in the detected regions. The hierarchical organization of the spatial relations between the returned features provided by the detector has a potential of being exploited as the part of aggregation or indexing schemes used for the retrieval tasks.

Additional flexibility in modifying the pixel ordering in the MSR detector could be achieved by filtering the basic component trees (cf. e.g. [161]) or changing the hierarchy inherent distance by imposing different levels to the tree regions based on an attribute of choice [30] or directly using a different distance between the nodes of the tree for the calculation of the stability function.

Chapter Summary

The validation of component tree based MSR detectors was presented in this chapter. First, the three proposed detecotrs (ToS-MSR, α-MSR and (ω)-MSR) are evaluated in the image matching framework of Mikolajczyk et al. [116], and compared to the tree implementation of MSER as well as the original MSER implementation. The performance of the detectors is analyzed by scene type as well as invariance to different image transformations.

Secondly, the ToS-MSR detector, which performed best out of the three proposed novel component tree based detectors, was evaluated in an image retrieval setup. We achieve an improvement in image retrieval performance on two different databases (*'INRIA holidays'* and *'oxford5k'*), which we explain by the increased number of detections, as well as better centered measurement regions due to detecting only regions without holes.

After exploring the application of component trees to feature detection, the following chapters will focus on feature description as the complementary part of image retrieval systems. In Chap. 6, a region descriptor based on pattern spectra is extended to be applicable to local image patches, while the performance evaluation of descriptors is presented in Chap. 7.

Chapter 6

Local Pattern Spectra

Contents

Pattern spectra are histogram-like structures originating from mathematical morphology, commonly used for image analysis and classification [107], and contain the information on the distribution of sizes and shapes of image components. They can be viewed as probability density function (PDF) estimates of the image content over range of size and shape classes. They can be efficiently computed using a technique known as granulometry [37] on a Max-tree and Min-tree hierarchy [159, 88].

We study here the 2D pattern spectra, targeting applications in image classification and retrieval (cf. Sec. 1.2 and Sec. 5.2) in which the aim is to retrieve the database images describing the same object or scene as the query. Previous success in using the pattern spectra as image descriptors computed at the global [195, 190] or pixel scale (known as DMP [21] or DAP [55, 141]) inspired investigating their behavior as local descriptors. Two versions of the descriptor are presented - a version directly derived from Global Pattern Spectra which is only rotation invariant [34], as well as a scale invariant version [32]. Moreover, a special attention is given to calculating the proposed descriptors on MSER regions [108] (cf. also Sec. 4.2) as both the region detection and descriptor calculation can be done on the same structure.

Following the contributions presented in the previous two chapters pertainingto the feature detection, we focus here on feature description as the next step in a typical image retrieval system (cf. Sec. 1.2 and Sec. 5.2. First, in Sec. 6.1 we describe feature description in general, further motivating this step of the pipeline and and offering examples of popular state-of-the-art descriptors. Following, in Sec. 6.2 we recall the important characteristics of the SIFT descriptors used throughout the thesis. This descriptor based on histograms of gradient orientations is presented and compared to the proposed pattern spectrum based descriptor due to its prevalent use as a local region descriptor robust against rotation and scale changes. Lastly, in Sec. 6.3 we present the pattern spectra, explaining their definition and structure in terms of granulometries, their construction algorithm as well as the parameters used. Finally, the section is concluded by studying the proposed transition from using them as global image descriptors to applying them to describe local image regions.

6.1 Feature Description

Feature description is a second step of image processing and computer vision systems relying on local regions, keypoints or features, such as image classification, matching or retrieval. The invariance of the description relies on the assumption that the detected salient points (cf. Sec. 4.1) will be localized on the same scene element and that the associated measurement region around the interest point will cover the same part of the scene. The descriptors are constructed to be highly distinctive allowing a single feature to be correctly matched with good probability in a large collection of features, as well as to capture the specific visual appearance of the scene region covered by the measurement region. Thus, it is desirable that the descriptors are invariant (or approximately invariant) with respect to the changes in viewpoint and lighting [54] to aid determining the correct correspondences. Starting from simplest normalized cross correlation between the regions [54, 99], many new different descriptors such as DAISY [188, 189, 214, 213], SURF [19] or FREAK [3] were developed, focusing additionally on fast computation or low computational load. Still, the SIFT descriptors by Lowe [99] remain most widely used while still being improved [89, 19, 12] and still show superior performance according to several surveys [114, 54].

6.2 SIFT Descriptors

SIFT, or Scale Invariant Feature Transform is an approach to transform the image data into scale-invariant vectorial notation at the location of local features [99], presented together with a difference-of-Gaussians (DoG) based detector. However, SIFT keypoint descriptors

have since been applied to responses of various local detectors, provided an ellipse-shaped measurement region.

Typically, when using an arbitrary detector, an elliptical measurement region is first assigned to each detected keypoint. The first step to insure invariant keypoint description is assigning a consistent orientation to a keypoint. For this, the size of the MR associated to each particular keypoint is considered to select the correct scale for the descriptor calculation (this corresponded to selecting the appropriate Gaussian smoothed image if the DoG detector is used [99]), which ensures scale invariance. A dominant orientation is assigned to each keypoint in order to further calculate the descriptor relative to this orientation and additionally achieve rotation invariance of the descriptors.

In the following step, every elliptical MR will be mapped to a circular region of a constant radius and normalized. Additionally, before size normalization, large regions are smoothed with a Gaussian kernel given by the size ratio of the measurement region and the normalized region [114] to emulate selecting the correct scale image in the original approach [99].

Using a circular region, the orientation histogram gradient is formed with 36 bins covering the 360° orientation range. Points in the region are evenly sampled and their orientation calculated to determine the correct bin in the orientation histogram. The gradient magnitude is also calculated for each sample and used to weight the contribution of the sample point to the histogram. The highest peak in the histogram is then used as the dominant orientation of the keypoint, and the further description is done rotated according to this orientation. In case of multiple dominant orientations within 80% of magnitude difference, multiple descriptors will be created at the same scale and location.

While assigning an orientation ensures the rotation invariance, the final step in descriptor computation aims to provide distinctiveness and invariance to other variations such as change in illumination and 3D viewpoint. The descriptor uses the image gradient magnitudes and orientations sampled on the scaled image patch, rotated relative to the dominant orientation. A Gaussian weighting function with σ equal to half of the scaled region width is used to give less emphasis to the gradients far from the keypoint center, as the content far from the region center usually has the greatest effect on misregistration errors. The region is finally divided into an $n \times n$ grid, where $n = 4$ is commonly used, and an orientation histogram with 8 directions is calculated for each of the 4×4 patches. The accumulated magnitudes from each of the sample windows are used as keypoint descriptor, forming a vector of length $4 \times 4 \times 8 = 128$.

6.3 Pattern Spectra as Descriptors

In order to construct a powerful local descriptor exibiting invariance properties to image transformation, we extend [195] to compute the 2D size-shape pattern spectra locally. Unlike the SIFT descriptors, the local pattern spectra can be calculated directly on the detected region of an arbitrary shape, and do not require a measurement region. If a measurement region is still used, there is no limitations on its shape. Hereafter, the challenges faced in order to keep the good characteristics (scale, translation and rotation invariance, as well as the computational efficiency) of the global image pattern spectra descriptor are described. Mainly, the parameters used with global pattern spectra are reexamined together with the parameters newly introduced by the local description scheme.

6.3.1 Attributes and Filtering

In order to characterize an arbitrary region, we can capture its characteristics by assigning *attributes* measuring the interesting region aspects. *Increasing* attributes $K(\cdot)$ give increasing values when calculated on a nested sequences of regions $\mathcal{R}_1 \subseteq \mathcal{R}_2 \subseteq ...$, otherwise they are *nonincreasing* [175]. The simplest increasing attribute one can assign to the region is its Lebesgue measure (corresponding to area $A(\mathcal{R})$ in the 2D case). Other examples of increasing attributes include the radius or area of the largest circle or square fitting into the region, the area or perimeter of the convex hull of the region, diameter of the smallest enclosing circle [37]. Increasing attributes typically describe the *size* of the region.

Nonincreasing attributes are better suited for describing the shape of the region. If the attribute values depend only on the region shape and are invariant to the scaling, rotation and translation of the region, we call them *strict shape* attributes [37]. One example of an attribute describing the shape of a region is an elongation measure called *corrected noncompactness*:

$$NC(\mathcal{R}) = 2\pi \left(\frac{I(\mathcal{R})}{A(\mathcal{R})^2} + \frac{1}{6A(\mathcal{R})} \right). \tag{6.1}$$

$I(\mathcal{R})$ is here the moment of inertia of the region, and the term $\frac{I(\mathcal{R})}{A(\mathcal{R})^2}$ without the correction factor $\frac{1}{6A(\mathcal{R})}$ is equal to the first moment invariant of Hu [78] $I = \mu_{2,0} + \mu_{0,2}$. The correction factor appears when transitioning from the original formula in the continuous space to the discrete image space [209]. A perfectly circular region would theoretically achieve the lowest value of $NC(\cdot)$ equal to 1, where the value would increase towards infinity for a infinetely long thin line.

Another example of a non-increasing attribute is Shannon entropy, defined using the

probability $p(i, \mathcal{R})$ or frequency with which a pixel of gray level i occurs in the region $\mathcal{R}$:

$$\mathcal{H}(\mathcal{R}) = -\sum_{i \in \text{ gray levels}} p(i, \mathcal{R}) \log_2 p(i, \mathcal{R}). \tag{6.2}$$

Low attribute values of $\mathcal{H}(\cdot)$ will be achieved when the region content in homogeneous in terms of gray level distribution, while heterogeneous regions containing many different gray levels will have high values of this attribute.

Raw region moments and the attributes derived from them, such as normalized central moments, center of mass, covariances, skewness or kurtosis [211] can also be used to describe basic geometrical properties of the regions. More examples and a discussion of shape attributes can be found in [37, 159].

By using the attribute to either accept or reject a region, we construct a criterion. A simple criterion $C(\cdot)$ can be constructed to compare the attribute value to a threshold $C_t(\mathcal{R}) = K(\mathcal{R}) > t$, or a more complex criterion can be used. On a binary image B, an *attribute filtering* consists in applying a criterion $C(\cdot)$ to every conneted component of the image and keeping only the regions satisfying the criterion, and can be denoted as an operator $\phi(\cdot)$. An attribute filtering is an *idempotent* operation, as applying it twice to the same image or region has no effect, $\phi(\phi(B)) = \phi(B)$. Furthermore, it is also an *anti-extensive* operation, meaning that it only removes image elements without adding any, $B \geq \phi(B)$. According to these characteristics, an attribute filtering defines a *thinning* operation [175]. Attribute filterings also have the property of not affecting the shape of the preserved regions because only the whole regions are kept.

An *attribute opening* is a specific kind of attribute filtering, where the attribute used is increasing, which we will denote by $\gamma(\cdot)$. This kind of opening can easily be extended to grayscale images I by applying the attribute opening to each binary image in the threshold decomposition of I. In a grayscale image, it will have an effect of removing all foreground (bright) components that do not satisfy the given criterion $C(\cdot)$. A complementary operation of *attribute closing* can also be defined to remove all the background (dark) components that do not satisfy the criterion (i.e. simply by working on an inverted image $-I$). These operations can be efficiently realized by pruning a Max-tree for attribute openings and Min-tree for attribute closings (cf. Sec. 3.1). When a *pruning* is performed on a tree, all the descendants of a node n are removed as soon as the node is removed. This is in accordance with the increasingness of the criterion: as soon as a region does not satisfy the criterion, neither will any of its subsets (i.e. node descendants), so they can all be removed from the image or tree.

However, extending a general attribute filtering (i.e. using a non-increasing criterion) to a grayscale image is not as straightforward. When processing a Min or Max-tree image

decomposition on a tree, there are several different pruning and non-pruning strategies that can be applied:

- The simplest is the *direct rule*, preserving all the regions on the tree that satisfy the criterion, while the content of the nodes that do not satisfy the criterion is merged with their nearest preserved ancestor [159]. However, in practice this criterion is not robust as the decisions are local and do not depend on the decisions on the neighboring nodes. The filtered and restituted image can also have artificial edges if the direct rule is used [37].

  This is a *non-pruning* strategy, meaning that the decision about removing a node does not influence the decisions about preserving the descendants of that node and the children of any removed node are added to the oldest surviving ancestor of the removed node.

- The *max rule* prunes the tree from the leaves up to the first node that satisfies the criterion and has to be preserved. The node is only removed if all of its descendants do not satisfy the criterion. It is more permissive than the direct rule (keeping more of the nodes from the hierarchy).

- The *min rule* prunes the tree from the leaves up to the last node that does not satisfy the criterion and has to be removed. The node is only preserved if all of its descendants satisfy the criterion.

- The *Viterbi rule* (based on dynamic programming problem that can be solved by the Viterbi algorithm [205]) assigns a cost of removal and preservation to each node, and the total cost of pruning is calculated by considering the different combinations of preserve and remove decisions. However, this makes it more complex to implement than the mentioned direct, max and min rules.

  The max, min and Viterbi rules are all *pruning* strategies, meaning that all the descendants of the removed node will also be removed from the hierarchy. As such, applying any of these strategies either removes some of the components which satisfy the criterion or keep some nodes which do not.

- The *subtractive rule* was proposed by Urbach, Roerdink, and Wilkinson [196, 195] in order to realize grayscale shape decomposition on an image. The same nodes are preserved and removed as in with the direct rule, but the (gray) level of the surviving descendants of the removed nodes is also lowered so that their contrast with the background does not change.

  Using the subtractive rule to perform a filtering on a tree achieves a decomposition of an image into its constituent components based on shape rather than size. The filtered

image will contain only the components satisfying the chosen shape attribute and no artificial edges will be created in the image. The method also does not introduce any artificial edges in the difference image due to respecting the component background contrast while filtering, and the difference consequently contains only the components which do not satisfy the attribute.

Further filtering strategies have been proposed in [217], including levelings when starting from the Min and Max-tree as well as shapings when performed on the Tree of Shapes. These filters rely on constructing a Min-tree of the original tree, based on the attribute values calculated from the base hierarchy.

6.3.2 Size and Shape Granulometries

A family of openings $\{\gamma_\psi(\cdot)\}$ characterized by a positive size or scale parameter ψ follows the absorption property [109, 175] if applying a filter of a larger scale after a filter at a smaller scale has no effect:

$$\gamma_\psi(\gamma_\mu(I)) = \gamma_{\max(\mu,\psi)}(I). \tag{6.3}$$

Such a family of openings that satisfies the absorption property is known as a *size granulometry* or a *size distribution* [109]. While algebraic granulometries based on algebraic openings [129] can be constructed, we will be using an attribute opening, based on a criterion $C_t(\cdot)$ comparing the attribute value to the threshold t, which we donote by $\gamma_t(\cdot)$. Then, by using a series of such openings $\{\gamma_{t_i}(\cdot)\}$ with an increasing threshold $t_{i+1} > t_i$ we can formulate a granulometry (by opening).

The granulometric analysis of a 2D image can be compared to a *sieving* process on an image, where the openings used for a granulometry are viewed as a set of sieves of increasing grades [203, 195]. Each opening, corresponding to a certain sieve, removes more components than the previous one until the empty set is reached. They are able to extract size information (description) of the image without any prior segmentation, and with tolerance to overlap. An extensive review of granulometries was offered by Vincent [203].

Using a (strict) shape attribute, the filtering does not exibit the increasing property and thus can not directly form a granulometry. However, attribute thinnings using strict shape attributes are defined to be insensitive to scale, i.e. if λI is the scaling of an image I by a factor λ, then $\lambda\phi(I) = \phi(\lambda I)$. Such an attribute thinning thus satisfies the properties of scale invariance (instead of increasingness) as well as anti-extensivity and idempotence. An ordered set of such strict shape attributes, $\{\phi_{t_i}(\cdot)\}$, where t_i is again the threshold used for the decision criterion, is called a *shape granulometry* in the case it also satisfies the absorption property, cf. Eq. (6.3) [196]. In the digital case, the pure scale invariance of such operators

is often hard to achieve due to discretization artefacts, but a good approximation can be achieved [196].

Performing both the size and the shape granulometry simply amounts to applying consecutive attribute filterings on a Max-tree [196], where the attribute openings and size granulometries will be realized as tree prunings and the shape granulometries make use of the subtractive non-pruning strategy for filtering the tree.

6.3.3 Global Pattern Spectra

Instead of focusing on the details remaining, one can consider the amount of detail removed between consecutive openings of the ordered set $\{\gamma_{t_i}(\cdot)\}$:

$$(s_\gamma(I))(t_i) = -\frac{\mathrm{d}\xi(\gamma_t(I))}{\mathrm{d}t}\bigg|_{t=t_i}, \tag{6.4}$$

where ξ denotes the Lebesgue measure of the image (i. e the number of pixels in the binary case or the sum of gray levels in the grayscale case). This analysis was introduced by Maragos under the name *(size) pattern spectra*, and produces a 1D histogram $s_\gamma(I)$ for an image I storing the amount of image detail for each size class or filtering residue.

The *shape pattern spectrum* can be defined in a similar way, only with using the ordered set of thinnings $\{\phi_{t_i}\}$ forming a shape granulometry. A shape spectrum $s_\phi(I)$ is again obtained by noting the Lebesgue measure of the residue left after each thinning [195]:

$$(s_\phi(I))(t_i) = -\frac{\mathrm{d}\xi(\phi_t(I))}{\mathrm{d}t}\bigg|_{t=t_i}. \tag{6.5}$$

Combining the shape and size pattern spectra, one can obtain *2D size-shape pattern spectra* [195], corresponding to 2D histograms where the amount of image detail for different shape-size classes is stored in dedicated bins. Using a size granulometry $\{\gamma_{t_i}\}$ based on the increasing size attribute $K(\cdot)$ and a shape granulometry $\{\phi_{k_j}\}$ based on the strict shape attribute $M(\cdot)$, we denote with $S(t_i, k_j)$ the bin in the size-shape histogram which contains the Lebesgue measure (i.e. the sum of gray levels) of the connected components falling in the size class between t_{i-1} and t_i, and shape class between k_{j-1} and k_j. The computation of the 2D spectrum, just like the computation of the granulometries, can be performed using a Max-tree (cf. Sec. 3.1) and was proposed by Urbach, Roerdink, and Wilkinson [195]. If we denote the size attribute used by $K(\cdot)$ and the shape attribute by $M(\cdot)$, the global pattern spectrum is computed as follows:

- Set all elements of the array S to zero.
- Compute the Max-tree of the image I.

- As the Max-tree is built compute the size attribute $K(\cdot)$ and the shape attribute $M(\cdot)$ for each node n.
- Also compute the area (i.e. the Lebesgue measure) $A(\cdot)$ for each node n.
- For each node n:
  - Compute the size class t_i from the size attribute $K(n)$.
  - Compute the shape class k_j from the shape attribute $M(n)$.
  - Compute the gray level difference δ between the current node n and its parent.
  - Add $\delta \times A(n)$ to the histogram bin $S(t_i, k_j)$.

