


Thesis submitted to obtain the degree of doctor of philosophy from

Sorbonne Université

École Doctorale Informatique, Télécommunications et Électronique
Laboratoire de Recherche et Développement de l'EPITA

Taking into account inclusion and adjacency information in morphological hierarchical representations, with application to the extraction of text in natural images and videos

---

Lê Duy HUỲNH

Under the supervision of **Thierry GÉRAUD**, Professor at EPITA, LRDE
and **Yongchao XU**, Associate Professor at HUST, China, MCLab

Presented on December 13, 2018 in front of a jury composed of:

Reviewer :	**Beatriz MARCOTEGUI**, Professor at MINES ParisTech, CMM
	Hugues TALBOT, Professor at CentraleSupelec, CVC
Examiner :	**Isabelle BLOCH**, Professor at Telecom ParisTech, LTCI
	Laurent NAJMAN, Professor at Université Paris-Est, LIGM
	Camille KURTZ, Associate Professor at Université Paris Descartes, LIPADE

ACKNOWLEDGEMENTS

I would like to sincerely thank Mrs. Beatriz Marcotegui and Mr. Hugues Talbot, who accepted to review this thesis. I would also like to show my gratitude to Mrs. Isabelle Bloch, Mr. Laurent Najman and Mr. Camille Kurtz, who agreed to be members of my jury.

I am deeply grateful to Mr. Thierry Géraud for his support, his cheerfulness, his patience, motivation, and immense knowledge. He is the one helps me getting into the world of mathematical morphology. He taught me lots of computer skills and, from which I will benefit for all my life. Thank you for having been able to encourage me even when conditions were not the most favorable. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study.

I am thankful to Mr. Yongchao Xu, my other supervisor, for his massive assistance, ideas, fruitful discussions, patience and for pushing me toward my goal. I would also like to tell him how much I appreciated his great availability while he worked here. It was sad that you leave early but congratulation on your new position.

I would like to thank my fellow doctoral students, Anna, Jim, Ludovic, Minh and all of my colleagues from LRDE, for your friendship, cooperation, support, feedback, and for making this thesis experience unforgettable. It was fantastic to have the opportunity to work with you. A special thank goes to Daniela for her unfailing support and assistance throughout these years, especially for helping me going through all the administrative procedures.

I would like to thank my loved ones, my parents Huỳnh Kim Ngân and Lê Thị Phú and my sister Huỳnh Lê Thạch Thảo, for having been supporting me throughout my life, not just because that is what family do, but because they love me. I could not ask for more, really.

Thanks to my parent-in-law Cao Hồng Luận and Vũ Văn Lợi for supporting me spiritually throughout writing this thesis.

And finally, a particularly warm thank you to my wife, Linh, for her love, patience, and support, and for sometimes having to tolerate me over the past three years.

ABSTRACT

With the rising need for a higher understanding of images, the pixel-based representation is not enough. To answer this, the mathematical morphology framework provides several multi-scale, region-based image representations which include the hierarchies of segmentation (e.g., alpha-tree, BPT) and trees based on the threshold decomposition (Min/Max-trees and Tree of Shapes). Because objects in the real world rarely appear in isolation but a typical context with other related objects, we should consider the spatial relationships between image regions.

We are interested in two type of relationship, namely the inclusion and adjacency (in the sense of "being nearby") since they usually carry contextual information. The adjacency between regions gives us a sense of how regions are arranged in images and have been widely used. On the other hand, while fitting the human's perception of the object-background relationship: the objects are included in their background, the inclusion relationship is usually not taken into account. Both these drastic information opens up possibilities for image analysis. In this thesis, we take advantage of both inclusion and adjacency information in morphological hierarchical representations for computer vision applications.

We introduce the spatial alignment graph w.r.t inclusion (SAG) that is constructed from both inclusion and spatial arrangement of regions in the tree-based image representations.

For simple scenes, we introduce the Tree of Shapes of Laplacian sign. It encodes the inclusion of 0-crossing of a Morphological Laplacian map and performs well even in the case of uneven illumination. The ToSoL is computed in linear time w.r.t the number of pixels thanks to an optimization that mimics well-composedness. In this representation, the spatial alignment graph is reduced to a disconnected graph where each connected component is a semantic group of objects.

For higher detail representation, the spatial alignment graph becomes more complex. To address this issue, we expand the idea of the shape-spaces morphology. Our expansion has two primary results: 1) It allows the manipulation of any graph of shapes that encode different information, which encompasses the SAG. 2) It allows any tree filtering strategy proposed by the connected operators frameworks. Within this expansion, the SAG could be analyzed with an alpha-tree.

We demonstrated the application aspect of our method in text detection. The experiment results show the efficiency and effectiveness of our methods, which robust to noise, blur, or uneven illumination. These features are appealing to mobile applications.

Keywords: discrete topology, mathematical morphology, hierarchies, tree of shapes, hierarchy of segmentation, connected filters, text detection.

RÉSUMÉ

Avec la nécessité croissante d'une meilleure compréhension des images, la représentation d'image à base de pixel n'est pas bien adaptée. Pour répondre à cette demande, la Morphologie Mathématique a fourni plusieurs représentations d'images multi-échelles basées sur des régions, qui incluent les hiérarchies de segmentation (l'arbre de partition binaire, la hiérarchie des quasi-zones plates) et les hiérarchiques s'appuient sur la relation d'inclusion (arbres min/max et l'arbre des formes). Étant donné que les objets dans le monde réel apparaissent rarement isolément mais constituent un contexte typique avec d'autres objets associés, nous devons considérer les relations spatiales entre les régions d'image.

Nous nous intéressons à deux types de relations, à savoir l'inclusion et l'adjacence, car elles comportent des informations contextuelles. L'adjacence entre les régions nous donne une idée de la façon dont les régions sont organisées en images. D'autre part, La relation d'inclusion correspond à notre perception de la relation s'objet-fond: les objets sont inclus dans leur arrière-plan. Le but de cette thèse est de tirer partie à la fois des relations d'inclusion et d'adjacence pour mener à bien des tâches de vision par ordinateur.

Nous introduisons le graphe d'alignement spatial (GAS) qui est construit à partir de l'inclusion et de l'arrangement spatial des régions dans les représentations d'images basées sur des arbres. Pour l'application ne nécessite qu'une représentation grossière, nous présentons aussi l'arbre des formes du laplacien (AdFL) qui représente l'inclusion du passage de zéro d'une carte morphologique Laplacienne. Dans le cas de AdFL, le GAS est réduit à un graphe déconnecté où chaque composant connecté est un groupe sémantique d'objets. Il donne de bons résultats en cas d'illumination inégale, ce qui est un problème courant dans les images naturelles. Grâce à une optimisation qui imite du bien-composé, l'AdFL est construit en temps linéaire vis-à-vis du nombre de pixels. Pour les représentation plus détaillée, par exemple, un l'AdF classique, le GAS devient plus complexe. Pour résoudre ce problème, nous proposons d'élargir notre raisonnement à la morphologie basée sur la forme. Notre expansion a deux résultats principaux: 1) Elle permet de manipuler n'importe quel graphe des formes qui code des informations différentes. 2) Elle permet toute stratégie de filtrage dans la cadre de opérateurs connexes. Par conséquent, le GAS pourrait être analysé avec une hiérarchie des quasi-zones plates.

Nous avons démontré l'aspect applicatif de notre méthode dans l'application de la détection de texte. Les résultats de l'expérience montrent l'efficacité et l'efficience de nos méthodes, robustes au bruit, au flou ou à un éclairage irrégulier. Ces fonctionnalités sont attrayantes pour les applications mobiles.

Mots-clés: topologie discrète, morphologie mathématique, hiérarchies, arbre de formes, arbres d'adjacence, opérateurs connexes, détection de texte.

Contents

Introduction

Because of an increasing number of applications that require a higher understanding, image processing becomes pattern recognition or computer vision. For these higher-level tasks, it is insufficient to work at the pixel level. These tasks would benefit from region-based representation, in which the subject is not individual pixels but a local context (or regions) and how they are related together. Moreover, a general purpose application would also benefit from a multi-scale representation.

In that context, we are interested in the multi-scale region-based representations provided by the mathematical morphology framework. Unlike the scale-space, all regions encoded in these representations are presented in the original image. They could be classified into the hierarchies of segmentation (α-tree, Binary Partition Tree) and trees based on the threshold decomposition (Min/Max-trees and Tree of Shapes). Although they are all defined based on the set inclusion relationship ($\subset$), the former renders the adjacency of components in the image while the latter encodes how these component are included in each other.

The first type of hierarchical image representation is the hierarchies of segmentation, also known as pyramids [76]. The first example of this class is the quadtree [24], which is a top-down approach, was published in the early 1970's. Other bottom-up approaches were later proposed, which include the α-tree (also known as the hierarchy of quasi-flat zone) [63, 109, 57, 94, 74], and the binary partition tree (BPT) [81, 104]. They composed of a set of partition from fine to coarse and are usually represented by a tree structure. The leaves are corresponding to regions of the finest partition while other nodes represent the unions of all regions represented by their children. The root node represents the entire image domain as a single region. Some may argue that the hierarchies of segmentation are more flexible (than the next type of tree) since their framework could adapt to specific applications, e.g., by modifying the region model and dissimilarity function for BPT. Because these type of trees are built by merging adjacent regions (or breaking a region into multiple adjacent regions in case of top-down approaches), we can only deduce frome them the adjacency relation of sibling nodes.

The trees based on the threshold decomposition [86, 66] include the dual couple Min-tree and Max-tree [40, 41, 82] and the self-dual Tree of Shapes (ToS) [59, 97, 96]. In contrast to the hierarchies of segmentation, which usually rely on a dissimilarity measure between regions that make them contrast-dependant, the trees based on the threshold decomposition are generally contrast-invariant. The reason is that they rely on the pixel-value ordering, so any linear change in contrast does not affect the ordering. To be clear, they are affected by illumination changes, but robust to local change of contrast. The Min- and Max-tree are dual, which mean that a Min-tree on an image is the same as

the Max-tree on the image obtained by inversion of contrast. In that aspect, the ToS [17, 59], which is usually seen as a fusion of Min-tree and Max-tree, possess an interesting property: it is self-dual, which means it is invariant to inversion of contrast. This property is interesting when an assumption about the contrast of object vs. background cannot be made. Moreover, because the ToS encode the inclusion of level-lines, i.e., the boundaries of upper/lower level sets, it also encodes the inclusion relationship between flat-zones on images.

The inclusion of regions is interesting because it fits our perception of the object-background relationship: the object is usually included in its background. Having the inclusion of regions from the smallest up to the whole image allows us to handle the object-background notion relatively. It is desirable for two reasons. First, the object-background relationship is relative to the context. One region could act as the object but at the same times the background for others. Second, the object-background relationship also carries contextual information. Semantic regions are not only spatially closed but also usually in the same background. Having a hierarchical structure encodes the inclusion allow us to handle that relationship flexibly.

On the other hand, we should also consider the adjacency and other spatial relationship between regions. We use the term adjacency in the sense of "the state of being nearby", that is the spatial relationship between a region and nearby regions that is not its background. This relationship gives us a perception of how regions are arranged in images and also carries contextual information.

Problem statement:

The inclusion relationship gives us a flexible way to handle the objects-background relationship while the adjacency and other spatial relationship give us the relative position of regions in images. Such a drastic (but structured) information retrieval opens up possibilities for image analysis. Being able to consider both adjacency and inclusion information is a real issue in order to successfully express spatial relationships between objects of interested, i.e., between regions of a segmentation. The problem and the subject of this thesis are how to get *a representation that takes advantage of both inclusion and adjacency information* and how to *analysis the information encoded in that structure*.

Solution and contribution:

We explore one approach that starts with the inclusion of level lines, which already encoded in the ToS, then we add adjacency or other spatial relationship between these by a low-cost spatial search. This eventually leeds to the construction of a graph of shapes (or shape-spaces), from which regions of interest will be extracted with the help of connected filters. The contributions of this work are:

- A variant of ToS: the Tree of Shapes of Laplacian sign (ToSoL), which encodes the inclusion of 0-crossing of a Morphological Laplacian map and an efficient algorithm to construct such representation. This representation has a high resistance to contrast change thanks to the robustness of Morphological Laplacian.
- A simple grouping process on the ToSoL which take advantage of both inclusion and adjacency.
- An extension of the shape-spaces morphology [115].
- A graph, representing a shape-space, that is construct from both inclusion and spatial arrangement, called spatial alignment graph w.r.t inclusion.
- Application of the generalized shape-spaces morphology and spatial alignment graph w.r.t inclusion in text detection, with the possibility to expand to other application such as image filtering, image simplification.

Manuscript organization

This thesis is organized into three main chapters. **Chapter 1** presents the mains theoretical background that relates to our works which include a review of hierarchical representations provided by the mathematical morphology framework, the connected operator framework which deals with these representations and a short review of text detection in natural images, which is the application aspect in which we are interested.

- **Section 1.1** is a quick review of classical image representations, in particular, the definition of some image concept through the representation of images as a graph. We also discuss the necessity of hierarchical representations and the general concept of tree-based image representation.
- **Section 1.2** is a review of the first type of hierarchical, namely the hierarchies of segmentation. We will discuss some well-known structures which include the α-tree [94] and the Binary Partition Tree [81]
- **Section 1.3** reviews the morphological hierarchies based on the threshold decomposition which include the Min/Max-trees [40] and the Tree of Shapes [59]. We also review an extension of Tree of Shapes to multivariant images [16].
- **Section 1.4** reviews the connected operators, which is a class of filters that has an interesting property: they do not create new contours nor modify positions of existing ones. We review the tree-based implementation of connected filters [84], as well as the tree-based shape-spaces frameworks [115].
- **Section 1.5** is a small review of different approaches on text detection and recognition in natural images, which is the focused application of this thesis.

Chapter 2 presents our first approach in using both inclusion and adjacency information in the application of text localization and segmentation. In this chapter, we propose a variant of Tree of Shapes called the Tree of Shapes of Laplacian sign (ToSoL) and a regrouping process that uses both these types of information to obtain text candidates.

- **Section 2.2** and **2.3** present the ToSoL, which is a Tree of Shapes since it encode the inclusion of level lines. However, instead of the level lines of the image function like the classical ToS, the ToSoL encode the inclusion of the 0-crossing of the morphological laplacian. Although using Laplacian operator for contour detection a classical idea, our version is innovating for two reasons. First, we rely on the morphological Laplace operator which have higher resistance to uneven illumination, which is a common problem in natural images. Second, we organize the 0-crossings into an inclusion tree, from which the object-background relationship could be deduced. We also present a top-down linear time complexity algorithm to compute the ToSoL, as well as the possibility to directly perform filtering operator during its construction.
- **Section 2.4** and **2.5** we focus on text localization and segmentation in natural images using ToSoL. In these section, we presents a text characters segmentation method which is a good trade-off between efficiency (linear time complexity) and quality (with a competitive F-score) with an efficient grouping of characters into text boxes, taking full advantage of the inclusion information encoded in ToSoL as well as the adjacency of regions obtained by a simple spatial search.

Chapter 3 introduces the spatial alignment graph w.r.t inclusion with respect to inclusion as well as the generalization of shape-spaces morphology [115]. In the same chapter, we will apply the generalized shape-spaces morphology framework and the spatial alignment graph w.r.t inclusion for text detection.

- **Section 3.2** presents our extension of shape-spaces morphology [115]. Our extension is twofold. First, we expand the shape-spaces definition to any spaces represented by the graph $G(V, E)$ where the vertices set V corresponding to the nodes set of a hierarchical image representation. Second, instead of following Xu et al. [115], who use the second tree, which represents the shape-spaces, to select nodes that are filtered from the first tree. We propose to reconstruct the shape-spaces, and as a result, the first tree, in the same manner as the image space is reconstructed from the first tree: through the associated function. This extension has two main consequence: 1) We can fit any graph that encodes the relationship between nodes on a tree to shape-spaces morphology framework rather than only the graph that represents the parent-children relationship. 2) We can perform any tree filtering strategy proposed by the connected operator frameworks rather than only tree pruning ones.
- **Section 3.3** presents the general structure of a spatial alignment graph w.r.t inclusion. The graph is constructed by generalized the grouping process presented in

Section 2.4. That graph represents the alignment of regions encoded in the ToS. Neighbors of a node are regions that are not its background (regions in which it is included) and spatially aligned (regions that are in the desired direction in the image space). This structure takes advantage of both types of information. The distance on the tree, as well as the distance on image space, will be reflected in the weight of the edges. This graph represents a shape-spaces and could be analysis follow the generalized shape-spaces morphology.

- **Section 3.4** we present the application of generalized shape-spaces morphology on a spatial alignment graph w.r.t inclusion for text detection in natural images. By constructing an α-tree with a dissimilarity measure adapt to the application, we demonstrate that nodes on thats tree represent text characters on the images with high quality. We also propose some approaches to obtain a small set of candidates for the later stages of end-to-end text detection and recognition pipeline.

In **Chapter 4**, we conclude this thesis by a quick review of our approaches on taking advantage of both inclusion and spatial arrangement, in which "adjacency" is encompassed. We also present some possible improvement and further research on these applications. Finally, we give some suggestion for additional study of our approach.

Chapter 1

Background

1.1 Classical Image Representations

1.1.1 Image representation

Digital image processing is a process performed on an image in order to enhance or extract useful information. An image is a projection of the 3D real-life scene into the 2D plane of the image sensor. It is stored as a finite set of digital values, called picture elements or pixels, corresponding to the smallest elements of the image sensor. Each pixel holds values represent the data collected. Typically, the pixels are stored and displayed as a grid map to reflect the image sensor array. Like any other signal, images can be modeled and processed by different mathematical tools.

Classical image representation decomposes the image into fundamental elements: pixels. An image could be defined as a function with the spatial support is defined as a subset of $\mathbb{Z}^2$ to represent the pixel grid, and the values space depends on the type of image. Some classical tools for this representation include Fourier transform or wavelet transform for geometric analysis, image denoising, image compression, and texture analysis. The image grid could also be expressed as a matrix. Thus matrix tools are also applicable, especially eigen-decomposition for principal component analysis. Another interesting representation is modeling images as a graph. Graph-based image representation allows us to use tools provided by the graph theory framework.

With an increasing number of applications in image processing and computer vision that need a higher level of image understanding, handling the image at the pixel level is not sufficiently efficient. In a higher level representation, the region-based image representation, the images are modeled in general not by individual pixel but by sets of pixels or regions. This representation includes superpixels and hierarchical image representation. The latter is the focus of this work and will be reviewed in later sections. Usually, these region-based image representations are built on lower representation. Some term, e.g., connected components, path, flat-zones, could be easily explained by the graph-based representation. Let us take a look at some of these terms, presented within the graph-based representation of images.

Image as a graph

In graph theory, a graph is a pair $G(V, E)$ where V is a set of vertices (or nodes, points) and E the set of edges. Each edge in E joins two vertices in V. We usually denote the edges connecting two vertices v_1, v_2 as e_{v_1,v_2}. The two vertices v_1, v_2 are endpoints of e_{v_1,v_2} and they are said to *be adjacent* or *neighbors*. Edges may be directed or undirected. Directed edges, also called arcs or arrows, define an order between their two ends. Undirected edges, also called lines, are used when the direction from one vertice to another is not important. A graph is said directed or undirected according to this nature of its edges.

In the graph-based image representation, an image is usually depicted as an undirected graph $G(V, E)$. The set of vertice V could be naturally chosen to correspond to the set of image pixels. However, the set of edges does not come that naturally. We have to define the connectivity to determines the set of edges. The most commonly used connectivities are 4- or 8-connectivity for 2D images and 6- or 26-connectivity in 3D ones. In this representation, only vertices corresponding to image boundary have a lower degree (number of edges connected to it) than the value defined by the connectivity. The image function will then map each vertex to a value on the value space.

On a graph, a path π in G from x to y is an ordered set of vertices $\pi(x \to y) = \{p_1 = x, \dots p_N = y\}$ that for every $1 \leq i < N$, $e_{p_i,p_i+1} \in E$. Two vertices x and y is said to be connected in G if $\exists\pi(x \to y)$. A graph is said to be connected if every pair of vertices are connected. Usually, the image domain is connected.

A subgraph $X(V_X, E_X)$ of $G(V, E)$ is a graph that $V_X \subseteq V$ and $E_X \subseteq E$. X is said an induced subgraph if it is a subgraph and $E_X = \{e_{p,q} | p, q \in V \wedge e_{p,q} \in E\}$. X is said a spanning subgraph of G if $V_X = V$.

A subgraph X is a connected component of $G(V, E)$ if it is the maximal connected subgraph of G, i.e. if there exists a connected subgraph $X'(V_{X'}, E_{X'})$ of G and $V_X \subseteq V_{X'}$ then $V_X = V_{X'}$. We denote the set of connected components of a graph G by $\mathcal{CC}(G)$.

Let R be a set of pixels of image I represented by graph $G(V, E)$. We called R a region of I. If not specified, the graph representing R (the set of vertices corresponding to the set of pixels of R) is an induced subgraph of G denoted by $G_R(V_R, E_R)$.

The region boundary of R is the set: $E_{boundary}(R) = \{e_{p,q} \in E | p \in V_R \wedge q \notin V_R\}$. The inner (resp. outer) boundary points of R is the set of endpoints of $E_{boundary}(R)$ belong (resp. do not belong) to R.

Let $G_{\overline{R}}$ the graph presenting $\overline{R}$ (the complement set of R). The set of connected components of $G_{\overline{R}}$ could be divided into two class depend on whether that connected component contains image boundary pixels or not. The latter would be called holes and the former outer regions. We denote $\mathrm{Sat}(R)$ the hole filling operation that returns the regions $\mathrm{Sat}(R)$ that equal the union of R and all of its hole.

Let $G_0(V, E_0)$ a subgraph of $G(V, E)$, formed from the same vertice set V and a subset $E_0 \subset E$ that $\forall e_{x,y} \in E_0, I(x) = I(y)$. Connected components of G_0 are flat zones of the images. It is implied that a total order could be defined over the image's value space. It

is most successfully used with gray level images [83] where the flat zones are connected regions of the maximal size having the same gray level.

Flat zones are local maxima (minima) if all pixels on its outer boundary have strictly lower (higher) value.

In similar way, the semi-flat zones can be defined as connected component of $G_\alpha(V, E_\alpha)$ that generate from G by $E_\alpha \subset E$ that $\forall e_{x,y} \in E_\alpha, |I(x) - I(y)| \leq \alpha$. The partition by semi-flat zones (and its hierarchy) will be present in later section.

1.1.2 Hierarchical representations of images

In this section, we will take a deeper look into the hierarchical representation of images, also called tree-based image representation. Motivated by the multi-scale nature of images, this representation has received attention in image analysis, especially in the field of mathematical morphology. In section 1.1.2.2 and 1.1.2.3, we will introduce the tree-based image representations by a mathematical morphology viewpoint. Section 1.2 and 1.3 will take a deeper look in some well-known hierarchical images representation.

1.1.2.1 The necessity of hierarchical representations

A digital image is a pixels grid where each box is associated with a value. For a human, that set of pixels may have more than their individual value. We process the images by groups of neighboring pixels that have some semantic meaning and how these groups relate with each other. How exactly the human brain does that is in the scope of another field of study, in the field of images processing, we try to achieve that level of understanding (and maybe better). In traditional images processing, we work at with a small neighborhood of pixels. When a higher level of image understanding is needed, to become pattern recognition or computer vision, it is no longer sufficient to work at that level. To better analyze the scene, we need to consider larger regions.

The regions of interest can be of various sizes anywhere in the image. Therefore a semantic analysis of the images should be capable of doing so. Sometimes, the scale at which the image should be analyzed may be dictated by the underlying application. For example, separate the sky and building Figure 1.1 only require a coarse analysis of the scene since these regions are large; counting the number windows may need a finer analysis. Retrieve texts is that image challenging task since they appear at different scales, some are at the same scale as the windows, some are near the pixels level and barely readable.

Because of this multiscale nature of images, image processing methods would benefit from a multiscale representation. It is more useful to decompose the image into potential scales of interest and browse that collection for the proper one than to chose a priori which scale is best for each application.

One popular multi-scale analysis tool for multiscale image analysis is the scale-space representation. It represents an image as a one-parameter family of smoothed images

Figure 1.1 – The view outside my office window.

[11]. The most popular is the linear Gaussian scale-space. However, by using a smoothing kernel, they usually modify images contour, and they are not invariant to contrast changes.

Because of that, we are more interested in another multiscale image representation: the tree-based image representation. Contrast to the scale-space, connected components in the shape-space are already presented in the original image and their contour is actually those of the original image. Consequently, they are more interesting than scale-space in terms of contours shapes and locations. The tree-based image representation has been popularized by connected filters (will be reviewed in Section 1.4). Moreover, there are many applications in image processing and computer vision relying on this kind of image representation.

1.1.2.2 The lattice of partitions

The complete lattice theory is widely accepted as the appropriate algebraic basic for mathematical morphology[2]. This section recalls some mathematical background notions, especially lattice theory which are required when working with the hierarchical representation of the image, e.g., hierarchies of segmentation or hierarchies of partial partitions. As their name implies, they comprise of partitions, and they are hierarchies which consist of an order defined on a set. We will go through the definition of partitions, the lattice of partitions and how it leads to the so-call tree-based image representation.

Let an image I defined as a function from the domain space D_I to the value space V:

$$\begin{aligned} D_I &\to V \\ p &\mapsto I(p) = v \end{aligned}$$

A partition P of an image I is a division of its definition domain D_I into non-void, non-overlapping regions R which cover the entire D_I. Mathematically speaking, let D_I be an arbitrary set. A partition P of a set D_I is a family $\{R_i \neq \emptyset | R_i \subset D_I\}$ of subsets of D_I so that $\forall R_i \neq R_j \in P, R_i \cap R_j = \emptyset$ and $\bigcup_{R \in P} R = D_I$. If the second condition is not satisfied, P is a partial partition. Let denote the set of all partition of a set D_I by Π_I.

Let recal the notion of partially ordered set (poset) in *order theory*. Let S a set and $\leq$ a relation on S. The relation $\leq$ is a partial order if it is:

- Reflexivity: $\forall a \in S$, $a \leq a$
- Transitivity: $\forall a, b, c \in S$, $a \leq b$ and $b \leq c \Rightarrow a \leq c$
- Antisymmetry: $\forall a, b \in S$, $a \leq b$ and $b \leq a \Rightarrow a = b$

A partital order will become a total order if it is totality: $\forall a, b \in A$, $a \leq b$ or $b \leq a$.

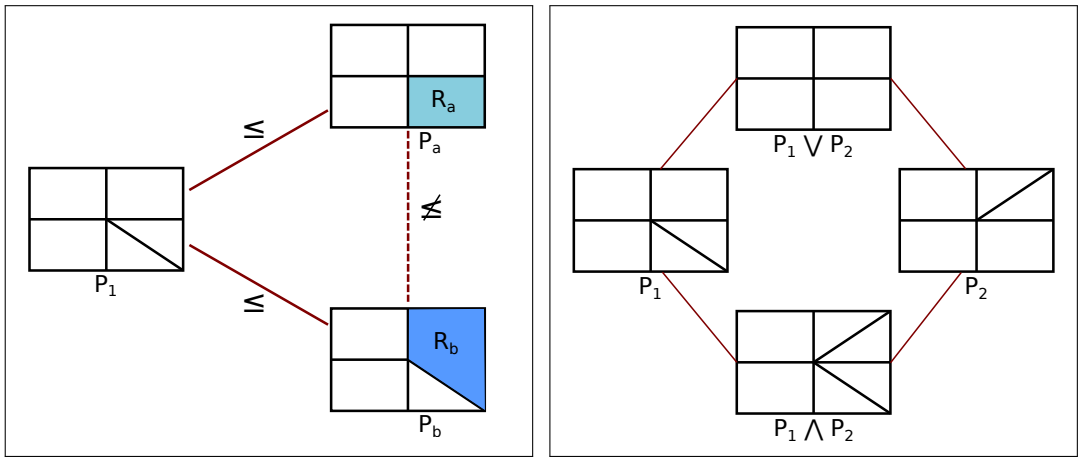
In [85], Serra noted that two partitions of an image I could be ordered to reflects their refinement. We denote the refinement order defined on then set of all partition of the domain of image I Π_I by $\leq$. For any two partition P, P' of a set D_I, P is said to be finner than P', denote by $P \leq P'$, if $\forall R \in P$, $\exists R' \in P'$ that $R \subset R'$. The partition P' is said to be coarser than P. Informally, we can say that $P \leq P'$ if P "contains" all the boundaries of P'.

The couple $(\Pi_I, \leq)$ is a partially ordered set (poset), any subset $\Pi' \in \Pi_I$ may admit a greatest lower bound (or refinement infimum) and a least upper bound (or refinement supremum). The former, denoted by $\bigwedge_{P \in \Pi'} P$, is the coarsest partition $P \in \Pi_I$ that is finer than any element of Π'. The latter, denoted $\bigvee_{P \in \Pi'} P$, is the finest partition $P \in \Pi_I$ that is coarser than any element of Π'. The poset $(\Pi_I, \leq)$ is a complete lattice because all subsets have both a supremum and an infimum [92]. The coarsest element has one region which is the whole set D_I itself, the finest partition is the set of all singletons of D_I [85].

The hierarchies of segmentation which we will present in Section 1.2 consist of a subset $\Pi \subset \Pi_I$ so that the refinement order has the totality property. The finest partition is a starting partition of the image, and the coarsest one is always the image domain itself. However, handling Π would be redundancy since one region may appear in more than one partition. This is resolved by replacing the set of partition Π by the set of regions $T_p = \bigcup_{\pi \in \Pi} \{\pi\}$ and the refinement order $\leq$ by the set inclusion $\subseteq$, on which the refinement order is based.

The hierarchies based on threshold decomposition in Section 1.3 in general only produce partial partitions of images. The set of regions T_d in all of these partial partitions could also be ordered by inclusion $\subseteq$.

On both types of hierarchical image representation, the region sets T_p and T_d consist of disjoint or nested regions. Therefore, the cover graphs of the posets $(T_x, \subseteq)$ are trees (in graph theory terminology) since they are acyclic and connected graphs. This leads to the tree-based image representation.



(a) (b)

Figure 1.2 – Illutration of refinement order 1.2(a), infimum and supremum 1.2(b). The refinement order $\leq$ defined on Π_I is a partial order. In 1.2(a), we see that $P_1 \leq P_a$ and $P_1 \leq P_b$ because every region of P_1 is included in one of the regions of P_a and P_b. However P_a and P_b are incomparable because $\exists R_a \in P_a$ but $\nexists R \in P_b$ that $R_a \subseteq R$ and $\exists R_b \in P_b$ but $\nexists R' \in P_a$ that $R_b \subseteq R'$. In 1.2(a), two set P_1 and P_2 and its infimum and supremum are represented in a The Hasse diagram ordered by refinement.

1.1.2.3 Tree-based image representation

A tree-based image representation $(T, \subseteq)$ (or usually T with the inclusion order implied) of an image I is a set of non-empty, disjoint or nestest regions T ordered by inclusion $\subseteq$: $T = \{D_I\} \cup \{R \subseteq D_I | R \neq \emptyset\}$ and $\forall R_i, R_j \in T, R_i \cap R_j \in \{\emptyset, R_i, R_j\}$

$(T, \subseteq)$ may be couple with a set function $F\colon T \to V$ that return the value associated with each region. F could return the level at which the region is obtain (e.g., in case of Min-,Max- tree or Tree of Shapes), the mean or median of image value in that region.

$(T, \subseteq, F)$This is an equivalent image representation since the image could always be reconstructed from the set of regions $(T, \subseteq)$ and its associated value function F. The reconstructed image I' from a tree-based image representation $(T, \subseteq)$ has the definition domain $D_{I'} = \bigcup R \subset T$ and the image function I':

$$\begin{aligned} D_{I'} &\to V \\ p &\mapsto I'(p) = F\left(Min\{R \in T | p \in R\}\right) \end{aligned}$$

In words, the value of a pixel p of the reconstructed image is the associated value of the minimum of all regions containing p in T. Because the disjoint or nested property, there always exists such mini=imum.

The tree-based implementation of connected operators discussed in Section 1.4.2 rely partially on this reconstruction process to obtain a filtered image.

T is usually presented as a graph $\mathcal{G}_T(V, E)$ where each vertex $v \in V$ accosicated with a region $R(v) \in T$ and each edges $e_{x,y} \in E$ connect two vertex x, y so that $R(x)$ covers $R(y)$ in $(T, \subseteq)$ (i.e., $R(y) \subset R(x)$ and $\nexists x'$ that $R(y) \subset R(x') \subset R(x)$) or reverse. That graph is a

tree (in graph theory terminology) and its root is alway chosen to be the node associated with the entire definition domain D_I. If there is not any confusion, we may use the node x and its associated region $R(x)$ interchangely. Several terminology are usually used when speaking about nodes on a tree-based representation:

In a tree-based representation $(T, \subseteq)$ of a set I, let $x \in \mathcal{G}_{\mathcal{T}}(V, E)$

- The parent of x is the node $par(x)$ that is on the path from x to root and to which x is directly connected. We always have $R(x) \subset R\left(par(x)\right)$ Every nodes except the root has a unique parent. Sometime we may refer to the grand-parent of x which is $par(par(x))$
- The children of x are all node $x' \in T$ that x is its parent. We denote the set of children of x by $C(x)$
- The leave nodes are nodes that have no children.
- The sibling of x is the set of nodes $Sib(x)$ that have the same parent as x. Sometimes, we may refer to uncles of x which is the set $Sib(par(x))$.
- Ancestors of x are elements on the path from x to root.
- Descendants of x are either children of x or a descendant of any of the children of x. In another word, it is the set of nodes for which x is an ancestor.
- The subtree root at x, denote by $T(x)$ contain x and all of its descendant.
- The height of a node is the length of the longest downward path to a leaf from that node. The height of the tree is the height of the root.
- The depth of a node is the number of its ancestors.

Some of these terms are illustrated in Figure 1.3.

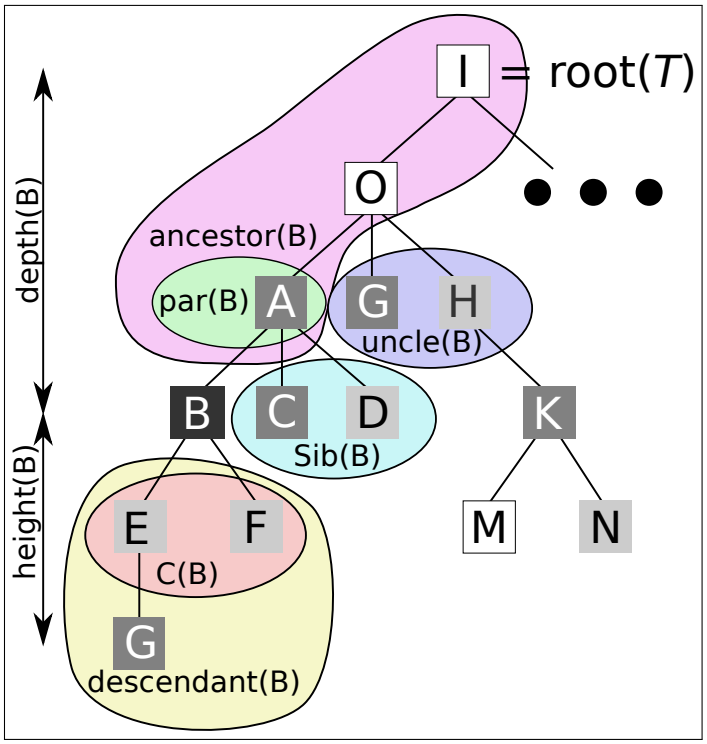


Figure 1.3 – Illutration of tree-based representationt terminology.