The area of each node $A(n)$ can be additionally normalized by the area of the image (i.e. the root node of the tree).

Previous work [195, 190] as well as our own experiments [32, 34] suggest that the lower attribute values carry more information. Thus, a logarithmic binning is used for both attributes, producing higher resolution bins for low attribute values. Let v be the attribute value for one of the attributes, N_b the total desired *number of bins* and m the *upper bound* for that attribute (which can be the maximal attribute value in the hierarchy, or a smaller value if we decide to ignore attribute values above a certain threshold). If the minimal value for the attribute is 1 (as with popular attributes such as e.g. area and the corrected noncompactness), the base for the logarithmic binning b, and the final bin c, are determined as:

$$b = \sqrt[N_b]{m}, \tag{6.6}$$

$$c = \lfloor \log_b v \rfloor \tag{6.7}$$

Enumerating the bins starting from 1, the i-th bin has the range $[b^{i-1}, b^i]$. If the attributes $K(\cdot)$ and $M(\cdot)$ are used, the size of the histogram will be $N_{b_K} \times N_{b_M}$.

When used as global image descriptors [195, 190], the $N_{b_K} \times N_{b_M}$ histograms are mapped in lexicographic order into 1D vectors. As an additional step, the bin values can be equalized by an arbitrary function $f(\cdot)$ (our experiments show that using $f(S(t_i, k_j)) = \sqrt[5]{S(t_i, k_j)}$ ensures good performance in image retrieval as well as a pleasing visualization). An example of a global pattern spectrum (based on a Max-tree) is shown in Fig. 6.1, together with examples of regions contributing to the bins of the spectrum. Additionally, as images tend to contain both bright and dark structures, a anti-size and anti-shape granulometries are required (based on closing and thickening) [195]). This is implemented by running the same algorithm on the inverted image $-I$, effectively working on the Min-tree of the original image I.

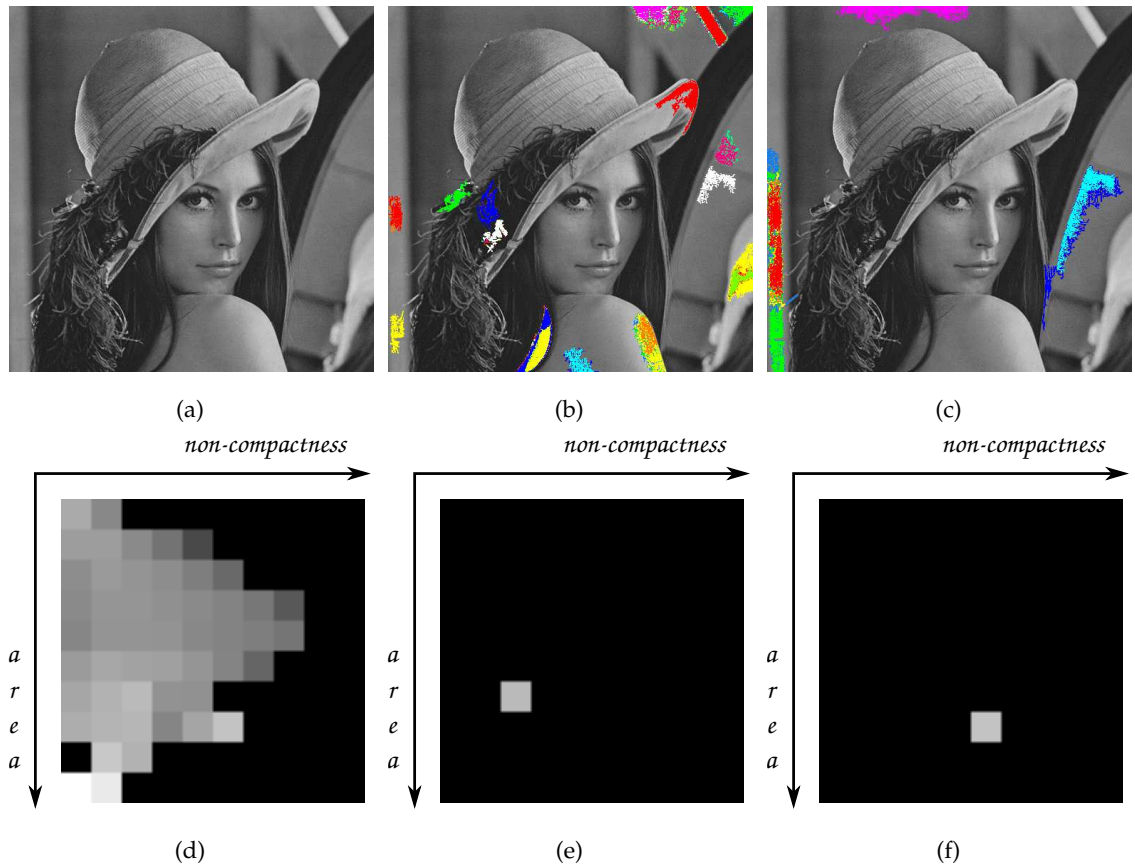


Figure 6.1: The figure gives an example of the global pattern spectra (based on a Max-tree) for the *'lena'* image. The used image is shown in (a), while the corresponding pattern spectrum is shown in (d). In (e) and (f), two distinct bins of the pattern spectrum are highlighted, while the corresponding regions contributing to the highlighted bins are shown in (b) and (c).

6.3.4 Local Pattern Spectra

The aim of *local pattern spectra* (LPS) (introduced in [34, 32], also cf. [31] for an extended version) is to adapt global pattern spectra, introduced in the previous Section, to use in combination with salient region detectors (cf. Sec. 4.1). The LPS are calculated like the global ones, except that the calculation has to be done on a tree corresponding to an image patch returned by the feature detector.

In a general case, the corresponding Min and Max-trees will have to be calculated explicitely for each detected image region. In case of dense approaches used in specific applications such as sattelite imaging [146, 33], extensions of parallel construction algorithms working by first building sub-trees to be merged into a final hierarchy [212, 143, 110] can be used to increase calculation efficiency. However, the MSER regions (cf. [108, 136] as well as Sec. 4.2) are specific since they can be efficiently calculated on a Min and Max-tree, the

Table 6.1: Parameters used in LPS calculation

symbol	significance
$m_{\mathcal{K}}$	upper bound for the size attribute
$m_{\mathcal{M}}$	upper bound for the shape attribute
$N_b^{\mathcal{K}}$	number of size bins
$N_b^{\mathcal{M}}$	number of shape bins
RS	reference scale for the size attribute

same as pattern spectra. Thus we specifically examine the LPS descriptors calculated for MSER regions, both from an algorithmic as well as performance point of view (presented in Chap. 7).

Transitioning to the local version of the descriptor has an additional consequence. While the global pattern spectra were typically calculated for images of same or similar size [190] or taken at the same scale [195], it is desirable for both feature descriptors and detectors to describe regions of different sizes taking the scale information into account [99]. For this reason, we also consider a new parameter influencing the scale invariance property of the descriptors, and propose both scale invariant LPS descriptor (SI-LPS, [32, 33]) and a version that is only invariant to rotation and translation (SV-LPS, [34]).

Achieving Scale Invariance. If we choose to determine the binning base for each region separately in the local description scheme and base it directly on the area of that region, the resulting LPS descriptor is not scale invariant.

Let us consider two versions of the same region at different scales, with the area values belonging to the range $[1, m_1]$ and $[1, m_2]$ respectively. The scale invariance property requires that, for a value $v_1 \in [1, m_1]$, the bin c_1 determined in the original scale is the same as the bin c_2 for the value $v_2 = v_1 \frac{m_2}{m_1}$ scaled to the range $[1, m_2]$. However, this is not the case for $m_1 \neq m_2$, as:

$$c_1 = \log_{\sqrt[N_b]{m_1}} v_1 \neq c_2 = \log_{\sqrt[N_b]{m_2}} v_2. \tag{6.8}$$

Therefore, to ensure the scale invariance, the areas used to determine the binning and the logarithmic base have to be the same for all the regions. This area becomes a parameter of the size attribute in LPS, called the *reference scale RS*.

Using a common scale RS can be seen as rescaling all the regions to the same reference

scale, and has two consequences. First, for a region of size $m > RS$, the minimal value v of this region that can contribute to the spectrum when using a common binning is such that $v' = v\frac{RS}{m} = 1$, and all the (sub)regions with the area smaller than $\frac{m}{RS}$ will be ignored. However, some particular regions with a large enough area can still disappear when rescaling. This is the case for long thin objects with the width (along any dimension) small enough to downscale to under 1 pixel. Such regions should be ignored in the pattern spectrum, even if their attribute values fit with the binning. Because of this, we also determine the maximal possible value of the noncompactness attribute for all of the available area bins and use it as a criterion to discard regions.

Second, the minimal area value (1 pixel) of a region of size $m < RS$ will be rescaled to the value $v' = \frac{RS}{m} > 1$, and the lower area bins at the common scale will be empty. The first area bin $c_{\min}$ that will contain information is then:

$$1 = b^{c_{\min}-1}\frac{m}{RS} \rightarrow c_{\min} = \left\lfloor \log_b \frac{RS}{m} \right\rfloor + 1. \tag{6.9}$$

We compare 2 versions of the descriptor: a) the *scale variant version* (SV-LPS), where the area of each region is used as the local reference scale RS, and b) the *scale invariant version* (SI-LPS) where RS is the same for all regions. The performance of the two versions, as well as stability under the choice of reference scale are studied in the next chapter, together with examining the choices of other parameters. All the parameters used in LPS descriptor calculation are listed in Tab. 6.1.

Algorithm Efficiency. We examine here the special case of LPS calculation when they are used in conjunction with the MSER detector [108]. When using the non-recursive Max-tree algorithm of Nistér and Stewénius [136], several detection and description steps can be done concurrently with the tree construction. However, since the minimal and maximal MSER regions are detected on two different trees (the Min-tree and the Max-tree respectively), the descriptor for the maximal MSER will only be based on the Max-tree, and similarly for the minimal MSER. As the tree is built, all the attribute values for the nodes can be calculated (if the attribute choice permits dynamic calculation from the children node attributes). Additionally, the stability function $q'(\mathcal{R})$ in Eq. (4.5) for the MSER regions can be computed simultaneously (as it is treated as another region attribute). The method is as follows:

- Compute the Max-tree and the Min-tree of the image.
- As the trees are built, compute:
  - Shape and size attributes $K(\cdot)$ and $M(\cdot)$ for each node n (or region $\mathcal{R}$).
  - The area $A(\cdot)$ for each node, and any other useful attributes (i.e. region moments [78]).

  - The stability function $q'(\cdot)$ for each node.
  - Local minima of the stability function, forming the sets of MSER regions.
  - Global pattern spectra [195] (if they are used as additional descriptors to complement the set of LPS).

- For each selected MSER region, repeat the computation of the pattern spectra locally in the sub-tree associated to the region using the calculated attributes.
- Map each calculated pattern spectrum into a $1D$ vector by lexicographic order.
- If desired, combine the produced descriptor vector with other precalculated attributes or indicator values denoting the source tree (i.e. the Min-tree or the Max-tree).

Unlike the calculation of global pattern spectra, the local pattern spectra use the constructed hierarchy but can not be computed concurrently because of different upper limits (for area) and binning scaling value. However, adopting the scale invariant version to concurrent computation can be considered. While it would sacrifice true scale invariance, if the value RS is used as a reference scale, and we are calculating for a region of size m, we can set the largest bin to be $[b^{\lceil log_b m \rceil - 1}, b^{\lceil log_b m \rceil}]$, with the smallest bin having the upper bound $b^{\lceil log_b m \rceil - N_b}$. While not all the values from the whole range of the largest bin will be possible for all the regions, the bin values of the children can be used directly by their parents. When the upper bound of the largest bin changes, the child values can still be used with discarding the values from the smallest bin: the scale of those details is too low to be considered.

Chapter Summary

After revising the concepts used in feature description, the extension of pattern spectra descriptors from global image descriptors to descriptors applicable to local image patches is presented. Region attributes, component tree and image filtering and finally granulometries are introduced in order to define the 2D global pattern spectra.

The pattern spectra are re-examined in the local setting with the LPS descriptors in order to keep the rotation, translation and scale invariance properties of GPS. As keeping the scale invariance property requires introducing an additional parameter, both the scale variant SV-LPS and the scale invariant SI-LPS descriptors are introduced.

The following chapter evaluates the proposed local descriptor in a general image classification task, as well as a satellite image retrieval framework. As the MSER regions [108, 136] are found particularly suitable for use in combination with the LPS descriptors, the classification is performed using MSER regions as input for the proposed descriptor and compared

to the performance of SIFT [99] in the classification application. Due to the nature of satellite image data, an approach using dense image sampling to predetermine the image patches used as descriptor input is used in the second evaluation framework.

Chapter 7

Descriptor Validation

Contents

7.1 Image Classification

In the context of machine learning (also as applied to image processing), classification problems belong to the class of supervised learning. Similarly to image retrieval, it aims to identify a category to which a new observation belongs, thus retrieving a set of similar images or observations, however this is done based on the training set of image data containing instances of observations with previously known category or class [64]. Further, it also relies on distinctive and discriminative features, which are then classified based on distance functions or similarity measures.

The method used for classification is based on the *k nearest neighbors* (kNN) classification technique [98, 187], from the family of non-parametric classification methods within the Bayesian framework. Non-parametric models make no assumptions on the probability distributions (i.e. no assumptions on the number or the values of the parameters of the distribution from which the samples are drawn) of the training samples. Thus, the parameters of the non-parametric methods are derived directly from the samples themselves, and not from

the model [64]. The kNN approach is a simple approximation of the Bayes classifier, where the a priori probability $P(c_i)$ of a sample belonging to the class i is undirectly estimated by $P(c_i) = \frac{n_i}{N}$ in the majority vote, with n_i being the number of class samples and N the total number of samples. In the majority voting that is the basis for kNN, category weighting can be used to alter the prior probability of classes and alleviate the problems caused by skewed sitributions [49]. While in the direct kNN approaches, each of the k nearest neighbors will cast one vote towards determining the category of the query sample, the weight of a single vote can be further altered proportional to the inverse of the distance to the neighboring samples or the category size to which the sample belongs. The kNN density estimator can be written as a kernel estimate of variable bandwith (i.e. the size of the used kernel is varied depending on either the location of the samples or the query point), if the kernel used is chosen to have an uniform density of the unit sphere take into account by the estimator [187].

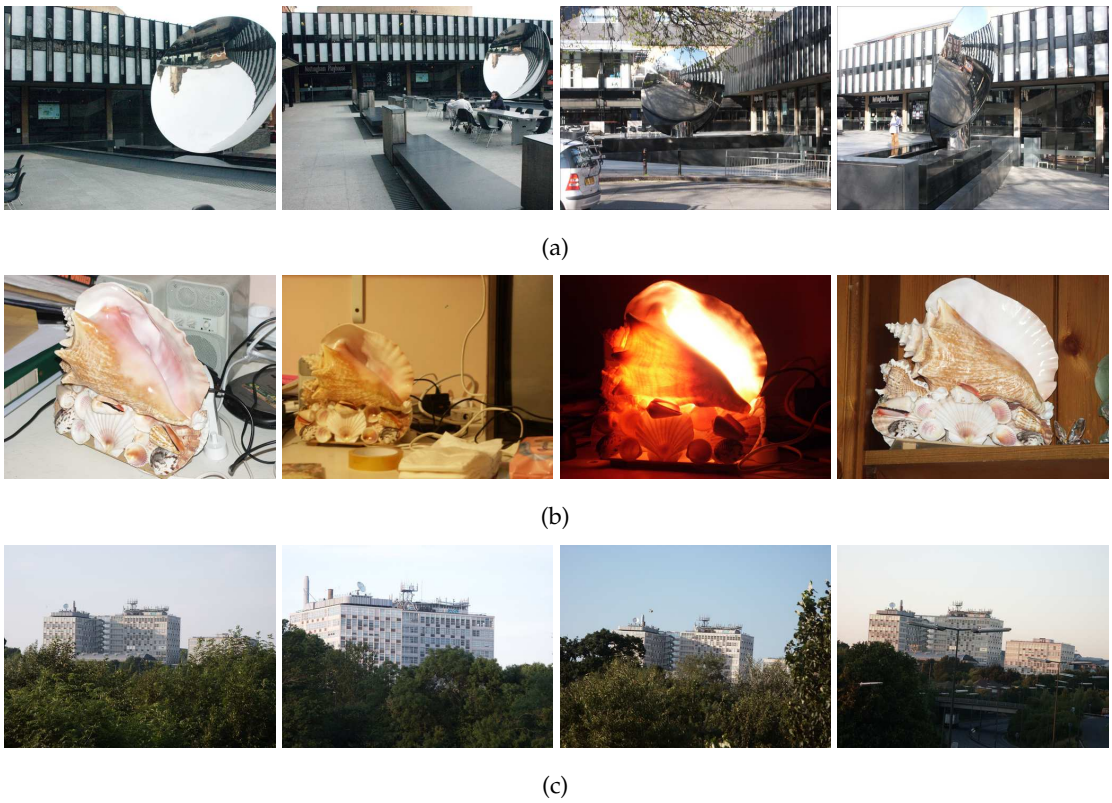
7.1.1 Database and Experimental Setup

To evaluate the retrieval performance of the descriptors without introducing noise in the results with approximate search approaches [170, 86] (cf. also Sec. 5.2), we chose a relatively small *UCID* database [162], on which we can perform an exact search. The performance of our proposed descriptors is compared to SIFT [99].

The whole *UCID* database contains 1338 images of size 512×384 pixels, divided into 262 unbalanced categories with one query image assigned to each category. Examples of images from this database are shown in Fig. 7.1. All the images are treated as grayscale in the performed experiments. We use the MSER Max-tree approach [136] for feature detectors, and compare the performance of the SIFT descriptors [99] with both the scale variant (SV-LPS) and scale invariant (SI-LPS) version of our descriptor.

Global pattern spectra are added to the list of LPS for every image and treated equally to other local descriptors. Note that they are also calculated on a common *RS* when combined with the SI-LPS. We also append several region moments based on the shape of the detected regions to a final version of all the LPS descriptors in order to enhance performance. The influence of adding this additional information to the descriptor is validated experimentally in the next section.

The measurement regions in the evaluation framework by Mikolajczyk and Schmid [114, 116], are based on the ellipses with the same corresponding second moments as the detected region. The ellipse size is then increased three times using affine covariant construction. This approach was used to determine the MR of the detections when used with SIFT descriptors. However, when using the LPS we want to avoid using MR that do not appear in the Max-



(a) (b) (c)

Figure 7.1: Example images from the *UCID* dataset. Each of the rows (a) – (c) is showing images belonging to the same category.

tree or the Min-tree, as we want to be able to use the hierarchies for descriptor calculation. In order to carry out a fair comparison where both descriptors are using the MR of a similar size, but the LPS are still calculated on the regions from the hierarchy, we chose to use the ancestor regions of the detected MSER instead. A descriptor for a region contained in a node n will be calculated on the ancestral region of said node, such that the size of the ancestor is no larger than $xA(n)$. We determine experimentally that using $x = 7.5$ will yield the same average area increases in measurement region size as compared to the detected region size as is obtained for the elliptical MR in [114, 116]. The reason for $x > 3$ even though the region size for elliptical MR is only increased 3 times is that many regions have a much bigger parent region, which is then not considered, and the size increase is on average smaller than x times. This also means that due to using the MR belonging to the hierarchy used for detection and description, the obtained LPS descriptors will also include the shape information (of either the detected region directly or an ancestral region) which gives it an additional advantage when used with detectors returning regions of arbitrary shapes such as MSER.

As we approach this as a classification task, after region detection and description, a sin-

gle database entry for every category is constructed, comprising the descriptors from all the images of that category. A full KD-Tree index [71] is built based on the category descriptors, and stored for querying using the FLANN library [124]. Since there is no descriptor aggregation performed, the number of descriptors for every category will differ. We chose to perform kNN with category weighting to account for the skewed distributions caused by the detector responding with varying amount of responses depending on the scene type. We also use the distance-weighted voting to calculate the contribution of each of the $k = 7$ nearest neighbors, where the contribution of each of the neighbors is inversely proportional to its distance from the query descriptor. This choice was made by examining the distance between several queries and a larger number of their nearest neighbors. While we rarely found more than the closest $3 - 4$ database entries to be at a similar distance, the distances to the query descriptor increased rapidly after the first few nearest neighbors. As the neighbors at a large distance from the query descriptor contribute very litle to the total vote, they permit the choice of $k = 7$ to allow for tolerance. We perform a query on the database with 1 image for every database category. The final category is given through a voting mechanism where each nearest neighbor d_i of a query descriptor q_j will cast a vote for the category $cat(d_i)$ it belongs to:

$$vote(cat(d_i)) = \frac{1}{(L_1(d_i, q_j) + 0.1) \times |cat(d_i)|^{w_{cat}}}. \tag{7.1}$$

$L_1(d_i, q_j)$ refers to the L_1 distance between these two descriptors and $|cat(d_i)|$ is the number of descriptors in the category of the i-th nearest neighbor. w_{cat} is a parameter of the experimental setup taking into account the difference in the number of descriptors belonging to each category (and a choice of $w_{cat} = 0$ permits observing the behavior of the system without this weighting).

However, even if the weighting is used to alleviate the problems caused by skewed distributions, the kNN scheme still performs best if the category sizes are a least of the same order of magnitude. Therefore, in order to prevent a large imbalance in the category sizes when examining the performance of the descriptors depending on the database size as well as the number of examples per class, we use different subsets of the *UCID* database for the experiments rather than the whole database at one. The subsets are chosed in such a way that the number of example images per database category is constant in each database subset. This is ensured by taking only the required number of images from the categories containing a large enough number of examples in the order provided by the ground truth, consequently selecting fewer categories as more example images are required per category. Table 7.1 summarizes the subsets of the database used for experiments presented herein. To separately study how the number of categories and the number of examples per category affect the performance, further experiments were performed on the subsets of *ucid5*–*ucid3* for a decreasing number of examples per category to investigate the influence of changing

Table 7.1: Subsets of the UCID database used in experiments.

	# categories / examples	**categories selected**
ucid5	31 / 5	all UCID categories with ≥ 5 examples
ucid4	44 / 4	all UCID categories ≥ 4
ucid3	77 / 3	all UCID categories ≥ 3
ucid2	137 / 2	all UCID categories ≥ 2
ucid1	262 / 1	all UCID categories

Table 7.2: Rescaling on different databases used.

	query	**DB #1**	**DB #2**	**DB #3**	**DB #4**	**DB #5**
ucid5r	$\times 1$	$\times 4$	$\times 2$	$\times 1$	$\times 0.5$	$\times 0.25$
ucid5q4	$\times 4$	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$
ucid5q2	$\times 2$	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$
ucid5q05	$\times 0.5$	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$
ucid5q025	$\times 0.25$	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$

only the number of example images.

The parameter tuning was done using the *ucid5* subset. Furthermore, in order to test the influence of scale change on the performance we use the *ucid5r* database, obtained from *ucid5* by upscaling 2 of the database images, and downscaling another 2, while the query and one of the database images are left at the original scale. Additionally, to examine separately the influence of rescaling by different amounts, *ucid5q4*–*ucid5q025* databases are constructed, where only the query is rescaled. The precise scales for each of the rescaled databases are shown in Tab. 7.2.

The measures we used are mean Average Precision (mAP) and precision at one, precision(1) or P@1. Performance for different values of w_{cat} are shown in Fig. 7.2(d) and Figs. 7.4(a)–7.4(e), but when summarizing the results, only the performance for the optimal w_{cat} value for each experiment is shown. This choice is made in order to present a fair comparison, and since not all the descriptors reach their peak performance for the same value of w_{cat}. This is additionally justified as this parameter is not present when using an aggregation

scheme.

7.1.2 Parameter Tuning

We perform the classification experiments with LPS descriptors using the area attribute $\mathcal{A}(\cdot)$ as the size attribute, and the corrected noncompactness as the shape attribute (cf. Eq. 6.1).

Binning parameters. With the area attribute, the upper bound used, $m_{\mathcal{A}}$, is simply the size of the region: we can plausibly expect regions of all sizes lower than the size of the region itself to be present in its decomposition. We confirm the assumption that very few regions have high values of the noncompactness attribute established in [195, 190], by examining the attribute values of noncompactness for regions detected on a random selection of *UCID* images. Based on this observation, noncompactness values higher than a certain threshold can be safely ignored. The optimal values for this threshold m_{NC} for both SV-LPS and SI-LPS were determined by examining the performance of the values close to the ones used in [195, 190]. Similar experiments were done to determine N_b^{NC} and N_b^A. The parameter tuning experiments for the *ucid5* database are shown in Fig. 7.2, where a technique similar to coordinate descent [138] was used to find the optimal combination of parameters.

For both descriptors, we chose $N_b^{NC} = 6$ and $N_b^A = 10$. To choose between several values of m_{NC} performing well on *ucid5*, we compare their performance on *ucid4–ucid1* as well. This was done as the performance for different values of m_{NC} is fairly stable (only about 5% difference for values shown on Fig. 7.2(a)). Surprisingly, we also found an alternative set of values for SV-LPS with the lower value of $N_b^A = 9$ but a higher $m_{NC} = 57$. The optimal values as well as the best alternative choices are shown in Tab. 7.3. As an alternate set of parameters was found producing shorter SV-LPS descriptors, the possibility of further shortening the SI-LPS without the loss in performance should also be investigated.

Image moments and global pattern spectra. All normalized central moments up to the order 5 were considered to be appended to the LPS descriptors, by examining the influence of each of the moments separately to the final descriptor performance. The weighting factors are also determined in this fashion for the 5 best performing moments, with the goal of the moment components of the descriptor vector being of the same order of magnitude as the components originating from the LPS bins. The final weights used are 20 for $n_{1,1}$ and 10 for other moments used. We also append an indicator value signifying if the region described is a maximal or minimal MSER, additionally increasing the distance between such descriptors. Different values for the indicator value were also tested, and appending 0 for the minimal MSER and 2 for maximal MSER was found to have the most beneficial influence on the performance. The global pattern spectra were also appended to the list of image descriptors for every image, as they achieve mAP around 70% on the *ucid5* database by themselves. The

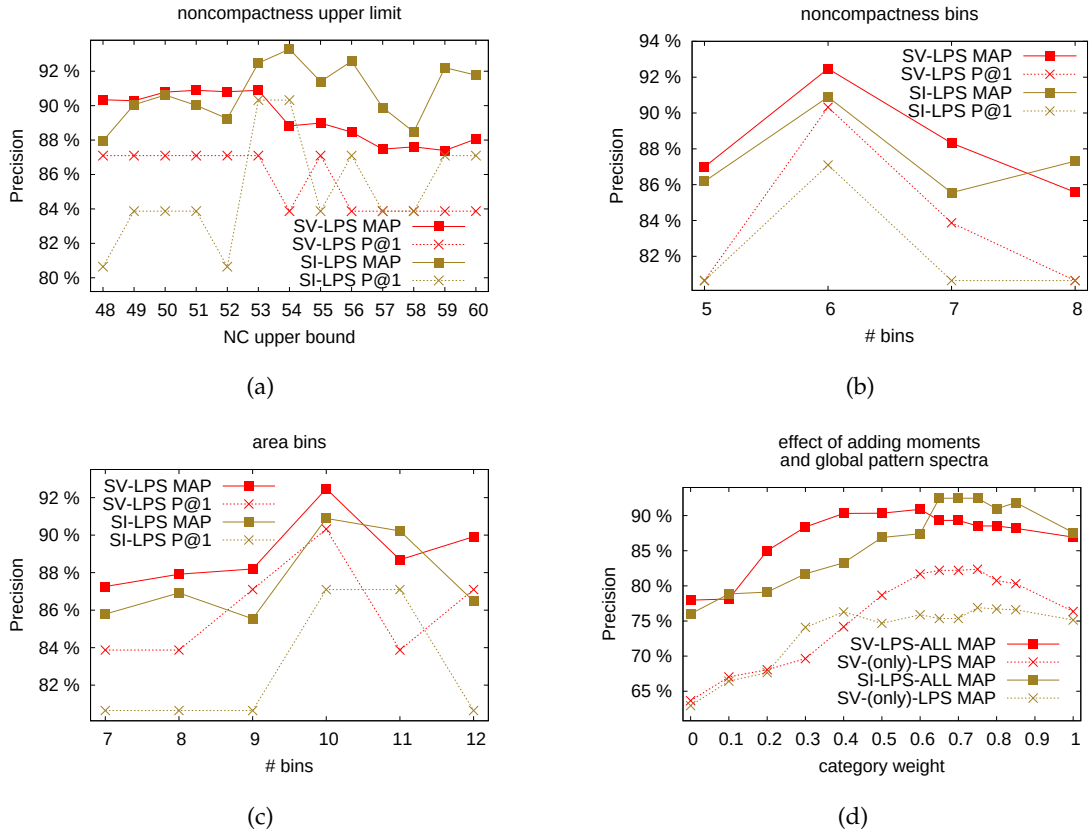
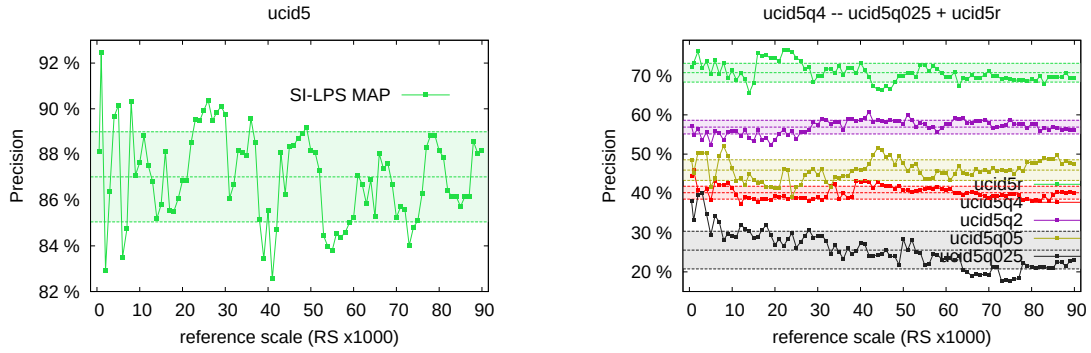


Figure 7.2: Parameter tuning on *ucid5* database. The effect of varying the upper bound for noncompactness is shown on (a), similar for the amount of noncompactness bins on (b), and the area bins on (c). The effect of adding the moments and indicator value to the descriptor, with the best parameter settings is shown in (d). Note that the global descriptors for the SI-LPS are calculated with the scale value used for the other descriptors, and not using image size.

improvement achieved by combining these values, as well as the indicator values distinguishing minimal and maximal MSER, with the LPS descriptors shown in Fig. 7.2(d) for the optimal parameter choice.

Reference Scale influence. We test the performance of the SI-LPS for a range of reference scales between 500 and 90000. The upper limit of the tested *RS* corresponds roughly to half of the size of the database images, and we test different *RS* values in steps of 1000 (the value 500 is considered instead of the loewst value of 0, as scaling the region to size 0 would discard all the content). The results in terms of mAP, as well as their mean and standard deviation are shown in Fig. 7.3(a). The performance is fairly stable under varying reference scale, with the difference between best and worst performance lower than 10% and a small



(a) (b)

Figure 7.3: Performance of SI-LPS for a range of reference scales (mean and standard deviation displayed). The performance for the *ucid5* database is shown in (a), while (b) summarizes the influence of this parameter for all the databases with scale changes (listed in Tab. 7.2).

standard deviation for the chosen range. Some significant local maxima and minima still exist, most likely due to quantisation effects, and should be examined more closely. All further results on *ucid1–ucid5* are obtained using the scale parameter $RS = 1000$ resulting in best performance on *ucid5*. The influence of the reference scale when scale changes are introduced to the database is also analyzed, and shown in Fig. 7.3(b). This will be discussed together with other experimental results regarding scale invariance under strong scale changes in Sec. 7.1.3, but clearly demonstrates that the stability under reference scale is not negatively influenced by scale changes in the database.

Optimal choices for all the parameters are shown in Tab. 7.3. Based on this, the final size of the pattern spectra is 10×6 for both versions of the descriptor, and the final length of the descriptor if $60 + 5 + 1 = 66$ due to adding normalized region moments as well as the indicator variable to the descriptor. Thus, the LPS descriptors used in these experiments are only half the size of SIFT descriptors. Additionally, the alternate parameter values in Tab. 7.3 suggest that it should be possible to construct even shorter versions of this descriptor without the loss of distinctiveness.

7.1.3 Results

Varying the Number of Categories and Examples. We compared the performance of SIFT with that of our LPS descriptors, and both descriptor versions perform closely to SIFT descriptors in the experiments on *ucid1–ucid5* databases. These results, for a **(reduced)** range

Table 7.3: Optimal parameter values for the LPS (best alternative parameter choices also given).

parameter	value SI-LPS	value SV-LPS
$m_{\mathcal{A}}$	region size	
m_{NC}	53 (54, 56)	53 (57)
$N_b^{\mathcal{A}}$	10	10 (9)
N_b^{NC}	6	
RS	1000	region size
$w(n_{1,1})$	20	
$w(n_{2,0}), w(n_{0,2}),$ $w(n_{4,0}), w(n_{0,4})$	10	

of weights w_{cat} and the best MSER and LPS parameters (as shown in Tab. 7.3) are shown in Fig. 7.4, with a summary in Fig. 7.4(f).

The performance expectantly decreases with the increase of database size and the decrease of the number of examples per category. Further experiments aiming to separately examine the influence of these two factors are shown in Fig. 7.5, where the experiments on *ucid3*–*ucid5* were repeated while decreasing the category size. The rate of precision decline w. r. t. the number of examples per category is lower for the both versions of LPS descriptors (cf. Figs. 7.5(a) and 7.5(b) and compare to Fig. 7.5(c)).

When considering the results presented in Fig. 7.4 and Fig. 7.5, we can claim that our descriptors outperform the SIFT descriptor on the *ucid4* and *ucid5* databases. Their performance is comparable on the whole database subsets, but further reducing the number of examples clearly shows the advantage of using LPS descriptors on these databases. We can report comparable results with SIFT on the *ucid3* and a slightly worse performance than SIFT on *ucid2* dataset. On the *ucid1* dataset, both our LPS descriptors are significantly outperformed by SIFT. However, it is known that minimal number of examples (growing when more categories are used) is required for classification. As the *ucid1* dataset is the subset with the largest number of categories used, the classification results, using only the example images of this dataset as a model, might depend on chance and are not as reliable as the results on *ucid2*–*ucid5*.

Besides the performance, it is important to note here that the descriptor is also calculated faster than SIFT for the MSER regions, and that on the largest database subset used, the

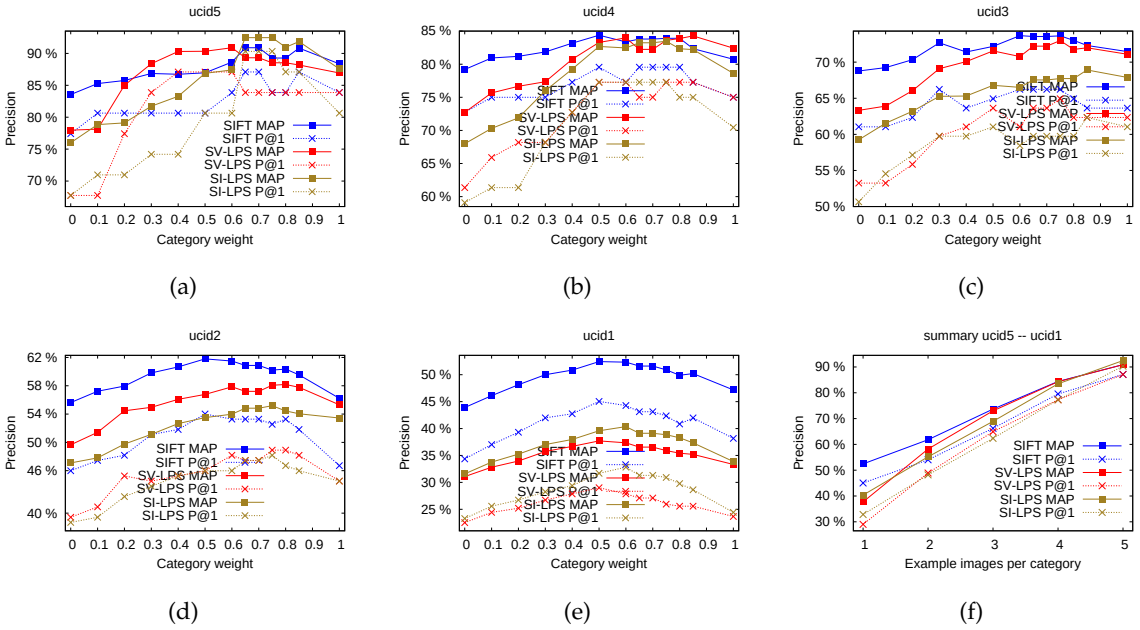


Figure 7.4: The results for the final version of the descriptors expressed in terms of mean Average Precision (mAP) and precision at 1 (P@1) for *ucid5–ucid1* dataset for varying category weights are shown in (a)–(e). The results for *ucid5–ucid1* are summarized on (f) (performance shown for optimal weight w_{cat} for every dataset).