1.2 Hierarchies of Segmentation

1.2.1 Minimum spanning tree

The Minimum Spanning Tree (MST) is not defined as a tree-based image representation. However, the MST, especially Kruskal's algorithm [47], is the core of some efficiency algorithms to compute other hierarchical image representation [65].

Given an edge-weighted undirected graph $G(V, E)$, a spanning tree of G is a subgraph that is a tree and includes all the vertices of G. By definition, an unconnected graph could not contain a spanning tree (but a spanning forest). A connected graph would have more than one spanning tree. On an edges-weighted connected graph, we are interested in finding a minimum spanning tree (MST) [47] which is a spanning tree and its total weight is less than or equal any other spanning tree of the same graph. An example of MST is given in Figure 1.4.

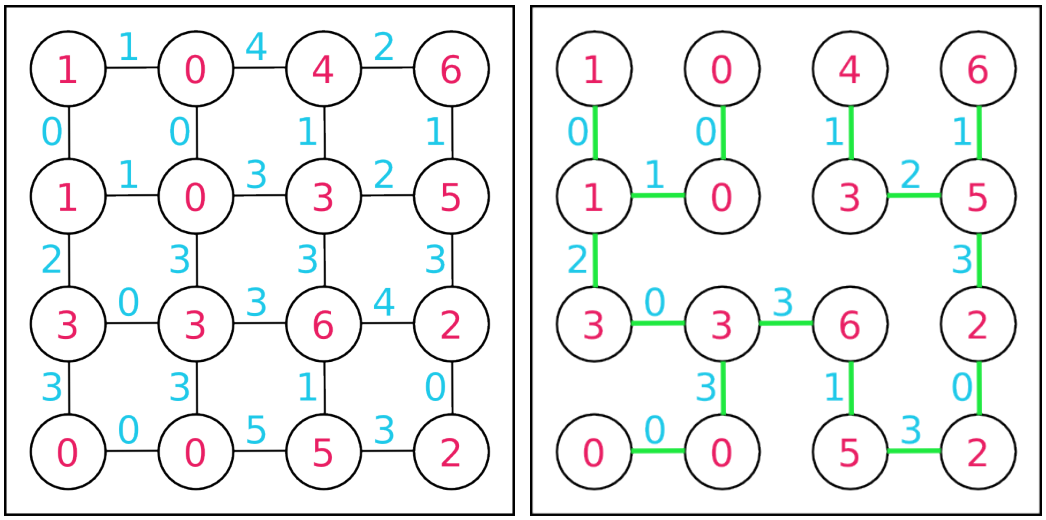
There are several algorithms that compute MST for undirected graph, include Boruvka's algorithm [6], Prim's algorithm [38], reverse-delete algorithm and Kruskal's algorithm [47]. All of them are the greedy algorithm. Kruskal's one is the most commonly used and could be present as in Algorithm 1.

The MST is a well-known problem in graph theory and has found application in images analysis [99]. It is also useful in the construction of other structure in the image processing field. In [20], Cousty et al. introduce watersheds as optimal spanning forest. In the follow-up works [65], Najman et al. introduce several variations of Kruskal's algorithm to compute various hierarchical structures, in particular, the tree corresponding to hierar-

Input: An Edge-weighted graph $G(V, E)$
Output: A minimum spanning tree if G is connected, a minimum spanning forest otherwise

```
Kruskal(G(V, E)):
    Create a forest where each vertex in V is a separate tree ;
    For each edge e ∈ E in increasing order, if e links two trees then merge them
    into a single one and add e to the forest;
```

Algorithm 1: Kruskal's Algorithm.



(a) An image represented as a Graph (b) Its Minimum Spanning Tree

Figure 1.4 – An example of a minimum spanning tree, 1.4(a) is the input image, represented as a graph whose vetices weighted by their gray value and edges weighted by their gray value different. 1.4(b) shows one of the Minimum Spanning Tree of that graph.

chies of watershed cuts and α-tree. This is achieved by post-processing a binary partition tree corresponding to an ordered version of the edges of the minimum spanning tree.

1.2.2 Binary partition tree

The Binary Partition Tree (BPT) proposed by Salembier et al. [81] encodes the hierarchical decomposition of an image. The BPT reflects the similarity between neighboring regions. Nodes on the BPT represents image regions at different scales. Those that are close to the root usually correspond to larges regions while leaves represent small detail.

Its construction starts from initial partitions (pixels, flat zones or precompute partition). At each step, a similarity measure between all neighboring region is calculated, and only the most similar pair of regions will be merged (thus the hierarchy is a binary tree). The merging repeat until only one component left. The BPT is obtained by keeping track of this merging process. The leaves in BPT are regions in the initial partitions. Each merging is done by creating a new parent node which is linked to its children.

The region model, which specify how to model the region and their union, and the

merging criterion, which is the similarity measure between two region models, drive the construction of the BPT and thus define the meaning of the final structure. The region model proposed in the first paper [81] was the mean color of each region, with an assumption that the color is homogenous within each region. The mean color could be a scalar (gray-level image) or a vector (color image). That region models have been applying in different color space: RGB [4], CIELUV [81], CIE L*a*b* [54] [53]. The similarity measure should depend on the application. One measure that has been used by paper mention above is express by:

$$O(R_1, R_2) = |R_1| \times ||M_{R_1} - M_{R_1 \cup R_2}||_2 + |R_2| \times ||M_{R_2} - M_{R_1 \cup R_2}||_2$$

With R_1, R_2 denote two region, M_R denotes the region model of R and $||.||_2$ is the L_2 norm. Other norms such as L_1 or L_∞ could also be used.

Depend on how the similarity measure is defined, the similarity measure would need to be updated for every step. One example of BPT is given in Figure 1.5.

BPT found application in image segmentation and object detection [81, 104, 54]. The BPT framework is also a versatile hierarchy representation. It has been extended to hyperspectral image [102] by redefining the region models and similarity measure. The Binary partition Tree by altitude ordering formalized in [65] is an intermediate step to obtain a hierarchy of watershed cuts and α-tree.

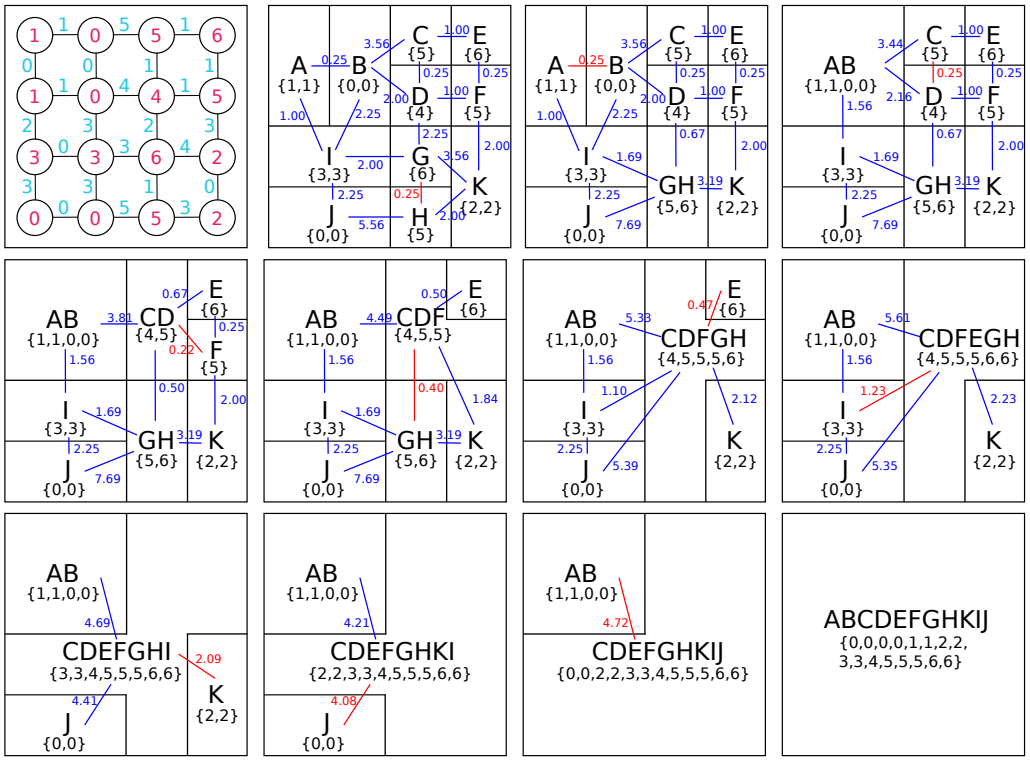
1.2.3 Alpha-tree and its variations

The α-tree [94], which is also known as quasi-flat zone hierarchy [57], is a hierarchical structure built from the α-connected components (or quasi-flat zone). It is based on the α-connectivity, which could be seen as a parameter-dependent connectivity [109] or constrained connectivity [94]. A similar method has been described by Nago et al. [63] in 1979.

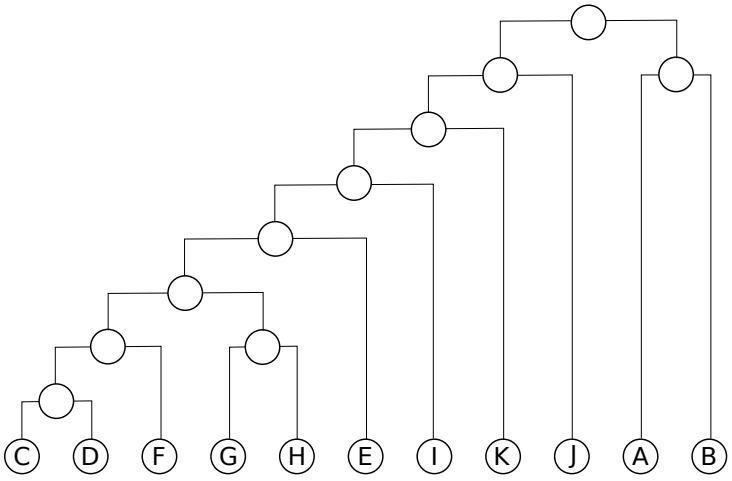
The connectivity is defined on image I presented as a graph $G(V, E)$. Two pixels x, y are α-connected (i.e., belong to the same quasi-flat zone) if there exists a path $\pi(x \rightarrow y) = \{p_1 = x, \ldots, p_N = y\}$ from x to y that $d(p_i, p_{i+1}) \leq \alpha; \forall 1 \leq i < n$. Usually, the α-connectivity is demonstrated on a scalar image and edges are weighted by the intensity difference between a pair of neighboring points $d(x, y) = |I(x) - I(y)|$.

The α-connected relationship is an equivalence relation. First, it is reflexive because a point is $\alpha\text{-}connected$ to itself. Second, it is symmetric because if x is $\alpha\text{-}connected$ to y via π then y is $\alpha\text{-}connected$ to x via the reversal of π. Finally, it is transitive because if x is $\alpha\text{-}connected$ to y via π_1 and y is $\alpha\text{-}connected$ to z via π_2 then x is $\alpha\text{-}connected$ to z via the concatenation of pi_1 and pi_2. The α-connected components (α-CC) is defined as equivalence classes on the image definition domain [94], i.e. the maximum set of points that are $\alpha\text{-}connected$. Consequently, the set of all α-CCs on an image definition domain defines a partition of that domain.

The greater the value of α is, the larger α-CCs. Moreover, the α-CCs contain a pixel



(a) Input image and step-by-step BPT construction



(b) The Binary Partition Tree

Figure 1.5 – An example of a Binary Partition Tree. A region is modeled by the set of its pixels' value, and the similarity measure is the variance of the union. With this merging criteria, the neighborhood and similarity measure is needed to be updated for every step. In this example, we build the BPT of the same image in the MST section. Figure 1.5(a) shows the original image and the regions adjacency graph at each step (blue edges, weighted by variance). The red edge is the one with minimum weight, which connects regions that will be merged next step.

p form of an ordered sequence when the value of α is increased : α-$CC(p) \subseteq \alpha'$-$CC(p)$ with $\alpha \geq \alpha'$. As a consequence, when α increase, α-CCs grow and merge and the chain of

α-partition forms a hierarchy. An α-tree example is given in Figure 1.6

From the implementation point of view when constructing the α-tree, the alpha level where two adjacency region A and B merge is determined by the edge(s) of minimal weight connecting them. Other edges connecting the two subsets could be ignored without effect the final tree. This is why the implementation of the α-tree is related to MST and single-linkage clustering [30]. For that reason, Kruskal's MST algorithm allows an effective way to compute the α-tree [65].

The α-tree has two well-known major problems. The first one could be observed clearly in noisy images. Noise pixels will form small isolated regions close to root because of its high different to surrounding pixels. A segmentation using α-tree would need post-processing (e.g., a gain filter) to remove those small regions. The second problem is the chaining effect in regions with a low gradient. Because the α-connectivity consider only local parameter (dissimilarity between two pixels), clusters may grow faster than expected for a specific α.

To address the chaining effect, in [31] Hambrusch et al. proposed a technique to control the growth of α-CC by using a global parameter to limit the property variation (e.g., pixel intensity) within a component. With d is the dissimilarity between two adjacent pixels and R returns the maximum dissimilarity in π, the (α,ω)-Hambrusch relation (as called in [94] by Soile et al.) could be presented as:

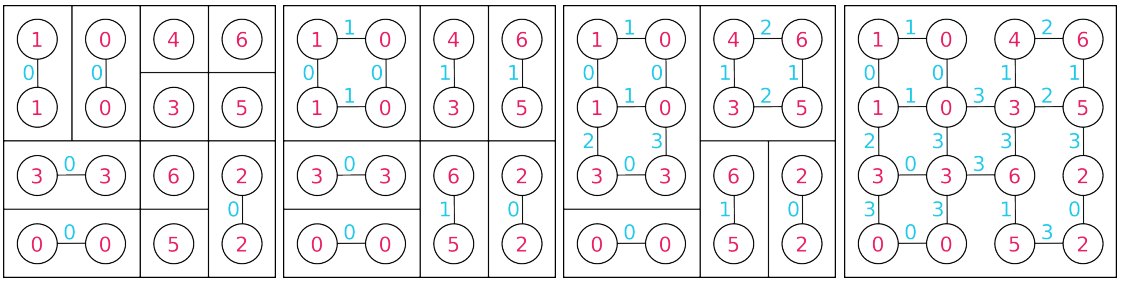
$$\begin{aligned} x, y \text{ are } (\alpha,\omega)\text{-Hambrusch related} \Leftrightarrow \exists \pi(x \to y) = \{p_1 = x, ..., p_N = y\} \\ : \forall i < N, d(p_i, p_{i+1}) \leq \alpha \wedge R(p \in \pi) \leq \omega \end{aligned} \tag{1.1}$$

Because the (α,ω)-Hambrusch relation is not an equivalence relation (it is not transitive), it does not lead to a unique partition of the image definition domain. For this reason, the image partition algorithm of Hambrusch et al. needs a decision making process to guide the growing of (α,ω)-Hambrusch components. In their paper, Hambrush et al. define the (α,ω)-Hambrusch component is the set of pixels that all pairs are (α,ω)-Hambrusch related. A partition produced by (α,ω)-Hambrusch relation is required to satisfy the maximum property: no valid segment could be merged with other valid segments to form a valid segment.

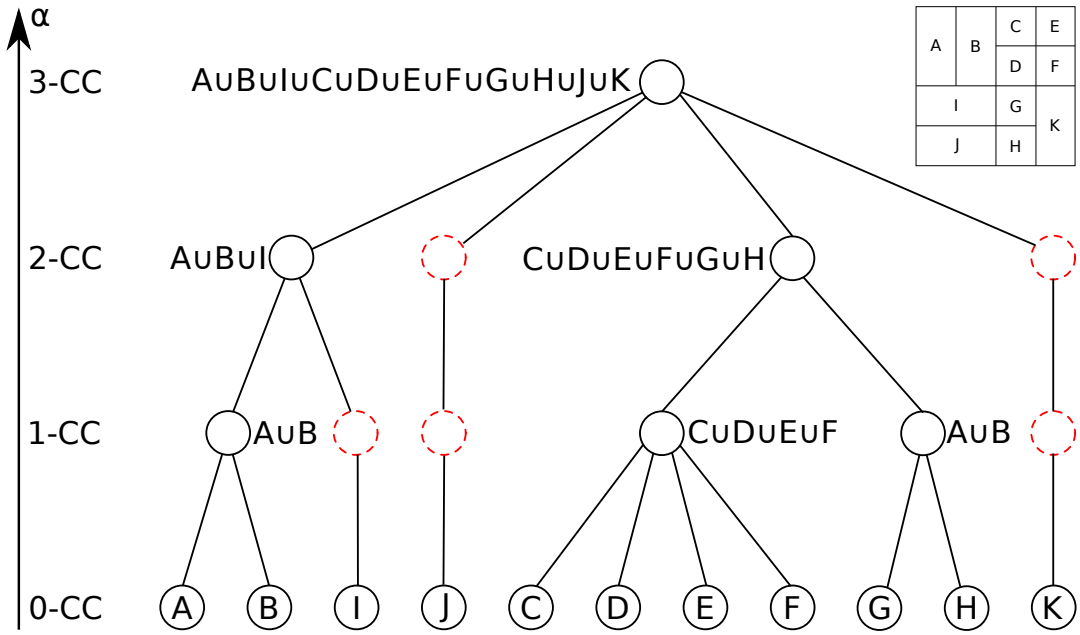
Unsatisfy with the non-unique nature of the (α,ω)-Hambrusch partition of image, in [94], Soille et al. propose another connectivity relation relying on both local and global parameters. According to Soille's notion, (α,ω)-connected component contains a pixel p (α,ω)-CC(p) is the largest α_i-CC(p) that $\alpha_i \leq \alpha$ and value range is not greater than ω:

$$(\alpha, \omega) - CC(p) = max\left\{\alpha_i\text{-}CC(p) | \alpha_i \leq \alpha \wedge R(x \in \alpha_i\text{-}CC(p)) \leq \omega\right\} \tag{1.2}$$

Finding (α, ω)-CC(p) is actually finding the nearest node to root on α-tree that contains p and satisfies the local and global variation parameter. An (α, ω)-partition of the image could be obtained simply by a cut on the α-tree. Thanks to the ordering relation between



(a) 0-CCs (b) 1-CCs (c) 2-CCs (d) 3-CCs



(e) The alpha tree

Figure 1.6 – An example of an alpha tree.

α-CCs, the following ordering relation holds for every pixel p:

$$(\alpha,\omega)\text{-}CC(p) \subseteq (\alpha',\omega')\text{-}CC(p) \text{ with } \alpha \leq \alpha' \text{ and } \omega \leq \omega' \tag{1.3}$$

However, due to the unknown order when $\alpha \leq \alpha'$and$\omega \geq \omega'$, the set of all (α,ω)-CCs cannot form a hierarchy [7].

In the same paper, observing that the local parameter does not play a role when $\alpha \geq \omega$ because $(\alpha' > \omega,\omega)\text{-}CC(p) = (\alpha = \omega,\omega)\text{-}CC(p)$, Soille et al. proposed the concept of (ω)-CC. It is the largest $\alpha \geq \omega$ whose global range does not exceed ω. The set of ω-CCs forms a hierarchy, called the (ω)-tree. The (ω)-tree could be seen as a reorganized of the α-tree. A non-horizontal cut on the α-tree using global ranges as criteria corresponds to a horizontal cut on the (ω)-tree. Similar to the notion of (α,ω)-CC of Soille et al., it does not contain any region that is not already on the α-tree and therefore, does not solve the chaining effect. However, it does reflect better the region criteria.

The α-tree have been generalized for multichannel images [94].

1.3 Trees Based on the Threshold Decomposition

In this section, we review the three classical hierarchical representations based on threshold decomposition, to be specific, the Min- and Max- tree [82] and the topographic maps [17] which are also known as the Tree of Shapes (ToS) [59]. The Min- and Max- trees are dual, which mean that the Min-tree of an image is the Max-tree of its complementary and vice versa. The ToS is a self-dual representation which is obtained by merging the Max-tree and Min-Tree. These trees are fundamentally different from the hierarchy of segmentation discussed in the earlier section. First, a horizontal cut in the former yields a partial partition while a horizontal cut in the later yields a partition. Second, the former represents the inclusion of components while the later renders the adjacency of regions.

1.3.1 Min/Max tree

The component trees were introduced by Jones [40, 41] and popularized as the Min- and Max-trees by Salembier et al. [82] who found them to be an efficient image representation adapted for connected filters[83]. The Min (resp. Max)-tree based on inclusion relationship between dark (resp. light) structures in the image.

Given X a image definition domain, let an image $u : X \to F$ where F is endowed with an ordering relationship $\leq$ and a level $\lambda \in F$, the lower and upper level sets at level λ are respectively defined by $[u \leq \lambda] = \{x \in X \mid u(x) \leq \lambda\}$ and $[u \geq \lambda] = \{x \in X \mid u(x) \geq \lambda\}$. We denote $\mathcal{CC}$, the operator that takes a set of pixels and gives its set of connected components. The $\Gamma^-_\lambda = \mathcal{CC}([u \leq \lambda])$ and $\Gamma^+_\lambda = \mathcal{CC}([u \geq \lambda])$ are the lower and upper peak components at level λ. The lower and uper set of connected components are denote by $\Gamma^- = \bigcup_\lambda\{\Gamma^-_\lambda\}$ and $\Gamma^+ = \bigcup_\lambda\{\Gamma^+_\lambda\}$. If the ordering relation $\leq$ is total, two connected components in the lower or upper set are either disjoint or nested. The inclusion relationship between connected components of Γ^- and Γ^+ yields the structure of Min-tree and Max-tree respectively. The leaves of Min (resp. Max)-trees corresponds to local image minima (resp. maxima). The root node of the tree is the whole image domain.

In implementations, the node representing the connected component at level λ only store pixels that have level λ. We refer these pixels as *proper pixels*. The set of pixels of the connected component represented by n is given by the whole subtree rooted at n. Images could always be reconstructed from the Min- and Max- trees [17]: $u(x) = sup\{\lambda \in F | x \in \Gamma^+_\lambda\} = inf\{\lambda \in F | x \in \Gamma^-_\lambda\}$. An image reconstructed from a pruned Min (Max)-tree will have their dark (light) regions enlarged.

Various algorithms have been proposed to compute efficiently the Max-tree and Min-tree. They are classified in [15] into three classes: the flooding algorithms, the immersion algorithms, and the merge-based algorithms. The first one is based on the flooding procedure [82, 71, 111] which is a top-down construction. It starts with pixels at the root (pixels with the maximum value in case of Min-tree and the minimum value in case of Max-tree), then a depth-first propagation is performed to build the final tree. The second

approach [64] based on the Tarjan's union-find algorithm [100]. It consists of two steps. First, N image pixels are sorted according to their intensity value and then form N disjoint singleton sets. In the second step, in reverse order, those singletons are merged to form a tree. A complexity comparison of these two approaches is given in [7]. Finally, the merge-based algorithm [73, 61] is an adaptation of the first two classes for parallelism. It starts by dividing the image into blocks. Then Min/Max-trees for each block is computed by another algorithm and then merged to form a single tree for the whole image.

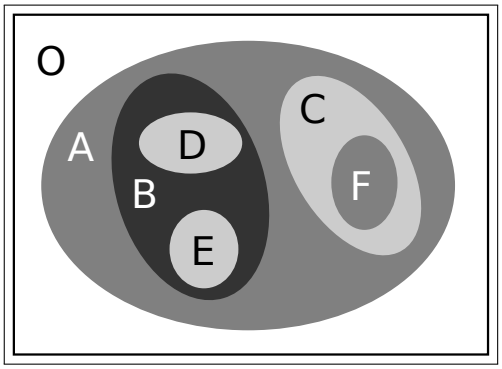
1.3.2 Tree of Shapes

The Min- and Max-tree represent the bright and dark components on image respectively. Representing both types of objects at the same time by handling both trees leads to redundant and difficulty in ensuring consistency. Several authors proposed to consider the inclusion of level lines (the topological boundaries of the upper and lower level sets). This representation of images is called the Tree of Shapes (ToS) [59], which is also known as the topographic map [59], monotonic tree [97] or level line tree [96]. The ToS could represent both the bright and dark structure simultaneously since it makes no assumption about the object contrast.

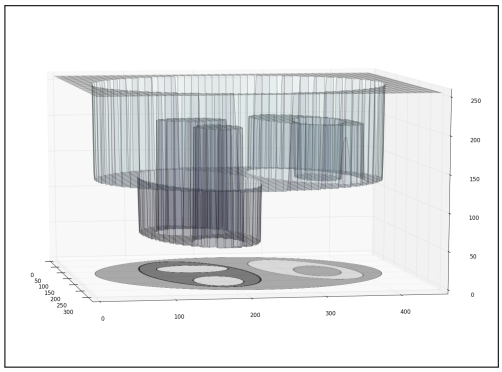
The Tree of Shapes is based on the concept of shapes [59]. A shape is a connected component of the upper or lower level sets with the holes filled. With $\mathcal{CC}$, the operator that takes a set and gives its set of connected components, and the cavity-fill-in operator Sat mentioned in Section 1.1.1, the set of shapes of an image $u : X \to F$ is defined as $\mathfrak{S}(u) = \bigcup_{\lambda \in F} \{ \mathrm{Sat}(\Gamma);\ \Gamma \in \mathcal{CC}([u < \lambda]) \cup \mathcal{CC}([u \geq \lambda]) \}$. These shapes are proven to be either nested or disjoint [59], thus $\mathfrak{S}(u)$ could be organized into a tree. The ToS is self-dual which means that $\mathfrak{S}(u) = \mathfrak{S}(\bar{u})$. It is because the upper and lower sets are only swapped in the dual image $\bar{u}$ thus they yield the same ToS. Although being seen as a combination of the Min- and Max-tree, the hole-filled operation may create components that do not belong to both tree.

Similar to the Min- and Max-tree, in implementation, nodes on the ToS only store their proper pixels. An image could be reconstructed from the Tree of Shapes by a direct or indirect reconstruction [18]. The direct reconstruction recovers the value at x by getting the level of the smallest shape containing x. The indirect reconstruction deduces the upper or lower level sets and then reconstructs the image in the same way as with the Min- and Max-tree.

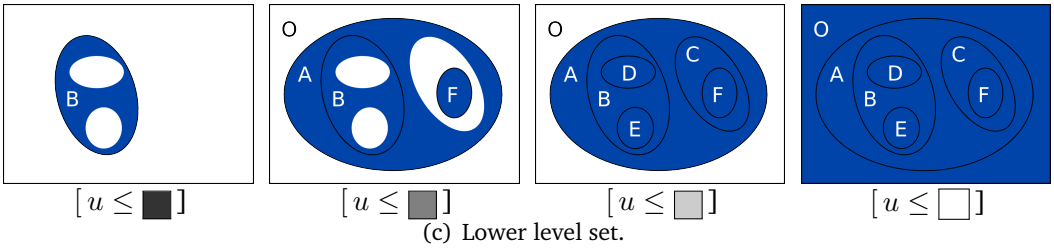
The early approaches to construct the ToS include the fast level line transform and its improved version, fast level set transform by Monasse et al. [59]. They take a region growing approach to build both the Max- and Min- tree and then compile them. A top-down approach was proposed by Song et al. [96] which find the level lines directly instead of level set components. Both this approach have the worst-case time complexity of $O(N^2)$ and cannot easily be extended to multidimensional images. Géraud et al. [29] overcome both these drawbacks. Their approach is based on the immersion algorithm to compute



(a) A simple image.

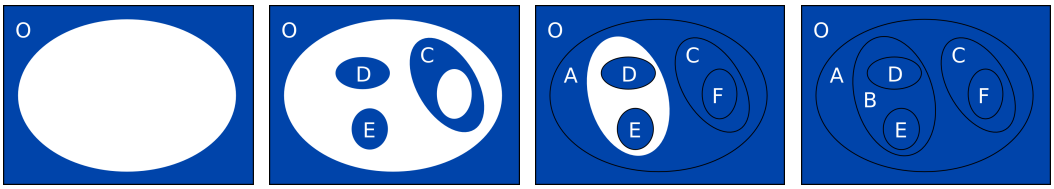


(b) Its 3D representation.



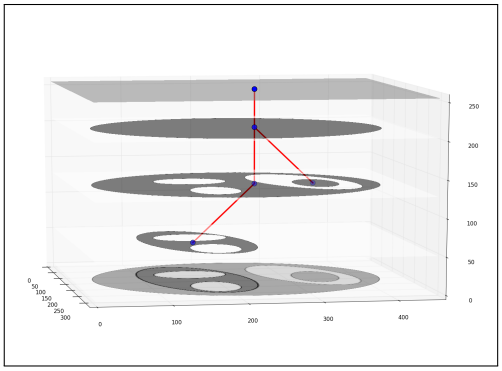
$[u \leq \blacksquare]$ $[u \leq \square]$ $[u \leq \square]$ $[u \leq \square]$

(c) Lower level set.

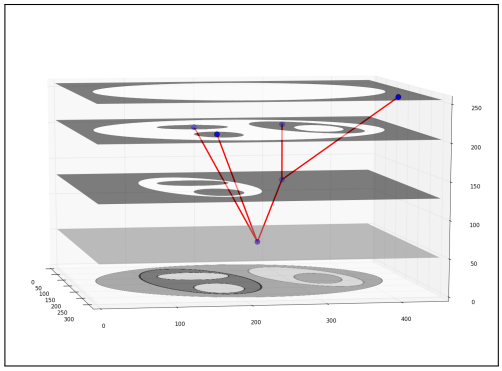


$[u \geq \square]$ $[u \geq \square]$ $[u \geq \square]$ $[u \geq \blacksquare]$

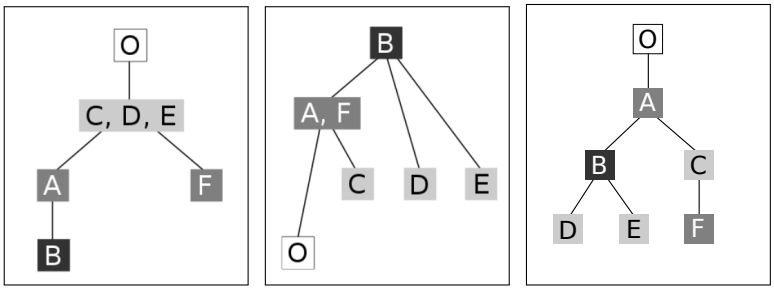
(d) Upper-level set.



(e) Inclusion relation of lower level sets.



(f) Inclusion relation of upper level sets.



(g) The Min-tree. (h) The Max-tree. (i) The Tree of Shapes.

Figure 1.7 – An image and its Min-,Max-trees and ToS.

Max-tree, with the sorting step replaced so that images pixels are placed in a top-down browsing order of the ToS. To enable that, the image is represented as a set-valued map on a Kahlimsky grid to describe the inter-pixel space. For a low quantized image, this algorithm runs in $O(kD)$.

1.3.3 Tree of Shapes for multivariate image

The Tree of Shapes requires the image value space to be totally ordered. It is trivial for a scalar image, but in case of multivariant images, e.g., RGB images, a total order that fits the rule of color perception is not apparent, hence making the extension from gray-scale to multivariate image challenging. In practice, there are two approaches to handle multivariate image: marginal and vectorial. The vectorial approaches define an ordering on the vectorial value space while the marginal ones process each scalar channel of the image independently. A comparative review of these aspects could be found in [2].

We are interested in an extension of the ToS to color images by Carlinet and Géreaud [16] which feature marginally the same properties of the gray-level ToS. Instead of imposing an arbitrary ordering to the value space, they only rely on the inclusion relationship between marginally computed shapes. Carlinet's Multivariate Tree of Shapes (MToS) computation is summarized in Figure 1.8(a). It could be divided into two main step:

- **Computation of the Graph of Shapes through merging marginal ToSs** : The input image I is decomposed in individual channels $I_1, I_2, \ldots, I_n$, which could be treated as scalar image for the computation of their marginal ToS $T_1, T_2, \ldots, T_n$. The Graph of Shapes of I the cover graph of the poset $(\mathcal{G}, \subset)$, with $\mathcal{G}$ obtained from the union of marginal ToS $\mathcal{G} = \cup T_i$. Because regions of two different tree are not guaranteed to have the nested or disjoint properties, the Graph of Shapes no longer has the tree form.

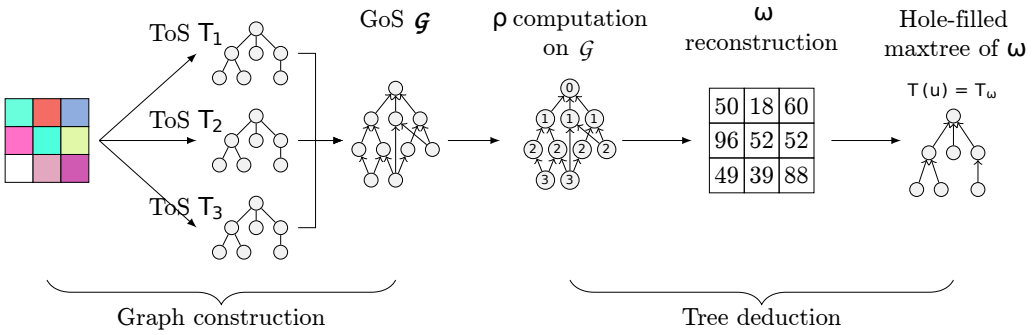
- **Deducing an MToS from the GoS**: In this step, some elements of $\mathcal{G}$ are merged, and then their holes are filled so that the final set forms a valid ToS. First, we weight each element of $\mathcal{G}$ by a decreasing shape attribute ρ i.e $\forall x, y \in \mathcal{G}, x \subset y \Rightarrow \rho(x) > \rho(y)$. This attribute was chosen to be the depth, i.e., the longest path to the node represent the whole image domain. Then we perform the merging process. If two elements of $\mathcal{G}$ have the same attribute and intersect but are not nested, they will be replace with their holes filled union: $\forall x, y \in \mathcal{G}, x \cap y \notin \{\emptyset, x, y\} \Rightarrow \mathcal{G} = \mathcal{G} - \{x, y\} \cup \{sat(x \cup y)\}$. Because of this merging strategy, the MToS may contain shapes that do not exist in any of the marginal trees. The cover graph of the final set has been proved to be a tree. The authors implement this by three small step:

  - **Computation of the decreasing shape attribute** ρ. The authors choose the depth amongs three proposed attributes because it is the fastest to compute.
  - **Create a** ρ **map from** $\mathcal{G}$ for every point of I. The deep map is defined by

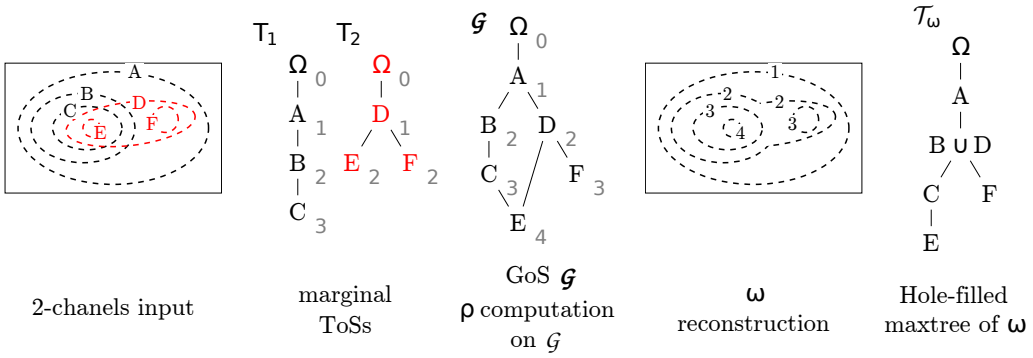
$\omega(x \in D_I) = \max_{X \in \mathcal{G}, x \in X} \rho(X)$, in other word, each pixel is map to maximum depth of shapes that contains that pixel.