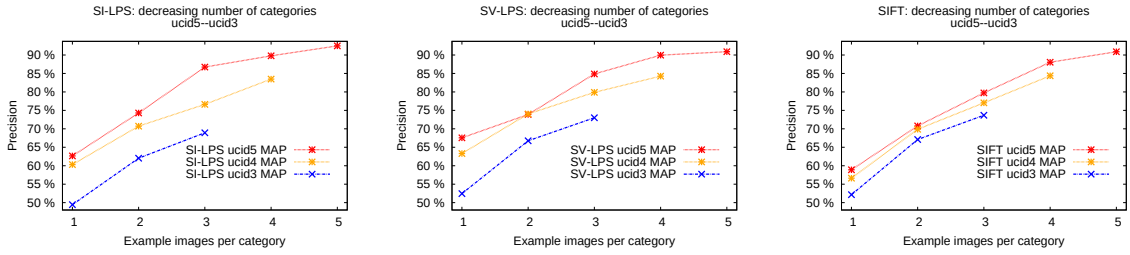
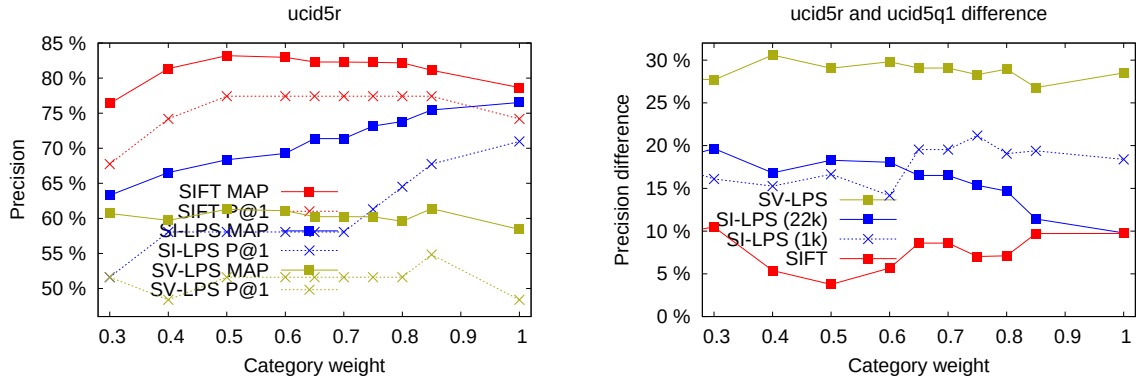


Figure 7.5: Summarized experimental results on *ucid5* (using 5–1 examples per category), *ucid4* (4–1 examples) and *ucid3* (3–1 examples). Only the highest precision per dataset is shown. The results are shown separately for the three descriptors, with SI-LPS shown in (a), the SV-LPS shown in (b) and SIFT shown in (c).

query speed for LPS is around 4× faster than that for SIFT (when the LPS descriptor of size 66 is used). As a smaller version of SV-LPS was already found, it is likely possible to further shorten the SI-LPS as well and achieve even faster query speeds without a loss in performance.

Scale Changes. As the UCID database is not very challenging in terms of scale change,



(a) (b)

Figure 7.6: The performance of SI-LPS descriptors using the optimal value of $RS = 22000$ is compared to the performance for SIFT and SV-LPS descriptors on *ucid5r* dataset for a range of w values in (a). The decline in performance when as compared to the performance on the dataset before any rescaling (*ucid5*) for all three descriptors is shown in (b). For the SI-LPS, this difference is shown for both the optimal RS value on *ucid5r*, as well as the optimal value for the *ucid5* dataset, $RS = 1000$.

further experiments were done after manually rescaling some of the images in the *ucid5* database subset. In one set of experiments, only the query image was rescaled (downscaled or upscaled), corresponding to *ucid5q025*–*ucid5q4* datasets. Additionally, to examine the influence of introducing different scale changes at once, all the example images were resized by different scale factors in *ucid5r*. All the database subsets with introduced scale changes are listed in Tab. 7.2.

Before examining the performance on these datasets, we need to validate the choice of the reference scale parameter RS. This is shown for all the rescaled subsets in Fig. 7.3(b), where it can be seen that the performance on the (composite) *ucid5r* database is in fact more than a combination of the performance contributions when only one type of scale change is introduced. We can conclude, expectedly, that downscaling has a more severe effect on the performance than upscaling as it always results in the loss of image detail. We can also see that a relative stability under the range of reference scales is preserved after introducing scale changes, however the optimal performance is achieved for a reference scale $RS = 22000$. Still, the relative stability under the reference scale change can be seen in Fig. 7.3(b), and comparing with Fig. 7.3(a) confirms that using any of the two optima ($RS = 1000$ or $RS = 22000$) still gives good performance on either of the datasets.

Finally, the performance comparison of LPS and SIFT descriptors for a *ucid5r* database, comprising different scale changes, is shown in Fig. 7.6. The performance with the best

choice of the w_{cat} parameter of the SI-LPS descriptor comes close to the performance of SIFT in Fig. 7.6(a). It is also consistently higher than the performance of SV-LPS for all the values of w_{cat} (and for all the values of *RS*). The decline in performance on the *ucid5r* database as compared to the database with no rescaling is shown for all descriptors on Fig. 7.6(b) (for SI-LPS, for both the reference scale best performing on *ucid5r* and *ucid5*). In this figure it is clearly visible that the performance drop is much stronger for the SV-LPS, i.e. that the SI-LPS indeed have scale invariant properties.

Discussion. Prompted by the previous successful application of global pattern spectra in image retrieval context [196, 190], here we validate a local region descriptor based on pattern spectra. On the chosen subsets of the *UCID* database [162], the classification results obtained were improved when compared to only using global pattern spectra (almost 20% in MAP on *ucid5*), and matched the performance of the SIFT descriptor. The constructed SI-LPS descriptors keep all the invariance properties of the global pattern spectra (translation, rotation and scale invariance).

The proposed descriptors have another advantage. In addition to the description calculation process being slightly faster for the pattern spectra than for the SIFT descriptors, our descriptors length is only half of the length of SIFT. This makes using these descriptors much faster – performing 262 queries on an index of the size 262 (*ucid1* dataset) took 4 times longer using SIFT descriptors. This suggests that (especially in large scale retrieval systems), we can use more example images in order to enhance the precision, while still performing faster than SIFT.

As the performance of the descriptors depends on a lot of parameters, we need to explore a way to determine the optimal parameters automatically. Also, while the LPS descriptors are rotation invariant, enforcing scale invariance introduces an additional parameter. In addition to examining this new parameter closer, both SI-LPS and SV-LPS were evaluated on a database focused on scale changes to determine the value of true scale invariance in such cases, which confirmed additional stability properties of SI-LPS.

It is probable that the results could be even further improved by combining the current LPS with pattern spectra based on other shape attributes, like in [190]. Lastly, the L_1 distance, designed to compare vectors of scalar values, is not the best choice for comparing histogram-like structures. Using different distances such as Bhattacharya distance (also called Hellinger distance) [25], or even divergences such as one proposed by Mwebaze et al. [126] which take into account the nature of the descriptor should also improve the performance. Application of techniques such as rootSIFT [12], aimed at improving the performance of histogram-based descriptors should be considered. Direct application of rootSIFT was not yet considered as the the rootSIFT approach requires vector normalized to unit length as input, while the LPS descriptors were not normalized to preserve the information about the amount of image

detail (and amont of image detail per scale), However, this information could be preserved as a separate component of the final descriptor calculated based on the norm of the vectorized descriptor, while the rootSIFT could be separately applied to the normalized part of the descriptor.

7.2 Satellite image retrieval

Proliferation and increasing performances (spatial precision, revisiting frequency) of Earth Observation satellites lead to massive amount of satellite image data. Mining such data is of primary importance and required to solve various problems. Previous success of attribute profiles, pixel-wise features similar to pattern spectra [55], as well as other morphological features [7] in solving image retrieval issues in remote sensing motivated examining the performance of LPS in satellite image retrieval tasks.

As in general image retrieval tasks, satellite image retrieval is achieved by means of computing descriptors, either globally for the whole image or locally on image patches. Those descriptors (first aggregated in case when multiple descriptors per image are used resulting from using patches of the image) are further used in dedicated indexing/retrieval schemes [207, 65, 146, 16]. However, it is possible to preselect the image patches defined on a regular grid over the image for descriptor calculation [146] and avoid the feature detection step. This is due to the nature of remote sensing images, moreover specifically for this dataset, where the image patches used are small and of limited content, which that greatly alleviates the need for extracting regions of interests. They are often characterized mostly by texture, or containing only very few prominent structures (objects). As such, the expected response of the MSER detector would not return a sufficient number of keypoints, and it was shown that the dense SIFT approaches [146] outperform similar SIFT approaches based on keypoints [220].

Thus, for the second evaluation experiment of LPS descriptors, we examine their performance in satellite image retrieval that allows retrieving geographic objects having a similar appearance when visually observed from Earth Observation satellites [34]. If all the preselected local patches are of the same (or very similar) size, the scale invariance property holds. In an approach using multiple local patch sizes to achieve image description at multiple scales, a common reference scale *RS* is used to retain scale invariance. We also present the improvement gained by using the LPS over using GPS with similar settings as global image descriptors.

7.2.1 Dataset and Evaluation Metrics

We have conducted our experiments on two publicly available datasets: ImageNet Large Scale Visual Recognition Challenge 2010 (ILSVRC2010) Validation dataset [157] when a training set had to be used, while the validation was done on a satellite image retrieval dataset, namely the UC Merced Land Use Dataset [1] [220]. The Merced Dataset contains 2100 color *RGB* images organized into 21 classes (100 images per class), examples of which are shown in Fig. 7.7. All images are *RGB* color samples of size equal to 256×256 pixels. We compute our descriptors firstly on the grayscale versions of the images, with the conversion $Gray = 0.299 \times R + 0.587 \times G + 0.114 \times B$. For our most successful global approach we have additionally tried tripling the size of the descriptor by concatenating the descriptors obtained when applying the same approaches separately on R, G and B channels.

The evaluation metrics chosen is ANMRR, as it is the most commonly metric used on this dataset and allows for straightforward comparison with other published results [7, 6, 220, 146]. ANMRR stands for *average normalized modified retrieval rank* and is commonly used to measure effectiveness of MPEG-7 retrieval [105]. Given a query q or all the queries of a same class, a number $\mathrm{K}(q)$ is defined, which denotes that only the first $K(q)$ returned images are considered as feasible in terms of retrieval evaluation and is often set as twice the size of the ground truth set $\mathrm{NG}(q)$. Assume that the k^{th} ground truth image is retrieved at $\mathrm{Rank}(k)$, a penalty function $\mathrm{Rank}^*(k)$ is defined for each retrieved item:

$$\mathrm{Rank}^*(k) = \begin{cases} \mathrm{Rank}(k), & \text{if } \mathrm{Rank}(k) \leq \mathrm{K}(q) \\ 1.25\,\mathrm{K}(q), & \text{if } \mathrm{Rank}(k) > \mathrm{K}(q) \end{cases} \tag{7.2}$$

From all the penalties $\mathrm{Rank}^*(k)$ for each query q, the average rank (AVR) for that q is defined:

$$\mathrm{AVR}(q) = \frac{1}{\mathrm{NG}(q)} \sum_{k=1}^{\mathrm{NG}(q)} \mathrm{Rank}^*(k) \tag{7.3}$$

After the intermediate step, ANMRR is directly defined as:

$$\mathrm{ANMRR} = \frac{1}{\mathrm{NQ}} \sum_{q=1}^{NQ} \frac{\mathrm{AVR}(q) - 0.5(1 + \mathrm{NG}(q))}{1.25\,\mathrm{K}(q) - 0.5(1 + \mathrm{NG}(q))} \tag{7.4}$$

where NQ is the number of queries. Thus ANMRR obtains values in range of 0 for best results, and 1 for worst results.

7.2.2 Settings of Pattern Spectra Approaches

Global Pattern Spectra. In the base approach, we calculate the GPS descriptors directly on the complete image samples. The area $\mathcal{A}(\cdot)$ is chosen as the size attribute, while the shape

[1] available at: http://vision.ucmerced.edu/datasets/landuse.html

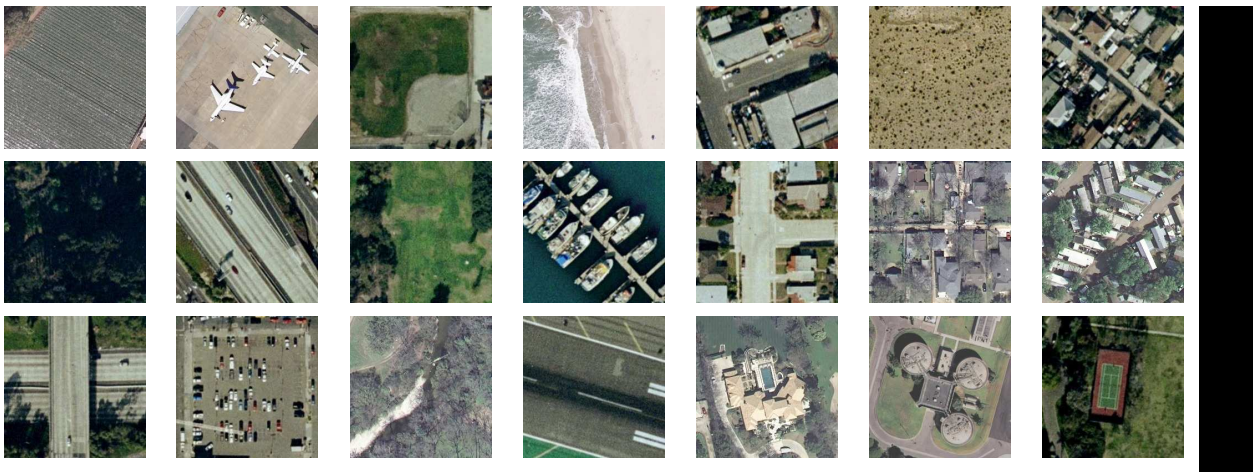


Figure 7.7: Merced dataset

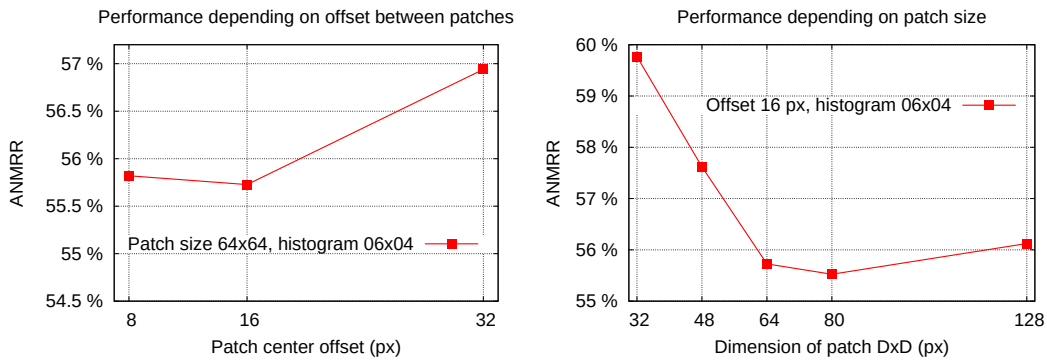


Figure 7.8: Experiments for choosing the patch size and offset between patches. The tuning was done using smaller descriptors, with histograms of dimensions 6×4 and histograms size 48, while the chosen patch size (80×80) and offset between patches (16 pixels, resulting in 64 pixels of overlap for the chosen patch size) are later used with larger histogram sizes.

information is tackled through using both the corrected noncompactness $NC(\cdot)$ (cf. Eq. 6.1) and Shannon entropy $\mathcal{H}(\cdot)$ (cf Eq. 6.2) as shape attributes. We chose to use 10 bins for the size (area) attribute, and 6 for the shape attribute. These parameters were chosen as the image patches in this satellite retrieval task are similar in size as some of the larger regions considered for description in the classification setup of Sec. 7.1, and we found using the same binning parameters performed well in both settings. As we calculate one GPS from a Min-tree and one from a Max-tree, this produces global descriptors of size 120. We also observe a further improvement when combining the descriptors from two shape attributes into a single descriptor of length 240 per image. We also report an improvement over the base performance when applying the GPS to each of RGB channels separately.

Local Pattern Spectra. Further, we attempt a single-scale local approach. The area $\mathcal{A}(\cdot)$

is again used for the size attribute, and corrected noncompactness $NC(\cdot)$ for the shape. We densely sample the image, calculating the LPS on regular rectangular patches over a grid on the image. Before determining the optimal patch size, we reexamine the number of bins used to form a LPS histogram as the size of the described regions is now even smaller compared to the GPS. Between 5 and 10 bins were considered for the size atribute, and between 3 and 6 for the shape. As the higher number of bins does not show significant improvement but makes the retrieval experiments slower, we chose a histogram size of 8×6 for performing final experiments. In order to improve the efficiency of running the tuning experiments regarding the patch and overlap size, and as all descriptor sizes show the same trends with the patch sizes tested, smaller histograms of size 6×4 were used for the tuning. Finally, different patch dimensions and offsets between patch centers are tried out, resulting in different overlap between patches (cf. Fig. 7.8 for the results of the tuning experiments). Our final choice of patch size is 80×80 with 16 pixels distance between the patch centers, on which a descriptor of size 96 is calculated based on two 8×6 histograms. We only report the results using NC as combination with Shannon entropy attribute $\mathcal{H}$ does not result in an improvement over using the area-noncompactness spectrum.

Finally, we attempt a multi-scale approach based on a pyramid of patches. Here, we start with patch size 32×32 and the size of patch increases for each level of the pyramid ($2\times$ along each dimension), so the scale-invariance of LPS becomes relevant. We report the results of this approach both with SV-LPS [34] as well as with choosing a common reference scale produce SI-LPS [32]. The distance between patch centers is again set to 16. The length of the descriptors is 96, the same as for the base local approach, but the number of descriptors is increased more than three-fold.

For both single-scale and pyramid approach using LPS, we use VLAD indexing to produce global image descriptors [86], using 8 cluster centers. We use a different subset of the ImageNet 2010 Validation set in every repetition of the experiment for building the visual vocabulary for VLAD, which is consequently formed both independent of the evaluation dataset as well as of its geographical context. The training sample contains 200 times more descriptors than the number of cluster centers used for building the vocabulary, randomly cosen from the descriptors extracted from 500 images of the ILSCRC2010 Validation dataset. The same approach to extracting descriptors as well as the same patch size (stopped at 256×256 for pyramid approaches) was used as for the evaluation dataset (which yields substantially more descriptors per image on the training set due to much larger image size).

Table 7.4: The retrieval performances of different local and global approaches on Merced dataset

approach	**ANMRR**
SIFT (on keypoints, [220])	0.601
dense SIFT ([146])	***0.4604*** (using VLAD)
global texture descriptors ([7])	0.575
local texture descriptors ([6])	0.585 (Bag of Words)
GPS - area $\mathcal{A}$ + noncompactness NC	0.579
GPS - area $\mathcal{A}$ + Shannon Entropy $\mathcal{H}$	0.670
GPS - both shape attributes ($NC + \mathcal{H}$)	**0.557**
GPS - RGB decomposition ($\mathcal{A} + NC$)	0.562
dense LPS (area $\mathcal{A}$ + noncompactness NC)	0.538
pyramid LPS (scale variant)	0.534
pyramid LPS (common scale 64×64)	**0.529**

7.2.3 Retrieval results

With our base GPS approach, we outperform both previously proposed global and local morphological approaches based on texture [7, 6], as well as the seminal SIFT approach on this dataset [220]. Combining two different shape attributes is the preferred technique for improving these base results while still working with global descriptors. It results in a bigger performance improvement than decomposing the image into channels and is suited for possible use on images with more than three channels. While the Shannon Entropy itself did not perform well outside of the combination, the improvement achieved indicated complementary properties to that of the noncompactness attribute. Further combinations with different shape attributes should be considered, as well as the possibility of using a different measure of size.

Further improvements are achieved by using a dense local approach, and that only by using 144 descriptors per image. It is interesting to note that with LPS, no improvement is achieved when descriptors based on both noncompactness and Shannon Entropy shape

parameters are used in conjunction. The same approach used with GPS descriptors was attempted, however with the LPS it did not increase the discriminative power of the combined descriptors. This could be explained by the noise introduced due to the discretization artefacts, which become more pronounced in this satellite image retrieval setup due to small histogram sizes in addition to small image patches on which the descriptors are calculated.

The final improvement comes from using the multiscale approach for selecting the local image patches on which the LPS descriptors are calculated. The patch dimensions are doubled in every layer of the pyramid, inducing calculation of LPS on a bigger scale. If the patch size is used directly as the scale parameter in LPS calculation, the descriptors of a single image are not at the same scale and thus not comparable. Thus, the SI-LPS approach is applied in combination with pyramidal patch selection to produce the best pattern spectra results of 52.9% ANMRR. The summary of these results can be found in Tab. 7.4.

While it still remains to outperform the dense SIFT descriptors [146], which produce state of the art results on the dataset, we show here an improvement over previous morphology-based approaches as well as the seminal SIFT approach to retrieval on this dataset. These experiments also validate the use of LPS descriptors in image retrieval and promise even more competitive results after considering the proposed improvemens to the descriptor presented hereafter.

7.3 Discussion and Perspectives

Depending on the size and shape attributes used, we can look at the components of the image used in histogram calculation as 2D continuous random variables. As such the pattern spectra can be seen as estimates of probability density functions (PDF) in histogram forms. Indeed, the probability density function describes a relative likelyhood for a random variable to take on a given value, and histograms are a basic, oldest form of density estimation [169]. Despite being widely used, histograms have several problems, including discontinuity and quantization effects, as well as dependance on both the choice of origin and the amount of smoothing (bin size) used in calculation. In addition to these effects being more prominent as the amount of data gets smaller (i.e. when transitioning from global to local pattern spectra), they also oblige the user to determine the parameters before using the pattern spectra as descriptors.

While on one hand, using machine learning methods to determine the optimal parameters optimally certainly is an option, the problem is only artificially alleviated as the problems related to quantization and discrete approaches are not mitigated. On the other hand, a different approach to construct a parameter-independant pattern spectra structure could

instead rely on statistics and moving towards more sofisticated approaches of estimating the underlying probability density function model from which the image component distribution was drawn. First step would be to storing the data in a way that does not lead to as much of an information loss. The options would range from using a binning that is purposefully too fine, using a mesh instead of a histogram (where the information could be divided across the points in a weighted fashion), or even an adaptive mesh or directly working with the (size, shape) coordinates of the components. Instead of using a histogram, a model could then be produced by clustering the data points, approximating the PDF with Gaussian mixture models or using other density estimators [169] such as kernel or variable kernel density estimation.

Finally, a way to compare the new PDF estimations will depend on the exact estimation method used to produce the model. In case of adaptive meshes, it might focus on determining the precise locations in an adaptive mesh where the comparison should be done, comparing the spatial and size distribution of the calculated cluster centers, or using statistical methods to determine how likely the estimated PDFs are to come from the same distribution, taking into account the sample size and confidence in the obtained estimates. Instead of working on PDFs, obtaining (estimated) cumulative distribution functions (CDF) would allow for a comparison using Kolmogorov–Smirnov Goodnes-of-Fit Test [44] to compare the two probability distributions.

Chapter Summary

The evaluation of LPS was presented in this chapter. Firstly, both the SI-LPS and SV-LPS are applied to general image domain in the context of image classification. The MSER detector [108] is used in the detection step to exploit both the fact that the LPS descriptors can be calculated on the same structure the detector works on, thus speeding up the descsriptor calculation, as well as the ability of the LPS descriptors to exploit shape information returned by the MSER detector. The performance of SI-LPS and SV-LPS descriptors is compared to the performance of SIFT (using a common approach of estimating the detected region with an ellipse to obtain a measurement region and thus ignoring the region shape) on different subsets of the *UCID* database, thus examining the performance of the descriptor in correlation with category size as well as the number of provided examples. A competitive performance is reached, also suggesting a higher tolerance for the number of database examples presented, with a descriptor only half the size of SIFT. Additionally, the scale invariance properties of the SI-LPS descriptor are confirmed by repeating the experiments on a manually resized sample of the database.

Secondly, both the GPS and LPS descriptors are evaluated in an image retrieval frame-

work focusing on remote sensing satellite data on the Merced dataset. Due to the nature of the image samples comprising the dataset, a dense sampling method is used to predetermine the image patches for calculation when the LPS descriptors are used. Finally, to extend this dense sampling method to operate on multiple scales, a pyramid approach is used where sampling grids of decreasing resolution are used to select the predetermined patches. In this final approach, the SI-LPS descriptors are used as calculating all the pyramid descriptors on the same scale results in the best performance. Even though the LPS descriptors still remain to reach the similar dense approach using SIFT descriptors, we show an improvement in performance over all other morphology-based methods applied to this dataset.

Lastly, an approach eliminating the LPS parameters is proposed, emerging from the usage of histograms. This potential research direction would be a step towards solving the discretization problems by using a continuous representation to summarize the size-shape signature of the image.

This chapter concludes the presentation of component based techniques specifically aimed at image retrieval. However, while the tree simplification technique presented in the next chapter can be applied to various application domains as a preprocessing step, it also opens the possibility of specifically adapting the hierarchy to the image domain or other known properties of the database in either of the previously presented image retrieval approaches.

Chapter 8

Complexity Driven Tree Simplification

Contents

A common property among all tree representations is that the leaf nodes represent the fine image structures, increasing in complexity with proximity to the root. The *coarseness inherent to the representation* could be defined as a distance from the node to the root of the tree, or using a more sophisticated method, such as indexing based on the tree construction presented in Chap. 3. This inherent coarseness and even the levels assigned by indexing the hierarchy *do not, in the general case, accurately reflect the region complexity* (e.g. the chaining effect in the α-trees, cf. Sec. 3.4) and can not be used to compare any two regions. But, if some coarseness measure for the objects of interest is known prior to main image analysis step, the relevant search space could be limited to structures with a similar level of coarseness.

The transformation presented hereafter assigns an external coarseness measure to all the nodes and rearranges them accordingly while preserving the hierarchical relations. New coarseness measure is chosen among increasing attributes on the tree, reflecting that the complexity of regions increases along each branch even if it can not be directly compared. The nodes of the same coarseness are pruned and at most one region of a certain coarseness per branch is kept. The result is a representation where the node levels correspond to the coarseness of the regions represented by the nodes and every tree level comprises only

nodes of the same coarseness. This in turn enables limiting the search space when dealing with objects whose coarseness can be estimated by directly accessing only the regions of the relevant coarseness. Additionally, the search space is reduced even for objects with unknown coarseness, as the number of regions after the transformation can only decrease. This property makes the transformation suited for processing hierarchies that are too fine before the image analysis step.

Imposing constraints on components of partitioning hierarchy in a way that the hierarchical relations between the remaining components are preserved was first explored in [176]. A hierarchy obtained after imposing such constraints then contains *constrained components* and is referred to as a *constrained connectivity hierarchy*. The work in [176] only provides the definitions of constrained components and the potential applications while the algorithm for selecting such constrained components is not proposed. The approaches for computing a constrained connectivity hierarchy presented in [142, 130] were demonstrated on α-trees with the goal of mitigating the problems caused by the chaining effect [178, 177]. However, the concept of constrained connectivity introduced by Soille [173] is only directly applicable to the partitioning hierarchy, where the component range constraint can be viewed as an external coarseness measure. In [142], the approach to extract just one level of the (ω)-hierarchy at a time is presented, and provided inspiration for the approach presented herein. They also rely on a bottom-up approach for implementing an attribute filtering, but stop the tree traversal as soon as the attribute values are above the chosen threshold, effectively only using the leaf nodes of the filtered hierarchy. In [130], the hierarchies are represented as ultrametric watersheds, and they propose an approach to calculate the ultrametric watershed representation of a new hierarchy by imposing an increasing constraint on the initial hierarchical segmentation. However, while the complexity of the approach proposed in [130] is the same as the complexity of the transformation presented herein, their approach is only applicable to partitioning trees.

Section 8.1 explains the conditions and assumptions about the hierarchy and the coarseness measure used. The effects of the proposed transformation, the algorithm and the estimation of the algorithm complexity are presented in Sec. 8.2. The Chapter is concluded by summarizing the advantages and potential application of the presented technique.

8.1 Premises of the Algorithm

When constructing the algorithm, we will presume that the tree is constructed *with no redundancies*, i.e. no two nodes represent the same region of the image I (the term is also used in [142] in the context of α-trees). The example of the same tree shown with and without redundancies is displayed in Fig. 8.1. Instead of just a level, we assign a *level range* $[lMin, lMax)$ to

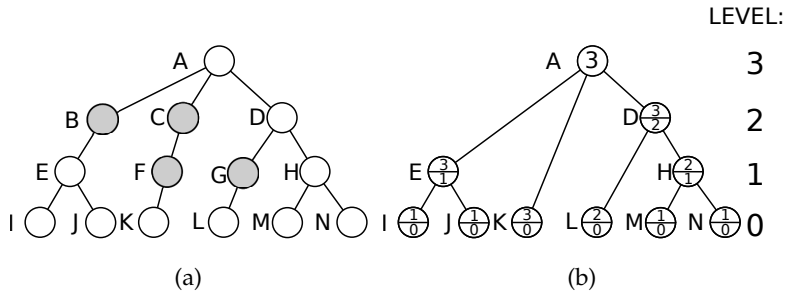


Figure 8.1: The tree shown in (a) has redundant nodes, marked in gray. The levels of the nodes are displayed on the right of the trees. By removing the redundancies we get the tree shown in (b) (the level range $[lMin, lMax)$ is displayed inside the nodes).

the node. *lMin* and *lMax* are then defined as the lowest levels in the tree on which the node does and does not appear on, respectively. Due to the implementation of the construction process in the case of partitioning trees (especially α-tree and the (ω)-tree, cf. Secs. 3.4 and 3.5), the same region may appear multiple times in the hierarchy. Instead, we will simply include multiple levels in the level range of the node, without duplicating the node. None of the inclusion tree construction algorithms produce trees with redundancies.

The second condition pertains to the attribute $K(\cdot)$ used as a new coarseness measure for regions. An attribute $K(\cdot)$ chosen as the new coarseness measure must be an increasing attribute, and the algorithm assumes that the values of this increasing attribute were assigned to the nodes of the tree before the transformation. Many interesting attributes (e.g. intensity range, component area) can be assigned to nodes directly during tree construction. A discussion on increasing attributes can be found in Sec. 6.3.1, but the final choice will always depend on the intended application and known properties of object of interest and image domain.

8.2 The Simplification Technique

We now present the algorithm for imposing an external coarseness measure on a tree representation of an image without redundancies. The output is also a tree representation with no redundancies, whose levels comprise nodes with the same value of the coarseness measure.

Transformation results can be interpreted as a hierarchy formed by stacking, for threshold values ranging from zero to maximal value of the attribute, the leaf nodes of trees obtained by performing attribute filtering (cf. Sec. 6.3.1) on the original tree with an increasing attribute. The result is a tree representation of this hierarchy with no redundancies. A node present as a leaf in the hierarchy after an attribute filtering with a threshold t will have t

included in its level range in the result. This is very similar to storing the results of a granulometry (cf. Sec. 6.3.2), where the finite set of sieve sizes corresponds to the set of attribute values present in the hierarchy. However, in contrast to granulometry or pattern spectra, where only a single numerical measure of the amount of remaining or removed content is noted for a single filtering step, we propose to store the results after applying each size filter. The attribute, or criterion, used to produce a granulometry will be assigned to the tree nodes as the new coarseness measure. After the transformation, the tree cuts stacked to produce a new tree can be directly accessed. This definition extends easily to attributes that take continuous values, where the node can belong to a continuous range of levels.

The algorithm presented here can be compared to the direct rule of simplifying the tree with a non-increasing criterion (cf. Sec. 6.3.1). The criterion is based on the chosen increasing attribute $K(\cdot)$ but the condition for keeping the node is that the attribute value assigned to it is strictly smaller than that of its parent.

Under the assumptions put forward in Sec. 8.1, the algorithm can be described in very simple terms: in a bottom-up traversal of the tree, if we discover a node with an attribute value equal to the attribute value of its parent, we should add all the child-nodes of this node to the children set of its parent, and then delete the node. This is summarized in Algorithm 1. The attribute value assigned to a node in the original tree becomes the minimal level of that node if the node is kept after the transformation. The tree before and after the transformation is shown in Fig. 8.2(a) and 8.2(b).

```
1 function rearrangeTree(Node):
2     foreach Child ∈ Node.children do
3         rearrangeTree(Child)
4     if Node.attributeValue = Node.parent.attributeValue then
5         add Node.children to Node.parent.children
6         delete Node
7     else
8         Node.minLevel ← Node.attributeValue
9         Node.maxLevel ← Node.parent.attributeValue
```

Algorithm 1: The proposed transformation

If the tree is stored in the straightforward way, the memory requirements are proportional to number of image pixels (cf. Sec 8.2.1). Highest cut of the tree comprising nodes with coarseness lower or equal to the desired level is then selected by performing a top-down traversal of the tree and keeping the first node in each branch with satisfying coarse-

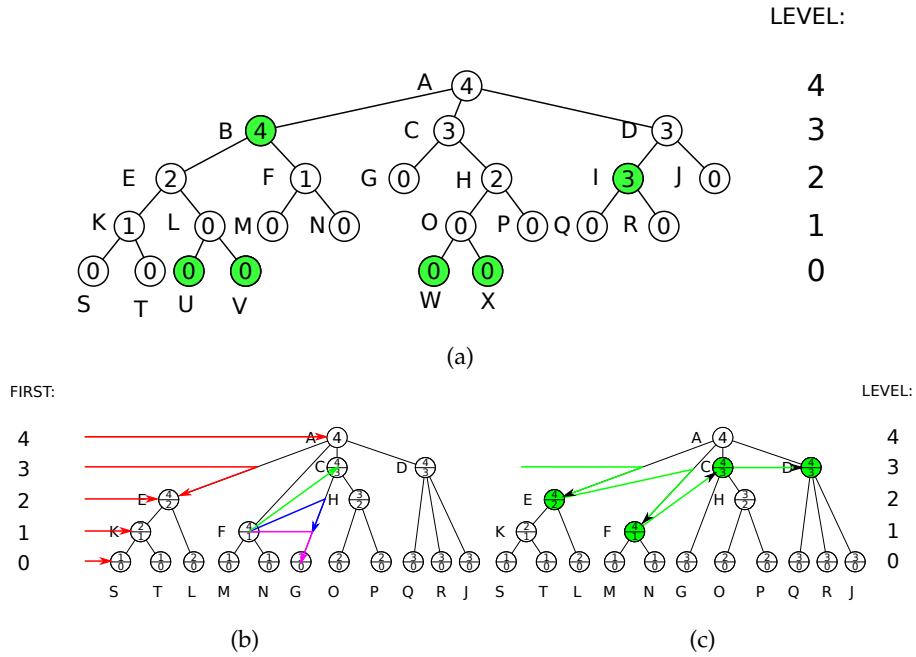


Figure 8.2: Subfigure (a) shows the original tree before the transformation, with attribute values displayed in the nodes and the nodes to be removed highlighted in green. In both subfigures (b) and (c), the level range is displayed inside the nodes. Subfigure (b) shows pointers to beginning of every tree level, needed for direct access to levels. Entries like the one shown for node F keep the next node for every level $[1, 4\rangle$ (level 1: G (purple), 2: G (blue), 3: C (green)) and need to be stored for every node. Subfigure (c) shows accessing the third level of the tree using the stored pointers.

ness level. Memory requirements rise if we want faster access. For each level of the tree we store a pointer to the left-most node and, for each node and each level in the level range of the node, the first next node at that level in the tree. Figures 8.2(b) and 8.2(c) illustrate the information which needs to be stored to enable direct access to any tree level. Once the first node in a level of a tree is accessed, the pointers to the next nodes can be followed to access all the nodes of that level.

8.2.1 Complexity Analysis

From the pseudocode, it is visible that the algorithm is linear in the number of nodes in the tree. The maximum number of nodes in the partitioning tree is achieved if all image pixels are used as the initial partition. As every node has at least 2 distinct child nodes, the number of inner nodes is less or equal than that of a binary tree with the same number of leaves, and never exceeds the number of leaves in the tree. This makes the maximum number of

nodes in any partitioning tree lower than $2N$, where N is the number of image pixels. The maximum number of nodes in an inclusion tree is achieved if every node adds only 1 new pixels in order to represent a new region, and is no higher than N.

Considering that the number of nodes in any tree image representation without redundancies is linear in the number of image pixels and the transformation algorithm is linear in the number of tree nodes, we can conclude that the complexity of the algorithm is linear in the number of image pixels, $O(N)$. The overall complexity of producing such a transformed tree from the original image depends on the choice of the original underlying tree, the complexity of the construction algorithm for the chosen tree type and additional costs (if any) of calculating node attribute values.

8.3 Proposed Applications

In image segmentation, regions of a hierarchical image partition are treated as "puzzle pieces" [176, 173] used to compose a segmentation. The transformation reduces hierarchy size, lowering the number of "puzzle choices" and simplifying the calculation of the segmentation. Binary partition trees used for object detection [202] generate partitions so fine that a second merging criterion is used in order to generate a coarse partition in which the potential detections are marked before the object can be detected in the fine parts of the hierarchy. By reducing the size of the hierarchy, better detection could be achieved by using more complex algorithms or a more exhaustive search.

In the domain of inclusion trees, many applications would benefit of the reduction in the search space. Finding the k most prominent structures in an image (cf. [131]) depends directly on the size of the tree. The proposed simplification technique is equally applicable to both types of hierarchy, resembling in effect the simplification techniques for partitioning trees [176, 130]. The image simplification technique using Trees of Shapes [120], based on area size, can also be applied to a hierarchy first simplified using a different coarseness attribute. An image comparison method proposed in [120] relies on assigning attributes to describe the regions of the hierarchy and then checking one of the images for presence of shapes similar to shapes present in the other image. Since the method is already working by finding *similar* (and not the same) shapes, a simplification of the hierarchies before image comparison would reduce the overall number of comparisons and speed up the process. An approach to image retrieval relying on examining the image structural elements corresponding to the nodes of the tree [182] would also benefit from the reduction in hierarchy size. As the approach to background detection presented in [183] depends on the values of several thresholds, multiple precision results could be obtained simultaneously by applying the presented transformation instead of a simple tree filtering only.