– **Obtain MToS** by the hole filled max-tree of the depth map ω: $MToS = \{\mathrm{Sat}(\mathcal{CC}([\omega \geq h]))|h \in (N)\}$. The hole filling Sat operator ensures that components are valide shapes. The associated value for each shapes on $MToS$ is chosen to be the average vector value from the original image.

Although the original image could be obtained from the GoS, by merging (and hole filling) element of $\mathcal{G}$, the MToS lost information. Therefore the MToS is not an equivalent representation of the image. However, the MToS is still interesting enough to be considered since the merging process is done "in the most sensible way as possible" and the authors have demonstrated its usefulness in various application [16].

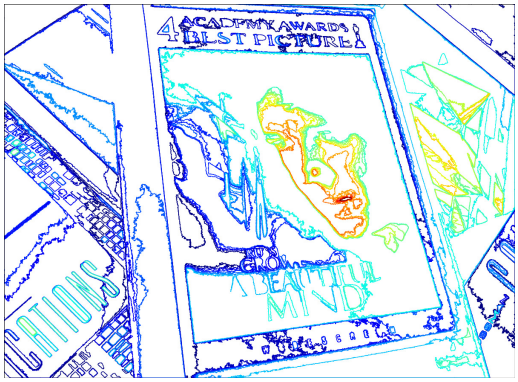


(a) Computation steps of MToS.



(b) A simple example of MToS.

(c) Input.



(d) Some meaningfull level lines from the MToS

Figure 1.8 – Illustration of Carlinet'sMToS. 1.8(a) shows the 5-steps of MToS construction. (1) Marginal ToS computation, (2) Merges ToSs to create GoS, (3) Computation of a decreasing shape attribute, (4) yields an attribute map ω, (5) obtain the final tree. 1.8(b) shows the computation process with a simple 2 chanel image.1.8(d) show some meaningful level lines extract from a MToS computed on 1.8(c).

1.4 Connected Operators

In this section, we will review a class of morphological operators called connected operators and its popular implementation based on tree filtering. The connected operators [39, 83, 82, 84], also called attribute filters [9] or connected filters [41], is a class of morphological operators based on attributes rather than on structuring elements. The most interesting about connected operators is that they can remove boundaries but will not add

new nor shift existing ones.

1.4.1 General definition

In general, the connected operators work with a set of connected components (C) instead of individual pixels as in the case of classical morphological operators. They act by preserving or by removing some element in (C) [115].

These operators are first defined for binary image X where the set of connected components can be divided into two classes: objects and background. An operator Ψ working in X is said to be connected if the set of different $X \setminus \Psi(X)$ is exclusively composed of connected components of X or its complement X^C [39]. This implies that Ψ preserve or remove connected components of foreground and background of X. The class of morphological filters called "filter by reconstruction", e.g., opening by reconstruction, is an early example of binary connected operators.

To extend this notion for grayscale image Salembier and Serra [83] replies on the elemental connected components: flat zones. An operator Ψ is connected if the partition of flat zone of the input image X is finer than the partition of flat zone of the output: $\mathbb{FZ}(X) \leq \mathbb{FZ}(\Psi(X))$. By this definition, regions of the output partition are a union of regions of the input partition. As a result, connected operators cannot create any new boundary and keep the location and shape of preserved ones. Because of this, connected operators have good contour preservation properties.

The connected operators are usually considered as a filtering tool because they transform an input grayscale image into a filtered grayscale image. Moreover, because the conception of grayscale connected operators relies on the notion of partition, they often claimed to bridge the gap between classical filtering and segmentation [41, 27]. Some theoretical notions about connected operators have been extended to pure segmentation applications [85]. This approach is known as connective segmentation.

One of the successful implementations of these connected operators is based on the reconstruction process, which is reviewed in [84]. Another popular implementation of these operators replies on transforming an image into a hierarchical representation, e.g., Min/Max-tree, α-tree, BPT [82, 81]. This efficient implementation is the main focus of the following section.

1.4.2 Tree-based implementation of connected operators

In this implementation, the image is first transformed into a tree-based image representation such as those reviewed in Section 1.2 and 1.3. These trees are equivalent representations in the sense that the original image could be reconstructed from its associated tree. The choice of a tree depends on the input image and the application so that interesting connected components present in that tree. The tree is then filtered by an attribute (criterion) that tell which node to be preserved and which one to be removed. The filtering

image is then reconstructed from the filtered tree. The schematic overview is depicted in Figure 1.10

1.4.2.1 Increasing and non-increasing attributes

After the tree construction, an attribute function $\mathcal{A}$ is designed to weight some interesting feature of nodes on the tree. Such attribute could be as simple as the gray level at which the node first appears or its distance from the leaves. On the other hand, $\mathcal{A}$ could also be a complicated function that measure "shape attributes" which do not depend on the value of the points inside the connected components such as circularity, compactness or elongation [110] or other measures based on the whole connected components such as average gray levels.

The attribute function are distinguished based on whether they are increasing or not because it affects the tree filtering strategy. On a hierarchical represenation (T) of image f, an attribute function $\mathcal{A}$ is said to be increasing if $\forall n \in (T), \mathcal{A}(n) \leq \mathcal{A}(par(n))$. Some increasing commonly use attributes include:

- Area: number of point in n.
- Height: $\underset{p\in n}{Max}(f(p)) - \underset{p\in n}{Min}(f(p))$.
- Diameter of maximum inscribed circle and minimum covering circle of n [9].

However, many interesting attributes are not increasing, especially those "shapes attributes". Some examples are given as follows:

- Perimeter $P(n)$.
- Compactness [60] ($\frac{4\pi Area(n)}{P^2(n)}$) .
- Elongation [110]: ratio of major to minor axes of the minimum covering ellipse of n.
- Maximum geodesic distance [60]

1.4.2.2 Tree filtering and reconstruction

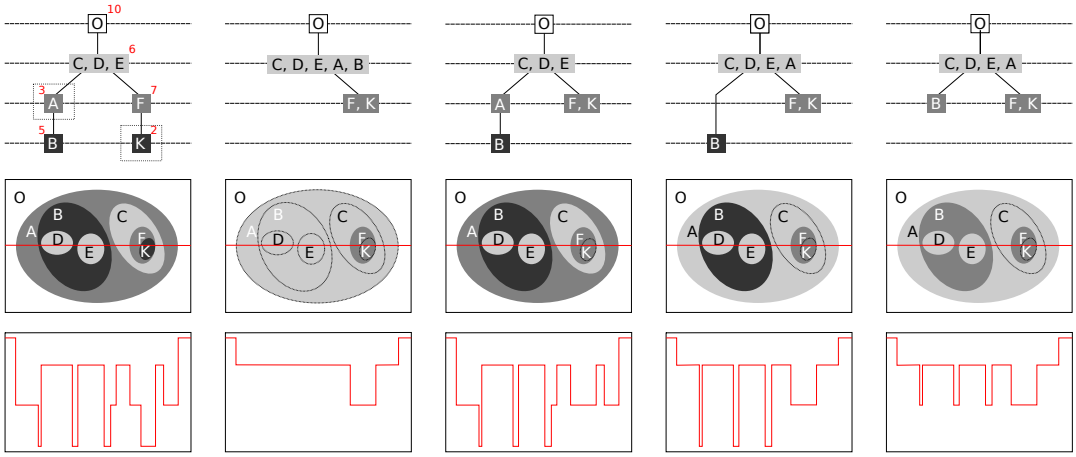
The filtering process will remove some nodes on the tree based on $\mathcal{A}$. Depend on the filtered nodes, tree filtering could be divided into two classes: tree pruning and no-pruning strategies. Tree pruning consists of removing the whole sub-trees rooted in some nodes. If a node is filtered, then all of its descendants are also filtered. The idea is to eliminate image components represented by leaves and branches (e.g., image extrema in case of ToS or union of the most similar flat zones in $\alpha-$tree or BPT). In contrast, with a non-pruning strategy, descendants of a filtered node may be preserved. When a node is removed, its contents (e.g., its points, children) are merged with its nearest preserved ancestor.

When the attribute function $\mathcal{A}$ is increasing, tree pruning strategies is straight-forward. If a node does not pass the threshold, neither do its descendant. In case the attribute $\mathcal{A}$ is not increasing, several tree filtering strategies have been proposed which include three tree pruning strategies: Min, Max, Viterbi [82]; and a non-pruning one: attribute thresholding [82, 112]. Given an attribute threshold t, these filtering strategies are described as:

- **Min**: A node is removed if $\mathcal{A}(n) < t$ or if there exists one of its ancestors $anc(n)$ that $\mathcal{A}\left(anc(n)\right) < t$.
- **Max**: A node is removed if $\mathcal{A}(n) < t$ and for all of its descendants $des(n)$ that $\mathcal{A}\left(des(n)\right) < t$ holds.
- **Viterbi**: The filtering is determined by a cost optimization process using Viterbi algorithm [105]. From a leaf to the root, each transition of decision is assigned a cost. The minimal path cost for each leaf is then chosen. This strategy is a tree pruning one because the cost of keeping a node while removing its parent is infinity.
- **Attribute thresholding**: A node is removed if and only if it does not pass the threshold, other nodes are preserved. This approach is simple and straightforward since the decision is made locally. Based on the image reconstruction rule, two tree filtering rules are defined: direct and subtractive.
  - **Direct rule** [82]: when a node's contents merged with its ancestor, all of its descendants retain their value. This rule is straightforward. However, the local contrasts of reserved nodes will change. This may cause problems, for example filtering a Tree of Shapes using direct rule may lead to a reconstructed image that does not correspond to the tree from which it is reconstructed. This could be resolved by using the subtractive rule.
  - **Subtractive rule** [9, 112, 101] aims to preserve the contrast between the remaining components. In this approach, the value difference between the removed node and its parent is passed to its descendants.

Some examples of filtering strategy are given in Figure 1.9

The image reconstruction from the filtered tree depends on the type of tree that we are working with. The filtered image is guaranteed to be coarser than the input image because the tree filtering process only merged existing nodes.



(a) Input (b) Min (c) Max (d) Direct (e) Subtractive

Figure 1.9 – Tree filtering strategies. First row: input and filtered Min-trees, Second row: Input image and reconstructed images from the corresponding tree. Third row: image function at the red line on the second row. The first column is the input image with its Min-tree. Nodes on the Min-tree are weighted with the red value shown on the corner of the node. It is obvious that our attribute is non-increasing. We set the threshold $t = 4$ so that only nodes 'A' and 'K' does not pass. Min and Max are pruning strategy since they remove the whole subtree. On the other hand, direct and subtractive only remove nodes that do not satisfy the threshold. We could observe in the reconstruct images the direct strategy preserves the local value of preserved nodes but alter the local contrast while the subtractive strategy does the opposite.

1.4.3 Tree-based shape-spaces connected filtering

In [115], Yongchao Xu et al. propose another approach on connected filtering which they called the tree-based shape-spaces connected filtering. A schematic overview is presented in 1.10. Roughly speaking, the filtering step of the classic tree-based connected operator implementation is replaced by a pruning process on the second tree, which is constructed from the first one. This approach is more flexible and brings new possibilities than filtering strategy in Section 1.4.2.2.

1.4.3.1 First tree construction

Similar to the tree-based connected operator implementation above, a tree-based image representation of the image will be obtained. The tree choice depends on the application so that objects of interested appears as nodes of that tree. We will refer to this tree as the first tree T. An attribute $\mathcal{A}$ weighted each node of T. In the earlier section, T is then filtered by thresholding $\mathcal{A}$. However, Yongchao Xu argues that more than one object of interest may appear on one branch of the tree so a pruning strategy could not adapt. On the other hand, attribute thresholding strategies only takes into account local parameter of a single node but ignore the tree structure. Moreover, a single threshold may not be enough to retrieve all relevant objects.

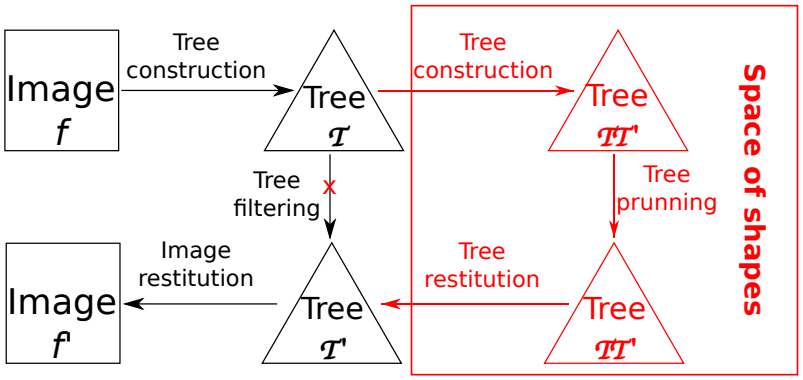


Figure 1.10 – Schematic overview of tree-based connected operators implementation [82, 112] (black path) compare to tree-based shape-spaces connected operators [115] (black and red path).

That thresholding problem on T, in a way, is the same as the problem when thresholding image that leads us to tree-based image representation in the first place. Yongchao Xu's solution is to construct a second tree from T so that we could analyze it in a structured way.

1.4.3.2 Second tree construction

As we explained in Section 1.1.2.3, T is usually presented as a graph [1] $T(V, E)$ with the set of nodes V is the set of regions (shape) and the set of edges E represented the parenthood relationship. Just like how the first tree T is constructed from the graph representing the input image, the second tree TT is built from T based on the attribute $\mathcal{A}$. TT is chosen to be either a Min-tree or a Max-tree. The choice of TT is based on the nature of $\mathcal{A}$. For example, if we would like to remove non-desired shape and $\mathcal{A}$ reflex how likely a node is what we are looking for, a Min-tree of T would be ideal since all the local minimum will become leaves of TT. These local minimum (on T) are more likely to be unwanted shapes in comparison to their neighbors.

1.4.3.3 Second tree filtering

A second attribute $\mathcal{AA}$ is introduced to characterize nodes on TT. That second attribute is chosen to be increasing, so the second tree filtering becomes a simple tree pruning. $\mathcal{AA}$ could the $\mathcal{A}$ range of each node so that could be computed incrementally during TT construction or based on the part of the image domain that each node represents.

Depend on the application, the tree filtering would be a pruning strategy or a branch reserving one. The pruning strategy removes subtrees rooted at nodes that do not pass the threshold. The branch preserving strategy, in contrast, remove nodes that are closer to the root node of TT and preserve small branch near leaves.

1. That graph is acyclic and connected, hence it is a tree

1.4.3.4 Tree and image restitution

The simplified tree is obtained by removing nodes corresponding to filtered nodes on TT and update the parenthood. The filtered image could be then deduced from the simplified tree. The image-restitution could be similar to direct rule to preserve the local value of remaining nodes or similar to the subtractive rule to preserve the local contrast.

1.5 Text Detection on Natural Image

This section presents the scene text understanding problem which is the main application subject of our work. In this section, we provide information about the problems, its applications, and challenges.

1.5.1 Text in images and challenges

In document image analysis, the extracted information could be divided into two categories: textual information (which are text elements) and graphics (symbols, diagram, logo) [72]. Text in images can be later divided into two classes: born-digital text and natural scene text [45]. Born-digital text (Figure 1.11(a)) includes text in digital images or graphically added into an image, e.g., on overlay captions, subtitles, and notation in videos and images on webs and email. Scene text refers to those that are captured in their natural environment (such as signs, advertising, clothing, vehicle license plates). They could be in focus (Figure 1.11(b)) or incidentally captured (Figure 1.11(c)). The later would be a difficult task, even for the human brain.

Text in images and video contains valuable information. They can provide semantic information and useful features for many content-based images and video analysis tasks. Examples are content-based image search and multimedia retrieval, visual input and access, and industrial automation.

- **Content-based image search and multimedia retrieval:**

(a) (b) (c)

Figure 1.11 – Example of different *text in image* problems. 1.11(a) Born-digital text, 1.11(b) Focused scene text, 1.11(c) Incidental scene text.

Texts in web images and videos are usually relevant to its content. Graphically added captions and captured scene texts usually give information about the situation of the event such as location, people. Text recognition and keyword extraction in these resources improve multimedia retrieval.

- **Visual input and access:**

  With the spread of imaging devices and the improvement of digital processors, a mobile visual input application can be realized. A mobile device can be used to digitize documents or automatically input name cards, bank cards, whiteboards [50]. As the input becomes automatic, user experience and efficiency are improved.

  Since signs, banners, advertises and product labels carry relevant information, automatic text recognition, and translation can help people overcome language barriers or support visually impaired people in daily life.

- **Industrial automation:**

  Text recognition in images of packages, letters, containers, houses, signs, and maps is beneficial for industrial automation. For example, a mail sorting system will benefit addresses recognition. Another example, the automatic identification of container numbers will improve logistics efficiency [33].

Although many approaches have been proposed, and many advances have been made on text detection in digital born images, text detection in scene images is an open and challenging problem which has been receiving much attention. This is a difficult task due to high variations of scene texts such as illumination, contrast, blur, distortion, and variation of text content [50]. They are summarized in Table 1.1 .

Scene complexity: Man-made objects in real life could have similar shapes and presentations to texts. Background complexity also degrades the ability to discriminate text from non-text.

Uneven illumination: Images captured in the natural environment are not always in good illumination conditions. Text elements may subject to shadow, brightness or reflec-

Table 1.1 – Scene text detection chalenges

Category	**Challenges**
Environment	Uneven illumination
	Scene complexity
Acquisition	Optical aberration
	Resolution
	Noise
	Compression
Text content	Variation of orientation and curved text
	Variation of fonts
	Multilingual

tion which leads to color distortions, degradation of visual feature. This result makes the segmentation and recognition tasks difficult.

Optical aberration: Photos are not taken with ideal lenses so optical aberrations might appear. Perspective distortion occurs when the sensor plane and the text plane are not parallel. Other distortions appear such as barrel, pincushion and mustache distortion due to the quality of lenses. They can deform text sharp. Defocus occurs because the focus cannot always be maintained in normal working conditions, e.g., visual aid system for the visually impaired. At flexible working conditions and focus-free cameras, defocusing and blurring of text happen. A moving object or a moving camera can also create motion blur, which is usually present in the video. Chromatic and coma aberration introduce false colors and reduce edge contrast. These optical aberrations make characters lost their sharp and their edges response. Since sharp edges response is required for character segmentation and recognition, blurring effect causes serious problem to the system [50].

Resolution: Images that are taken by cameras usually have low resolution. This problem makes detection and segmentation difficult. While most OCR engines are tuned to the resolution between 150 and 400 dpi, text in a video frame may be at or below 50 dpi [50].

Noise: Noises degrade the quality of images. It is caused by shutter speed, sensor size, and temperature.

Compression: Most images captured by cameras are lossy compressed. The compression process is optimized for general uses instead of document analysis, which means it does not always preserve the topology and sharpness of contents.

Variation of fonts: Italic and script fonts may have overlap characters, which make it is difficult to recognize and segment them [51]. Variation of fonts introduces large within-class-variations, which may challenge approaches which use learning machine.

Multilingual: Languages that use the Latin alphabet have around tens of characters, but other languages such as Chinese, Japanese and Korean have thousands of characters. Other languages have characters shape connected or changed such as Arabic and Hindu. OCR in scanned multilingual documents remains a research problem [93] and it is much more difficult in case of scene images.

1.5.2 Text detection and recognition system

Scene text understanding is to process and extract textual information from natural (i.e., real-world) images into a digital format. This problem is usually divided into two main stage: text detection and text recognition. They could be later divided into sub-problems: localization, validation, extraction, rectification and optical characters recognition (OCR). They are processed individually [113] [52] [21] [120]) or jointly [68]. In these problems, the text detection modules are the most important part. It has been shown to affect the performance of text in image retrieval algorithms critically [21]. Text detection in scene images poses significant challenges to state-of-the-art methods. The research community has proposed numerous dataset to push development in this field, see [117] for

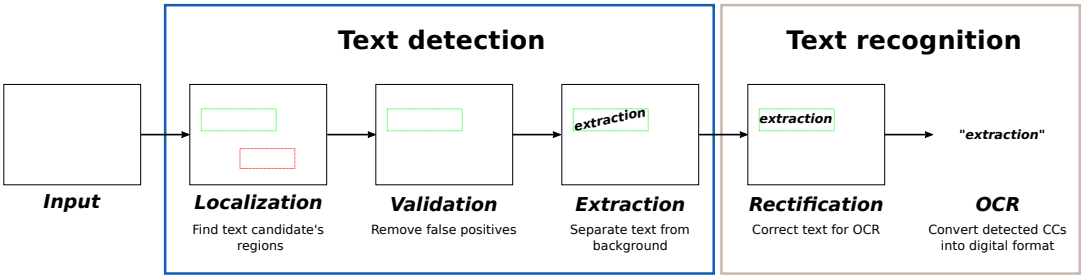


Figure 1.12 – Stages of an end-to-end scene text understanding system.

a summary of commonly used datasets. Notably the ICDAR RRC datasets, which include a database of *Focused Scene Text* and *Incidental Scene Text* [44], as well as the *COCO-Text* dataset, which is the largest scene text dataset currently available [103].

Text localization aims at finding the location of text components. These methods assume that text regions can be regarded as a kind of uniform patterns. In consequence, there are some features that are invariant over this patterns such as color, edges, strokes width and texture. Related works can be divided into three catalogs: texture-based, connected components-based and hybrids methods.

Texture-based methods [113, 48, 42, 19] and deep learning based methods [108, 89, 80, 34] use a sliding window to look for all possible texts in the image. It is based on hypotheses that text regions in images have distinct textural properties from non-text regions (gradient distribution, texture, structure, or learned features) that can be identified with a classifier such as Adaboost [48], Random Ferns [42], linear SVM [19] or neural network [108]. These approaches may be expensive in computation when uses complex classification methods or a large number of windows with different scales need to be analyzed. These approaches are also limited by the training database, and it is sensitive to text alignment orientation [23].

The connected-component-based methods [21, 120, 69, 49] group pixels that have similar properties such as color, luminance or geometrical arrangement of edges, into connected regions. These methods separate the input images into small connected components. The advantage of this approach is the detection of texts regardless of image properties such as scales, orientation, and font type. It also provides a segmentation that can be useful in the OCR step. On the other hand, it might produce a high amount of false positives that usually removed in a validation step. Detected characters are usually grouped together to form higher level detection (words, lines).

The hybrid methods combine both approaches. For example, Pan et al. in [75] uses HOG features and a Waldboost cascade classifier to generate a text confidence map, based on which connected-components extraction is done by a local binarization. Liu et al. [52] is also a hybrid approach since it uses an edge detector on three channels to separate connected components which will be verified using Harr wavelet transform as texture characterization.

Text validation eliminates the false positives introduced by the text localization steps.

They could be divided into knowledge-based methods and feature discrimination methods [117]. The former presumes a priori knowledge of text characters (e.g., size, color or projection profile). The latter makes no assumption on the characteristics of text but relies on "learned" features.

Text extraction, also referred to as segmentation, separates text from non-text at the pixel level. This extraction could also be at character, word or line level. Segmentation is one of the most challenging problems [117].

Text Rectification corrects detected text regions for OCRs. Current OCRs are tuned for horizontal texts. On the other hand, texts in natural images often appear inclined because they are subject to perspective deformation. We could also face curved or vertical texts due to design.

Text recognition converts text candidates into machine-encoded texts. The characters recognition classifies each CC while at a higher level, word recognition usually integrates a language dictionary to predict the detection.

These steps are summarized in Figure 1.12. However, an end-to-end text detection and recognition system may not consist of all these steps or in that orders. For example, a connected components-based method does not require a segmentation step. The validation step may be performed multiple times, during the localization, e.g., by eliminating some CCs and OCR stage. Some methods only focus on the detection stage and are referred to as text detection algorithms.

1.5.3 Text features

Different features were used in text localization including color in different color spaces [119] [56], edges and gradient [77], [113]. New features such as stroke width [98] [21] , corner [114], extremal region [69] [120] or character appearance [118] have been recently studied.

Color: A readable text must have a consistent and distinguishable color which contrasts with its background [50]. Color is widely used as the feature to localize text [37] [46] [49] although it seems to be sensitive to uneven lighting, multi-color and complicate textured texts. Nikolaou $etal.$ [70] simplifies the color images by color reduction, and then clusters and groups connected components into candidates. Some authors use different color spaces to extract and analyze the color feature, such as Hue-Saturation-Intensity (HSI) [26], HSI plus intensity gradient [68] or Hue-Lightness-Saturation [43]

Edge and gradient approach: They assume that texts have strong edges against their background. Therefore edge detector and gradient map can be used to detect character candidate. Edge detectors and gradient [78] [77] [91] was used widely in the connected component. Sometimes they are used as a feature in sliding window-based approaches [113] [32].

Stroke width: From the assumption that text stroke is consistent within a character, Epshtein et al. proposed the stroke width transform (SWT) [21]. The SWT returns a map

which shows for each pixel the stroke width of the stroke to which it most likely belongs. Recently, Mosleh et al. [62] showed that SWT could be improved by introducing a bandlet-based edge detector which enhances text edges and dismisses noisy and foliage edges.

Corner: With assumption that dense presences of corner points in text region, [114] [35] used Harris corners detector as the feature. Text candidate will then be detected using connected components-based approach [35] or analysis with a decision tree classifier.

Extremals region has been widely explored in [68] [120] [90]. Using the same assumption that text components usually have significant color contrast with backgrounds and tend to form homogenous color regions, MSER algorithms adaptively detects stable color region to localize them as text candidate. The effectiveness of using MSERs as character candidates is the main advantage of this approach.

Chapter 2

Text Localization and Segmentation With Tree of Shapes of Laplacian Sign

2.1 Introduction

With the dramatic increase of images and video acquired with mobile devices, content-based analysis techniques have received a great deal of attention over the last few years; this is, in particular, the case of text detection. In this chapter, we focus a simple version of ToS that targets application of text localization and segmentation in natural images, that is, finding candidate components for text characters; see Fig. 2.1 for an illustration. To accomplish this task, we propose a hierarchical image representation based on the morphological Laplace operator, also called morphological Laplacian, with an application that segments text characters, and groups them into text boxes/lines thanks to two kinds of spatial relations: adjacency and inclusion. The contributions of this work are the following:

- a hierarchical (i.e., tree-based) representation of the image contents, where adjacency between components is related to inclusion;
- a character segmentation method which is a good trade-off between efficiency (linear time complexity) and quality (with a competitive F-score);
- an efficient grouping of characters into text boxes, taking full advantage of the tree structure;
- an illustration on another application (document binarization) of the capabilities of the proposed tree-based representation.

In the first part of this chapter, we focus on two points: how we can rely on some discrete topology tools to get *a sound definition of a hierarchical decomposition* of an image into regions, and how we take benefit from some properties to get *an efficient algorithm* to compute such a representation. In the second part, we address the application of this representation in text localization and segmentation. This method is interesting for several reasons:

1. To select candidate regions for characters, we rely on the morphological Laplace operator. We can observe that this operator is relatively well robust to poor contrast and

uneven illumination in Section 2.2.1.

2. Based on the 0-crossings of the morphological Laplace operator, we compute a hierarchical representation of the image contents, so that regions form an inclusion tree. Such a structure is attractive because we have a strong simplification of the image contents, and because dealing with a simple tree structure allows for some powerful decision taking when grouping regions/characters into text boxes. Although using the 0-crossings of the morphological Laplace operator as a contour detector is not new [106], getting such a hierarchical representation from it is a novelty.

3. This text segmentation method is suitable for real-time implementation on mobile devices. Indeed computing the hierarchical representation eventually translates to a simple labeling algorithm that gives both an inclusion tree and a label image; in addition, identifying and grouping text components are achieved by an easy tree-based processing.

4. As compared with many methods of the literature, the method that we propose in this chapter features many properties: it is invariant to contrast inversion (so we also extract reverse-video text without any special processing); it is invariant to contrast change; it is invariant to scale and rotation; and it handles a large variety of scripts (Latin, Hebrew, Chinese, etc.). Although some apparently irrelevant components were filtered out, this study does not cover the whole text detection pipeline. However, according to the quantitative results of the proposed text candidate, our method outperforms some widely used text component extraction methods.

In Section. 2.2 we recall the morphological Laplacian and how that leads to the inclusion tree of 0-crossings. Then, an effective algorithm to compute the hierarchical structure is presented in Section. 2.3. In Section. 2.4 we detail on how we rely on the ToSoL to regroup components. In Section. 2.5 we proceed to experiments and show that we compete with classical component-based text segmentation methods. Last we conclude and give perspectives in Sec. 2.6.

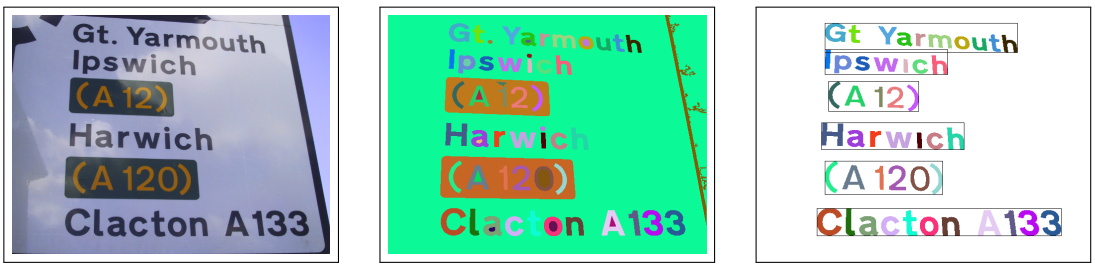


Figure 2.1 – A hierarchical image decomposition (center) leading to text detection (right).

2.2 A Tree of Shapes of Laplacian Sign (ToSoL)

2.2.1 Morphological Laplace operator

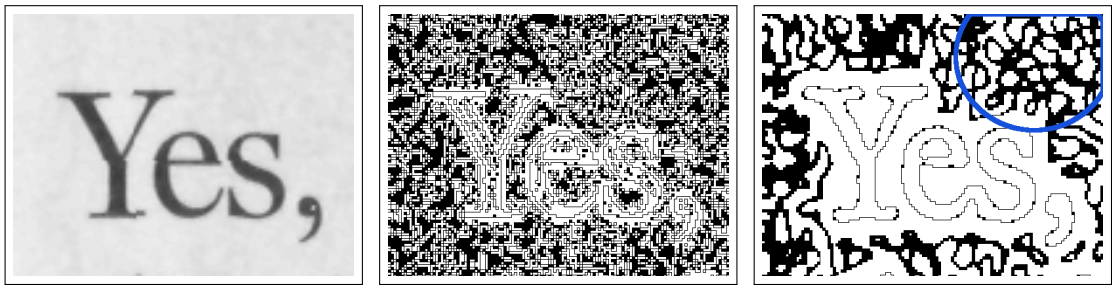
A seminal method to detect objects in an image is looking for their boundary with the background and other objects using contour detection. That boundary could be as simple as the 0-crossings of a discrete Laplace operator: given a gray-level image u, the contours of interest are given by $\Delta u = u_{xx} + u_{yy} = 0$. This method is interesting for several reasons: **1.** it is a very simple approach; **2.** it provides closed contours; **3.** labeling the components of the image having the same sign, resp. positive and negative; gives a segmentation; **4.** it is self-dual, *i.e.*, it processes dark objects and bright ones the same way.

The simplest discretization of this linear operator relies on a cross-shaped convolution kernel. Yet this elementary operator is very sensitive to noise, so many 0-crossings arise as it can be seen in Figure 2.2(b). To get rid of this problem, one can rely on a larger kernel, e.g., by considering the approximate given by the Laplacian of Gaussian (LoG) operator. Unfortunately, its smoothing effect alters the location of contours, as illustrated by Figure 2.2(c).

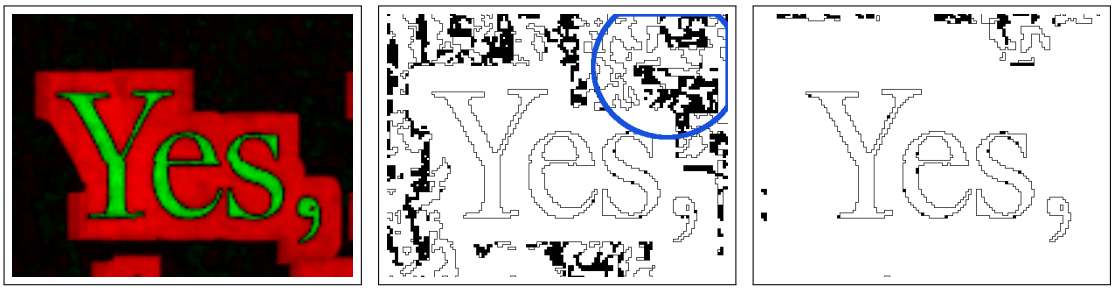
The morphological Laplace operator has been defined in [107] by $\Delta_{\mathcal{N}} = (\delta_{\mathcal{N}} - \mathrm{id}) - (id - \varepsilon_{\mathcal{N}})$, relying on the elementary dilation (δ) and erosion (ε) morphological operators. A natural extension of the elementary operator uses a structuring element B to replace the neighborhood $\mathcal{N}$; it is depicted in Figure 2.2(d) with B being a centered square (denoted by $\square$) of size 17×17. Although it has the same "simplification strength" as the linear LoG version, one can see when comparing the resulting 0-crossings (LoG in Figure 2.2(c) vs. morphological in Figure 2.2(e)) that the morphological non-linear version features a much higher fidelity to actual object contours than the linear version. Furthermore, when increasing the size of the structuring element, 0-crossing contours keep a strong fidelity to data, as illustrated in Figure 2.2(f) with B now being a 51×51 square.

One can also observe that the salient object contours are curiously very stable—they are not altered—when the size of the structuring element (the "morphological kernel") increases. From Figure 2.2(e) to Figure 2.2(f), the contours of the "Yes" word remain the same, whereas spurious non-interesting contours disappear. Furthermore, the size of the structuring element B, a 51×51 square, is much larger than the character thickness (note that the input image u has 130×100 pixels). It means that obtaining salient contours, thanks to the morphological Laplace operator hardly depends on the size of the parameter B. Despite this great advantage, the morphological Laplacian has been almost never used in the literature [95, 66].

Contours obtained by the 0-crossings also shows high resistance to uneven illumination. The image depicted in Fig. 2.3(a) has been created by the authors of [5] to make classical binarization methods fail, due to the presence of uneven illumination. On Fig. 2.3(c), one can observe that the object boundaries belong to the 0-crossings of the morphological Laplace operator.

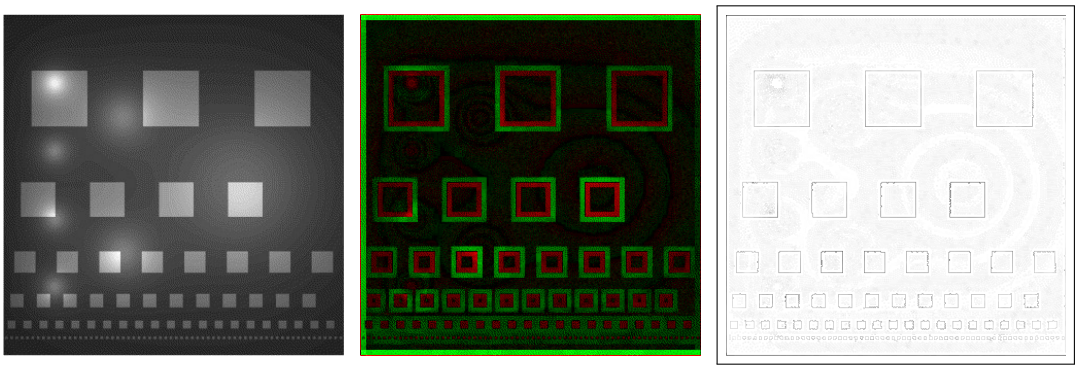


(a) Input u. (b) $\Delta_4(u) = 0$. (c) $\text{LoG}_{17}(u) = 0$.



(d) $\Delta_{\square_{17}}(u)$. (e) $\Delta_{\square_{17}}(u) = 0$. (f) $\Delta_{\square_{51}}(u) = 0$.

Figure 2.2 – Zero-crossing contours of different Laplace operators: (b) and (c) come from classical linear operators; (e) and (f) come from the morphological operators. On (d), the scalar morphological Laplacian is depicted with positive and negative values tinted resp. in green and red.

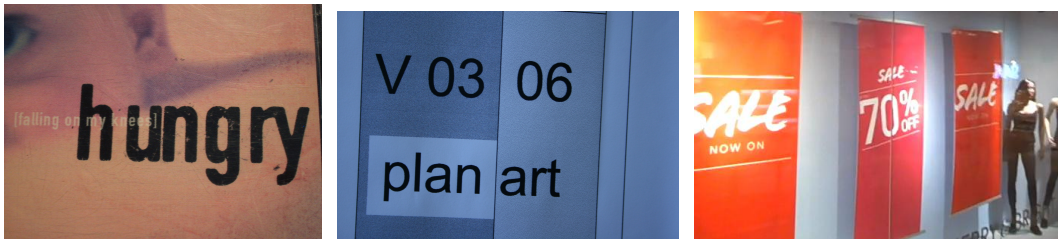


(a) Input. (b) Laplacian. (c) 0-crossings.

Figure 2.3 – Contour characterization by morphological operators; 2.3(b) is the morphological Laplacian $\Delta_B = \delta_B + \varepsilon_B - 2\text{id}$; 2.3(c) depicts on $\Delta_B = 0$ the gradient values $\nabla_B = \delta_B - \varepsilon_B$ (inverted) showing that the contours of actual objects are effectively salient.

2.2.2 Relativity of the objects-background notion

The 0-crossings of morphological laplacian do highlight regions of interest in images. From here, we could continue with the classic object detection scheme: filter these 0-crossings to keep only wanted objects, the grouping process will ignore all other regions as they are the background. However, as will be explained below, the objects-background notion depends on and imply the context, treating all unwanted regions as a single background



(a) (b) (c)

Figure 2.4 – The notion of objects-background is highly contextual. On the other hand, that notion brings more context to the image. In image 2.4(a), the word “hungry” could be considered as objects or part of the background (if we focused in “[falling on my knees]” parts). In contrast, that notion carries contextual information. In images 2.4(b) and 2.4(c), words and numbers in different backgrounds are implied to be separated.

leads to a loss of information.

The objects-background notion in images sometimes gets fuzzy. Objects sometimes become the background of other objects. This could be explained by the highly contextual of the objects-background notion. To illustrate this statement, let’s look at an image taken from ICDAR Robust Reading Competition. In the color image 2.4(a), if we focus on the bigger word “hungry” then all the outer region including some lighter words will be the background. Furthermore, we can choose the fainter sentence “[falling on my knees]”, therefore the “hungry” regions should be treated as part of the background.

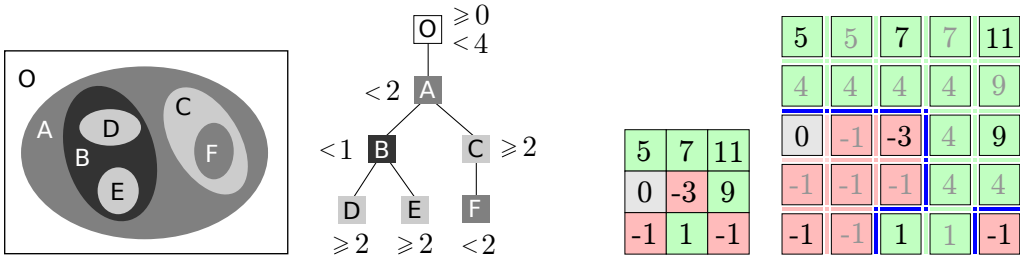
On the other hand, the objects-background notion often carries contextual information. Objects on the same background usually have a closer connection. Numbers and words in image 2.4(b) should be grouped and process based on their different background. Or words on red advertise stands in image 2.4(c) should be grouped into threes groups based on which banner they are on. That contextual information would be lost if we only focused on the relevant objects.

We consider that the inclusion of the 0-crossings parallels the objects-background notion. For that reason, we are interested in encoding the inclusion in a hierarchical structure: the Tree of Shapes of Laplacian sign (ToSoL).

2.2.3 A Tree of Shapes of Laplacian Sign

The hierarchical nature of the morphological laplacian makes it suitable to arranged into a tree of shapes, which encodes the inclusion of the level sets, i.e., the connected components whose border is a level-line. For a reason explained in an earlier section, we are only interested in one type of level-line, the 0-crossing of Laplacian. We call this representation the Tree of Shapes of the Laplacian *sign* image (ToSL), $\mathfrak{S}(\mathrm{sign}(\,\Delta^{\mathrm{wc}}_{\square}(u)\,))$, which encodes only level-line related to 0-crossings. It contains nodes corresponding to positive, negative, and null regions. It is depicted in Fig. 2.7(d).

Yet the 0-crossings, corresponding to nodes at level 0 (depicted in white), can be “thick”, that is, they can contain pixels (2-faces) instead of being only defined by 1D



(a) The tree of shapes of an image. (b) Level set contours as Jordan curves.

Figure 2.5 – Topological representations and some topological considerations.

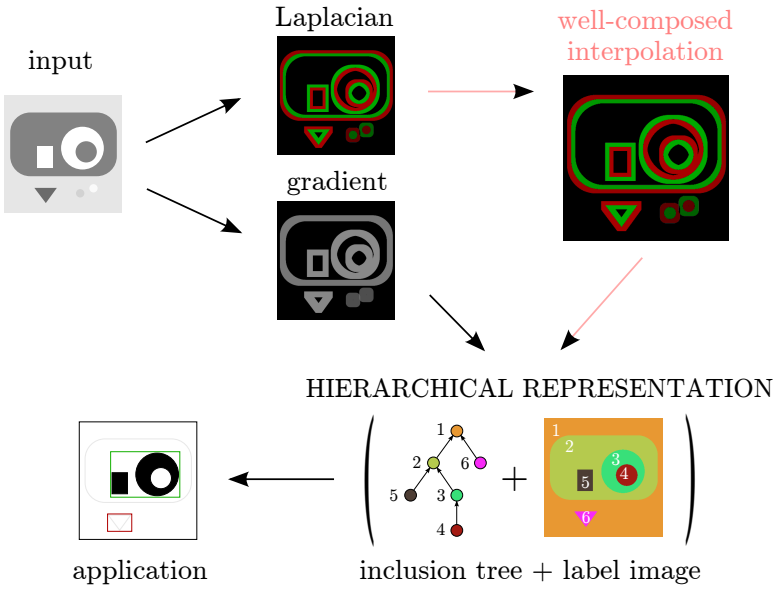
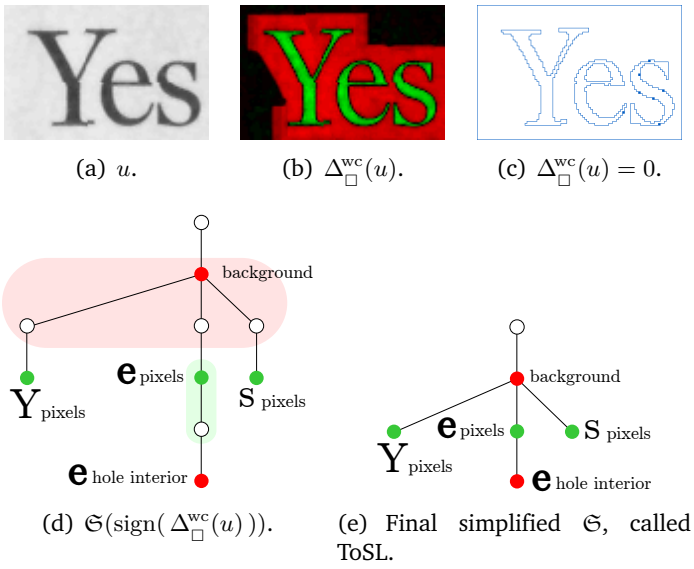


Figure 2.6 – Overview of the proposed method to get a hierarchical image decomposition.

objects (set of 0-faces and 1-faces). In Fig. 2.7(c) for instance, we can see that some pixels of the 's' contour belong to the 0-crossing region. Since we want to obtain a partition of the image into "positive" and "negative" regions, we merge null regions with their parent regions; that way, instead of the tree of shapes depicted in Fig. 2.7(d), we consider the simplified one, depicted in Fig. 2.7(e). Eventually, the actual "0-crossings" we are looking at are the boundaries of the shapes of this final tree, so they are effectively 1D objects. It is illustrated in Fig. 2.5(b) : the null region is merged with the negative one, so we consider the blue contour to be the 1D "0-crossing" separating regions having different signs.

To compute the hierarchical representation, we do not consider that we have a cubical complex as the space structure: we just ignore that 0-faces and 1-faces exist. Though, from a theoretical point of view, the contours / 0-crossings expressed in terms of 0-faces and 1-faces really are Jordan curves.

There exists effective algorithm to compute the ToS for scalar image [29] which would be applicable to the Laplacian map. Because we only interested in the 0-crossings for reasons explained earlier, therefore, the ToS computed from Laplacian map have to be



(a) u. (b) $\Delta^{\mathrm{wc}}_{\square}(u)$. (c) $\Delta^{\mathrm{wc}}_{\square}(u) = 0$.

(d) $\mathfrak{S}(\mathrm{sign}(\,\Delta^{\mathrm{wc}}_{\square}(u)\,))$. (e) Final simplified $\mathfrak{S}$, called ToSL.

Figure 2.7 – Tree of shapes of Laplacian sign (ToSL): positive and negative regions are respectively green and red nodes of the ToS, and null regions are white nodes.

simplified to keep only shapes corresponding to 0-crossings. This approach is computation heavy because we have to process unnecessary level-line. Consequently, we decide to develop a new approach that is not only adapt to the construction of ToSoL but also integrate part of the tree simplification step. The construction of ToSoL from the input image is straightforward. The method is composed of several steps, depicted in Fig. 2.6, and described and justified just below:

We start with a gray-level input image. If the primary data is a color image, we just take the luminance value of its pixels (we lose color information, but it almost never negatively affects text retrieval). We then compute its morphological dilation $\delta_{\square}$ and erosion $\varepsilon_{\square}$ to directly deduce two images. First, we obtain the morphological thick gradient $\nabla_{\square} = \delta_{\square} - \varepsilon_{\square}$; it is used later to discard contours that are not enough contrasted. Second, we obtain the morphological Laplace operator $\Delta_{\square} = \delta_{\square} + \varepsilon_{\square} - 2\,\mathrm{id}$ (where id is the identity function). The components' boundaries are expected to belong to the 0-crossing contours of this non-linear operator (actually they are, and their localization is precise).

We *virtually*[1] compute a particular interpolated image, $\Delta^{\mathrm{wc}}_{\square}$, of the Laplacian image $\Delta_{\square}$, having 4 times more pixels than the original. This resulting image is *well-composed*, meaning that the boundaries of every component of any threshold set are Jordan curves. As a consequence, the (boundaries of the) 0-crossings are simple closed curves: they cannot have the shape of an '8'. In addition, they are disjoint, and this set of curves can be organized in an inclusion tree.

Due to the fact that $\Delta^{\mathrm{wc}}_{\square}$ is well-composed, the regions delimited by the 0-crossings can be labeled very efficiently (by the classical blob labeling algorithm), and their inclusion

1. We will see later that we actually do *not* interpolate the Laplacian image, but proceed *as if* there were an interpolation. Practically, it means that we avoid the need of multiplying by 4 the number of pixels in the process.

tree is built. In addition, many 0-crossings are discarded on the fly during the labeling process, because they are not contrasted enough (based on $\nabla_{\square}$), or because they do no satisfy some geometrical criteria (*e.g.*, when they are too small). The resulting "inclusion tree + label image" is the hierarchical decomposition of the image contents into regions; it is depicted on the bottom-right part of Fig. 2.6. We end up with two structures: the inclusion tree, encoded by a parenthood relationship between labels and an image of labels so that each pixel is assigned to a region. In addition, we could collect additional information related to every region, that are computed during the labeling process: the region area, its bounding box, etc.

2.3 Fast Computation of the Hierarchical Representation

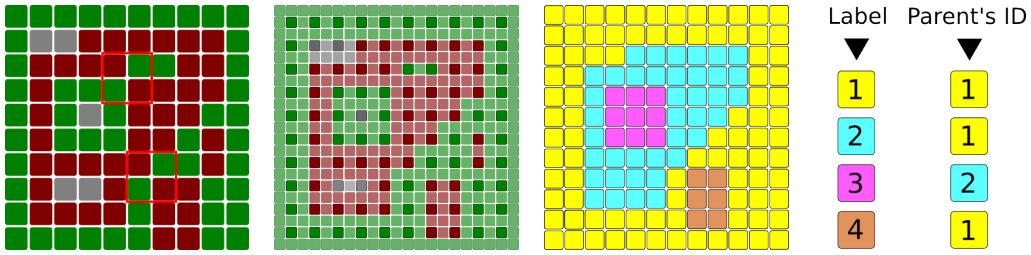
2.3.1 A particular well-composed non-local interpolation

In the following, we do not need to consider the cubical complex, so we only deal with pixels; they are, for instance, the valued 2-faces in Fig. 2.5(b) (right).

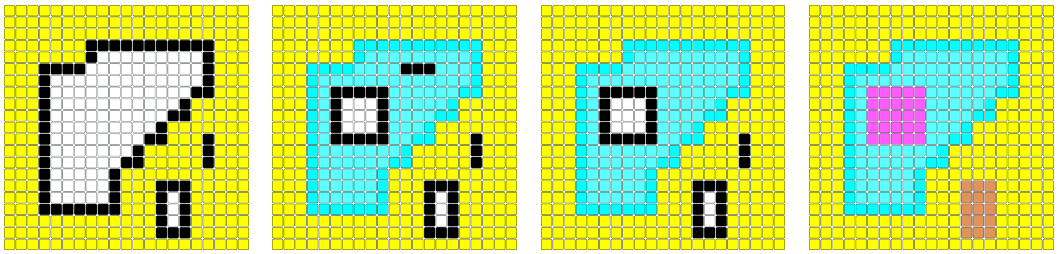
The hierarchical representation is computed from an interpolated image, $\Delta^{\text{wc}}_{\square}(u)$, which is a very particular well-composed version of the Laplacian image $\Delta_{\square}(u)$. This particular interpolation takes its origin from the work in [29], and is detailed in [8]. Briefly put, it is a *non-local* interpolation driven by a propagation from the border of the image, which browses the nodes of its tree of shapes from the root to the leaves. The interpolated pixels are assigned with the current gray-level value, which evolves "continuously" during the process. This interpolation has two important features [28]: it is related to the tree of shapes of the input image, so it actually follows the same scheme as a blob labeling algorithm where a blob would be a level set, and its topological behavior is deterministic.

There are two main consequences: this particular interpolation makes sense in the present context since we want to label regions that are in an inclusion relationship, and we can optimize the computation of the hierarchical representation (the label image and the inclusion tree) by actually *emulating* the interpolation.

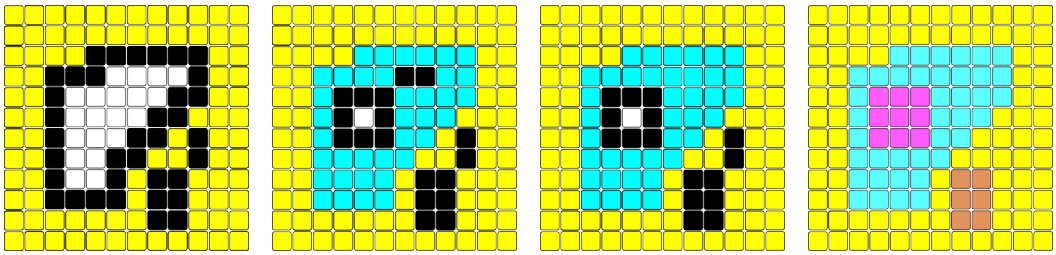
An example on the image of Laplacian sign given in Fig. 2.8(a) is depicted in Fig. 2.8(b)[2], with the signs -1, 0, and 1 respectively depicted in red, gray, and green. We can observe in Fig. 2.8(b) that we obtain the desired properties: the boundaries of the regions are Jordan curves, and the regions are in an unambiguous spatial inclusion relationship. The inclusion tree of the Laplacian sign image, called ToSL, is thus a hierarchical image decomposition.



(a) Sign of $\Delta_{\square}$. (b) Interpolation. (c) Label image. (d) Parenthood.



(e) Different stages of the proposed algorithm based on interpolation.



(f) Different stages of the proposed fast algorithm without using interpolation.

Figure 2.8 – An example of the proposed labeling algorithm. Black pixels: contours of components inside labeled ones. White pixels: pixels that are not yet labeled and are also not marked as inside contours at the current stage of the labeling process.

2.3.2 Label the interpolated laplacian map to construct the ToSL

Thanks to the fact that $\Delta_{\square}^{wc}$ is well-composed, the regions delimited by the 0-crossings can be labeled very efficiently (by the classical blob labeling algorithm), and their inclusion tree is built. This resulting inclusion tree is the tree of shapes of the sign of the Laplacian, a ternary-valued image—with pixels set to -1 (red), 0 (gray), or 1 (green) as depicted in Algorithm. 2.8. The whole labeling process is depicted in Algorithm. 2. It computes a label image $\mathfrak{L}$ (a label is assigned to every region delimited by the 0-crossings of $\Delta_{\square}$), and a tree structure encoded in an array of parenthood $parent$. Having $parent(l_1) = l_2$ means that the region with label l_1 is included in the region with label l_2, and the particular root label, say l_r, is such that $parent(l_r) = l_r$. Q is a queue of pixels, $border$ is an auxiliary image that marks the inner component contours as active a or inactive $\bar{a}$ state, $nlabels$ is the current number of labels, ℓ is the current label value, and $\mathbb{N}$ represents a neighborhood

2. Note that the two identical local configurations enclosed by red rectangles in Fig. 2.8(a) do not lead to the same interpolation; this is due to the non-local interpolation process that depends on the outer region, which is different in the two cases: respectively negative for the top configuration, and positive for the bottom one.

```
1  LABELING (Δ□,∇□):
2      forall p do
3          Ł(p) ← 0;
4          border(p) ← undef;
5      nlabels ← 1;
6      forall p do
7          if Ł(p) ≠ 0 then
8              continue;
9          if p = p0 then
10             ℓ ← 1;
11             parent[ℓ] = 1;
12         else
13             P ← FOLLOW_CONTOUR(p);
14             if evaluate(P) then
15                 nlabels ← nlabels + 1;
16                 ℓ ← nlabels;
17                 parent[ℓ] ← Ł(p−1);
18             else
19                 ℓ ← Ł(p−1);
20         BLOB_LABELING (p,ℓ);
21     return parent, Ł ;
22 BLOB_LABELING (p, ℓ) :
23     Ł(p) ← ℓ; Q.push(p);
24     while Q is not empty do
25         q ← Q.pop(), N ← N4;
26         if border(q) = undef then
27             N ← N8; /*optimized*/
28         forall n ∈ N(q) do
29             if Ł(n) = 0 and
30             Δ□(p) × Δ□(n) ≥ 0 then
31                 Ł(n) ← ℓ; Q.push(n);
32             else
33                 border(n) ← a;
34 FOLLOW_CONTOUR (p) :
35     P.init(); border(p) ← ā; Q.push(p) ;
36     while Q is not empty do
37         q ← Q.pop(); P.update(q);
38         for n ∈ N(q) /* N8 or N4*/ do
39             if border(n) = a then
40                 Q.push(n) ;
41                 border(n) ← ā;
42     return P ;
```

Algorithm 2: Computation of ToSL by labeling interpolated image (black part) and its fast version (adding red part) without using interpolation.

corresponding to either 4-connectivity ($\mathbb{N}_4$) or 8-connectivity ($\mathbb{N}_8$).

The core of the algorithm is an alternate application of the following two routines (depicted on the right side of Fig. 2): BLOB_LABELING is a classical queue-based "blob labeling" algorithm that labels the underlying connected component with current label value ℓ, and also marks the inner component contours as active state a. Note that actually, we do not want regions representing null values in the final tree, so we merge nodes corresponding to 0-crossings with their parent (see line 30 and Fig. 2.7). FOLLOW_CONTOUR is also a queue-based process which is similar to previous "blob labeling" algorithm, but applied on the underlying active contour of an unlabeled connected component. Instead of labeling the active contour, this routine browses it and marks it as inactive $\bar{a}$ (see lines 35 and 41), and collects a measure $\mathcal{P}$ characterizing the contour (*e.g.*, its length). Note that both these two routines are very efficient thanks to the queue-based "blob labeling".

More precisely, for the main algorithm (depicted on the left side of Fig. 2), we browse the pixels in raster scan order (main loop, line 6). When we reach an unlabeled pixel p, if this is the first pixel p_0 in the scan order (*i.e.*, the top left pixel), we know that its label value is 1, and the label value of its parent (*i.e.*, itself) is also 1 (lines 9 to 11), because p_0 is in the root node thanks to the added external boundary described in the previous section. For all other unlabeled pixels $p \neq p_0$, we follow the contour of the unlabeled region, which is a hole in the label image, thanks to the *border* image. The FOLLOW_CONTOUR routine computes on the fly a contour-based measure $\mathcal{P}$ characterizing the hole, such as

the average of gradient's magnitude along the contour and the bounding box of the hole. If this contour-based measure $\mathcal{P}$ does not satisfy some criterion (for instance if it is not contrasted enough), or if it does not satisfy some geometrical criterion (for instance if the hole is too small), we do not create a new label value for this region; this acts as if the region were discarded (in Fig. 2.8, two regions are discarded between the 2nd and the 4th columns). Let p_{-1} be the pixel just before p in the raster scan order (p_{-1} is guaranteed to be labeled), we assign the label value of p_{-1} to the underlying hole region, which means we merge it with its parent region. If the contour measure $\mathcal{P}$ satisfies the corresponding criterion, we create a new label value to label the hole region, and update the parenthood relationship of this new label value to $\mathfrak{L}(p_{-1})$ (see lines 14 to 19). Then we proceed the routine BLOB_LABELING to label the connected set of pixels having the same Laplacian sign as p or being null and update the auxiliary $border$ image. Note that since we use the 4-connectivity neighborhood ($\mathbb{N}_4$) in the routine BLOB_LABELING to label regions and mark neighboring pixels having different sign of $\Delta_\square$, we need to use the 8-connectivity neighborhood ($\mathbb{N}_8$) to follow completely the active contour of an unlabeled region (see also the longest contour in the left image in Fig. 2.8(e)).

An example of the proposed algorithm on the interpolated image in Fig. 2.8(b) is depicted in Figs. 2.8(c-e). Different stages of the algorithm are depicted in Fig. 2.8(e). Note that the pixels having null Laplacian sign are grouped with the parent region. The two small regions inside cyan and respectively yellow region are also discarded, they are grouped with the parent region. The resulting "label image + tree" are depicted in Figs. 2.8(c) and 2.8(d).

2.3.3 Optimization of ToSL Construction

The well-composed interpolation described in Section 2.3.1 resolves all the topological issues at critical configurations with the cost of quadrupling the number of pixels. Yet, in practice, we do not need to apply this interpolation, which can be emulated efficiently without subdividing the image domain thanks to the particular non-local way of interpolation described in Section 2.3.1.

The optimized version of the proposed algorithm by emulating the interpolation is depicted in Fig. 2 by the black and red parts. More precisely, the used particular non-local interpolation is based on the inclusion relationship of components. The interpolated values at critical configurations are given by the values of the outer region. Besides, the proposed algorithm for such interpolated image labels regions from outside to inside. Consequently, it is equivalent to use 8-connectivity ($\mathbb{N}_8$) when we use blob labeling algorithm to label pixels that are not yet activated thanks to the $border$ image (see line 26 and line 27 of BLOB_LABELING in Fig. 2). This makes the other two pixels of critical configurations having a different sign of Laplacian disjoint and marked as active contours. Consequently, when we proceed the FOLLOW_CONTOUR routine in the following, we need to use 4-connectivity ($\mathbb{N}_4$) (see line 38 of FOLLOW_CONTOUR in Fig. 2) to avoid connecting two disjoint unla-

beled regions.

An example is given in Fig. 2.8(f) which shows different stages of the proposed optimized algorithm without using interpolation. Note that the green pixels (resp. red pixels) of the critical configuration in the top (resp. bottom) red rectangle in Fig. 2.8(a) are considered as connected using 8-connectivity ($\mathbb{N}_8$) when we proceed the routine BLOB_LABELING. The active contours (black pixels) in Fig. 2.8(f) are processed using 4-connectivity ($\mathbb{N}_4$). The two runs depicted in Figs. 2.8(e) and 2.8(f) result in the same "label image + tree" depicted in Figs. 2.8(c) and 2.8(d).

2.4 Text Extraction With Tree of Shapes of Laplacian Sign

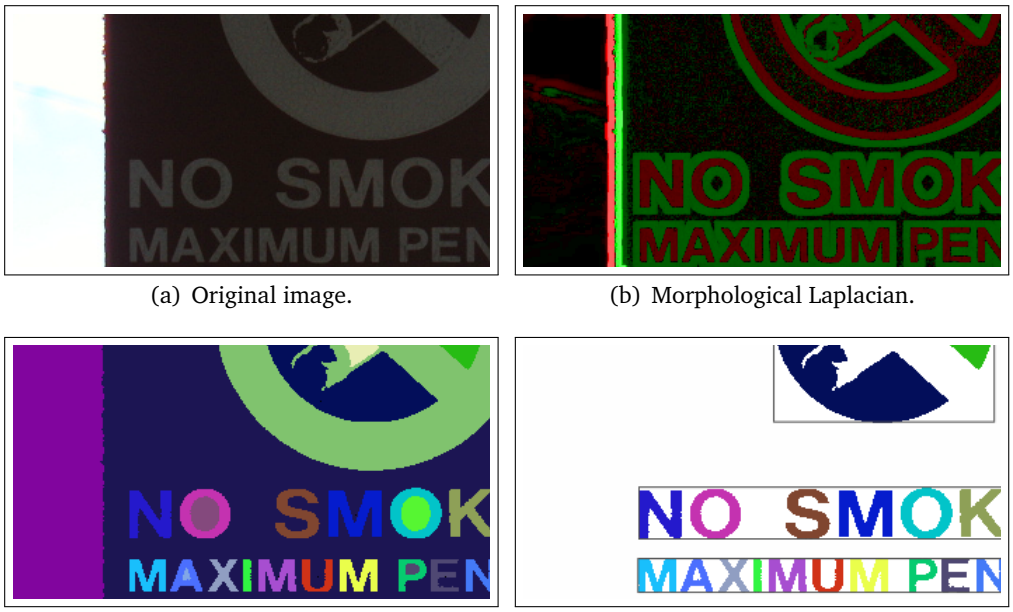
In this section, we focus on how the Tree of Shapes of Laplacian sign can be used to segment text in images.

2.4.1 Method overview

The method that we propose to segment text in natural images is very simple; put very shortly, text components are selected among nodes on the ToSoL of the input image. The advantages of ToSoL are threefold: first, regions contours are localized precisely thanks to the morphological Laplacian; second, its computation is efficient; finally, the inclusion encodes by the ToSoL parallel the object-background relationship, which carries useful contextual information.

After the ToSoL is computed, we group similar components together to form text boxes. For that, we only consider the bottom of this tree (the leaves and sometimes their parent): for each component, we search spatially in the label image what are their left and right components to be grouped into a text box. In this step, we highly take advantage of the tree structure: it allows very easily to discard many regions as non-text and to determine if a leaf region is a character hole or a plain character.

The leaves (and sometimes their parents) of the resulting tree are then grouped together to form text line candidates. We only consider roughly horizontal words, containing at least two characters. We assume that related characters belong to the same background, this implies that they have the same parent in the tree structure. As a consequence, the grouping process can be performed efficiently: the only candidate regions for characters to be grouped into text lines are *siblings in the tree structure*. Starting from each tree leaf, we thus use a classical search in the image space to group siblings (some additional geometric information such as region height and maximal inter-distance between regions are also used to control the grouping process). Note that we know when a leaf is a character hole because both left and right neighbor regions are its grand-parent in the inclusion tree (the background region being the parent); we then consider its parent node (its background being the character). An illustration is given in Figure 2.9.



(a) Original image. (b) Morphological Laplacian.

(c) Result of labeling. (d) Node selection and text boxes.

Figure 2.9 – Illustration of the proposed method: mathematical morphology tools are contrast-invariant so we successfully deal with low-contrasted data (note that the Laplacian image (b) has been lightened to be readable).

The key features of this method are the following: **1.** It runs very fast since the processing chain is very simple and since all operations have a linear time complexity (see Sec. 2.4.4); **2.** The proposed method, based on the morphological Laplace operator, outperforms more "classical" component-based methods that select candidate regions for characters (see Sec. 2.5); **3.** The fact that regions form an inclusion tree, thanks to the *well-composedness* property, allows for powerful decision taking when grouping regions into text boxes.

2.4.2 Construction and simpification of ToSoL

The ToSoL is computed as described in earlier sections. The Morphological Laplacian and gradient are computed using an 11x11 square as the structuring element. Although the structuring element size does not alter contours, a larger one will increase resistance to noises and compression artifacts as shown in Section 2.2.1. For this particular application, some geometric criteria are used to simplify the ToSoL on the fly (Algorithm 2 lines 13 and 26) are average gradient magnitude on the contour and some geometric properties of that region. The average gradient magnitude threshold is fixed at 30 for a scalar luminance map in range 0-255. That value is roughly 10% of maximum luminosity different, which, according to [3], allow human subjects to obtain more than 90% recognition performance. The height and width threshold is chosen at 5px so that remain components is big enough to distingue both 'E' and 'M'. A criterion on the height-width ratio is set to eliminate thin and long components. 0-crossings that do not satisfy these

Property	Criteria
Average Gradient Magnitude	$G_{avg} > 30$
Height and Width	$h > 5px$ and $w > 5px$
Height over Width	$0.1 < \frac{h}{w} < 10$
Area over Bouding Box area *	$\frac{area}{h*w} > 0.1$

Table 2.1 – 0-Crossing filtering criteria.

Property	Criteria
Parent	$par(c_i) = par(c_j)$
Distance	$d(c_i, c_j) < 2min(h_{c_i}, h_{c_j})$
Height similarity	$SH(c_i, c_j) = \frac{min(h_{c_i}, h_{c_j})}{max(h_{c_i}, h_{c_j})} < 0.5$

Table 2.2 – Grouping criteria.

criteria will be discarded. These criteria are summarize in Table 2.1. All criteria except the last one use properties that are obtained by the subprocess contour following. The last criterion is defined with the region's area, that is obtained by the labeling process and thus does not require a second pass through the whole image. Our tree construction and labeling are therefore effective.

2.4.3 Component grouping by spatial search

In this step, a classic spatial search is performed to group remained segments. Only roughly horizontal text lines are considered but the searching direction could be modified to include vertical text lines. We consider only nodes at the bottom of ToSoL (leaves and parents of leaves) as the candidate. Each component will be linked with at most two neighbors, which are the firsts on each side that satisfy grouping criteria. The neighborhood finding accomplishes by searching pointers going on the horizontal line through the center of that component. These pointers will go pixel by pixel of the label map on both directions. Text candidates of our method are groups that have more than two components. With c_i, c_j and h respectively denote two components and the height property, a link between components is created if these criteria are met criteria in Table 2.2.

The first criterion takes advantage of the inclusion structure to limit the search space effectively. The classic spatial search would stop after stepping out of the parent's region. The second criterion further limits the search space because semantic characters normally close together. The third criterion makes sure the height of characters in the same group does not vary more than two times. More criteria could be added to improve precision. However, we should note that this method focuses only on proposing candidate but not remove all the false positive. Nevertheless, a large variety of feature use by common false positive elimination methods are collected during the ToSoL construction (contour length, bounding box size, color, area, number of hole. . .)

Method	Recall	Precision	F-score	Consistency
SWT [21]	0.464192	0.8861	0.609232	0.505042
ER [67]	0.613059	0.892023	0.629221	0.726689
TMMS [22]	**0.784568**	0.7522	**0.768043**	0.791303
Our	0.636168	**0.933058**	**0.756528**	**0.849754**

Table 2.3 – Text segmentation comparison.

2.4.4 Complexity analysis

The morphological gradient and Laplacian rely on dilation and erosion using a square structuring element, which can be efficiently implemented thanks to a 2-pass (horizontal then vertical) incremental (heap-based) process. In addition, this local process is easily parallelizable, and eventually, it has a linear time complexity w.r.t. the number of pixels. The blob labeling process (see Algo. 2) has also a linear time complexity: every pixel are only visited once with the main 'for' loop and the queue-based propagation, and browsing the 0-crossings contours is also limited by the number of pixels. Last the grouping process, dealing with very few nodes of the tree and browsing a few pixels of the label image, is trivially linear.

2.5 Experimental Results

2.5.1 Quantitative results on text segmentation

We have evaluated the proposed method of text segmentation in the context of task 2 of Challenge 2 in ICDAR 2015 "Robust Reading" competition. The dataset contains 233 natural images with focused scene texts. The ground truth of text segmentation results is available. Some qualitative results are given by Figure 2.10; they include reverse-video, uneven illumination, fancy fonts, blur, and different text sizes.

We have compared our method with three popular methods for generating text candidate regions: Stroke Width Transform (SWT) [21][3], text detection based on Extremal Regions (ER) [55, 67] (implemented in OpenCV), and Toggle Mapping Morphological Segmentation (TMMS) [22]. The first two methods are widely used as the first step of many state-of-the-art pipelines. For a fair comparison, we compare the performance of text candidate region generation of the four methods, that is *text segmentation*, and we discard the rest of the pipeline (mainly false positive removal). For that, we only consider generated regions that touch the ground truth (GT) texts. We use the evaluation scheme proposed by [13, 14], based on the recall and precision scores in terms of pixels; we also compute a consistency value measuring how much ground-truth text components are split into several pieces. The results are given by Table 2.3; our method achieves a competitive recall with high precision. Figure 2.11 depicts in detail how each method behaves w.r.t. all the ground-truth texts in the dataset: the plots illustrate the distribution of the segmented

3. The implementation of SWT is provided by `https://sites.google.com/site/roboticssaurav/strokewidthnokia`.

Figure 2.10 – Qualitative results using "ICDAR RRC: Focused Scene Text" database: input (left), labeling (middle), final boxes (right).

text components at different coverage level (left) and accuracy level (right). The coverage (resp. accuracy) represents the percentage of the matched surface between the GT and a detection object with respect to the GT (resp. detection) surface; see [12] for details. One can see that our method covers the ground-truth texts at relatively high coverage levels (mostly distributed between 50% to 100%), which is not the case of the other methods.