Additionally, this preprocessing step could be applied to the feature detection method presented in Chap. 4 and 5 in order to either filter out the tree for faster detection, or change the levels of the tree in order to improve the properties of the stability function. As the technique also resembles a 1D granulometry, processing a tree with an attribute before calculating the pattern spectra (cf. Chap. 6 and 7) could impose the characteristics of the increasing attribute used as the coarseness measure on the hierarchy on the LPS. This could be studied as a way to introduce beneficial effects of a third attribute to the calculation of the 2D pattern spectra descriptors (typically based on a single size and shape attribute).

Chapter Summary

Applying additional constraints on a partitioning tree of an image [176, 142, 130] was previously considered by varying the constraint threshold parameters to control the degree of image simplification. When the node level does not coincide with the perceived complexity of the represented region (e.g. the chaining effect in α-trees), applying constraints rearranges the hierarchy according to a more precise external coarseness measure [176, 142]. A simple bottom-up technique was presented, applicable both to the partitioning as well as inclusion trees. It imposes an additional constraint to the hierarchy based on an increasing attribute proposed as a new coarseness measure of the regions represented by the component tree. The results can also be interpreted as performing an attribute filtering with all threshold values simultaneously and storing all the results within a same structure [142], making it similar to a granulometry.

Additionally, several approaches from different application domains where the proposed simplification could be applied were proposed. The relation between the proposed simplification and a granulometry, as well as combining the two operations is an interesting possibility for further examination. Additionally, application of this technique as a preprocessing technique in combination with description and detection techniques presented herein remains to be explored.

Chapter 9

Conclusions and Perspectives

Contents

In this, final, chapter, we conclude this manuscript. The following section summarizes the contributions put forward in the thesis and offers a comprehensive discussion of the concepts presented herein. Sec. 9.2 offers several interesting applications of component trees to image retrieval as potential directions for future, divided into short and long term perspectives. Finally, Sec. 9.3 looks into open problems on component tree hierarchies outside of the domain of image retrieval and from a more general perspective.

9.1 Conclusions

This thesis attempted to tackle image retrieval tasks using various component trees from mathematical morphology. The component trees are constructed in a way to represent structures and objects present in the images being processed, as well as provide the information about their spatial relations across multiple scales. This, as well as previous successful application of component trees and other mathematical morphology concepts to image retrieval tasks (i.e. [136, 190, 55, 9, 6]), convinced us to attempt constructing novel approaches which would exploit the good properties of the morphology-based image hierarchies.

As general techniques, applicable to a wide range of structures (i.e. hierarchies in our case), are always of more interest than the ones designed with specific constrictive requirements in mind, we first begin by presenting the component trees from a generalized point of view in Chap. 2. This chapter offers the traditional formalization of component tree hierarchies as well as a novel formalization based on Stackable Hierarchies of Regions (SHoR) which reflects the fact that these hierarchies comprise either image segmentations or partial segmentations which start from fine detail but progress into coarser and coarser image approximations. Based on this general tree formalization, a distinction between two superclasses of partitioning and inclusion trees in proposed. Finally, indexing is explained as assigning a measure or level of aggregation to each element of the hierarchy, and the dendrogram framework used to visualize indexed partitioning trees is extended to reduced dendrograms in order to represent the inclusion trees as well. Following, in Chap. 3, different trees from both superclasses were presented, with a focus on their properties and the types of regions they represent. For each tree, an indexing method is offered based on their definition or construction algorithm. An overview of seminal and state-of-the-art construction algorithms is also presented for each tree, with paying special attention to the BPT construction algorithms which lack a consistent analysis in the literature.

The main part of the thesis applies the presented hierarchies to feature detection and feature description tasks from image retrieval. Chapter 4 extends the tree-based MSER detection algorithm [136] in an attempt to construct detectors based on other trees and exploit their properties. Replacing the Min and Max-trees in the MSER construction with a different component tree corresponds to changing the underlying connectivity originally used to process the image. The direct consequence of this is that the stability of the detected regions, depending on the region contrast in the original approach [108], is now related to different measures reflecting the tree structure used as well as the between-node distance defined for the detector (in case of ToS-MSR) or the ultrametric distance associated to the tree (in case of α-MSR and (ω)-MSR). All the three proposed detectors are evaluated in Chap. 5 in the image matching framework of Mikolajczyk et al. [116], which uses 8 small datasets of 6 images of the same scene each to examine the detector behavior depending on the scene type as well as the image transformation type introduced by each dataset. While the α-MSR and (ω)-MSR show some interesting properties on certain types of scenes (i.e. strongly structured scenes with clear edges, ideally resembling cartoon drawings), the ToS-MSR detector exhibits good performance over all the framework datasets. Based on these preliminary results, the ToS-MSR detector is also tested in an image retrieval setup, outperforming the MSER detector in terms of mAP on both chosen datasets.

After feature detection, we focus on feature and region description of the detected regions. This work is based on granulometries and pattern spectra, presented in Chap. 6.

Starting from the global pattern spectra, previously used as global image descriptors in retrieval and classification [195, 190], and motivated by a previously existing adaptation of pattern spectra to the pixel-scale descriptors (DMP [21] and DAP [22, 55, 141]), we study the challenges faced when calculating the pattern spectra on local image patches. While the rotation and translation invariance properties of the global pattern spectra are kept in a straightforward manner, keeping the scale invariance property requires the introduction of a common scale parameter *RS* which also leads to some loss of information. Thus, two different versions of the LPS descriptors were introduced, namely SI-LPS which keeps all three inherent invariances of the GPS (namely, rotation, translation and scale invariance), and SV-LPS which is only rotation and translation invariant. Special attention is given to the case when the LPS descriptors are used in combination with MSER detector [108], as both algorithms utilize the Min and Max-trees in their execution and allow for optimization of computation and consequently a speedup in the retrieval system. As with feature detection, the proposed feature description method was also evaluated in two distinct frameworks. The preliminary framework focuses on image classification and compares the LPS descriptors to SIFT, while taking into account the database size, number of categories and number of examples per category. A competitive performance with SIFT descriptors is achieved on all the categories with a satisfactory number of examples per category, while using LPS descriptors of only half the size of SIFT. Additionally, the scale invariance property of SI-LPS is confirmed on specifically designed experiments which included manual resizing of the database images. Finally, the descriptors are also examined in the context of satellite image retrieval, where the feature detection step is replaced by dense sampling which predetermines the image patches used for descriptor calculation. While the LPS approach still remains to outperform the dense SIFT approach [146] on the UC Merced Land Use Dataset [220], we do outperform other morphology-based descriptors on the dataset [6, 7]. The proposed LPS descriptors still remain smaller and faster to calculate than the SIFT descriptors, which also benefits the latter steps of the retrieval process due to handling shorter descriptor vectors by the indexing methods.

9.2 Perspectives in Image Retrieval

In this section, we propose several interesting directions for the continuation of the work presented in this thesis. First we summarize various direct improvements to the methods proposed throughout the chapter conclusions. We then offer a selection of possible approaches with applications in image retrieval, and more broady image processing, inspired by the work presented herein as potential topics for research.

9.2.1 Improvements to the Proposed Methods

With MSR detectors, the stability of the detected regions is influenced by two factors: the underlying component tree which holds the potential detections, as well as the distance function defined between the elements of the hierarchy. While a specific distance function was proposed for the ToS-MSR detector, the α-MSR and (ω)-MSR detectors use the ultrametric distances defined for the corresponding trees. Using more advanced distances could improve the detections obtained from the partitioning trees, as well as provide more fine-grained control over the set of accepted detections using the detector parameters. A preprocessing step proposed in Chap. 8 could also be used to modify either of the hierarchies and change the behavior of the stability function on the remaining regions. The behavior of the α-MSR and (ω)-MSR detectors suggests using the α-tree as the initial quality of the detected regions is higher, while factoring the global range parameter normally used for the (ω)-tree into the distance function defined between the hierarchy elements to improve the quality and quantity of the selected regions. Additionally, further studying the specific properties of the regions returned by the partitioning tree detectors could point towards a specific application domain or a different type of images on which using these detectors would be beneficial.

The LPS descriptors could also be calculated on other hierarchies (like in [142] where global pattern spectra were calculated on α-trees). However, some hierarchy-specific challenges arise when exchanging the used hierarchy. If a hierarchy is self-dual like with the Tree of Shapes, the regions contained in the tree will correspond to objects both darker and lighter than the background. As the contribution of a component to the pattern spectrum bin is weighted by region contrast, directly applying the calculation algorithm could result in negative or overriding bin contributions from different components. For this reason, special attention should be payed to component contributions to the bins, or a separate light and dark pattern spectrum could be built from a single tree. As different hierarchies provide different hierarchy-specific challenges in descriptor construction, it would be of interest to define the precise conditions which need to be met to define a pattern spectrum on an arbitrary component tree. As we are working with histograms, the L_1 distance used thus far is not the best suited for their comparison. A fast way to improve LPS would be to replace the L_1 distance used to compare the descriptors with a different distance, or even a divergence to use as a similarity measure. Additionally, the rootSIFT [12] approach could be considered if the norm of the LPS descriptor is stored separately (in their current form, the LPS descriptors are not normalized as the amount of content carries descriptive information). Finally, as the current form of pattern spectrum features as many as 5 distinct parameters, Machine learning techniques could be employed to determine the optimal parameters of the method to improve performance and potentially even further shorten the descriptor length.

Finally, in addition to all the potential applications listed for the proposed tree simplification technique, its relation to the granulometries as well as pattern spectra should be interesting for further examination.

9.2.2 A Step Further

Hierarchical Image Indexing. It would be of particular interest to better exploit the hierarchical and spatial relations between the image components present in any component tree hierarchy. As most trees are too big to be compared directly efficiently, exploring the hierarchical relations between MSR detections seems like a viable compromise. Most aggregation and indexing methods are not designed to work with hierarchically organized detections. However, the spatial and inclusion information about the detections could prove beneficial to identifying the image content. As the MSR detections are organized into much smaller hierarchies, tree-comparison methods such as the subpath-based kernel presented in [53] could be used to compare the hierarchies. The tree similarity kernel presented in [53] returns a similarity measure calculated directly between any two hierarchies, taking into account the different subpaths of the two trees. Additionally, this method requires a similarity measure to be defined between any two nodes of either of the trees being compared (which is then used for subpath comparison). Thus, any feature descriptor could easily be associated to the elements of the hierarchy and used by this technique in order to compare the hierarchy similarity. As such, the reduced hierarchies of MSRs themselves would become global image descriptors in the proposed indexing method.

Continuous Pattern Spectra. An interesting direction to take with pattern spectra would be transitioning from the discrete representation of image content through histograms and moving towards a continuous representation (as already mentioned at the end of Chap. 6). Histograms can be viewed as a simplest form of probability density estimation. However, due to using a predetermined binning, it suffers from quantizaion effects which only become more prominent when the sample and histogram size decreases, as when transitioning from global to local descriptors. The goal here would be approximating the PDF of the shape-size distribution of the image content using statistical methods, such as Gaussian mixture models. The comparison between such descriptors could then be expressesed as likelihood for the two estimates to come from the same original distribution.

Texture Feature Detection. If a similarity measure between image pixels can be established based on texture, texture-based component trees can be built. While the inclusion trees require a strict ordering between all image pixels, the partitioning trees could be considered even if just a distance between pixels is provided. Such trees could then be used for texture detection or segmentation, or for extracting texture-based features from image. Such

features could be considered for description and retrieval of images featuring prominent textured areas.

Improving Region Quality in Selective Search. Even though the application of Binary Partition Tree to any specific retrieval applications was not covered herein, the user ability to define the initial partition and region similarity measure used for the merging criterion allows it to represent the most complex, and accurate, regions out of all the presented component trees. As the selective search [193] technique relies on a similar hierarchy constructed bottom-up from an initial oversegmentation, its application to tasks such as object recognition and semantic segmentation [193, 38] could benefit from more quality region proposals produced by the BPT approach. The quality of the BPT regions (i.e. the region borders and precision of segmentation throughout the hierarchy) could be further improved in a preprocessing step [102] before the hierarchy is used in selective search.

9.3 Open Challenges on Component Trees

In this section, we identify some open challenges pertaining to component trees in general not discussed herein, which would provide viable directions for future research.

This thesis focuses mainly on monochannel images, but as the examined hierarchical structures have a primary goal of providing good estimated locations of object contained in the image, the color (and any other kind of) information contained in multichannel images has to be considered. For a general partitioning tree, working with vectorial instead of scalar image element values is fairly easy, as the total order between the values of image elements is not required. As such, the BPT was originally defined with color in mind [158], and the trend of including color information has persisted in current literature [100, 202]. For the α-tree, several adaptations for multichannel images exist [221, 173, 178, 10, 112].

However, for the inclusion trees, where a total ordering of image element values is required, the extension to multivariate data is not straightforward. Investigation of the Max-trees and Min-trees for multivariate data was done by Naegel and Passat [127] in the context of connected filtering, and later by Perret et al. [148] for more general applications. An algorithm for extracting distinguished features from color images, defined by Forssén [68], also indirectly defines a Max-tree and Min-tree structure for color images. It was suggested that extensions similar to those applied for the Max-tree [68] could be applied to the Tree of Shapes as well [43]. However, the extensions of the Tree of Shapes have only recently been explored [41, 42].

Other open problems include finding the solution to the chaining effect present in α-trees (cf. Sec 3.4). This led to defining logical predicate connectivity [176] and constrained

connectivity [173, 178, 174, 177, 135] as the simplest attempts at a solution, which resulted in establishing the (ω)-tree (explained in Subsec. 3.5) as the most widely-used constrained connectivity hierarchy. More recently, masked-based connectivities and hyperconnectivities, as well as the advanced hierarchies based on them [210, 149], were introduced as a way to handle both visually disconnected object and overlapping objects in images.

An interesting challenge would also be formalizing the theoretical relations between various hierarchies. These relations were only briefly mentioned in this manuscript as they are implied by the construction algorithms in Chap. 3. Examples include building the α-trees as the Min-tree of the edges [130, 77], and using the Max-tree algorithm as the canvas for the Tree of Shapes construction algorithm [73]. Relations between some other hierarchies have also been explicated in [51], but relations between all the presented hierarchies are still unknown and merit further examination.

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.11 (2012), pp. 2274–2282.

[2] A. Al-Dujaili, F. Merciol, and S. Lefèvre. "GraphBPT: An efficient hierarchical data structure for image representation and probabilistic inference". In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2015, pp. 301–312.

[3] A. Alahi, R. Ortiz, and P. Vandergheynst. "Freak: Fast retina keypoint". In: *Computer Vision and Pattern Recognition (CVPR) 2012, IEEE Conference on*. 2012, pp. 510–517.

[4] P. F. Alcantarilla, A. Bartoli, and A. J Davison. "KAZE features". In: *Computer Vision–ECCV 2012*. Springer, 2012, pp. 214–227.

[5] P. F. Alcantarilla, J. Nuevo, and A. Bartoli. "Fast explicit diffusion for accelerated features in nonlinear scale spaces". In: *Proc. Of British Machine Vision Conference (BMVC 2013)* (2013).

[6] E. Aptoula. "Bag of morphological words for content-based geographical retrieval". In: *International Workshop on Content-Based Multimedia Indexing (CBMI)*. 2014.

[7] E. Aptoula. "Remote Sensing Image Retrieval with Global Morphological Texture Descriptors". In: *Geoscience and Remote Sensing, IEEE Transactions on* 52.5 (2014), pp. 3023–3034.

[8] E Aptoula. "The impact of multivariate quasi-flat zones on the morphological description of hyperspectral images". In: *International Journal of Remote Sensing* 35.10 (2014), pp. 3482–3498.

[9] E. Aptoula and S. Lefèvre. "Morphological description of color images for content-based retrieval". In: *Image Processing, IEEE Transactions on* 18.11 (Nov. 2009), pp. 2505–2517.

[10] E. Aptoula, J. Weber, and S. Lefèvre. "Vectorial quasi-flat zones for color image simplification". In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2013, pp. 231–242.

[11] R. Arandjelović and A. Zisserman. "Extremely low bit-rate nearest neighbor search using a Set Compression Tree". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 36.12 (2014), pp. 2396–2406.

[12] R. Arandjelović and A. Zisserman. "Three things everyone should know to improve object retrieval". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 2911–2918.

[13] R. Audigier and R. Lotufo. "Seed-relative segmentation robustness of watershed and fuzzy connectedness approaches". In: *Computer Graphics and Image Processing, 2007. SIBGRAPI 2007. XX Brazilian Symposium on*. IEEE. 2007, pp. 61–70.

[14] Y. Avrithis and K. Rapantzikos. "The medial feature detector: Stable regions from image boundaries". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1724–1731.

[15] A. Babenko and V. Lempitsky. "The inverted multi-index". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 3069–3076.

[16] Reza Bahmanyar, Shiyong Cui, and Mihai Datcu. "A Comparative Study of Bag-of-Words and Bag-of-Topics Models of EO Image Patches". In: *Geoscience and Remote Sensing Letters, IEEE* 12.6 (2015), pp. 1357–1361.

[17] C. Ballester, V. Caselles, and P. Monasse. "The tree of shapes of an image". In: *ESAIM: Control, Optimisation and Calculus of Variations* 9 (2003), pp. 1–18.

[18] A. Baumberg. "Reliable feature matching across widely separated views". In: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. Vol. 1. IEEE. 2000, pp. 774–781.

[19] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. "Speeded-up robust features (SURF)". In: *Computer Vision and Image Understanding* 110.3 (2008), pp. 346–359.

[20] M. A. Bender and M. Farach-Colton. "The LCA problem revisited". In: *LATIN 2000: Theoretical Informatics*. Springer, 2000, pp. 88–94.

[21] J. A. Benediktsson, M. Pesaresi, and K. Arnason. "Classification and Feature Extraction for Remote Sensing Images from Urban Areas based on Morphological Transformations". In: *Geoscience and Remote Sensing, IEEE Transactions on* 41.9 (2003), pp. 1940–1949.

[22] J. A. Benediktsson, L. Bruzzone, J. Chanussot, M. Dalla Mura, P. Salembier, and S. Valero. "Hierarchical Analysis of Remote Sensing Data: Morphological Attribute Profiles and Binary Partition Trees". In: *Mathematical Morphology and Its Applications to Image and Signal Processing*. Springer, 2011, pp. 306–319.

[23] C. Berger, T. Géraud, R. Levillain, N. Widynski, A. Baillard, and E. Bertin. "Effective component tree computation with application to pattern recognition in astronomical imaging". In: *Image Processing, 2007. ICIP 2007. IEEE International Conference on*. Vol. 4. IEEE. 2007, pp. IV–41.

[24] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. "When is "nearest neighbor" meaningful?" In: *Database Theory – ICDT'99*. Springer, 1999, pp. 217–235.

[25] A Bhattachayya. "On a measure of divergence between two statistical population defined by their population distributions". In: *Bulletin Calcutta Mathematical Society* 35 (1943), pp. 99–109.

[26] L. Bo and C. Sminchisescu. "Efficient Match Kernels between Sets of Features for Visual Recognition". In: *Advances in neural information processing systems (NIPS)*. 2009, pp. 135–143.

[27] C. Böhm, S. Berchtold, and D. A. Keim. "Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases". In: *ACM Computing Surveys (CSUR)* 33.3 (2001), pp. 322–373.

[28] P. Bosilj, E. Kijak, and S. Lefèvre. "Beyond MSER: Maximally Stable Regions using Tree of Shapes". In: *Proc. Of British Machine Vision Conference (BMVC 2015)*. 2015.

[29] P. Bosilj, E. Kijak, and S. Lefèvre. "Partition and Inclusion Hierarchies of Images: A Comprehensive Survey". In: *Image Processing (TIP), IEEE Transactions on* (2015). (under review).

[30] P. Bosilj, S. Lefèvre, and E. Kijak. "Hierarchical Image Representation Simplification Driven by Region Complexity". In: *Image Analysis and Processing–ICIAP 2013*. Springer, 2013, pp. 562–571.

[31] P. Bosilj, M. H. F. Wilkinson, E. Kijak, and S. Lefèvre. "Local 2D Pattern Spectra as Connected Region Descriptors". In: *Mathematical Morphology – Theory and Applications (Special Issue on Contributions from the 12th International Symposium on Mathematical Morphology)* (2015). (submitted).

[32] P. Bosilj, M. H. F. Wilkinson, E. Kijak, and S. Lefèvre. "Local 2D Pattern Spectra as Connected Region Descriptors". In: *Mathematical Morphology and Its Applications to Signal and Image Processing (ISMM)*. Vol. 9082. Lecture Notes in Computer Science. Springer, 2015, pp. 182–193.

[33] P. Bosilj, E. Aptoula, S. Lefèvre, and E. Kijak. "Satellite Image Retrieval with Pattern Spectra Descriptors". In: *2015 Conference on Image Information Mining : Earth Observation meets Multimedia (IIM)*. 2015.

[34] P. Bosilj, E. Kijak, M. H. F. Wilkinson, and S. Lefèvre. "Short Local Descriptors from 2D connected Pattern Spectra". In: *Image Processing (ICIP), IEEE International Conference on*. 2015.

[35] A. C. Bovik, M. Clark, and W. S. Geisler. "Multichannel Texture Analysis Using Localized Spatial Filters". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 12.1 (1990), pp. 55–73.

[36] G. Bradski. *OpenCV Library*. 2000.

[37] E. J. Breen and R. Jones. "Attribute openings, thinnings, and granulometries". In: *Computer Vision and Image Understanding* 64.3 (1996), pp. 377–389.

[38] H. Caesar, J. R. R Uijlings, and V. Ferrari. "Joint Calibration for Semantic Segmentation". In: *26th British Machine Vision Conference (BMVC)* (2015).

[39] J. Cardelino, G. Randall, M. Bertalmio, and V. Caselles. "Region based segmentation using the tree of shapes". In: *Image Processing, 2006 IEEE International Conference on*. IEEE. 2006, pp. 2421–2424.

[40] E. Carlinet and T. Géraud. "A comparison of many max-tree computation algorithms". In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2013, pp. 73–85.

[41] E. Carlinet and T. Géraud. "Getting a morphological tree of shapes for multivariate images: paths, traps, and pitfalls". In: *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 615–619.

[42] Carlinet, E. and Géraud, T. "A Color Tree of Shapes with Illustrations on Filtering, Simplification, and Segmentation". In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2015, pp. 363–374.

[43] V. Caselles and P. Monasse. *Geometric description of images as topographic maps*. Springer Publishing Company, 2009.

[44] I. M. Chakravarti and R. G. Laha. "Handbook of methods of applied statistics". In: *Handbook of methods of applied statistics*. Vol. 1. John Wiley & Sons, 1967.

[45] S. Chatzichristofis and Y. S. Boutalis. "Fcth: Fuzzy color and texture histogram-a low level feature for accurate image retrieval". In: *Image Analysis for Multimedia Interactive Services, 2008. WIAMIS'08. Ninth International Workshop on*. IEEE. 2008, pp. 191–196.

[46] H. Chen, S. S. Tsai, G. Schroth, D. M. Chen, R. Grzeszczuk, and B Girod. "Robust text detection in natural images with edge-enhanced maximally stable extremal regions". In: *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE. 2011, pp. 2609–2612.

[47] K. L. Chung and P. C. Chen. "An efficient algorithm for computing moments on a block representation of a grey-scale image". In: *Pattern Recognition* 38.12 (2005), pp. 2578–2586.

[48] K. L. Chung and J. G. Wu. "Improved Image Compression using S-Tree and Shading Approach". In: *Communications, IEEE Transactions on* 48.5 (2000), pp. 748–751.

[49] D. Coomans and D. Luc. Massart. "Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-Nearest neighbour classification by using alternative voting rules". In: *Analytica Chimica Acta* 136 (1982), pp. 15–27.

[50] J. Cousty and L. Najman. "Incremental Algorithm for Hierarchical Minimum Spanning Forests and Saliency of Watershed Cuts". In: *Mathematical Morphology and Its Applications to Image and Signal Processing*. Springer, 2011, pp. 272–283.

[51] J. Cousty, L. Najman, and B. Perret. "Constructive Links between Some Morphological Hierarchies on Edge-Weighted Graphs". In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2013, pp. 86–97.

[52] J. Cousty, G. Bertrand, L. Najman, and M. Couprie. "Watershed Cuts: Minimum Spanning Forests and the Drop of Water Principle". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 31.8 (2009), pp. 1362–1374.

[53] Y. Cui, L. Chapel, and S. Lefèvre. "A subpath kernel for learning hierarchical image representations". In: *Graph-Based Representations in Pattern Recognition*. Springer, 2015, pp. 34–43.

[54] A. L. Dahl, H. Aanæs, and K. S. Pedersen. "Finding the best feature detector-descriptor combination". In: *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on*. IEEE. 2011, pp. 318–325.

[55] K. Dalla Mura, J. A. Benediktsson, B. Waske, and L. Bruzzone. "Morphological Attribute Profiles for the Analysis of Very High Resolution Images". In: *Geoscience and Remote Sensing, IEEE Transactions on* 48.10 (2010), pp. 3747–3762.

[56] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. "Locality-sensitive Hashing Scheme Based on P-stable Distributions". In: *Proceedings of the Twentieth Annual Symposium on Computational Geometry*. SCG '04. Brooklyn, New York, USA, 2004, pp. 253–262.

[57] R. Datta, D. Joshi, J. Li, and J. Z. Wang. "Image retrieval: Ideas, influences, and trends of the new age". In: *ACM Computing Surveys (CSUR)* 40.2 (2008), p. 5.

[58] E. De Castro and C. Morandi. "Registration of Translated and Rotated Images Using Finite Fourier Fransforms". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 5 (1987), pp. 700–703.

[59] A. Desolneux, L. Moisan, and J.-M. Morel. "Edge detection by Helmholtz principle". In: *Journal of Mathematical Imaging and Vision* 14.3 (2001), pp. 271–284.

[60] M. N. Do and M. Vetterli. "The Contourlet Transform: An Efficient Directional Multiresolution Image Representation". In: *Image Processing, IEEE Transactions on* 14.12 (2005), pp. 2091–2106.

[61] M. N. Do and M. Vetterli. "The Finite Ridgelet Transform for Image Representation". In: *Image Processing, IEEE Transactions on* 12.1 (2003), pp. 16–28.

[62] W. Dong, M. Charikar, and K. Li. "Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces". In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2008, pp. 123–130.

[63] M. Donoser and H. Bischof. "Efficient maximally stable extremal region (MSER) tracking". In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2006, pp. 553–560.

[64] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000. ISBN: 0471056693.

[65] D. Espinoza-Molina and M. Datcu. "Earth-observation image retrieval based on content, semantics, and metadata". In: *Geoscience and Remote Sensing, IEEE Transactions on* 51.11 (2013), pp. 5145–5159.

[66] P. F. Felzenszwalb and D. P. Huttenlocher. "Efficient graph-based image segmentation". In: *International Journal of Computer Vision* 59.2 (2004), pp. 167–181.

[67] J. Flusser. "Refined Moment Calculation Using Image Block Representation". In: *Image Processing, IEEE Transactions on* 9.11 (2000), pp. 1977–1978.

[68] P.-E. Forssén. "Maximally stable colour regions for recognition and matching". In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[69] P.-E. Forssen and D. G. Lowe. "Shape descriptors for maximally stable extremal regions". In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE. 2007, pp. 1–8.

[70] F. Fraundorfer and H. Bischof. "A novel performance evaluation method of local detectors on non-planar scenes". In: *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*. IEEE. 2005, pp. 33–33.

[71] J. H. Friedman, J. L. Bentley, and R. A. Finkel. "An algorithm for finding best matches in logarithmic expected time". In: *ACM Transactions on Mathematical Software (TOMS)* 3.3 (1977), pp. 209–226.

[72] L. Garrido, P. Salembier, and D. Garcia. "Extensive operators in partition lattices for image sequence analysis". In: *Signal Processing* 66.2 (1998), pp. 157–180.

[73] T. Géraud, E. Carlinet, S. Crozet, and L. Najman. "A quasi-linear algorithm to compute the tree of shapes of nD images". In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2013, pp. 98–110.

[74] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. 2007.

[75] R. L. Graham and P. Hell. "On the history of the minimum spanning tree problem". In: *Annals of the History of Computing* 7.1 (1985), pp. 43–57.

[76] L. Guigues, J. P. Cocquerez, and H. Le Men. "Scale-sets image analysis". In: *International Journal of Computer Vision* 68.3 (2006), pp. 289–317.

[77] J. Havel, F. Merciol, and S. Lefèvre. "Efficient Schemes for Computing α-tree Representations". In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2013, pp. 111–122.

[78] M.-K. Hu. "Visual pattern recognition by moment invariants". In: *Information Theory, IRE Transactions on* 8.2 (1962), pp. 179–187.

[79] L. Igual. "Image segmentation and compression using the tree of shapes of an image. Motion estimation". PhD thesis. Universitat Pompeu Fabra, 2006.

[80] K. Iqbal, X-C. Yin, X. Yin, H. Ali, and H-W. Hao. "Classifier comparison for MSER-based text classification in scene images". In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE. 2013, pp. 1–6.

[81] T. Jaakkola, D. Haussler, et al. "Exploiting generative models in discriminative classifiers". In: *Advances in neural information processing systems (NIPS)* (1998), pp. 487–493.

[82] A. C. Jalba and M. A. Westenberg. "A comparison of two tree representations for data-driven volumetric image filtering". In: *Mathematical Morphology and Its Applications to Image and Signal Processing*. Springer, 2011, pp. 405–416.

[83] C. J. Jardine, N. Jardine, and R. Sibson. "The structure and construction of taxonomic hierarchies". In: *Mathematical Biosciences* 1.2 (1967), pp. 173–179.

[84] H. Jégou, M. Douze, and C. Schmid. "Hamming embedding and weak geometric consistency for large scale image search". In: *European Conference on Computer Vision*. Ed. by Andrew Zisserman David Forsyth Philip Torr. Vol. I. LNCS. Springer, Aug. 2008, pp. 304–317.

[85] H. Jegou, M. Douze, and C. Schmid. "Product quantization for nearest neighbor search". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33.1 (2011), pp. 117–128.

[86] H. Jégou, M. Douze, C. Schmid, and P. Pérez. "Aggregating local descriptors into a compact image representation". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 3304–3311.

[87] S. C. Johnson. "Hierarchical clustering schemes". In: *Psychometrika* 32.3 (1967), pp. 241–254.

[88] R. Jones. "Component trees for image filtering and segmentation". In: *Proc. IEEE Workshop on Nonlinear Signal and Image Process.* 1997.

[89] Y. Ke and R. Sukthankar. "PCA-SIFT: A more distinctive representation for local image descriptors". In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2004, pp. II–506.

[90] J. Kim and K. Grauman. "Boundary preserving dense local regions". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 1553–1560.

[91] J. B. Kruskal. "On the shortest spanning subtree of a graph and the traveling salesman problem". In: *Proceedings of the American Mathematical society* 7.1 (1956), pp. 48–50.

[92] C. H. Lampert, M. B. Blaschko, and T. Hofmann. "Beyond sliding windows: Object localization by efficient subwindow search". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[93] S. Lazebnik, C. Schmid, and J. Ponce. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2006, pp. 2169–2178.

[94] T. S. Lee. "Image Representation Using 2D Gabor Wavelets". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 18.10 (1996), pp. 959–971.

[95] S. Lefèvre, L. Chapel, and F. Merciol. "Hyperspectral image classification from multiscale description with constrained connectivity and metric learning". In: *6th International Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS 2014)*. 2014.

[96] H. Lejsek, B. Þ. Jónsson, and L. Amsaleg. "NV-Tree: Nearest Neighbors at the Billion Scale". In: *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*. ICMR '11. Trento, Italy, 2011, 54:1–54:8.

[97] P. Lienhardt. "Topological models for boundary representation: a comparison with n-dimensional generalized maps". In: *Computer-aided design* 23.1 (1991), pp. 59–82.

[98] D. O. Loftsgaarden and C. P. Quesenberry. "A nonparametric estimate of a multivariate density function". In: *The Annals of Mathematical Statistics* 36.3 (1965), pp. 1049–1051.

[99] D. G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[100] H. Lu, J. C. Woods, and M. Ghanbari. "Binary partition tree analysis based on region evolution and its application to tree simplification". In: *Image Processing, IEEE Transactions on* 16.4 (2007), pp. 1131–1138.

[101] J. MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[102] E. Maggiori, Y. Tarabalka, and G. Charpiat. "Optimizing Partition Trees for Multi-Object Segmentation with Shape Prior". In: *26th British Machine Vision Conference*. 2015.

[103] S. G. Mallat. "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 11.7 (1989), pp. 674–693.

[104] S. G. Mallat and W. L. Hwang. "Singularity Detection and Processing with Wavelets". In: *Information Theory, IEEE Transactions on* 38.2 (1992), pp. 617–643.

[105] B. S. Manjunath, J.-R. Ohm, V. V. Vasudevan, and A. Yamada. "Color and texture descriptors". In: *Circuits and Systems for Video Technology, IEEE Transactions on* 11.6 (2001), pp. 703–715.

[106] C. D Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[107] P. Maragos. "Pattern spectrum and multiscale shape representation". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 11.7 (1989), pp. 701–716.

[108] J. Matas, O. Chum, M. Urban, and T. Pajdla. "Robust wide-baseline stereo from maximally stable extremal regions". In: *Proc. Of British Machine Vision Conference (BMVC 2002)* (2002), pp. 384–396.

[109] G. Matheron. *Random Sets and Integral Geometry*. John Wiley & Sons, 1975.

[110] D. Menotti, L. Najman, and A. de Albuquerque Araújo. "1D Component tree in linear time and space and its application to gray-level image multithresholding". In: *Proceedings of the 8th International Symposium on Mathematical Morphology*. 2007, pp. 10–13.

[111] F. Merciol, L. Chapel, and S. Lefèvre. "Hyperspectral image representation through alpha-trees". In: *ESA-EUSC-JRC 9th Conference on Image Information Mining*. 2014, pp. 37–40.

[112] F. Merciol and S. Lefèvre. "Fast image and video segmentation based on alpha-tree multiscale representation". In: *Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on*. IEEE. 2012, pp. 336–342.

[113] F. Meyer. "Minimum spanning forests for morphological segmentation". In: *Mathematical morphology and its applications to image processing*. Springer, 1994, pp. 77–84.

[114] K. Mikolajczyk and C. Schmid. "A performance evaluation of local descriptors". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27.10 (2005), pp. 1615–1630.

[115] K. Mikolajczyk and C. Schmid. "An affine invariant interest point detector". In: *Computer Vision–ECCV 2002*. Springer, 2002, pp. 128–142.

[116] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. "A comparison of affine region detectors". In: *International Journal of Computer Vision* 65.1-2 (2005), pp. 43–72.

[117] D. Mishkin, J. Matas, and M. Perdoch. "MODS: Fast and robust method for two-view matching". In: *Computer Vision and Image Understanding* (2015).

[118] S. A. Mohamed and M. M. Fahmy. "Binary Image Compression Using Efficient Partitioning into Rectangular Regions". In: *Communications, IEEE Transactions on* 43.5 (1995), pp. 1888–1893.

[119] P. Monasse. "Contrast invariant representation of digital images and application to registration". PhD thesis. PhD thesis, University Paris XI, 2000.

[120] P. Monasse and F. Guichard. "Fast Computation of a Contrast-Invariant Image Representation". In: *Image Processing (TIP), IEEE Transactions on* 9.5 (2000), pp. 860–872.

[121] P. Monasse and F. Guichard. "Scale-space from a level lines tree". In: *Journal of Visual Communication and Image Representation* 11.2 (2000), pp. 224–236.

[122] J.-M. Morel and G. Yu. "ASIFT: A new framework for fully affine invariant image comparison". In: *SIAM Journal on Imaging Sciences* 2.2 (2009), pp. 438–469.

[123] O.J. Morris, M. de J. Lee, and A.G. Constantinides. "Graph theory for image analysis: an approach based on the shortest spanning tree". In: *Communications, Radar and Signal Processing, IEE Proceedings F* 133.2 (1986), pp. 146–152.

[124] M. Muja and D. G. Lowe. "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration". In: *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 2009, pp. 331–340.

[125] M. Muja and D. G. Lowe. "Scalable nearest neighbor algorithms for high dimensional data". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 36.11 (2014), pp. 2227–2240.

[126] E. Mwebaze, P. Schneider, F.-M. Schleif, J. R. Aduwo, J. A. Quinn, S. Haase, T. Villmann, and M. Biehl. "Divergence-based classification in learning vector quantization". In: *Neurocomputing* 74.9 (2011), pp. 1429–1435.

[127] B. Naegel and N. Passat. "Component-trees and multi-value images: A comparative study". In: *Mathematical Morphology and Its Application to Signal and Image Processing*. Springer, 2009, pp. 261–271.

[128] M. Nagao, T. Matsuyama, and Y. Ikeda. "Region extraction and shape analysis in aerial photographs". In: *Computer Graphics and Image Processing* 10.3 (1979), pp. 195–223.

[129] W. Nagel. "Image Analysis and Mathematical Morphology. Volume 2: Theoretical Advances". In: *The Monthly Microscopical Journal* 3.3 (1870). Ed. by J. Serra, pp. 597–597.

[130] L. Najman. "On the Equivalence Between Hierarchical Segmentations and Ultrametric Watersheds". In: *Journal of Mathematical Imaging and Vision* 40.3 (2011), pp. 231–247.

[131] L. Najman and M. Couprie. "Building the Component Tree in Quasi-Linear Time". In: *Image Processing (TIP), IEEE Transactions on* 15.11 (2006), pp. 3531–3539.

[132] L. Najman, J. Cousty, and B. Perret. "Playing with Kruskal: Algorithms for Morphological Trees in Edge-Weighted Graphs". In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2013, pp. 135–146.