2.5.2 Applying the method to document binarization

The method proposed in this paper has been applied to binarize documents in the challenge #2 (Smartphone OCR) of "ICDAR 2015 Competition on Smartphone Document

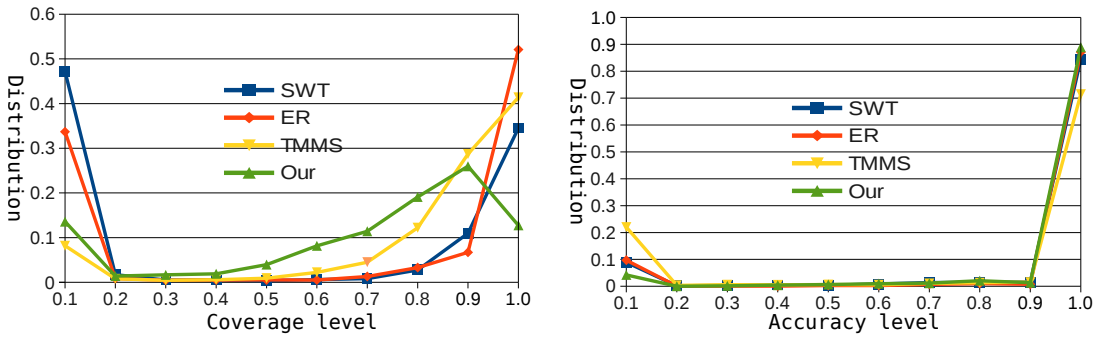


Figure 2.11 – Evaluation based on coverage and accuracy [12].

(a) Original image. (b) Self-dual binarization.

Figure 2.12 – Text segmentation with our method can be seen as a binarization technique, providing also reverse-video text.

Capture and OCR (SmartDoc)" [10]. We ranked **2nd** in this competition among 8 contestants, with a character accuracy of 95.85% (note that the winner has taken advantage of the redundancy of documents in the test set to correct each individual document). Relying on the Laplace operator has turned out to be robust for the binarization of both blurred text and low-contrasted text. In addition, our method is self-dual since it naturally handles the same way the case of dark objects over bright backgrounds and the opposite case, as depicted in Figure 2.12.

2.6 Conclusion

In this chapter, we have presented a hierarchical representation of the image contents based on the inclusion of the 0-crossings of the morphological Laplace operator. Although we use the very "classical" idea of relying on the 0-crossings of the Laplace operator image to obtain the objects of interest, our version is innovative for several reasons. First, we rely on the morphological Laplace operator, which performs well in the case of uneven illumination, which is a common defect in natural images. Second, we ensure that the "0-crossings" are really 1D objects. For that, we use a well-composed Laplacian image, we compute the tree of shapes of its sign, and we consider the 1D topological boundary of shapes. As a consequence, the positive and negative regions can be organized into a tree without topological ambiguity. Last we present a linear time complexity algorithm, which is a hardly more sophisticated blob labeling algorithm, to compute the hierarchical structure. We also provide a way, directly during the computation process, to ignore some regions if they are not relevant, and an optimization that mimics well-composedness and then avoids to duplicate pixels.

We have explained how to rely on this representation to segment text lines in natural images, and we have shown that it competes with classical methods of text candidate extraction. Thanks to the linear time complexity of the ToSoL algorithm, our methods are also efficient (from our first experiments with an ordinary computer, it runs in about 0.2s

on a 1M Pixel image). We also have applied a similar scheme to document image binarization, which has been used in the ICDAR 2015 SmartDoc competition. As a perspective, we intend to integrate our text segmentation approach in a text detection pipeline, thus including false positives detection, to get an end-to-end evaluation.

Chapter 3

Spatial Alignment Graph With Respect to Inclusion

3.1 Introduction

Classical image representations, such as those review in Section 1.3, are based on the notion of connected component (CC) with the assumption that each object of interest could be represented by a CC. However, due to various reason (noise, illumination condition, occlusion, nature of that object), we could found that an object is represented by several CCs. On the other hand, some application require the analysis of a group of objects of the same class and sometimes these objects are not connected. One example is the field of text detection in image. Because text characters are usually appears in group (e.g., words, sentences, paragraph), processing groups of text-like CCs will not only reduce false possitive but also yield a better understanding of the scene (e.g., individual characters “s”,“t”,“o”,“p”, versus a word “stop”). In this chapter, we will explore the problem of non-connected components grouping using a generalized shape-space morphology on a graph constructed with both inclusion and adjacent information.

The most straightforward approach on non-connected components grouping is making a complete graph out of these regions. This graph is then segmented to obtain groups of similar objects that satisfy some criteria. This approach is computation expensive and therefore it is only reasonable if we working with a low number of CCs. However, the number of CCs could be large, especially in case of hierarchical image representations. A reduction graph order (number of vertices) or graph size (number of edges) is desirable if we want to keep the computation cost reasonable.

The graph order could be reduced by binarizing the image, i.e.,by marking some relevant CCs as objects, and the others as background. We will then build a graph to analyse the relationship between objects and ignored all the backgrounds. The graph size would be reduced by changing the way the edges set are created. We could use more a priori assumption about objects of interest, especially the spatial relationships which usually carry contextual information. Usually, the adjacency, aligment relationships between objects, or the distance between them are used in this case while another important information, the background/object relationship, is discarded. The reason is that all unrelevant CCs are treated as they belong to the same background. For example, most text detection methods reviewed in Section 1.5 could be divided into two-step: an image binarization into text

and background; and then a regrouping of horizontal regions which may or may not be guided by a refining classifier.

We argue in Section 2.2.2 that treating all non-objects regions as the same background regions would cause the loss of contextual information. Firstly, related objects are on the same background. Secondly, sometimes it is difficult to make a clear binarization of the image because the object/background relationship is relative, which means that an object could be a background of other objects (of the same class or different class). In order to consider this powerful contextual information, we are interested in the inclusion relationship. We find that the background-object relationship could be deduced from the inclusion of regions: It is reasonable to treat the region that contains the object as its background. This information along with other spatial relationship will give us a better understanding of the scene.

In Chapter 2, we tackle this idea. Instead of making a clear object-background separation, we try a more dynamic one: all nodes on the ToS could be treated as objects and its ancestors are the background. On a highly simplified ToS as the ToSoL, we could expect that only siblings nodes are semantically related. In other words, we assume the distance on the tree between semantically related nodes equal two. The regrouping process only groups these sibling nodes which results on a disconnected graph where each connected one is a grouped of relevant objects. While providing a fast way to group text characters, that method is based on an assumptions that only hold on a simplified inclusion tree such as the ToSoL. However, in cases that the objects of interest appear on both the fin and coarse parts of the image, we may want to work with a less simplified representation so that we could preserve more detail. In that case, our workflow needs to be extended. This is the main focus of this chapter.

Our extension results in a graph of shapes that is build from both inclusion and spatial relationship between image regions. By applying the connected operators frameworks on that graph, we could analyse both these information at once. This approach requires the construction of two trees and one graph. The first tree is a hierarchical representation of the input image. Then we create a graph that represents the relation between nodes on that tree, which we will call collectively “shapes”. Finally, we use a second tree, which is a hierarchical representation of that graph, to analyze that graph. This approach is summarized in Figure 3.1. Our approach fits into the connected operators frameworks: it does not create, nor shift flat zones’ boundaries.

The structure of this chapter is as follows: First, we will recall the connected operators framework and an earlier approach that rely on a graph of shapes, the shape-space morphology, in Section 3.2. In the same section, we will explain our central concept: a generalized shape-space morphology. Second, in Section 3.3, we will discuss the necessity of using a less simplified tree, and how we handle the background/object relationship, which is deduced from the inclusion of regions. We also consider different strategies to obtain the spatial relationship between nodes on a tree. In the end, we chose a spatial search approach to make the final graph of shapes which we call the spatial alignment

graph w.r.t inclusion. Thirdly, in Section 3.4, we will present how the spatial alignment graph w.r.t inclusion could find application in text detection and segmentation. Finally, we will conclude the chapter and give some perspectives.

3.2 Expansion of Shape-spaces Morphology

In this section, we will present the concept of a shape-space and the application of morphological tools on that space. Compare to the shape-based morphology of Xu et al. [115], we still use connected filters on the shape-space but with two main differences. The first difference is that the edges set of our shape-spaces does not necessarily coincide with the inclusion relationship between image regions. Such shape-spaces are desired if we want to analyze the relationship between nodes that do not have a parent-children relationship on a tree representing the image. The second difference is the way we apply connected operator on the shape-spaces. We have to take a different approach that is more consistent with the way connected operators act on image space. First, we briefly recall the classical tree-based connected operator and tree-based shape-spaces morphology of Xu et al [115]. Then we will then present how connected operator could be applied on more general shape-spaces.

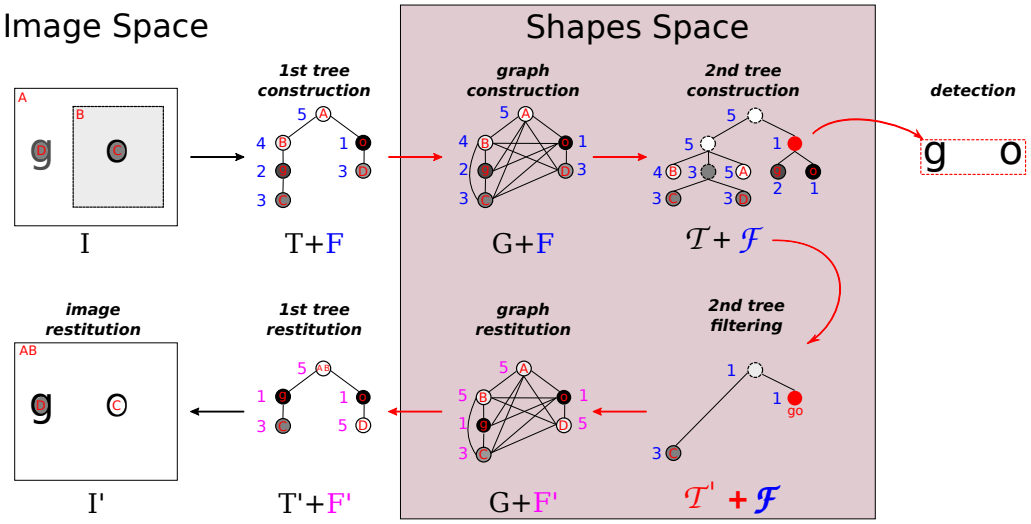


Figure 3.1 – Shape-space filtering overview. Blue and purple number: nodes' associated value. Red characters: labels of region. After obtaining the second tree $\mathcal{T}$, we could retrieve groups of objects of interest among its nodes, or we could filter that tree and reconstruct a filtered image. Compare to Xu et al.[115], there are two main differences: firstly, the graph represents our shape-space does not necessarily have the same topology as the first tree; secondly, the tree reconstruction process works by the restitution of the associated value of nodes (on the shape-space and the first tree), similar to how we restitute the image function from the first tree.

3.2.1 Classical tree-based connected operators implementation

We have reviewed in Section 1.4 the classical tree-based implementation of connected operators. It consists of three steps that can be summarized as follows:

- Tree construction: From an image $I : D_I \subseteq \mathbb{Z}^2 \rightarrow V$ with V the value space and D_I the domain of I, let T a set of disjoint or nested regions $T = \{R \subseteq D_I\}, R_i, R_j \in T \Rightarrow R_i \cap R_j \in \{\emptyset, R_i, R_j\}$. The set of shapes T could be the set of connected components obtained by thresholding the image at different levels in case of trees based on threshold decomposition (Section 1.3), or by merging adjacent regions of an initialized partition of the image in case of the hierarchies of segmentation (Section 1.2). The poset $(T, \subseteq)$ is tree-based image representation. It may couple with an associated function $F : T \rightarrow V$ that give the value of each region. F could be the mean value of each region, or the threshold that we obtain it.

- Tree filtering: In this step, we remove some regions from the set T. Each element of T is evaluated by an attribution function $\mathcal{A}$ which could be increasing or non-increasing. Based on $\mathcal{A}$, we remove from T some sub-trees (pruning strategy) or only nodes that do not pass the criteria (non-pruning strategies). This process results in a filtered tree $T' \subset T$. The associated function F' of the filtered tree could be just the restriction of F to T' or a modified function (e.g., with subtractive rule).

- Image reconstruction. A filtered image I' is reconstructed from $(T', \subseteq)$ and F'. Each pixel gets the value of the smallest regions (when ordered by inclusion) that contain that pixel:

$$I'(p \in D_I) = F'\left(Min\{R \in T' : p \in R\}\right) \tag{3.1}$$

The most important process of the tree-based connected operator is the tree filtering step. If the attribute function $\mathcal{A}$ is increasing, the filtering is simply done by removing all nodes that do not pass the threshold. When the attribute function $\mathcal{A}$ is non-increasing, which is usually happens, there are different approaches. The pruning strategies take into account the attributes $\mathcal{A}$ of more than one nodes. However, because they remove or preserve a whole subtree, these strategies may not be suitable in case several relevant objects are on the same branch of the tree. On the other hand, the attribute thresholding strategies do not have that drawback. Being non-pruning strategies, they can remove any nodes that do not pass the threshold. However, they are based only on local information of a single node without considering their relationship with others.

3.2.2 Tree-based shape-spaces morphology

In [115], Xu et al. come up with a non-pruning approach that can take into account the attributes of multiple nodes. They notice that the tree-based image representation

could be regarded as a node-weighted graph, and therefore classical tools that treat an image as a graph could also be applied. Such graph is called a “tree-based shape-space”. The shape-spaces consist of a set of “shapes”, which are nodes of the tree-based images representation, and a neighborhood, which is defined by the parent-children relationship between these nodes. This type of neighborhood comes naturally and allows the inclusion of nodes on the tree to be considered at the same time as the attribute function. This approach was reviewed with more detail in Section 1.4.2. In short, the method applies the tree-based connected filters on the tree-based shape-space. First, a Min/Max-tree or the ToS of the tree-based shape-space will be constructed. That second tree will be pruned based on another attribute, which is designed to be increasing. Finally, the set of pruned nodes will be then removed from the original tree, which is an equivalent representation of the filtered image. This method allows us to remove nodes on the first tree that are local extrema and the neighboring nodes. An example of this method is given in Figure 3.3.

By defining the neighborhood of the shape-space coincided with the parent-children relationship of nodes on the first tree, the authors integrate that information into the shape-space. This is an interesting idea because later analyses of that graph will be affected by that information. However, Xu et al. only use the internal relationship which creates the tree-base image representation in the first place. We are interested in expanding this idea to analyze other relationship between nodes. In particular, we want to take into account not only the inclusion of shapes but also their spatial arrangement on the image space. As explained in Chapter 2, we found both these pieces of information would be interesting in case objects of interest is not represented by a single node but by multiple nodes on different branches. Another concern about the original approach of Xu et al. is that the reconstruction of the first tree from the second tree is not consistent with how the image is reconstructed from the first tree. As a results, that process only allows pruning approach on the second tree, which may not always be our intention.

Before getting into a specific shape-space, in the next section, we try to address these concerns, 1) how to expand Xu’s framework to a shape-space defined with a neighborhood different from parenthood relationship of the first tree; 2) and how to allow non-pruning strategies on the second tree.

3.2.3 The shape-spaces and the tree-based connected operator on the shape-spaces

In this section, we formulate the notion of shape-spaces, which is a generalized version of tree-based shape-spaces. This space consists of a set of points, which is the set of regions of the first tree, and a neighborhood, which is defined to encode the information that we would like to analyze. We will also discuss what we could obtain from a shape-space and how we fit it into the connected operator framework.

3.2.3.1 Shape-spaces

Like any space, a shape-space must consist of a set of points and a set of edges. We define the shape-space as any graph created from the set of regions (or "shapes") of hierarchical image representations and any set of edges. By this definition, the tree-based shape-space is a particular case of the shape-space, when the parent-children relationship defines the neighborhood.

We could obtain a shape-space by creating a complete graph from the set of shapes and then filter out edges that do not pass specific criteria. We could obtain it differently by doing a constrained neighborhood search. The latter is preferable when computational cost is a concern since the former is $\mathcal{O}(n^2)$ with respect to the number of nodes on the first tree. We will explore that approach in later sections to obtain a shape-space that have the neighborhood defined with both types of spatial relationships of image region: inclusion and spatial alignment.

3.2.3.2 First tree construction and shape-space formation

The first step is to construct a tree-based image representation $(T, \subseteq)$ of the input image I. The tree and its associated function $F : T \to V$ form an equivalent representation of the image. The choice of trees depends on the targeted application so that we could find nodes on the tree represent the objects of interest. During or after the construction of the tree, the features characterizing each node are computed. These attributes could be classified as increasing or non-increasing, some of them are reviewed in Section 1.4.

From T, we create a shape-space by defining a neighborhood which will form a graph $G(T, E)$ with T the set of shapes and E the set of edges. E should encode the relationship between nodes that we would like to analyze. This graph could be node-weighted (e.g., by its attributes) or edges-weighted (e.g., by a dissimilarity measure between nodes).

Let us denote the image region represented by a node $n \in T$ by $R(n)$.

3.2.3.3 Second tree construction

With the same motivation that leads us to a hierarchical representation of the image in the first place, we create a second tree $\mathcal{T}$ to represent the shape-space $G(T, E)$ hierarchically. The second tree comprises of a set of regions $\mathcal{T} = \{\mathcal{R} \subseteq T\} \cup \{T\}$, whose elements satisfy the disjoint or nested properties, along with an association function $\mathcal{F} : \mathcal{T} \to V$. It is an equivalent representation of T, which means that we could restitute T from $\mathcal{T}$ and $\mathcal{F}$. Each node of the second tree represents a group of nodes on the first tree.

The type of second tree also depends on the application, and the nature of the graph representing the shape-space. In case of objects detection, we may want to have regions with desire properties represent by some nodes on the tree. As a result, we may want to chose a tree that can be guided to form such shapes. For example, if we want to obtain a group of regions with similar color, we could use a BPT that prioritize the merging nodes

that are closer in term of color. In another case, if a node on the shape-space could be weighted with an attribute function $\mathcal{A}$ that reflects how likely a shape to be of the desired type. The minima and its neighbors on that shape-space would be nodes that less likely to be the desired objects. We could perform non-desire shapes filtering by building and pruning leaves on a Min-tree represented the shape-space.

Nodes on the second tree should be characterized by a second attribute to guide the filtering of the second tree. The second attribute could be designed as an increasing one if we want to perform a simple tree pruning. It could also be a non-increase attribute to select particular nodes on the hierarchy with pruning or non-pruning strategies. We should emphasize that non-pruning strategies are not supported in the original tree-based shape-spaces approach but, they are with ours. In case the second attribute is deduced from the characteristic of nodes on the first tree, it could be computed effectively during the construction of the second tree.

The image region represented by a node $\mathcal{N} \in \mathcal{T}$ is $R(\mathcal{N}) = \bigcup\{R(n)|n \in \mathcal{N}\}$. Let us denote the set of all image regions represented by nodes on the second tree $\Re(\mathcal{T}) = \{R(\mathcal{N})|\mathcal{N} \in \mathcal{T}\}$.

3.2.3.4 Second tree filtering and image reconstruction

According to the application, the filtering of second tree $\mathcal{T}$ is performed by removing some nodes on the second level tree based on the second attribute. For some applications, e.g., object detection, objects or groups of objects of interest could be retrieved from remaining nodes on the second tree (the detection step in Figure 3.1). For other applications such as image simplification, we could reconstruct the filtered image from the simplified second tree.

Let us look into our proposed image-restitution process, which is different from the original approach and allows non-pruning strategies on the second tree. The set $\Re(\mathcal{T}')$ does not guarantee to have the disjoint or nested property. Therefore, the subset of $\Re(\mathcal{T}')$ that contains a certain pixel may not have a minimum (when ordered by set inclusion). An example is shown in Figure 3.2. On $\mathcal{T}'$ the image regions represented by the rectangle node (which is the union of the hole of 'g' and the small rectangle surround 'r') and the triangle one (which is the union of the shapes of 'g' and 'r') have an intersection but are not nested. For that reason, we can not reconstruct the image directly from $\Re$ by Equation 3.1 as we reconstruct the image from T. Instead, we have to reconstruct the first tree first.

In the original approach, the filtered tree T' is achieved by subtracting from the original tree T the set of nodes that are contained in the filtered nodes of the second tree, i.e., $T' = T - \{R|R \in \mathcal{R}|\forall\mathcal{R} \in \mathcal{T} - \mathcal{T}'\}$. The filtered tree T' has the disjoint or nested property because $T' \subseteq T$. In this approach, the associated function of the tree will remain the same, i.e., $F' = F|_{T'}$, and the filtered image is obtained by $I'(p \in D_I) = F'(Min\{R \in T'|p \in R\})$. We should remark that the root node of the original tree T must not be removed if we want I' to cover the original image domain. In this approach, despite

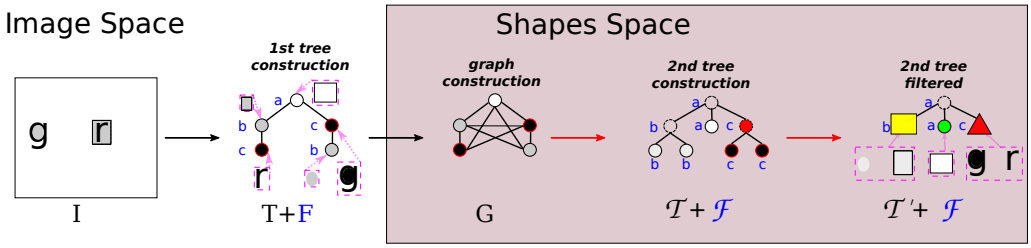


Figure 3.2 – Example of second tree filtering. In this example, T is a ToS. The associated value is shown in blue next to each node. Dashed windows show the image regions associated with each node on T and $\mathcal{T}$. We see that the regions represented by nodes on the first tree satisfy the disjoint or nested property, but those represented by nodes on $\mathcal{T}'$ (and $\mathcal{T}$) do not, e.g., the yellow rectangle and red triangle nodes on $\mathcal{T}'$.

the filtering strategy on the second tree, it will always act like a tree pruning. The reason is that a node on $\mathcal{T}$ is a superset of any of its descendants. Therefore, the act of removing a node gives the same T' as removing the subtree rooted at that node. This limitation could be seen as a disadvantage if we may want to perform a non-pruning strategy on the second tree, e.g., when two interesting nodes appear in the same branch.

We propose using another restitution approach. Because $\mathcal{T}$ is a complete representation of the shape-space, we can restore that space similar to the way we restitute the image space and image value from the first tree T. With the filtered second tree $\mathcal{T}'$, and its associated function, which is the restriction of $\mathcal{F}$ to $\mathcal{T}'$, i.e., $\mathcal{F}' = \mathcal{F}|_{\mathcal{T}'}$, the restored shape-spaces could be obtained by:

- **The set of point:** $T' = \{R \in \mathcal{R} | \mathcal{R} \in \mathcal{T}\}$
- **The set of edges:** $E' = \{e_{x,y} \in E | x, y \in T'\}$
- **The associate function:** $F'(R) = \mathcal{F}\left(Min\{\mathcal{R} | \mathcal{R} \in \mathcal{T}' \wedge R \in \mathcal{R}\}\right)$.

In other words, the restituted shape-space is the induced subgraph of $G(T, E)$ from the set of first tree's nodes remains in the $\mathcal{T}'$. The associated value of each node is the value of the smallest node of $\mathcal{T}$ (by inclusion order on $\mathcal{T}$) that contains that node. If the root of $\mathcal{T}$ remains, the restituted shape-space is identical to the original one. However, changes in the filtered second tree are still reflected by the associated function F' : each node has a new value, which is the value of the smallest node on $\mathcal{T}'$ containing it.

In application, we are more interested in the filtered tree T'. The cover graph of $(T', \subseteq)$ is usually a tree (if the root node is removed, it would be a forest). There may exist nodes on T' whose children have the same associated value, which is redundant and make no impact on the reconstructed image. We could make T' more compact by removing all nodes that have the same associate value with their parent. The final filtered tree is therefore $T' = \{R | R \in \mathcal{R} \wedge \mathcal{R} \in \mathcal{T} \wedge F'(R) \neq F(par(R))\}$. From $(T', \subseteq)$ and F', the restitution of filtered image I' is trivial.

This approach is still a connected operator because it only removes some region boundaries (a region n is removed in image space if $F'(n) = F'(par(n))$) or simplifies the value space (when two nodes were merged on $\mathcal{T}$, but they are not connected to the image space) but does not create new one nor shift existing boundaries.

We present a comparison of two approaches when the G is a tree-based shape-space and the second tree is filtered with a pruning strategy in Figure 3.3 or non-pruning one in Figure 3.4. In the former example, we see that our approach yields a different result from Xu's. However, it still is a connected filter. In the latter example, our approach is able to keep two nodes in the same branch (node {C,E,H} and the root of the tree) while Xu's removes a whole branch. Xu's approach could be modified to remove only "proper" element (those that does not belong to any of its descendants). However, this only works if the second tree is a hierarchy based on the threshold decompositions.

3.3 Spatial Alignment Graph With Respect to Inclusion

In this section, we present one shape-space that is created from two types of information: inclusion and spatial arrangement of image regions. This shape-space could be analyzed by shape-spaces framework in an earlier section. We are motivated in using these types of information because they are useful for non-connected components grouping, e.g., text characters grouping, especially after the results presented in Chapter 2. As present earlier, the inclusion relationship, from which the background/object relationship could be deduced, and the spatial relationship give us a better scene understanding.

This section will present general ideas on the shape-spaces. First, we will discuss how we handle the background/object relationship on the inclusion tree. Then, we will explain how we encode the spatial arrangement of image regions into the shape-spaces. Its application will be presented in the following section.

3.3.1 Handling the objects-background relationship by the inclusion in the ToS

In Chapter 2, we deduce the background/object relationship from the inclusion of 0-crossing since the boundary of the background should contain the boundary of objects on it. This relationship is encoded in the ToSoL. In the same manner, the inclusion of image level lines encoded in the ToS would serve the same purpose. For example, thanks to the ToS in Figure 3.5(b), one could say that the flat zone B is included in A because it is A's descendant in the inclusion tree. In addition, the ToS have the advantage of being self-dual, i.e., we could build it without any assumption about the contrast of objects. The ToS could be computed effectively for scalar image [29]. A sound extension of ToS for multivariate images (MToS) [16] has also been reviewed in Section 1.3.3. For these reasons, we will use the MToS to deduce the background/object relationship.

However, we should note that the ToS may face performance reduction in uneven

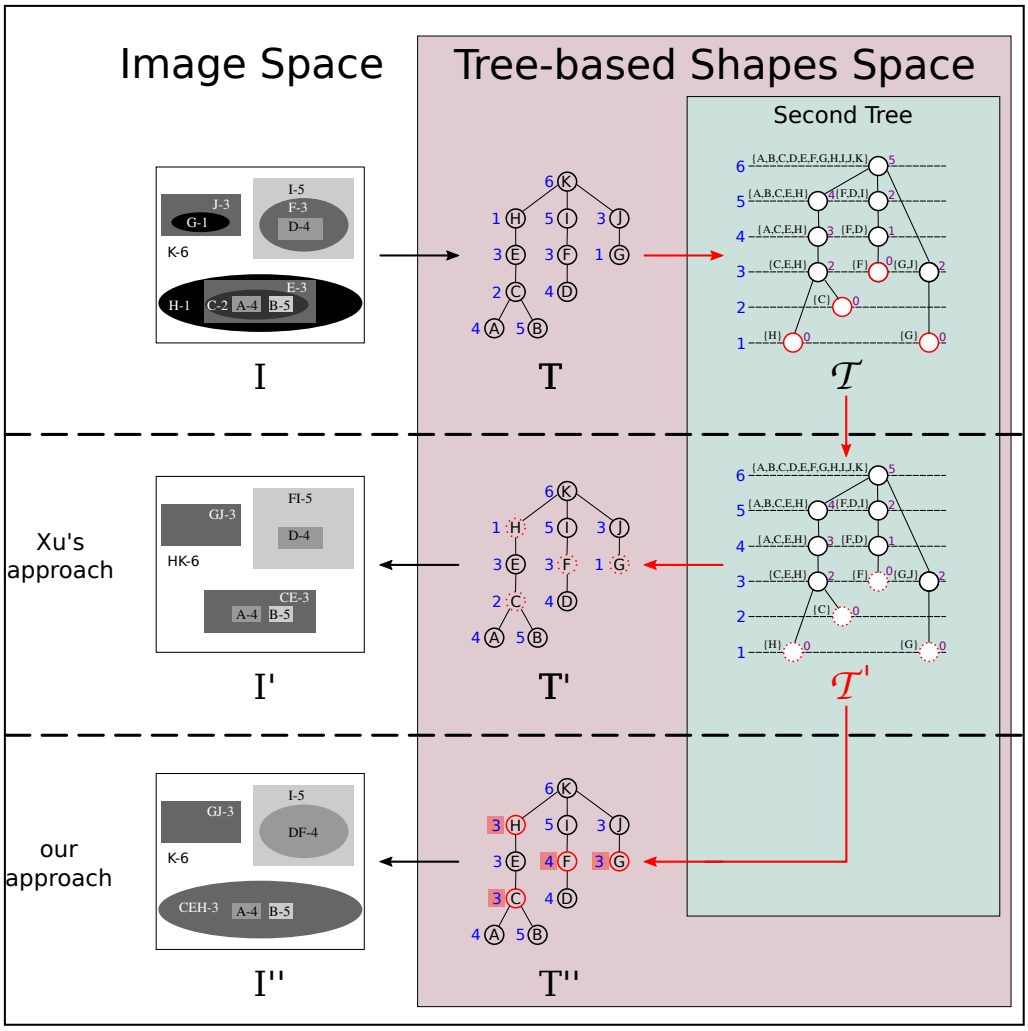


Figure 3.3 – Our approach to shape-spaces filtering compares with Xu's in case the second tree is filtered with a pruning strategy. Blue number: associated value F which is also the attribute function $\mathcal{A}$ in this example; Purple number: second attribute (which is the height of $\mathcal{A}$). Flat zones of top-left images are denoted by a letter along with a number showing its gray value. In this example, we prune the leaves of the second tree. The shape-space is a tree-based shape-space, i.e., it has the same topology as the first tree, which is the ToS of the input image. The second tree is a Min-Tree. Xu's approach is shown in the second row and our approach, the reconstruction through shape function, is shown in the third row. Red contours and red highlights indicate notable changes (removed or to be removed nodes, changes in value).

lighting condition. For example, in the ToS whose labeling map shown in Figure 3.6(b), we could not find any node or group of nodes that could represent some characters in the input image (e.g. "s" in "short"). The best representation is a set of unconnected regions, represented by nodes on different branches of the ToS, whose lowest common ancestor represents a large region of the background. However, since our objective comprises the regrouping of unconnected components, we expect that these "broken" objects would be retrieved.

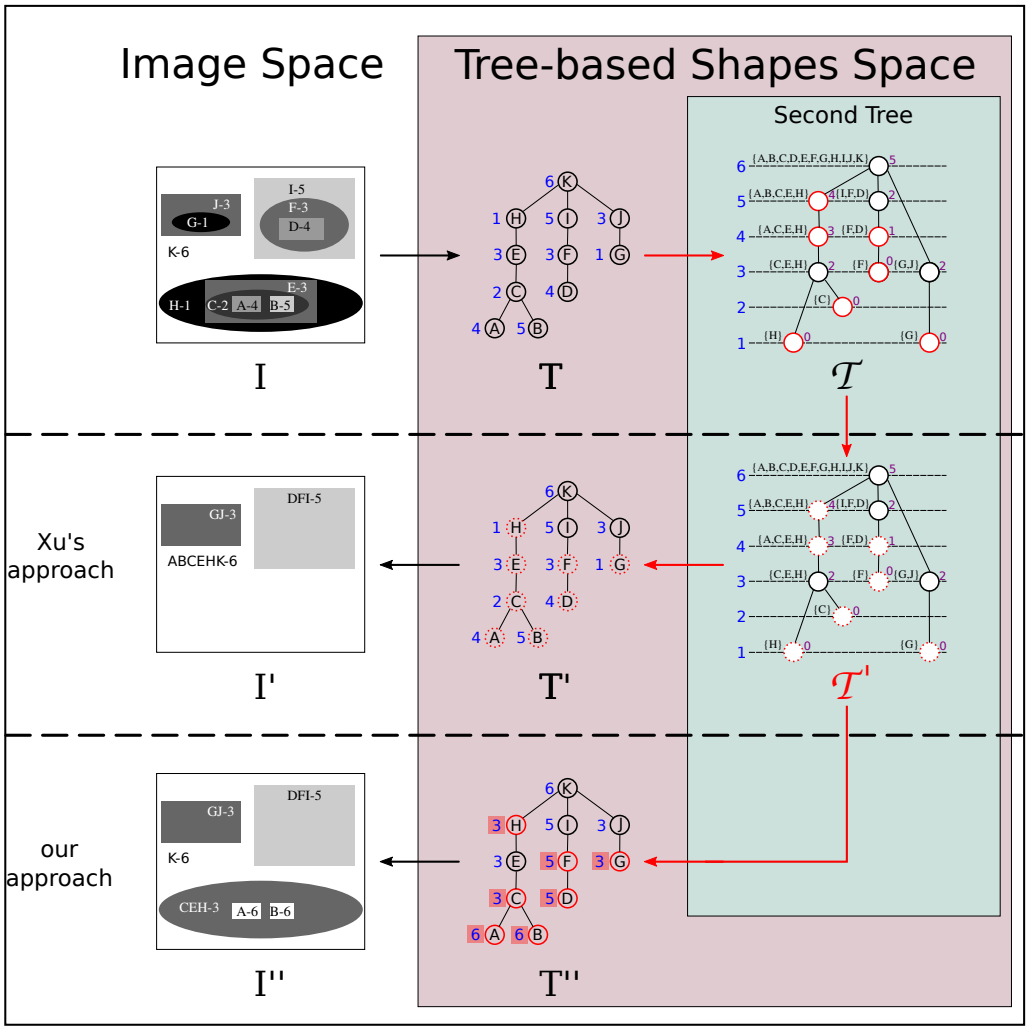
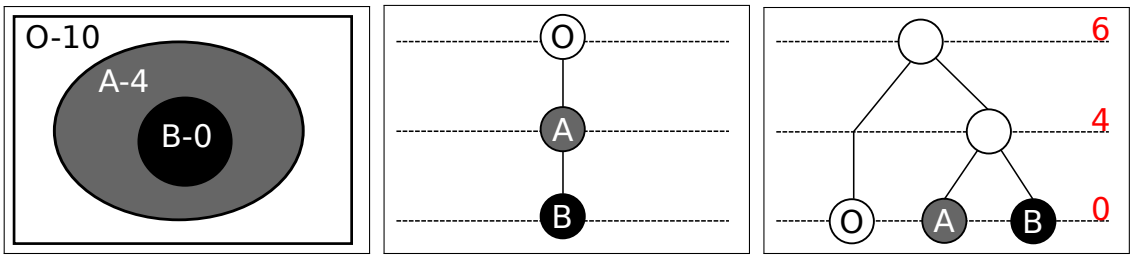


Figure 3.4 – Our approach to shape-spaces filtering compares with Xu's in case the second tree is filtered with a non-pruning strategy. We filtered nodes whose $\mathcal{AA} \neq 2$ while keeping the root node of the second tree to make sure the image domain could be restored.



(a) A simple image (b) The ToS (c) The α-tree

Figure 3.5 – A simple example of ToS and α-tree. The inclusion of level lines is encoded from the ToS, but that information could not be obtained naturally from the α-tree, and other hierarchies of segmentation in general

We acknowledge that sometimes it is easier to form desired shapes in a hierarchy of segmentation, e.g., by controlling the initial segmentation, the region models and merging

criteria of a BPT, or by modifying the dissimilarity function of an α-tree. For instance in Figure 3.7, there are nodes on an α-tree, with dissimilarity function is ΔE^*_{94} [1], that better represent some characters than the ToS. However, because these type of tree form higher nodes by merging adjacent lower ones with no concern for their inclusion, the same inclusion relationship could not be deduced naturally as in the ToS. For example, in Figure 3.5(c), we know that two flat-zones A and B are adjacent because they are both contained in $A \cup B$ (the higher node). However, we cannot deduce that B is included in A or vice versa. This inclusion of a quasi flat-zones' boundary should be retrieved by an external process. For instance, by hole-filling connected components of and verify their inclusion: B is contained in $A \Leftrightarrow \mathrm{Sat}(B) \subset \mathrm{Sat}(A)$. However, this research direction would require more study.

In Chapter 2, because of the simple nature of the ToSoL, we can expect all relevant nodes will be children its background. This dramatical reduction has been proved to be useful in text characters grouping. However, we have based on two assumptions: there exist nodes on the simplified ToS that represent text characters, and related ones are siblings.

The first hypothesis may not hold because it is difficult to adjust the ToS simplification process to preserve objects of interest. A simplification process based on low-level features may not be able to preserve objects of interest at different scales (e.g., text characters). For example, the tree filter using the area and average gradient magnitude (for ToSoL [36]) or the tree filter by minimizing the Mumford-Shah cartoon model (for MToS [16]) both aim at keeping "meaningful level lines" which may not be ones that represent text characters (or parts of them). Text characters may appear in the fine part of the image, which is more likely to be removed on a more simplified ToS. For example, in Figure 3.9, small characters are removed when we increase the simplification level. We may also face "broken" objects on ToS as mention earlier. These broken parts, when being evaluated independently, are easily misclassified as unmeaningful nodes. For example, in Figure 3.8, uneven illumination and reflection make the "A short break" difficult to be segmented correctly. In this case, a higher level of simplification leads to more parts of these characters merged with the background. However, a simplified ToS is more desirable. Although a non-simplified ToS preserves all the detail, having to deal with all the level lines, some of which are slightly different from the other is expensive. For that reasons, we still work with a simplified tree for keeping the computational cost reasonable, but we will try to keep the tree simplification strength low to retain more detail.

The second assumption may not hold in a less simplified ToS. The less simplify the ToS is, the higher the chance those regions that we would call the background will be segmented, i.e., be represented by multiple nodes on the tree. In that case, semantic objects would become cousins, uncle-nephew nodes or even further on the hierarchical structure. One example is presented in Figure 3.10 where sibling regions in a more simplified tree become nephews-uncles in a less simplified one. The segmented background may also re-

1. A color distance metric by CIE

(a) Input Image (cropped).

(b) Labeling Map of a simplified MToS.

(c) Nodes on that MToS that best represent text characters in Input Image.

(d) Shape of the Lowest Common Ancestor of nodes that best represent “s” in “short”

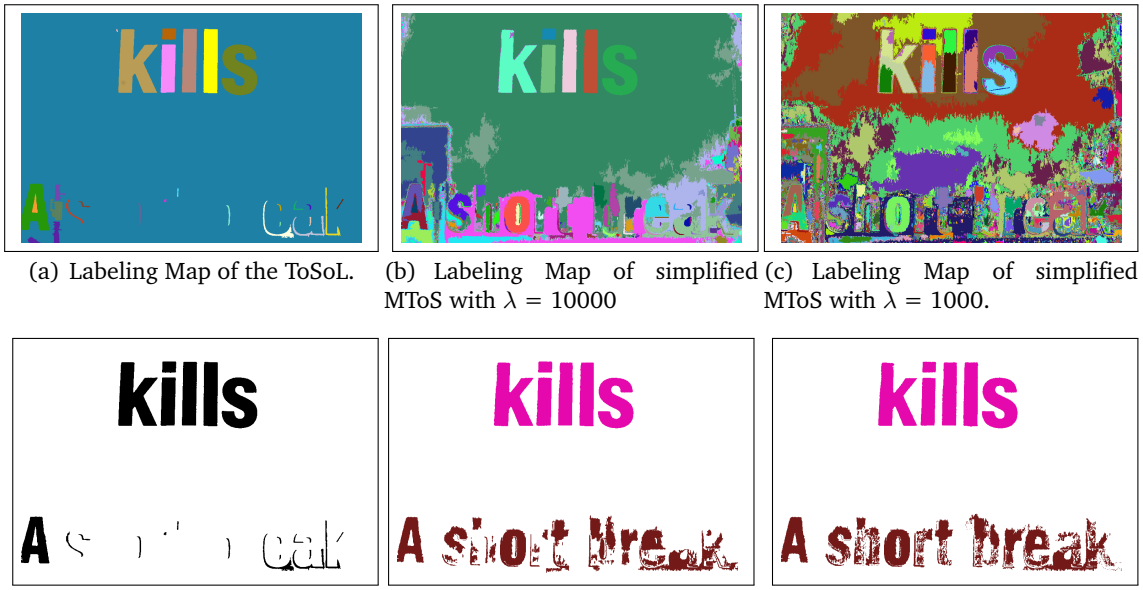
Figure 3.6 – Non-homogeneous objects of interest in MToS. The MToS [16] is simplified by minimizing the Mumford-Shah cartoon model [116] with $\lambda = 1000$. We filter out all nodes, except the root, that is not in the best fitting subset (the set of ToS's nodes that is best fit the ground truth, more detail in Section 3.4.4). The labeling map of that filtered tree is shown in 3.6(b). In case of uneven illumination or reflection that objects of interest is not homogenous, it is possible that a different part of one object will grow in a different direction and merge very high on the tree structure. Because of that, we may not found a ToS node that could represent the object of interest.

sult in new connected components in-between relevant ones. Therefore the nearest sibling (or nephew/uncle or further relative) of a node may not be a semantically related object. For this reason, we have to extend our assumption to adapt to the changes in the tree's structure:

- All ancestors of a node (in a hierarchy that encode inclusion) will be counted as its background.

(a) Input Image (cropped). (b) Best fitting subsets of a simplified MToS. (c) Best fitting nodes of a α-tree

Figure 3.7 – Performance of MToS and α-tree on segmentation of a non-homogeneous objects. The α-tree is computed from the color image using ΔE^*_{94} [1] as the dissimilarity function. The best α-CC that fit the GT are shown in 3.7(c). In this case, the line "A short break" is better segmented using the α-tree.



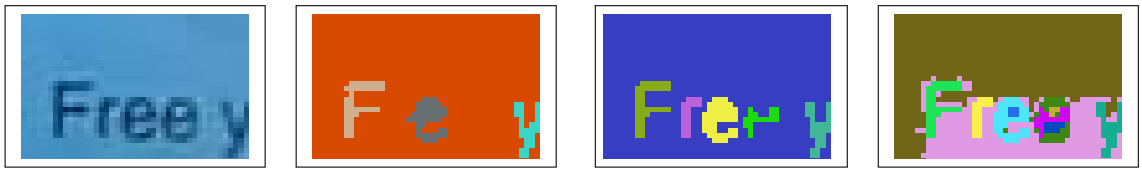
(a) Labeling Map of the ToSoL. (b) Labeling Map of simplified MToS with $\lambda = 10000$ (c) Labeling Map of simplified MToS with $\lambda = 1000$.

(d) Best fitting subset of ToSoL whose labeling map in 3.8(a). (e) Best fitting subset of MToS whose labeling map in 3.8(b). (f) Best fitting subset of MToS whose labeling map in 3.8(c).

Figure 3.8 – Different levels of tree simplification (where broken text characters occured). Trees are built on from the image in 3.7(a). The ToSoLs (Chapter 2) are built with an 11x11 pixels window and the average gradient magnitude threshold is 30. The MToSs [16] is simplified by minimizing the Mumford-Shah cartoon model [116]. The higher λ is, the more simplified the MToS. The best fitting subset is the set of ToS's nodes that are best fit the ground truth (more detail in Section 3.4.4). A simplification process using low-level features (keep only contrast 0-crossing in ToSoL [36] or meaningful level lines [116]) may lead to loss of desire level lines.

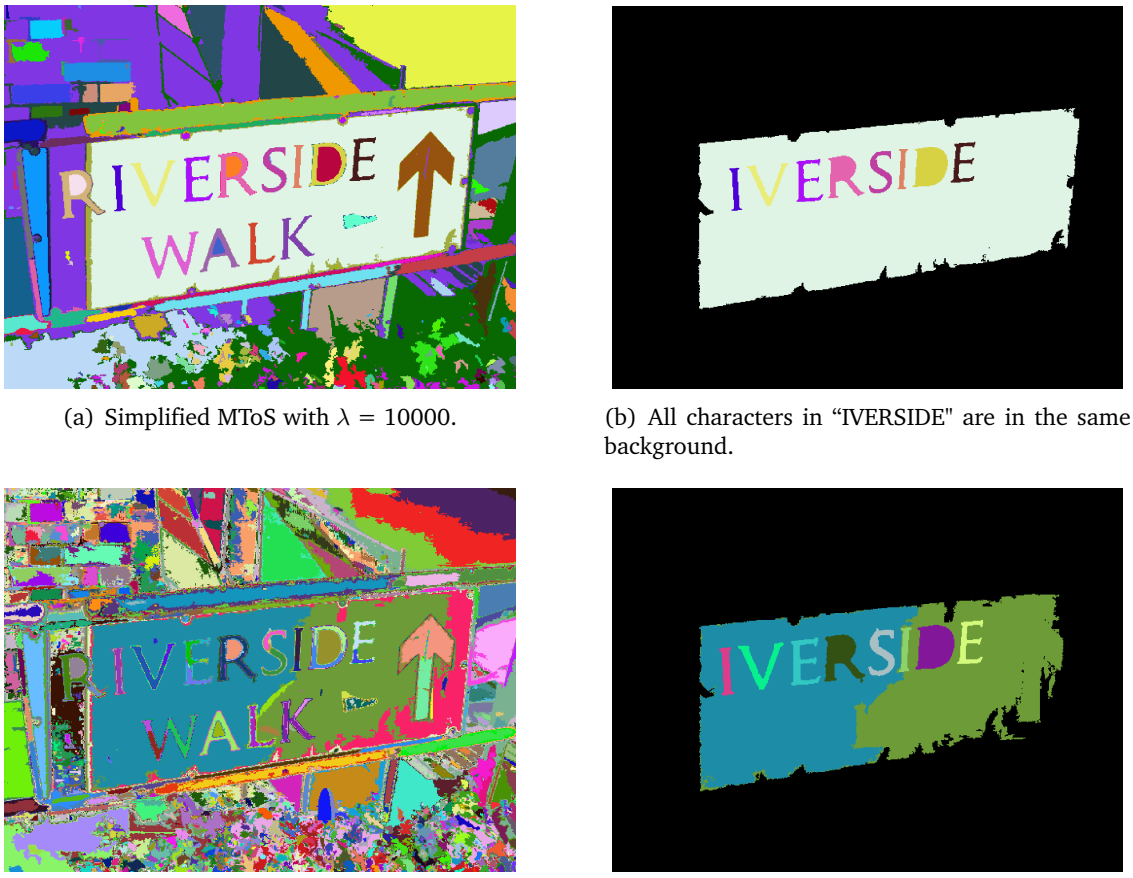
- No limit on the number of neighbors a node could have.

The first point compensates the relative relationship on a complicated hierarchical structure. It extends the image region treated as background for each object. This modification allows nodes that have cousins, nephew-uncle relationship and further could be



(a) input (cropped). (b) Labeling Map of the ToSoL. (c) Labeling Map of the simplified MToSwith λ = 10000 (d) Labeling Map of the simplified MToS with λ = 1000.

Figure 3.9 – Loss of detail at different levels of tree simplification (small text characters). ToSoL and MToS are simplified by same parameter as in Figure 3.8. Small text characters are more likely to be removed on a more simplified tree, e.g. letter “e” in “free”.



(a) Simplified MToS with λ = 10000. (b) All characters in “IVERSIDE" are in the same background. (c) Simplified MToS with λ = 1000. (d) Characters in “IVERS" and “IDE" are nephews-uncles in the tree structure.

Figure 3.10 – Illustration of the necessity to expand the background assumption. Labeling map of a simplified MToS [16] with different level of simplifying (left) and some noticeable regions (right). Compare to figure 3.10(a), the ToS whose labeling map is in figure 3.10(c) is less simplified. In consequence, some sibling nodes (for example “S” and “D”) in figure 3.10(a) now become uncle (“D")- nephew (“S") in a less simplified ToS. Note that in both cases, the first letter ’R’ of “RIVERSIDE" is further on the hierarchical structure. For that reason, reflect the objects-background relationship only by children-parent relationship on the ToS is not satisfaction.

viewed as having the same background. The second point, let the searching process continue even if there are unrelated segments in-between semantic ones. Although it will add more edges, it also retains more information for later processing. The neighbors finding could be done with the help of a capable classification method that could tell the neighbors is an object of interest. However, we avoid making binary decisions (because "broken" objects would be difficult to recognize when it is treated independently).

Rather than a disconnected graph where each connected components is a text candidate as the earlier approach in ToSoL, this expansion results in a more complex graph. In this new graph, each vertex corresponds to a node on the ToS from which it is constructed. The connectivity of this graph still signifies the spatial relationship w.r.t the inclusion between the shapes of the ToS. To be specific, neighbors of a node are those it is not included in and has a spatial relationship (which depend on the searching process). We call this graph the spatial alignment graph. Because of the extension above, we could not extract the text candidate directly as have been done with ToSoL. Instead, we will weight this graph by those properties which were previously handled by binary decisions, e.g., the distance on the tree, the distance on image, geometry or color similarity.

3.3.2 Handling the spatial alignment by spatial search in the image space

With the extension above, we can handle the inclusion relationship on the ToS. However, the set of possible neighbors of a node (those that are not its ancestors, which is part of its background, nor its descendants, of which it is the background) is significantly larger. Analyzing all these possible neighbor is computation expensive. Instead, we will define neighborhood with a priori knowledge about the spatial arrangement of objects on image space or distance on the hierarchical structure (i.e., the scale of objects of interest). This does not only reduce the size of the graph but also make it more meaningful.

For instance, we could usually assume the spatial arrangement of objects of interest to restrict the neighborhood, e.g., Latin text usually horizontal or vertical. Along with the inclusion relation, we could effectively reduce the graph size. Aligned regions could be found by computing the relative position of a node with all its possible neighbors. However, because of the size of that set is usually large, the computation would be expensive. For instance, in Figure 3.11, 353 nodes are neither descendant nor ancestor of the one represents the letter 's' in 'kills' but only five are horizontally aligned. For that reason, we prefer making alignment links by a spatial search on the image space.

We could set a maximum distance for the searching process instead, both on image space (e.g., measured by the number of pixel separate two regions) and the tree space (e.g., measured by the number of edges of the shortest path connecting two nodes on the ToS). These limits should be set relatively to the node's size so that it does not favor one particular scale. We will address these distances in the weight of the edges by using a penalty function that reduces (res. increase) the similarity (res. dissimilarity) weight on edges.

(a) Labeling Map of simplified MToS of 3.7(a)

(b) Possible neighbor of 'S'

(c) Horizontally aligned neighbor w.r.t inclusion of 'S'.

Figure 3.11 – An example of aligned components. There are 353 nodes that are neither descendant nor ancestor of the one represents the letter "S". However, only 5 nodes are horizontally aligned with "S". These nodes could be retrieved by checking the labeling map on a horizontal line (e.g. the red line in 3.11(a)), i.e. a spatial search on image space. That approach is less computationally expensive than measuring the relative spatial position of "S" and all of its possible neighbors.

There are several strategies to do the spatial search. We could use a single searching line or multiple parallel searching lines. Single line searching strategy checks pixel by pixel (on the labeling map) for components other than the ones marked as background [2]. To implement searching line strategies, we need to define the line rotation based on context. We use horizontal lines for roughly horizontal aligned text (e.g., Latin script). Single searching line strategy works on a line going through the center of the current component. Multiple parallel searching lines strategy checks pixels on multiple parallel lines. Compare to the former, the latter has a higher cost but is more robust against variation in height or when the assumed direction is off from the real alignment. In our implementation of this strategy, we use three parallel lines, one goes through the center of the component and two line that is 40% n_{height} away and parallel with the center one. An example of one line searching strategy is presented in Algorithm 3.

3.3.3 Section summarization

To summarize, this section introduces a shape-space that is created with both inclusion and spatial alignment information. It is represented by a graph of shapes that we call spatial alignment graph with respect to inclusion or spatial alignment graph in short. The neighbors of a node are ToS's shapes that are aligned and not in the same branch.

To construct this representation, we first create a graph $G(V, E)$ with empty edges set $E = \emptyset$ and the set of vertices V is the set of nodes on ToS. The set of edges will be created by doing a spatial search on image space from each vertex $v \in V$. A step-by-step example is given in 3.12. We will weight the edges set by a combination of similarity between shapes, as well as the distance on image space and distance on the tree. We analyze this graph with the shape-spaces morphology framework.

2. Note that the relation "is an ancestor of" could be verified in constant times in our implementation with Pylene.

Input: Tree of Shapes ToS, direction vector of the searching line $shift$, maximum distance md
Output: adjacency graph $G(V, E)$

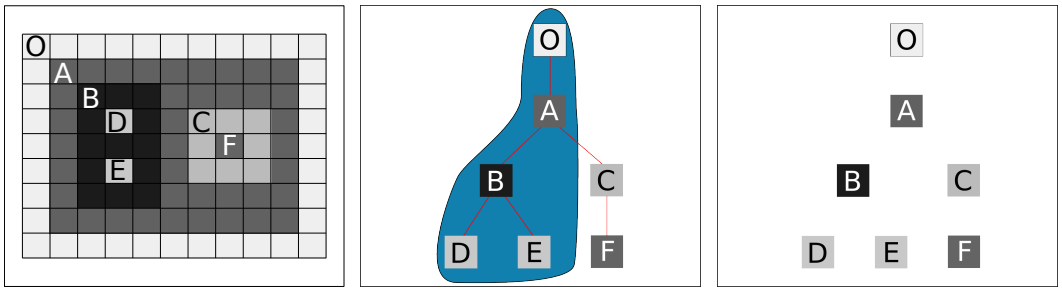
```
1  ComputeGraph(ToS, shift, md):
3      G(V,E).init_V (ToS.nodes()) ;
5      for n ∈ ToS.nodes() do
7          marker = vector < bool > (ToS.nodes(), true);
9          oneLSearch(G(V,E), n, shift, marker, md);
11         oneLSearch(G(V,E), n, −shift, marker, md);
13     return G(E,V) ;
14 oneLSearch(G(V,E), n, shift, marker, md):
16     p = n.center();
18     while p ∈ G_domain and |p − n| ≤ md do
20         n' = G.getNode(p) ;
22         if not marker[n'] or n'.isAncestor(n) or n.isAncestor(n') then
24             continue ;
26         G.addEdge(V[node], V[n']);
28         marker[n'] = False ;
30         p+ = shift ;
```

Algorithm 3: Computation of adjacency graph $G(E, V)$. The relation "is an ancestor of" could be verified in constant times in Pylene, see Appendix A for detail. We could also mark ancestor and descendant of n before calling the searching procedure.

3.4 An α-tree on the Spatial Alignment Graph W.r.t Inclusion for Text Detection

In this section, we will focus on the application of text detection on natural images, where regrouping similar components to form a set of text candidate is important. Because text characters are usually similar (e.g., size, color...), regrouping similar segments before OCR would reduce effort in detection and also serve as a false positive elimilation step.

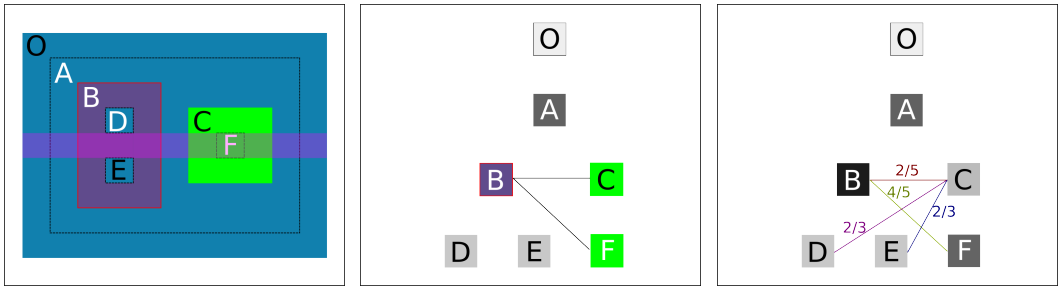
We will apply the shape-space morphology framework to the spatial alignment graph to obtains groups of similar and aligned objects. Each node on the second tree will represent a group of similar components. Let us remark that both inclusion and spatial arrangement are embedded in the connectivity of this spatial alignment graph. Thus, further analysis that relies on connectivity will passively rely on the semantic relationship implied by this information. Other assumptions about objects will be used to weight the edges of the spatial alignment graph with a notion of similarity between shapes. From that edges weighted graph, an α-tree is constructed. We will demonstrate that nodes on that α-tree provide good text candidate in compare with the best possible grouping result. We also propose some simple approach to select a small set of nodes to serve as the text candidate set.



(a) Input and .

(b) ToS and branch contains node B (blue).

(c) Edgeless spatial alignment graph with vertices correspond to ToS nodes .



(d) Spatial search for B's neighbors.

(e) Edges added after a spatial search from B.

(f) Final Graph.

Figure 3.12 – Example of an spatial alignment graph created with searching line strategy. From the input image in Figure 3.12(a), the ToS is computed in Figure 3.12(b). We will run a spatial search for every vertex on the edgeless graph in Figure 3.12(c). For example, from node B, with nodes on its branch (blue) ignored, we found C and F having pixels on the horizontal line through B (Figure 3.12(d)). This will result in two new edges in our spatial alignment graph (Figure 3.12(e)). Continue for every node (except the ToS root) to obtain the final graph on Figure 3.12(f). Here, we weight the edges by height similarity between their ends.

3.4.1 Spatial alignment graph w.r.t inclusion for text detection

First, we have to specify the parameters of the spatial alignment graph that we use for our application. We first built the spatial alignment graph with similar hypotheses as in Chapter 2: semantic text characters are in the same backgrounds, and they are roughly horizontal aligned; they should have some similarity in characteristics (e.g., height, color...). We will use the former criteria to define the neighborhood of our spatial alignment graph and the latter to weight the edges of our graph.

We first compute a tree of shapes of the input image. Cameras usually capture images in the sRGB color space which results in a large base of three-channels images. To avoid imposing a total order on the vector space, we use Carlinet et al.'s MToS [16] as the inclusion tree representing the input image. The ToS T is then simplified by minimizing the Mumford-Shah cartoon model constrained by the tree topology. The simplified tree is obtained as a subset of shapes $T' \subset T$ that minimizes the energy function $E(T') = \sum_{R \in T'} \sum_{x \in R | R_x = R} ||f(x) - \bar{f}(R)||_2^2 + \lambda |\partial R|$. With R_x denote the smallest shape containing a

point x, $\bar{f}(R)$ is the average value of the region R and $|\partial R|$ the length of the shape's boundary. The minimization is done using a greedy algorithm provided by the authors of [116]. In short, we sort the ToS's nodes by meaningfulness and then remove the less meaningful one step by step until the image energy stop decreasing. The parameter λ affect the level of simplification: the higher λ is, the more simplified the ToS is.

After obtaining the simplified ToS T, we then create the spatial alignment graph w.r.t inclusion. Using one horizontal searching line strategy, we obtain approximately aligned neighbor of every node. In our approach, we do not set a maximum distance on the hierarchical structure nor a maximum distance on the tree. However, these distances will be used as descriptors for nodes of the second tree.

A second tree will be built from this shape-space. There is two type of hierarchical representation we can choose from: the tree based on threshold decomposition and hierarchy of segmentation. The former, which includes Min-, Max-tree, ToS, is built from a node-weighted graph. They can handle multivariate features (in the same way as the MToS). However, this type of tree is somewhat extrema oriented (i.e., leaves are local extrema). It focuses more on the node's weight than the connectivity makes it difficult to control and obtained the desired group. The other type of tree (which includes BPT and α-tree) generally based on the dissimilarity between image regions. Since the objective is to obtain groups of similar components, we prefer this type of tree because it is easier to control. We consider the α-tree since its single linkage structure allows us to faster group a long text line with small variation between consecutive characters. Moreover, it could be computed efficiently by a modified Kruskal's algorithm [65]. When using with a custom dissimilarity measure, we could force the creation of the desired node on the tree.

To construct a hierarchy of segmentation, the edges of the spatial alignment graph have to be weighted by a dissimilarity between their ends. We consider some properties, which are usually used in text grouping methods: the similarity between the bounding box dimension, filling rate, color ....

- **Shapes' dimension**: We should compare shapes' height for horizontal texts since characters should not change significantly between semantically related text characters. Even in the case of perspective distortion, nearby characters' height does not dramatically change. The height and width should be obtained by the major and minor elongation of shapes. However, because the minimum bounding boxes could be obtained easier during construction of ToS, we prefer using them as an approximate. We could define the height similarity by the minimum over the maximum:

$$S_{height}(n_1, n_2) = \frac{min\left(height_1, height_2\right)}{max\left(height_1, height_2\right)}$$

- **Height over width**: This ratio should not vary much between semantically related text characters. With $HoW_x = \frac{H_x}{W_x}$ the height over width of node x, the similarity

could be defined by:

$$S_{HoW}(n_1, n_2) = \frac{min\,(HoW_1, HoW_2)}{max\,(HoW_1, HoW_2)}$$

- **Filling rate**: or sometimes called the saturation rate is the shape's area over its boundibox's area:$FR(n) = \frac{area(n)}{H*W}$. The similarity could be defined as:

$$S_{FR}(n_1, n_2) = \frac{min\,(FR_1, FR_2)}{max\,(FR_1, FR_2)}$$

- **The color different**: The standard means of determining the different in color is the Euclidean distance. For example, in RGB space:

$$\Delta_{RGB}(n_1, n_2) = \sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}$$

There are also attempt to weight each element differently RGB to better fit human perception. A good approximation would be the coefficients of 3,4 and 2 [58]:

$$\Delta_{W_{RGB}}(n_1, n_2) = \sqrt{3(R_2 - R_1)^2 + 4(G_2 - G_1)^2 + 2(B_2 - B_1)^2}$$

Although their computations are efficient, these measure does not include changes in sensitivity of the eye caused by changes in the brightness. The International Commission on Illumination (CIE) define a distance metric CIE L*a*b* $\Delta E*$ which take into account the human perception of color. The first CIE $\Delta E*$ was just the Euclidean distance in the CIE L*a*b* color space. Their latter formulas, ΔE^*_{94} [1] and ΔE^*_{00}[88], address the perceptual non-uniformities. They take into account weighting factors for lightness, chroma, and hue. [3]

We also address the distance between regions in image space and tree space by multiply the similarity above to a penalty factor. This factor is defined in such a way that closer regions should have a higher similarity measure and vice versa. We define the penalty function to be non-increasing, bounded between 0 and 1 for this purpose:

$$\mathcal{G}(tor, max)_{n_1,n_2} \colon \Delta(n_1, n_2) \in (N) \to [0, 1]$$

With $\Delta(n_1, n_2)$ the distance between two node n_1, n_2, tor is a tolerant distance and max the maximum distance we want to make the connection. The latter two are the parameters of our method. In case of distance on the image space, the max is implemented directly in the searching process (by stopping that process after getting out of the maximum range).

3. All RGB to L*a*b* conversion in this work has been made using the D65 white point, standard 2° observer angle.

Because a node is a set of image pixels, distance between them should be either the length of the shortest path connecting points between two node, i.e. the minimum distance between two set:

$$\Delta_I(n_1, n_2) = min\{||x - y||, \forall x \in n_1, y \in n_2\}$$

However, since that distance is conputational heavy, we use the Euclidean distance between two bounding box center as an approximation:

$$\Delta_I(n_1, n_2) = ||c_1 - c_2||$$

The distance on the tree space is more straightforward: it is the length of the shortest path connecting two nodes on the tree which equal to:

$$\Delta_T(n_1, n_2) = depth(n_1) + depth(n_2) - 2 * depth\left(lca(n_1, n_2)\right)$$

With lca the lowest common ancestor and depth is the distance from root. [4]

To sumarize, we will define the penalty function as:

$$\mathcal{G}(tor, max)_{n_1,n_2} = \begin{cases} 1 & \text{if } \Delta(n_1, n_2) \leq tor \\ 1 - \frac{\Delta(n_1,n_2)-tor}{max-tor} & \text{if } tor \leq \Delta(n_1, n_2) \leq max \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

An example is given in Figure 3.13. In that example, we set a large max, which is the image width in image space and twice the height of the tree, so that we retain most edges. The tor is set to about 20% max. These value will be used in the test presented later in this Chapter if not stated otherwise.

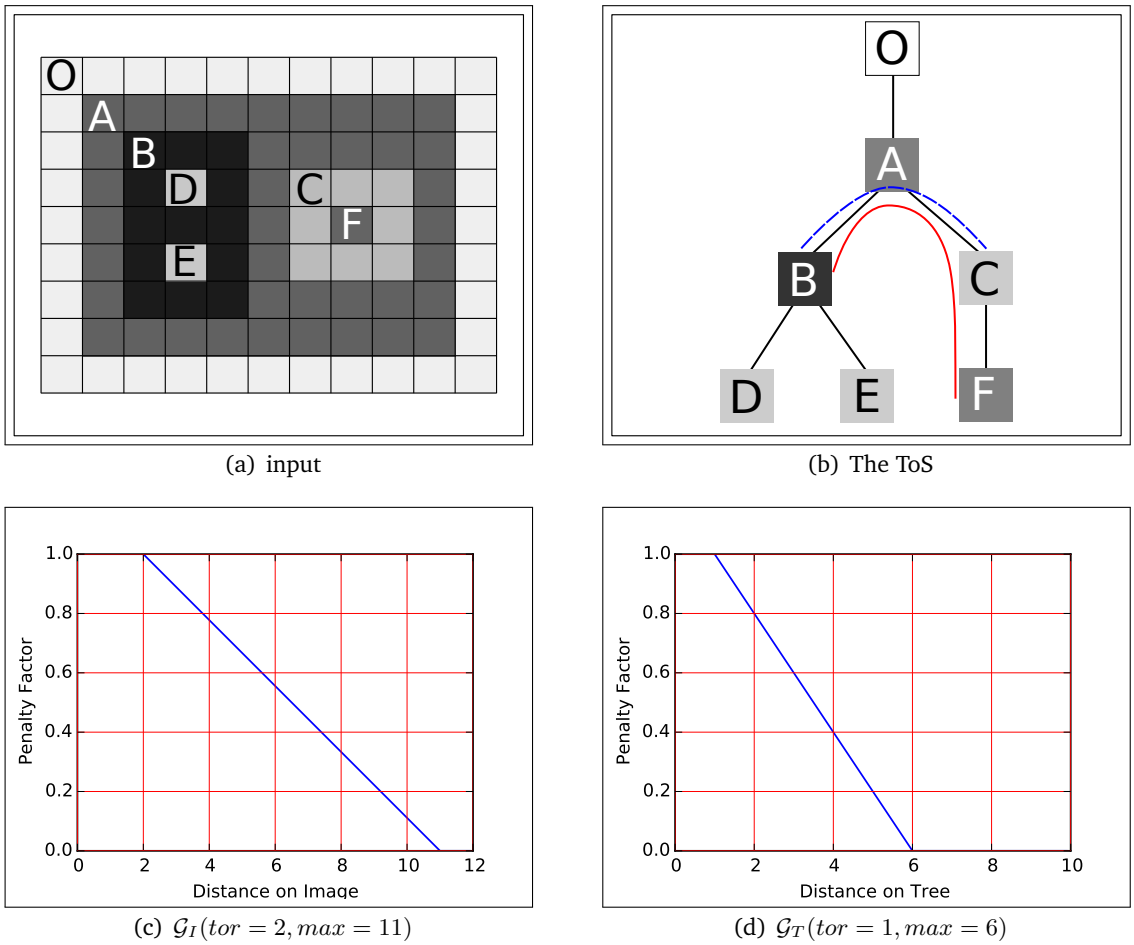
To keep the notation consists with the classic α-connectivity on image, we assume the spatial alignment graph $G(V, E)$ has its edges weighted by a dissimilarity value DS which measure the difference between two node. If the measure is a similarity (e.g. minimum over maximum), let the dissimilarity is the complement of that measure. For example

$$DS_{height} = 1 - S_{height} * \mathcal{G}_I(tor_I, max_I) * \mathcal{G}_T(tor_T, max_T)$$

3.4.2 Alpha-connectivity and alpha connected components of the spatial alignment graph

We first look at α-connectivity and α-connected components on our spatial alignment graph since it is the basic of the α-tree. Let us recall the α-connectivity notion: two node x,y are α-connected if there exists a path $\pi(x \to y) = \{x = x_1, x_2, ..., x_N = y\}$ from x to y that the dissimilarity between two consecutive node $DS(x_i, x_{i+1}) \leq \alpha$.

4. The lca could be obtained effectively thanks to the component tree structure in Pylene. See Appendix A.



(a) input (b) The ToS

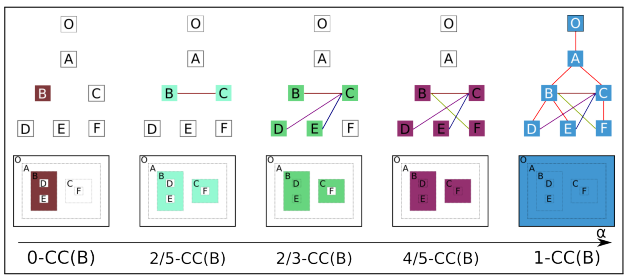
(c) $\mathcal{G}_I(tor = 2, max = 11)$ (d) $\mathcal{G}_T(tor = 1, max = 6)$

Figure 3.13 – Illustration of penalty factors. (a) The input image; (b) Its ToS, the blue dash is the path from B to C and the red one is the path from B to F; (c,d) Examples of the penalty function on image space and on tree space. $\Delta_I(B, C) = \Delta_I(B, F) = 4$ because C and F have the same center. On the other hand, $\Delta_T(B, C) = 2$ and $\Delta_I(B, F) = 3$. The weight of edge B-F in Figure 3.12(f) (which is a similarity measure) will be multiplied by a factor of 0.47 and 0.62 for edge B-C.

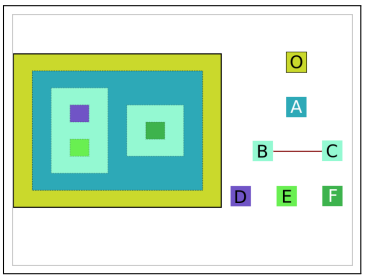
x,y are α-connected $\Leftrightarrow \exists\pi(x \to y) = \{x = x_1, x_2, ..., x_N = y\}$ and $DS(x_i, x_{i+1}) \leq \alpha; \forall i \leq N$

An α-connected component containing a vertex x, α-CC(x), is the set contains x and all vertices that is α-connected to x. A set of all α-CCs with a given α value is a partition of input graph $G(V, E)$. Example of α-CCs of the spatial alignment graph are given in Figure 3.14 and 3.15.