[133] L. Najman and T. Géraud. "Discrete set-valued continuity and interpolation". In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2013, pp. 37–48.

[134] L. Najman and F. Meyer. "A short tour of mathematical morphology on edge and vertex weighted graphs". In: *Image Processing and Analysis with Graphs: Theory and Practice* (2012), pp. 141–174.

[135] L. Najman and P. Soille. "On Morphological Hierarchical Representations for Image Processing and Spatial Data Clustering". In: *WADGMM 2010*. Ed. by U. Köthe, A. Montanvert, and P. Soille. Vol. 7346. LNCS. Heidelberg: Springer, 2012, pp. 43–67.

[136] D. Nistér and H. Stewénius. "Linear time maximally stable extremal regions". In: *Computer Vision–ECCV 2008*. Springer, 2008, pp. 183–196.

[137] D. Nistér and H. Stewénius. "Scalable Recognition with a Vocabulary Tree". In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2006, pp. 2161–2168.

[138] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[139] Š. Obdržálek and J. Matas. "Object recognition using local affine frames on maximally stable extremal regions". In: *Toward Category-Level Object Recognition*. Springer, 2006, pp. 83–104.

[140] A. Oliva and A. Torralba. "Modeling the shape of the scene: A holistic representation of the spatial envelope". In: *International journal of computer vision* 42.3 (2001), pp. 145–175.

[141] G. K. Ouzounis, M. Pesaresi, and P. Soille. "Differential Area Profiles: Decomposition Properties and Efficient Computation". In: *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on* 34.8 (2012), pp. 1533–1548.

[142] G. K. Ouzounis and P. Soille. "Pattern Spectra from Partition Pyramids and Hierarchies". In: *ISMM 2011*. Ed. by P. Soille, M. Pesaresi, and G. K. Ouzounis. Vol. 6671. LNCS. Heidelberg: Springer, 2011, pp. 108–119.

[143] G. K. Ouzounis and M. H. F. Wilkinson. "A parallel implementation of the dual-input Max-Tree algorithm for attribute filtering". In: *Proc. Int'l Symp. Math. Morphology, GJ Banon, J. Barrera, UM Braga-Neto, and NS Hirata, eds*. Vol. 1. 2007, pp. 449–460.

[144] G. K. Ouzounis and M. H. F. Wilkinson. "Hyperconnected attribute filters based on k-flat zones". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33.2 (2011), pp. 224–239.

[145] G. K. Ouzounis and M. H. F. Wilkinson. "Mask-based second-generation connectivity and attribute filters". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29.6 (2007), pp. 990–1004.

[146] S. Ozkan, T. Ates, E. Tola, M. Soysal, and E. Esen. "Performance Analysis of State-of-the-Art Representation Methods for Geographical Image Retrieval and Categorization". In: *Geoscience and Remote Sensing Letters, IEEE* 11.11 (2014), pp. 1996–2000.

[147] S. J. Perantonis, B. Gatos, and N. Papamarkos. "Block decomposition and segmentation for fast Hough transform evaluation". In: *Pattern Recognition* 32.5 (1999), pp. 811–824.

[148] B. Perret, S. Lefevre, C. Collet, and É. Slezak. "Connected Component Trees for Multivariate Image Processing and Applications in Astronomy". In: *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE. 2010, pp. 4089–4092.

[149] B. Perret, S. Lefèvre, C. Collet, and É. Slezak. "Hyperconnections and hierarchical representations for grayscale and multiband image processing". In: *Image Processing, IEEE Transactions on* 21.1 (2012), pp. 14–27.

[150] F. Perronnin and C. Dance. "Fisher kernels on visual vocabularies for image categorization". In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[151] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. "Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases". In: *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*. 2008.

[152] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. "Object retrieval with large vocabularies and fast spatial matching". In: *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*. IEEE. 2007, pp. 1–8.

[153] R. C. Prim. "Shortest connection networks and some generalizations". In: *Bell system technical journal* 36.6 (1957), pp. 1389–1401.

[154] P. Pritchett and A. Zisserman. "Wide baseline stereo matching". In: *Computer Vision, 1998. Sixth International Conference on*. IEEE. 1998, pp. 754–760.

[155] C. Ronse. "Partial Partitions, Partial Connections and Connective Segmentation". In: *Journal of Mathematical Imaging and Vision* 32.2 (2008), pp. 97–125.

[156] A. Rosenfeld. "Adjacency in digital pictures". In: *Information and Control* 26.1 (1974), pp. 24–33.

[157] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* (Apr. 2015), pp. 1–42. DOI: 10.1007/s11263-015-0816-y.

[158] P. Salembier and L. Garrido. "Binary Partition Tree as an Efficient Representation for Image Processing, Segmentation, and Information Retrieval". In: *Image Processing, IEEE Transactions on* 9.4 (2000), pp. 561–576.

[159] P. Salembier, A. Oliveras, and L. Garrido. "Antiextensive Connected Operators for Image and Sequence Processing". In: *Image Processing (TIP), IEEE Transactions on* 7.4 (1998), pp. 555–570.

[160] P. Salembier and J. Serra. "Flat zones filtering, connected operators, and filters by reconstruction". In: *Image Processing, IEEE transactions on* 4.8 (1995), pp. 1153–1160.

[161] P. Salembier and M. H. F. Wilkinson. "Connected Operators". In: *IEEE Signal Processing Magazine* 26.6 (2009), pp. 136–157.

[162] G. Schaefer and M. Stich. "UCID: An Uncompressed Colour Image Database". In: *Electronic Imaging 2004*. International Society for Optics and Photonics. 2003, pp. 472–480.

[163] C. Schmid and R. Mohr. "Object recognition using local characterization and semi-local constraints". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.5 (1997), pp. 530–534.

[164] R. Sedgewick. *Algorithms (2Nd Ed.)* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1988. ISBN: 0-201-06673-4.

[165] J. Serra. "A Lattice Approach to Image Segmentation". In: *Journal of Mathematical Imaging and Vision* 24.1 (2006), pp. 83–130.

[166] J. C. Serra and P. Salembier. "Connected operators and pyramids". In: *Image Algebra and Morphological Image Processing IV*. Ed. by E. R. Dougherty, P. D. Gader, and J. C. Serra. Vol. 2030. SPIE. San Diego: SPIE Press, 1993, pp. 65–76.

[167] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. 2006.

[168] J. Shi and J. Malik. "Normalized cuts and image segmentation". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.8 (2000), pp. 888–905.

[169] B. W. Silverman. *Density estimation for statistics and data analysis*. Vol. 26. CRC press, 1986.

[170] J. Sivic and A. Zisserman. "Video Google: Efficient Visual Search of Videos". In: *Toward Category-Level Object Recognition*. Ed. by J. Ponce, M. Hebert, C. Schmid, and A. Zisserman. Vol. 4170. LNCS. Springer, 2006, pp. 127–144.

[171] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. "Content-Based Image Retrieval at the End of the Early Years". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.12 (2000), pp. 1349–1380.

[172] P. Sneath. "The application of computers to taxonomy". In: *Journal of general microbiology* 17.1 (1957), pp. 201–226.

[173] P. Soille. "Constrained Connectivity for Hierarchical Image Partitioning and Simplification". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.7 (2008), pp. 1132–1145.

[174] P. Soille. "Constrained connectivity for the processing of very-high-resolution satellite images". In: *International Journal of Remote Sensing* 31.22 (2010), pp. 5879–5893.

[175] P. Soille. *Morphological Image Analysis: Principles and Applications*. Second. New York: Springer-Verlag, 2003.

[176] P. Soille. "On genuine connectivity relations based on logical predicates". In: *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*. IEEE. 2007, pp. 487–492.

[177] P. Soille. "Preventing Chaining through Transitions While Favouring It within Homogeneous Regions". In: *Mathematical Morphology and Its Applications to Image and Signal Processing*. Springer, 2011, pp. 96–107.

[178] P. Soille and J. Grazzini. "Constrained Connectivity and Transition Regions". In: *Mathematical Morphology and Its Application to Signal and Image Processing*. Springer, 2009, pp. 59–69.

[179] R. R. Sokal and F. J. Rohlf. "The comparison of dendrograms by objective methods". In: *Taxon* 11.2 (1962), pp. 33–40.

[180] A. Solé, V. Caselles, G. Sapiro, and F. Arándiga. "Morse description and geometric encoding of digital elevation maps". In: *Image Processing, IEEE Transactions on* 13.9 (2004), pp. 1245–1262.

[181] Y. Song. "A topdown algorithm for computation of level line trees". In: *Image Processing, IEEE Transactions on* 16.8 (2007), pp. 2107–2116.

[182] Y. Song and A. Zhang. "Analyzing scenery images by monotonic tree". In: *ACM Multimedia Systems J.* 8.6 (2003), pp. 495–511.

[183] Y. Song and A. Zhang. "Locating Image Background By Monotonic Tree". In: *6th Joint Conf. on Information Sciences*. Ed. by H. Caulfield, SH. Chen, H-D. Cheng, R. Duro, V. Honovar, E. E. Kerre, M. Lu, M. Romay, T. Shih, D. Ventura, P. Wang, and Y. Yang. Durham: Association for Intelligent Machinery, Inc., 2002, pp. 879–884.

[184] Y. Song and A. Zhang. "Monotonic tree". In: *Discrete Geometry for Computer Imagery*. Springer. 2002, pp. 114–123.

[185] R. E. Tarjan. "Efficiency of a good but not linear set union algorithm". In: *Journal of the ACM (JACM)* 22.2 (1975), pp. 215–225.

[186] P. Teeninga, U. Moschini, S. C Trager, and M. H. F. Wilkinson. "Improved Detection of Faint Extended Astronomical Objects Through Statistical Attribute Filtering". In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2015, pp. 157–168.

[187] G. R. Terrell and D. W. Scott. "Variable kernel density estimation". In: *The Annals of Statistics* (1992), pp. 1236–1265.

[188] E. Tola, V. Lepetit, and P. Fua. "A fast local descriptor for dense matching". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[189] E. Tola, V. Lepetit, and P. Fua. "Daisy: An efficient dense descriptor applied to wide-baseline stereo". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.5 (2010), pp. 815–830.

[190] F. Tushabe and M. H. F. Wilkinson. "Content-based image retrieval using combined 2D attribute pattern spectra". In: *Advances in Multilingual and Multimodal Information Retrieval*. Springer, 2008, pp. 554–561.

[191] F. Tushabe and M. H. F. Wilkinson. "Image Preprocessing for Compression: Attribute Filtering". In: *Proc. World Congress on Engineering & Computer Science*. Citeseer. 2007, pp. 999–1005.

[192] T. Tuytelaars and L. J. Van Gool. "Wide Baseline Stereo Matching based on Local, Affinely Invariant Regions". In: *BMVC*. Vol. 412. 2000.

[193] J. R. R Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. "Selective search for object recognition". In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[194] E. R. Urbach. "Intelligent Object Detection Using Trees". In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2015, pp. 289–300.

[195] E. R. Urbach, J. B. T. M. Roerdink, and M. H. F. Wilkinson. "Connected Shape-Size Pattern Spectra for Rotation and Scale-Invariant Classification of Gray-Scale Images". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29.2 (2007), pp. 272–285.

[196] E. R. Urbach and M. H. F. Wilkinson. "Shape-only granulometries and grey-scale shape filters". In: *Proc. Int. Symp. Math. Morphology (ISMM)*. Vol. 2002. 2002, pp. 305–314.

[197] S. Valero, P. Salembier, and J. Chanussot. "Comparison of merging orders and pruning strategies for binary partition tree in hyperspectral data". In: *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE. 2010, pp. 2565–2568.

[198] C. Varytimidis, K. Rapantzikos, and Y. Avrithis. "WαSH: Weighted α-Shapes for Local Feature Detection". In: *Proceedings of European Conference on Computer Vision (ECCV 2012)*. Florence, Italy, Aug. 2012.

[199] A. Vedaldi and B Fulkerson. *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. http://www.vlfeat.org/. 2008.

[200] M. Vetterli and J. Kovačević. *Wavelets and Subband Coding*. Vol. 87. Prentice Hall PTR Englewood Cliffs, New Jersey, 1995.

[201] A. Vichik, R. Keshet, and D. Malah. "Self-dual morphology on tree semilattices and applications". In: *Mathematical Morphology and its Applications to Image and Signal Processing, Proc. of ISMM* 2007 (2007), pp. 49–60.

[202] V. Vilaplana, F. Marques, and P. Salembier. "Binary Partition Trees for Object Detection". In: *Image Processing (TIP), IEEE Transactions on* 17.11 (2008), pp. 2201–2216.

[203] L. Vincent. "Granulometries and opening trees". In: *Fundam. Inform.* 41.1-2 (2000), pp. 57–90.

[204] L. Vincent and P. Soille. "Watersheds in digital spaces: an efficient algorithm based on immersion simulations". In: *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on* 13.6 (1991), pp. 583–598.

[205] A. J. Viterbi and J. K. Omura. *Principles of Digital Communication and Coding*. McGraw-Hill, New York, 1979.

[206] H. Wang and S. Zhang. "Evaluation of global descriptors for large scale image retrieval". In: *Image Analysis and Processing–ICIAP 2011*. Springer, 2011, pp. 626–635.

[207] M. Wang, Q. M. Wan, L. B. Gu, and T. Y. Song. "Remote-sensing image retrieval by combining image visual and semantic features". In: *International journal of remote sensing* 34.12 (2013), pp. 4200–4223.

[208] Y. Weiss, A. Torralba, and R. Fergus. "Spectral hashing". In: *Advances in neural information processing systems (NIPS)*. 2009, pp. 1753–1760.

[209] M. Westenberg, J. B. T. M. Roerdink, and M. H. F. Wilkinson. "Volumetric attribute filtering and interactive visualization using the max-tree representation". In: *Image Processing, IEEE Transactions on* 16.12 (2007), pp. 2943–2952.

[210] M. H. F. Wilkinson. "A fast component-tree algorithm for high dynamic-range images and second generation connectivity". In: *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE. 2011, pp. 1021–1024.

[211] M. H. F. Wilkinson. "Generalized pattern spectra sensitive to spatial information". In: *Pattern Recognition, International Conference on*. Vol. 1. IEEE Computer Society. 2002, pp. 10021–10021.

[212] M. H. F. Wilkinson, H. Gao, W. H. Hesselink, J.-E. Jonker, and A. Meijster. "Concurrent computation of attribute filters on shared memory parallel machines". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30.10 (2008), pp. 1800–1813.

[213] S. A. J. Winder and M. Brown. "Learning local image descriptors". In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[214] S. A. J. Winder, G. Hua, and M. Brown. "Picking the best daisy". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 178–185.

[215] C. S. Won. "A Block-Based MAP Segmentation for Image Compressions". In: *Circuits and Systems for Video Technology, IEEE Transactions on* 8.5 (1998), pp. 592–601.

[216] Y. Xu, T. Géraud, and L. Najman. "Context-based energy estimator: Application to object segmentation on the tree of shapes". In: *Image Processing (ICIP), 2012 19th IEEE International Conference on*. IEEE. 2012, pp. 1577–1580.

[217] Y. Xu, T. Géraud, and L. Najman. "Morphological filtering in shape spaces: Applications using tree-based image representations". In: *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE. 2012, pp. 485–488.

[218] Y. Xu, P. Monasse, T. Géraud, and L. Najman. "Tree-based morse regions: A topological approach to local feature detection". In: *Image Processing, IEEE Transactions on* 23.12 (2014), pp. 5612–5625.

[219] Shengye Yan, Xinxing Xu, Dong Xu, Stephen Lin, and Xuelong Li. "Beyond spatial pyramids: A new feature extraction framework with dense spatial sampling for image classification". In: *Computer Vision–ECCV 2012*. Springer, 2012, pp. 473–487.

[220] Y. Yang and S. Newsam. "Geographic Image Retrieval Using Local Invariant Features". In: *Geoscience and Remote Sensing, IEEE Transactions on* 51.2 (2013), pp. 818–832.

[221] F. Zanoguera and F. Meyer. "On the implementation of non-separable vector levelings". In: *Proc. of VIth ISMM, Sydney, CSIRO* (2002), pp. 369–377.

[222] C. L. Zitnick and K. Ramnath. "Edge Foci Interest Points". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 359–366.

Résumé

Cette thèse explore l'utilisation de représentations hiérarchiques des images issues de la morphologie mathématique, les arbres des coupes, pour la recherche et la classification d'images. Différents types de structures arborescentes sont analysés et une nouvelle classification en deux superclasses est proposée, ainsi qu'une contribution à l'indexation et à la représentation de ces structures par des dendogrammes. Deux contributions à la recherche d'images sont proposées, l'une sur la détection de régions d'intérêt et l'autre sur la description de ces régions. Les régions MSER peuvent être détectées par un algorithme s'appuyant sur une représentation des images par arbres min et max. L'utilisation d'autres structures arborescentes sous-jacentes permet de détecter des régions présentant des propriétés de stabilité différentes. Un nouveau détecteur, basé sur les arbres des formes, est proposé et évalué en recherche d'images. Pour la description des régions, le concept de spectres de formes 2D permettant de décrire globalement une image est étendu afin de proposer un descripteur local, au pouvoir discriminant plus puissant. Ce nouveau descripteur présente de bonnes propriétés à la fois de compacité et d'invariance à la rotation et à la translation. Une attention particulière a été portée à la préservation de l'invariance à l'échelle. Le descripteur est évalué à la fois en classification d'images et en recherche d'images satellitaires. Enfin, une technique de simplification des arbres de coupes est présentée, qui permet à l'utilisateur de réévaluer les mesures du niveau d'agrégation des régions imposé par les arbres des coupes.

Mots-clés: Traitement d'images, recherche d'images, représentations hiérarchiques d'images, arbres des coupes, morphologie mathématique, détection de régions d'intérêt, description d'images.

Abstract

This thesis explores component trees, hierarchical structures from Mathematical Morphology, and their application to image retrieval and related tasks. The distinct component trees are analyzed and a novel classification into two superclasses is proposed, as well as a contribution to indexing and representation of the hierarchies using dendrograms. The first contribution to the field of image retrieval is in developing a novel feature detector, built upon the well-established MSER detection. The tree-based implementation of the MSER detector allows for changing the underlying tree in order to produce features of different stability properties. This resulted in the Tree of Shapes based Maximally Stable Region detector, leading to improvements over MSER in retrieval performance. Focusing on feature description, we extend the concept of 2D pattern spectra and adapt their global variant to more powerful, local schemes. Computed on the components of Min/Max-tree, they are histograms holding the information on distribution of image region attributes. The rotation and translation invariance is preserved from the global descriptor, while special attention is given to achieving scale invariance. We report comparable results to SIFT in image classification, as well as outperforming Morphology-based descriptors in satellite image retrieval, with a descriptor shorter than SIFT. Finally, a preprocessing or simplification technique for component trees is also presented, allowing the user to reevaluate the measures of region level of aggregation imposed on a component tree. The thesis is concluded by outlining the future perspectives based on the content of the thesis.

Keywords: Image processing, image retrieval, hierarchical image representation, component trees, Mathematical Morphology, feature description, feature detection.