Because it is defined based on local dissimilarity, one may concern this connectivity is also affected by "leakages": two regions in $G(V, E)$ with distinct global properties could be merged together because of some small number of interconnections (e.g., Figure 3.16). One common approach for this problem is using a global parameter ω to con-



(a) α-CC(B) of the spatial alignment graph in Figure 3.12(f) with different α value



(b) a partition of image in Figure 3.12(a) by 2/5-CCs

Figure 3.14 – An example of α-CCs extracted from the spatial alignment graph. α-CCs on the second tree are presented by reconstructing the image following our approach shown in Section 3.2.3.3. With the spatial alignment graph in Figure 3.12(f) weighted by height dissimilarity DS_{height}, we see in Figure 3.14(a) that the α-CC(B) grows when similarity threshold increase. In case the graph is complete, the α-CC(B) would grow until it covers the whole graph. The 1-CC(B) is presented as if unconnected edges on Figure 3.12(f) existe with dissimilarity value 1. If it is not the case, 1-CC(B) = 4/5-CC(B). In Figure 3.14(b) all 2/5-CCs is shown, which provides a partition of input image.

trol the growth of α-CC. We have reviewed Hambrusch's and Soille's approach on the use of a global parameter on image space in Section 1.2.3. As discussed earlier, the (α, ω)-Hambrusch-CCs are not uniquely defined, therefore we cannot create a hierarchical based on this notion without an arbitrary choice. However, it can still find application in extracting objects with input markers. Soille's (α, ω)-connectivities cannot create connected components that could not be obtained by α-connectivities in the first place, i.e., it cannot prevent "leakage" from happening.

On the other hand, α-CCs in the spatial alignment graph do not grow as fast as the α-CC on the gray-scale image. It is because the value space of the proposed dissimilarity functions is larger than the typical gray-scale one. Even if we quantizer to reduce the number of levels, we have total control of that process to make sure the CCs does not grow too quickly. Therefore the leakage is not a big problem in our approach.

3.4.3 Alpha-tree of the spatial alignment graph

The chains of α-partitions on our graph increasing α value form a hierarchy. Unlike in images, there may not exist an α-CC that covers the whole shape-space because the input graph $G(V, E)$ may not be connected. For that reason, the set of all α-CCs, ordered by inclusion, will usually form a forest. To make it a single tree, we define a root node which is the set of all ToS's nodes. This result comes naturally by seeing $G(V, E)$ as a complete graph and any edges that are not originally in E will have the maximum dissimilarity. The α-tree of Figure 3.12(a) is illustrated in Figure 3.17. We could construct the α-tree with Najman's algorithm [65].

As the value space of dissimilarity measure may be continued, the α-tree is mostly a binary tree, i.e. in most case, a node only have two children. One may want to quantizer

(a) A natural image

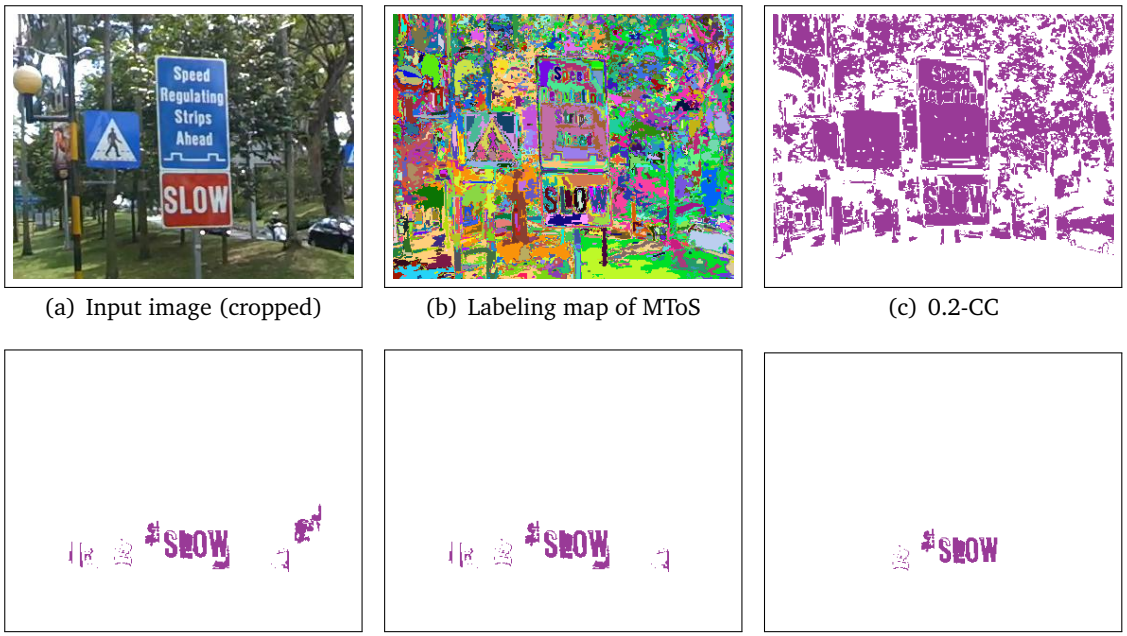
(b) Segmentation of input by a simplified MToS (sMToS)

(c) 0.1-CCs of the spatial alignment graph of sMToS

(d) Some notable 0.1-CCs

Figure 3.15 – An example of α-CCs extracted from the spatial alignment graph of a natural image. α-CCs are presented by the image regions they represent. The natural image 3.15(a) belongs to the ICDAR RRC test sets [44]. The MToS [16] is simplified by minimizing the Mumford-Shah cartoon model [116] with a low simplification parameter ($\lambda = 1000$). This provide an under segmentation of the image 3.15(b). The spatial alignment graph is constructed using single horizontal line search and weighted by DS_{height}. Its 0.9-CCs are shown in 3.15(c). Some notable 0.9-CCs are shown in 3.15(d).

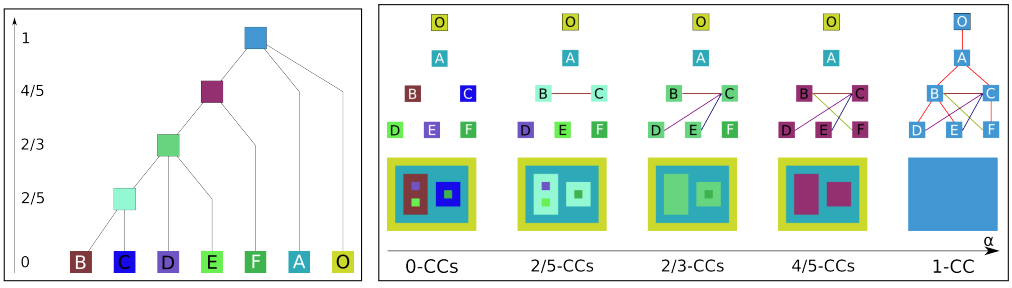
the dissimilarity measure, either to reduce the value space to a certain number of bits, or to make the difference between two level meaningful (e.g. by each amount of 2.3 for the ΔE^*_{76} in L*a*b* space, which is its Just Noticeable Difference (JND) [87]).



(a) Input image (cropped) (b) Labeling map of MToS (c) 0.2-CC

(d) $(0.2, 0.5)$-Hambrusch-CC (e) $(0.2, 0.5)$-Soille-CC (which is 0.13-CC) (f) Best fitting node (0.06-CC)

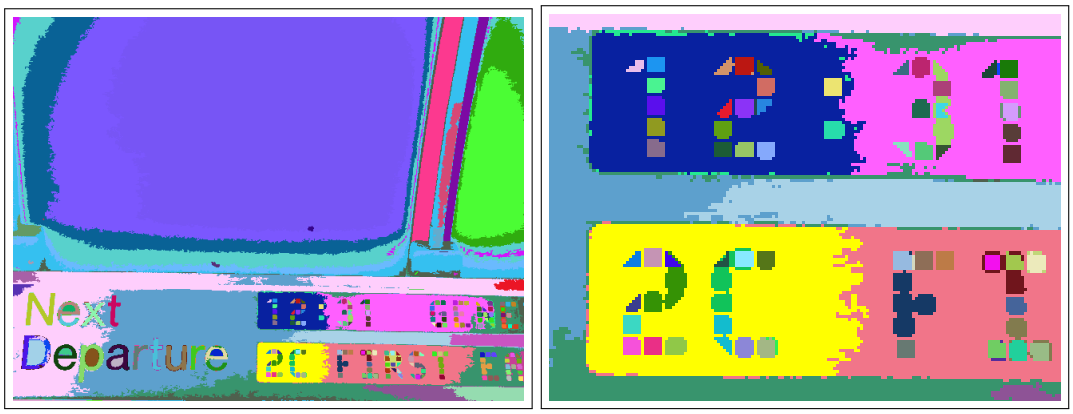
Figure 3.16 – "Leaking" on an α-tree of the spatial alignment graph. The simplified MToS is compute from the input image on 3.16(a). The image region represented by each node is marked with unique color in 3.16(b). The spatial alignment graph's edges are weighted by DS_{height} to highlight the leaking effect. We could see the 0.2-CC that contains the shape of 'S' in 'SLOW' (3.16(c)) cover a large potion of the image. The DS_{height} between the smallest and largest component is approximate 0.93. Using a global parameter could reduce the leaking. $(0.2, 0.5)$-Hambrusch-CC (in 3.16(d)) and $(0.2, 0.5)$-Soille-CC (in 3.16(e)) are much smaller and contain more relevants shapes. They are different however, $(0.2, 0.5)$-Hambrusch-CC is computed following his greedy algorithm in [31] and (0.2,0.5)-Soille-CC is actually the 0.13-CC (the largest α-CC that satisfy both local and global condition). Since our tree grow slowly, we actually can find better nodes such as the 0.06-CC in 3.16(f).



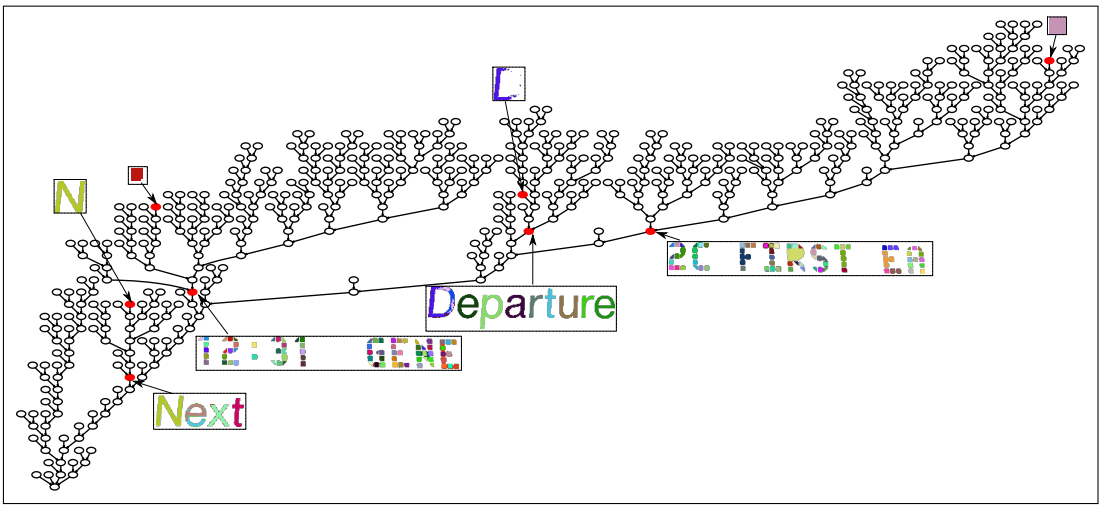
(a) α-tree. (b) α-CCs at different thresholds.

Figure 3.17 – Example of α-tree on the spatial alignment graph. α-tree and α-CC at different similarity threshold for the graph in Figure 3.12(f) with implication that it is a complete one and all unshown edges are weighted 1.

(a) Input Image and a zone of interest.



(b) Labeling map of simplified MToS.



(c) α-tree and some notable α-CC

Figure 3.18 – Example of α-tree on the spatial alignment graph of a natural image. The spatial alignment graph built from the simplified MToS ($\lambda = 2000$) whose labeling map shown in 3.18(b) as described in Section 3.4 and the edges are weighted by DS_{lab}. The α-tree on that graph is shown in 3.18(c) with some notable nodes shown in red. The regions represented by these nodes are colored corresponding to the ToS node they belong to

3.4.4 Quality of nodes on the alpha tree

The group of text characters should be retrieved from the nodes on the alpha tree. However, because there are different possibility de define the dissimilarity measure, we would like to measure the quality of nodes on the alpha tree, i.e., the performance of the grouping process. Because objects of interest may not be represented or represented partially as segments on the ToS, we will compare the best fitting nodes (bfn) on the α-trees with the best fitting subsets (bfs) of the ToS. The best fitting node is the node that represents the most similar image region to ground truth. The best fitting subsets are the set of nodes whose representing image regions is the most similar to ground truth.

We use the Intersection over Union (IoU) as the evaluation metric to measure how similar a segmentation is in comparison to the GT. With I_D the intersection between detection and GT, which have area R_D and R_{GT}, the IoU score is:

$$IoU = \frac{I_D}{(R_{GT} + R_D - I_D)}$$

The quality q of the tree for each element of the GT will be determined by the ratio between the IoU score of bfn and bfs:

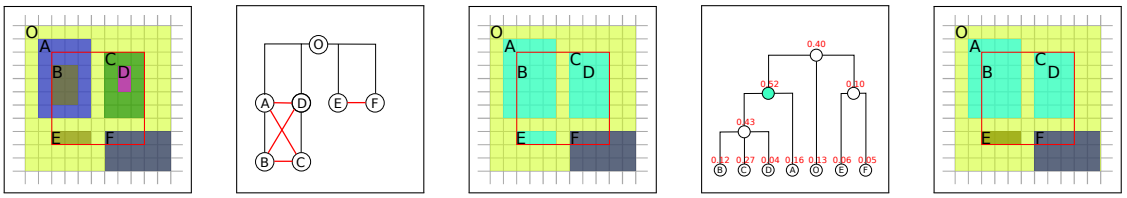
$$q = \frac{IoU_{bfn}}{IoU_{bfs}}$$

Since nodes on the alpha tree is obtained by merging nodes of the ToS, $IoU_{bfn} \leq IoU_{bfs}$ and $q \leq 1$.

The ground truth contains the pixel-level mapping to each class of characters and the background. In the ground truth images, white pixels should be interpreted as background pixels, while non-white pixels as text. Each class of characters is a group of text characters that have semantic relations: on the same background (e.g., on the same line, on the same banner, on the same sign) and horizontally aligned. The pixels are color coded so that the pixels of each class has a unique color.

First, we determine the best fitting node from the α-trees. We evaluate every node on that tree by how good of a segmentation obtained from that node in comparison to each class of the ground truth (GT) and chose the best. In our implementation, the tree-based image representation is stored by a parent map, which tells the parent of each node; and a labeling map, which maps each pixel to the lowest node to which it belongs. To get all the intersection, we check the labeling map for all node that intersects with the ground truth and then propagates the intersection count up the tree structure. Having the intersection, computing and obtaining the node with the highest score is trivia. An example of btn is given in Figure 3.19.

After that, we will retrieve a best fitting subset of first tree's nodes. Such as with the α-trees, we can easily obtain the set S of nodes that intersect with the GT. From there, we will choose a subset $S' \subset S$ that maximize the IoU score. Let denote the intersection of a node or a set of node x and the GT as I_x and its area as R_x. The score of a subset $S' \subset S$



(a) Segmentation by a ToS and GT(regions limited by red box)

(b) The tos (black edges) and a spatial alignment graph (red edges)

(c) Best fitting subset (cyan)

(d) An alpha tree of the spatial graph

(e) Best fitting node (cyan)

Figure 3.19 – Illustration of best fitting subsets and best fitting nodes. We have an image with flat zones noted by letters in 3.19(a). Its ToS and the spatial alignment graph is presented in 3.19(b) (with the same set of nodes, ToS edges are colored black and spatial alignment graph's edges are colored red). The best fitting subset of the ToS's node is {A,B,C,D,E} and these nodes represent the cyan regions in 3.19(c) with the $IoU = 0.57$. Let assume there is a dissimilarity function that can produce an alpha tree of the spatial graph as in 3.19(d). The red number above each node on 3.19(d) is the IoU score of that node compare to the red GT. The best fitting node represent the cyan region on 3.19(e) with $IoU = 0.52$. The quality ratio will be $\frac{0.52}{0.57} = 91\%$

after adding a new node x to S' is:

$$IoU_{S' \cup \{x\}} = \frac{I_{S'} + I_x}{R_{GT} + R_{S'} - I_{S'} + R_x - I_x}$$

We observe that $IoU_{S' \cup \{x\}}$ always increase if $I_x = R_x$. Therefore we initialize set S' with all nodes in S whose $IoU = 1$. We continue insert $x \in S - S'$ that maximize $IoU_{S' \cup \{x\}}$ until $S - S' = \emptyset$ or $IoU_{S'} > IoU_{S' \cup \{x\}}$. For example in Figure 3.19, we first generate a set of nodes that totally included in the GT $S' = \{B, C, D, E\}$. Among other nodes that proper regions intersect with the GT (which are A,O,E), adding A will increase the score the most, therefore $S' = \{A, B, C, D, E\}$. After that, adding any other nodes (F or O) will decrease the score.

We run the comparison on a base of 233 images from the ICDAR RRC: Focus Scene text. Among some text features mentions in the earlier section, we define some the dissimilarity measure based on component heights and color. We do not consider filling rate FR or height over width HoW because they would have a low performance with "broken" text. With $\mathcal{G}(tor, max)$ the penalty function 3.2, these measure are:

- $DS_{height} = 1 - S_{height} * \mathcal{G}_I(0, I_{width}) * \mathcal{G}_T(0.2 * T_{depth}, 2 * T_{depth})$
- $DS_{rgb} = 1 - (1 - \widehat{\Delta_{RGB}}) * \mathcal{G}_I(0, I_{width}) * \mathcal{G}_T(0.2 * T_{depth}, 2 * T_{depth})$ [5].
- $DS_{lab} = 1 - (1 - \widehat{\Delta E^*_{76}}) * \mathcal{G}_I(0, I_{width}) * \mathcal{G}_T(0.2 * T_{depth}, 2 * T_{depth})$ [5].

Table 3.1 shows the average IoU and node quality of trees built on the spatial alignment graph weighted with different dissimilarity measure. Some example of bfs and bfn

5. We denote"^" for normalization to [0,1] range

Table 3.1 – Quality of nodes on different α-trees (factor by distance on image space and on tree)

Dissimilarity function	Lambda	Average IoU	Average Quality q
DS_{height}	2000	66,29%	79,38%
	3000	67,71%	82,63%
Δ_{RGB}	2000	70,29%	84,16%
	3000	70,26%	85,75%
ΔE_{94}	2000	71,18%	**87,74%**
	3000	**71,27%**	86.48%

Figure 3.20 – Comparison between nodes on different alpha trees and the best fitting subsets of the ToS. From left to right: Ground Truth, Image labeled by MToS's node, best fitting subsets of the MToS, best fitting nodes for tree building using DS_{height}, Δ_{RGB} and ΔE_{94}. The number of MToS level lines from top to bottom: 4451, 447, 101, 740.

are shown in Figure 3.19. We observe that the color dissimilarity on L*a*b* DS_{lab} gives the best performance. The simple Euclidian distance of color in RGB space is overall just a slightly behind, but that measure could be computed more efficiently. The height dissimilarity, which is the simplest and had high performance on the ToSoL, falls behind. The reason is that in case the ToS could not provide a good segmentation of the image, the background could be broken into multiple regions which have the same height as the object. On the other hand, the objects could also be broken into smaller pieces. Therefore the height similarity would introduce more false positives. Even in the best cases (with ΔE_{94}), the segmentation is still far from producing a perfect one. The segmentation is readable for human, but it still poses challenges for OCR. For example, the first row of Figure 3.20 contains broken characters or the third row contains multiple characters which are connected by a long line.

3.4.5 Selecting a set of candidate

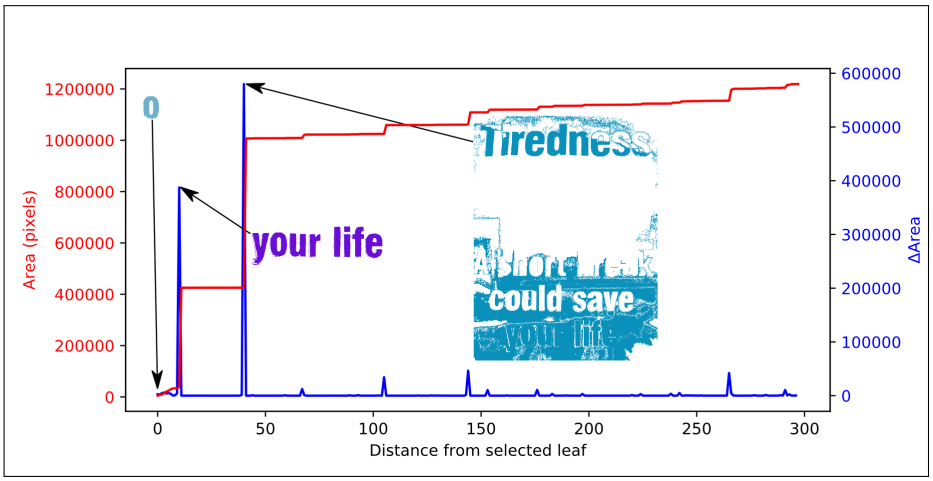
Although nodes on the second tree could reflect desired groups of objects on the images, we still have to select them from a rather large set of nodes. In this part, we consider one approach to extract some notable nodes from that set. With the advancement of machine learning, especially deep learning methods, we observe that state-of-the-art text detection methods usually use strong classification tools to separate text from non-text connected components. We could take the same approach to verify every node on the second tree and select the best text groups. In our opinion, this approach is overkill when being used to refine the text candidate set. The final objective of text recognition always requires an OCR step, which should be capable of telling text from non-text. For that reason, in this section, we only consider simple approaches to extract a small number of meaningful nodes on the α-tree as text candidates.

We could do this without using new hypotheses about objects of interest. For example, we could detect sudden changes in the node size (e.g., number of shapes, area) along the path from leaves to root. Higher changes indicate the upper node is much more different than the lower one. To do that, we weight each nodes by the increasing rate: $IR_{\mathcal{A}}(N) = \frac{\mathcal{A}(par(N)-\mathcal{A}(N)}{\mathcal{A}(N)}$ with $\mathcal{A}$ the attributes in analysis. We choose the increasing rate rather than the difference to make this attribute less scale dependence. Let us look at the second ("your life") and third (a large portion of image) marked nodes in Figure 3.21(a) and 3.21(b) which we will refer respectively as node A and B. We see that node A and B have similar area difference (compare to other nodes), but the increasing rate shows that the difference between A and $parent(A)$ is significantly more noticeable than between B and $parent(B)$. We could obtain our set of text candidate by choosing nodes whose $IR_{\mathcal{A}}$ passes a threshold or by choosing a fix-size set of strongest nodes.

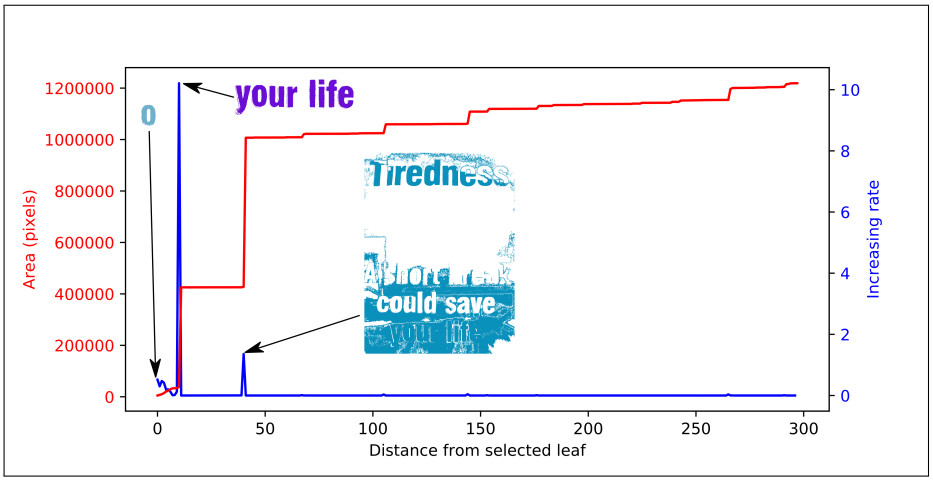
We could also use other a priori assumption about text characters that have not been used to construct the tree in the first place. We propose a simple filter based on the idea of Soille's (α, ω)-CC [94]. The idea is to weight each α-CC by a scalar value $\mathcal{A}_x$ that represent the range of an attribute x within that CC. $\mathcal{A}_x$ globally represents the variation of x in each component. The attribute range could be effectively computed during the construction of α-tree if needed. For the text characters detection application, according to our assumption about the semantic text, we propose using shapes' height h. Each node on the α-tree will be weighted by the Maximum over Minimum of height's range:

$$MoM_h(N \in TT) = \frac{Max_{n \in N}(h(n))}{Min_{n \in N}(h(n))}$$

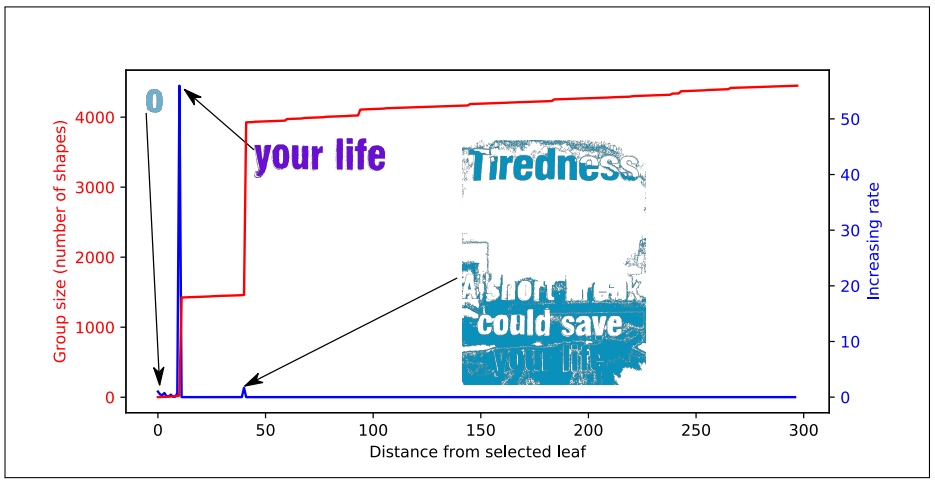
The Maximum over Minimum MoM_h is increasing in the α-tree because a higher node is a superset of the lower one. We will select the largest nodes that satisfy the threshold. In other words, they are the highest nodes on tree that validate the ranges conditions. This approach is related to the concept of Soille's (ω)-CC [94]. Therefore, we will denote them similarly. Selected nodes at a given threshold ω are denoted:



(a) Area and ΔArea



(b) Area and IR_{Area}



(c) Group size and IR_{size}

Figure 3.21 – Evolution of area and group size, area difference and increasing rate along a path from a leaf (representing letter "o" in "your life") to root. The MTOSs are simplified by minimizing the Mumford-Shah cartoon model [116] with a low simplification parameter ($\lambda = 2000$)

Figure 3.22 – Different approaches to select candidates. Left to right: ToS Labeling Map, 5% highest IR_{area}, 5% highest $IR_{Nodessize}$, $(MoM_h < 5)$-CCs. Top to bottom: Image 1 (greate difference in illumination) and 5 (text at difference scale) from ICDARRR: Forcus Scene Text and Image 5 (zoomed, small text, incidentally captured in photo) in ICDARRR: Incidental Scene Text. In addition to these criteria, we also filter out nodes that represent an area larger than 30%of the image or have less than 3 CC.

$$(MoM_h \leq \omega)\text{-}CC = max\{\alpha_i\text{-}CC | MoM_h(\alpha_i\text{-}CC < \omega)\}$$

Compare to Soille's original approach, there are two main differences. First, the attribute $\mathcal{A}'$ whose global range we keep track of may not be the same as the attribute $\mathcal{A}$ that the α-connectivity is defined. A total order on $\mathcal{A}$'s value space is needed to define a range. However, this may not be the case in our α-tree. For example, we still can define a local dissimilarity based on the RGB color space. However, there is no explicit ordering on that space. Moreover, as we argued earlier, the global parameter does not prevent the leakage from happening but to signal when that happens. We could use other a priori assumption about the desired object for this purpose, which is height in our case. Second, the second attribute does not need to be a range length function. In our case, a small height difference is noticeable for components at lower scales but could be neglected at higher ones. We find MoM more interesting due to its invariant to scale.

Examples of these two approaches are presented in Figure 3.22.

3.5 Conclusion and Perpestive

In this chapter, we continue working on semantic objects regrouping by using the background-object relationship imposed by inclusion relation, as well as the spatial arrangement of nodes on a hierarchical image representation. There are two main contributions: first, an expansion of shape-spaces framework; and second, a spatial arrangement graph with respect to inclusion for unconnected component grouping and application of that graph in text characters grouping.

Our expansion on the framework of connected filter on shape-spaces [115] is twofold. First, We expand the definition of the shape-spaces to encompass spaces that are represented by a graph of shapes, whose vertices are defined by the nodes set of the tree-based image representation, and their edges are defined with any relationship between these nodes, which may not coincide with the parent-children one. Second, we take a different approach than Xu et al. on applying the connected filter on the shape-spaces. We propose the reconstruction of the shape-spaces through its associated function. This approach allows us to performed non-pruning strategy on the second tree while still fits into the connected filters frameworks.

We apply that framework for text characters detection in natural images. First, we propose the spatial arrangement graph with respect to inclusion. That graph embeds both these types of information into a single structure, and is useful for semantic component grouping. Second, we apply that graph as a method to group similar components, which are likely to be text characters. We start with a simplified MToS that represent the original image and construct the spatial alignment graph. On that graph, neighbors of a “shape”, i.e., node on the first tree, are those that are not its background and spatially aligned. By constructing an α-tree with customized dissimilarity function on that graph, we could represent groups of semantically related text characters by nodes on that tree. We have proven these nodes have good quality in comparison with the best possible group of started segmentation provided by the ToS. We also propose some approaches to select essential groups from the α-tree for later processing.

As a perspective, three different research directions are of interest. First, we would like to study different approaches to construct a graph with both inclusion and spatial relationship, as well as different ways to construct the second tree. Second, we will integrate our approach into an end-to-end text detection and recognition system. Finally, we also could study the image simplification aspect of our framework.

To begin, several aspects of our works need more study. As discussed in Section 3.3.1, although the hierarchical of segmentation such as α-tree and especially BPT is more adaptable to obtain objects of interest, the inclusion relationship is not natively encoded. We consider the possibility of imposing the inclusion relationship on these type of tree to adapt them to our framework. On the other hand, we are also considering other approaches to construct the second tree. Instead of presenting the difference between regions’ multivariant attributes by a scalar dissimilarity function, we could consider other approaches

in extending hierarchical image representation to multivariant images, e.g., [16, 94]. A learned dissimilarity function that adapts to each application is also an interesting perspective.

Next, we intend to integrate our text detection and segmentation approach in a text detection pipeline for a complete end-to-end method. A text rectification step (to correct broken texts) and OCR step is needed for an end-to-end evaluation.

Finally, we intend to study the image simplification aspect of this framework. As mentioned in Section 3.2.3, it is possible to reconstruct the filtered image from the second tree. Because nodes on the second tree (e.g., the α-tree on a spatial alignment graph) are usually groups of non-connected components, the simplified image may not be simplified in term of number level lines but the value space.

Chapter 4

Conclusion and Perspectives

This thesis has been devoted to the study of inclusion and other types of spatial information on hierarchical image representations. The inclusion relationship of regions in an image is interesting because it allows us to deduce the objects-background relationship between image regions, which carries contextual information. Along with other spatial arrangement relationship: adjacency, in the sense of "nearby" or more general the alignment of image regions, they could benefit many context-based applications. Hierarchical image representations, on the other hand, are great devices that allow us to render the intrinsic multiscale nature of images in a simple, understandable structure.

The objective of this thesis has been the exploitation of inclusion and other spatial information in hierarchical image representations, order to better analyze the image contextually in a multiscale manner. In that context, we had chosen to integrate spatial information into a hierarchical image representation that encodes the inclusion relationship, which yields interesting results. The main results of the works presented in this dissertation are:

- A variant of ToS, the Tree of Shapes of Laplacian sign (ToSoL) which encodes the inclusion of 0-crossing of a Morphological Laplacian map. Although we rely on the "classical" 0-crossings of the Laplace operator image to obtain objects of interest, our version is innovative for several reasons. First, we rely on the morphological Laplace operator, which performs well in the case of uneven illumination, which is a common problem in natural images. Second, we ensure that the "0-crossings" are real 1D objects. We do that by using a well-composed Laplacian image and considering the 1D topological boundary of shapes. As a consequence, the positive and negative regions can be organized into a tree without topological ambiguity. Last we present a linear time complexity algorithm to compute the hierarchical structure. We also ensure that our algorithm allows us to, directly during the computation process, ignore some regions if they are not relevant. We also propose an optimization that mimics well-composedness to avoids the real interpolation process.

- A simple grouping process on the ToSoL which take advantage of both inclusion and adjacency. Thanks to the structure of the ToSoL, relevant regions are sibling on the tree structure. Combine with adjacency obtained by a simple spatial search, our method has a good balance between efficiency (linear time complexity) and quality (with a competitive F-score).

- An expansion of the shape-space morphology [115]. Our expansion has two primary results: 1) It allows the manipulation of any graph of shapes, which encodes different information. 2) It allows any tree filtering strategy proposed by the connected operator frameworks.
- A graph, representing a shape-space, that is construct from both inclusion and spatial arrangement, called spatial alignment graph w.r.t inclusion. Neighbors of a node are regions that are not its background (regions in which it is included) and spatially aligned (regions that are in the desired direction in the image space).
- Application of the generalized shape-space morphology and spatial alignment graph w.r.t inclusion in text detection. By constructing an α-tree with a dissimilarity measure adapt to the application, we could obtain groups of semantic text characters with high quality. We also propose some approaches to obtain a small set of candidates for the later stages of end-to-end text detection and recognition pipeline.

Extension And Future Studies

Improvement of the spatial graph w.r.t inclusion

Choice of the first tree. The ToS may not produce a proper segmentation in some case, in particular in uneven illumination, which results in "broken" objects, i.e., the object is represented by a set of unconnected regions. Although we could regroup these broken parts, one may want to start with a better segmentation. In this case, we may choose with a different tree, e.g., a BPT adapts to highlight objects of interest, and impose the inclusion relationship to that tree.

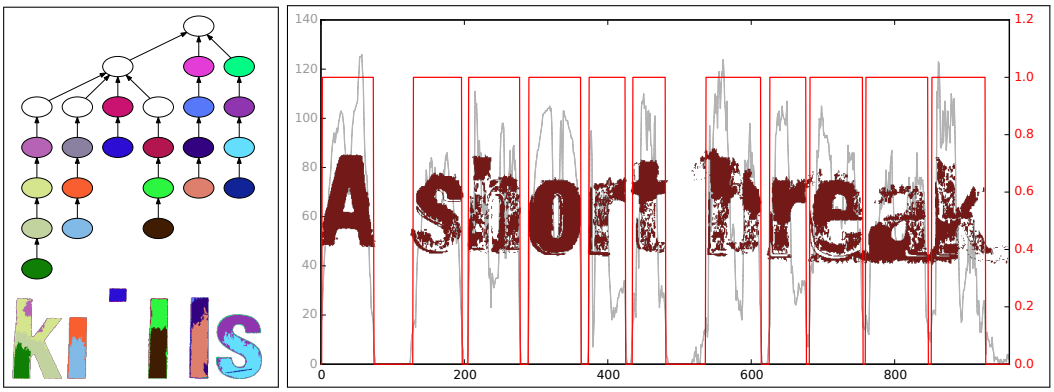
Construction of the second tree. Many interesting attributes of elements of the shape-space is multivariate, e.g., color. Moreover, we also have the distance on the tree and the distance on the image to consider. As a results, our shape-space is multivariate. In this study, we took a simple approach that combines these features into a simple linear dissimilarity function to construct the α-tree. There are other approaches to this problem:

There are several extension of hierarchical image presentation for multichannel image [79, 94, 16], which could be adapted to our graph.

On the other hand, one considers a "learned" feature that adapts to the application, so that desire objects are grouped on the second tree.

Application aspect of our work

Continuation of the text detection application. Because our method only produces text candidates, integration to an end-to-end text detection and recognition is needed. Depend on the ability of the OCR step, we could separate the text group into individual characters easily (by separate a group based on the subtrees they belong to, or by analysis the distribution of that group, e.g., as in Figure 4.1).



(a) (b)

Figure 4.1 – Separation of a text group into individual characters. These text group in Figure 3.8(f) could be separate in different way. Most of the time, we face the case of "kills" where characters are in disconnected subtrees. On the other hand, for "broken" characters, we may rely on the distribution of the text group for separation. For example, in 4.1(b), we could use the red function to separate the text group. (The gray function is the histogram of points in each column. We threshold $hist$ and then filter out small gain (less than 5-pixel width) to obtain the red function.)

Other applicative aspects of this work, in particular, filtering/image simplification. Because the shape-spaces framework allows the reconstruction of images from the filtered second tree, we could use this work for image simplification purpose. The reconstituted image may not be simplified in the number of flat-zones but the value space. We could also follow the node removal approach [115] to remove similar repetitive regions. These regions, especially at a lower scale, sometimes may not be the subject of the application and should be filtered out.

Appendices

A Ancestor Relationship and Lowest Common Ancestor in Pylene

A.1 Olena and Pylene

Our works were implemented with the help of Olena and its modernized version, Pylene.

Olena is a platform devoted to image processing and pattern recognition. Its core component is Milena, a generic and efficient C++ library. Milena provides a framework to implement simple, fast, safe, reusable and extensible image processing toolchains. The library provides many ready-to-use image data structures (regular 1D, 2D, 3D images, graph-based images, etc.) and algorithms. Milena's algorithms are built upon classical entities from the image processing field (images, points/sites, domains, neighborhoods, etc.). This design allows image processing developers and practitioners to easily understand, modify, develop and extend new algorithms while retaining the core traits of Milena: genericity and efficiency.

Pylene is a fork of Milena (http://www.lrde.epita.fr/olena), an image processing library targeting genericity and efficiency. Pylene is a modernized version of Milena with the following objectives:

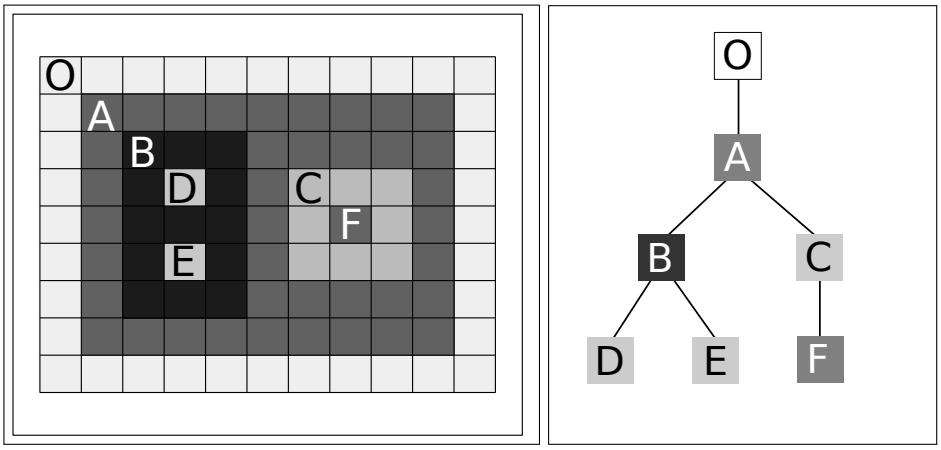
- Simplicity: both python bindings and simple C++ syntax
- Efficiency: algorithms are written in a simple way and could be run as if they were written in C. They follow one guideline: zero-cost abstraction.
- Genericity: algorithms are able to run on any kind of images with, yet, zero-cost abstraction.
- interoperability: algorithms in Pylene run on images coming from external libraries.

The component tree structure

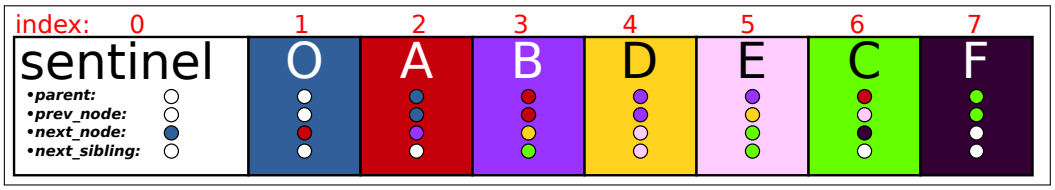
In pylene, component tree is a data structure that encodes any morphological tree e.g. mintree, maxtree, tree of shapes, α-tree, binary partition tree. They states the inclusion of connected components.

This structure is basically a triplet (N, S, pmap) where:

- N is a vector of nodes. Index of a node in this vector is its id.
- S is a vector of points.



(a) A simple image (b) The ToS



(c) The N vector

Figure A.1 – Example of the component tree structure.

- pmap is mapping $S \rightarrow N$. It is the labeling map show the deepest node each point belong to. In case of a hierarchy of segmentation, this map shows the finnest partition. In case of a hierarchy based on the threshold decomposition, it shows the proper pixels of each node.

A node is a quintet (parent, prev_node, next_node,next_sibling,first_point)

- $parent$: Index (in N) of the parent node.
- $prev\_node$: Index (in N) of the previous node.
- $next\_node$: Index (in N) of next node.
- $next\_sibling$: Index (in N) of the next sibling in the subtree. If there is not any sibling, this field is set with the sibling of its parent.
- $first\_point$: The first point index (in S) of the component

N and S are supposed to be ordered by depth-first search, but it might be too strong for some application and requires an extra step to produce this ordering, thus the structure has a tag, that tells if the ordering as been computed. The vector N has an extra sentinel node at the end to ease traversal processes. This sentinel is at index 0 and is composed by the quintet (parent: 0, prev_node: root, next: 0, next_sibling: 0, first_point: S.size()). An example is given in Figure A.1.

A.2 Ancestor relationship and Lowest Common Ancestor in Pylene

In our method, the ancestor relationship is required in multiple occassion, for example during the spatial search on the labeling map to construct the spatial alignment graph, we have to ignore the background regions which are represented by ancestor nodes on the first tree (see Section 3.3). The Lowest Common Ancestor is require to compute the distance on tree, which is a requirement in our approaches.

Thanks to the component tree structure, we could verifier the is-ancestor-of relationship with constant time complexity. Because of the order of N, ID of a node n must be in between the id its ancestors and their next siblings, except next sibling is the sentinel. For that reason, we could verify this relationship with constant time complexity. For example, in Figure A.1, we know that F is a decendant of A because $id(F) < id(A) and A.next\_sibling = sentinel$. On the other hand, F is not an decendant of B because $id(F) \notin (id(B), id(B.next\_sibling = C))$. Because the LCA is needed when we already have a depth map of the tree (to compute the distance on tree), the LCA is obtained by climbing from the nearest node from root until we reach the other node its ancestor. This approach take $O(h)$ time with h is the height of the tree. The worst case is when the every node in the tree have only one child except the root have two, thus h = floor(N/2)+1. A pseudo code for these two query using the component tree structure is presented in Algorithm 4. This query could be done in constant time, for example by reducing the LCA problem into a Range Minimum Query problem [25].

```
1  Function isAncestor(node, Ancestor):
2  |   return node> Ancestor and (node < N[Ancestor].next_sibling or
   |   N[Ancestor].next_sibling ==0);
3  Function LCA(node1, node2):
5  |   if depth(node1)<depth(node2) then
7  |   |   swap(node1,node2);
9  |   while not isAncestor (node1, node2) do
10 |   |   node2 = N[node].parent;
11 |   return node2;
```

Algorithm 4: Ancestor verification and Lowest Common Ancestor

Bibliography

[1] David H. Alman. “CIE technical committee 1–29, industrial color-difference evaluation progress report”. In: *Color Research & Application* 18.*2* (Apr. 1993), pp. 137–139 (cit. on pp. 68, 75).

[2] E. Aptoula and S. Lefèvre. “A comparative study on multivariate mathematical morphology”. In: *Pattern Recognition* 40.*11* (Nov. 2007), pp. 2914–2929 (cit. on pp. 10, 23).

[3] Galia Avidan et al. “Contrast Sensitivity in Human Visual Areas and Its Relationship to Object Recognition”. In: *Journal of Neurophysiology* 87.*6* (June 2002), pp. 3102–3116 (cit. on p. 49).

[4] C. F. Bennstrom and J. R. Casas. “Binary-partition-tree creation using a quasi-inclusion criterion”. In: *Proceedings. Eighth International Conference on Information Visualisation, 2004. IV 2004.* July 2004, pp. 259–264 (cit. on p. 16).

[5] Ilya Blayvas, Alfred Bruckstein, and Ron Kimmel. “Efficient Computation of Adaptive Threshold Surfaces for Image Binarization”. In: *Pattern Recognition* 39.*1* (2006), pp. 89–101 (cit. on p. 39).

[6] Otakar Borůvka. “O jisíém problému minimálním (About a certain minimal problem)”. In: *Práce moravské přírodovědecké společnosti* (1926), p. 24 (cit. on p. 14).

[7] Petra Bosilj, Ewa Kijak, and Sébastien Lefèvre. “Partition and Inclusion Hierarchies of Images: A Comprehensive Survey”. In: *Journal of Imaging* 4.*2* (Feb. 1, 2018), p. 33 (cit. on pp. 19, 21).

[8] Nicolas Boutry, Thierry Géraud, and Laurent Najman. “How to Make nD Functions Digitally Well-Composed in a Self-dual Way”. In: *Mathematical Morphology and Its Applications to Signal and Image Processing*. Ed. by Jón Atli Benediktsson et al. Vol. 9082. Cham: Springer International Publishing, 2015, pp. 561–572 (cit. on p. 44).

[9] Edmond J. Breen and Ronald Jones. “Attribute Openings, Thinnings, and Granulometries”. In: *Computer Vision and Image Understanding* 64.*3* (Nov. 1996), pp. 377–389 (cit. on pp. 25, 27, 28).

[10] J. Burie et al. “ICDAR2015 competition on smartphone document capture and OCR (SmartDoc)”. In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. Aug. 2015, pp. 1161–1165 (cit. on p. 53).

[11] Wenli Cai et al. “Structure-analysis method for electronic cleansing in cathartic and noncathartic CT colonography”. In: *Medical Physics* 35.*7* (July 2008), pp. 3259–3277. JSTOR: {PMC}2809717 (cit. on p. 10).

[12] Stefania Calarasanu, Jonathan Fabrizio, and Severine Dubuisson. “Using histogram representation and Earth Mover’s Distance as an evaluation tool for text detection”. In: IEEE, Aug. 2015, pp. 221–225 (cit. on p. 52).

[13] Stefania Calarasanu, Jonathan Fabrizio, and Severine Dubuisson. “What is a good evaluation protocol for text localization systems? Concerns, arguments, comparisons and solutions”. In: *Image and Vision Computing* 46 (Feb. 2016), pp. 1–17 (cit. on p. 51).

[14] Stefania Calarasanu, Jonathan Fabrizio, and Séverine Dubuisson. “From Text Detection to Text Segmentation: A Unified Evaluation Scheme”. In: *Computer Vision – ECCV 2016 Workshops*. European Conference on Computer Vision. Ed. by Gang Hua and Hervé Jégou. Lecture Notes in Computer Science. Springer International Publishing, Oct. 8, 2016, pp. 378–394 (cit. on p. 51).

[15] E. Carlinet and T. Géraud. “A Comparative Review of Component Tree Computation Algorithms”. In: *IEEE Transactions on Image Processing* 23.*9* (2014), pp. 3885–3895 (cit. on p. 20).

[16] Edwin Carlinet and Thierry Geraud. “MToS: A Tree of Shapes for Multivariate Images”. In: *IEEE Transactions on Image Processing* 24.*12* (Dec. 2015), pp. 5330–5342 (cit. on pp. 3, 23, 24, 63, 66–69, 73, 79, 89, 92).

[17] Vicent Caselles, Bartomeu Coll, and Jean-Michel Morel. “Topographic Maps and Local Contrast Changes in Natural Images”. In: *Int. J. Comput. Vision* 33.*1* (Sept. 1999), pp. 5–27 (cit. on pp. 2, 20).

[18] Vicent Caselles and Pascal Monasse. *Geometric description of images as topographic maps*. Lecture notes in mathematics 1984. Heidelberg ; London ; New York: Springer, 2010. 186 pp. (cit. on p. 21).

[19] Adam Coates et al. “Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning”. In: *Proceedings of the 2011 International Conference on Document Analysis and Recognition*. ICDAR ’11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 440–445 (cit. on p. 34).

[20] Jean Cousty et al. “Watershed cuts: Minimum spanning forests and the drop of water principle”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.*8* (2009), pp. 1362–1374 (cit. on p. 14).

[21] Boris Epshtein, Eyal Ofek, and Yonatan Wexler. “Detecting text in natural scenes with stroke width transform”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2963–2970 (cit. on pp. 33–35, 51).

[22] Jonathan Fabrizio et al. “TextCatcher: A Method to Detect Curved and Challenging Text in Natural Scenes”. In: *International Journal on Document Analysis and Recognition* (2016), pp. 1–19 (cit. on p. 51).

[23] Yi-Feng Pan, Xinwen Hou, and Cheng-Lin Liu. “A Hybrid Approach to Detect and Localize Texts in Natural Scene Images”. In: *IEEE Transactions on Image Processing* 20.*3* (Mar. 2011), pp. 800–813 (cit. on p. 34).

[24] R. A. finkel and J. L. Bentley. “Quad trees a data structure for retrieval on composite keys”. In: *Acta Informatica* 4.*1* (1974), pp. 1–9 (cit. on p. 1).

[25] Johannes fischer and Volker Heun. “A New Succinct Representation of RMQ-Information and Improvements in the Enhanced Suffix Array”. In: *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Vol. 4614. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 459–470 (cit. on p. 97).

[26] C. Garcia and X. Apostolidis. “Text detection and segmentation in complex color images”. In: *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 4. IEEE, 2000, pp. 2326–2329 (cit. on p. 35).

[27] D. Gatica-Perez et al. “Extensive partition operators, gray-level connected operators, and region merging/classification segmentation algorithms: theoretical links”. In: *IEEE Transactions on Image Processing* 10.*9* (Sept. 2001), pp. 1332–1345 (cit. on p. 26).

[28] Thierry Géraud, Edwin Carlinet, and S. Crozet. “Self-Duality and Discrete Topology”. In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Vol. 9082. LNCS. Springer, 2015, pp. 573–584 (cit. on p. 44).

[29] Thierry Geraud et al. “A Quasi-Linear Algorithm to Compute the Tree of Shapes of n-D Images.”. In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Vol. 7883. LNCS. Springer, 2013, pp. 98–110 (cit. on pp. 21, 42, 44, 63).

[30] J. C. Gower and G. J. S. Ross. “Minimum Spanning Trees and Single Linkage Cluster Analysis”. In: *Applied Statistics* 18.*1* (1969), p. 54 (cit. on p. 18).

[31] S. Hambrusch, X. He, and R. Miller. “Parallel Algorithms for Gray-Scale Digitized Picture Component Labeling on a Mesh-Connected Computer”. In: *Journal of Parallel and Distributed Computing* 20 (1994), pp. 56–68 (cit. on pp. 18, 80).

[32] Shehzad Muhammad Hanif, Lionel Prevost, and Pablo Augusto Negri. “A cascade detector for text detection in natural scene images”. In: *Pattern Recognition (ICPR), 2008 19th International Conference on*. IEEE, Dec. 2008, pp. 1–4 (cit. on p. 35).

[33] He ZhiWei et al. “A new automatic extraction method of container identity codes”. In: *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*. Vol. 2. IEEE, 2003, pp. 1688–1691 (cit. on p. 32).

[34] Kaiming He et al. “Mask R-CNN”. In: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870 (cit. on p. 34).

[35] Xiaodong Huang and Huadong Ma. “Automatic Detection and Localization of Natural Scene Text in Video”. In: *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, Aug. 2010, pp. 3216–3219 (cit. on p. 36).

[36] Lê Duy Huỳnh, Yongchao Xu, and Thierry Géraud. “Morphology-Based Hierarchical Representation with Application to Text Segmentation in Natural Images”. In: *Proceedings of the 23st International Conference on Pattern Recognition (ICPR)*. Cancún, México: IEEE Computer Society, Dec. 2016 (cit. on pp. 66, 68).

[37] Anil K. Jain and Bin Yu. “AUTOMATIC TEXT LOCATION IN IMAGES AND VIDEO FRAMES”. In: *Pattern Recognition* 31.*12* (Dec. 1998), pp. 2055–2076 (cit. on p. 35).

[38] Vojtěch Jarník. “O jistém problému minimálním (About a certain minimal problem)”. In: *Práce Moravské Přírodovědecké Společnosti* 6.*4* (1930), pp. 57–63 (cit. on p. 14).

[39] Jean C. Serra and Philippe Salembier. “Connected operators and pyramids”. In: vol. 2030. 1993, pp. 2030–12 (cit. on pp. 25, 26).

[40] Ronald Jones. “Component Trees for Image filtering and Segmentation”. In: *Proceedings of the 1997 IEEE Workshop on Nonlinear Signal and Image Processing*. Mackinac Island, 1997, p. 5 (cit. on pp. 1, 3, 20).

[41] Ronald Jones. “Connected filtering and Segmentation Using Component Trees”. In: *Computer Vision and Image Understanding* 75.*3* (1999), pp. 215–228 (cit. on pp. 1, 20, 25, 26).

[42] Kai Wang, Boris Babenko, and Serge Belongie. “End-to-end scene text recognition”. In: *international conference on computer vision*. IEEE, Nov. 2011, pp. 1457–1464 (cit. on p. 34).

[43] D. Karatzas and A. Antonacopoulos. “Text extraction from Web images based on a split-and-merge segmentation method using colour perception”. In: *Pattern Recognition (ICPR), International Conference on*. IEEE, 2004, 634–637 Vol.2 (cit. on p. 35).

[44] Dimosthenis Karatzas et al. “ICDAR 2013 Robust Reading Competition”. In: *2013 12th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, Aug. 2013, pp. 1484–1493 (cit. on pp. 34, 79).

[45] D. Karatzas et al. “ICDAR 2011 Robust Reading Competition - Challenge 1: Reading Text in Born-Digital Images (Web and Email)”. In: *2011 11th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, Sept. 2011, pp. 1485–1490 (cit. on p. 31).

[46] Kongqiao Wang, J.A. Kangas, and Wenwen Li. “Character segmentation of color images from digital camera”. In: *Proceedings of Sixth International Conference on Document Analysis and Recognition*. IEEE Comput. Soc, 2001, pp. 210–214 (cit. on p. 35).

[47] J.B. Kruskal. “On the shortest spanning subtree of a graph and the traveling salesman problem”. In: *Proceedings of the American Mathematical Society* 7 (1956), pp. 48–50 (cit. on p. 14).

[48] Jung-Jin Lee et al. “AdaBoost for Text Detection in Natural Scene”. In: *2011 International Conference on Document Analysis and Recognition*. IEEE, Sept. 2011, pp. 429–434 (cit. on p. 34).

[49] SeongHun Lee et al. “Scene Text Extraction with Edge Constraint and Text Collinearity”. In: *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, Aug. 2010, pp. 3983–3986 (cit. on pp. 34, 35).

[50] Jian Liang, David Doermann, and Huiping Li. “Camera-based analysis of text and documents: a survey”. In: *International Journal of Document Analysis and Recognition (IJDAR)* 7.*2* (July 2005), pp. 84–104 (cit. on pp. 32, 33, 35).

[51] Jie Liu et al. “A Novel Italic Detection and Rectification Method for Chinese Advertising Images”. In: *2011 11th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, Sept. 2011, pp. 698–702 (cit. on p. 33).

[52] Y. Liu. “A Contour-Based Robust Algorithm for Text Detection in Color Images”. In: *IEICE Transactions on Information and Systems* E89-D.*3* (Mar. 1, 2006), pp. 1221–1230 (cit. on pp. 33, 34).

[53] Zhi Liu, Liquan Shen, and Zhaoyang Zhang. “Unsupervised image segmentation based on analysis of binary partition tree for salient object extraction”. In: *Signal Processing* 91.*2* (Feb. 2011), pp. 290–299 (cit. on p. 16).

[54] Huihai Lu, J. C. Woods, and M. Ghanbari. “Binary Partition Tree for Semantic Object Extraction and Image Segmentation”. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 17.*3* (Mar. 2007), pp. 378–383 (cit. on p. 16).

[55] J. Matas et al. “Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions”. In: *Image and Vision Computing* 22.*10* (2004), pp. 761–767 (cit. on p. 51).

[56] Navid Mavaddat, Tae-Kyun Kim, and Roberto Cipolla. “Design and evaluation of features that best define text in complex scene images”. In: *Proceedings of the Eleventh IAPR Conference on Machine Vision Applications*, p. 4 (cit. on p. 35).

[57] Fernand Meyer and Petros Maragos. “Nonlinear Scale-Space Representation with Morphological Levelings”. In: *Journal of Visual Communication and Image Representation* 11.*2* (June 2000), pp. 245–265 (cit. on pp. 1, 16).

[58] Wojciech Mokrzycki and Maciej Tatol. “Color difference Delta E - A survey”. In: *Machine Graphics and Vision* 20 (Apr. 1, 2011), pp. 383–411 (cit. on p. 75).

[59] P. Monasse and F. Guichard. “Fast computation of a contrast-invariant image representation”. In: *IEEE Transactions on Image Processing* 9.*5* (May 2000), pp. 860–872 (cit. on pp. 1–3, 20, 21).

[60] Raul S Montero and Ernesto Bribiesca. “State of the Art of Compactness and Circularity Measures”. In: *In International Mathematical Forum* (), p. 31 (cit. on p. 27).

[61] U. Moschini, A. Meijster, and M. H. F. Wilkinson. “A Hybrid Shared-Memory Parallel Max-Tree Algorithm for Extreme Dynamic-Range Images”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.*3* (Mar. 2018), pp. 513–526 (cit. on p. 21).

[62] Ali Mosleh, Nizar Bouguila, and A. Ben Hamza. “Image Text Detection Using a Bandlet-Based Edge Detector and Stroke Width Transform”. In: *Proceedings of the British Machine Vision Conference*. British Machine Vision Association, 2012, pp. 63.1–63.12 (cit. on p. 36).

[63] Makoto Nagao, Takashi Matsuyama, and Yoshio Ikeda. “Region extraction and shape analysis in aerial photographs”. In: *Computer Graphics and Image Processing* 10.*3* (July 1979), pp. 195–223 (cit. on pp. 1, 16).

[64] L. Najman and M. Couprie. “Building the Component Tree in Quasi-Linear Time”. In: *IEEE Transactions on Image Processing* 15.*11* (Nov. 2006), pp. 3531–3539 (cit. on p. 21).

[65] Laurent Najman, Jean Cousty, and Benjamin Perret. “Playing with Kruskal: Algorithms for Morphological Trees in Edge-Weighted Graphs”. In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Vol. 7883. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 135–146 (cit. on pp. 14, 16, 18, 74, 78).

[66] Laurent Najman and Hugues Talbot, eds. *Mathematical morphology: from theory to applications*. London: ISTE [u.a.], 2010. 507 pp. (cit. on pp. 1, 39).

[67] Lukáš Neumann and Jiří Matas. “Real-Time Lexicon-Free Scene Text Localization and Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* () (cit. on p. 51).

[68] Lukáš Neumann and Jiří Matas. “Real-time scene text localization and recognition”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3538–3545 (cit. on pp. 33, 35, 36).

[69] Lukáš Neumann and Jiří Matas. “Text Localization in Real-World Images Using Efficiently Pruned Exhaustive Search”. In: *2011 International Conference on Document Analysis and Recognition*. IEEE, Sept. 2011, pp. 687–691 (cit. on pp. 34, 35).

[70] N Nikolaou and N Papamarkos. “Color Reduction for Complex Document Images”. In: *International Journal on Imaging and System and Technology* 19.*1* (Mar. 2009), pp. 14–26 (cit. on p. 35).

[71] David Nistér and Henrik Stewénius. “Linear Time Maximally Stable Extremal Regions”. In: *Computer Vision – ECCV 2008*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 183–196 (cit. on p. 20).

[72] Larry O’Gorman. *Document Image Analysis: An Executive Briefing*. Ed. by Lawrence O’Gorman and Rangachar Kasturi. 1st. Los Alamitos, CA, USA: IEEE Computer Society Press, 1997 (cit. on p. 31).

[73] Georgios K Ouzounis. “A parallel implementation of the dual-input Max-Tree algorithm for attribute filtering”. In: *International Symposium on Mathematical Morphology (ISMM), 2007*. Vol. 1. 2007, p. 13 (cit. on p. 21).

[74] Georgios K. Ouzounis and Pierre Soille. “Pattern Spectra from Partition Pyramids and Hierarchies”. In: *Mathematical Morphology and Its Applications to Image and Signal Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 108–119 (cit. on p. 1).

[75] Yi-Feng Pan, Xinwen Hou, and Cheng-Lin Liu. “Text Localization in Natural Scene Images Based on Conditional Random field”. In: *International Conference on Document Analysis and Recognition*. IEEE, 2009, pp. 6–10 (cit. on p. 34).

[76] T. Pavlidis. “Hierarchies in structural pattern recognition”. In: *Proceedings of the IEEE* 67.*5* (May 1979), pp. 737–744 (cit. on p. 1).

[77] A. V. Pillai et al. “Detection and localization of texts from natural scene images using scale space and morphological operations”. In: IEEE, Mar. 2013, pp. 880–885 (cit. on p. 35).

[78] Qixiang Ye et al. “A robust text detection algorithm in images and video frames”. In: *Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing*. Vol. 2. IEEE, 2003, pp. 802–806 (cit. on p. 35).

[79] Jimmy Francky Randrianasoa et al. “Binary Partition Tree construction from multiple features for image segmentation”. In: *Pattern Recognition* 84 (2018), pp. 237–250 (cit. on p. 92).

[80] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497 (cit. on p. 34).

[81] P. Salembier and L. Garrido. “Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval”. In: *IEEE Transactions on Image Processing* 9.*4* (Apr. 2000), pp. 561–576 (cit. on pp. 1, 3, 15, 16, 26).

[82] P. Salembier, A. Oliveras, and L. Garrido. “Antiextensive connected operators for image and sequence processing”. In: *IEEE Transactions on Image Processing* 7.*4* (Apr. 1998), pp. 555–570 (cit. on pp. 1, 20, 25, 26, 28, 30).

[83] P. Salembier and J. Serra. “Flat zones filtering, connected operators, and filters by reconstruction”. In: *IEEE Transactions on Image Processing* 4.*8* (Aug. 1995), pp. 1153–1160 (cit. on pp. 9, 20, 25, 26).

[84] Philippe Salembier and Michael Wilkinson. “Connected operators”. In: *IEEE Signal Processing Magazine* 26.*6* (Nov. 2009), pp. 136–157 (cit. on pp. 3, 25, 26).

[85] Jean Serra. “A Lattice Approach to Image Segmentation”. In: *Journal of Mathematical Imaging and Vision* 24.*1* (Jan. 2006), pp. 83–130 (cit. on pp. 11, 26).

[86] Jean Serra. *Image Analysis and Mathematical Morphology*. Orlando, FL, USA: Academic Press, Inc., 1983 (cit. on p. 1).

[87] Gaurav Sharma. *Digital Color Imaging Handbook*. Boca Raton, FL, USA: CRC Press, Inc., 2002 (cit. on p. 79).

[88] Gaurav Sharma, Wencheng Wu, and Edul N. Dalal. “The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations”. In: *Color Research & Application* 30.*1* (Feb. 2005), pp. 21–30 (cit. on p. 75).

[89] Baoguang Shi, Xiang Bai, and Serge Belongie. “Detecting Oriented Text in Natural Images by Linking Segments”. In: *arXiv:1703.06520 [cs]*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Mar. 19, 2017, pp. 3482–3490 (cit. on p. 34).

[90] Cunzhao Shi et al. “Scene text detection using graph model built upon maximally stable extremal regions”. In: *Pattern Recognition Letters* 34.*2* (Jan. 2013), pp. 107–116 (cit. on p. 36).

[91] Palaiahnakote Shivakumara, Weihua Huang, and Chew Lim Tan. “Efficient video text detection using edge features”. In: *Pattern Recognition (ICPR), 2008 19th International Conference on*. IEEE, Dec. 2008, pp. 1–4 (cit. on p. 35).

[92] Jean Claude Simon. *Patterns and operators: The foundations of data representation*. McGraw-Hill, 1986 (cit. on p. 11).

[93] Ray Smith, Daria Antonova, and Dar-Shyang Lee. “Adapting the Tesseract open source OCR engine for multilingual OCR”. In: *MOCR ’09: Proceedings of the International Workshop on Multilingual OCR*. ACM Press, 2009, p. 1 (cit. on p. 33).

[94] P. Soille. “Constrained connectivity for hierarchical image partitioning and simplification”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.*7* (July 2008), pp. 1132–1145 (cit. on pp. 1, 3, 16, 18, 19, 85, 89, 92).

[95] P. Soille, ed. *Morphological Image Analysis—Principles and Applications*. 2nd. Springer-Verlag, 2004 (cit. on p. 39).

[96] Yuqing Song. “A Topdown Algorithm for Computation of Level Line Trees”. In: *IEEE Transactions on Image Processing* 16.*8* (Aug. 2007), pp. 2107–2116 (cit. on pp. 1, 21).

[97] Yuqing Song and Aidong Zhang. “Analyzing scenery images by monotonic tree”. In: *Multimedia Systems* 8.*6* (Apr. 2003), pp. 495–511 (cit. on pp. 1, 21).

[98] K. Subramanian et al. “Character-Stroke Detection for Text-Localization and Extraction”. In: *International Conference on Document Analysis and Recognition*. IEEE, Sept. 2007, pp. 33–37 (cit. on p. 35).

[99] Minsoo Suk and Ohyoung Song. “Curvilinear Feature Extraction using minimum Spanning trees”. In: *Computer Vision, Graphics, and Image Processing* 26.*3* (June 1984), pp. 400–411 (cit. on p. 14).

[100] Robert Endre Tarjan. “Efficiency of a Good But Not Linear Set Union Algorithm”. In: *Journal of the ACM* 22.*2* (Apr. 1, 1975), pp. 215–225 (cit. on p. 21).

[101] Erik R. Urbach, Jos B.T.M. Roerdink, and Michael H.F. Wilkinson. “Connected Shape-Size Pattern Spectra for Rotation and Scale-Invariant Classification of Gray-Scale Images”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.*2* (Feb. 2007), pp. 272–285 (cit. on p. 28).

[102] Silvia Valero, Philippe Salembier, and Jocelyn Chanussot. “Comparison of merging orders and pruning strategies for Binary Partition Tree in hyperspectral data”. In: *International Conference on Image Processing*. IEEE, Sept. 2010, pp. 2565–2568 (cit. on p. 16).

[103] Andreas Veit et al. “COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images”. In: *arXiv:1601.07140 [cs]* (Jan. 26, 2016) (cit. on p. 34).

[104] V. Vilaplana, F. Marques, and P. Salembier. “Binary Partition Trees for Object Detection”. In: *IEEE Transactions on Image Processing* 17.*11* (Nov. 2008), pp. 2201–2216 (cit. on pp. 1, 16).

[105] Andrew J. Viterbi and James K. Omura. *Principles of Digital Communication and Coding*. 1st. New York, NY, USA: McGraw-Hill, Inc., 1979 (cit. on p. 28).

[106] Lucas J van Vliet, Ian T Young, and Guus L Beckers. “A nonlinear laplace operator as edge detector in noisy images”. In: *Computer Vision, Graphics, and Image Processing* 45.*2* (Feb. 1, 1989), pp. 167–195 (cit. on p. 38).

[107] Lucas J van Vliet, Ian T Young, and Guus L Beckers. “An Edge Detection Model Based on Non-Linear Laplace filtering”. In: *Proceedings of International Workshop on pattern recognition and artificial intelligence*. 1988, pp. 63–73 (cit. on p. 39).

[108] Tao Wang et al. “End-To-End Text Recognition with Convolutional Neural Networks”. In: *Pattern Recognition (ICPR), 2012 21st International Conference on*. Nov. 2012, p. 5 (cit. on p. 34).

[109] Yang Wang and Prabir Bhattacharya. “On parameter-dependent connected components of gray images”. In: *Pattern Recognition* 29.*8* (1996), pp. 1359–1368 (cit. on pp. 1, 16).

[110] Michel A. Westenberg, Jos B. T. M. Roerdink, and Michael H. F. Wilkinson. “Volumetric Attribute filtering and Interactive Visualization Using the Max-Tree Representation”. In: *IEEE Transactions on Image Processing* 16.*12* (Dec. 2007), pp. 2943–2952 (cit. on p. 27).

[111] M. H. F. Wilkinson. “A fast component-tree algorithm for high dynamic-range images and second generation connectivity”. In: *2011 18th IEEE International Conference on Image Processing*. Sept. 2011, pp. 1021–1024 (cit. on p. 20).

[112] Michael H. F. Wilkinson and Michel A. Westenberg. “Shape Preserving filament Enhancement filtering”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2001*. Vol. 2208. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 770–777 (cit. on pp. 28, 30).

[113] Xiangrong Chen and A.L. Yuille. “Detecting and reading text in natural scenes”. In: *Conference on Computer Vision and Pattern Recognition*. Vol. 2. IEEE, 2004, pp. 366–373 (cit. on pp. 33–35).

[114] Xu Zhao et al. “Text From Corners: A Novel Approach to Detect Text and Caption in Videos”. In: *IEEE Transactions on Image Processing* 20.*3* (Mar. 2011), pp. 790–799 (cit. on pp. 35, 36).

[115] Yongchao Xu, Thierry Géraud, and Laurent Najman. “Connected filtering on Tree-Based Shape-Spaces”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.*6* (June 2016), pp. 1126–1140 (cit. on pp. 3, 4, 26, 29, 30, 57, 58, 88, 92, 93).

[116] Yongchao Xu, Thierry Géraud, and Laurent Najman. “Salient level lines selection using the Mumford-Shah functional”. In: *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE, 2013, pp. 1227–1231 (cit. on pp. 67, 68, 74, 79, 86).

[117] Q. Ye and D. Doermann. “Text Detection and Recognition in Imagery: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.*7* (2015), pp. 1480–1500 (cit. on pp. 33, 35).

[118] Qixiang Ye and David Doermann. “Scene Text Detection via Integrated Discrimination of Component Appearance and Consensus”. In: *Camera-Based Document Analysis and Recognition*. Ed. by Masakazu Iwamura and Faisal Shafait. Vol. 8357. Cham: Springer International Publishing, 2014, pp. 47–59 (cit. on p. 35).

[119] Chucai Yi and Yingli Tian. “Localizing Text in Scene Images by Boundary Clustering, Stroke Segmentation, and String Fragment Classification”. In: *IEEE Transactions on Image Processing* 21.*9* (Sept. 2012), pp. 4256–4268 (cit. on p. 35).

[120] Xu-Cheng Yin et al. “Robust Text Detection in Natural Scene Images”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.5 (May 2014), pp. 970–983 (cit. on pp. 33–36).

List of Figures

List of Tables